034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number D3.6

Report on the Results of the Application Experiments During the Final Testbed Experiments

Due date of deliverable: 2008-07-31
Actual submission date: 2008-07-31
Document code: Phosphorus-WP3-D3.6

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
Forschungszentrum Jülich GmbH

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 document reports on the results of the final phase of application test bed experiments. They have been conducted in the PHOSPHORUS test bed after the second phase of developments as a reaction to the requirements found during the first test phase. All applications participating in the test bed experiments have been updated and conducted further tests during this testing phase. The results of these tests are conveyed in this document. The test bed has been described in D6.1 and D6.2, the initial adaptation of applications has been described in D3.3.

Project:	Phosphorus
Deliverable Number:	D3.6
Date of Issue:	31/07/2008
EC Contract No.:	034115
Document Code:	Phosphorus-WP3-D3.6

Table of Contents

Table of Figures

0 Executive Summary

The UNICORE 6 Grid middleware and the Meta Scheduler Service (MSS) have been enhanced and integrated to make synchronous reservations of compute nodes and network bandwidth available. This is achieved by interfacing UNICORE to a co-allocation service that provides this feature, relying on resource reservation systems with advance reservation capability.

WP3 (Middleware and Applications-Service Plane) provided these middleware services and applications. So a major part of this activity (part of WP3) is 'porting' selected applications onto the PHOSPHORUS testbed. Applications middleware were adapted to prove the concept and to demonstrate the benefit from the integrated test-bed environment developed in the project. The applications that have been selected to demonstrate the benefit of the PHOSPHORUS environment come from different areas spanning from large scale compute and data intensive simulations via collaborative environments exchanging enormous data volumes in real-time to remote backup and archival services requiring huge bandwidth with appropriate QoS. These chosen applications and their results after the final experiments are the following:

- WISDOM (Wide In Silico Docking On Malaria) consists of virtual screening techniques, computing compounds of large-scale molecular dockings on targets implicated in diseases like malaria. The goal in Phosphorus is to implement the WISDOM workflow based on UNICORE 6 in order to improve the reliability in comparison to previous experiments. The PHOSPHORUS experiments lead to the following results for this use-case. The client side of the WISDOM application is now using the UNICORE 6 rich client, which is a great improvement compared to the manual engagement used so far. Additionally many necessary adjustments to the software environment of both the WISDOM applications AutoDock and FlexX are made to realize a basis for an automatic workflow handling by the UNICORE 6/MSS integration. It is planned to demonstrate the application at the Supercomputing 2008 and ICT 2008 events in November. More visible network performance is planned to be shown at those occasions.

- KoDaVis – Collaborative Data Visualisation provides distributed collaborative visualizations of huge atmospheric data-sets. The software has been enhanced to use UNICORE to manage its resources and provide access to the test-bed functionality. As a result of the PHOPHORUS experiments KoDaVis has been enhanced by flexible performance monitoring, enabling users to identify performance bottlenecks and take appropriate actions during collaborative visualization sessions. Tests between the sites of PSNC and FZJ showed the utility of these new functions in the client and server layers.

- TOPS – Technology for Optical Pixel Streaming is a system that enables remote viewing of large scientific datasets on high resolution display devices (Tiled Panel Displays). TOPS streams these pixels, uncompressed, from the data centre to remote displays. For this, high bandwidth networks are required. The experiments on the PHOSPHORUS test bed has been proven for TOPS to be able to make use of all available network bandwidth and services that's given to it.

- DDSS – Distributed Data Storage Systems provides two use-cases, one based on GridFTP and one based on Backup-Restore/Archive applications. In both use-cases, the applications have been enhanced to interact directly with the network resource reservation systems. The software has been executed with different scenarios in the test-bed experiments. To resume these DDSS experiments, the DDSS application has been tested extensively between a number of participating sites of the Test bed. Various network and application parameters have been tried during the tests to find optimal settings for different scenarios of small and big file transfers. It was shown that the use of reserved bandwidth has an advantage over using public internet connections.

1 Introduction

Within the German project VIOLA, the UNICORE Grid middleware has been enhanced to make synchronous reservations of compute nodes and network bandwidth between the nodes. This is achieved by interfacing UNICORE to a co-allocation service that provides this feature, relying on resource reservation systems with advance reservation capability. Within PHOSPHORUS (WP3), this system will be adopted to the network reservation services provided by NRPS and Grid-GMPLS (WP1&2) and will be extended to orchestrate network, compute and other Grid resources in a more general way, as required by the showcase applications.

This document reports on the results of the final phase of application Test bed experiments mentioned above. They have been conducted after the second phase of developments as a reaction to the requirements found during the first test phase. The applications covered by this document are in a short overview:

- WISDOM (Wide In Silico Docking On Malaria) is a docking workflow/service which allows the researcher to compute millions of compounds of large scale molecular dockings on targets implicated in diseases like malaria (in silico experimentation). In silico docking enables researchers to compute the probability that potential drugs will interfere with a target protein and it is one of the most promising approaches to speed-up and reduce the cost to develop new drugs to treat diseases such as malaria. In this case Grids provide resources to identify potentially promising compounds and to refine virtual screening and further assess selected compounds. WISDOM presents both a compute and data challenge.

- KoDaVis – Collaborative Data Visualisation is a distributed, collaborative visualisation system with remote access to huge atmospheric simulation data originating from simulations of the transport of chemical tracers in the troposphere. Datasets with a typical size of 1 TByte are stored in one or more central data servers or at the supercomputer where the simulation was performed. In a collaborative session, two or more visualization clients team up, each connects to the data server, and triggered by user interaction, fragments of the data are sent to all visualization systems on demand, e.g. specific time steps or selected tracers. Although the clients may be implemented using different visualisation environments, a synchronization service ensures that all share the same view on the data. For reliable fluent visualization sessions, the reservation of bandwidth between the data server and each of the clients is required. The main goal within PHOSPHORUS is to enable KoDaVis to perform such reservations using the UNICORE middleware.

- TOPS: TOPS is a system that enables remote viewing of large scientific datasets on high resolution display devices (Tiled Panel Displays). These displays typically have 30 to 100 million pixels. TOPS streams these pixels, uncompressed, from the data centre to remote displays. For this, high bandwidth networks are required. A PHOPHORUS adapted TOPS will be able to utilize user/application controlled lambdas for the visualization service. Datasets viewed with TOPS are typically gigapixel scientific visualizations, from many different fields of science, among which climatology and biology are important application drivers.

- DDSS: Distributed Data Storage Systems (DDSS) are widely used in scientific and commercial applications in order to transport, exchange, share, store, backup/archive and restore data. Two DDSS applications are examined in the PHOSPHORUS project: GridFTP service used for large data transfers between Grid sites and backup/archive application used across Grid sites. Backup/archive application (B/A) is used for performing automatic backups and/or archive copies of data that are originally stored in the Grid nodes or end-user nodes. B/A clients collect the backup/archive data and send them using TCP/IP streams to a B/A server. The main goal within PHOSPHORUS is to enable both DDSS applications to perform reservations of bandwidth and/or guarantee low latency for one-to-one and/or one-to-many high-volume data transfers. The bandwidth and latency guarantees are needed in order to make Grid FTP data movements and B/A data copies effective and robust.

More details about these PHOSPHORUS applications and the results of the final experiments on the test bed are described in the following sections.

This deliverable is a follow on of the following deliverables

- D3.4 – Report on middleware extensions and implementation

- D6.5 – Application and Middleware Testing Report

- D6.6 – Plan of Testing

2 Application Tests

2.1 WISDOM

2.1.1 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.

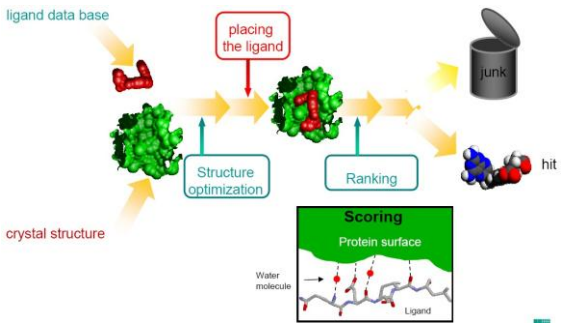


Figure 2.1: Virtual docking processes of a protein target and ligand molecules resulting in candidates for further expensive inspections

- 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 [D3.3] and [D3.5].

2.1.2 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.

After analyzing of the WISDOM workflow, first tests were made on the Test bed to identify application specific requirements for middleware Meta Scheduling Service (MSS) and UNICORE 6 workflow support. After having the availability of needed features and support, MSS/UNICORE 6 enabled executions of such workflows will be performed. Currently, the workflow engine of UNICORE 6 is only available in a prototype status. Thus the execution of both codes is actually done based on scripts, which are executed via the UNICORE 6 Rich Client.

More details about the WISDOM codes' workflows and requirements for execution are described in [D3.4].

2.1.3 Achievements and results

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

For the first test phase of WISDOM a test data suite from BioSolveIT was selected. Based on local installations tests were done on all four PHOSPHORUS sites to test the different docking software modules.

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. The BioSolveIT flexx-200 Testdata suite is a subset from the Protein Database (PDB), where each ligand was separated from the protein-ligand-complex and hand-fixed concerning protonation, aromaticity, delocalisation, and formal charges. Additionally start scripts and data environments were modified to realize a good basis for UNICORE 6 executions. Finally code installations and test data sets were made available on all participating PHOSPHORUS sites (except FlexX at PSNC, because of operating system incompatibility).

Data Transfer methods

On the technical side, UNICORE 6 complies with the OASIS WSRF 1.2 and OGF JSDL 1.0 standards, and provides pluggable file transfer mechanisms with the OGSA ByteIO standard as default. The usual way of transferring a file starts with the first the client sending a SOAP message to the server, which initiates a file transfer object. Then, 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.

From the user perspective special job storage locations are used. If a user creates a job, he gets a link to the generated objects and is enabled to transfer files from the client or from storage in the Grid. For each job a so called User Space (USpace) is generated by the UNICORE system, which contains programs, scripts and other data. After termination of the job output data will be exported back to the client or any specified remote storage locations.

UNICORE-ByteIO is the standard method for file transfer in UNICORE 6. It uses the feature in UNICORE to transfer information via web services. Thus ByteIO needs no special connections between the sites. All data will be sent in small portions with SOAP messages via HTTP to the gateways, which forward again via HTTP to the target systems. In this way ByteIO uses automatically the encoding and authentication methods of UNICORE. UNICORE file transfer is based on one basis class, which offers functionality of a web service and exchange of data and error handling. For each web service a specific class for the client exists, which executes the exchange of SOAP messages. This is the basis for different file transfer classes, whereas in practice only the ByteIO method is used. Thus to resume: Basically, this is a simple, flexible and secure method, which encapsulates transferred data in SOAP messages. Data is following the same encoded way between client, gateway and server like the control information. Unfortunately this method lacks performance and reaches max. 400 KB/s. Additionally, UNICORE 6 is offering Basic File Transfer (BFT), which uses HTTP for file transfer and achieves network transfer performance of several MB/s.

Further there will be soon an improved high performance data transfer method called UDT available.

In an internal report of T. Oistrez (Forschungszentrum Juelich GmbH) a prototype implementation of the UDT method is compared with Globus' GridFTP:

- GridFTP is based on the FTP-based data transfer protocol. Data is exchanged between Server and Client/Server via multiple TCP connection. Different to normal FTP GridFTP send data encoded and uses more than one data channel. The middleware Globus uses GridFTP for data transfer in Grids.

- UDT uses the UDP-Hole-Punching method. The User Datagram Protocol (UDP) is a non reliable and non connection oriented transport layer protocol. The application that uses UDP has to make sure that the data is completely transmitted. Although no connections exist, firewalls use a simple mechanism to feign a connection. The client generates the UDP datagram and sends it to the firewall. The firewall examines the datagram and forwards it to the destination. The concept of UDP hole punching can be easily modified to be used in Grid environments.

GridFTP (1 Stream)	GridFTP (4 Streams)	UDT 3	UDT 4
81 Mbits/s	294 Mbits/s	700 Mbits/s	930 Mbits/s

Table 2.2: Comparison of data transfer using Globus GridFTP and UNICORE 6 UDT (Source T. Oistrez, FZJ-JSC-IB-2008-02)

These results show improved performance using UDT, which is approximately three times faster than Globus GridFTP.

UNICORE 6 Rich Client

UNICORE 6 offers three different clients to the users, UNICORE Commandline 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).

The heart of the client is the so-called "Grid browser" view, which shows the Grid resources available to the user in a tree-like fashion. In the following Figure shows part of the Test bed with sites JUGGLE, PACKCS and PSNC (during the screen shot UESSEX node was not accessible).

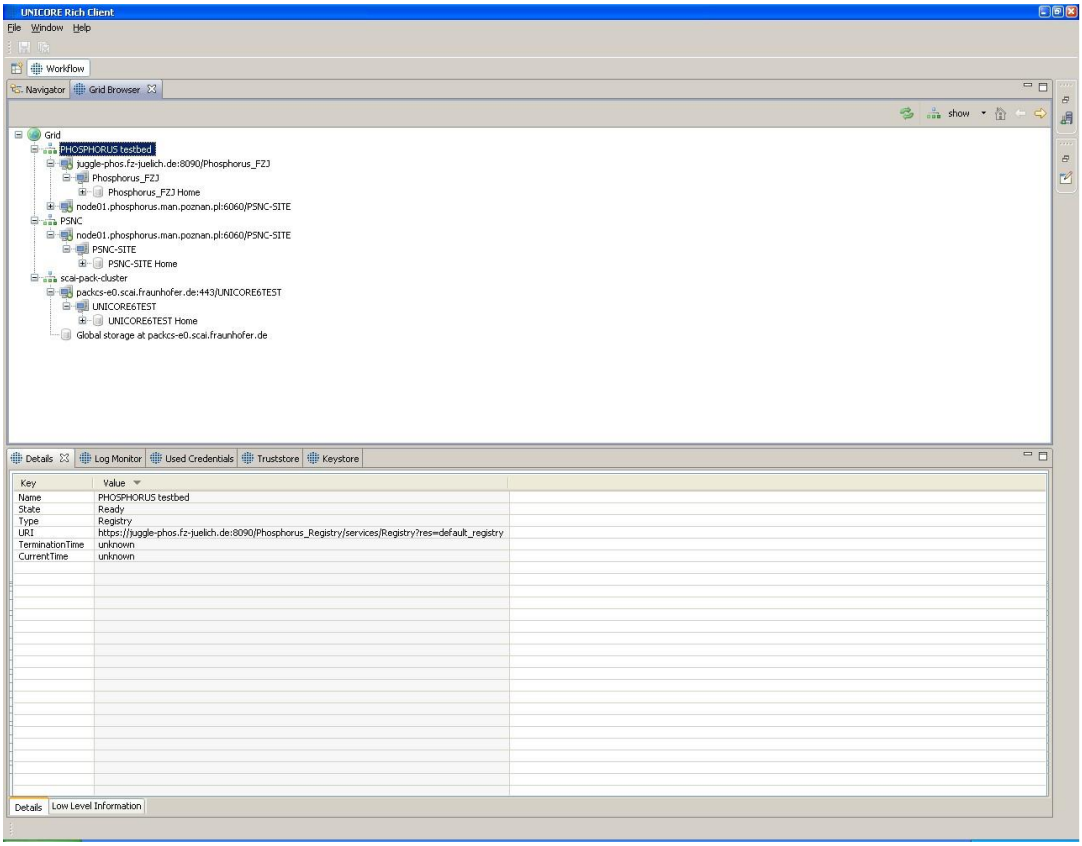


Figure 2.3 Screenshot of sites JUGGLE, PSNC and SCAI, part of the Test bed, in the UNICORE 6 Rich Client view

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 DGRID 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 visualized.

Next sections are focused 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.

WISDOM run: AutoDock job creation

Selecting sites in the UNICORE 6 rich client offers the creation and execution of WISDOM jobs. This can be AutoDock or FlexX jobs. This is realized by choosing script-based execution and entering the WISDOM applications calls.

