034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number D3.4

Report on middleware extensions and implementation

Due date of deliverable: 2008-02-29
Actual submission date: 29/02/2008
Document code: <Phosphorus-WP3-D3.4>

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
FHG

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)	

Abstract

This report describes the requirements derived from the first results of the testbed experiments and the planned further optimisations of middleware during the second phase of the project as defined for the milestone M3.4.

Project:	Phosphorus
Deliverable Number:	<D.3.4>
Date of Issue:	29/02/2008
EC Contract No.:	034115
Document Code:	<Phosphorus-WP3-D3.4>

Table of Contents

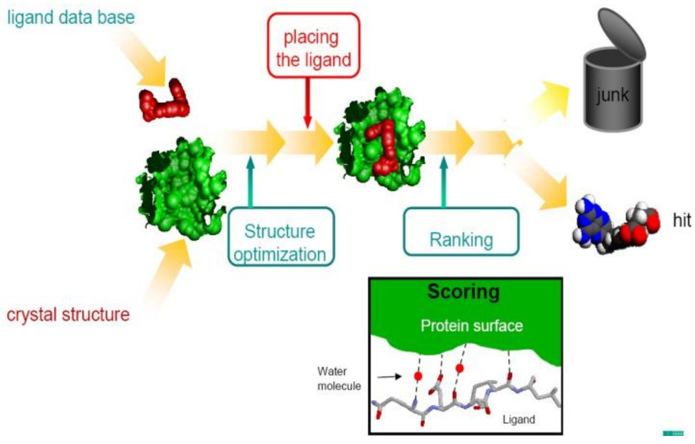
Table of Figures

0 Executive Summary

This report describes the requirements derived from the first results of the testbed experiments and the planned further optimisations of the PHOSPHORUS middleware during the second phase of the project.

The first testbed experiments comprise the following activities

- WISDOM_
  The acronym WISDOM, for Wide In Silico Docking On Malaria, comes from the first screening experiment at a large scale against malaria in EGEE [WISDOM-EGEE] and is now used as a generic name for an initiative, that aims to demonstrate the relevance and the impact of the grid approach to address drug discovery for neglected and emergent diseases. The PHOSPHORUS use case WISDOM consists of the same virtual screening techniques, including two of the most popular protein docking simulation tools, AutoDock and FlexX.



  Both are in silico docking techniques to compute the probability that potential drugs will dock with a target protein [WISDOM2007]. This allows researchers to rule out the vast majority of potential drugs, so that they can concentrate on the most promising compounds in wet-laboratory tests.

  One of our experiences made during the EGEE WISDOM data challenge was, that 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 by this we will concentrate in PHOSPHORUS besides the test of network and middleware functionalities in the implementation of a perfect workflow for WISDOM. Thus WISDOM will be mainly used to implement and test UNICORE 6-enabled workflow jobs with stage-in, execution (run) and stage-out phases.

- KoDaVis activities during the first period of testbed experiments comprised
    - A significant extension of KoDaVis.
        - The main design change of the KoDaVis software was to enhance it with respect to resource reservation and access control by adapting it to the Grid middleware UNICORE
    - A VISIT/KoDaVis gridbean for the UNICORE GPE application client has been developed.
    - The VTK/Qt based visualisation client has been extended to receive the parameters required to establish the communication to the server components from the UNICORE client.
    - An additional Web Services Resource Framework (WSRF) based VISIT/KoDaVis Grid service (ADS) for the UNICORE server has been developed.
    - The data- and collaboration-servers have been extended to support a common XML-based control protocol.
- TOPS activities during the first testbed experiments comprised the following
    - Adaptation of TOPS software for Fraunhofer display variables
    - Adaptation of content for display on Fraunhofer displays
    - Installation of TOPS software on display machines at Fraunhofer
    - Installation of network monitoring tools on display machines at Fraunhofer
    - Debugging of network issues
    - Accomplishing multiple test runs over time and monitoring of network performance
- DDSS activities during the first testbed experiments period include following actions:
    - Performing multiple transmission tests between various location pairs and with various transmission parameters (number of transmission threads, size of transmission data block and number and size of file(s) to be transmitted). Tests were performed using both DDSS applications: GridFTP and TSM Backup/Archive.
    - Development and usage of test automation tool and test results collection and visualisation tool
    - Collection and analysis of test results.
    - Dealing with network and firewall issues.
- INCA activities during the first testbed period include:

  - INCA made useful observations through the testbed experiments 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

Based on the experience made and results obtained from the experiments the following topics for optimisation of the PHOSPHORUS middleware have been identified

- Workflow support in UNICORE
- UNICORE remote site registry
- Support for applications as UNICORE resources
- Extensions for the UNICORE client (plugins)
  - allowing to specify parameters of an application controlling input and output
  - allowing to specify the number of sites and the number of nodes to be used for an application at a site
  - Globus Toolkit:
    - CPU, link bandwidth and latency co-allocation,
    - On-demand firewall passes creation between DDSS GridFTP data transmission parties.
- Improvements of the INCA Data Movement Layer and Storage Protocol Layer

1 Experience, results and requirements

1.1 WISDOM

Several modifications are needed to enable parallel WISDOM runs on the Phosphorus testbed. In the following activities are described for both WISDOM codes, AutoDock and FlexX. Whereas AutoDock is based on the genetic docking algorithm, FlexX analyses the interaction of a target protein with a ligand:

Specifics for AutoDock:

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. AutoDock has applications in:

- X-ray crystallography;
- structure-based drug design;
- lead optimization;
- virtual screening (HTS);
- combinatorial library design;
- protein-protein docking;
- chemical mechanism studies.

The original AutoDock program is in itself inefficient to be used in virtual screening because the grids of interaction energy have to be calculated for each putative ligand in chemical database. However, the automation of the AutoDock program with the potential grids defined in common for all putative ligands leads to more than twofold increase in the speed of virtual database screening. The utility of the automated AutoDock in virtual screening is further demonstrated by identifying the actual inhibitors of various target enzymes in chemical databases with accuracy higher than the other docking tools including DOCK and FlexX.

More information at: Morris, G. M.; Goodsell, D. S.; Huey, R.; Hart, W. E.; Halliday, R. S.; Belew, R. K.; Olson, A. J. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. J Comp Chem 1998, 19, 1639- 1662.

AutoDock was installed without any problems on all participating sites: UESSEX (Wisdom node), Fraunhofer (PACK), SC Poznan, JSC Juelich (Cray-XD1). After the the Cray-XD1 system at JSC Jüich was becoming unavailable due to an unrecoverable hardware failure, a new installation on the replacement system (Juggle) has to be implemented.

