034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number D3.8

Final Report on the achievements and results of WP3

Due date of deliverable: 2008-11-30
Actual submission date: 2008-11-30
Document code: Phosphorus-WP3-D3.8

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Final Report on the achievements and results of WP3

Abstract

The final report will summarize the results and achievements of WP3, in particular the middleware integration, the integration of the middleware layer and the network layer, and the deployment and behaviour of the applications in the optical testbed.

Project:	Phosphorus
Deliverable Number:	D3.8
Date of Issue:	30/11/2008
EC Contract No.:	034115
Document Code:	Phosphorus-WP3-D3.8

Table of Contents

Table of Figures

0 Executive Summary

The final report will summarize the results and achievements of WP3, in particular the middleware integration, the integration of the middleware layer and the network layer, and the results and achievements of deployment and behaviour of the applications in the optical testbed.

.

1 Overview

The final report will summarize the results and achievements of WP3, in particular the middleware integration, the integration of the middleware layer and the network layer, and the results and achievements of deployment and behaviour of the applications in the optical testbed.

During the period of the initial two years considered into this report the main tasks of WP3 were:

- Middleware development and adaptation (UNICORE, MetaScheduling Service (MSS), Resource Selection Service (RSS), INCA)
- Interfacing between G^2MPLS and Resource Management System via G-OUNI
- Integration and Evaluation of Middleware
- Adaptation, deployment and tests of the application suite selected for the PHOSPHORUS project

The report builds on this separation of tasks, and presents for each task reports on the results of and achievements made. Thus, section 2 presents results of adaption and developments of the UNICORE system, section 3 the results of adaption and developments of the MetaScheduling Service, in section 4 we enumerate the results of the Resource Selection Service developments, section 5 highlights the results of the development of the interface between middleware and G^2MPLS, and section 6 describes the achievements so far with the INCA middleware. Finally, in Section 7 the results of and achievements made with the four PHOSPHORUS applications are described. The entire report is summarized in section 8, where we also draw some conclusions.

1.1 Work package objectives and main tasks

The work-package objectives during the first 24 months timeframe were:

- Formulation of user requirements to contribute to the definition of the functions and services to be implemented by the overall project

- Development and provisioning of the middleware services to orchestrate requested for an application, i.e. network, compute resources, visualisation devices, etc.
- Integration with network layer developments (Control Plane and Grid-NPRS) with Grid middleware to enable high performance applications running efficiently in the test-bed
- Integration with the network layer by means of the G^2MPLS developed by workpackage 2
- Adaptation of selected Applications to showcase the benefit from the integrated test-bed environment developed in the project
- Definition of the properties that network resources (services) should have in order to integrate seamlessly into an environment with other grid services
- Enabling user and applications to automatically get access to the resources that match their requirements e.g. in terms of performance and QoS.
- Enabling users and application to plan resource usage in advance and to dynamically adapt the resource usage to the changing requirements of the application thus avoiding expensive waste of resources.
- Managing all resources required to run an application in an integrated manner with a single service based interface towards to user or the application respectively.
- Testing the provided networking functionality based on application use cases defined in the first project stage and deriving requirements for extensions and enhancements to be carried out in the second 12 month period from the test-bed experiments

1.2 Integrated Approach

The experiments conducted in the reporting period aim at presenting the successful integration of the four layers the PHOSPHORUS testbeds are built upon:

- Physical layer
- Operating systems layer
- Middleware layer
- Application layer

Details of the experiments are described in the deliverables D3.6 - Report on the Results of the Application Experiments During the Final Testbed Experiments [D3.6] and D3.7 - Report on the results of the middleware experiments during the final testbed experiment [D3.7].

The developments during the last phase of the two years also included the interface to make the physical layer additionally accessible via the G^{2}MPLS and the G.O-UNI interface between the middleware and the G^{2}MPLS. More details on G^{2}MPLS can be found in the deliverable “Preliminary Grid-GMPLS Control Plane prototype” [D2.5].

The approach for the integration between the middleware and G^{2}MPLS is described in the “Report on the results of the middleware experiments during the final testbed experiment” [D3.7].

The interface between the application layer and the middleware layer, in other words how the applications may access middleware services and through these middleware services may benefit from all layers of the PHOSPHORUS service stack is described in several deliverables with “Report on the Results of the Application Experiments During the Final Testbed Experiments” [D3.6] being the latest one.

2 UNICORE

For the PHOSPHORUS project we have chosen UNICORE as the middleware to be linked with both the WISDOM and KoDaVis applications. UNICORE provides a plug-in architecture at the client side that allows for application specific plug-ins to be included. The suitability of the plug-in concept has been evaluated for both applications. The work on middleware design started in March 2007. While UNICORE 5 was the system for productive use when PHOSOPHORUS started, WP3 decided to migrate to UNICORE 6 – the web-service based implementation of UNICORE – for the PHOSPHORUS testbed.

The first component of the UNICORE middleware to be adapted to the KoDaVis use-case was the client. An application specific plug-in was written that could be added to any UNICORE client via the plug-in architecture described above. In the following, a specific service for the UNICORE server side was developed, which allowed the creation and management of KoDaVis data server instances as well as collaborative sessions. This service was then coupled with the graphical KoDaVis client through the UNICORE client.

The MetaScheduling Service (MSS), which is another component of the middleware landscape employed in PHOSPHORUS, had to be integrated with UNICORE. Thus, it would be possible to co-allocate network and Grid resources by means of the MSS and make advance reservations of UNICORE resources through the middleware. For UNICORE, this meant to extend the interface of the target systems to expose the advance reservation capabilities of the underlying batch system. Also, as UNICORE supports a host of batch systems, the Target System Interface (TSI) needed to be adapted for the particular batch system running on the Grid resource. The TSI was first developed for the Cray XD1, which was the Grid resource dedicated to PHOSPHORUS at FZJ. The TSI had to be implemented for the combination of the Torque batch system and Maui scheduler, when the XD1 later was replaced with another machine due to a hardware failure.

The requirements from the PHOSPHORUS application use-cases concerning work-flow support were communicated to the core UNICORE development team and were implemented in the work-flow support of the 6.1 version of UNICORE. This new features are used in particular by the WISDOM application.

For training the use of UNICORE for multiple applications within the project giving all developers further insight into the middleware, a UNICORE workshop was held at FZJ on July 25th and 26th, 2007. Further information about features like clients, job submission, data distribution, and data staging was provided.

UNICORE 6 servers were initially installed on the Cray XD1 at FZJ. Clients are available on those sites participating in the KoDaVis and WISDOM experiments:

- FHG
- FZJ
- PSNC
- UoESSEX

The UNICORE 6 server landscape was extended during the course of the first 24 months. FZJ created and maintains a central UNICORE registry where all participating sites are registered. Clients can thus discover all available UNICORE sites through this central registry. The UNICORE installation, including the additional KoDaVis service, has also been used for demonstrations at SC07, SC08, ICT 2008 and - in an interim version - at the Phosphorus review on December 13th, 2007 in Poznan.

UNICORE needed to be installed a second time at FZJ due to a hardware failure. This involved a number of steps. For one, as the new resource was equipped with a different batch system, the TSI wrapper had to be adapted to this to support the advance reservation requirements in Phosphorus. Secondly, this new installation was taken as an opportunity to change from the demonstration certificates used previously to real Grid certificates issued by the German DFN network’s DFN Grid CA.

During the last period of the project the reservation interface for the Resource Selection Service was implemented.

During the entire period the testbed implementation was following the general advances in UNICORE middleware.

3 MetaScheduling Service

The MetaScheduling Service (MSS) was initially developed in the German project VIOLA. Its main purpose was the co-allocation and reservation of compute and network resources for demanding Grid applications that used more than one compute infrastructure.

Since the VIOLA testbed was a part of the PHOSPHORUS testbed from the very beginning MSS was selected for the PHOSPHORUS project for the co-allocation and reservation of compute and network resources as well. To satisfy the requirements of the applications, the different network service and control planes the MSS has to interact with in the PHOSPHORUS testbed MSS has been extended.

The MSS is composed of a set of web-services that allow to

- find the next available slot of resources, both compute and network
- negotiate the usage of an available slot and reserve it for a user
- create a service level agreement (SLA) with the responsible software entity of the resources to reserve such a negotiated slot (local resource management systems (RMS) in case of compute resources, network resource provisioning systems (NRPS) in case of network resources
- monitor the SLAs
- display the state of a reservation
- cancel a reservation

For the MetaScheduling in PHOSPHORUS two different scenarios for co-allocation and reservation are provided now:

1. The classical approach like in the VIOLA project where the MSS is aware of potential resources based on information coming from the middleware. The MSS is negotiating with all resources to be available for a job, talking to the local RMS and the NRPS.

2. The new approach of PHOSPHORUS where the G²MPLS is aware of the resources connected to the network and queries the MSS for availability information, reservations etc, using the web-services of the middleware layer.

To allow an easy integration also in other web-service based middleware environments and to ease the interoperability across different middleware systems the MSS implementation is standards-based:

- Web Service Resource Framework (WSRF from OASIS)
- Web-services Agreement (WS-Agreement from the OGF)
- Web Services Security (WS-Security from OASIS)
- Resource selection Service (OGSA-RSS from the OGF)
- Job Submission Description Language (JSDL from the OGF)

During the first months of the project the integration of MSS into the UNICORE system was enhanced, making the overall architecture of the PHOSPHORUS middleware more consistent. Figure 1 presents the resulting architecture. In contrast to the VIOLA approach where the MSS was using own adapters for the communication with the local resource management systems (RMS), the MSS in this version of the integration is already using the target system interfaces (TSI) of UNICORE to communicate with the local RMS. In order to allow also reservations of resources and the negotiation of available time-slots for the execution of an application the TSIs have been extended. For the communication with the UNICORE systems components the MSS uses the UNICORE protocol language (UPL). To avoid extensive changes in the UNICORE internal communication it was decided to use an adapter that maps the WS-Agreement interface of the MSS to the UPL interface of UNICORE. To that end, we achieved interoperability while the internal protocol of both UNICORE and MSS could remain unchanged. With this new architecture we additionally prepared the ground for the integration with other middleware systems like the Globus Toolkit 4. Moreover, the ARGON NRPS is also connected using the same adapter mechanism: the MSS WS-Agreement protocol is mapped to the ARGON web service interface.

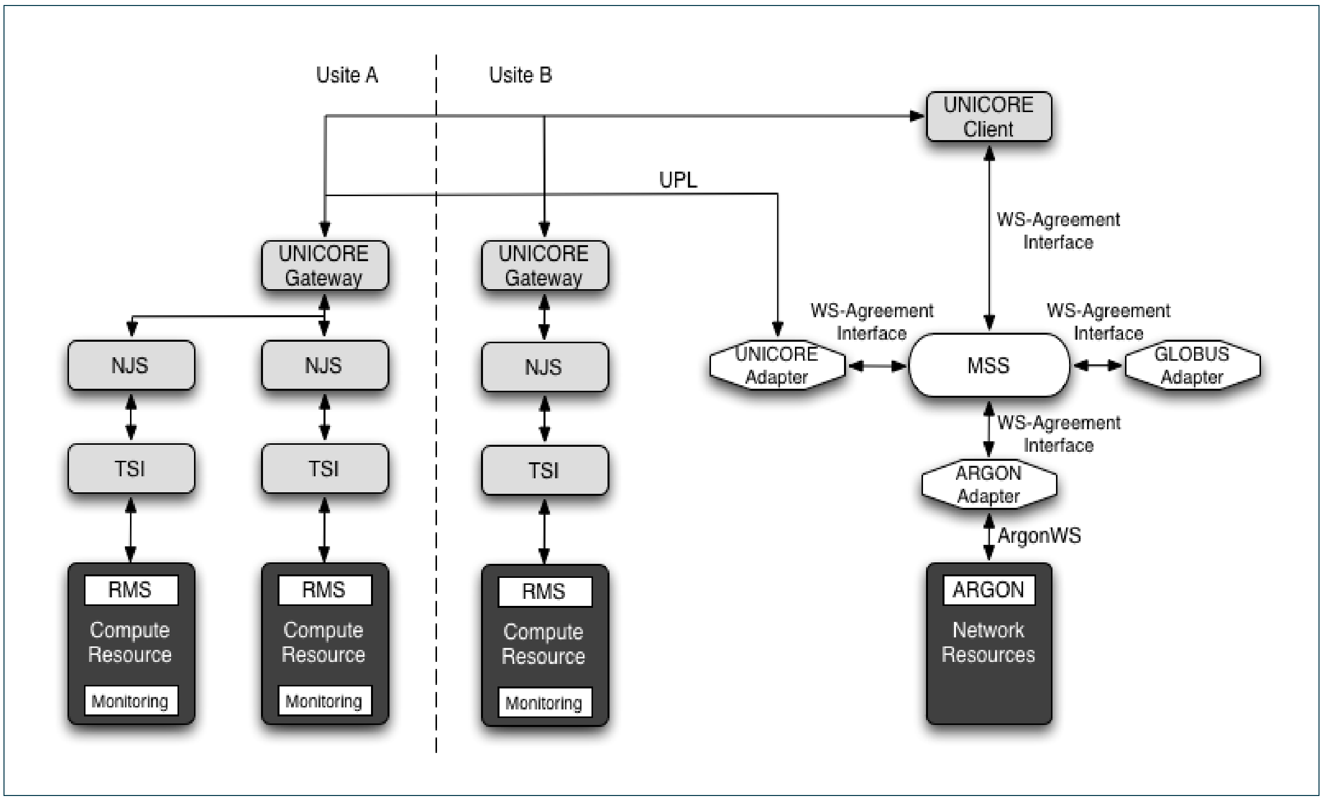


Figure 1: Integration of the MetaScheduling Service and UNICORE

Based on the WP1 specification of the network service plane (NSP) the MSS adapter towards the ARGON NRPS was replaced by an adapter for the NSP. Figure 2 shows the current architecture that was used for the demonstrations during the first project review in December 2007 and which is now used as part of the regular PHOSPHORUS testbed infrastructure. As Figure 2 depicts the different NRPS available in the PHOSPHORUS network environment having an interface to the NSP while the request for network resources from the MSS is handled by the NSP and transparently forwarded to the respective NRPS.

