European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

Integration of ISS into the VIOLA Meta-Scheduling Environment

Kevin Cristiano, Pierre Kuonen
{kevin.cristiano, pierre.kuonen}@eif.ch
Ecole d'Ingénieurs et d'Architectes, Switzerland

Ralf Gruber, Vincent Keller, Michela Spada, Trach-Minh Tran
{ralf.Gruber, vincent.Keller}@epfl.ch
{michela.spada, trach-minh.tran}@epfl.ch
Ecole Polytechnique Fédérale, Switzerland

Sergio Maffioletti, Nello Nellari, Marie-Christine Sawley
{sergio.maffioletti, nello.nellari, sawley}@cscs.ch
Swiss National Supercomputing Centre, Switzerland

Oliver Wäldrich, Wolfgang Ziegler
{oliver.waeldrich, wolfgang.ziegler}@scai.fraunhofer.de
Fraunhofer Institute SCAI, Germany

Philipp Wieder
ph.wieder@fz-juelich.de
Research Centre Jülich, Germany

CoreGRID Technical Report
Number TR-0025
January 25, 2006

Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Integration of ISS into the VIOLA Meta-Scheduling Environment

Kevin Cristiano, Pierre Kuonen
{kevin.cristiano, pierre.kuonen}@eif.ch
Ecole d'Ingénieurs et d'Architectes, Switzerland

Ralf Gruber, Vincent Keller, Michela Spada, Trach-Minh Tran
{ralf.Gruber, vincent.Keller}@epfl.ch
{michela.spada, trach-minh.tran}@epfl.ch
Ecole Polytechnique Fédérale, Switzerland

Sergio Maffioletti, Nello Nellari, Marie-Christine Sawley
{sergio.maffioletti, nello.nellari, sawley}@cscs.ch
Swiss National Supercomputing Centre, Switzerland

Oliver Wäldrich, Wolfgang Ziegler
{oliver.waeldrich, wolfgang.ziegler}@scai.fraunhofer.de
Fraunhofer Institute SCAI, Germany

Philipp Wieder
ph.wieder@fz-juelich.de
Research Centre Jülich, Germany

CoreGRID TR-0025

January 25, 2006

Abstract

The authors present the integration of the Intelligent (Grid) Scheduling System into the VIOLA meta-scheduling environment which itself is based on the UNICORE Grid software. The goal of the new, integrated environment is to enable the submission of jobs to the Grid system best-suited for the application workflow. For this purpose a cost function is used that exploits information about the type of application, the characteristics of the system architectures, as well as the availabilities of the resources. This document presents an active collaboration between Ecole Polytechnique Fédérale de Lausanne (EPFL), Ecole d'Ingénieurs et d'Architectes (EIF) de Fribourg, Forschungszentrum Jlich, Fraunhofer Institute SCAI, and Swiss National Supercomputing Centre (CSCS).

1 Introduction

The UNICORE middleware has been designed and implemented in various pro-jects world-wide, for example the German UNICORE Plus project [1], the EU projects EUROGRID [2] and UniGrids [3], or the Japanese NaReGI

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

project [4]. A recently developed extension to UNICORE, the VIOLA Meta-Scheduling Service, strongly increases its functionalities by adding capabilities needed to schedule arbitrary resources in a co-ordinated fashion. This meta-scheduling environment provides the software basis for the VIOLA testbed [5] and offers the opportunity to include proprietary scheduling solutions. The Intelligent (Grid) Scheduling System (ISS) [6] is such a scheduling system. It uses historical runtime data of an application to schedule a well suited computational resources for execution based on the performance requirements of the user. The goal of the work presented here is to integrate the ISS into the meta-scheduling environment to realise a Grid system satisfying the requirements of the SwissGRID. The Intelligent Scheduling System will add a data repository, a broker and an information service to the resulting Grid system. The scheduling algorithm used to calculate the best-suited system is based on a cost function that takes the data collected during previous executions into account describing inter alia the type of the application, its performance on the different machines in the Grid, and their availability.