For WISDOM jobs – AutoDock or FlexX – different amounts of storage are needed. In the case of WISDOM computing power to execute the applications is less important, but the input files and, moreover, the produced output files need a lot of storage as shown in the next table of figure 2.4:

WISDOM	Files	Storage
AutoDock Input	approx. 30 receptor files (*.glg, *.gpf, *.map, *.maps.fld, .maps.xyz, *.pdb, *.pdbqt)	30 MB
	ligand files (*.pdbqt, *.dpf)	6 KB
AutoDock Output	Result file (*.dlg)	400 KB
AutoDock docking scenario: 10 targets (receptors) docked with 100.000 ligands	collected input data target ligands	300 MB 600 MB
	collected output data	400 GB

FlexX Input	approx. 2 receptor files (*.mol2, *.rdf)	6 KB
	ligand files (*.mol2, *.pdb, *.sdf)	700 KB
FlexX Output	Result file (*.mat, *.sol, *.mol2)	35 KB
FlexX docking scenario: 10 targets (receptors) docked with 100.000 ligands	collected input data target ligands	60 KB 500 MB
	collected output data	35 GB

Table 2.4 Overview of WISDOM runs data amounts (files and storage)

Of course this is depending on the amount of ligands, which are proved during the docking process. But 100.000 to 250.000 ligands are typical amounts of ligands in docking scenarios. In our scenario of 100.000 ligands especially AutoDock results in large amounts of output data with 400 GB, approximately 10 times more than FlexX. Thus in this typical WISDOM scenario stage-in with storage requirements of 1 GB is today not a big hurdle but the bandwidth and storage intensive stage-out process with data transfers/storage requirement of up to 400 GB is really a big network challenge.

In a test a ZINC database subset was transmitted from PACK to the JUGGLE cluster using normal SCP. A file with 200.000 ligands and 564 MB storage size needed 56 sec. for the transmission with a performance of 10.1 MB/s. It depends on the number of nodes and connections how much time is needed to transmit the AutoDock output data of the above described docking scenario of 400 GB back to the WISDOM output server (one specified node getting all output data after compute termination). Using nearly all available nodes (approx. 50) of the Test bed this output data transmission with the standard ByteIO method needs more than 5 hours. Using the more performant BFT method (20 times faster) results in approximately 16 minutes, if all nodes are able to transmit their output data in parallel.

But let us come back to the WISDOM experiments. The next Figure shows the job creation process using the UNICORE 6 rich client. First the name of the job and the executing shell on the target systems must be specified. Afterwards the command for execution must be edited for this script-based realization. Additionally resources and resource requirements like CPUs, RAM, disk space, etc. can be defined too.

The client contains a browser for remote file system, that allows managing remote UNICORE storages seamlessly, using common file operations including copy, rename, delete, read, write, etc. The file browser supports the usual editing features such as drag and drop.

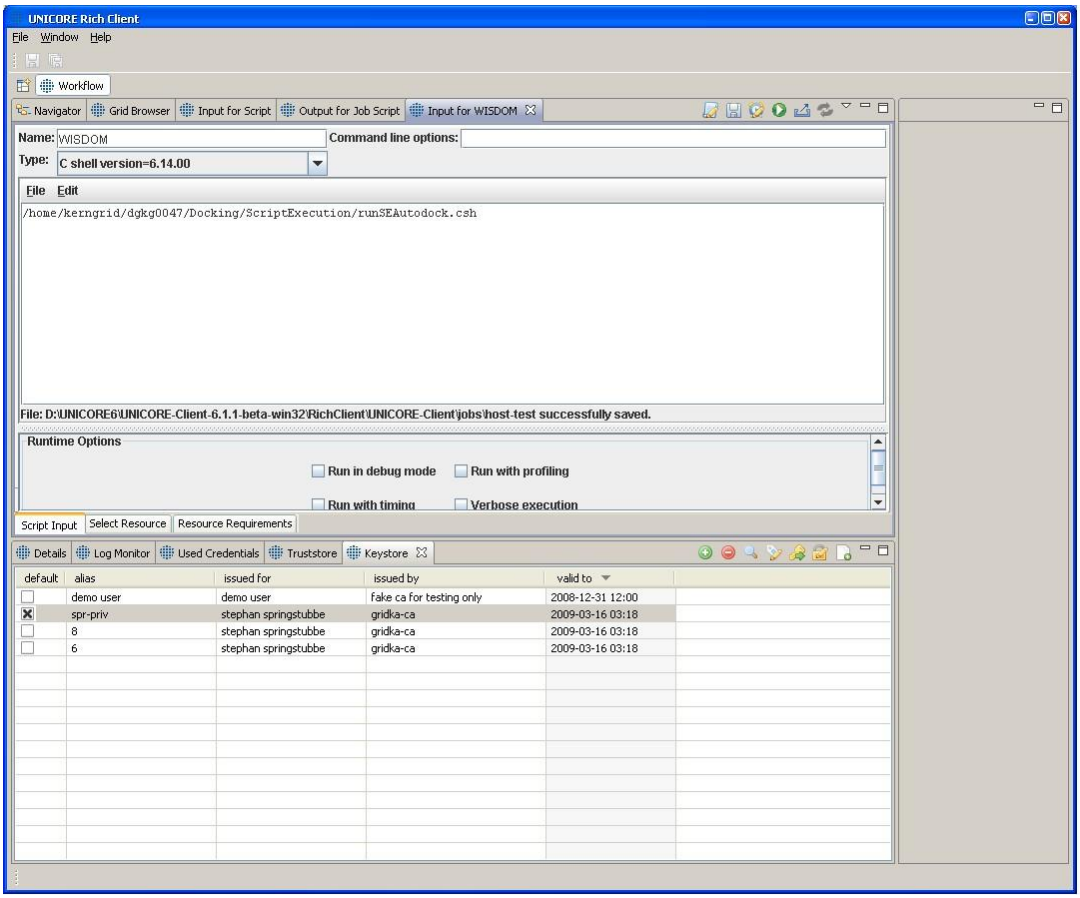


Figure 2.5 Screenshot WISDOM AutoDock job creation on Test bed via UNICORE 6 Rich Client

With the selection of the green “arrow” button the job creation is finished and the execution phase may start. In this phase it is the task of the UNICORE 6/MSS components to proof and enable job execution accordingly to the user specifications.

WISDOM run: AutoDock job execution

The next Figure shows the execution of a created WISDOM AutoDock job. The client assists in monitoring the state of the execution.

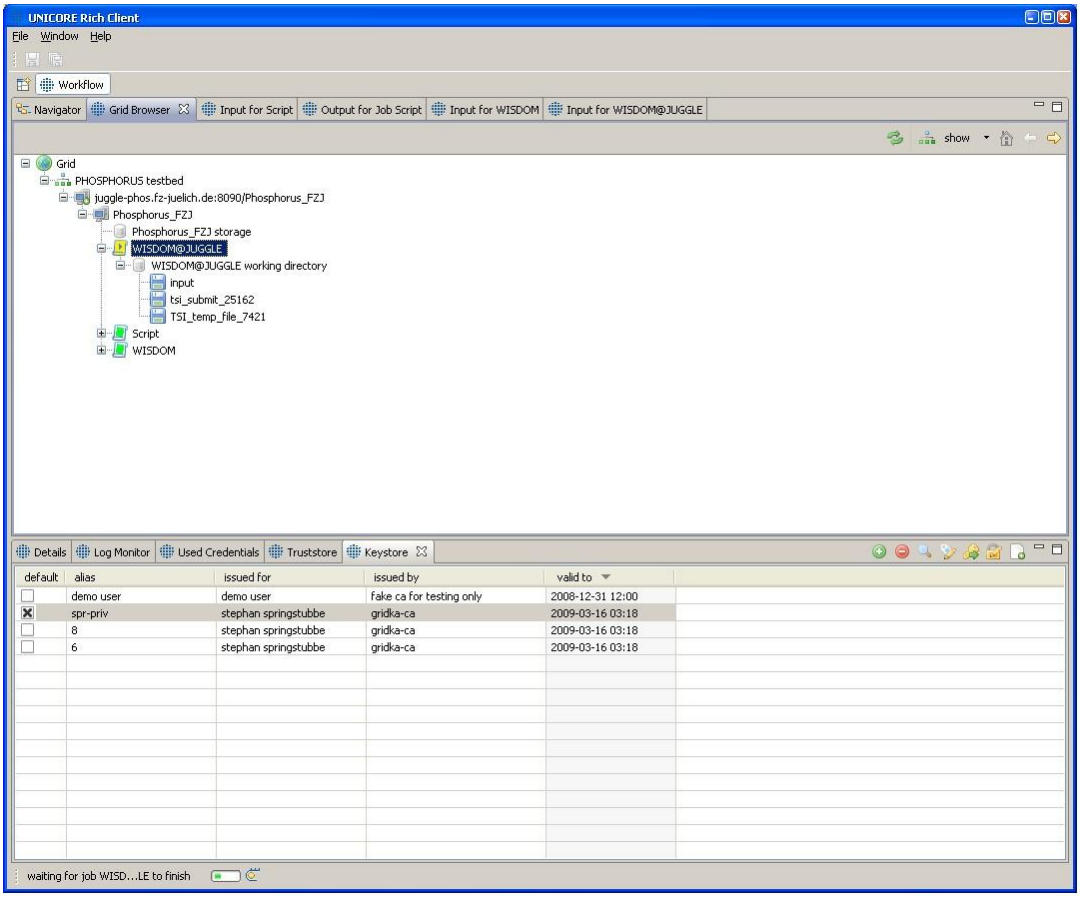


Figure 2.6 Screenshot WISDOM AutoDock job execution on Test bed via UNICORE 6 Rich Client

Several new links are indicating the successful job creation. During the execution a small bar at the lower end of the window shows the progress of this process. Further information can be found in the Rich client information folder “Details” and “Log Monitor” showing status and log messages of the Grid environment and the actual running job.

WISDOM run: AutoDock job terminated

After the termination of the WISDOM AutoDock job further links to system output files are created. The user may retrieve information about his job looking at:

- input: Showing the executed command.
- stderror: hopefully empty without error messages.
- stdout: showing results / output of the job execution.

- tsi_submit: Showing the batch data of the job.
- TSI_temp_file: Showing the main UNICORE TSI information.
- UNICORE_SCRIPT_EXIT_CODE: Showing the exit code of the executed script command.

The following Figure shows the user view after the job termination of the UNICORE 6 rich client.

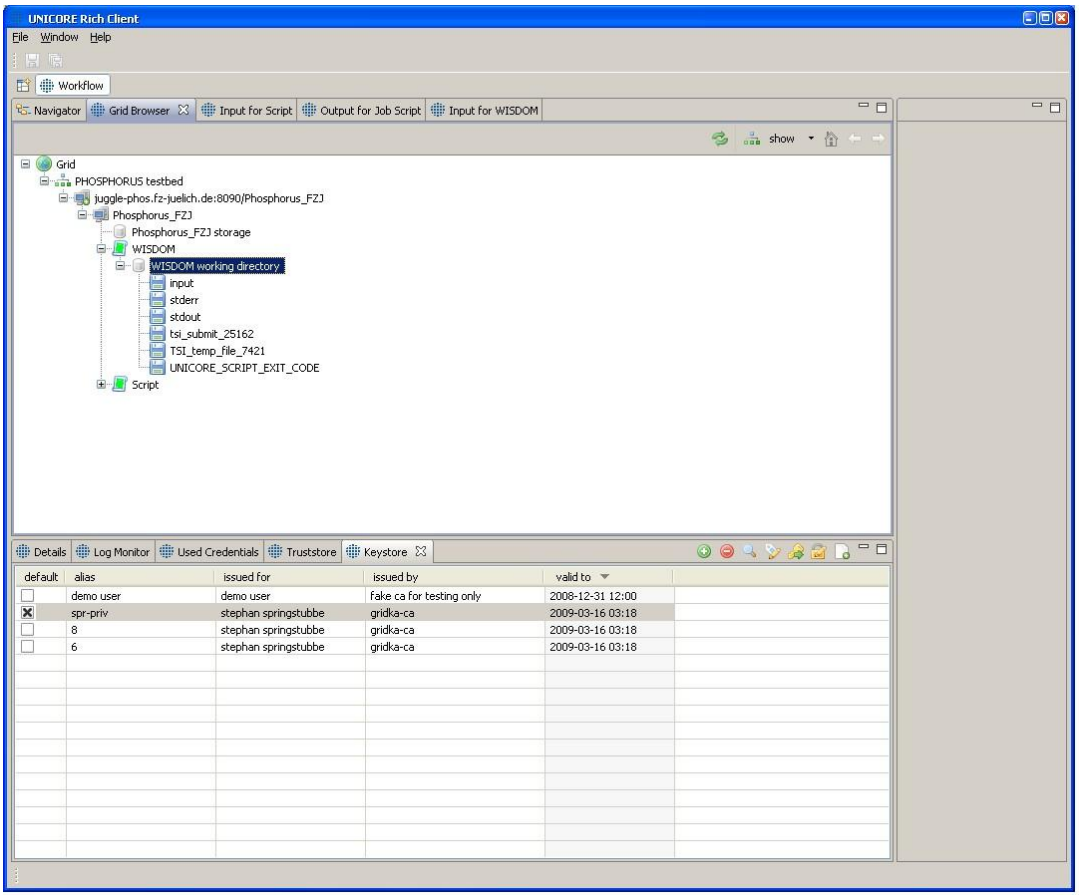


Figure 2.7 Screenshot WISDOM AutoDock job finalized on Test bed via UNICORE 6 Rich Client

WISDOM run: AutoDock job script output

The next Figure shows the content of system file stdout showing the output of the WISDOM script-based execution on nodes juggle37 and juggle39 via UNICORE 6 rich client after successful completion.

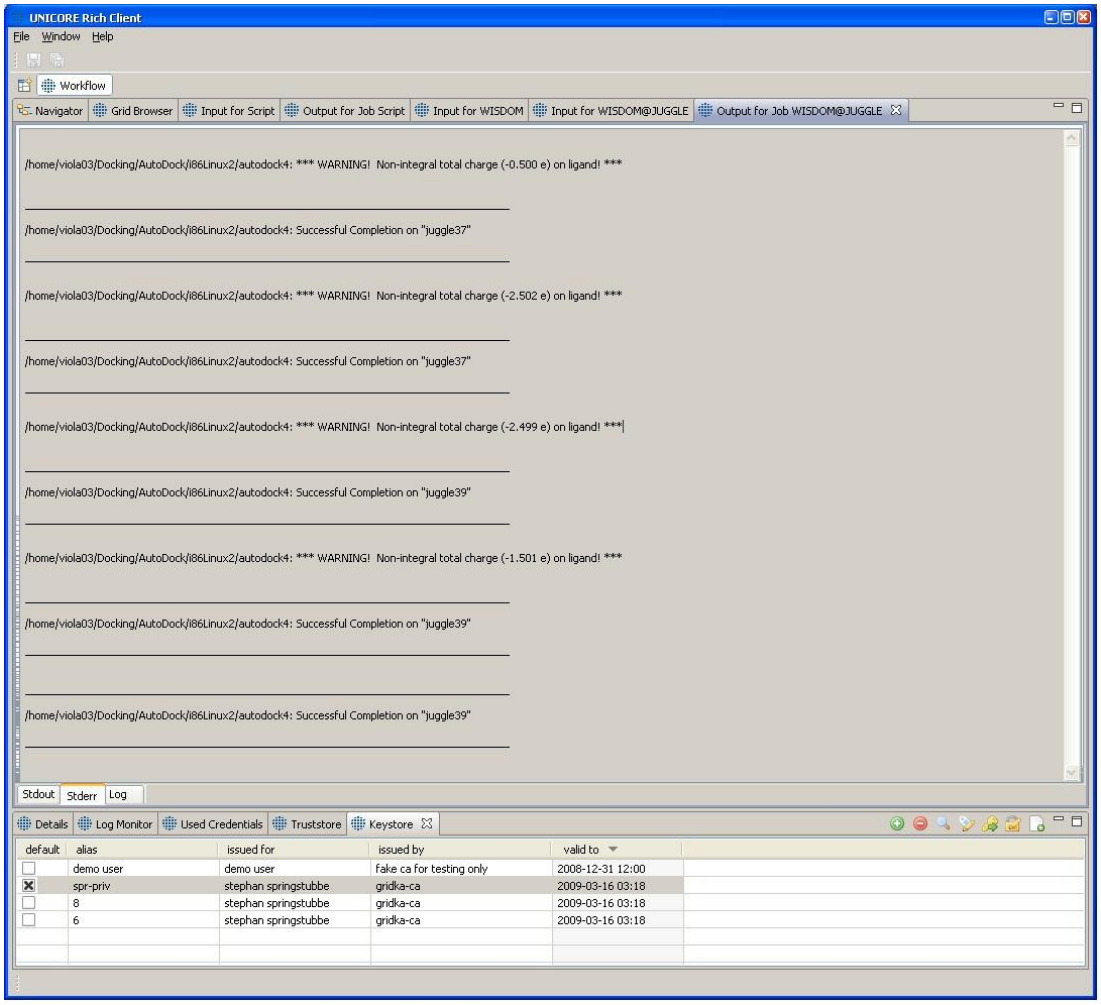


Figure 2.8 Screenshot WISDOM AutoDock job script output on test bed via UNICORE 6 Rich Client

Finally, the content of the generated AutoDock output files (*.dlg) must be analyzed. Usually a post-processing is needed to inspect all output data for correctness and results.