Figure 2 also shows the interface with the control plane and the G2MPLS developed by WP2 in the PHOSPHORUS project. The implementation of these interfaces has been mostly done after month 18 and is almost complete now. More details about the architecture can be found in section 5 (Task 3.3 – Interfacing between G²MPLS and Resource Management System via G-OUNI).

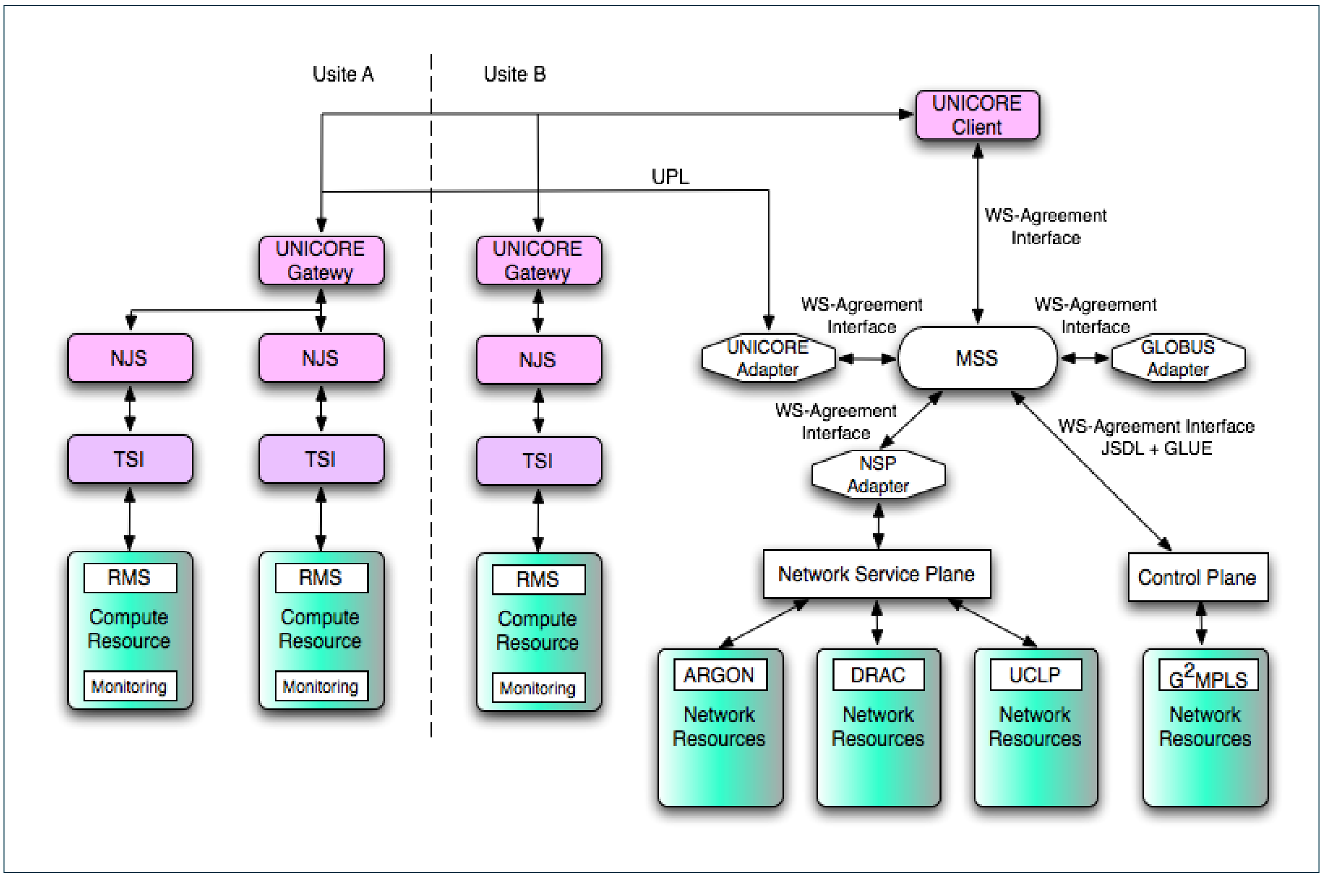


Figure 2: Integration of the MetaScheduling Service and the Network Service Plane and the Control Plane

The final modifications of the MSS concerned the integration in the new, web services-based UNICORE 6, which replaced UNICORE 5 as Grid middleware in the testbed. The modifications concerned mainly the new security model of UNICORE 6, which made changes in the MSS interfaces to the UNICORE client and the UNICORE gateway necessary. As a result, the middleware in the PHOSPHORUS testbed is following the WS-Security standard now.

At month 26 of the project the MSS installation in the testbed is stable and ready to be used in the final phase of testbed experiments. Test and potential improvements of the implementation of the interface to the network control plane to provide middleware service to the network layer will be in the focus during this period..

3.1 Summary of the achievements

Interface between the MetaScheduling Service and the Network Service Plane

- NSP exposes a Web Service Interface

- Reservation of connections, also in advance
  - Management of multiple connections inside one NSP service
- Queries about the state of a reservation
- Termination of a reservation

Interface between the MetaScheduling Service and the G2MPLS

- WS-Agreement has been selected as protocol and language to request co-allocations
- MSS exposes a co-allocation service
- Resulting in an SLA on the co-allocation
- Interface to G2MPLS based on BES and GLUE

User Authentication and Authorisation information (currently X.509 certificates) forwarded to the network layer

- Certificates to be used for token-based authentication and authorisation

Integration of MSS components with UNICORE 6

- WSAG4UNICORE6 features:
  - Support of UNICORE 6 security
  - trust delegation to support complete chain of trust between user, wsag4unicore6 adapter, and UNICORE 6 server components
  - WS-Security for trust delegation verification
  - Default implementation of WS-Agreement based SLA’s
  - File Stage-in / File Stage-out
    - UNICORE 6 server to server single file transfers
    - UNICORE BFT for stage-in / RBYTEIO for stage-out
  - File Stage-in from archive / File Stage-out to archive
    - Transfer of complete directory structures
    - Archives are unpacked on target system (stage-in)
    - Archives are created on source systems and transferred to target systems (stage-out)
    - UNICORE BFT for stage-in / RBYTEIO for stage-out

The overall result is WSAG4UNICORE6 – WS-Agreement based UNICORE 6 integration

4 Resource Selection Service

The Resource Selection Service (RSS) has been designed in order to allow the MetaScheduling Service (MSS) to place a job request automatically on an appropriate computing system taking into account the specific requirements of the job as imposed by the application to be executed.

To fulfil this requirement, the RSS framework has bee designed and implemented in WP3. This framework consists mainly of two different ontologies, which in turn will deliver on enquiry

- a list of cluster names suitable for the job out of which the MSS selects one by negotiating, or
- an empty list in case there is non available

4.1 Testbed description

Experiments were carried out using resources of the VIOLA testbed where the MetaScheduling Service of the PHOSPORUS testbeds is hosted. Additionally, resources of the PSNC testbed and resources of the testbed of the University of Essex were used as targets of the resource selection. The VIOLA testbed and the PSNC testbed are connected through a dedicated cross-border dark-fibre, the VIOLA testbed and the testbed of the University of Essex are connected by a dedicated 1 Gbit link.

4.2 Ontology design

We have designed two different ontologies (OWL-Files: one for the requirements of the applications and one for the offered resources of the clusters in the testbed). The classes and its instances are annotated with different data-types and the designated values of the application´ requirements and the properties of the computing systems which are provided in the testbed. Report D3.7 provides more details on the ontologies.

The RSS is modelled as a Web-Service within the Jena framework, which allows running SPARQL queries against the ontologies in a Java environment. The matching process of the results from the ontologies is conducted in this Jena framework.

4.3 Tests within the Jena Environment

Only local experiments in the VIOLA testbed have been conducted so far to ensure that the RSS framework runs properly. Since in the current design of the PHOSPHORUS testbed only one instance of the MSS running at one of the sites serving also all other sites (actually it is running at a node at FHG) is needed to provide the Grid Scheduler functionality to all testbed sites the resource selection capabilities are automatically available at all testbed sites.

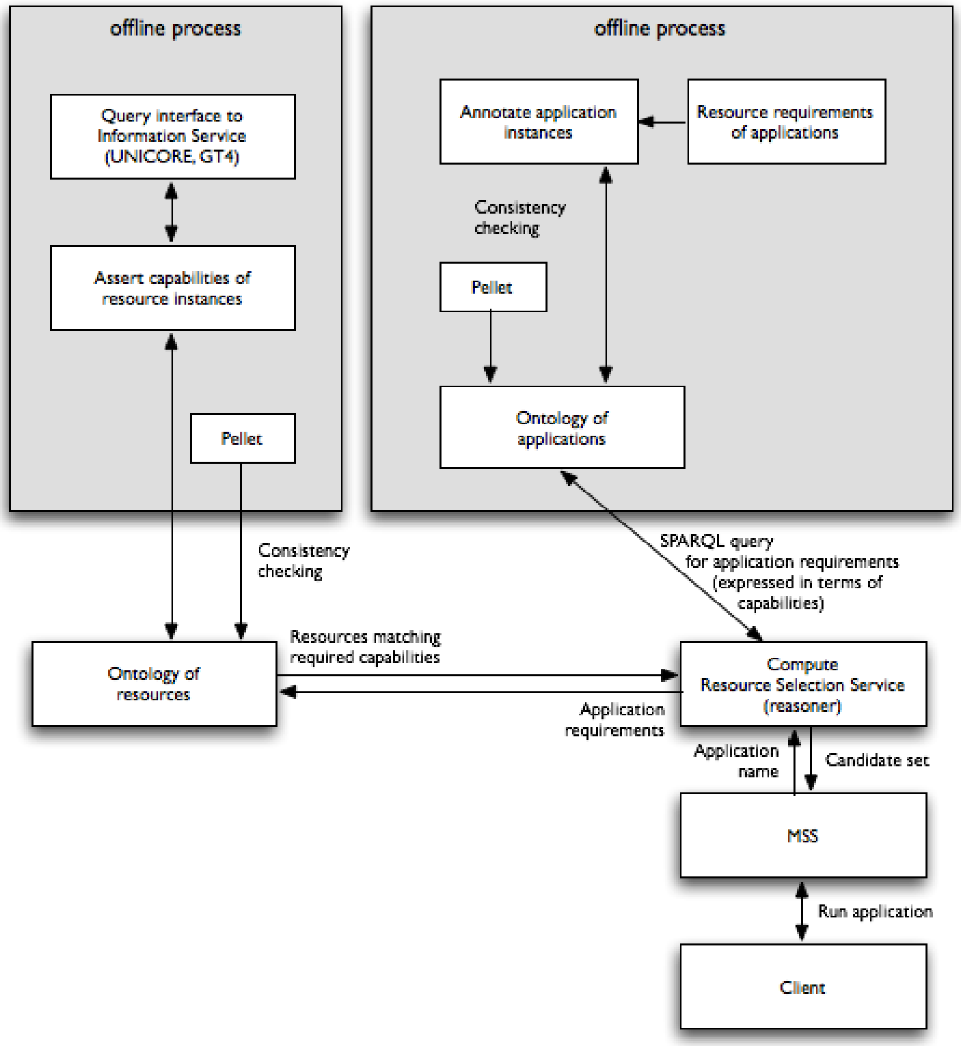

offline process
Query interface to Information Service (UNICORE, GT4)
Assert capabilities of resource instances
Pellet
offline process
Annotate application instances
Resource requirements of applications
Consistency checking
Pellet
Ontology of applications
Consistency checking
SPARQL query for application requirements (expressed in terms of capabilities)
Resources matching required capabilities
Ontology of resources
Compute Resource Selection Service (reasoner)
Application requirements
Application name
Candidate set
MSS
Run application
Client

Figure 3: Resource Selection Service Framework

In fact, we tested a) the framework with each application (WISDOM, KoDaVis, and DDSS) which will be available in the testbed. The TOPS application is only installed at two sites (SARA and FHG) where each site has a distinct functionality and role in the setup. Thus, the application may not move around in the testbed and TOPS is not included in the mapping. Figure 3 gives an overview on the framework at the end of month 26. More details on the implementation and the experiments may be found in D3.7.