Specifics for FlexX:

FlexX is an extremely fast, robust, and highly configurable (FlexX-able) computer program for predicting protein-ligand interactions. Its main applications are

- Binding mode prediction;
- Virtual high-throughput screening (vHTS).

More information at: Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. A fast flexible docking method using an incremental construction algorithm. J Mol Biol 1996, 261, 470–489.

Code owner BioSolveIT is contacted to get a FlexX license for the project runtime. BioSolveIT agreed to support the Phosphorus project with a FlexX license. To handle this license a FlexNet server is installed at Fraunhofer site. For communication and license check special specification and configurations are needed (e.g. open ports to communicate with the license server), which are provided by Fraunhofer.

FlexX was installed on all participating sites: UESSEX (Wisdom node), JSC Juelich (Cray-XD1), Fraunhofer (PACK cluster). After the the Cray-XD1 system at JSC Jüich was becoming unavailable due to an unrecoverable hardware failure, a new installation on the replacement system (Juggle PC Cluster) has to be implemented. Unfortunately no FlexX installation was possible for the platform of SC Poznan (Itanium cluster), because this hardware is up to now not supported.

General actions for AutoDock and FlexX:

INPUT
One of the first actions after the installation of both codes was to use the same input data. This is needed to combine results and get more insights using these different codes. Thus the same data set was individually preprocessed to be used as input for both codes. AutoDock uses Grid maps and DPF as input file format and FlexX uses RDF and Mol2 files.

For the first tests the BioSolveit flexx-200 test data suite, a subset from the PDB, was used for both the applications AutoDock and FlexX. The dataset consists of 200 protein-ligand complexes. For each test case, an rdf file and an active site file for the protein as well as mol2 files containing the ligand in crystal orientation (protonated and unprotonated) and in an energy minimised conformation (protonated) are provided. PDB files are collected for those cases where the PDB file used differ from the PDB standard file.

EXECUTION AND OUTPUT
An experience made during the huge EGEE WISDOM data challenge was, that 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 by this we will concentrate in PHOSPHORUS besides the

test of network and middleware functionalities in the implementation of a perfect workflow for WISDOM. Thus WISDOM will be mainly used to implement and test UNICORE 6-enabled workflow jobs with stage-in, execution (run) and stage-out phases.

In the following graphics both (very similar) workflows incl. pre- and post processing for AutoDock and FlexX are described. The pre-processing includes the preparation of target- and ligand descriptions in the specific data format. Post-processing will be much easier, if the results of the docking processes are collected on one specific server.

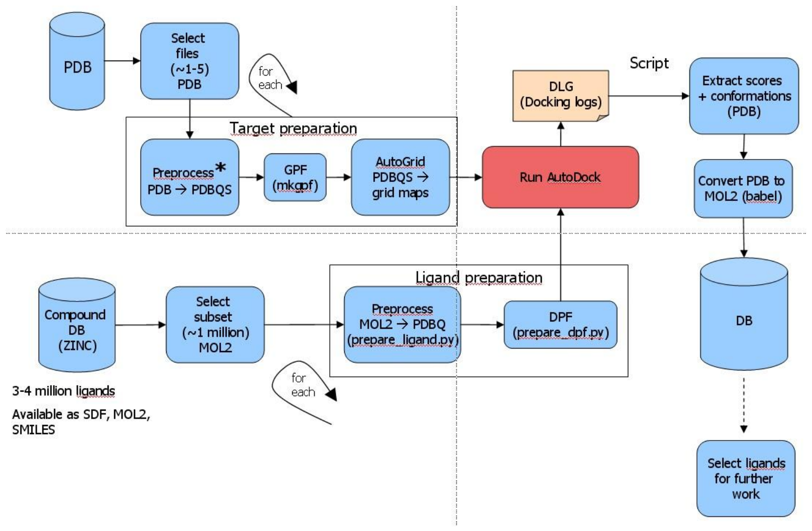

PDB
Select files (~1-5) PDB
for each
Target preparation
Preprocess* PDB → PDBQS
GPF (mkgpf)
AutoGrid PDBQS → grid maps
Run AutoDock
DLG (Docking logs)
Script
Extract scores + conformations (PDB)
Convert PDB to MOL2 (babel)
DB
Select ligands for further work
Compound DB (ZINC)
3-4 million ligands
Available as SDF, MOL2, SMILES
Select subset (~1 million) MOL2
for each
Ligand preparation
Preprocess MOL2 → PDBQ (prepare_ligand.py)
DPF (prepare_dpf.py)

Figure 1: Setup of the typical workflow of AutoDock

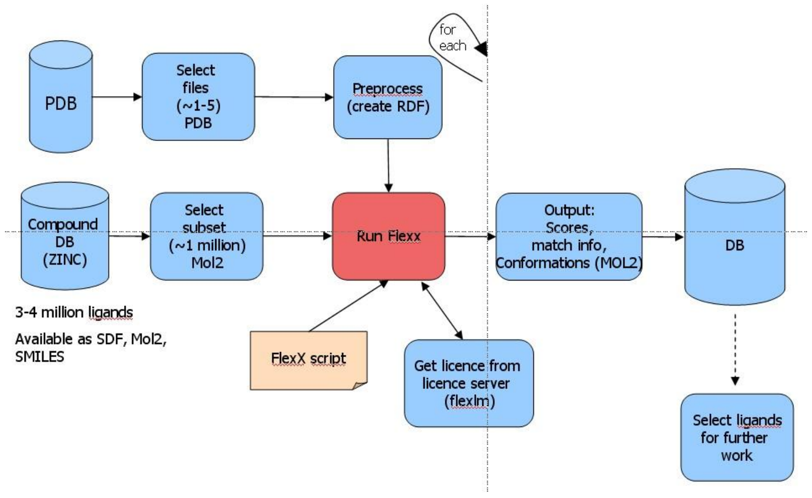

PDB
Select files (~1-5) PDB
Preprocess (create RDF)
for each
Compound DB (ZINC)
Select subset (~1 million) Mol2
Run Flexx
Output: Scores, match info, Conformations (MOL2)
DB
3-4 million ligands
Available as SDF, Mol2, SMILES
FlexX script
Get licence from licence server (flexlm)
Select ligands for further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

Figure 2: Setup of the typical workflow of FlexX

Like the input data, the output data format of the docking codes AutoDock and FlexX differ. AutoDock results are produced in DLG and Mol2 files format and FlexX generates output in Mol2 file format. The WISDOM stage-out phase includes the transfer of these local output data files after termination to a specified output server.