WISDOM run: FlexX job creation and execution

Very similar to the above description of the WISDOM AutoDock jobs WISDOM FlexX jobs are performed. So, it started again by selecting sites, creating the job with several specification options and executing the job with UNICORE 6/MSS.

The next Figure shows the execution of a WISDOM FlexX job on the JUGGLE system. The same resource and requirement specification are offered for these runs.

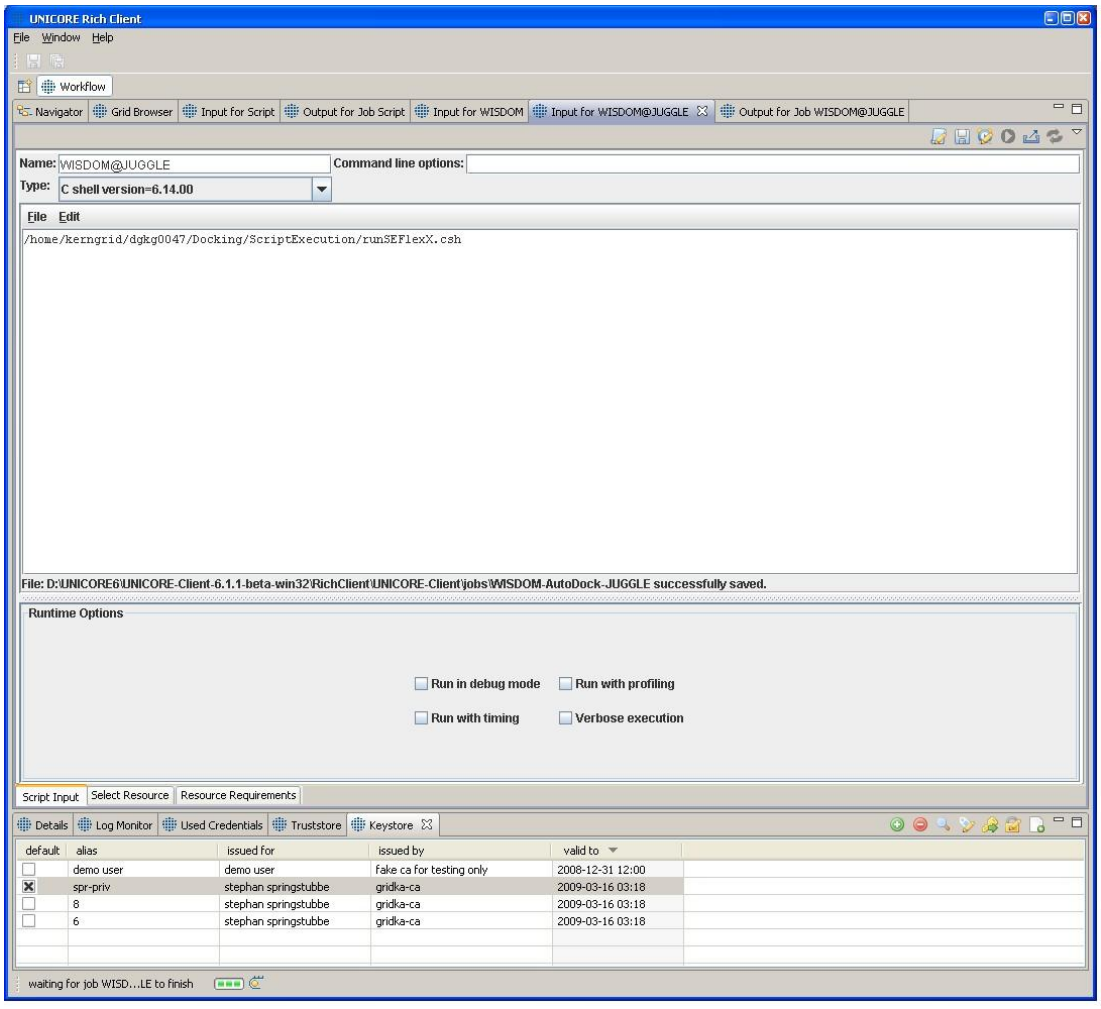


Figure 2.9 Screenshot WISDOM FlexX job execution on test bed via UNICORE 6 Rich Client

WISDOM run: FlexX job terminated

After the finalization of the WISDOM FlexX job, the same links to system output files as described above are accessible, as shown in the next Figure:

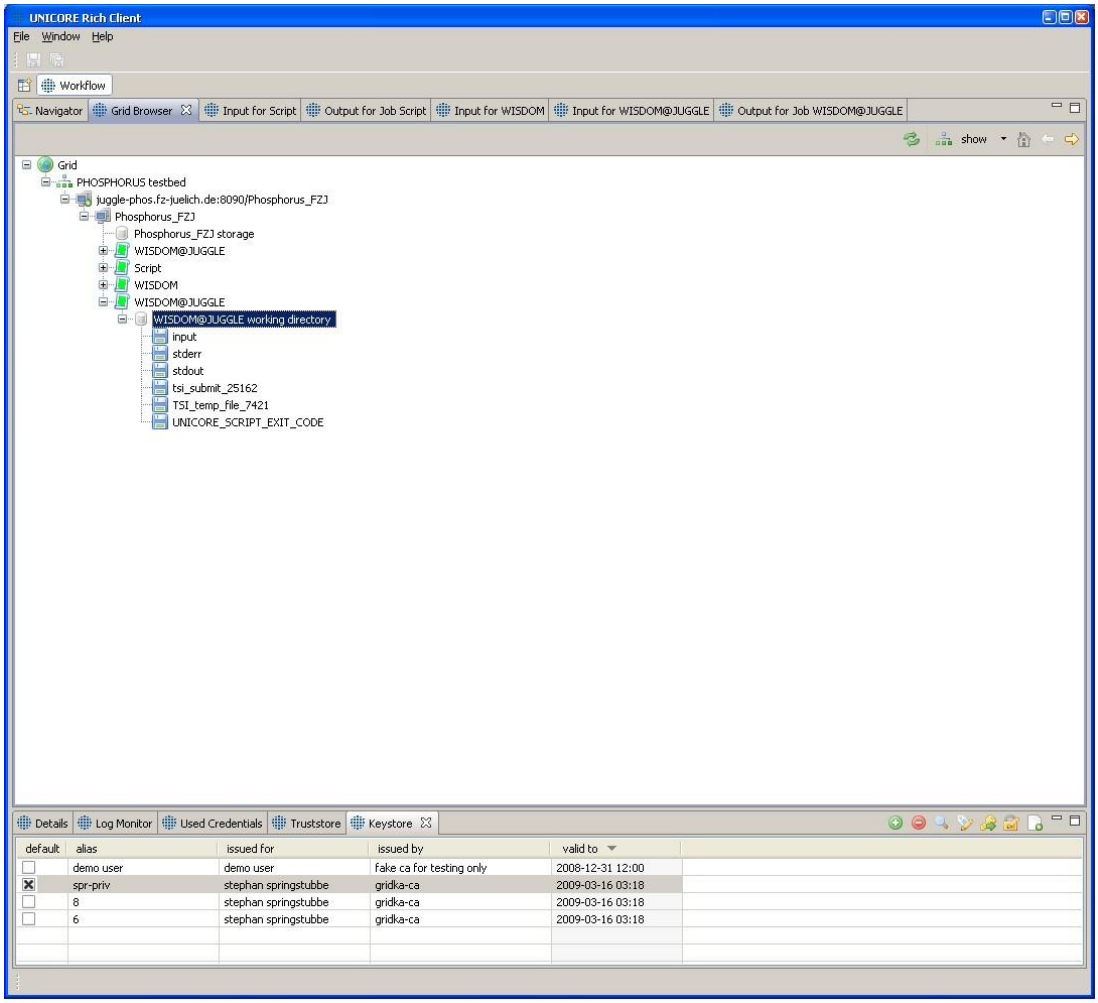


Figure 2.10: Screenshot WISDOM FlexX job finalized on test bed via UNICORE 6 Rich Client

The job execution progress bar is removed and the standard system output files input, stderr, stdout, tsi_submit, TSI_temp_file and UNICORE_SCRIPT_EXIT_CODE are shown in the Rich Client view and can be accessed for further inspection by the user.

WISDOM run: FlexX job script output

Finally, like for AutoDock, in the next Figure the output (stdout file) with the FlexX job execution output data is shown. It shows the successful license check, enabled via license server access at Fraunhofer site.

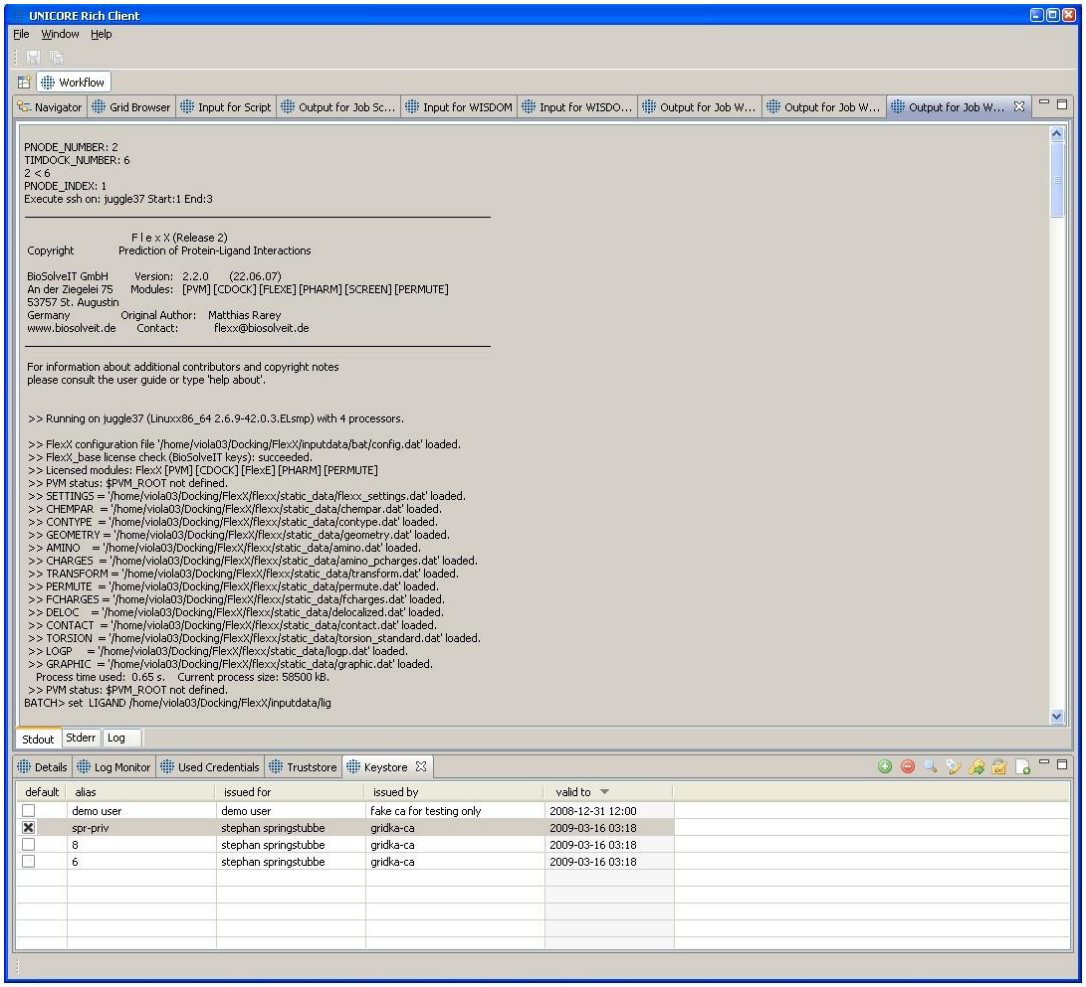


Figure 2.11: Screenshot WISDOM FlexX job script output on test bed via UNICORE 6 Rich Client

At the end of a WISDOM FlexX job output data is written to the output files. These data contain timings of the docking processes and some final statistics. The final part of this output is shown in Figure 2.12.

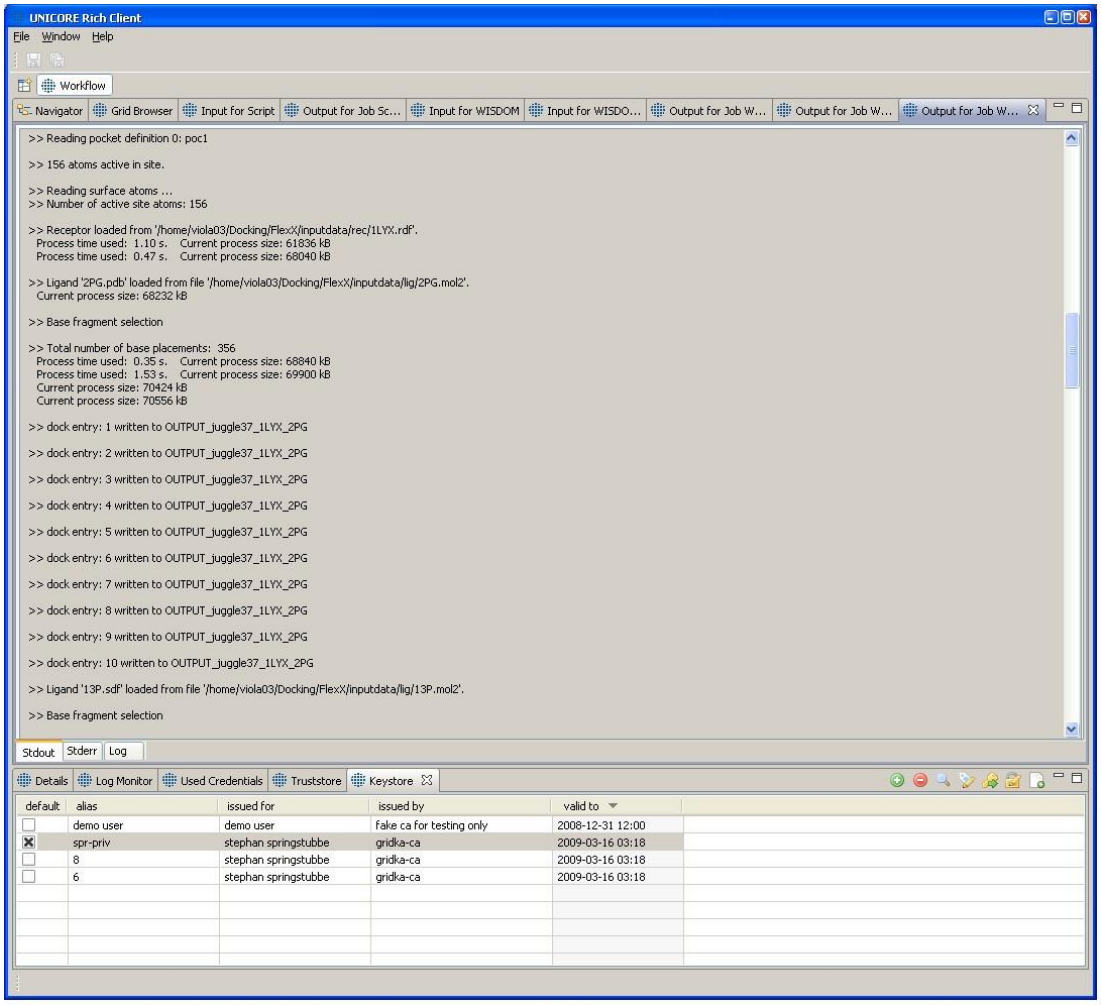


Figure 2.12: Screenshot WISDOM FlexX job script output indicating the local result files on test bed via UNICORE 6 Rich Client Screenshot

To conclude, WISDOM is running now on UNICORE 6 platforms on the test bed. Jobs are executed using the UNICORE 6 Rich Client, which offers an easy, user-friendly Graphical User Interface for pre-defined WISDOM AutoDock and FlexX runs. The workflow realization is based on the UNICORE 6/MSS integration.

For the events Supercomputing and ICT this year further work is planned to demonstrate more visible network performance on the test bed with WISDOM runs.

2.2 KoDaVis

2.2.1 Introduction

KoDaVis enables users from various remote sites to collaborate in visualizing 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. Figure 2.14 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).

2.2.2 Extensions after the first wave of experiments

One of the findings during the first phase of test bed 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 visualization 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 visualization 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 plugin. 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.

