

IANOS: Intelligent Application Oriented Scheduling for HPC Grids

Hassan Rasheed

Hassan.Rasheed@scai.fraunhofer.de, CoreGRID fellow

ERCIM

2004 Route des Lucioles, BP 93 06902 Sophia Antipolis, France

Ralf Gruber and Vincent Keller

{Ralf.Gruber, Vincent.Keller}@epfl.ch

EPFL

Lausanne, Switzerland

Wolfgang Ziegler and Oliver Wäldrich

{Wolfgang.Ziegler, Oliver.Wäldrich}@scai.fraunhofer.de

Fraunhofer Institute SCAI, Department of Bioinformatics

Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Philipp Wieder

philipp.wieder@udo.edu

Technical University Dortmund

Dortmund, Germany

Pierre Kuonen

Pierre.Kuonen@eif.ch

EIF

Fribourg, Switzerland



**CoreGRID Technical Report
Number TR-0160
July 3, 2008**

Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence

URL: <http://www.coregrid.eu>

IANOS: Intelligent Application Oriented Scheduling for HPC Grids

Hassan Rasheed

Hassan.Rasheed@scai.fraunhofer.de, CoreGRID fellow

ERCIM

2004 Route des Lucioles, BP 93 06902 Sophia Antipolis, France

Ralf Gruber and Vincent Keller

{Ralf.Gruber, Vincent.Keller}@epfl.ch

EPFL

Lausanne, Switzerland

Wolfgang Ziegler and Oliver Wäldrich

{Wolfgang.Ziegler, Oliver.Wäldrich}@scai.fraunhofer.de

Fraunhofer Institute SCAI, Department of Bioinformatics

Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Philipp Wieder

philipp.wieder@udo.edu

Technical University Dortmund

Dortmund, Germany

Pierre Kuonen

Pierre.Kuonen@eif.ch

EIF

Fribourg, Switzerland

CoreGRID TR-0160

July 3, 2008

Abstract

We present the architecture and design of the IANOS scheduling framework. The goal of the new Grid scheduling system is to provide a general job submission framework allowing optimal positioning and scheduling of HPCN applications. The scheduling algorithms used to calculate best-suited resources are based on an objective cost function that exploits information on the parameterization of applications and resources. This standard-based, interoperable scheduling framework comprises four general web services and three modules. The middleware is complemented with one client and one admin console. The implementation is based on proposed Grid and Web services standards (WSRF, WS-Agreement, JSDL, and GLUE). It is agnostic to a specific Grid middleware. The beta version of IANOS has been tested and integrated with UNICORE. The validation of IANOS is in progress by running different types of HPCN applications on a large-scale Grid testbed.

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

1 Introduction

In this paper we describe the integration of the Meta Scheduling Service (MSS) developed within the German VIOLA project [6], with the Swiss Intelligent Grid Scheduling (ISS) project [2], and the new monitoring system VAMOS [4]. The result is the new Grid scheduling framework IANOS (Intelligent ApplicationN-Oriented Scheduling) aims at increasing the throughput of a Grid of High Performance Computing and Networking (HPCN) resources. The first testbed includes parallel machines in Germany and Switzerland.

With respect to existing Grid scheduling systems, the IANOS framework uses information about the behavior of HPCN applications on computational resources and communication networks. The goal is to place an application on a well suited resource in a Grid, reducing costs and turn-around time and improving throughput of the overall computing resources. In future, a Grid should include different types of resources. Some have high processor performances, others a high main memory bandwidth, others a low latency network, others a high inter-node communication bandwidth, or a good access to huge data storage systems. In the set of target applications there are those needing a fast processor, a high main memory bandwidth, a low network latency, a high inter-node communication bandwidth, or need a good data access. It is an art to recognize the right resource for each type of applications. For this purpose, applications and computational resources are characterized by a set of parameters [1]. These parameters can be fix or are determined after each execution by the new VAMOS [4] application-oriented monitoring system. They are then used to make a-priori estimations demanding prediction models for the CPU time, the waiting time, or on the overall execution costs.