4.4 Summary of the achievements

Overall architecture and framework of the RSS defined and implemented

- Implementation of the OGSA-Resource Selection Service
  - as a Web-Service interfacing with the MSS based on the preliminary OGF specification
- Ontologies are designed and evaluated for both
  - Resource capabilities
  - Application requirements

5 G^{2}MPLS

G²MPLS is a Network Control Plane (NCP) architecture used in PHOSPHORUS that implements the concept of Grid Network Services (GNS). The GNS allows the provisioning of network and Grid resources in a single-step. The goal of the GNS is to make network resources in available in a similar way as other grid resources, like CPUs, memory, contents or OS processes.

G²MPLS is aimed to provide functionalities related to the selection, co-allocation and maintenance of both Grid and network resources. This goal translates in:

- Discovery and advertisement of Grid capabilities and resources of the participating Grid sites (Vsites);
- Grid and Network Service setup including:
  - Coordination with the Grid local job scheduler in the middleware responsible for the local configuration and management of the Grid job;
  - Configuration of the network connections among the Vsites participating to the Grid job;
  - Management of resiliency for the installed network services and possible escalation to the Grid middleware components that could be responsible for check-pointing and recovering the whole job;
  - Advanced reservations of Grid and network resources;
- Service monitoring both for the Grid job and the related network connections.

To achieve the above mentioned goal the implementation was providing three different interfaces to support

- the G²MPLS overlay model
- the G²MPLS integrated model
- the G.OUNI interface

The functionality of the interfaces is briefly described below. More information about the underlying technologies used, the frameworks and standards the implementation is based upon may be found in D3.7.

5.1 G²MPLS overlay model

In G2MPLS Overlay model, the Grid layer has Grid and network routing knowledge in order to provide Grid resource configuration and monitoring (as in its standard behaviour) plus network resource configuration and monitoring. G2MPLS acts as an information bearer of network and Grid resources and as a configuration "arm" just for the network service part.

This model is intended to be mainly deployed when most of the computational and service intelligence need to be maintained in the Grid layer for specific middleware design and functional behaviours. The Grid scheduler in this case plays the leading role, which is the overall responsible for initiation and coordination of the reservation process through the participating Grid sites and the network in between.

5.2 G²MPLS integrated model

In the G2MPLS Integrated model, most of the functionalities for resource advance reservation and commit are moved to the G2MPLS Network Control Plane. G2MPLS is responsible for scheduling and configuring all the job parts, those related to the Grid sites and those related to the network.

Grid sites are modelled as special network nodes with specific additional Grid resource information (ref. green and dark grey shapes in Figure 4. The resulting topology is flat and integrated with respect to the positioning of the Grid layer against the network layer.

5.3 The G-OUNI interface

The Grid Optical User Network Interface (G.OUNI) comprises a number of procedures to facilitate on demand as well as in-advance access to Grid services/resources by interfacing Grid end points and any Grid middleware with any type of network resource provisioning system.

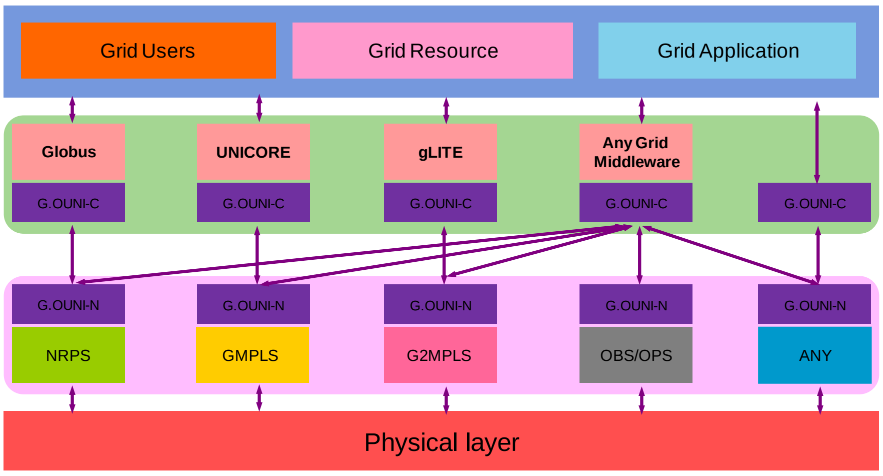


Figure 4: Grid User Network Interface with grid endpoints as well as Grid middleware with Network Provisioning Systems

The G.OUNI reference point acts as the interface between Grid End Points and the Grid Network Service Provisioning Systems as shown in Figure 4.

5.4 Communication with the G²MPLS layer

The gSOAP engine is used to implement a web service based communication between G.OUNI gateway and MSS. Since the gSOAP engine does not provide support for stateful web services, currently the OGSA BES (Basic Execution Service) protocol is foreseen as the basic protocol for the MSS-G.OUNI interaction. The OGSA BES interface is implemented as a stateless web service. This has makes the implementation on the G.OUNI gateway site much easier. Furthermore, UNICORE 6 provides an OGSA BES interface since version 6.1. Therefore, this functionality can be accessed directly from the GOUNI gateway.

5.5 Summary of the achievements

The interfaces implemented and available in the testbed by the end of the reporting period support both the G^{2}MPLS overlay model and the G^{2}MPLS integrated model.

G²MPLS Overlay Model

- the Network is seen as an additional Grid resource

- it can be used by Grid services as any other resource
- Grid services access G²MPLS via G.OUNI interface
- this supports the more “traditional” Grid approach

G²MPLS Integrated Model

- Grid intelligence is moved to network layer
- G²MPLS is responsible for scheduling and configuring (atomic) jobs
- Grid scheduler are still responsible for workflow execution
- Grid information (e.g. resource description of sites, availability information) is injected into G²MPLS via G.OUNI gateway
- Job submission to G²MPLS via G.OUNI gateway

To realise these interfaces a number of additional steps towards the integration of MSS, UNICORE and G²MPLS have been performed.

In order to enable the MSS to work with UNICORE 6, the new adapter is used to make UNICORE 6 functionality accessible via WS-Agreement protocol. Since UNICORE 6 poses high requirements in terms of security, a number of extensions were made to the MSS environment. These extensions include:

Enabling WS-Security for the MSS in order to digitally sign the SOAP messages send by the MSS client and server components, and to validate the digital signatures of the messages received by the MSS client and server components. Since these concepts are unknown on the network layer the MSS is bridging between these two environments.

Enable the MSS to support UNICORE 6 trust delegations. UNICORE 6 uses SAML 2.0 assertions to delegate trust from the issuer of a job (e.g. a user) to the entity that actually submits a job to UNICORE (e.g. a scheduler). Therefore, additional functionality to generate UNICORE 6 trust delegations (TD) by the MSS client, to validate TD objects via the digital message signature on the server side, and to generate and validate of trust delegation chains is required for the MSS.

Framework properties at month 26

- Glue based interface to inject and receive routing information from/to the G^{2}MPLS
- BES based interface to submit / receive Grid jobs from G²MPLS
- integration of MSS with G²MPLS via G.OUNI Gateway
- all required components implemented and tested

6 INCA middleware

As for the first layer (**Data Management Layer**) we have implemented and evaluated the following functionalities:

1. Node insertion and removal
2. Distributed routing table and DHT zone maintenance
3. Application layer routing mechanism
4. Metadata insertion and removal
5. Distributed metadata base maintenance
6. Metadata insertion and data placement
7. Metadata location and data retrieval

Additionally for the second layer (**Storage protocol layer**) we have evaluated:

1. Point-to-point bandwidth utilization
2. ACK strategies during high rate transmissions.
3. Traffic Congestion.
4. Rate Adaptation.
5. Implementation of an Authentication Checksum in order to secure both the non-corruption of packets and the non-spoofing.

Through the implementation we have done useful observations that have positively affected our **Data management layer**. For the maintenance of a DHT and the routing tables special attention has to be paid in order to keep the zones and the routing tables consistent. During node insertion and removal temporary inconsistency takes place and system has to handle this situation.

The implementation of the routing mechanism affects the performance, during the routing process, and the stability of the system in dynamic conditions. We have implemented a routing algorithm that infuses these two parameters in INCA.

The hash function that used in the DHTs towards the balanced metadata placement performs well. As for the balanced data placement small data chunks have to be used due to the variant file size of the files that inserted in the system.

At last as replication is the only way towards a fault tolerant system special attention has to be paid on it in order to avoid inconsistency and data loss.

As for the implementation of the **Storage protocol layer** our experiments we have pointed out some issues on the testbed, as we achieved performances that are 50% less the one achieved on another testbed. The issues are partially linked to the kind of access and hardware, but this brought to evidence some key points we are addressing now. More details are in D3.7.

In contrast to the previous phases in the last phase INCA used simulations to evaluate the enhanced functionality in different simulated network topologies.

The INCA system has been evaluated over extensive simulations. Instead of a custom made simulator the OPNET Modeller was used to improve the reliability of the simulations. In this section the results of the simulations are presented for various numbers of nodes and various underlying network topologies.

As an objective during this phase was to perform the simulations based on a realistic network model where even the triangle inequality is violated, real round trip time measurements have been used derived from Meridian King data set which provides RTT measurements among 2500 nodes.

The simulation experiments and their results are described in report D3.7.

Summarising the evaluation of our system, INCA successfully captures locality information, maintaining balanced routing tables and is orthogonal to any load balance algorithm for the distribution of the keys to the various nodes.

6.1 Summary of the achievements

Implemented and deployed in the testbed

- Storage Protocol layer, the final novell version of Storage Protocol Layer API and Storage Protocol Layer enables
    - dynamic adjustable bandwidth utilization;
    - loosely coupled congestion control mechanism;
    - asynchronous/synchronous transfer mode;
    - object, block, chunk transfer mode;
    - packet alteration monitoring
- Evaluation of the INCA system using extensive simulations with 2.500 nodes

- Performance
- Scalability

7 Adaptation, deployment and tests of the PHOSPHORUS application suite

7.1 WISDOM

7.1.1 Short introduction

The WISDOM use-case consists of the virtual screening techniques AutoDock and FlexX, computing compounds of large-scale molecular dockings on targets implicated in diseases like malaria.

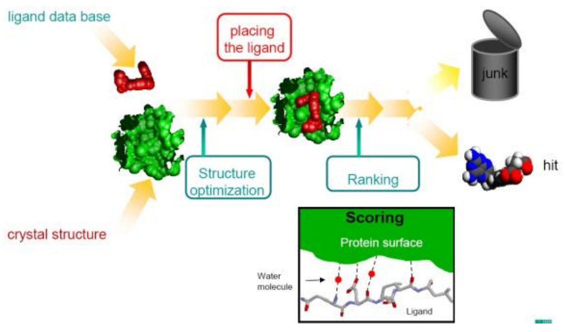


Figure 5: Virtual Screening Technique

- AutoDock is a suite of automated docking tools. It is designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of a known 3D structure.
- FlexX is an extremely fast, robust, and highly configurable (FlexX-able) computer program for predicting protein-ligand interactions.