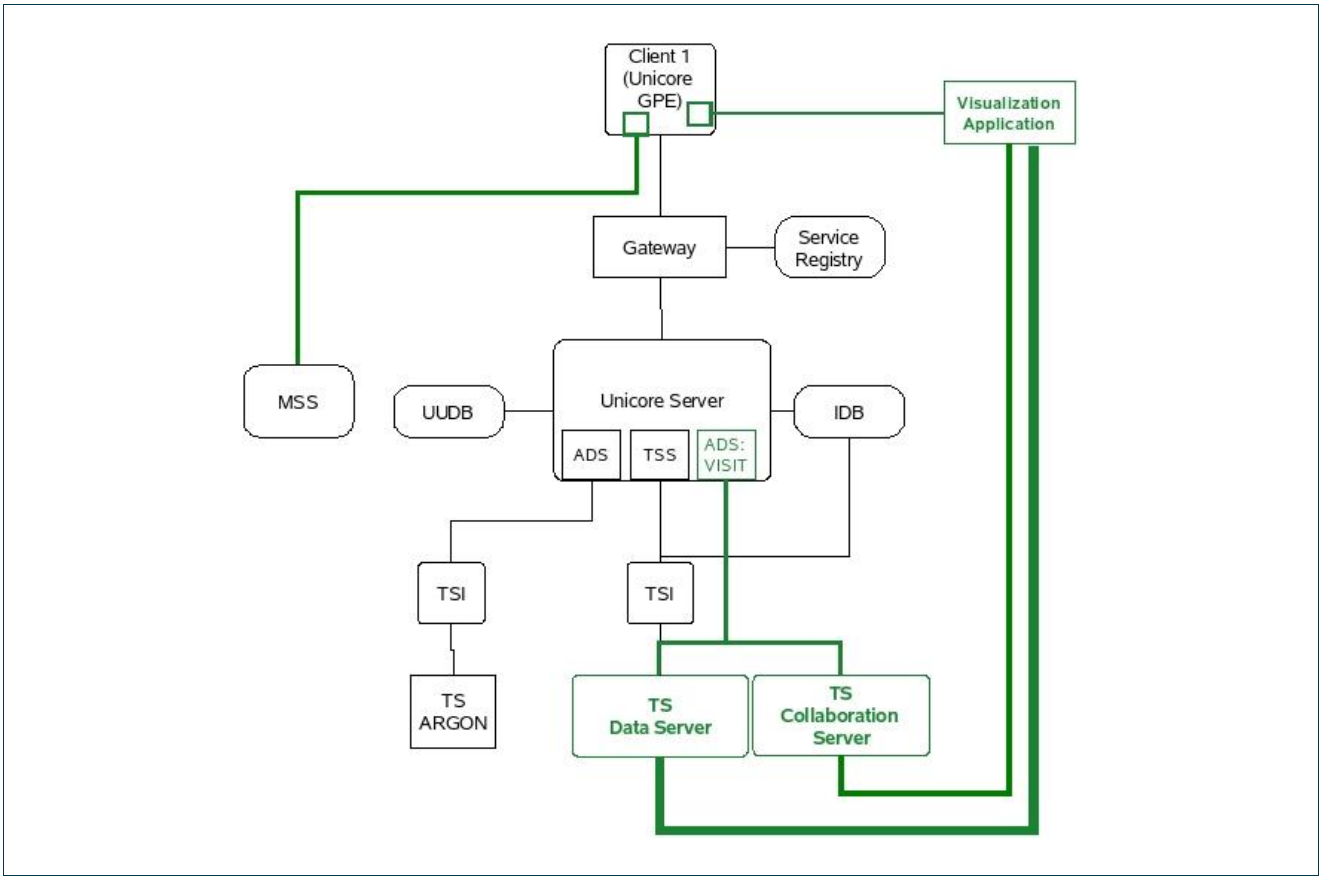


Figure 2.13: Architecture of the KoDaVis application embedded in UNICORE. Green Components have been developed in PHOSPHORUS and have been tested successfully within the experiments.

Figure 2.13 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 2.3 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.

2.2.3 Testbed experiments

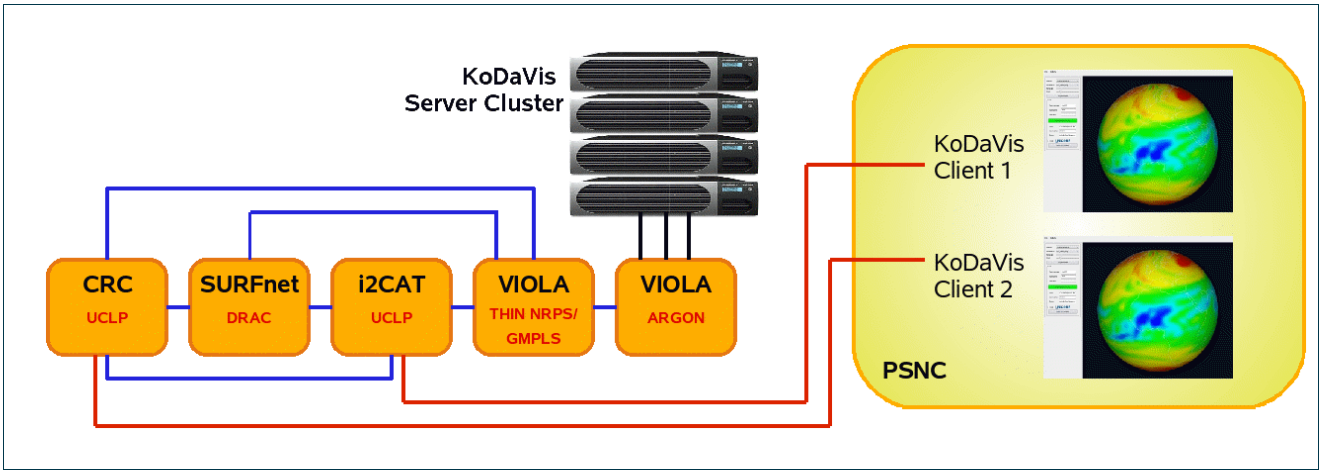


Figure 2.14: Example setup of a KoDaVis experiment, connecting two visualization clients at PSNC over different routes to the KoDaVis Server cluster in Jülich.

While quantitative experiments have been conducted during the first phase of experiments [D3.4], 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 2.15 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.

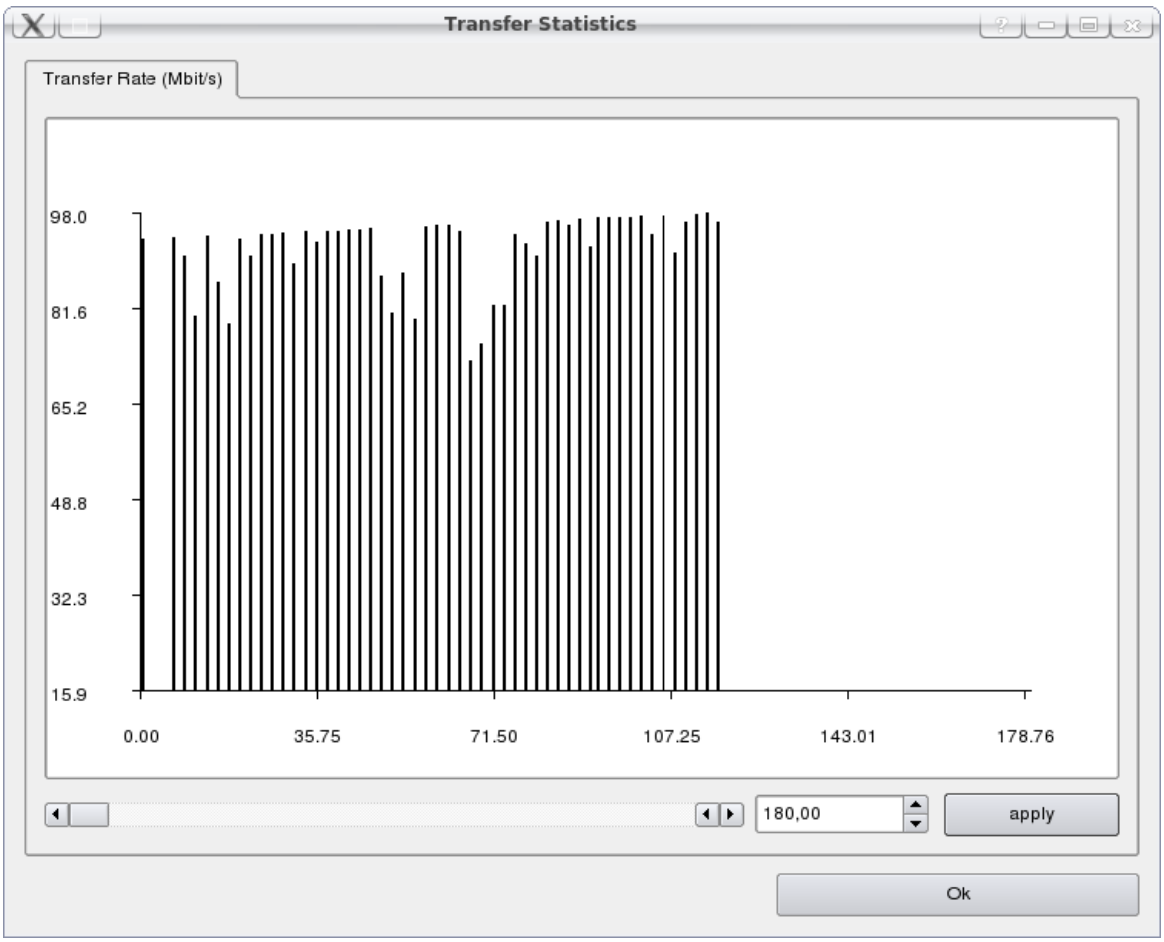


Figure 2.15: Transfer statistics displayed in the KoDaVis client.

Figure 2.15 shows the data transfer times of data from the server to a connected client. This performance has been observed during a session where both client and server were located at FZJ. Figure 2.16 displays the corresponding transfer times over the same link. Data in the KoDaVis application comes in bursts of approximately 3.4 Megabytes. A transfer time of ~0.18 seconds thus implies a transfer rate of 150 Mbits/s, which is impossible over a link of 100 Mbits/s, particularly if the receiving application measures 100 Mbits/s. This can be explained either by smaller chunks of data being transferred or a different method of measurement at the server side. In essence, the absolute values for transfer times aren't very important. The transfer times displayed in the UNICORE client plugin exist to bring various connection performances in relation to each other. That will be explained in the following.

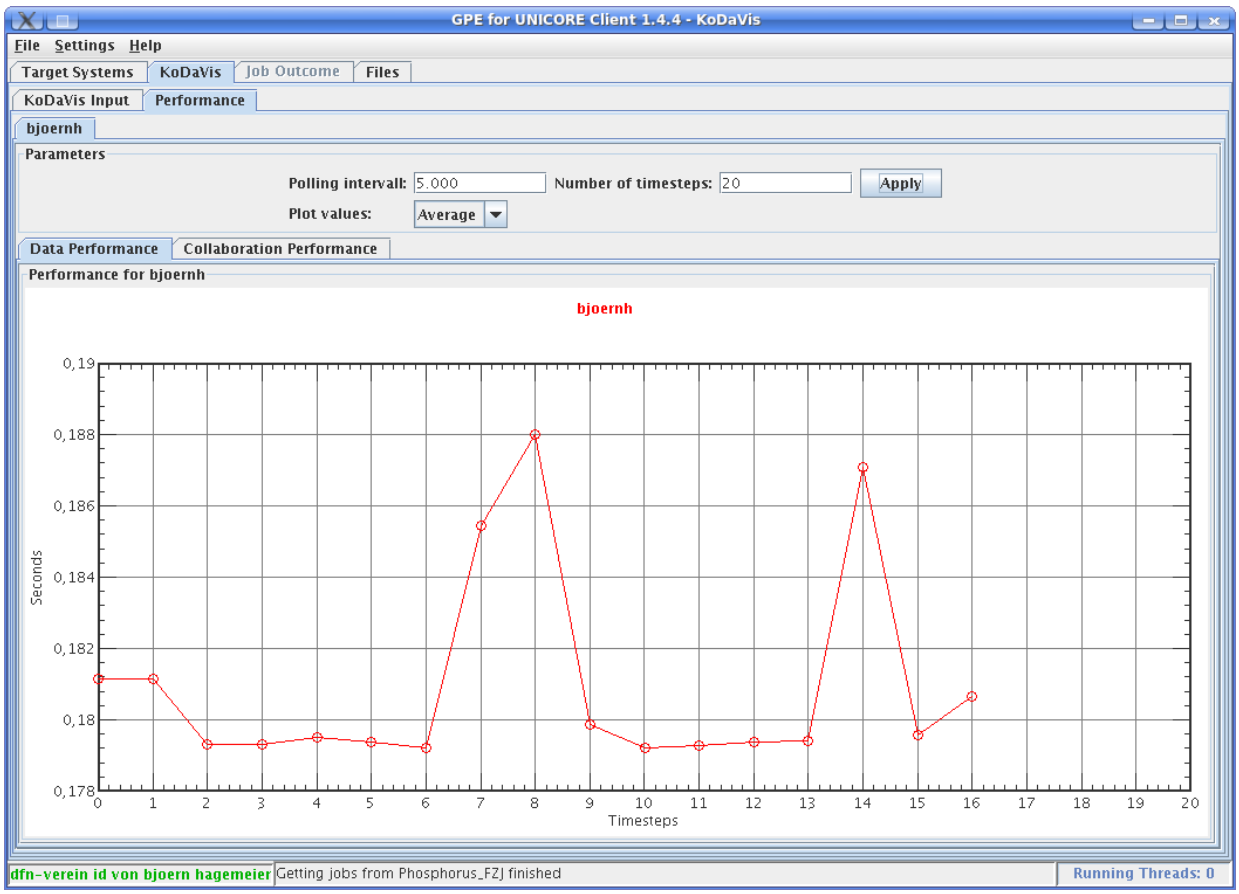


Figure 2.16: Transfer statistics of a data client connection displayed in the UNICORE client.

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 needed be the participant could be 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 below, you can see the time for each of these data transfers.

Figure 2.17 shows a session where initially only a client at FZJ has been connected and retrieves data initially (peak at time step 3). Later on, a client from PSNC joins the session (time step 14), and then both start to query data from the server regularly (time steps 14 – 20). Each time step has duration of 5 seconds. Data transfer rates to the client at FZJ is about the same as the transfer rate for the single client shown in figure 2.16. All data transferred to PSNC takes about twice as long as the data transferred to FZJ. This behavior has been verified with a single client connected form PSNC and thus isn't related to multiple client connections.

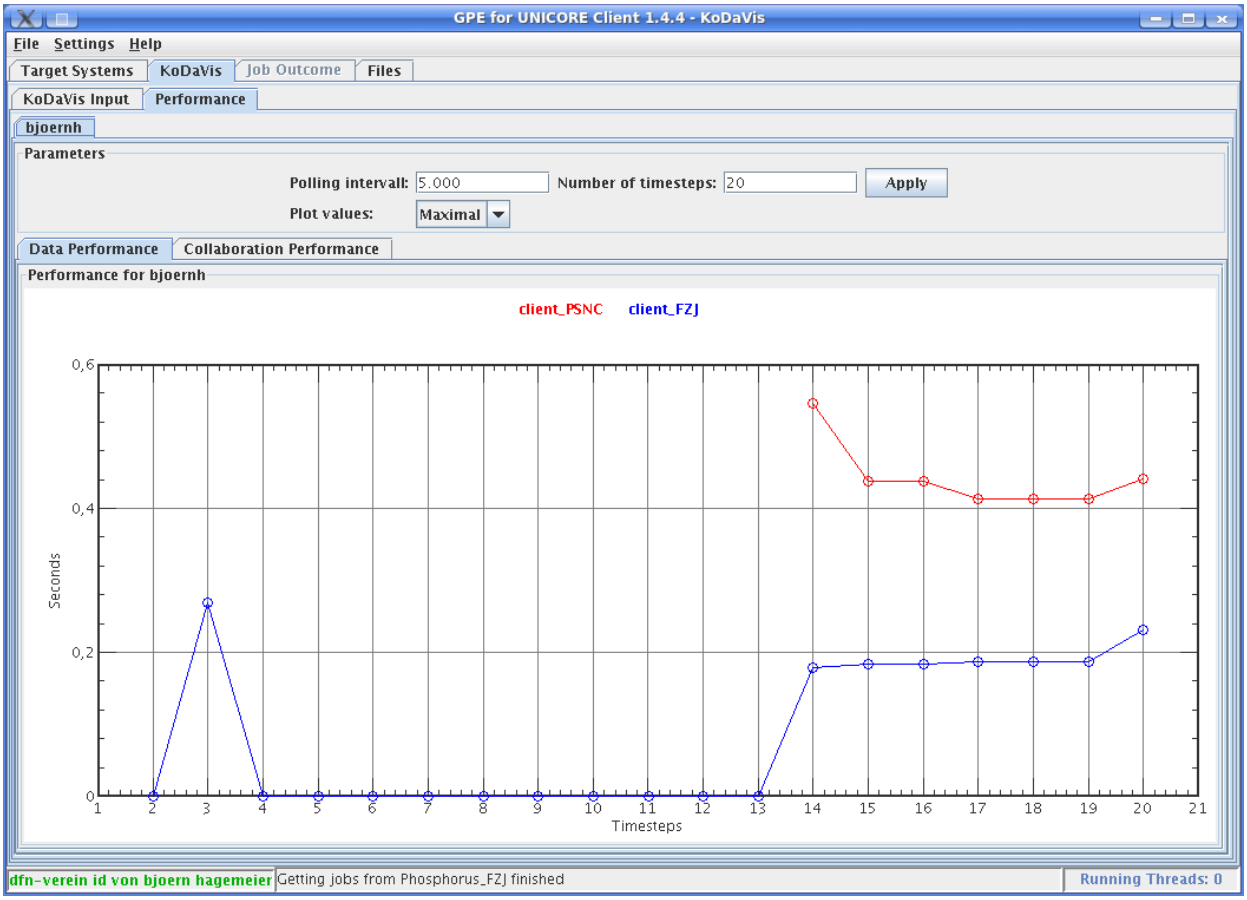


Figure 2.17: Transfer statistics of a data client connection displayed in the UNICORE client.