In the following section, the functions of UNICORE and the Meta-Scheduling Service are shortly presented. Then, the ISS model is introduced followed by a description of the overall architecture which illustrates the integration of the ISS concept into the VIOLA environment (Sections 3 and 4). Section 5 then outlines the processes that will be executed to schedule application workflows in the meta-scheduling environment. Subsequent to the generic process description an ORB5 application example that runs on machines with over 1000 processors is discussed in Section 6. We conclude this document with a summary and a brief outlook on future work.

2 UNICORE and the Meta-Scheduling Service

The basic Grid environment we use for our work comprises the UNICORE Grid system and the Meta-Scheduling Service developed in the VIOLA project. It is not the purpose of this document to introduce these systems in detail, but a short characterisation of both is given in the following two sections. Descriptions of UNICORE's models and components can be found in other publications [1],[7], respective in publications covering the Meta-Scheduling Service [8], [9], [10].

2.1 UNICORE

A workflow is in general submitted to a UNICORE Grid via the UNICORE Client (see Fig. 1) which provides means to construct, monitor and control workflows. In addition the client offers extension capabilities through a plug-in interface, which has for example been used to integrate the Meta-Scheduling Service into the UNICORE Grid system. The workflow then passes the security Gateway and is mapped to the site-specific characteristics at the UNICORE Server before being transferred to the local scheduler.

The concept of resource virtualisation manifests itself in UNICORE's Virtual Site (Vsite) that comprises a set of resources. These resources must have direct access to each other, a uniform user mapping, and they are generally under the same administrative control. A set of Vsites is represented by a UNICORE Site (Usite) that offers a single access point (a unique address and port) to the resources of usually one institution.

2.2 Meta-Scheduling Service

The meta-scheduler is implemented as a Web Service receiving a list of resources preselected by a resource selection service (a broker for example, or a user) and returning reservations for some or all of these resources. To achieve this, the Meta-Scheduling Service first queries selected local scheduling systems for the availability of these resources and then negotiates the reservations across all local scheduling systems. In the particular case of the meta-scheduling environment the local schedulers are contacted via an adapter which provides a generic interface to these schedulers. Through this process the Meta-Scheduling Service supports scheduling of arbitrary resources or services for dedicated times. It offers on one hand the support for workflows where the agreements about resource or service usage (aka reservations) of consecutive parts should be made in advance to avoid delay during the execution of the workflow. On the other hand the Meta-Scheduling Service also supports co-allocation of resources or services in case it is required to run a parallel distributed application which needs several resources with probably different characteristics at the same time. The meta-scheduler may be steered directly by a user through a command-line interface or by Grid middleware components like the UNICORE client through its SOAP interface (see Fig. 1). The resulting reservations are implemented using the WS-Agreement specification [11].