Currently the workflow engine of UNICORE 6 is only available in a prototype status. Thus the execution of both codes is actually done script-based. These scripts can be executed based on reservations of the available local resource management system and with the UNICORE 6 application client. Two different scripts are produced for the execution of AutoDock and FlexX jobs, which uses the schedule file of the RMS for the execution. The Wisdom jobs are executed by 'ssh calls' on the reserved computing nodes. On most sites the file system is mounted to all computing nodes and therefore is identically on all nodes. To avoid writing to the same output files, the naming of the output files uses the hostname of the node. Thus output-data files are produced based on the schedule file in a node-specific way. This naming convention is also needed to transfer all results back to one single node, the so-called output server node. It is intended to offer further post processing functionalities to focus on relevant results only.

Experiences and requirements

- The shell scripts are programmed and tested. These scripts perform parallel executions of AutoDOCK and FlexX. These executions are run on the participating sites UESSEX (Wisdom node), JSC Jülich (Cray-XD1), Fraunhofer (PACK cluster) without any problems. A renewed installation at JSC Jülich (cluster juggle) is needed.

- AutoDock and FlexX code availability has to be checked by the RMS.

- The FlexX License of the user has to be checked by the RMS. It has to been avoided, that FlexX executions without license are possible on the Phosphorus test bed.

- A remote registry for participating sites, which enables easily executions via the UNICORE 6 application client on all sites (XML file with URLs of participating sites) would be welcome.

- A UNICORE 6 client plugin with a GUI for the specification of input and output data location is needed and the specification of execution parameters would increase user friendliness significantly:

    - Before the data distribution can be done, a specification of the location of the input data is needed (site, directory, node (if the HOME-DIR is not mounted to all nodes))

    - AutoDock DPF execution parameter specifications (ga_num_evals def=250.000, ga_pop_size (50-200) def=50, ga_run def=10)

        - FlexX start script execution parameter specifications (place_particles def=0, max_overlap_vol (0-100) def=2.5)

- After the execution all output data should be transferred for further processing and analysis to one specific location (site, directory, node (if the HOME-DIR is not mounted to all nodes)).

- A UNICORE 6 client plugin for execution specifications like 'number of sites' and 'number of nodes for each site' for WISDOM jobs is a prerequisite for the implementation of the workload distribution.

- WISDOM executions would benefit from a stable UNICORE 6 workflow engine.

1.2 KoDaVis

The KoDaVis testbed experiments comprehend so far functionality tests of both application use-cases described in Deliverable D3.1:

- a single scientist, performing interactive visual analysis of remote data, and

- scientists at two or more different sites, collaboratively exploring the data

1.2.1 Components in the testbed

Originally, the software consisted of the following components:

- visualization applications,

- a data-server that distributes fragments of data selected by the clients,

- a collaboration server that synchronizes the clients, and

- an optional control GUI that monitors client activity and interacts with the ARGON system to handle immediate network connection requests.

All of these parts had to be started separately, that is, the users had to agree on session parameters allowing to connect to each other, had to log into the machines that were involved and start the server and client components manually. Afterwards, they had to establish the desired connections between the components.

In order to support the use-cases sketched above, the pre-existing version of KoDaVis has been extended significantly. The main design change of the KoDaVis software was to enhance it with respect to resource reservation and access control by adapting it to the Grid middleware UNICORE as outlined in Figure 1. These newly developed or adapted components, coloured green in this figure, have been tested within the KoDaVis testbed experiments:

- A VISIT/KoDaVis gridbean for the UNICORE GPE application client has been developed. This gridbean offers a seamless and intuitive interface for specification of the above mentioned resources and control of the data-server. This component also communicates with the Meta Scheduling Service to manage network connections. Furthermore, it is possible to administer users participating in the particular visualization session and deploy certain privileges among them.

- The VTK/Qt based visualisation client has been extended to receive the parameters required to establish the communication to the server components from the UNICORE client.

- An additional Web Services Resource Framework (WSRF) based VISIT/KoDaVis Grid service (ADS) for the UNICORE server has been developed. This service starts the data- and collaboration-server Web-Service. It consists of the actual data- and collaboration-server and and a wrapper providing the Web
  -Services interfaces to the outside world. Furthermore, it manages all exchange of status information and commands between the UNICORE clients and the data and collaboration servers.

- The data- and collaboration-servers have been extended to support a common XML-based control protocol. This protocol is used to communicate with the Web-Service wrappers and allows them to be managed from the UNIOCRE client.