Figure 2.18 displays the performance of the collaboration server connections. As one can see, clients showed very poor performance in an alternating pattern. It is not quite clear yet, what caused these performance problems. However, considering the fact that the apparently problematic links alternate, problems could well be located in the collaboration server.

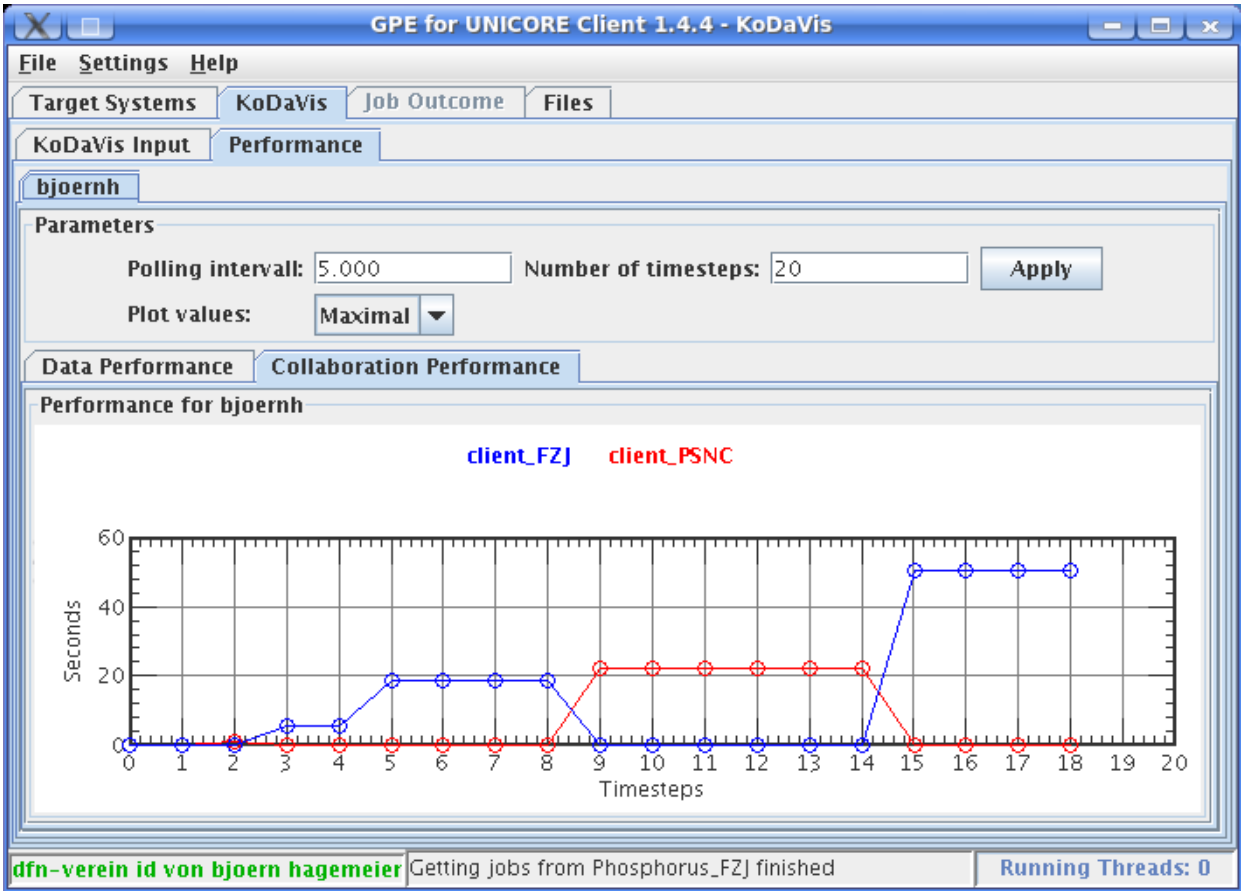


Figure 2.18: Transfer statistics of two collaboration client connections displayed in the UNICORE client.

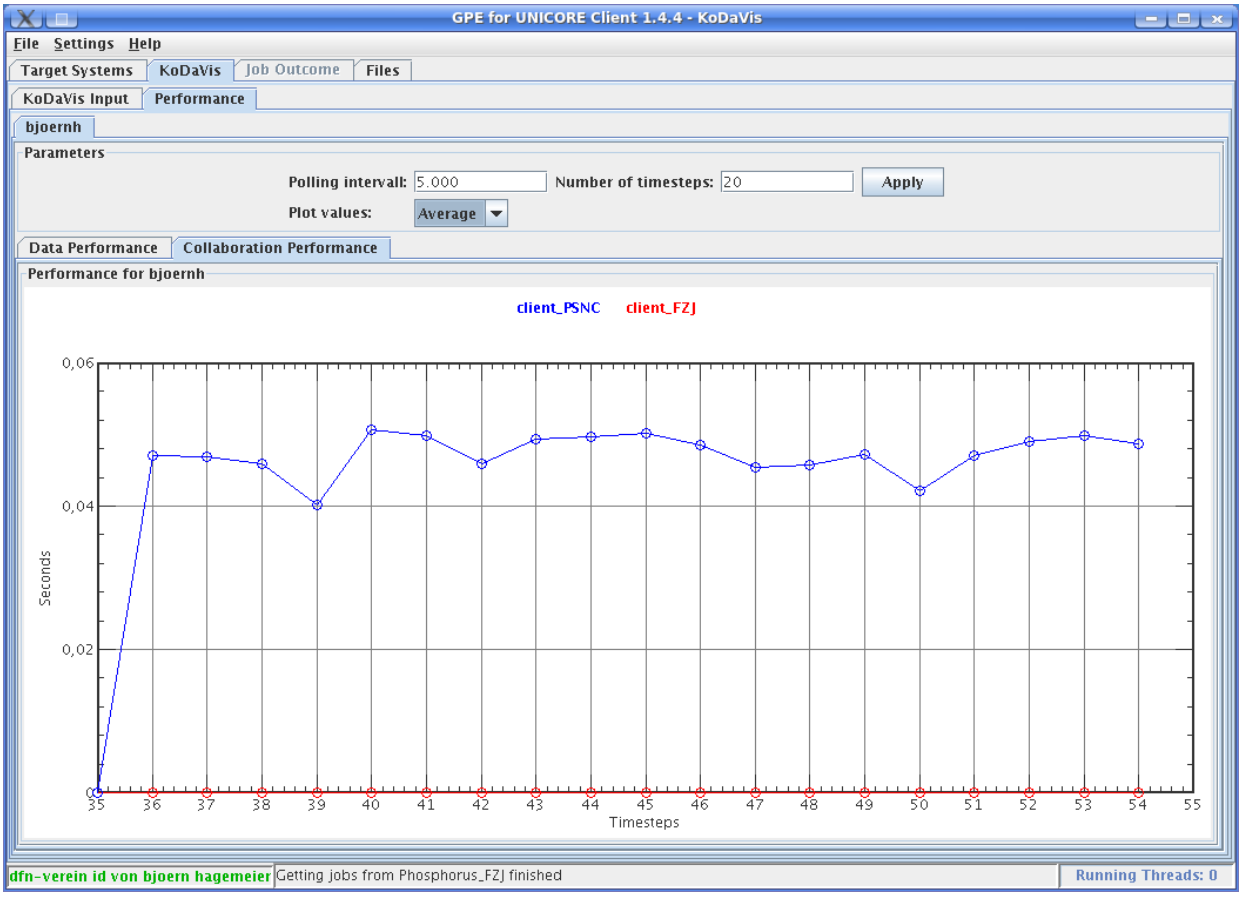


Figure 2.19: Transfer statistics

Figure 2.19 displays the performance of the collaboration part of the session. Collaboration messages are rather small and thus latency is dominating over bandwidth in this case. The data isn't surprising in any way, as the timing reflects the same values as they're returned by an ordinary ICMP Ping message. The precise Ping measures for the same link as displayed above are 23ms from FZJ to PSNC and 0.4ms from FZJ to the client also located at FZJ. This explains why in the diagram the client at FZJ is shown with no latency.

It was already mentioned in the report about the first testbed experiments [D6.5], that the serial version of the data server poses a bottle neck as the number of participants in a collaborative session grows. At that time, the parallel version of the data server was still immature and could not be used to improve performance. This has changed since, and better results can now be achieved through exploiting the parallel data server.

2.2.4 Summary

During the time of testing, it was experienced number of problems with the reservation system. Sometimes link reservation was not possible. At other occasions, reserved links would fail for no apparent reason. We are still investigating this behavior in cooperation with WP1 and WP6.

It could be shown that the newly developed features, 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.

2.3 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-Switch-Geant2+-Viola-FhG. On this connection a dedicated 1 Gb lightpath is available for TOPS. Figure 2.20 shows a picture of the data seen at the display site, the network configuration used for TOPS is described in figure 2.21. 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 2.20: 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 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.

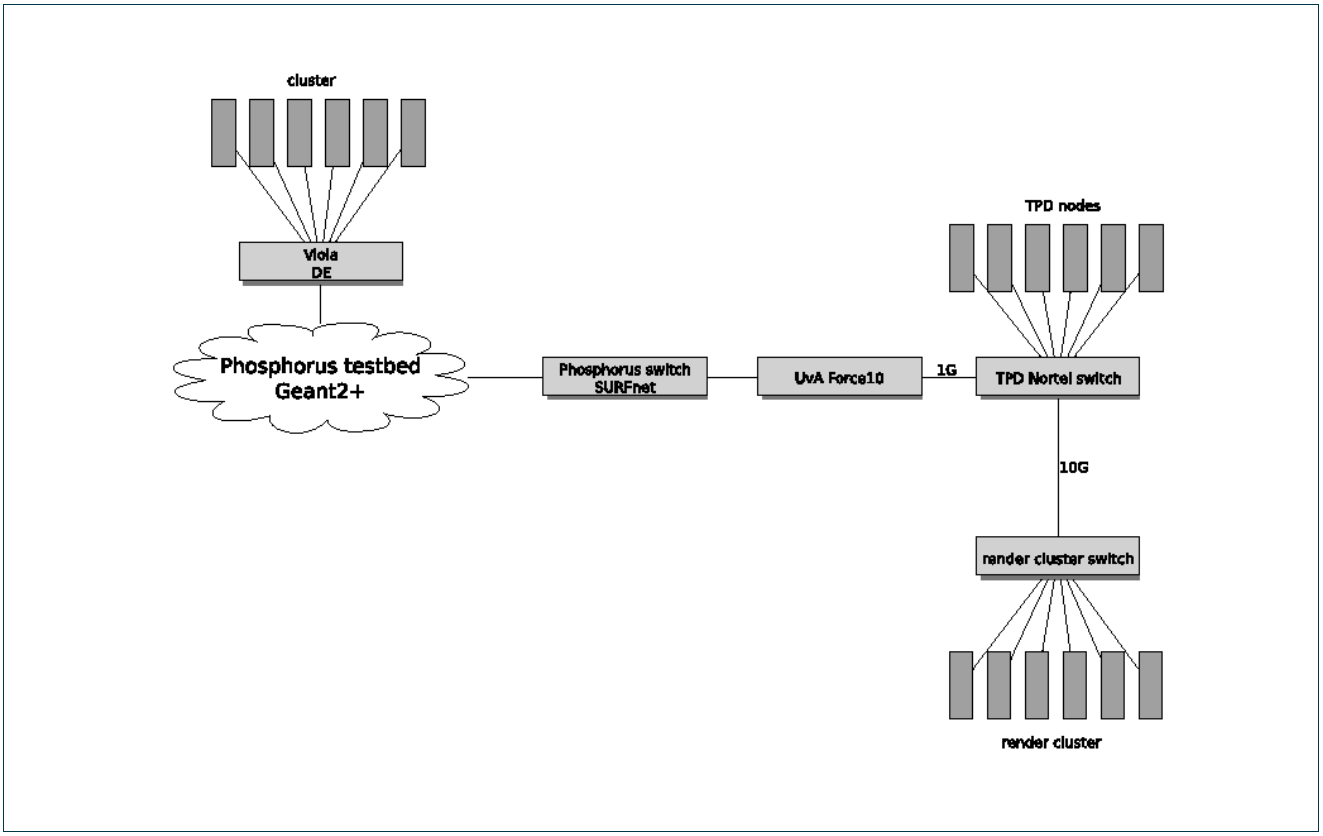


Figure 2.21: TOPS network configuration between SARA and Fraunhofer FhG using the PHOSPHORUS test bed

As it is shown in figures 2.22 and 2.3, TOPS is able to use the complete bandwidth that was made available for the experiments during the whole timeframe. 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.

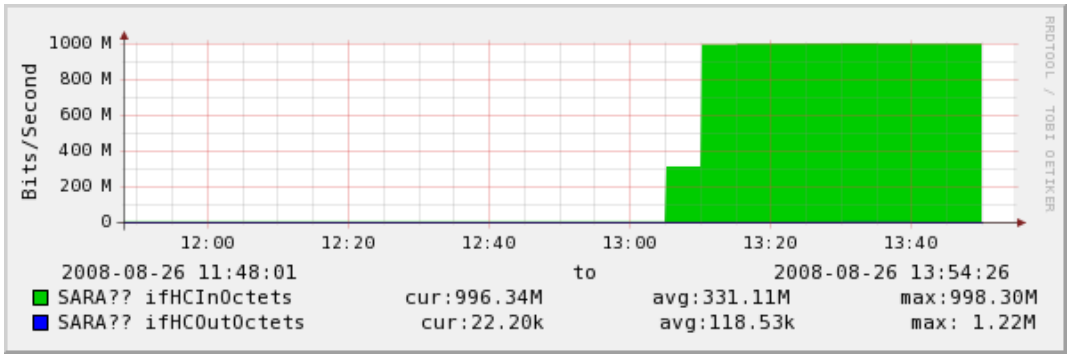


Figure 2.22: Network usage statistics of TOPS application at the SARA side

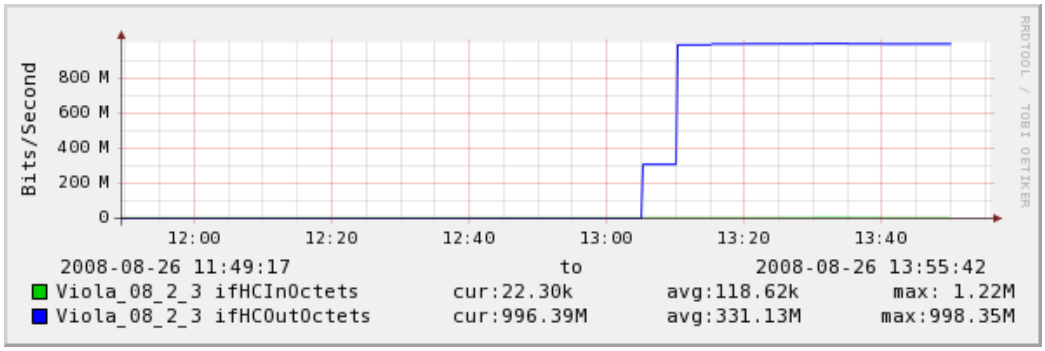


Figure 2.23: Network usage statistics of TOPS at the Géant2+ (Viola test bed connection) side

The next step in the adaptation of TOPS will be to implement an interface to the DRAC user controlled lightpath provisioning system. DRAC is one of the middleware implementations that is implemented in the PHOSPHORUS project.