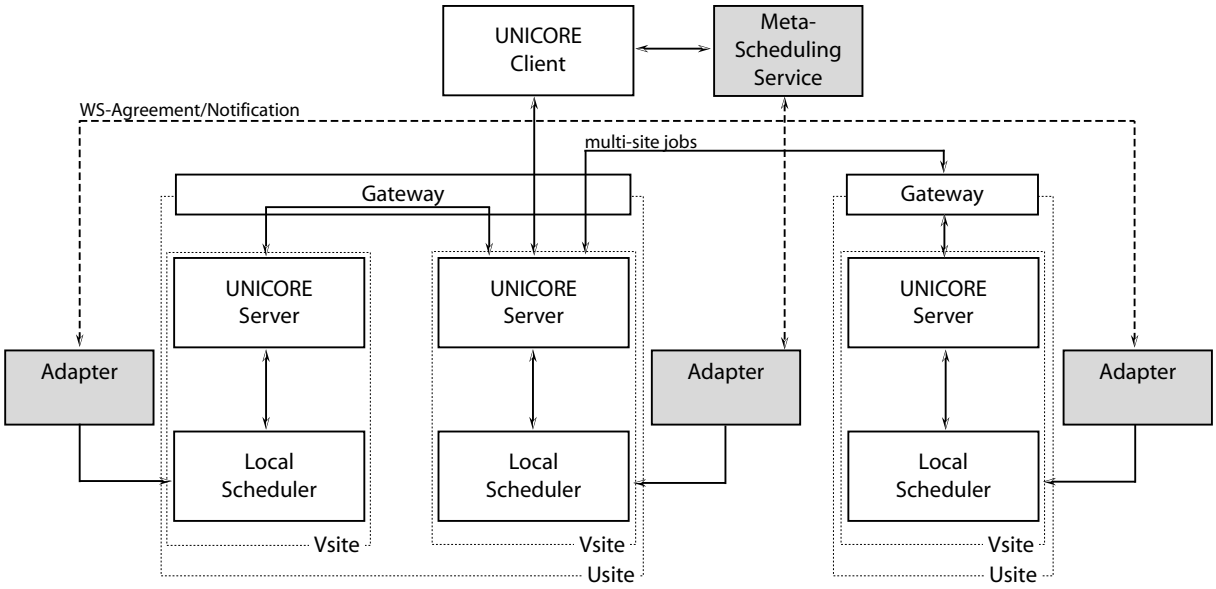

Figure 1: Architecture of the VIOLA Meta-Scheduling Environment

3 Intelligent Scheduling System Model

The main objective of the Intelligent GRID Scheduling System (ISS) project [6] is to provide a middleware infrastructure allowing optimal positioning and scheduling of real life applications in a computational GRID. According to data collected on the machines in the GRID, on the behaviour of the applications, and on the performance requirements demanded by the user, a well suited computational resource is detected and allocated to execute the application. The monitoring information collected during execution is put into a database and reused for the next resource allocation decision. In addition to providing scheduling information, the collected data allows to detect overloaded resources and to pin-point inefficient applications that could be further optimised.

3.1 Application types

The Intelligent Scheduling System model is based on the following application type system:

- **Single Processor Applications** These applications do not need any internode communication. They may benefit from backfilling strategies.
- **Embarrassingly parallel applications** This kind of applications requires a client-server concept. The internode communication network is not important. Seti@Home is an example of an embarrassingly parallel application for which data is sent over the Web.
- **Point-to-point applications** Point-to-point communications typically appear in finite element or finite volume methods when a huge 3D domain is decomposed in sub-domains and an explicit time stepping method or an iterative matrix solver is applied. If the number of processors grows with the problem size, and the size of a sub-domain is fixed, the local problem size is fix. Hence, that kind of applications can run well on a cluster with a relatively slow and cost-effective communication network that scales with the number of processors.
- **Multicast communications applications** The parallel 3D FFT algorithm is a typical example of an application that is dominated by multicast operations. The internode communication increases with the number of processors. Such an application needs a faster switched network such as Myrinet, Quadrics, or Infiniband. If thousands of processors are needed, special-purpose machines such as RedStorm or BlueGene might be required.

- **Multi components applications** Such applications consist of well-separable components, each one being a parallel job with little inter-component interaction. The different components can be submitted to different machines. An example is presented in [13].

The ISS concept is straight-forward: if a scheduler is able to differentiate between the types of applications presented above, it can decide where to run an application. For this purpose the so-called Γ model has been developed which is described in the following.

3.2 The Γ model

In the Γ model described in [12], it is supposed that each component of the application is ideally parallelized, i.e. each task of a component takes the same CPU and communication times.

The most important parameter Γ is a measure of the ratio of the computation over the communication times of each component. An application component adapted parallel machine should have a $\Gamma > 1$. Specifically, $\Gamma = 1$ means that communication and computation times are equal.