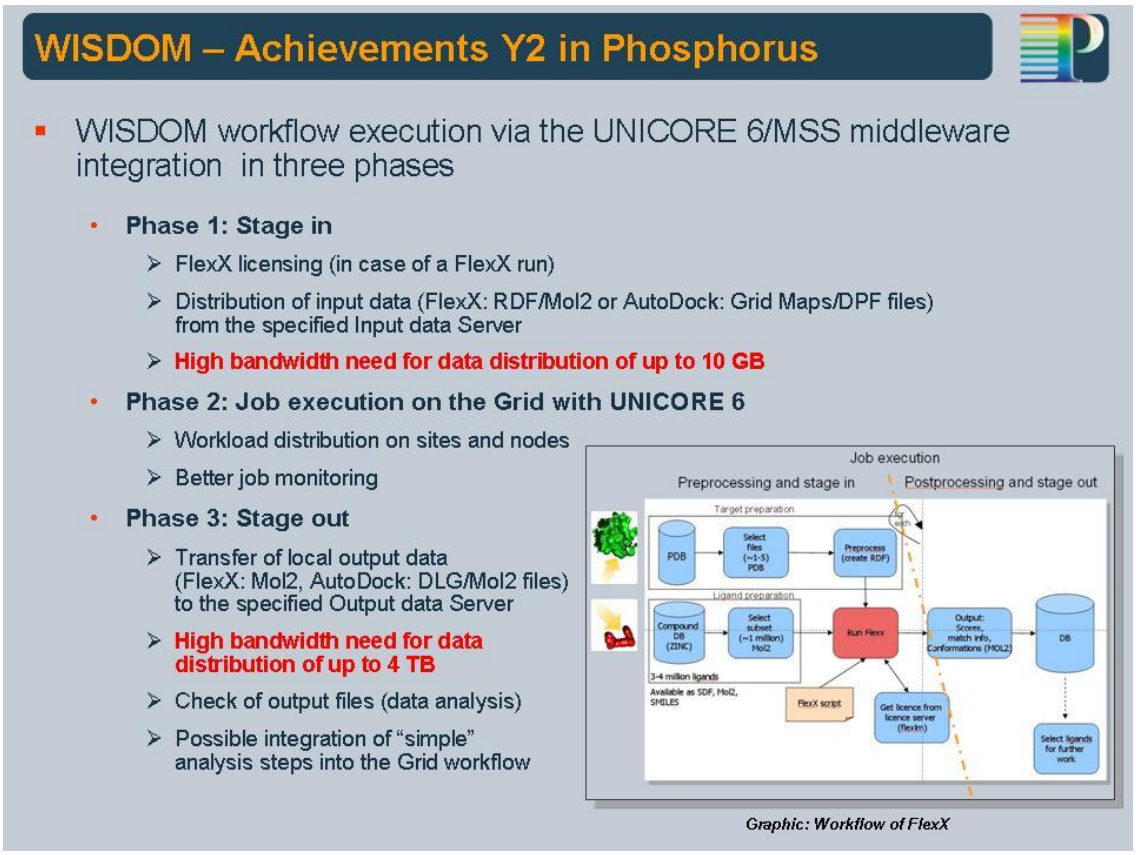
More details about the PHOSPHORUS use case WISDOM and its applications are described in deliverable D3.5, and D3.6.

7.1.2 Planned actions

Based on the experiences made during the EGEE WISDOM data challenge the goal in Phosphorus is to implement the WISDOM workflow with UNICORE 6 in order to improve correctness, completeness and reliability of results. Thus, both the distribution of input data and especially the transfer of result data of the millions of docking processes from the participating sites back to the user's site were complex, cumbersome and therefore resulting in significant data losses. As a consequence of this we will concentrate in PHOSPHORUS besides the test of network and middleware functionalities in the implementation of a perfect workflow for WISDOM. Thus MSS/UNICORE 6 integration enabled executions of WISDOM jobs. More details about the WISDOM codes' workflows and requirements for execution are described in D3.5.

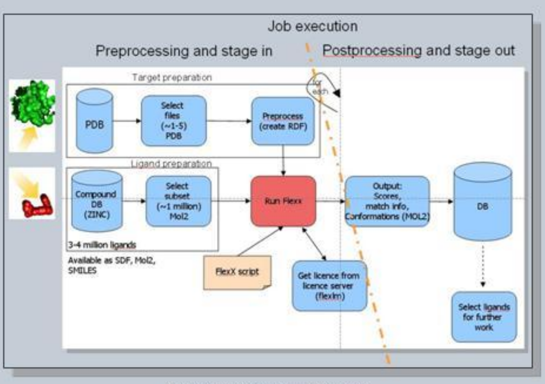
7.1.3 Achievements and results

After the inspection of the first EGEE Data Challenge and their WISDOM workflow necessary changes for the PHOSPHORUS environment were identified. As a result of this analysis different WISDOM workflow phases were identified, which must be enabled based on MSS and UNICORE 6 integration: Stage-in, execution and stage-out phases for each WISDOM job.



WISDOM – Achievements Y2 in Phosphorus

- WISDOM workflow execution via the UNICORE 6/MSS middleware integration in three phases
  - **Phase 1: Stage in**
    - FlexX licensing (in case of a FlexX run)
    - Distribution of input data (FlexX: RDF/Mol2 or AutoDock: Grid Maps/DPF files) from the specified Input data Server
    - **High bandwidth need for data distribution of up to 10 GB**
  - **Phase 2: Job execution on the Grid with UNICORE 6**
    - Workload distribution on sites and nodes
    - Better job monitoring
  - **Phase 3: Stage out**
    - Transfer of local output data (FlexX: Mol2, AutoDock: DLG/Mol2 files) to the specified Output data Server
    - **High bandwidth need for data distribution of up to 4 TB**
    - Check of output files (data analysis)
    - Possible integration of "simple" analysis steps into the Grid workflow


Job execution
Preprocessing and stage in
Postprocessing and stage out
Target preparation
for each
PDB
Select files (~1-5) PDB
Preprocess (create RDF)
Ligand preparation
Compound DB (ZINC)
Select subset (~1 million) Mol2
Run Flexx
Output: Scores, match info, Conformations (MOL2)
DB
3-4 million ligands
Available as SDF, Mol2, SMILES
FlexX script
Get licence from licence server (flexlm)
Select ligands for further work

Graphic: Workflow of FlexX

Figure 6: WISDOM workflow with stage in and stage out phase

For WISDOM application tests the BioSolveIT flexx-200 Testdata suite was selected and adapted to the different data formats of both the applications AutoDock and FlexX.

Data Transfer methods

On the technical side, UNICORE 6 complies with the OASIS WSRF 1.2 and OGF JSDL 1.0 standards, provides pluggable file transfer mechanisms with the OGSA ByteIO standard as default. The usual way of transferring a file is, that first the client sends a SOAP message to the server, which initiates a file transfer object. The client gets as an answer a link to this object, thus the client is enabled for direct accesses. Via methods of the object further information and data can be exchanged. After termination of the file transfer, the link will be deleted. This procedure is repeated for each file transfer operation.

UNICORE-ByteIO is the standard method for file transfer in UNICORE 6. It uses the feature in UNICORE to transfer information via web services. More information about this can be found in Deliverable D3.6.

UNICORE 6 Rich Client

UNICORE 6 offers three different clients to the users, UNICORE Command-line Client (UCC), UNICORE Grid Programming Environment Client (GPE) and the recently developed UNICORE 6 Rich Client. This graphical UNICORE Client is based on the Eclipse rich client platform (RCP). This client was used to monitor the MSS/UNICORE 6 executions of WISDOM jobs.

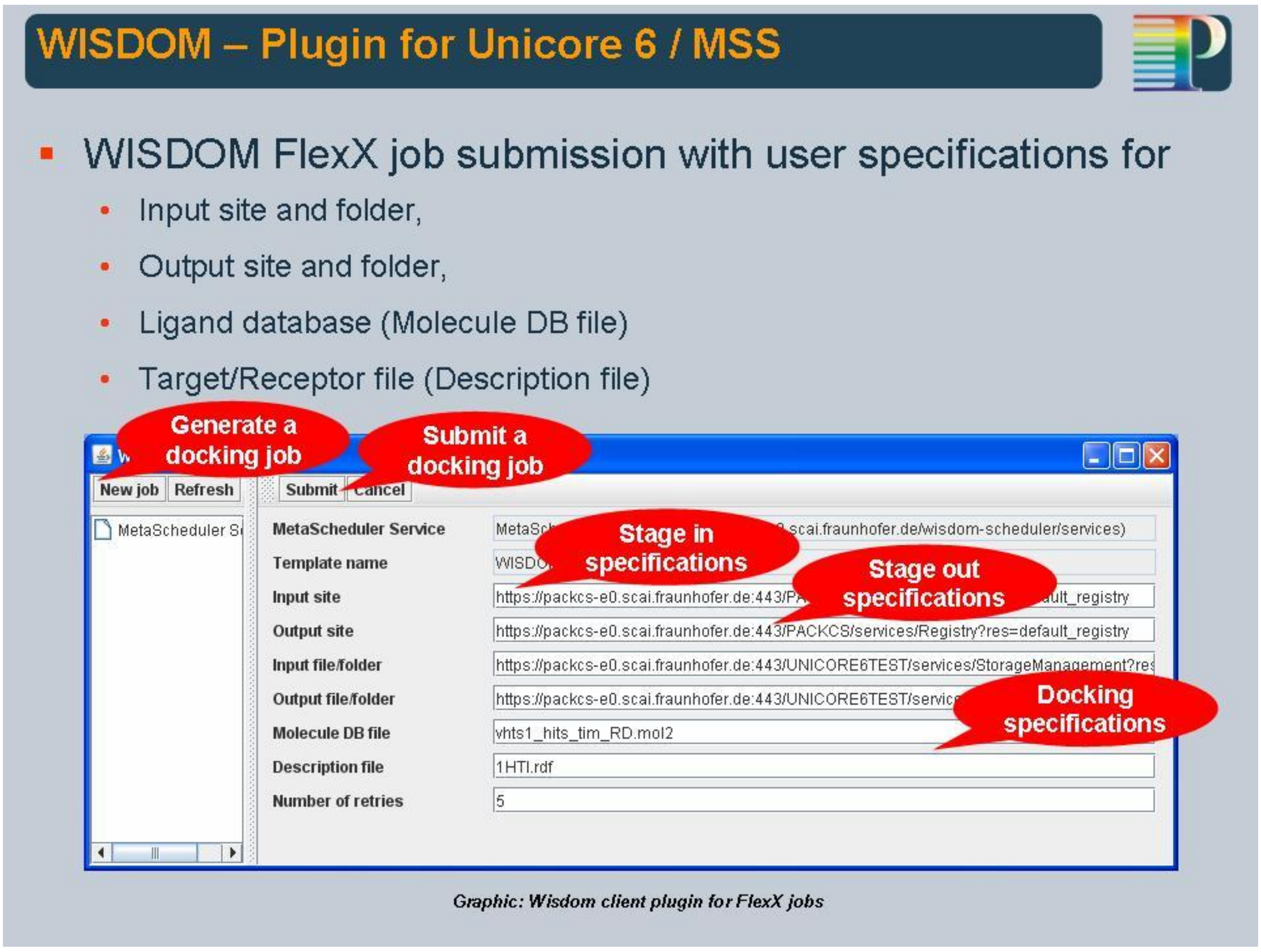
On each resource, appropriate operations and sub-items are available, for example a job can be created on a target system resource, and files can be downloaded from storage resources. Before any job creation can be performed, the authentication process should be completed successfully. For access to the PHOPHORUS site JUGGLE the D-Grid GridKA certificate (shown in the following Figure) is needed and accepted. Security views allow the user to manage her credentials and trusted certificates. The execution of jobs and workflows can be monitored then in different ways and job outcomes can be fetched and visualised.

During the next sections we are concentrating on the WISDOM workflow with the following phases:

- The WISDOM stage-in phase consists of licensing (for FlexX only) and distribution of input data from a specified input server. In the case of FlexX, these are RDF and Mol2 files. AutoDock uses Grid maps and DPF files for execution.
- The WISDOM execution phase needs a good job control and monitoring.
- The WISDOM stage-out phase includes the transfer of the local output data (FlexX Mol2 files, AutoDock DLG and Mol2 files) after termination from each site to a specified output server into a directory hierarchy or a database.
- Additionally there may follow pre- and postprocessing actions, e.g. input data format conversion, output data analysis, output data filtering.

In the following different Rich Client views are presented, showing the different phases of WISDOM application jobs.

A SWING-based WISDOM plug-in was implemented to realize job execution via the MSS / UNICORE 6 integrated Grid middlewares. Additionally job and application specifications can be done for the WISDOM FlexX jobs, e.g. input and output data location and specific parameters (receptor and ligand database). The following graphic shows this plug-in.