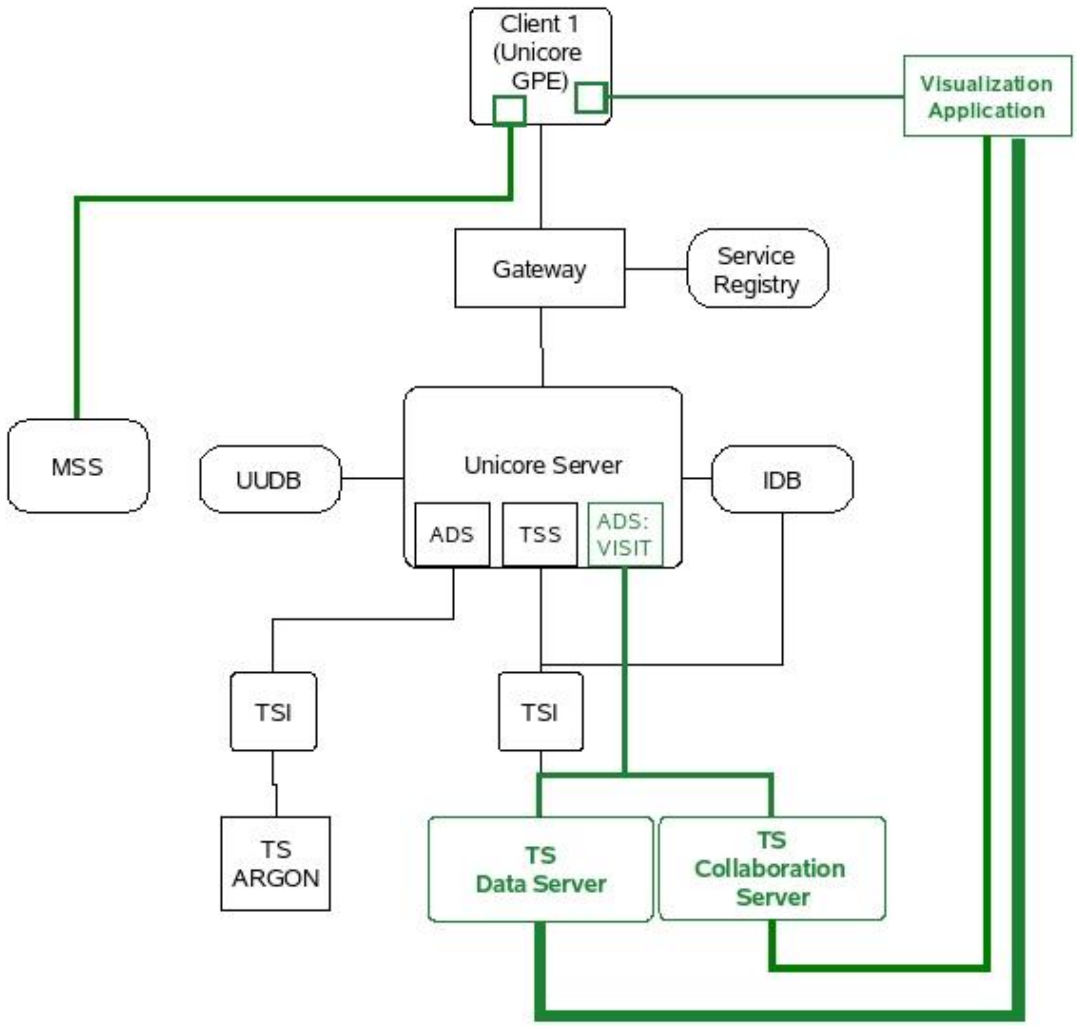
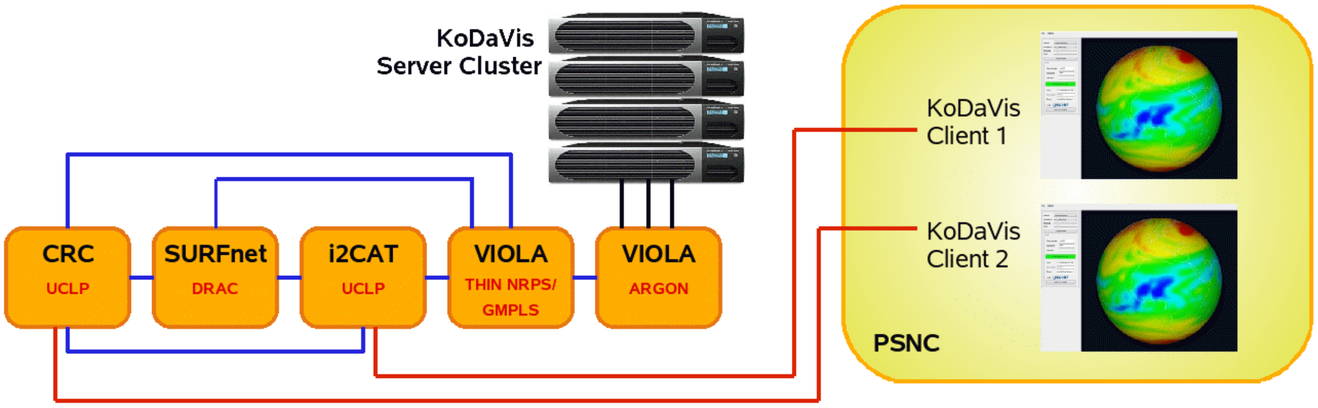


Figure 3: Setup Design of the components used for the KoDaVis application experiments. Green Components have been developed in Phosphorus and have been tested successfully within the experiments.

1.2.2 Setup in the testbed

The experimental setup comprised of the KoDaVis data- and collaboration server, which had been installed on the Cray-XD1 machine in Juelich, and the visualization clients located in Poznan, Essex and Juelich. Firstly, the communication between the servers and the visualisation clients was successfully tested via ssh-tunnel over the internet. This test was conducted by starting all parts of the application separately by hand, without using the UNICORE software. As a next step, the Phosphorus testbed was used to interconnect the KoDaVis data-



and collaboration server with the clients in Juelich and in Poznan. Initially, a static link was established between the servers and the clients. Then, the reservation systems NRPS, Argon, UCLP, DRAC, Thin-NRPS and GMPLS were applied to establish a connection, using the AJAX Demonstrator web interface as outlined in Figure 2. For this test, two visualization clients were run in Poznan, one of which was routed via CRC, Canada, and the other one via i2CAT, Spain to Juelich. Due to the huge distance between Canada and Juelich or Poznan, this setup was not feasible for a collaborative visualization, as the long latencies of the synchronization messages sent to and from the collaboration server inhibited a fluent handling of the visualization.

Figure 4: Setup Setup of a KoDaVis experiment, connecting two visualization clients at PSNC over different routes to the KoDaVis Server cluster in Jülich.

As a next step, the UNICORE client was deployed in order to start both the data- and collaboration server in Jülich and the visualisation clients in Poznan and Jülich, and to establish a connection between them. Moreover, the client was enabled to make advanced reservations of bandwidth between the participating sites.

The experiments showed that all components are functional and are correctly interacting with each other, providing the intended functionality and ease of use. The successful integration and deployment of this first version of middleware and application has also been demonstrated at the Phosphorus Review meeting in Poznan in December 2007.

A lesson learned from the testbed experiments is that due to the many distributed components involved, it is difficult for less experienced users to identify the cause of errors that may occur. Moreover, in collaborative mode, it is currently difficult to find out which visualisation client is responsible for a potentially poor performance of the whole system. A suitable performance analysis tool remains to be developed. Furthermore, it could be verified, that the KoDaVis data server, which was only used in serial mode so far, constitutes a bottleneck in the current system. Therefore, the performance might be enhanced by deploying the parallel data server. This will be tested in the continuative performance experiments.

1.3 TOPS

1.3.1 Experience, results and requirements

In the first network tests, we found a few problems: jumbo frames did not arrive, indicating that a node in the network path was not jumbo-enabled. Later, the network was at some point suddenly down. These problems partly were due to the prototype network used for testing. Some connections have been realized on existing connections and infrastructure set up within other research projects but not explicitly dedicated to the Phosphorus project. The lesson learned from this, is that it is very important to allocate enough resources for maintenance and monitoring of the network path.