4 Resulting Grid Middleware Architecture

The overall architecture of the ISS integration into the meta-scheduling environment is depicted in Fig. 2 and the different modules and services are presented in this section. Please note that it is assumed that the executables of the application components already exist before execution.

4.1 Meta-Scheduling Service

The Meta-Scheduling Service (MSS) receives from the Resource Broker (RB) the resource requirements of an application, namely the number of nodes (or a set of numbers of nodes in case of a distributed parallel application) and the planned or estimated execution time. The MSS queries for the availability of known resources. MMS selects a suited machine by optimizing an objective function composed by the Γ model (described above) and the evaluation of costs. The MSS tries to reserve the proposed resource(s) for the job. The result of the reservation is sent back to the RB to check whether the final reservation matches the initial request. In case of a mismatch the reservation process will be re-iterated.

4.2 Resource Broker

The Resource Broker receives requests from the UNICORE Client, collects the necessary information to choose the set of acceptable machines in the prologue phase.

4.3 Data Warehouse

We assume that information about application components exists at the Data Warehouse (DW) module. It is also assumed that at least one executable of all the application components exists.

The DW is the database that keeps all the information related to the application components, to the resources, to the services provided by the Vsites, to monitoring, and to other parameters potentially used to calculate the cost function.

Specifically, the Data Warehouse module contains the following information:

1. **Resources** Application independent hardware quantities.
2. **Services** The services a machines provides (software, libraries installed, etc.).
3. **Monitoring** Application dependent hardware quantities collected after each execution.
4. **Applications** Γ model quantities computed after each execution of an application component.
5. **Other** Other information needed in the cost function such as cost of one hour engineering time.