2.4 DDSS

Distributed Data Storage Systems (DDSS) are widely used to transport, exchange, share, store, backup/archive and restore data in many scientific and commercial applications. Proposed test scenarios include two DDSS use cases: data transfers performed with use of educational, open-source GridFTP application and backup/archive/restore operations made by a commercial application.

Experiments with DDSS application were planned 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 as shown on Figure 2.24.

General description of the tests has been provided in D3.4, however a more precise description is contained in this document.

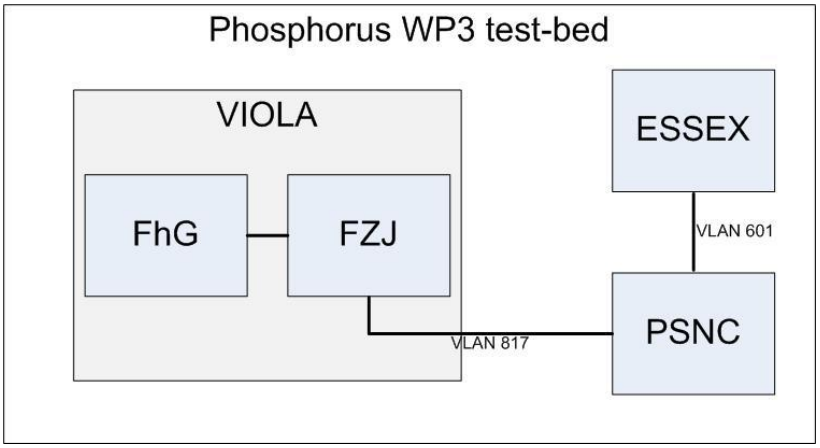


Figure 2.24: DDSS PHOSPHORUS test-bed overview.

2.4.1 Description of experiments

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

- big files,
- lots of small files.

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

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

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

- TCPW (TCP Window Size),
- TCPBUFSIZE (TCP Buffer Size).

Actually, four various test scenarios were prepared for each application,

The range of parameters values used for testing was determined empirically. Prior to main testing, a few test rounds with the applications and based on the tests results were ran, and the range of the parameters values used for further testing we chosen.

The main experiments were controlled by the test automation tools were developed for this purpose. The results of experiments were placed on the web page [www1][www2] hosted in PSNC. The experiments results are divided into two groups: GridFTP and TSM B/A tests. Each PARTICULAR experiment is identified by its launching date. All the collected results were disclosed as line charts, where the X-axis represents value of the parameter modified during the test round and the Y-axis represents the Average Transfer Speed (ATS). The ATS is evaluated from the following formula:

ATS = D / T

Where: D – Total data to transmit [in MB], T – Total transfer time [in sec.].

Note that, Total transfer time contains both the time needed to startup and close the testing application and the data transmission time. However, given that the start up and closing time is relatively short, comparing to the time needed to transmit the data, we find it an insignificant factor while evaluating the application efficiency.

In case of DDSS GridFTP, the experiments were run twice: the first time on the Internet and then on the PHOSPHORUS infrastructure. In that way, we were able to aggregate and compare both tests results. The transmissions were run between PSNC and three other locations: FZJ, FHG and UEssex.

The DDSS B/A experiments were run only over the PSNC-FZJ link, as only two backup/archive servers (one in PSNC, and the other in FZJ) were assigned to PHOSPHORUS tests. Moreover, the experiments were also run locally in PSNC, over a LAN, using the PHOSPHORUS test TSM server and the local client node. The results of local tests are used for comparison with those performed over the PHOSPHORUS link.

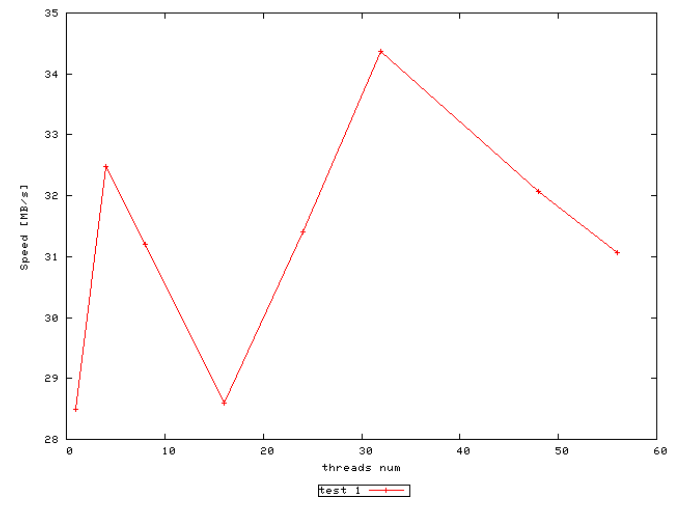
2.4.2 DDSS GridFTP tests results

The following figures depict the results acquired from GridFTP application tests. Figures 2.26-2.28 depict the tests parameters and results taken from the experiments in which the performance vs. number of parallel data transmission threads was examined. Each figure consists of two charts which represent the PHOSPHORUS tests results (left side of the figure) and the Internet tests results (right side of the figure).

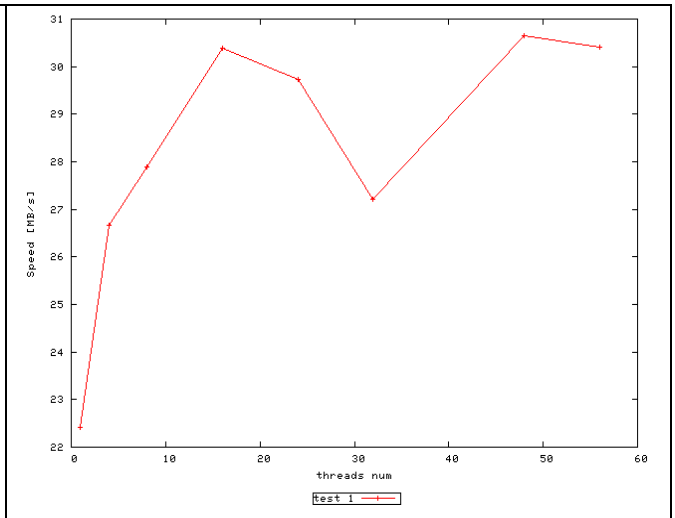
From	To	Figure	RTT Phosphorus [ms]	Distance [km]	Block size [B]
PSNC	FZJ	2.26 (big), 2.29 (small)	28.8	858	256000
PSNC	FhG	2.27 (big), 2.30 (small)	30.5	839	256000
PSNC	Essex	2.28 (big), 2.31 (small)	37.4	1277	256000

Table 2.25: DDSS Grid FTP application test cases (performance vs. threads number scenarios) with big and small files.

Big files transmission

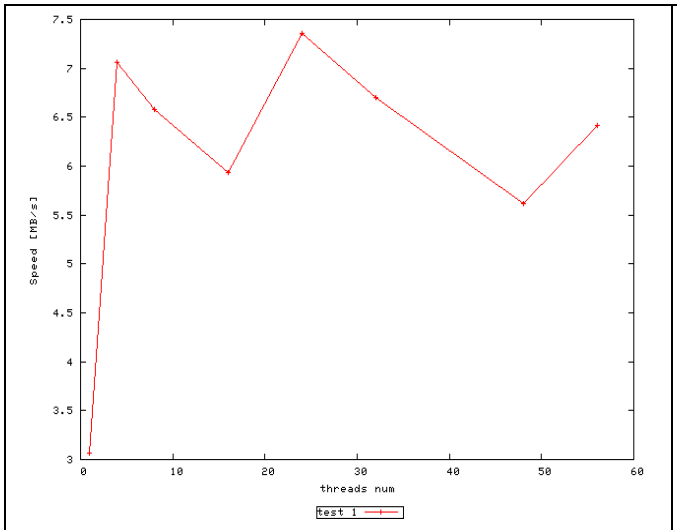


PHOSPHORUS

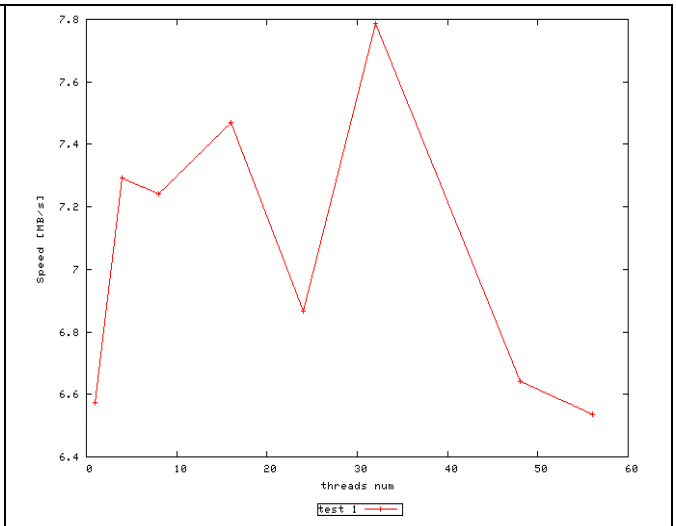


Internet

Figure 2.26: DDSS GridFTP PSNC-FZJ tests results (performance vs. number of threads) with big files.

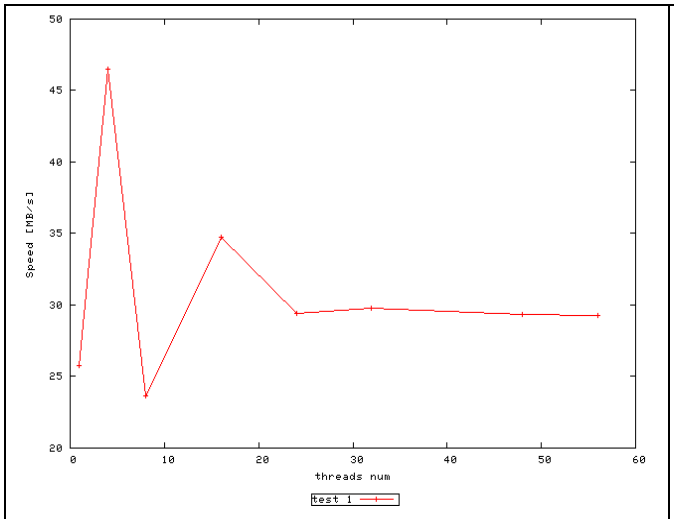


PHOSPHORUS

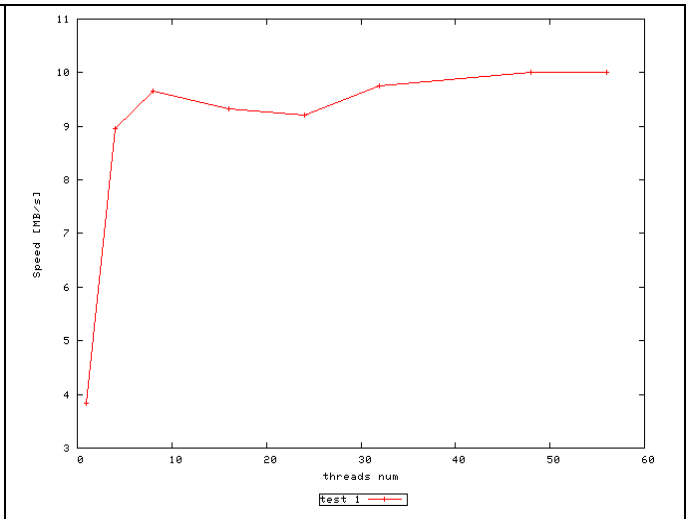


Internet

Figure 2.27: DDSS GridFTP PSNC-FHG tests results (performance vs. number of threads) with big files.



PHOSPHORUS



Internet

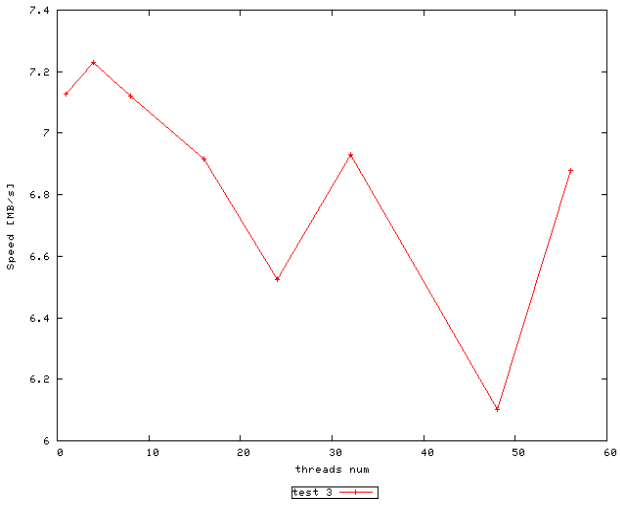
Figure 2.28: DDSS GridFTP PSNC-Essex tests results (performance vs. number of threads) with big files.

The results 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 more effectively than the regular Internet links.

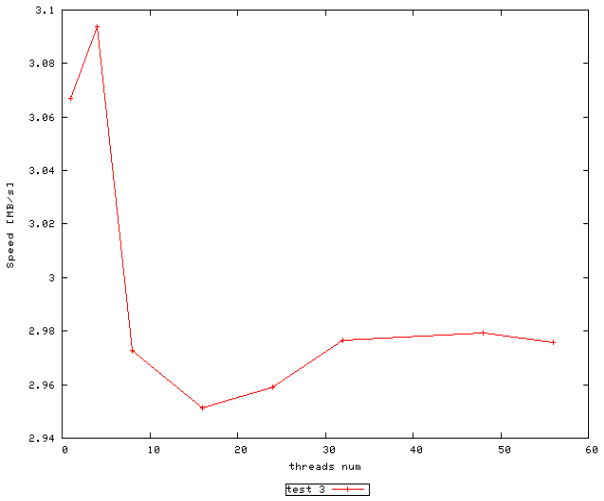
The average transmission bandwidth observed at the application level while running big files transmissions between PSNC and FZJ reaches the level of 35 MB/s (vs. up to 30MB/s over the Internet). In case of transferring big files between PSNC and Essex, the advantage of reserved links is much more significant: almost 47MB/s in PHOSPHORUS (vs. 10MB/s over the Internet). The exception from the general trend is the result acquired for PSNC-FHG transmissions – the DDSS GridFTP performance is poor for both PHOSPHORUS and Internet link (around 7.5 MB/sec). This is caused by reservation problems on the way between PSNC and FHG. Note, that in case of PSNC-Essex transmission the benefit of using a reserved link is much more noticeable than in case of PSNC-FZJ test. This is caused by the fact, that the regular Internet connection used for PSNC-FZJ transmissions is much more efficient than the Internet link between PSNC and Essex. Therefore, there is more space for the bandwidth optimisation in the former case.

The big file transmission scenarios are mainly bandwidth-dependent. Therefore, we may conclude that, in many cases, PHOSPHORUS bandwidth reservation feature gives significant improvement of links efficiency, comparing to regular Internet links. Note, that the Internet traffic was in fact routed through the Geant, high-quality links. Therefore the relative differences between the PHOSPHORUS and Internet links observed in some test cases might be bigger, if PHOSPHORUS links would be compared to commercial Internet.

Small files transmission

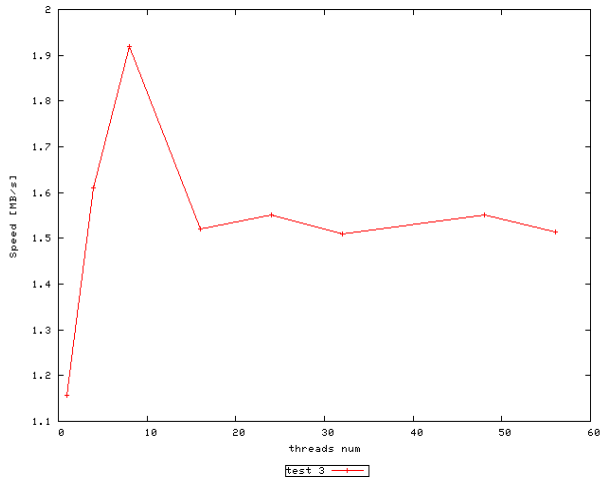


PHOSPHORUS

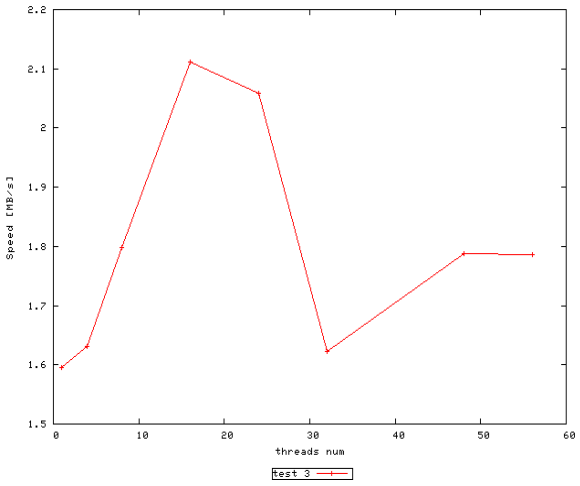


Internet

Figure 2.29: DDSS GridFTP PSNC-FZJ tests results (performance vs. number of threads) with small files.