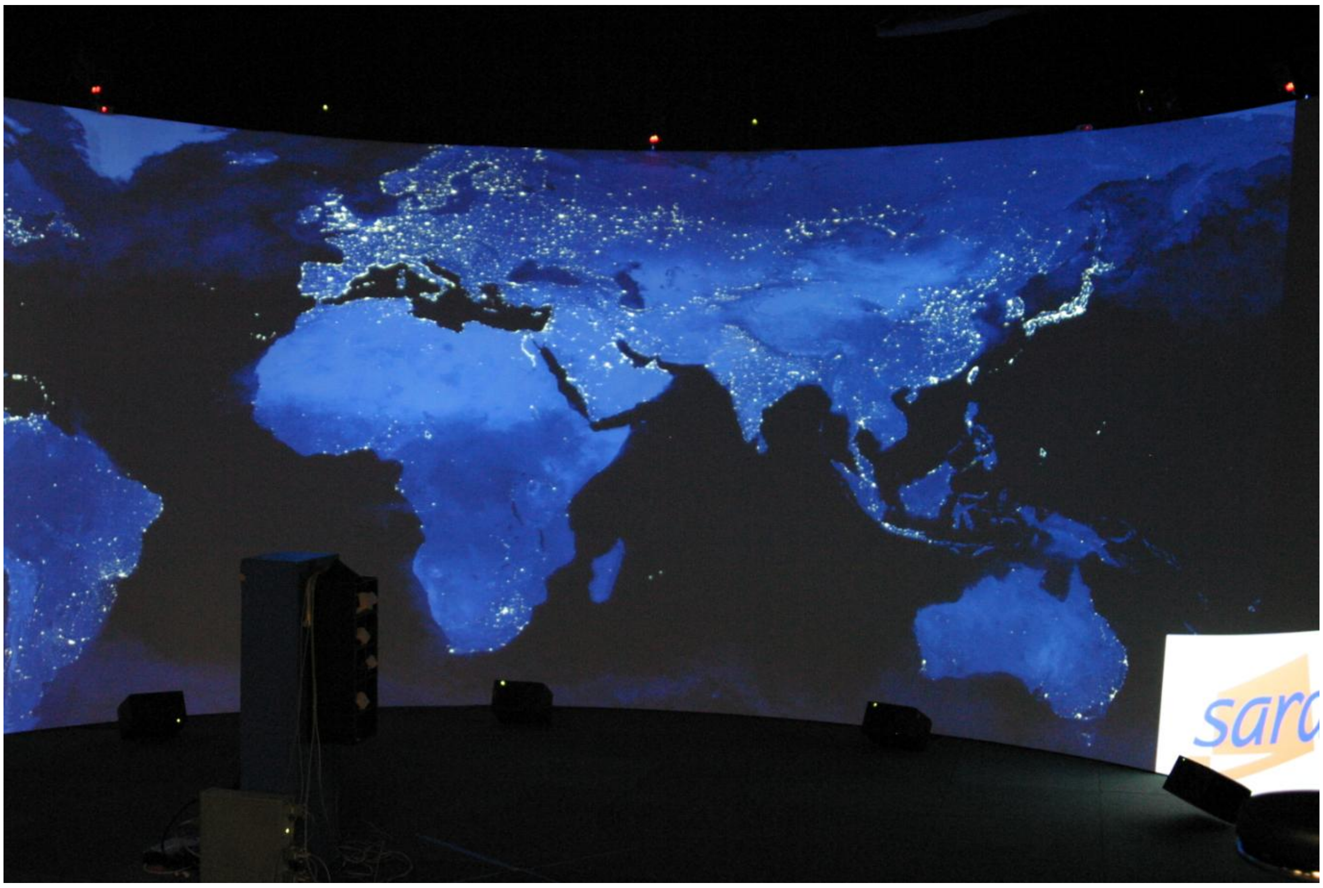


Figure 5: Test of TOPs, View at i-cone Display of FhG

The TOPS software was originally written for use on 1600x1200 displays. This does not match the vision of using the software on arbitrary displays, so the software will need to be adapted to allow parameterization of display variables (X, Y, bits per pixel, number of displays). To be able to perform the tests, we needed to adapt the current TOPS version for the display parameters of the Fraunhofer display (1600 x 1456, 24 bits per pixel, 4 displays). This has given us a chance to verify that arbitrary display sizes are possible, with the exception that the system needs displays with both horizontal and vertical size multiples of 16. This requirement is due to the way that tiled TIFFs are stored internally, and cannot be circumvented.

Moreover, the software does not have a 'grid-aware' remote invocation scheme. These problems partly were due to the prototype network used for testing. Some connections have been realized on existing connections and infrastructure set up within other research projects but not explicitly dedicated to the Phosphorus project. As the vision is to have a central scheduling system, a remote invocation scheme is a very important issue. Currently, the system relies on the user starting a central process that handles user input, remotely starts the display programs and remotely starts the rendering programs. This only works if the user has login access to all the rendering and display resources, a situation that is not congruent with the idea of grid-based computing/networking. A redesign needs to be made that provides a display service and a rendering service, running on available display systems and available render systems, respectively. A central web-based service will provide the interface for researchers to schedule sessions on these resources. When a session needs to be started, the central service sends a 'go' signal to the requested display server and the requested rendering server, which in turn starts the appropriate applications for use by the scientist. When the timeslot has ended, the central service may also stop the running service, to free the resources for other use.

1.4 DDSS

DDSS software is used to transport, exchange, share, store, backup/archive and restore data in scientific and commercial applications. Test scenarios run in the PHOSPHORUS test-bed include: data transfers performed with use of the GridFTP application [GridFTP] and backup/archive/restore operations made by the backup/archive application [TSMBA]. In the former scenario GridFTP server stores or retrieves the data from/to single (one-to-one setup) or multiple clients (many-to-one setup). The single client collaborates with many servers at the same time (one-to-many setup). Data transmission between client-server pair can be done over 1-128 parallel streams. In the latter scenario, the end-user data are collected on client machines by a B/A client and sent through TCP/IP streams to B/A server module. B/A server manages storage pools that can include disks, tapes and other storage media and stores user data in these pools, according to a policy defined for a user. Setup used in the test-bed contains a single B/A server and multiple B/A clients (many-to-one).

1.4.1 Experiments

DDSS applications were run in following local test-beds and among them: PSNC, UESSEX and VIOLA. Test scenarios run in test-beds included one-to-one DDSS Grid FTP transfers (one client, one server) as well as one-to-one DDSS Backup/Archive application scenarios. DDSS GridFTP has been run in PSNC (both GridFTP servers and clients in four HPC cluster nodes), UESSEX (clients and servers in cluster nodes), and in both VIOLA sites: FZJ and Fraunhofer. DDSS Backup/Archive application was run in Poznan (dedicated storage server acting as the B/A server and HPC cluster nodes acting as clients) and Jülich (dedicated machine acting as client or server, depending on the scenario). Tests were run periodically, each two hours. Each round of tests included data transfers with various parameters: number of transmission threads, size of transmission data block and number and size of file(s) to be transmitted (numerous small files versus some large files). Test results were automatically put to the website in a numeric and visual form (gnuplot graphs).

1.4.2 Experience

During the first phase of the test-bed exploitation it came out that the network bandwidth and the latency may be most important performance bottlenecks of DDSS applications. This is convergent to the theoretic considerations. The performance that were obtained between sites that used dedicated, optical interconnects was far better (up to 10 times) than the performance obtained over a public network (Internet). Therefore, from the perspective of DDSS applications, the latency and bandwidth reservation is a critical feature of NRPS.