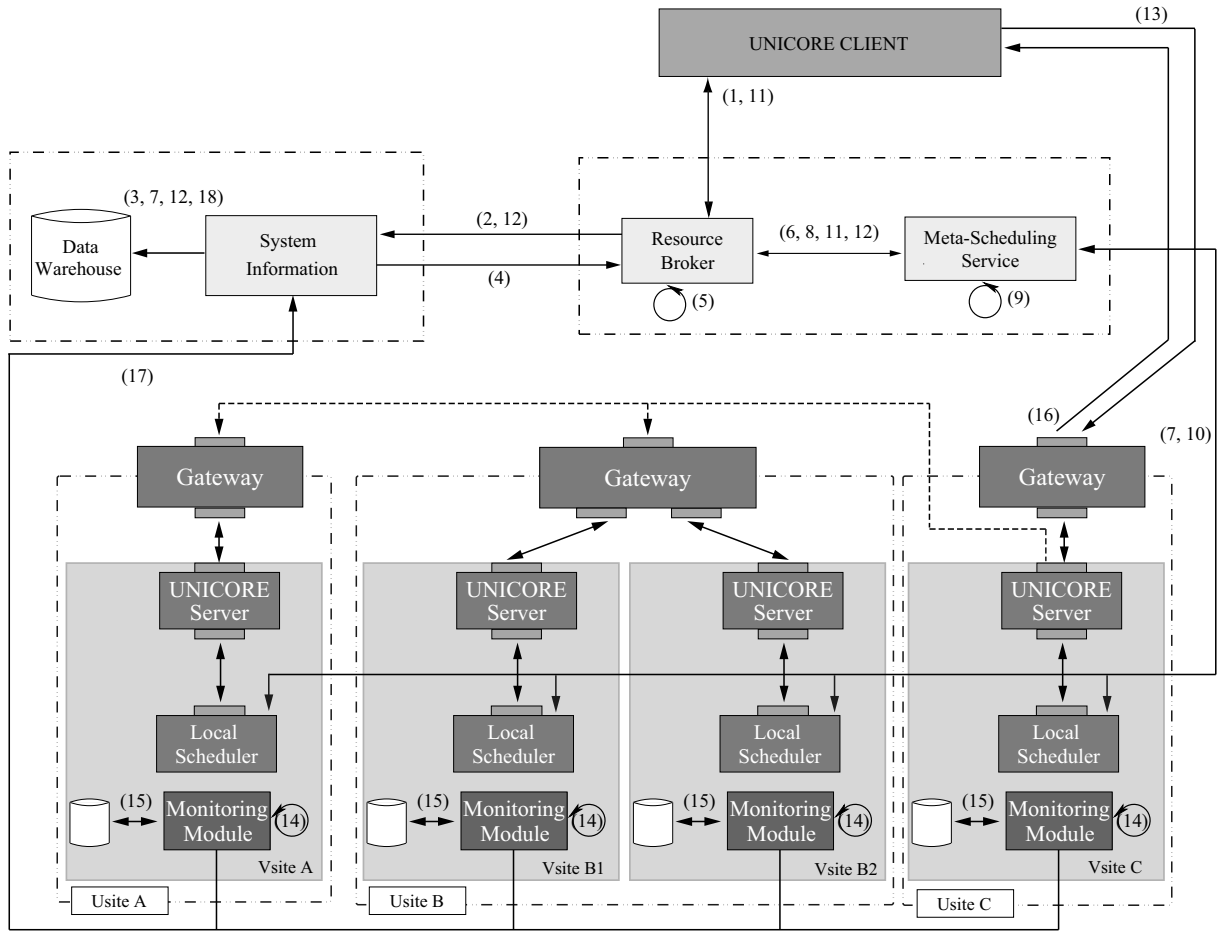

Figure 2: Integration of ISS into the Meta-Scheduling environment.

4.4 System Information

The System Information (SI) module manages the DW, accesses the Vsite-specific UNICORE information service periodically to update the static data in the DW, receives data from the Monitoring Module (MM) and the MSS, and interacts with the RB.

4.5 Monitoring Module

The Monitoring Module collects the application relevant data per Vsite during the runtime of an application. Specifically, it collects dynamic resource information (like CPU usage, network packets number and size, memory usage, etc..), and sends it to the SI (It is an extension of the present TSI).

5 Detailed Scheduling Scenario

Fig. 2 also shows the processes which are executed after a workflow is submitted to the Grid system we have developed. The 18 steps are broken down into three different phases: prologue, scheduling/execution, and epilogue.

First phase: Prologue

(1) The user submits a workflow to the RB through the UNICORE Client.

(2) The RB asks SI for systems able to run each workflow components (in terms of cost, amount of memory, parallel paradigme, etc...)

(3) The SI request the information from the DW

(4) The SI sends the information back to the RB.

(5) According to the information obtained in (3) the RB selects resources that might be used to run the job.

(6) The RB sends the list of resources together with further information (like number of nodes, expected run-time, etc.) and a user certificate to the MSS.

(7) The MSS collects information across all pre-selected resources about availability (e.g. of the compute nodes or of necessary licenses), user-related policies (like access rights), and cost-related parameters.

(8) The MSS notifies the RB about the completion of the prologue phase.

Second phase: Optimal Scheduling and execution

(9) The MSS can now choose among a number of acceptable machines that could execute the workflow. To select a well suited one, it uses consolidated information about each Vsite, e.g. the number of nodes, the memory size per node M_{Vsite}, or the cost for 1 CPU hour per node. The MSS then calculates the cost function to find a well suited resource for the execution of the workflow. Knowing the amount of memory needed by the application, M_a, the MSS can determine the number of nodes P ($P > M_a/M_{Vsite}$) and compute the total time T:
$Total\ time\ T\ =\ Waiting\ Time\ T_w\ +\ Computation\ Time\ T_c$
needed in the cost function. The MSS chooses the machine(s).

(10) The MSS contacts the local scheduling system(s) of the selected resource(s) and tries to obtain a reservation.

(11) If the reservation is confirmed the MSS creates an agreement, sends it to the UNICORE Client via the RB.

(12) The MSS then forwards the decision made in (9) via the RB to the SI which puts the data into the DW.