The IANOS is a standard based, interoperable scheduling framework. It comprises four general web services and three modules: the *Meta Scheduler* (MSS) performs resource discovery, candidate resource selection and job management; the *Resource Broker* is responsible for the selection of suitable resources based on a Cost Function model; the *System Information* is a frontend to Data Warehouse module, analyzes the stored execution data of a given application to compute certain free parameters to be used by scheduling models; the *Monitoring Service* passes the submission information received from MSS to the Monitoring Module and sends monitored data received from the Monitoring Module to the System Information; the *Monitoring Module* monitors the application during execution and computes execution relevant quantities; the *Data Warehouse* module is part of SI and stores information on applications, Grid resources and execution related data; the *Grid Adapter* module provides generic interfaces and components to interact with different Grid middlewares. The framework is complemented with one *IANOS client* that submits the application to the MSS using WS-Agreement, and the *Web Admin* that provides a web interface to store application and resource relevant parameters and data into the Data Warehouse.

IANOS allocates computing and network resources in a coordinated fashion [3]. The IANOS MSS prepares a list of candidate resources and sends it to the Broker. The Broker collects all the data needed to evaluate the cost function model, prepares a list of potentially optimal schedules that is sent back to the MSS. If still available, the latter module submits the application to the machine with lowest costs. Otherwise, the second most cost efficient resource is chosen and so on. If all the five proposals are not available, the Broker is reactivated to recompute another set of proposals.

The implementation is based on state-of-the-art Grid and Web services technology as well as existing and emerging standards (WSRF, WS-Agreement, JSDL, and GLUE). The IANOS is a general scheduling framework that is agnostic to a Grid middleware, and therefore can easily be adapted to any Grid middleware. The beta version of IANOS has been tested and integrated with UNICORE 5 by implementing a Grid Adapter for it. The present version includes all the models and monitoring capabilities. The free parameters in the cost function model have been validated on GbE clusters with well-known applications. The next step is to build a Grid of different type of resources and a fine tuning of the free parameters using a set of relevant applications coming from the HPCN community.

The IANOS middleware helps not only in optimal scheduling of applications but also the collected data on Grid resources and the monitored data on past application executions can be used to detect overloaded resources and to pin-point inefficient applications that could be further optimized.

The IANOS middleware architecture is detailed in Section 2. The reference scheduling scenario is explained in Section 3. In Section 4, a short description of IANOS scheduling models is presented. The last Section provides summary and information on future work.

2 Architecture & Design

The IANOS architecture is presented in Figure 1. In the following subsections the different modules are presented.