WISDOM – Plugin for Unicore 6 / MSS

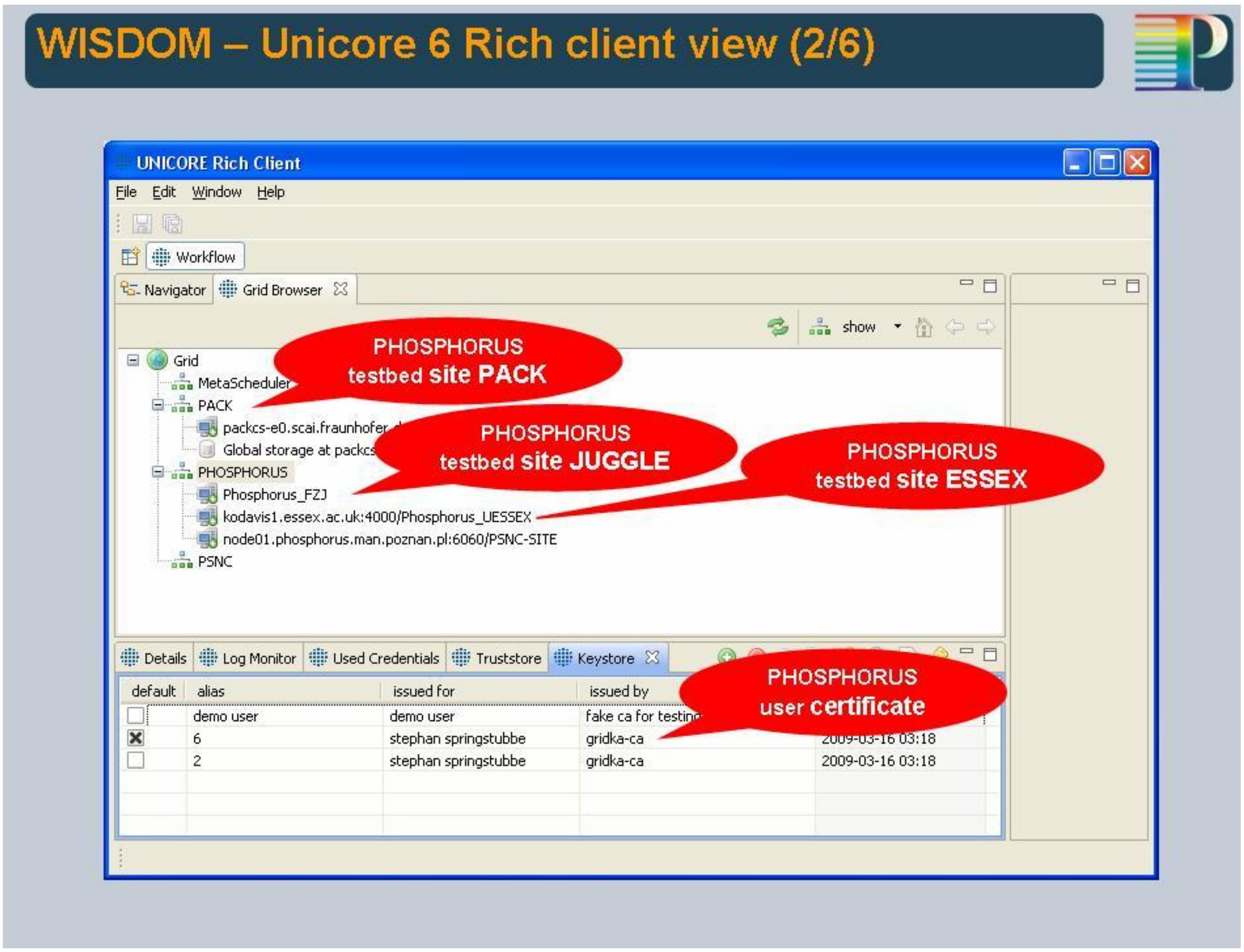
- WISDOM FlexX job submission with user specifications for
  - Input site and folder,
  - Output site and folder,
  - Ligand database (Molecule DB file)
  - Target/Receptor file (Description file)

MetaScheduler Service	MetaSch… …scai.fraunhofer.de/wisdom-scheduler/services)
Template name	WISDO…
Input site	https://packcs-e0.scai.fraunhofer.de:443/PA… …ult_registry
Output site	https://packcs-e0.scai.fraunhofer.de:443/PACKCS/services/Registry?res=default_registry
Input file/folder	https://packcs-e0.scai.fraunhofer.de:443/UNICORE6TEST/services/StorageManagement?res
Output file/folder	https://packcs-e0.scai.fraunhofer.de:443/UNICORE6TEST/servic…
Molecule DB file	vhts1_hits_tim_RD.mol2
Description file	1HTI.rdf
Number of retries	5

Graphic: Wisdom client plugin for FlexX jobs

Figure 7: Screenshot of WISDOM plug-in with specifications for a FlexX run

The user can easily generate jobs using and modifying default values. After these specifications the user starts WISDOM FlexX jobs via the Submit button on the PHOSPHORUS testbed. The different workflow phases can be monitored via the UNICORE 6 Rich client and its so-called "Grid browser" view, which shows the Grid resources available to the user in a tree-like fashion. The following Figure shows part of the PHOSPHORUS testbed with sites JUGGLE, PACK and ESSEX, which are the appropriate platforms for WISDOM FlexX jobs.



WISDOM – Unicore 6 Rich client view (2/6)

UNICORE Rich Client

File Edit Window Help

Workflow

Navigator | Grid Browser

```
Grid
  MetaScheduler
  PACK
    packcs-e0.scai.fraunhofer
    Global storage at packcs
  PHOSPHORUS
    Phosphorus_FZJ
    kodavis1.essex.ac.uk:4000/Phosphorus_UESSEX
    node01.phosphorus.man.poznan.pl:6060/PSNC-SITE
  PSNC
```

Details | Log Monitor | Used Credentials | Truststore | Keystore

default	alias	issued for	issued by	
☐	demo user	demo user	fake ca for testing	
☒	6	stephan springstubbe	gridka-ca	2009-03-16 03:18
☐	2	stephan springstubbe	gridka-ca	2009-03-16 03:18

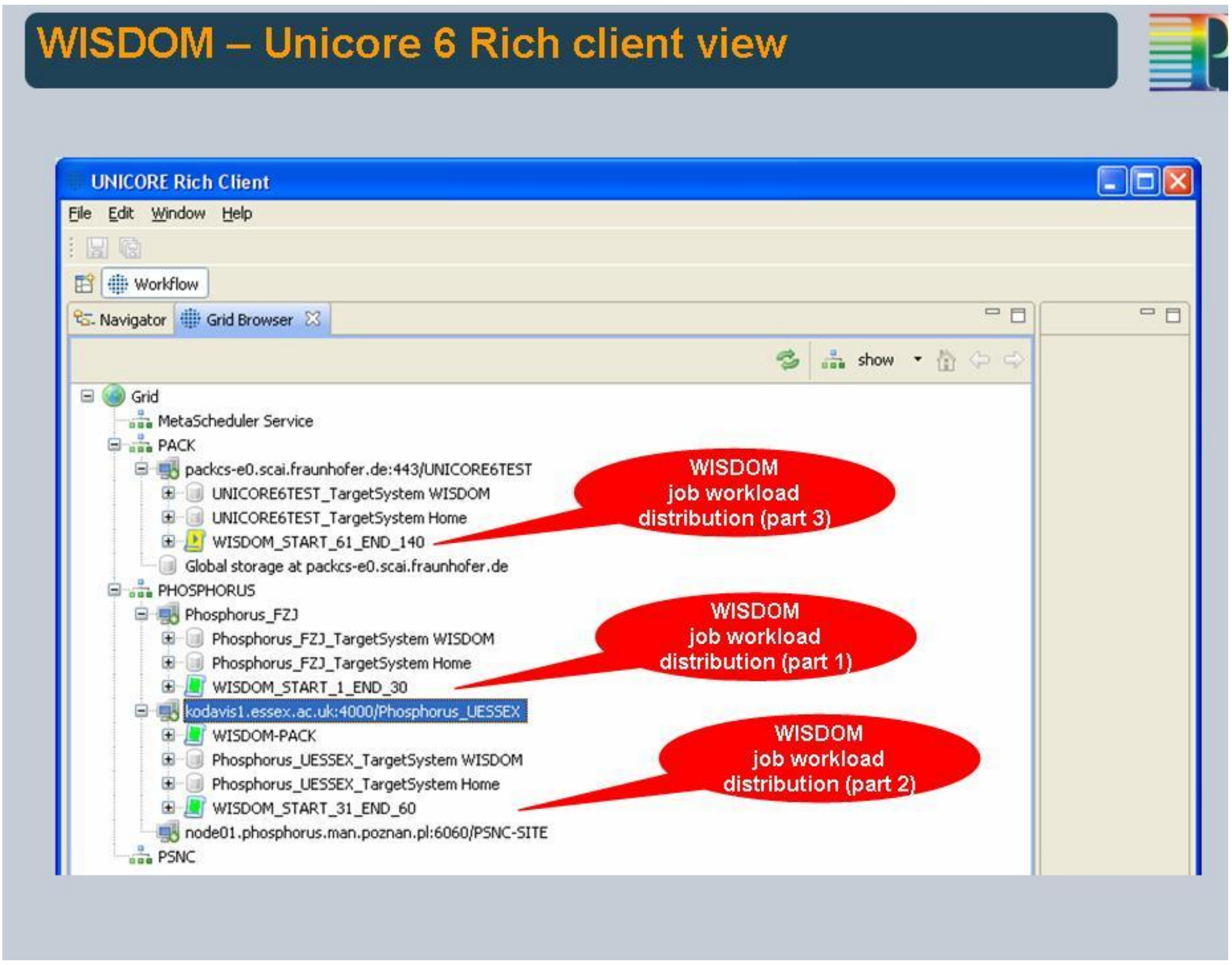
PHOSPHORUS testbed site PACK
PHOSPHORUS testbed site JUGGLE
PHOSPHORUS testbed site ESSEX
PHOSPHORUS user certificate
show

Figure 8: Screenshot of the UNICORE 6 Rich client view on the PHOSPHORUS testbed with sites and user certificate

The screenshot shows the content of the UNICORE 6 Rich client with PHOSPHORUS testbed sites and the used user certificate for authorization and authentication.

The next figure shows the submitted WISDOM FlexX jobs on the testbed. There is a job workload distribution on the three participating sites. It can be recognized by the different indices, which indicate, which database ligand entries are used for the virtual screening.

Here a very small test case with 140 ligands is used to demonstrate the MSS/UNICORE 6 integration and the workflow of the job execution.


WISDOM – Unicore 6 Rich client view

UNICORE Rich Client

File Edit Window Help

Workflow

Navigator Grid Browser

show

```
Grid
  MetaScheduler Service
  PACK
    packcs-e0.scai.fraunhofer.de:443/UNICORE6TEST
      UNICORE6TEST_TargetSystem WISDOM
      UNICORE6TEST_TargetSystem Home
      WISDOM_START_61_END_140
    Global storage at packcs-e0.scai.fraunhofer.de
  PHOSPHORUS
    Phosphorus_FZJ
      Phosphorus_FZJ_TargetSystem WISDOM
      Phosphorus_FZJ_TargetSystem Home
      WISDOM_START_1_END_30
    kodavis1.essex.ac.uk:4000/Phosphorus_UESSEX
      WISDOM-PACK
      Phosphorus_UESSEX_TargetSystem WISDOM
      Phosphorus_UESSEX_TargetSystem Home
      WISDOM_START_31_END_60
    node01.phosphorus.man.poznan.pl:6060/PSNC-SITE
  PSNC
```

WISDOM job workload distribution (part 3)
WISDOM job workload distribution (part 1)
WISDOM job workload distribution (part 2)

Figure 9: Screenshot of the UNICORE 6 Rich client view on the PHOSPHORUS testbed with job and workload distribution (ligand database indices)

The indices indicate which database elements are used for the docking process. Accordingly to this the input data is distributed to the sites of the PHOSPHORUS testbed. Data transfer of tar-files containing the required directories and files does this.

After the transfer of the tar-files and completion of the untar-processes the directory structure for WISDOM FlexX job execution is available on all sites and nodes containing the specific input data for each site and node.