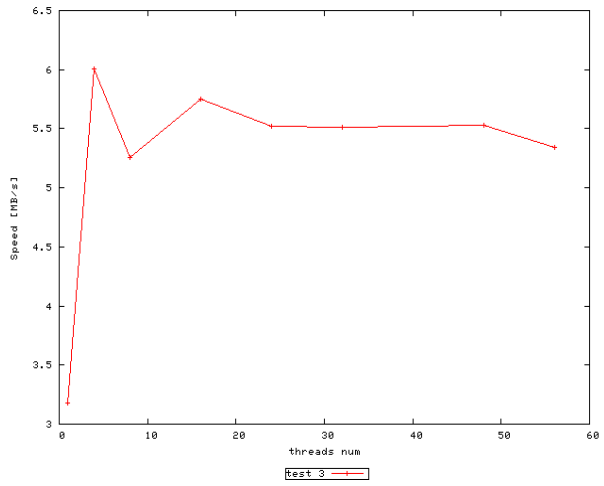


PHOSPHORUS

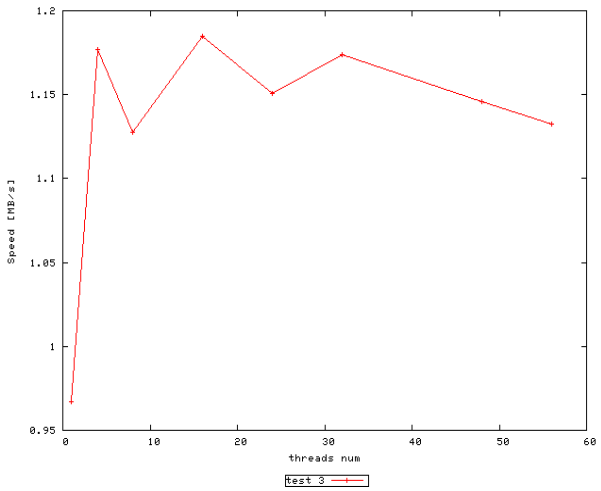


Internet

Figure 2.30: DDSS GridFTP PSNC-FHG tests results (performance vs. number of threads) with small files.



PHOSPHORUS



Internet

Figure 2.31: DDSS GridFTP PSNC-ESSEX tests results (performance vs. number of threads) with small files.

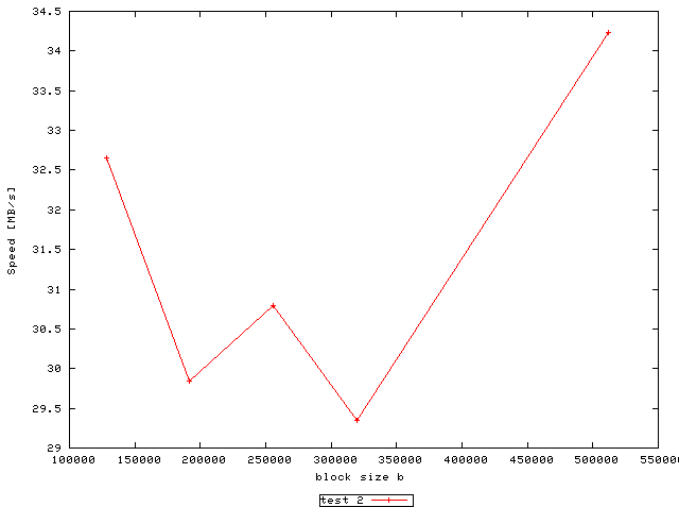
Small files transmission scenarios shown, that the reservation of the optical links using a PHOSPHORUS NRPS, make possible gaining the average transmission speed high, comparing to the public Internet setup. Reserving the bandwidth using PHOSPHORUS NRPS helped to improve the transmission efficiency significantly: i.e. from 3.1MB/s to 7.2MB/s for PSNC-FZJ tests and from 1.18MB/s to 6.0MB/s for PSNC-Essex experiments. Again, the PSNC-FHG was the exception from the general trend.

Note also, that performance trends depicted in Figures 2.26-2.31 confirm the initial beliefs, i.e. that the total bandwidth available for DDSS Grid FTP data transmissions can be optimised by using the appropriate number of parallel data transmission threads. Note that the optimal parallelism levels vary from relation to relation. This results from the fact that the same TCP/IP r/w window size parameter values were used in whole DDSS test-bed while the distances between particular test-bed nodes were different. Therefore, various numbers of threads were optimal for a given relation.

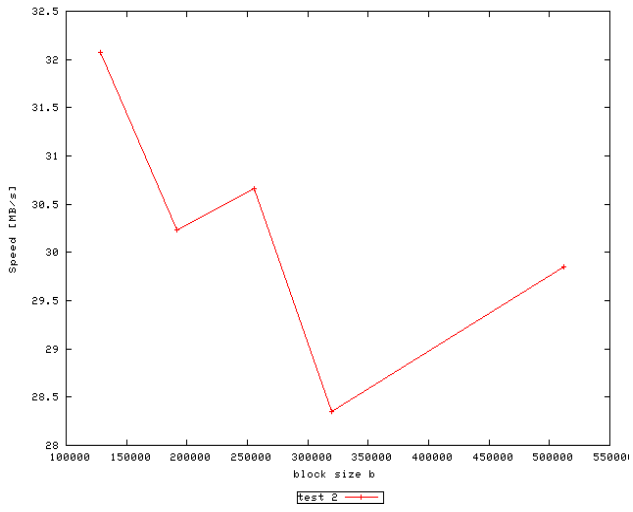
Figures 2.32-2.35 show the test parameters and the results acquired by the tests run for the various data block sizes used for data transfer.

From	To	Figure	RTT [ms]	Distance [km]	Number of Threads
PSNC	FZJ	2.33 (big), 2.36 (small)	28,8	858	8
PSNC	FHG	2.34 (big), 2.37 (small)	30,5	839	8
PSNC	Essex	2.35 (big), 2.38 (small)	37.4	1277	8

Table 2.32: DDSS Grid FTP application test cases (performance vs. block size scenarios)

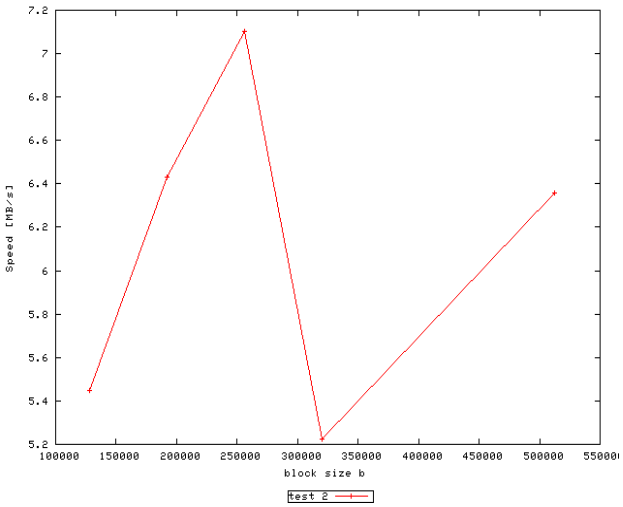


PHOSPHORUS

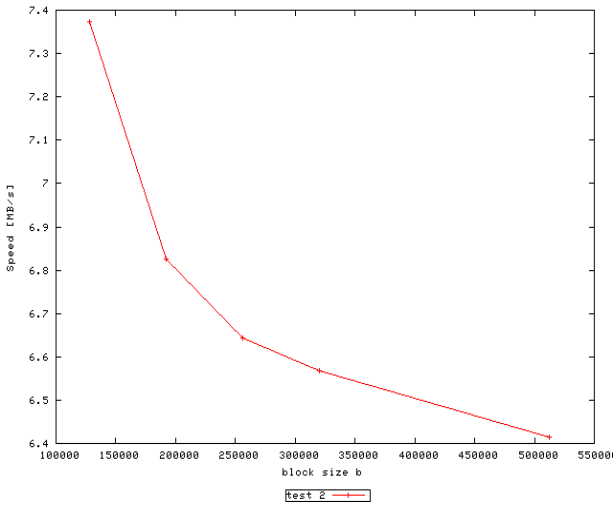


Internet

Figure 2.33: DDSS GridFTP PSNC-FZJ tests results (performance vs. block size) with big files.

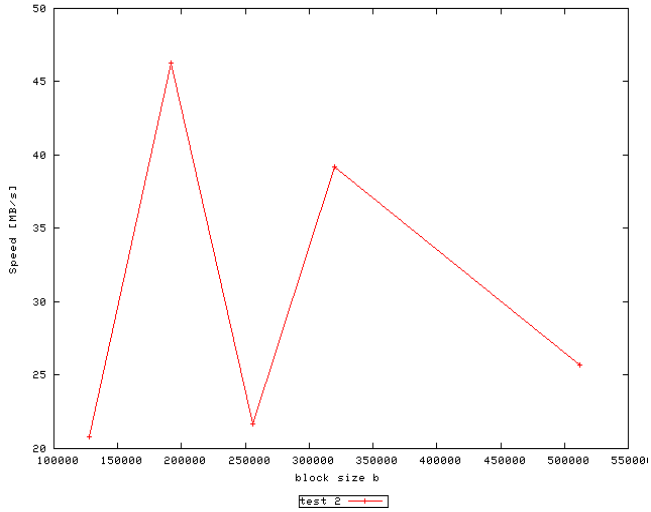


PHOSPHORUS

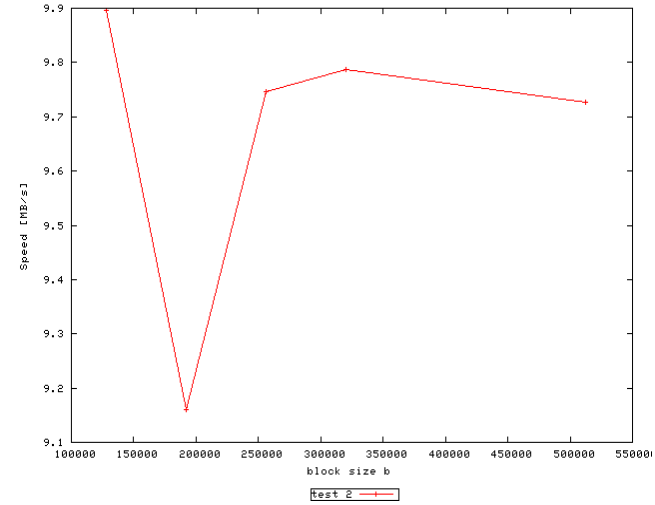


Internet

Figure 2.34: DDSS GridFTP PSNC-FHG tests results (performance vs. block size) with big files.



PHOSPHORUS

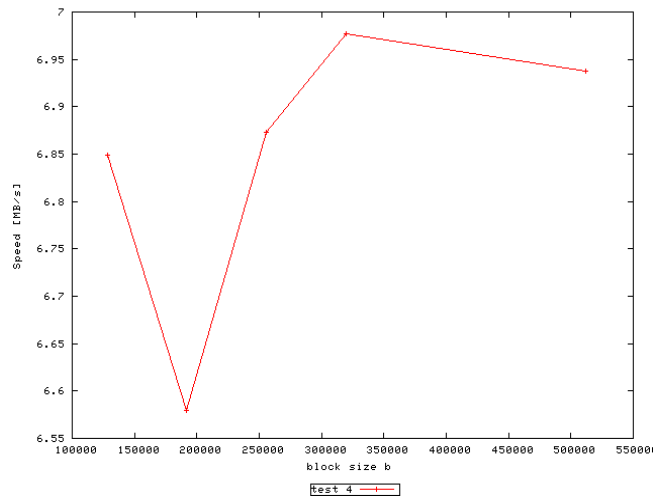


Internet

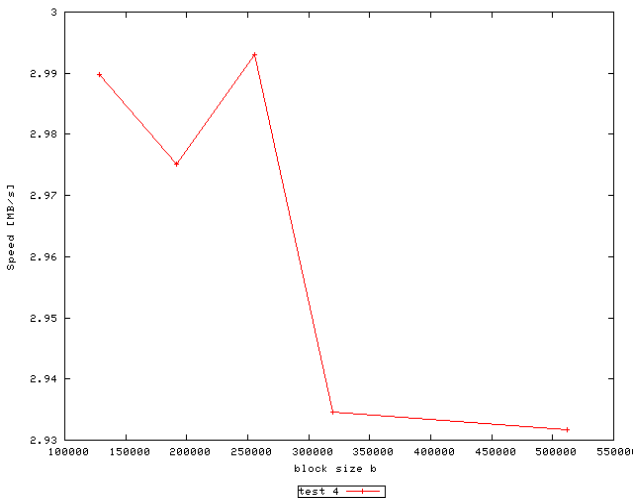
Figure 2.35: DDSS GridFTP PSNC-Essex tests results (performance vs. block size) with big files.

The results acquired from performance vs. block size test scenarios confirm the performance features of the PHOSPHORUS test network seen in the previously presented numbers. The average transmission bandwidth in the PHOPSHORUS network, observed by DDSS GridFTP client for big files transmissions between PSNC and FZJ, reaches the range of 35Mbytes/sec (vs. 32.5MB/s over the Internet). In the same time, the average transmission speed between PSNC and ESSEX reaches 46Mbytes/s (vs. 9.9MB/s over the Internet). Similar to previous cases, the PSNC-FHG link shows some reservation issues.

The big files transmission scenarios results again confirm the effectiveness of the PHOSPHORUS bandwidth reservation feature. When comparing to regular Internet links we see the advantages of the optical reserved links. This advantages is more significant in case of PSNC-Essex tests, as the Internet link used between these locations is less efficient than the PSNC-FZJ Internet connection (therefore there is more space for optimisations in the former case).

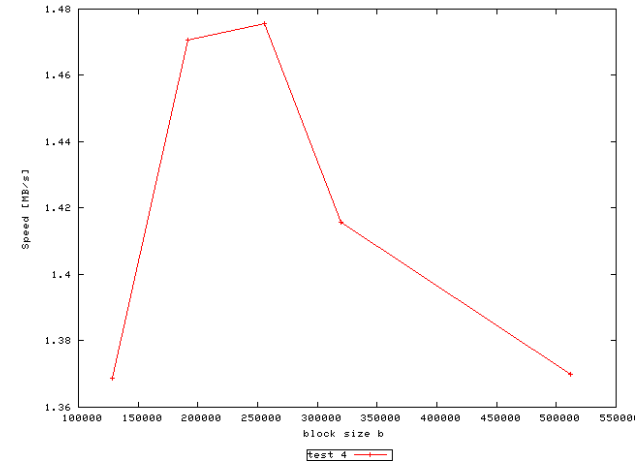


PHOSPHORUS

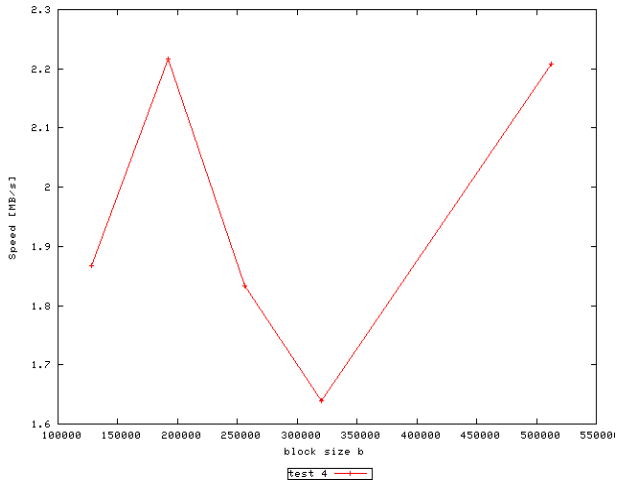


Internet

Figure 2.36: DDSS GridFTP PSNC-FZJ tests results (performance vs. block size) with small files.