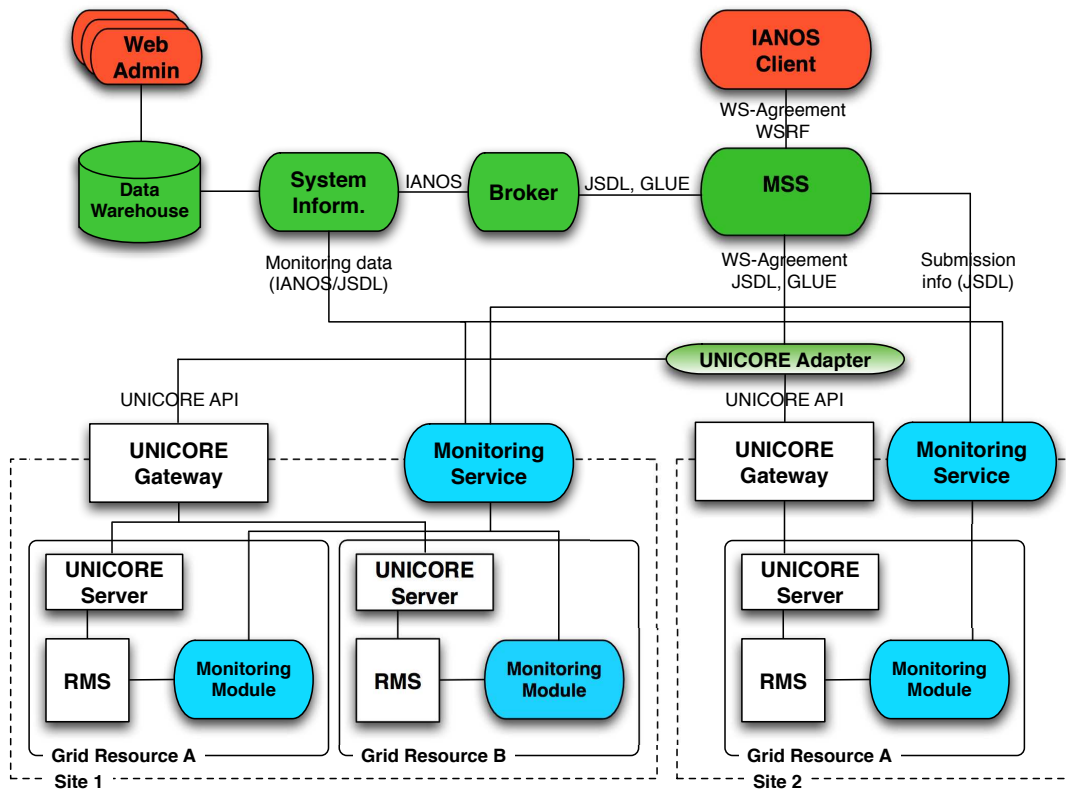


Figure 1: IANOS Middleware Architecture:

2.1 Grid Adapter

The Grid Adapter mediates access to a Grid System through a generic set of modules as shown in Figure 2. It provides information on the Grid resources including CPU time availability and handles the submission of jobs on the selected resources. The SiteManager queries the Grid system for a list of available Grid Sites based on their type, for example, Globus site or UNICORE site. The InfoManager provides static information on the hardware and software configurations, on usage policies, and dynamic information on resource availability, i.e. on the current loads. All this information is modeled as an *extended GLUE schema* [8]. The TemplateManager contacts available Grid Sites for their installed applications. Each application is modeled as a *WSAG-Template*. The DataManager is responsible for the stag in/out of job files. The SubmissionManager is responsible for job submission and management. The ReservationManager handles the reservation of computational and network resources for a job submission. The resources are reserved for certain duration of start and end time. The JobManager and the DataManager receives *JSDL* [7] as input while the ReservationManager receives *WS-Agreement* as an input.

The Grid Adapter is the only IANOS module that is connected to a Grid system. Therefore, necessary interfaces are defined for each Grid adapter module for easy integration with different Grid middlewares. We do not need to implement a new Grid Adapter for each Grid middleware. Instead, a plugin mechanism is used in the design of Grid Adapter modules. At the moment, Grid adapter plugins for the UNICORE Grid middleware have been implemented.

2.2 Meta Scheduling Service (MSS)

The MSS is the only IANOS service that is accessible by the client. It performs client authentication and candidate resources selection. It uses the Grid Adapter to access the Grid system, and the access to the client is provided by an *AgreementFactory* interface. The client and the MSS communicate over the *WS-Agreement* protocol. The Client requests installed applications by issuing an *AgreementTemplate* request. The MSS first validates user requests and then queries the underlying Grid system for the list of installed applications based on user authorization filtering and application availability. The MSS sends these applications to the client in the form of *AgreementTemplates*. The

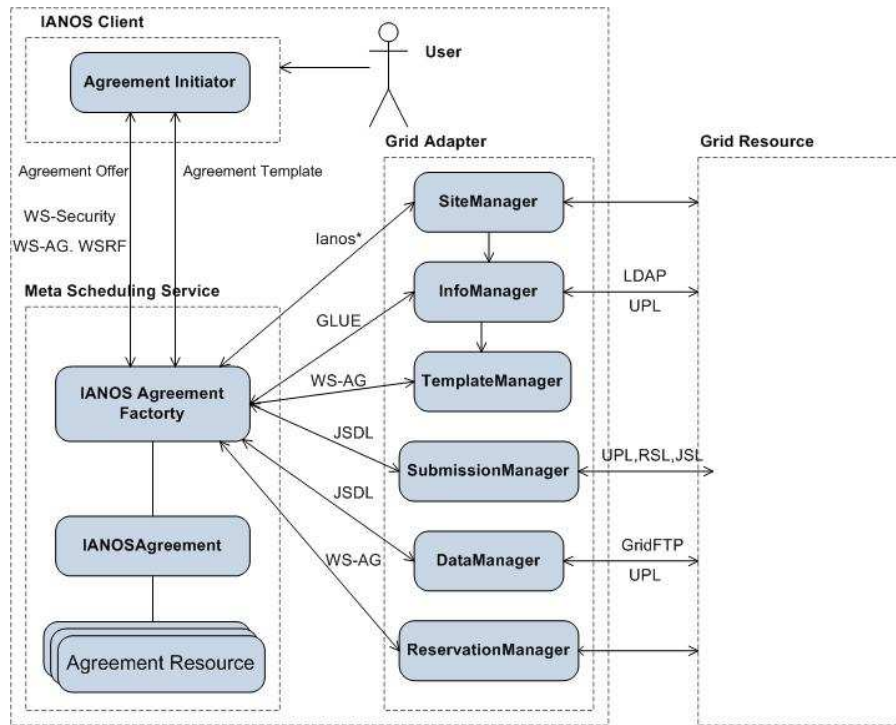


Figure 2: Grid Adapter Modules & their Interaction with MSS and Grid Resource

client selects and submits the application to MSS by sending an *AgreementOffer*. This *AgreementOffer* includes a job description, application parameters, and user QoS preferences (maximum cost, maximum turnaround time).