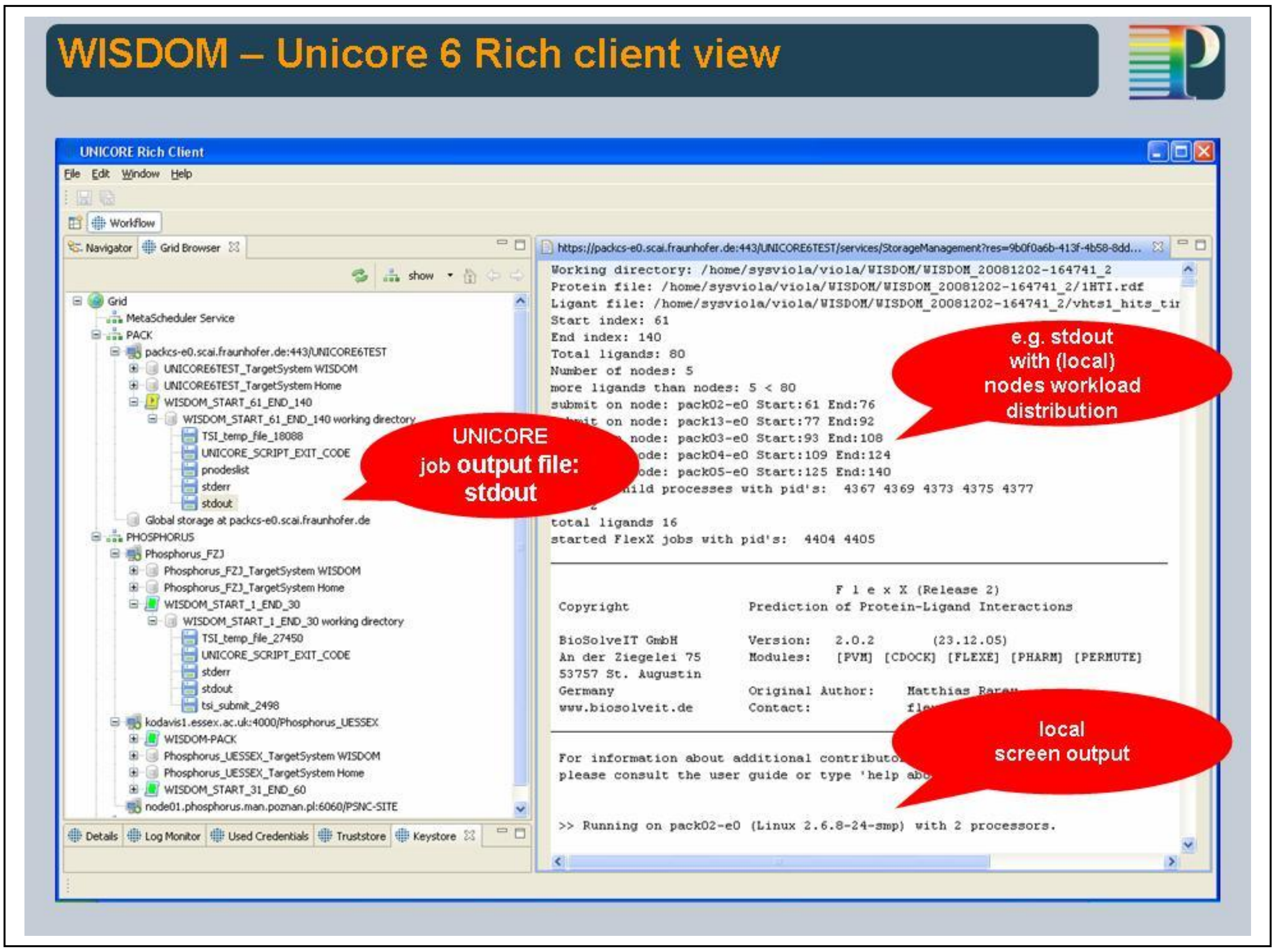
After stage-in process is finished the FlexX job can be executed on each node.

When the job execution is finished, the typical output files are produced and available. Thus there are for each site

- Exit_Code
- pnodeslist,
- stderr,
- stdout

which can be examined via the UNICORE client.

The next figure shows this part of the stdout file.


WISDOM – Unicore 6 Rich client view
UNICORE Rich Client
File Edit Window Help
Workflow
Navigator
Grid Browser
show
Grid
MetaScheduler Service
PACK
packcs-e0.scai.fraunhofer.de:443/UNICORE6TEST
UNICORE6TEST_TargetSystem WISDOM
UNICORE6TEST_TargetSystem Home
WISDOM_START_61_END_140
WISDOM_START_61_END_140 working directory
TSI_temp_file_18088
UNICORE_SCRIPT_EXIT_CODE
pnodeslist
stderr
stdout
Global storage at packcs-e0.scai.fraunhofer.de
PHOSPHORUS
Phosphorus_FZJ
Phosphorus_FZJ_TargetSystem WISDOM
Phosphorus_FZJ_TargetSystem Home
WISDOM_START_1_END_30
WISDOM_START_1_END_30 working directory
TSI_temp_file_27450
UNICORE_SCRIPT_EXIT_CODE
stderr
stdout
tsi_submit_2498
kodavis1.essex.ac.uk:4000/Phosphorus_UESSEX
WISDOM-PACK
Phosphorus_UESSEX_TargetSystem WISDOM
Phosphorus_UESSEX_TargetSystem Home
WISDOM_START_31_END_60
node01.phosphorus.man.poznan.pl:6060/PSNC-SITE
Details
Log Monitor
Used Credentials
Truststore
Keystore
https://packcs-e0.scai.fraunhofer.de:443/UNICORE6TEST/services/StorageManagement?res=9b0f0a6b-413f-4b58-8dd...
UNICORE job output file: stdout
e.g. stdout with (local) nodes workload distribution
local screen output

```
Working directory: /home/sysviola/viola/WISDOM/WISDOM_20081202-164741_2
Protein file: /home/sysviola/viola/WISDOM/WISDOM_20081202-164741_2/1HTI.rdf
Ligant file: /home/sysviola/viola/WISDOM/WISDOM_20081202-164741_2/vhts1_hits_tir
Start index: 61
End index: 140
Total ligands: 80
Number of nodes: 5
more ligands than nodes: 5 < 80
submit on node: pack02-e0 Start:61 End:76
      it on node: pack13-e0 Start:77 End:92
         node: pack03-e0 Start:93 End:108
          ode: pack04-e0 Start:109 End:124
          ode: pack05-e0 Start:125 End:140
          ild processes with pid's:  4367 4369 4373 4375 4377

total ligands 16
started FlexX jobs with pid's:  4404 4405
________________________________________________________________________

                                F l e x X (Release 2)
 Copyright               Prediction of Protein-Ligand Interactions

 BioSolveIT GmbH         Version:   2.0.2        (23.12.05)
 An der Ziegelei 75      Modules:   [PVM] [CDOCK] [FLEXE] [PHARM] [PERMUTE]
 53757 St. Augustin
 Germany                 Original Author:    Matthias Rarey
 www.biosolveit.de       Contact:            fle

 For information about additional contributo
 please consult the user guide or type 'help abo


 >> Running on pack02-e0 (Linux 2.6.8-24-smp) with 2 processors.
```

Figure 10: Screenshot of the UNICORE 6 Rich client view on the PHOSPHORUS testbed after job termination, e.g. stdout PACK

After termination of the stage-out phase the output directories and files are available on the specified output server.

To conclude, **WISDOM jobs** are running now on UNICORE 6 platforms on the PHOSPHORUS testbed. Jobs are executed using the WISDOM plug-in and the UNICORE 6 Rich Client, which offers an easy, user-friendly Graphical User Interface for pre-defined WISDOM AutoDock and FlexX runs.

WISDOM workflows are executed via the UNICORE 6 / MSS integration with stage in and stage out phase. No user actions needed to collect output data, no data losses. In the implementation WS-Agreement was used as a basic management protocol. Hereby the integration of MSS with UNICORE 6 is based on WSAG4UNICORE6. In this way we offer an open, standard based architecture, which is not coupled to a specific middleware, but actually coupled with UNICORE 6.

WISDOM benefits from the **network performance** provided transparently by the PHOSPHORUS network layer.

During the events Supercomputing 2008 and ICT 2008 **WISDOM demonstrations** were done on the PHOSPHORUS testbed. Jobs were started using the WISDOM plug-in and the Unicore 6 Rich client to execute WISDOM jobs in a user-friendly way on the Grid showing the different workflow phases of such jobs.

7.1.4 Summary of the achievements

- Workflow support has been implemented to overcome the limitations of the EGEE data challenge environment
- WISDOM workflows are executed via the UNICORE 6 / MSS integration with stage-in and stage-out phase. No user actions needed to collect output data, no data losses.
- Users may start jobs via the WISDOM plugin in the UNICORE 6 Rich Client offering an easy, user-friendly Graphical User Interface for pre-defined WISDOM AutoDock and FlexX runs.
- WISDOM applications benefit from the network performance provided transparently by the PHOSPHORUS network layer.

7.2 KoDaVis

7.2.1 Introduction

KoDaVis enables users from various remote sites to collaborate in visualising scientific data sets. It is made up of several components:

- Data server
- Collaboration server
- Visualisation client
- UNICORE 6 Client

For the purpose of the developments and tests conducted with this application, the data server and collaboration server components have been installed at FZJ. Client components (visualization and UNICORE) were deployed at various other sites. shows an experimental setup of two visualization clients both located at PSNC for demonstrational purposes. The visualization clients could as well be located at geographically dispersed sites. Any action taken by the user of either of the clients will be displayed in the other clients as well and thus allows the collaborative visualization of data. KoDaVis is a demanding application both in terms of latency and bandwidth. For high latencies, the user experience quickly deteriorates and may render the system unusable. At the same time, large sets of data may have to be transmitted, which requires high bandwidth links between KoDaVis servers and clients (~700 Mbits/s).

7.2.2 Extensions after the first wave of experiments

One of the findings during the first phase of testbed experiments with the KoDaVis application was the lack of information about poorly performing links or partners in a collaborative session. During a session, it was intractable to determine the cause of problems, if they occurred. Individual participants in a collaborative visualisation session can limit the entire session's performance. The endpoint with the worst performance in terms of latency and bandwidth determines the session's performance for all other participants. Therefore, monitoring capabilities were added to various components of the KoDaVis application. These monitoring capabilities are comprised of bandwidth monitoring of the connections between the data server and each of the clients. This is done by and displayed in the visualisation client and only helps the local user to see problems with his connection. Additionally, the collaboration and data servers keep track of the transfer times whenever bursts of data are transferred to the individual clients. This performance data is then queried by the UNICORE KoDaVis service, exposed in its properties and thus available in the KoDaVis UNICORE plug-in. The latter has been extended to display the performance data of the individual server to client connections. A participant in a collaborative visualization session would thus be able to determine those participants in the session that limit the overall session's performance. These participants could then be excluded from the session or informed about problems with their network link.

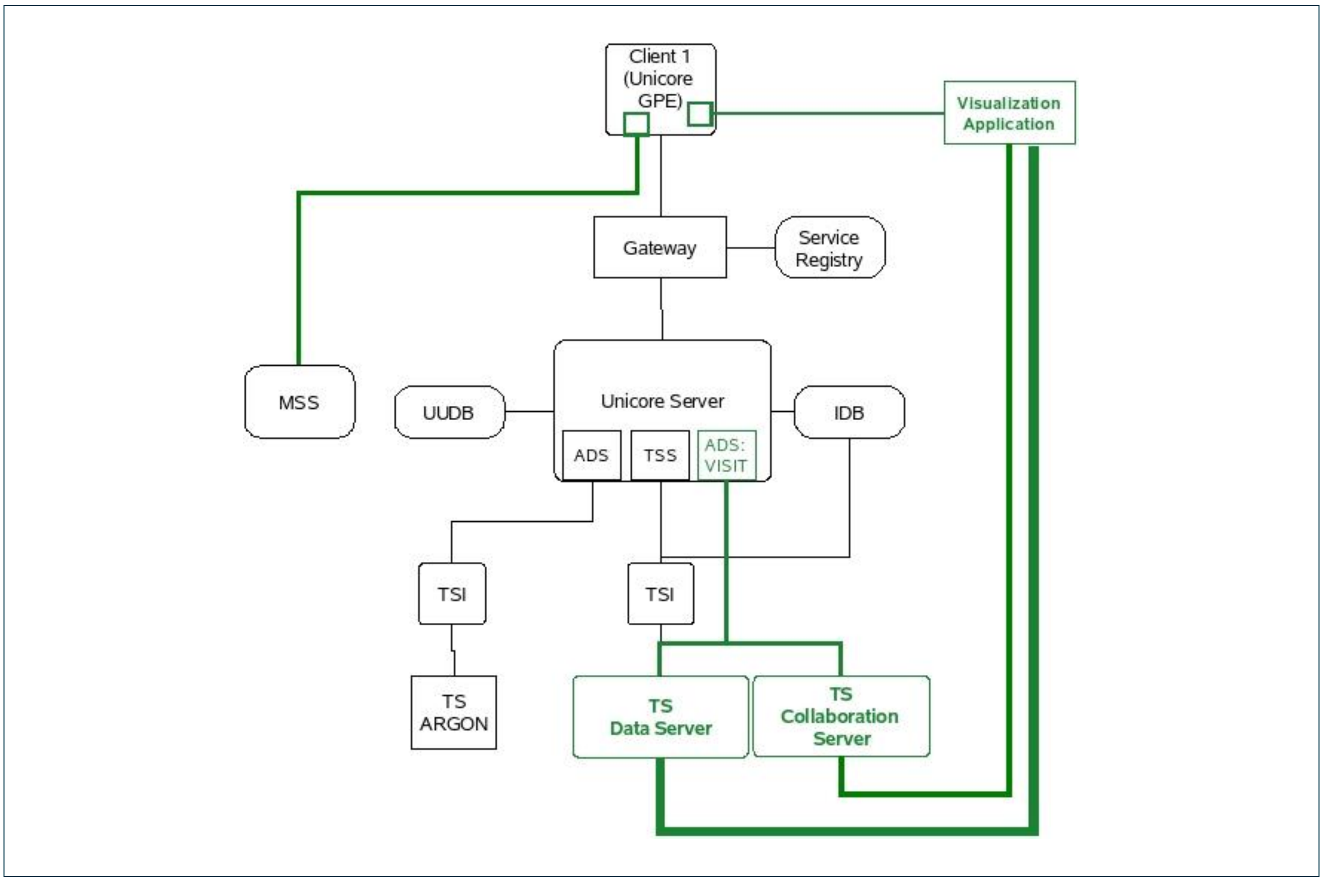


Figure 11: Architecture of the KoDaVis application embedded in UNICORE. Green Components have been developed in Phosphorus and have been tested successfully within the experiments.