(13) The UNICORE Client creates the workflow based on the agreement and submits it to the UNICORE Gateway. Subsequent parts of the workflow are handled by the UNICORE Server of the submission Usite.

(14) During the workflow execution, application characteristics, such as CPU usage, network usage, number and size of MPI and NFS messages, and the amount of memory used, are collected by the MM.

(15) The MM stores the information in a local database.

(16) The result of the computation is sent back to the UNICORE Client.

Third phase: Epilogue

(17) Once the workflow execution has finished, the MM sends data stored during the computation to the SI.

(18) The SI computes the Γ model parameters and writes the relevant data into the DW.

The user only has to submit the workflow, the subsequent steps including the selection of well suited resource(s) are transparent to him. Only if an application is executed for the first time, the user has to give some basic information since no application-specific data is present in the DW.

There is a number of uncertainties in the computation of the cost model. The parameters used in the cost function are those that were measured in a previous execution of the same application. However, this previous execution could have used a different input pattern. Additionally, the information queried from the different resources by the MSS is based on data that has been provided by the application (or the user) before the actual execution and may therefore be rather imprecise. In future, by using ISS, such estimations could be improved.

During the epilogue phase data is also collected for statistical purpose. This data can provide information about reasons for a resource's utilisation or a user's satisfaction. If this is bad for a certain HPC resource, for instance because of overfilled waiting queues, other machines of this type should be purchased. If a resource is rarely used it either has a special architecture or the cost charged using it is too high. In the latter case one option would be to adapt the price.

6 Application Example: Submission of ORB5

Let us follow the data flow of the real life plasma physics application ORB5 that runs on parallel machines with over 1000 processors. ORB5 is a particle in cell code. The 3D domain is discretised in $N_1 x N_2 x N_3$ mesh cells in which move p charged particles. These particles deposit their charges in the local cells. Maxwell's equation for the electric field is then solved with the charge density distribution as source term. The electric field accelerates the particles during a short time and the process repeats with the new charge density distribution. As a test case, $N_1 = N_2 = 128$, $N_3 = 64$, $p = 2'000'000$, and the number of time steps is $t = 100$. These values form the ORB5 input file.

Two commodity clusters at EPFL form our test Grid, one having 132 single processor nodes interconnected with a full Fast Ethernet switch (Pleiades), the other has 160 two processor nodes interconnected with a Myrinet network (Mizar).

The different steps in decision to which machine the ORB5 application is submitted are:

(1) The ORB5 execution script and input file are submitted to the RB through a UniCORE client.

(2) The RB requests information on ORB5 from the SI.

(3) The SI selects the information from the DW (memory needed 100 GB, $\Gamma = 1.5$ for Pleiades, $\Gamma = 20$ for Mizar, 1 hour engineering time cost Sfr. 200.-, 8 hours a day).

(4) SI sends back to RB the information.

(5) RB selects Mizar and Pleiades.

(6) RB sends the information on ORB5 to MSS

(7) MSS collects machine information from Pleiades and Mizar:

- **Pleiades:** 132 nodes, 2 GB per node, SFr. 0.50 per node*h, 2400 h*node job limit, availability table (1 day for 64 nodes), user is authorised, executable ORB5 exist.
- **Mizar:** 160 nodes, 4 GB per node, SFr. 2.50 per node*h, 32 nodes job limit, availability table (1 hour for 32 nodes), user is authorised, executable ORB5 exist.

(8) Prologue is finished.

(9) MSS computes the cost function values using the estimated execution time of 1 day:

- **Pleiades:** Total costs = Computing costs (24*64*0.5=SFr. 768.-) + Waiting time ((1+1)*8*200=SFr. 3200.-) = SFR 3968.-
- **Mizar:** Total costs = Computing costs (24*32*2.5=SFr.1920.-) + Waiting time ((1+8)*200=SFr. 1800.-) = SFR 3720.-

MSS decides to submit to Mizar.