Upon receiving an *AgreementOffer* from client, it first identifies potential candidate resources and then queries the Grid Adapter for each candidate resource's static and dynamic information, which is received in *GLUE* format. It also retrieves the CPU time availability information (TimeSlots) for each candidate resource by contacting their local RMS. The submitted application is represented by an extended *JSDL* with information on intrinsic application parameters, and user QoS preferences, and *GLUE* contains the information on candidate resources. The MSS sends *JSDL* and *GLUE* documents to the Resource Broker.

The response from the Broker is an ordered list of five execution configurations. Each configuration is represented by *JSDL* and includes start and end time of the execution, required number of nodes, and the cost value for this configuration. The MSS starts negotiation with resources and uses the GA to schedule one of the configurations following the preferences expressed in the ordered list. If it is not possible to schedule any of the five configurations, the Broker is recontacted to compute new configurations based on the new availability information. A *WS-Agreement Resource* is created for each successfully submitted job on a selected resource, with job information stored as resource properties. The MSS supports complete job management, such as job monitoring and control.

2.3 Broker

The Broker service exposes only one operation, *getSystemSelection*. The parameters of this operation are *JSDL* and *GLUE* documents; representing the candidate resources, the application parameters, and the user QoS preference. The Broker uses two modules to decide on the suitable resources for a given application. The *EtemModule* implements the Execution Time Evaluation model, and the *CfmModule* implements the Cost Function model. Applications can use the standard cost function implementation or a separate plugin tailored to a specific application can be implemented. This implementation framework allows separating implementations of different models and to extend or provide new implementations of the models.

Each candidate resource's availability information is in the form of TimeSlots, where every TimeSlot is represented by the start and the end time, and by the free number of nodes during this available time. To compute a cost function value for each TimeSlot of the candidate resources, the RB needs IANOS relevant resource parameters shown

in Table 1, and data on the characteristics and requirements of the submitted application. The RB contacts the System Information for this data, and, based on the application requirements (nodes, memory, libraries etc), filters out unsuitable candidate resources. It then calculates the cost values for all suitable (execution time is less than available time) TimeSlots of the candidate resources, prepares an ordered list of suitable configurations (including start-time and deadlines), and sends it to the MSS. Each configuration along with job requirements is represented by *JSDL*.

2.4 System Information

It exposes three operations: *getAppISSInfo*, *updateAppExecutionInfo* and *updateISSInfo*. The *getAppISSInfo* operation provides data on given resources and applications to the Broker. The *updateAppExecutionInfo* operation receives execution related data from the Monitoring service. The *updateISSInfo* operation receives IANOS relevant static resource information from Monitoring service.

The System Information is a frontend to the Data Warehouse module. An interface is designed to allow integration between System Information and Data Warehouse independent of the specific implementation of the Data Warehouse. The Data Warehouse is contacted by the System Information to query, add or update a stored data. It includes the ETEM Solver module that recomputes the execution time prediction parameters after each job execution. These parameters are then used by the Broker to predict the execution time for the next application submission.

2.5 Data Warehouse

The Data Warehouse is a repository of all information related to the applications, to the resources found, to the previous execution data, and to the monitoring after each execution. Specifically, the Data Warehouse contains the following information:

- Resources: Application independent hardware quantities, IANOS related characteristic parameters
- Applications: Application characteristics and requirements (software, libraries, memory, performance)
- Execution: Execution data for each executed run (Execution dependent hardware quantities, application intrinsic parameters, free parameters)

A Web Admin interface is designed to add or update information about resources and applications into the Data Warehouse.

2.6 Monitoring Service & Module

The Monitoring service receives submission information in *JSDL* from MSS and passes them to the Monitoring Module to monitor the "IANOS" application. Upon receiving the monitored execution data from the Monitoring Module, it sends the same data to the System Information.