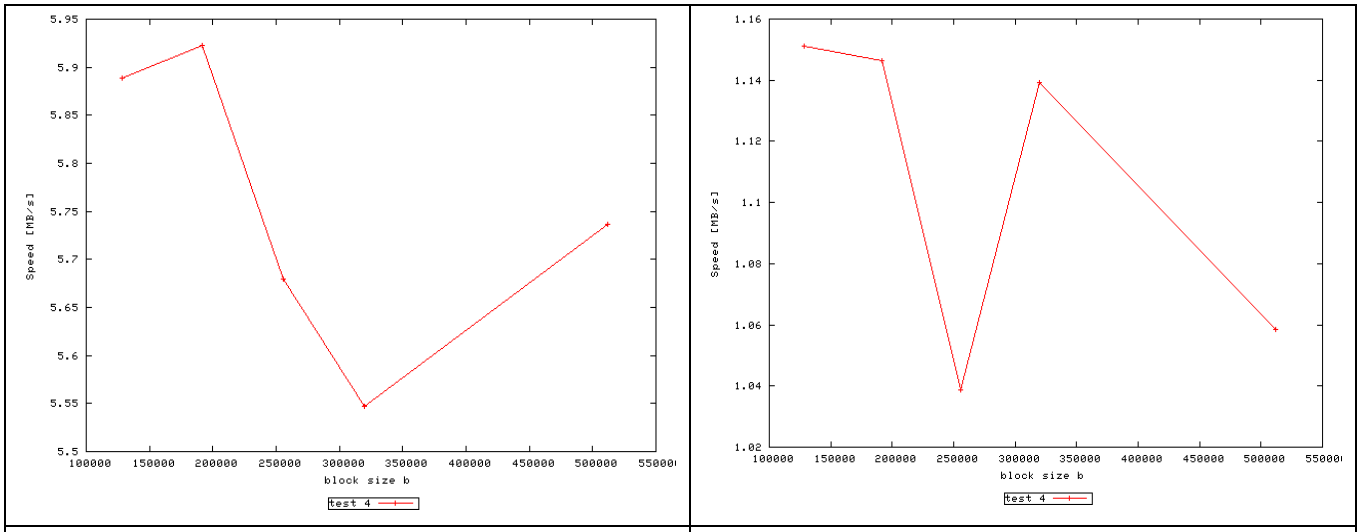


PHOSPHORUS



Internet

Figure 2.37: DDSS GridFTP PSNC-FHG tests results (performance vs. block size) with small files.



PHOSPHORUS	Internet

Figure 2.38: DDSS GridFTP PSNC-Essex tests results (performance vs. block size) with small files.

Small files transmission scenarios gain much lower average transmission speeds, as they are more latency-dependent and some additional processing is required on the client and the server side while making backup of numerous files. However, their results over the PHOSPHORUS link are again much better than the results acquired over the Internet: 7.0MB/s vs. 3.0 MB/s (PSNC-FZJ) and 5.9MB/s vs. 1.15MB/s (PSNC-Essex).

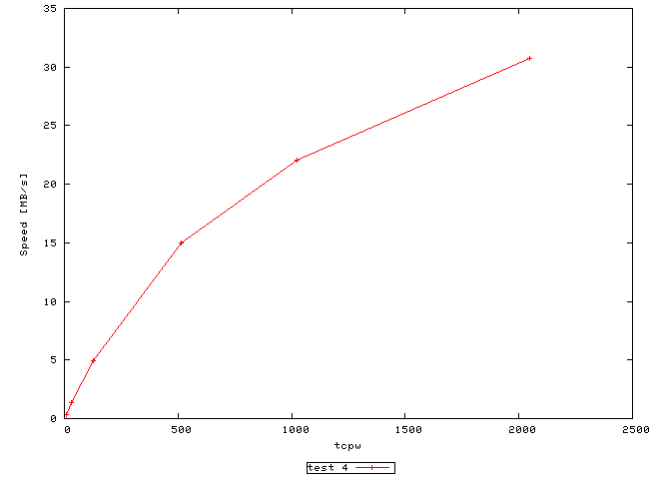
Figures 2.36-2.38 show also that it is hard to find an optimal block size for both big and small files transmission. The performance trends differ a lot in particular transmission relations. This shows, that the effectiveness of transferring the data transmission using DDSS GridFTP is related mainly to the level of parallelism used while transmission.

2.4.3 DDSS Backup/Archive tests

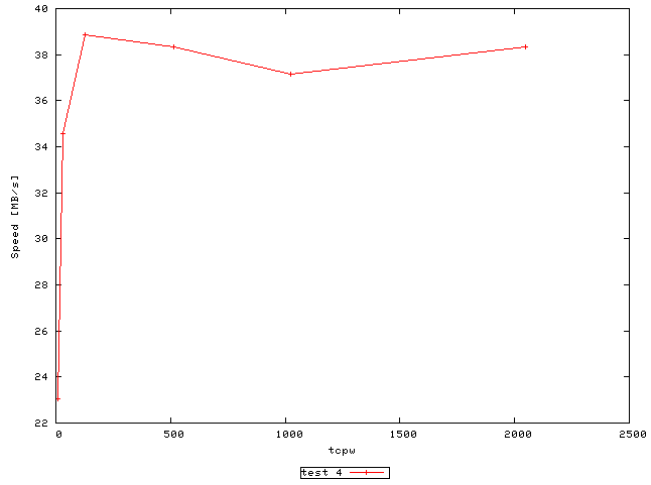
The following table 2.4 and figures 2.40-2.43 depict the results acquired from DDSS B/A application tests. They depict the results taken from the experiments in which the TSM application backup process efficiency was examined using various TCP parameters. Each figure consists of two charts which represent the PHOSPHORUS tests results (left side of the figure) and local tests results (right side of the figure).

Figure [file size]	Parameter	Constant [KB]
2.40 (big), 2.42 (small)	TCPW	TCPBUFSIZE=32
2.41 (big), 2.43 (small)	TCPBUFSIZE	TCPW=512 (local), TCPW=2048 (PHOSPHORUS)
2.40 (big), 2.42 (small)	TCPW	TCPBUFSIZE=32

Table 2.39: DDSS B/A application test cases with big files.

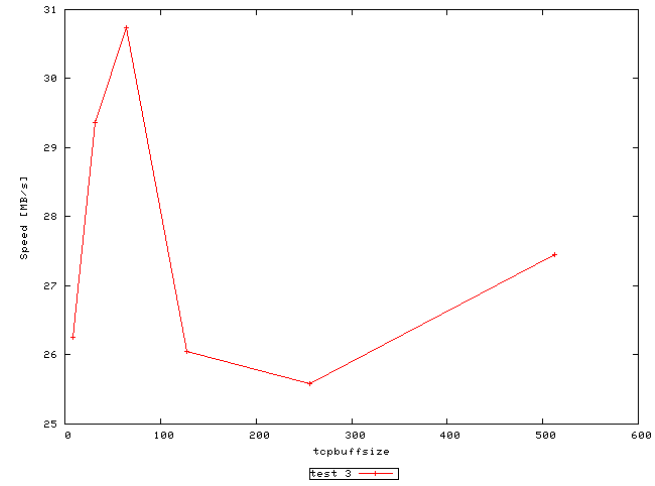


PHOSPHORUS

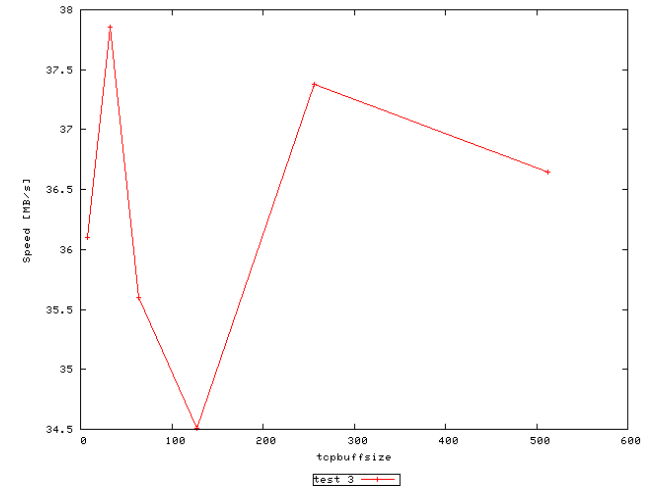


Local

Figure 2.40: DDSS B/A PSNC-FZJ vs. Local tests results (performance vs. TCP window size) with big files.

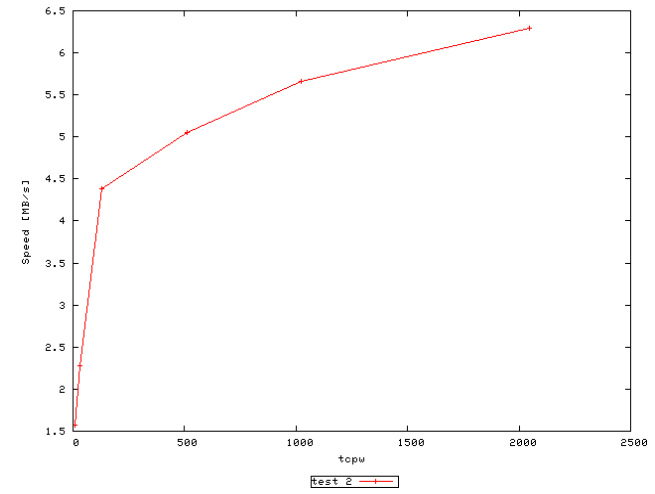


PHOSPHORUS

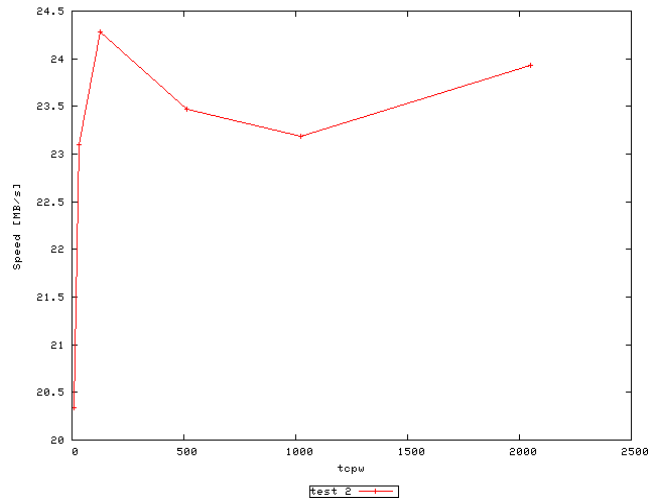


Local

Figure 2.41: DDSS B/A PSNC-FZJ vs. Local tests results (performance vs. TCP buffer size) with big files.

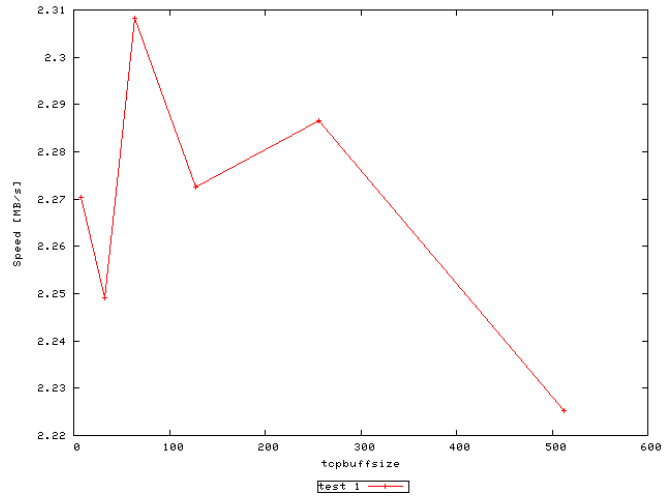


PHOSPHORUS

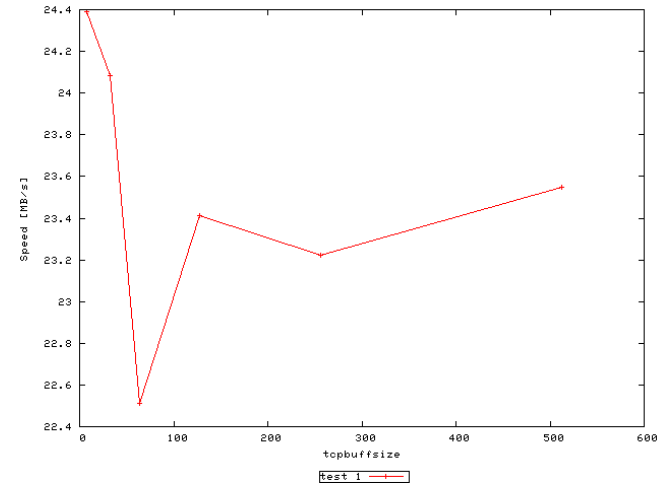


Local

Figure 2.42: DDSS B/A PSNC-FZJ vs. Local tests results (performance vs. TCP window size) with small files.



PHOSPHORUS



Local

Figure 2.43: DDSS B/A PSNC-FZJ vs. Local tests results (performance vs. TCP buffer size) with small files.

The DDSS B/A tests result confirmed that the link reservation feature provided by PHOSPHORUS network make this network suitable for bandwidth-consuming applications. Note, that the results of big files transmission over long-distance FZJ-PSNC PHOSPHORUS link (~30 MB/sec, see Figure 2.40 and Figure 2.41) are insensibly worse than the numbers acquired during local tests over the LAN network in PSNC (~39 MB/s).

Note also that, the tests results acquired while making backup copies of numerous small files are not impressive (6.5 MB/sec over FZJ-PSNC link) – see Figures 2.42 and 2.43. The reasons for that are high latencies observed in the long-distance network link and the fact, that the TSM application is not efficient while performing numerous small files. Creation of backup copy requires separate database transaction for each file on the server side and file system I/O processing in the client.

It is not surprising that TCP window size parameter have significant influence on transmission speed of big files – on long distance links with high latencies, the transmission efficiency grows significantly (from less than 5MB/s to almost 30MB/s) while increasing the window size (see Figures 2.40 and 2.42). Another examined transmission parameter, i.e. TCP buffer size, has not such a significant influence on the backup performance (~20%), however, it seems that its optimal value lies around 64kB for both kinds of tests: i.e. small and big files backups (see Figures 2.41 and 2.43).

2.4.4 DDSS Tests Summary

The DDSS GridFTP tests results show that the PHOSPHORUS optical network links reservation features make significant improvement of the performance of large data transmission possible. From the results of the big files transmission, it can be seen that PHOSPHORUS reservation features can provide a high transmission bandwidth to the data transfer application. From this part of the testing scenarios, it can be derived that using a high level of parallelism one may overcome the transmission limits caused by high latency in both PHOSPHORUS and Internet setup. However, it can also be observed that, the high bandwidth assured by the PHOSPHORUS reservation features guarantee significantly higher transfer rates than the Internet link can provide, nevertheless of the number of the transmission threads.

Similarly, the DDSS Backup/Archive tests results show, that the PHOSPHORUS links reservation help 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.

3 Conclusions

All PHOSPHORUS applications participated in this second round of application experiments to validate their enhancements after the first phase of experiments. Although, due to problems with the VIOLA testbed, tests have been largely delayed. We were unable to use reserved links between the nodes at PSNC and FZJ for quite a while. After that, links have been failing during their reserved runtime. We are evaluating these issues in cooperation with WP1 and WP6.

Tests, which were mainly influenced by these problems were those of KoDaVis and DDSS, as they were using parts of the VIOLA testbed for their tests. The WISDOM application could be tested successfully inside the VIOLA testbed, between FZJ and FhG, where no reservation of any links is required.

Finally, the following results after the final experiments can be summarized:

- WISDOM: The client side of the WISDOM application is now using the UNICORE 6 rich client, which is a great improvement compared to the manual engagement used so far. Additionally many necessary adjustments to the software environment of both the WISDOM applications AutoDock and FlexX are made to realize a basis for an automatic workflow handling by the UNICORE 6/MSS integration. It is planned to demonstrate the application at the Supercomputing 2008 and ICT 2008 events in November. More visible network performance is planned to be shown at those occasions.

- KoDaVis: KoDaVis has been enhanced by flexible performance monitoring, enabling users to identify performance bottlenecks and take appropriate actions during collaborative visualization sessions. Tests between the sites of PSNC and FZJ showed the utility of these new functions in the client and server layers.

- TOPS: TOPS has been proven to be able to make use of all available bandwidth that's given to it.

- DDSS: The DDSS application has been tested extensively between a number of participating sites. Various network and application parameters have been tried during the tests to find optimal settings for different scenarios of small and big file transfers. It was shown that the use of reserved bandwidth has an advantage over using public internet connections.

4 References

[D3.3]	Report on initial Adaption of Applications
[D3.4]	Report on middleware extensions and implementation
[D3.5]	Final Report on Achievements and Results of the first 18 Months' Period
[D6.5]	Application and Middleware Testing Report
[D6.6]	Plan of Testing
[www1]	http://node01.PHOSPHORUS.man.poznan.pl/gridftp/
[www2]	http://node01.PHOSPHORUS.man.poznan.pl/tivoli_fzj/

5 Acronyms

AAA	Authentication, Authorisation, Accounting
DDSS	Distributed Data Storage Systems
DDSS B/A	Backup/Archive application used in the PHOSPHORUS testbed as one of the DDSS applications
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 host 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	Intelligent Network Caching Architecture
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 (UNIform 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 Silico Docking On Malaria