Figure 11 is an overview of the architecture of the KoDaVis application, showing how it has been embedded into the UNICORE middleware. While this general setup hasn't changed from the initial reports on the architecture, it is worth noting that the interface between the "ADS: Visit" service, which is embedded in UNICORE and the collaboration and data servers has been greatly enhanced to allow for flexible performance monitoring.

Performance monitoring has been implemented as an extension to the XML based exchange protocol between the data and collaboration servers and their local clients facing UNICORE. In Figure 11 that's the green connection between ADS: VISIT and the TS Data Server and TS Collaboration Server. Both servers understand the same XML protocol, which controls the exchange of information about the data transfers. The data and collaboration servers can be configured dynamically to send this information in certain intervals.

7.2.3 Testbed experiments

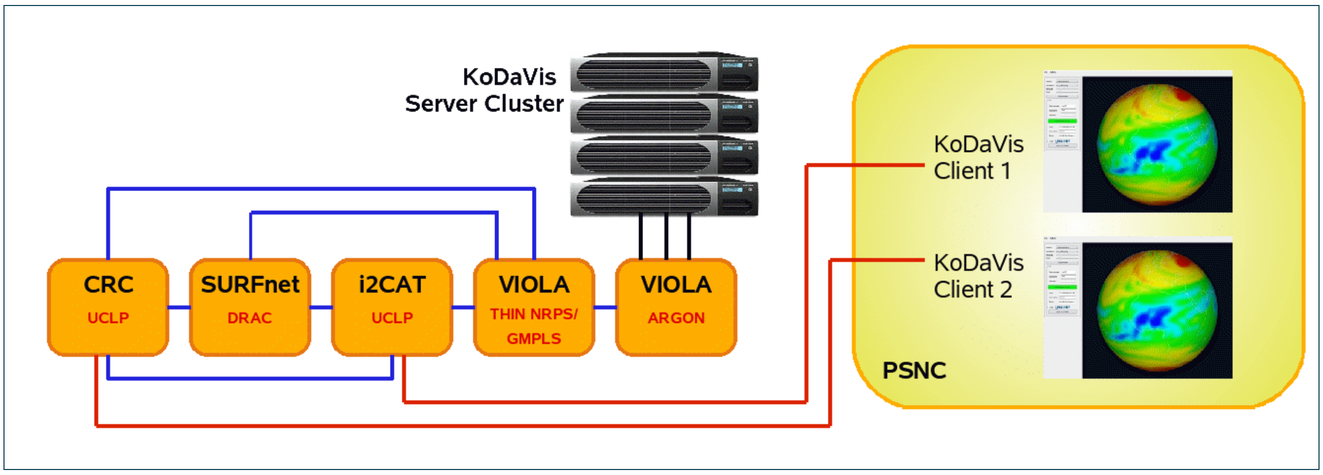


Figure 12: Example setup of a KoDaVis experiment, connecting two visualization clients at PSNC over different routes to the KoDaVis Server cluster in Jülich.

While quantitative experiments have been conducted during the first phase of experiments, the current wave of tests has a more qualitative focus. Their purpose is to evaluate the new features, which were added in between the two test phases. Its intention is to show their utility. During the first phase of experiments, it was still difficult for a user to identify those participants in a collaborative visualization session that make up the bottlenecks.

Figure 13 shows the transfer statistics window of the visualization client. It is only available on the client side where the data is provided to inform the user about the bandwidth of the connection between the data server and his client. The diagram was created on a machine with a 100Mbits/s connection.

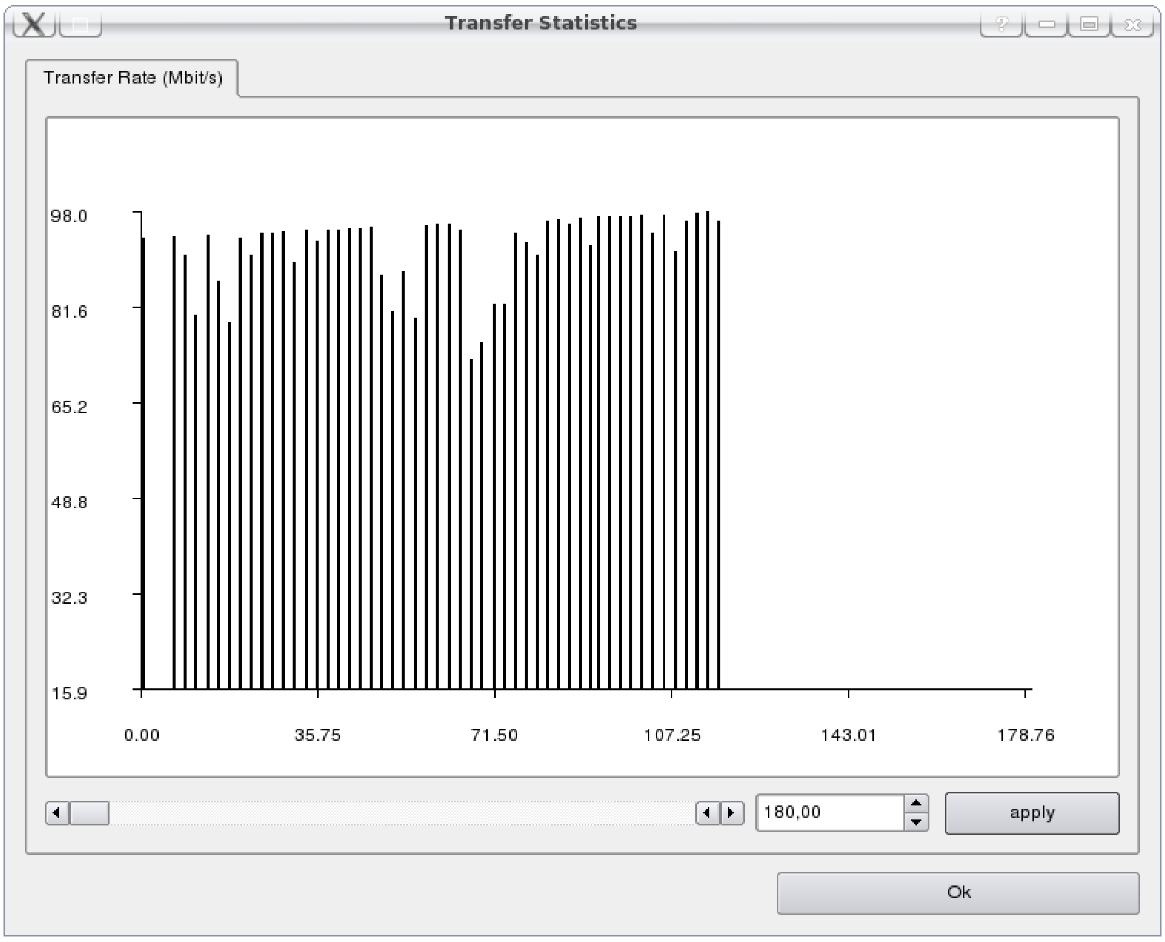


Figure 13: Transfer statistics displayed in the KoDaVis client.

To get more insight in the data transfer process and potential bottlenecks additionally the corresponding transfer times over the same link can be displayed. The transfer times displayed in the UNICORE client plug-in exist to bring various connection performances in relation to each other. That will be explained in the following.

When multiple clients are connected, great differences in the transfer times of the curves indicate differences in the performances of the individual server-to-client links. Those connections posing a bottleneck can be identified and if need be the participant informed about this or removed from the session. For each time step of the visualization, a data set is transferred to the clients. In the diagram, you can see the time for each of these data transfers. A detailed discussion usage and interpretation of these statistics may be found in report D3.6.

7.2.4 Summary of the achievements

Testbed experiments performed

- via Internet
- via testbed between FZJ and PSNC, tuned for performance
- performance monitoring helped evaluating testbed experiments during the second phase of experiments

Development

- parallel data server, in alpha state
- performance monitoring

Deployment

- using new computational infrastructure at FZJ

The features developed for KoDaVis during the first 24 months of the PHOSPHORUS project, mainly performance monitoring, can be used to better identify problems in a collaborative visualization session. For collaboration data packets, the application is able to make use of the minimum latency offered by the network.

7.3 TOPS

TOPS (Technology for Optical Pixel Streaming) enables remote viewing of large scientific datasets (2D or 3D) on high-resolution display devices (Tiled Panel Displays). TOPS streams these pixels, uncompressed, from the data center over the network to remote displays.

TOPS data center and the rendering machines are located in Amsterdam, the resulting picture is streamed over the network to Sankt Augustin and be viewed on the i-CONE™ display of FHG IAIS. At the display site the scientist interacts (navigates through the data set in real-time) with the application that runs at the rendering site. The i-CONE™ is a cylindrical 270-degree projection display system with high-resolution and evenly curved projection surfaces. The i-CONE™ has a visitor capacity of approximately 20 people. The display consists of four projectors with a resolution of 1600x1460 pixel at vertical refresh rate of at 105 Hz each. The projectors support active stereo mode which means that for each eye half of the refresh rate is available. The input for the projectors is provided by a cluster of four workstations equipped with NVIDIA Quadro FX graphics cards and a gigabit network interface each.

In the final phase of the Test bed experiments, SARA and FHG tested TOPS over the connection SARA-PhosphorusSwitch-Geant2+-Viola-FhG. On this connection a dedicated 1 Gb lightpath is available for TOPS. Figure 14 shows a picture of the data seen at the display site, the network configuration used for TOPS is described in Figure 15. As was indicated in deliverable D3.4, modifications were required to adapt TOPS to the I-Cone display at FHG. Since then, configuration of TOPS has been parameterized to allow for arbitrary display formats.

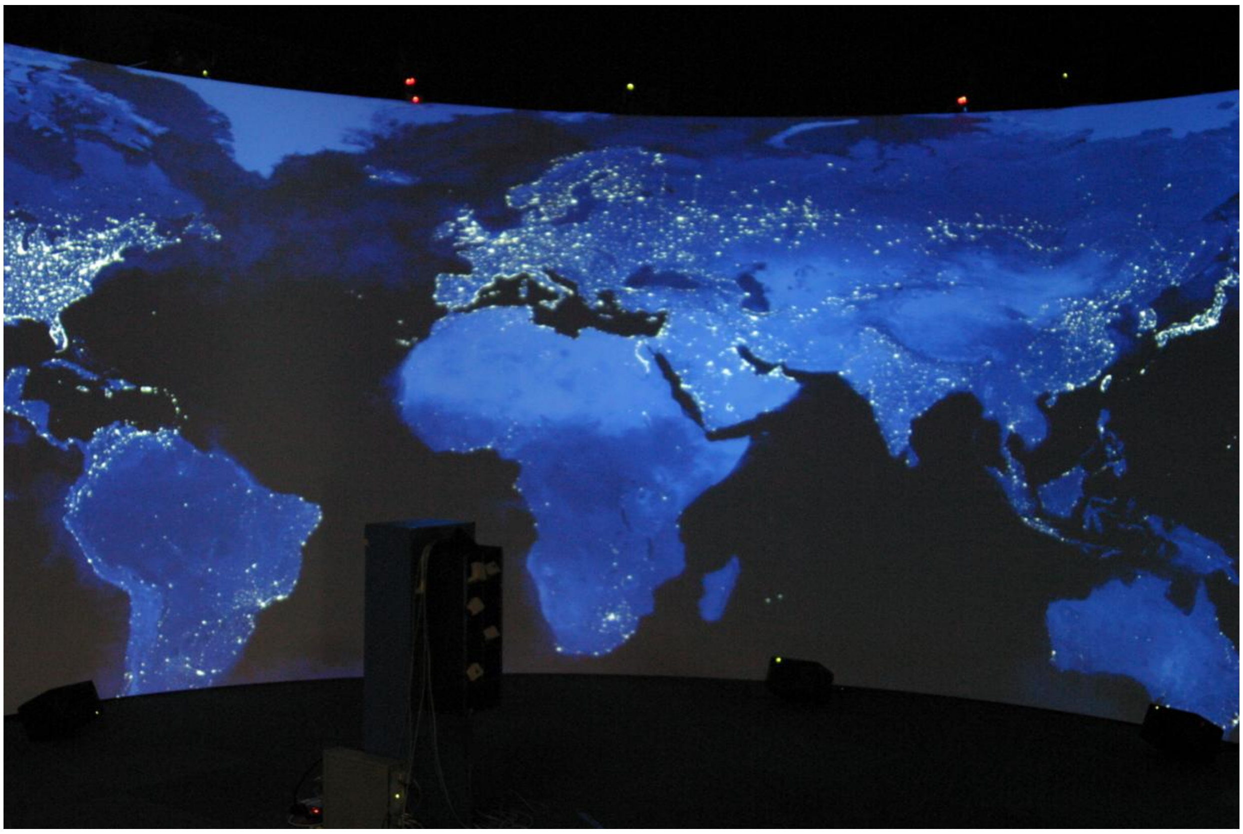


Figure 14: Data rendered at SARA Amsterdam, seen and controlled at FHG IAIS Sankt Augustin

Monitoring of the performance of TOPS was done on the SURFnet provided Phosphorus Test bed switch, located at SARA and connected to Geant2+ on one side and to number of Phosphorus partners via NetherLight. The SURFnet NOC, executed by SARA, controls and monitors the Phosphors switch.