The Monitoring Module [4] measures and collects IANOS relevant execution quantities (MFLOPS/s rate, memory needs, cache misses, communication and network relevant information, etc) during application execution. These quantities can be computed through a direct access to hardware counters using PAPI. It performs a mapping between hardware monitored data using the Ganglia service and application relevant data using the RMS (Local Scheduler, Torque/Maui). At the end of the execution, it prepares and sends monitored data to the Monitoring Service as shown on the right side of Table 1.

2.7 Integration with UNICORE Grid System

In a first phase of the IANOS project, we have integrated the IANOS framework with the UNICORE Grid System. The Grid Adapter with its generic functionality and the modules plugins specific for UNICORE has been implemented and tested. On the client side, a UNICORE Client plugin has been developed for this purpose. This client plugin provides a GUI interface to interact with the MSS.

3 Scheduling Scenario

This section describes a reference scenario of job submission using IANOS framework.

- 1 User log in to the client
- 2 User requests applications from MSS by sending WS-Agreement Template request
- 2a MSS validates and authenticates the client request, contact Grid Adapter for all Grid Sites, and then queries these sites for the list of installed applications based on user authorization filtering
- 2b Prepare and return WS-Agreement Templates (representing the applications) to the client
- 3 User selects one application; specifies application intrinsic parameters and QoS preference (Cost, Time, Optimal), and submits the application as an AgreementOffer to MSS
- 3a MSS validates agreement offer, selects candidate resources based on the user access rights and the application availability, queries the Grid Adapter for each candidate resource's static and dynamic information including the resource's local RMS availability
- 4 MSS sends candidate resources and application along with intrinsic parameters and user QoS preferences to Broker
- 5 Broker requests data on candidate resources and given application from System Information
- 7 System Information requests the required information from Data Warehouse through Data Warehouse interface
- 8 Data Warehouse sends collected information to System Information on the candidate resources, on the application, and on the previous execution data of the same application
- 9 System Information sends requested data including the free parameters to Broker
- 10 Broker filters out unsuitable candidate resources based on the application requirements (nodes, memory, libraries etc). It then evaluates the cost function model and prepares a list of cost function values and tolerances for all candidate resources based on user QoS preference
- 11 Broker selects an ordered list of suitable configurations after applying the cost function and sends them to MSS
- 12 MSS uses Grid Adapter to schedule one of the configurations following the preferences expressed in the ordered list, and WS-Agreement Resource is created for each successfully submitted configuration
- 13 MSS sends submission information along with configuration to Monitoring Service, which in turn sends the same submission information to Monitoring Module
- 14 Monitoring Module monitors the execution of application, computes the execution relevant quantities and sends them to Monitoring Service
- 15 Monitoring Service sends the monitored data received from Monitoring Module to System Information
- 16 At the end of execution, results are being sent to the client
- 17 System Information computes ETEM model's free parameters from previous execution data and application intrinsic parameters

4 IANOS Scheduling Models

The Broker of the IANOS middleware uses two models: Cost Function model and the Execution Time Evaluation model. They are based on a parameterization of the applications and the resources [1]. The *Cost Function model* calculates the cost value for each candidate resource. Details can be found in [2]. The *Execution Time Evaluation model* forecasts the execution time of a given application on a given resource that needs to know the CPU node performance of the application.

4.1 Assumptions

- User provides some input parameters such as size of the problem, number of time iterations, etc
- Applications are well balanced in computation, communication, and storage needs
- The Grid resources are homogeneous
- All Grid resources share their IANOS relevant resource and cost parameters as shown in Table 1

4.2 Cost Function Model (CFM)

The job submission process is based on an estimation of the overall cost of the HPCN application. A Cost Function model has been developed that depends on

- CPU Costs K_e
- Licensing Costs K_ℓ
- Waiting Time Costs K_w
- Energy Costs K_{eco}
- Data Transfer Costs K_d

All these quantities depend on the application, on the per hour costs of a resource, on the number of processors used, and on data transfer costs over the networks. We express the money quantity as Electronic Cost Unit ([ECU]). The best schedule is obtained when the cost function [2]

$$z = \alpha(K_e + K_\ell) + \beta K_w + \gamma K_{eco} + \delta K_d \quad (1)$$

is minimized. The four parameters $\alpha, \beta, \gamma,$ and δ can be used to weight user preferences. When a user wants to get the result as soon as possible, regardless of costs, he chooses $\alpha = \gamma = \delta = 0$ and $\beta > 0$. If he demands that execution costs are as small as possible, then $\beta = 0$. The user can prescribe two constraints: Maximum Cost and Maximum Turnaround Time. Then, the minimal cost solution satisfying the constraints is computed by the model.