In a typical DDSS scenario, the source and destination locations of the data to be transmitted are known a priori and fixed. GridFTP transfers are often a part of a bigger workflow. Most often they are used for transporting the input data to the computation system and copying the results back to the source system, or to another computing system. DDSS Backup/Archive applications are most often used in order to backup system files and the content of home directories of the system users. Therefore, the source and destination location of the data are fixed by definition.

For above reasons, we assumed, that, opposite to link bandwidth and latency, the possibility of reserving access to the node with a given class of CPU and/or amount of available RAM memory is not a crucial option.

However during the first phase of application testing we observed that CPU resources available in DDSS GridFTP transmission parties can seriously limit the performance of the data movement operation. Big number of DDSS GridFTP transmission threads (the top performance over long-distance links was obtained with 80-112 threads) may consume a lot of CPU resources on both ends of the transmission process. This happens especially if the test nodes are equipped in single or single-core processors. Considerable CPU resources are used for managing the big number of transmission threads and performing the actual data transfers. Therefore it would be useful, if the NRPS and the Grid middleware provided the functionality that enables CPU resources reservation for data transmission process. Given that transmission ends are fixed, DDSS GridFTP application could request a link (bandwidth and latency) between fixed transmission ends and some level (e.g. 10%) of CPU availability on these ends.

Another conclusion from the first phase of application testing is that configuring the firewall passes through the organisation's networks is a very time-consuming task. It requires a lot of mailing and phone-calls. Moreover, any changes in configuration may require doing the job again from the scratch. In our test-bed, this happened twice: after the reconfiguration of the test-bed nodes in FZJ and after the reconfiguration of test cluster in UESSEX. Therefore, in our opinion, the firewall passes are important issues that should be taken into account while designing a network resources management features for grid middleware. In such approach, firewall passes should be treated equally to another network resources like link, bandwidth, latency and CPU resources. This applies to environments, where the dedicated optical links ends at the firewall or router ports. This would require implementing the interface to organisation firewalls as well as the interfaces to a local firewalls running on computing nodes. We imagine, that firewall passes could be opened on-demand (obviously, respecting local rules and limits), similarly and in parallel to the network reservation process.

1.5 INCA

As described previously INCA is an intelligent storage network that provides data-transfers with high performance. As we depict in the following figure the functionalities of INCA are divided in two layers. The first layer is called **Data Management Layer** and its purpose is to infuse in the system totally distributed and automated management functionalities. In more detail we test and evaluate four major features that each storage network must have:

1. The first one is scalability. In order to achieve this goal we have implemented a distributed hash table (DHT) that is used for the distributed metadata maintenance and the distributed location of them.

2. The second is the balanced use of storage network resources in terms of storage space and network bandwidth. In order to achieve this goal any data chunk is hashed and according to the output of a uniform hash function is stored in the proper storage node.

3. The third is the fault-tolerance of a storage network. A stable routing algorithm used for the data and metadata location has been developed and has been evaluated.

Through our experiments we have evaluated our architectural decisions and the performance of the algorithms that we have used in order to enhance the storage network with these features. The results show that a DHT is capable to manage well a storage system but special attention has to be paid in order to maintain all the necessary data structures during dynamic conditions.

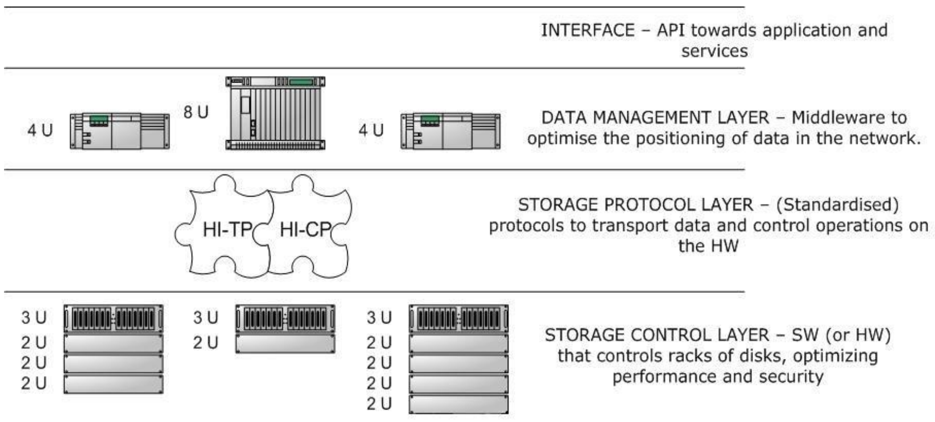

INTERFACE – API towards application and services
4 U
8 U
4 U
DATA MANAGEMENT LAYER – Middleware to optimise the positioning of data in the network.
HI-TP
HI-CP
STORAGE PROTOCOL LAYER – (Standardised) protocols to transport data and control operations on the HW
3 U
2 U
2 U
2 U
3 U
2 U
3 U
2 U
2 U
2 U
2 U
STORAGE CONTROL LAYER – SW (or HW) that controls racks of disks, optimizing performance and security

Figure 6: INCA Layers

As for the second layer it is called **Storage protocol layer**. It's purpose is to provide:

1. Point-to-point buffer transfer in order to store chunks in the choosen (by the upper layer) node;

2. The glue between **Data management layer** and the storage control layer.

2 Middleware optimisations

2.1 UNICORE

2.1.1 Modifications derived from the testbed experiments

During the first testbed experiments no major issues for modifications of the UNICORE middleware have been identified. However, there are a number of smaller issues for modifications that could improve user-friendliness and would improve the overall functionality.

- Support for workflows
- UNICORE remote site registry
- Support for applications as UNICORE resources
- Extensions for the UNICORE client (plugins)
    - allowing to specify parameters of an application controlling input and output
    - allowing to specify the number of sites and the number of nodes to be used for an application at a site

2.1.1.1 *Support for workflows*

In the current version of UNICORE (6.1) currently there is not workflow engine integrated like in UNICORE 5. As a result, all workflows have to be defined and implemented outside of UNICORE, e.g. via shell scripts. These externally defined workflows are then passed to UNICORE and executed on the selected resources of the PHOSPHORUS testbed. Adding the workflow support to the UNICORE system would allow the user to specify the workflow inside of UNICORE while allowing using the information available in UNICORE on resources like sites, nodes, applications already during the definition of its workflow.

2.1.1.2 *UNICORE remote site registry*

A remote registry including all sites of the PHOSPHORUS testbed would allow the easily select sites for the execution of applications from the UNICORE 6 application client. Currently, the user has to specify the target sites manually, thus the site registry would improve user-friendliness of the middleware .