(10) MSS requests the reservation of 32 nodes for 24 hours from the local scheduling system of Mizar.

(11) If the reservation is confirmed the MSS creates the agreement, sends it to UC. Otherwise the broker is notified and the selection process will start again.

(12) MSS sends the decision to use Mizar to SI via the RB.

(13) UC submits the ORB5 job to the UNICORE gateway.

(14) Once the job is executed on the 32 nodes the execution data is collected by MM.

(15) MM sends execution data to local database.

(16) Results of job are sent to UC.

(17) MM sends the job execution data stored in the local database to the SI.

(18) SI computes Γ model parameters (e.g. $\Gamma = 18.7$, $M = 87$ GB, Computing time=21h 32') and stores them into DW.

7 Conclusion

The ISS integration into the VIOLA Meta-Scheduling environment is part of the SwissGRID initiative and will be realised in a co-operation between CoreGRID partners. It is planned to install the resulting Grid middleware by the end of 2007 to guide job submission to all HPC machines in Switzerland.

8 Acknowledgements

Some of the work reported in this paper is funded by the German Federal Ministry of Education and Research through the VIOLA project under grant #123456. This paper also includes work carried out jointly within the CoreGRID Network of Excellence funded by the European Commission's IST programme under grant #004265.

References

[1] D. Erwin (ed.), *UNICORE plus final report – uniform interface to computing resource*, Forschungszentrum Jlich, ISBN 3-00-011592-7, 2003.

[2] The EUROGRID project, website, 2005. Online: http://www.eurogrid.org/.

[3] The UniGrids Project, website, 2005. Online: http://www.unigrids.org/.

[4] The National Research Grid Initiative (NaReGI), website, 2005. Online: http://www.naregi.org/index_e.html

[5] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN, website, 2005. Online: http://www.viola-testbed.de/.

[6] R. Gruber, V. Keller, P. Kuonen, M.-Ch. Sawley, B. Schaeli, A. Tolou, M. Torruella, and T.-M. Tran, Intelligent Grid Scheduling System, In *Proc. of Conference on Parallel Processing and Applied Mathematics PPAM 2005*, Poznan, Poland, 2005, to appear.

[7] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder, UNICORE - From Project Results to Production Grids, L. Grandinetti (ed.), *Grid Computing and New Frontiers of High Performance Processing*, Elsevier, 2005, to be published. Pre-print available at: http://arxiv.org/pdf/cs.DC/0502090.

[8] G. Quecke and W. Ziegler, MeSch – An Approach to Resource Management in a Distributed Environment, In *Proc. of 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000)*. Volume 1971 of Lecture Notes in Computer Science, pages 47-54, Springer, 2000.

[9] A. Streit, O. Wldrich, Ph. Wieder, and W. Ziegler, On Scheduling in UNICORE - Extending the Web Services Agreement based Resource Management Framework, In *Proc. of Parallel Computing 2005 (ParCo2005)*, Malaga, Spain, 2005, to appear.

[10] O. Wldrich, Ph. Wieder, and W. Ziegler, A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resource, In *Proc. of Conference on Parallel Processing and Applied Mathematics PPAM 2005*, Poznan, Poland, 2005, to appear.

[11] A. Andrieux et. al., Web Services Agreement Specification, July, 2005. Online: https://forge.gridforum.org/projects/graap-wg/document/WS-AgreementSpecification/en/16.

[12] Ralf Gruber, Pieter Volgers, Alessandro De Vita, Massimiliano Stengel, and Trach-Minh Tran, Parameterisation to tailor commodity clusters to applications, *Future Generation Comp. Syst.*, 19(1), pp.111-120, 2003.

[13] P. Manneback, G. Bergère, N. Emad, R. Gruber, V. Keller, P. Kuonen, S. Noël, and S. Petiton, Towards a scheduling policy for hybrid methods on computational Grids, submitted to CoreGRID Integrated Research in Grid Computing workshop Pisa, 28 - 30 November, 2005.