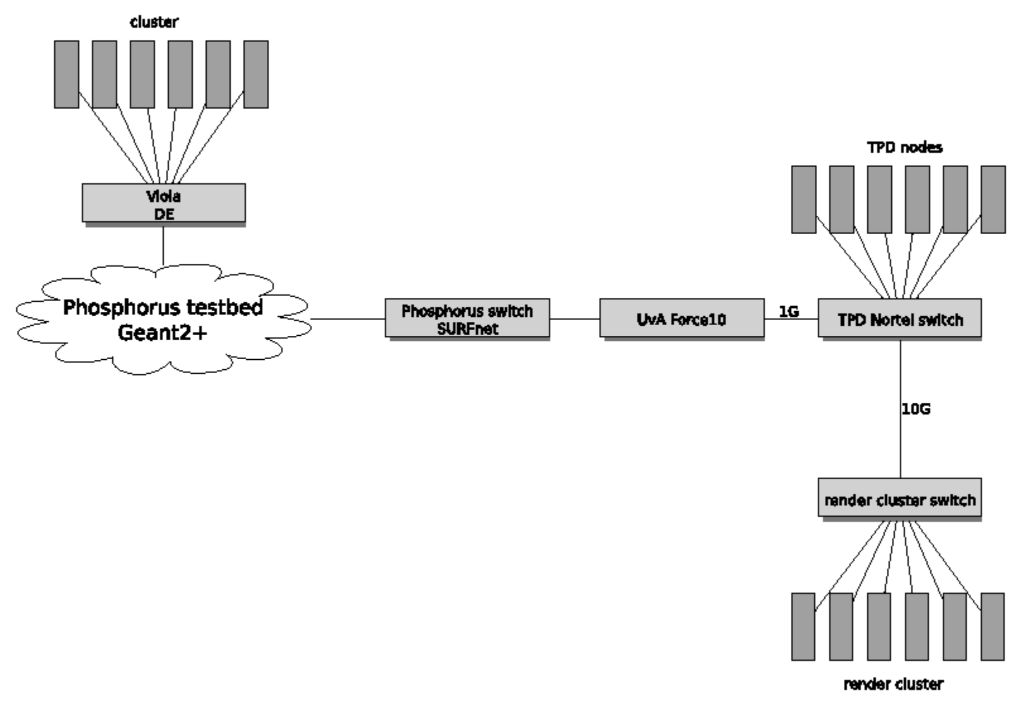


Figure 15: TOPS network configuration between SARA and FHG using the PHOSPHORUS testbed

The experiments conducted proved that TOPS is able to use the complete bandwidth that was made available for the experiments during the whole timeframe (see D3.6 for details). This indicates that the use of pan-European, multi-domain lightpaths will allow for continuous availability of required bandwidth and quality of service for high performance applications.

Although latency was not measured quantitatively, the tests demonstrated that with the available bandwidth and the configuration of the lightpaths, the latency is minimal and very acceptable for the end-user application.

7.3.1 Summary of the achievements

Further improvement of the TOPS approach

- resource allocation through central scheduling and configuration user interface

- complete separation of render and display code
- implementation as daemon invoking the actual display or render code

- enhanced flexibility through parameterization of
    - display size & bits per pixel
    - display configuration
    - MTU size
    - IP configuration
    - data set location
    - user interface location
- hardcoded values removed and put in configuration files

7.4 DDSS

Experiments with DDSS application were conducted to examine how the Phosphorus link reservation features support the efficient operation of this application. The experiments have been launched on a test bed spanning four centers: PSNC, Fraunhofer, FZJ and Essex. The centres were connected to each other with VLANs configured on the Phosphorus infrastructure.

General description of the tests has been provided the report D3.4, a more precise description is contained in document [D3.6].

7.4.1 Description of experiments

Two kinds of applications have been used in DDSS experiments. Testing of each application has been split into two scenarios:

- tests with big files,
- tests with lots of small files.

In case of DDSS GridFTP tests, in each of the phases values of two parameters have been modified, both having the influence on the data transfer speed:

- number of threads (parallel data streams),
- block size.

In case of DDSS B/A tests, in each of the phases two parameters have been modified:

- TCPW (TCP Window Size),

- TCPBUFSIZE (TCP Buffer Size).

One of the results presented in D3.6 acquired from performance vs. number of threads test scenarios run for big files show that the PHOSPHORUS test network is able to carry data traffic over a long distance links much more effectively than the regular Internet links.

The big file transmission scenarios are mainly bandwidth-dependent. Therefore, we may conclude that PHOSPHORUS bandwidth reservation features are very effective. Comparison to regular Internet links, clearly shows the advantage of the optical links, reserved through a PHOSPHORUS control plane.

The Small files transmission experiments have shown, that the reservation of the optical links using a PHOSPHORUS NRPS, allow high average transmission speed, comparing to the public Internet setup. Reserving the bandwidth using PHOSPHORUS NRPS helped to improve the transmission efficiency significantly.

Finally, as a real-world use-case Backup/Archive tests were performed and evaluated. The DDSS Backup/Archive tests result confirmed that the link reservation feature provided by PHOSPHORUS network make this network suitable for bandwidth-consuming applications.

The DDSS GridFTP tests results showed that the PHOSPHORUS optical network links reservation features make significant improvement of the performance of large data transmission possible. The results of the big files transmission indicate that the PHOSPHORUS reservation features can provide a high transmission bandwidth to the data transfer application. From this part of the testing scenarios, we could also show that using a high level of parallelism one may overcome the transmission limits caused by high latency in both PHOSPHORUS and Internet setup.

Similarly, the DDSS Backup/Archive tests results showed, that reservation of the PHOSPHORUS links helps to gain good performance for massive data transmissions. The numbers acquired in big files backup scenarios are comparable to results of reference tests performed in the local setup, over the LAN network in PSNC. This confirms the efficiency of bandwidth reservation functionality provided by PHOSPHORUS network.

7.4.2 Summary of the achievements

GridFTP in the testbed:

- scenarios between PSNC and FZJ, Fraunhofer, Essex,
- transfer of multiple small files (delay-dependent) vs a few large files (bandwidth-dependent),
- various number of parallel streams, various transfer block size,
- all scenarios run over the PHOSPHORUS links,
- the new results collected and compared with prior (tests run over the Internet) results.

Backup/Archive application:

- Backup/Archive servers installed and running in PSNC and FZJ, clients on both sides:
- Backups of multiple small files (delay-dependent) vs a few large files (bandwidth-dependent),
  - evaluation of various transmission parameters – compression, encryption enabled/disabled.
- Application binary wrapped to exploit NRPS (link reservation).

Overall:

- Scenarios run periodically (automatically, driven by scripts),
- many tests performed with the NRPS link reservation system,
- conclusions drawn up and reported in D3.6.

8 Conclusions

This report highlights the results and achievements of middleware components and applications during the first 26 months of the PHOSPHORUS project. The implementations of both middleware components and applications have been validated with various experiments as described in the last reports D3.6 - Report on the Results of the Application Experiments During the Final Testbed Experiments and D3.7 - Report on the results of the middleware experiments during the final testbed experiments.

This report summarizes the results of the testbed experiments of middleware and applications. Details have been included where new results have been achieved since the publication of the last reports, otherwise we refer to previous reports.

The results of the experiments show that the improvements of the middleware towards automated resource selection through the Resource Selection Service deliver the expected behaviour. Now a user only needs to specify the application that should be executed and the framework automatically selects the appropriate resources, starts the negotiation on availability, and does the necessary reservations for the user.

With the interface between G^2MPLS and the middleware it could be shown that the resource selection may already start at the network layer based on the information on topology, connectivity and link status available there. The relevant information is then passed to the middleware layer and presented to the user for the final decision.

Finally, the INCA experiments showed in simulations the scalability and performance for a network environment with a much larger number of nodes and links than available in the PHOSPHORUS network.

The experiments also served as a preparation of the demonstration and training events organised during the Supercomputing Conference in Austin and the ICT Conference in Lyon in November 2008.

9 References

[G2MPLS]	"Deployment and Interoperability of the Phosphorus Grid Enabled GMPLS (G2MPLS)" , DOI 10.1109/CCGRID.2008.120
[D2.5]	D2.5 - Preliminary Grid-GMPLS Control Plane prototype, http://www.phosphorus.pl/wiki/images/0/00/Phosphorus-WP2-D2.5v1.0.doc
[D3.6]	D3.6 Report on the Results of the Application Experiments During the Final Testbed Experiments, http://www.phosphorus.pl/wiki/images/3/3e/Phosphorus-D3.6.pdf
[D3.7]	D3.7 Report on the results of the middleware experiments during the final testbed experiment http://www.phosphorus.pl/wiki/images/3/3e/Phosphorus-D3.7.pdf
[GLUE]	GLUE Working Group (GLUE), http://forge.gridforum.org/sf/projects/glue-wg
[GRRService]	WSDL file, http://packcs-e0.scai.fraunhofer.de/phosphorus-wp2/services/GridResourceRegistryService?wsdl
[OGSA-BES]	OGSA-BES Working Group, http://forge.gridforum.org/sf/go/projects.ogsa-bes-wg/ http://www.scai.fraunhofer.de/index.php?id=2384&L=1

10 Acronyms

AAA	Authentication, Authorisation, Accounting
DDSS	Distributed Data Storage Systems
e2e	end to end
EGEE	Enabling Grids for E-sciencE (European Grid Project)
FC	Fibre Channel
FC-SATA	Fibre Channel to SATA technology (mixed technology used in disk matrices: disk matrix have Fibre Channel ports for hosts connectivity, but contains SATA disk drives)
GEANT2	Pan-European Gigabit Research Network
GEANT+	the point-to-point service in GEANT2
GMPLS	Generalized MPLS (MultiProtocol Label Switching)
G^{2}MPLS	Grid-GMPLS (enhancements to GMPLS for Grid support)
G.O-UNI	Grid Optical User Network Interface (integrating optical networks with grid services)
GT4	Globus Toolkit Version 4 (Web-Service based)
INCA	
KoDaVis	Tool for Distributed Collaborative Visualisation
MSS	MetaScheduling Service (a Grid Level Scheduler developed at the Fraunhofer Institute SCAI)
NREN	National Research and Education Network
NRMS	Network Resource Management System
NRPS	Network Resource Provisioning System
PoP	Point of Presence
Protégé	Ontology Editor and Knowledge Acquisition System
QoS	Quality of Service
SNMP	Simple Network Management Protocol
TOPS	Technology for Optical Pixel-Streaming
TPD	Tiled Panel Display
TUAM	Tool for Universal Annotation and Mediation
UNI	User to Network Interface
UNICORE	European Grid Middleware (UNIiform Access to COmpute REsources)
VLAN	Virtual LAN (as specified in IEEE 802.1p)
VIOLA	A German project funded by the German Federal Ministry of Education and Research (Vertically Integrated Optical Testbed for Large Applications in DFN)

VPN	Virtual Private Network
WISDOM	Wide In Silicio Dockong On Malaria