2.1.1.3 *Support for applications as UNICORE resources*

Currently it is the user's responsibility to assure that the application(s) he plans to use is available at the site he selects for execution. Adding applications as resources to the UNICORE middleware (like e.g. compute resources) would allow checking the availability of the application already during the specification of a job in the UNICORE client, the same way, the availability of other resources is checked by the system. This would increase the reliability of the submission and execution process as the risk of a user submitting a job to a site, where the required application is not available is lowered.

2.1.1.4 *Extensions for the UNICORE client*

Currently, the user has no interface to specify parameters for an application controlling input and output. Making this information available through an the UNICORE client is crucial e.g. for a better support of workflows during the specification and distribution to the resources. Adding a new or modifying an existing UNICORE client plugin thus would increase the user-friendliness.

Also, there is no possibility allowing a user to specify the number of sites and the number of nodes of a site to be used for an application. Without this information it is impossible to automatically distribute the data for multiple instances of an application like FlexX or AutoDock running in parallel on multiple sites. Extending a UNICORE client, to allow a user to specify this, would improve the process by making it more automatically and disburden the user from doing the distribution manually.

2.1.2 Plan for implementation and deployment

An initial version of workflow support is will be included into the next UNICORE version (6.1). The next version is scheduled for spring 2008. Version 6.1 will also include support for adding applications as UNICORE resources.

The modifications of the UNICORE client will be done during the next months and should be ready together with the release of version 6.1.

Once version 6.1 and the improved clients become available they will be installed for tests at both the FZJ and FHG testbed resources. After a brief period of testing the new version will be deployed to the other testbed sites as well to be ready for the next period of testbed experiments.

The UNICORE site registry will be available earlier and may be used from end of February on.

2.2 MetaScheduling Service (MSS)

2.2.1 Modifications derived from the testbed experiments

The UNICORE security model and implementation changed significantly moving from UNICORE 5 to UNICORE 6.

2.2.2 Plan for implementation and deployment

Work on first modifications of the MSS has started after the first experiments. Implementation will be completed during the first quarter of 2008. After a test-phase the new MSS version will be deployed in the PHOSPHORUS testbed. The deployment is expected to be completed end of March/bein of April 2008.

2.3 Automatic Selection of Resources based on Semantic Annotation of Applications

2.3.1 Modifications derived from the testbed experiments

We have not yet decided whether we use the Tool for Universal Annotation and Mapping (TUAM) or not [TUAM]. It has become clear in our own experiments that it is easier to use the query lanhuage SPARQL (RDF Query Language) directly on the ontology [SPARQL]. Instead of using TUAM, the annotation has been carried out in Protégé, an ontology editor [Protégé]. It is possible to enrich an ontology with semantic information in the ontology itself. The concepts (classes) of an ontology can be extendet by using individuals, i.e. instances of the concepts. We filled in those individuals with information retrieved from the questionnaires which we gathered from our partners (mentioned in D3.2). The result of this annotation is saved in an OWL-file (Web Ontology Language).

We will use Jena, which is a Java framework for building Semantic Web applications [Jena]. Jena is an opensource product of Hewlett-Packard Labs [Hewlett-Packard]. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. By using ARQ, which is a query engine for Jena and a SPARQL pocessoer,, we can obtain the SPARQLresults directly from the OWL-file and can use those results within the Jena framework, Thus, we do not need a RDF-file anymore.

2.3.2 Plan for implementation and deployment

Instead of using Tuam where we firstly had to combine two separate files in order to extract a RDF-file, which we could enquire via SPARQL, we now have only the OWL-file. We had to prepare some queries for the RDF-file to achive the same results as now with the OWL-file which provide the results with only one question.

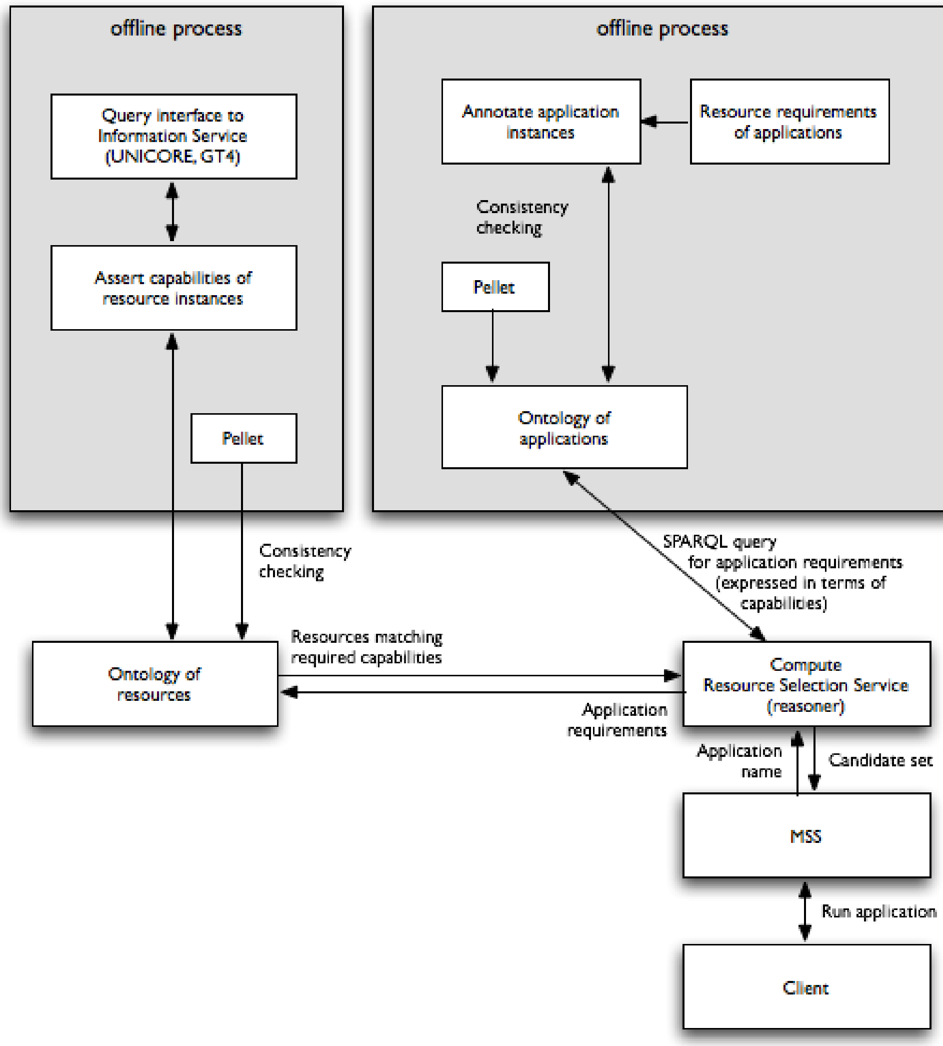

offline process
Query interface to Information Service (UNICORE, GT4)
Assert capabilities of resource instances
Pellet
Consistency checking
Ontology of resources
offline process
Annotate application instances
Resource requirements of applications
Consistency checking
Pellet
Ontology of applications
SPARQL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

Figure 7: Adopted architecture of the resource selection service.