4.3 Execution Time Evaluation Model (ETEM)

The execution time cost K_e is part of the cost function z . The execution time includes the CPU time and the communication time. Sometimes these timings have to be added, sometimes there is overlap. Their estimations are computed using historical execution data stored in the DataWarehouse. By means of the parameterization of the resources and the application, we estimate the execution time of an application on an unknown Grid resource based on the performance and on the execution time of the same application on a known resource. Then, we are able to predict the costs of this application on all the Grid resources. For this purpose, the application user has to deliver execution-relevant data such as the number of time steps, the matrix size, or other typical data that defines the job to be submitted. This model is quite complex and will be described somewhere else.

5 Conclusion and Future Work

We have presented the design and implementation of IANOS scheduling framework for HPCN applications aiming at optimizing the usage of an HPCN Grid and improve the QoS. It is Grid middleware independent and is based on proposed Grid and Web services standards. The present version of IANOS has been integrated within UNICORE and includes all the functionalities needed to compute the Cost Function model and to submit the application to a well-suited resource. In a first validation step, a small Grid of GbE clusters and simple, well-known applications have been used.

Currently, we are validating IANOS framework on a large scale testbed by running different types of HPCN applications. This testbed includes diverse classes of resources from three institutes: EPFL, Fraunhofer SCAI, and University of Dortmund. This new testbed will enable us to fine-tune the free parameters in the Cost Function model. It is planned to have a well-tested version by the end of the CoreGRID project end of August 2008.

Table 1: IANOS Cost Function relevent Quantities.

Cost Parameters	Resource Parameters	Execution Parameters
Machine Online Time	Number of nodes	Execution Time
Regression Factor	Processors Per Node	Average Efficiency
Factor (in [1/Year])	Cores Per Processor	Average Performance
Insurance Fee	Peak CPU Performance	Packets Sent
Investment Cost	CPU Performance Factor	Packets Received
Interest Cost	Peak Main Memory Bandwidth	Sent Packet Size
Personal Cost	Peak Network Bandwidth	Received Packet Size
The cost for a KWh	Machine Architecture	Memory used
Infrastructure Cos	Operating System	Swap used
Software licence Cost	Network Topology	
Management Overhead	Network NIC's Type	
Host Efficiency (over the year)		
Amortissement Time (in years)		
Hours (non-bissextile year)		
Node's Energy Consumption		

6 Acknowledgments

The ISS project is funded by the Swiss National Supercomputing Centre CSCS. MSS is funded by the German Federal Ministry of Education and Research through the VIOLA project under grant #01AK605F. The IANOS integration work is carried out jointly within the CoreGRID Network of Excellence funded by the European Commission IST programme under grant #004265.

References

- [1] Gruber, R., Volgers, P., De Vita, A., Stengel, M., Tran, T.-M. Parameterisation to tailor commodity clusters to applications, *Future Generation Computer Systems*, **19** (2003) 111-120.
- [2] Gruber, R., Keller, V., Thimard, M., Wäldrich, O., Wieder, P., Ziegler, W., and Manneback, P., Integration of Grid cost model into ISS/VIOLA meta-scheduler environment, *Lecture Notes in Computer Science* **4375** (Springer, 2007) 215-224.
- [3] Wäldrich, O., Wieder, P., and Ziegler, W. A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resources. *Lecture Notes in Computer Science* **3911** (Springer, 2006) 782-791.
- [4] Keller, V. VAMOS web frontend to the Pleiades clusters. In <http://pleiades.epfl.ch/vkeller/VAMOS>.
- [5] UNICORE Open Source Download, <http://www.unicore.eu/download/unicore5/>.
- [6] VIOLA, Vertically Integrated Optical Testbed for Large Application in DFN. <http://www.viola-testbed.de/>.
- [7] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, A. S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) specification, version 1.0. Internet, 2007. <http://forge.ogf.org/sf/projects/jsdl-wg/>.
- [8] S. Andreozzi, S. Burke, L. Field, S. Fisher, B. K'onya, M. Mambelli, J. M. Schopf, M. Viljoen, and A. Wilson. Glue schema specification version 1.2. Internet, 2007. <http://glueschema.forge.cnaf.infn.it/Spec/V12>.