The Jena framework provides the tools, which we need on the one hand to enquire both ontologies and on the other hand to match the results.

The implementation will take place, as mentioned above, in Java.

2.4 Globus Toolkit 4

2.4.1 Modifications derived from the testbed experiments

From the first phase of DDSS application experiments and DDSS application modifications we conclude that following additional functionalities Globus Toolkit middleware could be useful for PHOSPHORUS purposes. First, link (bandwidth and delay) and CPU co-allocation can guarantee that the computers on transmission ends are able to provide enough CPU resources to serve parallel, multiple-threads DDSS GridFTP transfers. Second, the firewall pass creation on demand feature would significantly simplify management of the testbed and production environment.

2.4.2 Plan for implementation and deployment

Link and CPU co-allocation features can be implemented in the form of meta-scheduling service. However, such services are external to the Grid middleware. Therefore, we do not see the need to implement GT4 modifications. However, extending the MSS as external co-allocation service also for Globus environments should be considered for a later phase of the project.

On-demand firewall pass creation can be useful for many Globus-based applications. However, we think that local management policies in testbeds and production environments are a serious obstacle in implementing and/or putting the feature into practice. Moreover, various types of firewall systems require implementing multiple interfaces between Globus Toolkit and firewalls. Therefore, we conclude, that, while on-demand firewall pass creation feature is demanded, it is not realistic to put it into practice in the Grid environment. Thus implementing it in the confines of PHOSPHORUS project doesn't make much sense.

2.5 INCA

2.5.1 Modifications derived from the testbed experiments

In order to describe the modifications that derived from the testbed experiments we will first describe briefly the basic functionalities that we implement in our system. In order to meet the goals and the requirements that we described in the previous section in INCA are implemented the following major functionalities:

Data Management Layer

- Node insertion and removal
- Distributed routing table and DHT zone maintenance
- Application layer routing mechanism

- Metadata insertion and removal
- Distributed metadata base maintenance
- Metadata insertion and data placement
- Metadata location and data retrieval

Through the testbed experiments 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 **Storage protocol layer.** Through our experiments we have pointed out some issues on the testbed, as we achieve 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 brings in evidence some key points:

1. The need of jumbo frames in order to exploit the gigabit link (ok, this is a well known issue but it seems to not be taken in account in other systems);
2. A good tuning of nodes is needed;

In absence of that some development has to be done to deal with kind of issues and to try to counter that in **storage protocol layer** (precisely **HITP**).

2.5.2 Plan for implementation and deployment

As it is obvious from the aforementioned description suitable environment for the implementation of our system can be any platform that can manage and support a storage space and network cards. The hole INCA consists only from software components that apart both layers that presented earlier.

The deployment could also be done in any site of the project and additionally in any other site. We are currently negotiating for more nodes in order to evaluate our system in a condition where more INCA nodes will be present.

Additionally in order to evaluate the **Storage protocol layer** we plan to deploy INCA in nodes with high performance hardware.

3 Conclusions

The work on middleware optimisation following the first wave of testbed experiments has been identified and plans for the individual components have been set up. The envisaged modifications and extensions are described in this document, the optimisation process is on track and in line with the schedules set up in the description of work.

As a result of the first wave of testbed experiments during month 13 and month 16 the partners in WP3 focussed the optimisation work on the following four areas of the PHOSPHORUS infrastructure:

- A new version of UNICORE will be deployed in the testbed. Among other enhancements this version will include changes bringing additional or improved functionality to the testbed middleware:
  - Support for workflows
  - UNICORE remote site registry
  - Support for applications as UNICORE resources
  - Extensions for the UNICORE client (plugins)
- The MetaScheduling Service will be extended to support the new security model of UNICORE 6.
- The Resource Selection Service will be implemented based on ontologies for testbed applications and testbed resources.
- Based on the experience made INCA has been improved through a number of modifications on the Data Management Layer and the Storage Protocol Layer.
- Modifications of Globus Toolkit were considered, basing on DDSS GridFTP application testing: CPU and link co-allocation and on-demand firewall pass creation. However we concluded that they are not to be implemented in the confines of the PHOSPHORUS project.

4 References

[ARGON]	Allocation and Reservation in Grid-enabled Optic Networks http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0051.pdf
[EASY]	Extensible Argonne Scheduling system http://www.springerlink.com/content/pn740068375677wt
[CIM]	Common Information Model http://www.dmtf.org/standards/cim/
[GT4]	Globus Toolkit 4 http://www.globus.org
[GridFTP]	GridFTP. http://www.globus.org/grid_software/data/gridftp.php
[OASIS]	Organization for the Advancement of Structured Information Standards http://www.oasis-open.org
[OGSA-RSS]	Open Grid Service Architecture Resource Selection Services Working Group http://www.ogf.org/gf/group_info/view.php?group=ogsa-rss-wg
[MSS]	MetaScheduling Service http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0010.pdf
[OGF]	Open Grid Forum http://www.ogf.org
[PELLET]	http://pellet.owldl.com/
[PROTÉGÉ]	http://protege.stanford.edu/
[SPARQL]	http://www.w3.org/TR/rdf-sparql-query/
[TORQUE]	http://www.clusterresources.com/pages/products/torque-resource-manager.php
[TSMBA]	Tivoli Storage Manager. http://www-306.ibm.com/software/tivoli/products/storage-mgr/
[TUAM]	Tool for Universal Annotation and Mapping http://www.scai.fraunhofer.de/index.php?id=2384&L=1
[U@SOURCEFORGE]	UNICORE Sourceforge Project http://sourceforge.net/projects/unicore/
[unicore]	**Uniform INterface to COmputing REsources** http://www.unicore.eu
[VIOLA]	Vertically Integrated Optical Testbed for Large Applications in DFN http://www.viola-testbed.de/
[WS-RF]	Web Services Resource Framework http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf - announcements

5 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)
GT4	Globus Toolkit Version 4 (Web-Service based)
INCA	
KoDaVis	Tool for Distributed Collaborative Visualisation
MSS	MetaScheduling Service
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