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

Using SLA for resource management and scheduling - a survey

Jan Seidel, Oliver Wäldrich, Wolfgang Ziegler
{Jan.Seidel, Oliver.Waeldrich, Wolfgang.Ziegler}@scai.fhg.de
Fraunhofer Institute SCAI, Department of Bioinformatics
D-53754 Sankt Augustin
Philipp Wieder
ph.wieder@fz-juelich.de
Research Centre Jülich, Central Institute for Applied Mathematics (ZAM)
D-52425 Jülich, Germany

Ramin Yahyapour
ramin.yahyapour@udo.edu
CEI, University Dortmund
D-44221 Dortmund, Germany

CoreGRID Technical Report
Number TR-0096
August 30, 2007

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

Using SLA for resource management and scheduling - a survey

Jan Seidel, Oliver Wäldrich, Wolfgang Ziegler
{Jan.Seidel, Oliver.Waeldrich, Wolfgang.Ziegler}@scai.fhg.de
Fraunhofer Institute SCAI, Department of Bioinformatics
D-53754 Sankt Augustin
Philipp Wieder
ph.wieder@fz-juelich.de
Research Centre Jülich, Central Institute for Applied Mathematics (ZAM)
D-52425 Jülich, Germany

Ramin Yahyapour
ramin.yahyapour@udo.edu
CEI, University Dortmund
D-44221 Dortmund, Germany

CoreGRID TR-0096
August 30, 2007

Abstract

Service Level Agreements may be used to establish agreements on the quality of a service between a service provider and a service consumer. The roles of service provider and service consumer may realised in different shapes ranging from individuals to institutions, software agents or other systems acting on behalf of physical entities or steered by those. This paper gives an overview on the state of the art using Service Level Agreements in the domain of Scheduling and Resource Management. An introduction is given where Service Level Agreements are considered, the technologies used, and what should be accomplished followed by descriptions of the systems that already explicitly implement and use Service Level Agreements.

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

1 Introduction

Service Level Agreements (SLA) are used in different domains and on different levels to establish agreements on the quality of a service (QoS) between a service provider and a service consumer. SLAs can be based on general agreements, e.g. framework agreements, that govern the relationship between parties, may include also legal aspects and may set boundaries for SLAs. However, this technical report addresses the usage of SLAs on a lower, technical level and focuses on agreements dealing with properties of services and resources in the area of Resource Management and Scheduling (RMS).

Inspired by previous developments in the web-service domain research and development in the Grid domain started addressing electronic Service Level Agreements where the negotiation is usually done automatically by appropriate instances of the service provider and service consumer. Today, the focus of these developments in the Grid domain is on Service Level Agreements for resource usage. This survey gives a brief overview on the state of the art using Service Level Agreements in the domain of Scheduling and Resource Management.

Besides the presentation of the individual developments the objective of the survey is to point out how interoperability might be leveraged by following this approach and to clarify to which extent the related research and development is already accomplished within the framework of the CoreGRID Network of Excellence.

In the first part of the report (i.e. in Section 2 and Section 3) we give an overview where Service Level Agreements are considered, the technologies used (namely Web Service Level Agreement (WSLA) and WS-Agreement), and what should be accomplished using Service Level Agreements. While IBM was driving the development of WSLA with a focus on web-services, WS-Agreement was developed in a joint working group of different partners from industry and research in the Open Grid Forum.

The major part of the document is dedicated to a brief presentation of existing or planned solutions for using Service Level Agreements for Grid Resource Management and Scheduling. For each system we provide a brief description and information on the Grid ecosystems the development is suited for, which technology is used for the implementation of the Service Level Agreements and some references. The descriptions are grouped as follows: Systems already using WS-Agreement, followed by those that are planning to use WS-Agreement, followed by those that are about to investigate and evaluate the technology to use for Service Level Agreements.

In detail we present the following systems:

- VIOLA MetaScheduling Service /Fraunhofer SCAI),
- AssessGrid CCS (University Paderborn/Technical University Berlin),
- ASKALON (University of Linz),
- CSF (Platform Computing Inc. and the Jilin University),
- AgentScape (Vrije Universiteit Amsterdam),
- CATNETS (University of Cardiff and partners),
- JSS (Umeå University),
- GRMS (Poznan Supercomputing Center),
- GridWay (University of Madrid),
- eNanos (Barcelona Supercomputing Center), and
- Grid Superscalar (Barcelona Supercomputing Center).

The remainder of the paper is organised as follows. Section 2 presents use-cases. Two existing frameworks for SLAs are introduced in Section 3. RMS systems already using SLA or planning to use are described in Section 4. Section 6 concludes the paper and gives a brief outlook on future work.

2 Use-Cases

In this section we present three classes of use cases where SLAs are either already used in the Grid RMS environment or will be used in near future.

2.1 Agreement on Resource usage

For a presentation with live demonstration of an application the necessary compute resources to run the application have to be available at the time of the presentation. In an normal cluster environment where the nodes of the cluster are used under a space-sharing policy, i.e. giving each user for the time of his job exclusive access to the part of the resources needed for the job-execution, the probability of finding a free slot that matches the requirements of a user immediately is low, thus his job usually will be queued and executed later. Fro a presentation with a fixed schedule this is not acceptable. In order to have the resources required at the time of the presentation the presenter needs to reserve the resources in advance to be sure that they can be used for the demonstration at the time foreseen. This reservation can be expressed as a Quality of Service and an SLA may be issued where the reservation is fixed. E.g. in the VIOLA project [14] this is done by the MetaScheduling Service (MSS), which negotiates the time-slot with the scheduler of the cluster and initiates the reservation of the nodes requested by the user At the time agreed upon the resources will then be available for the user to run his application.

2.2 Agreement on multiple QoS Parameters

In an environment consisting of several clusters operated in different administrative domains SLAs might be used for co-allocation or the resource allocation for workflows. A typical use-case is the co-allocation of multiple compute resources together with the network links between these resources with a dedicated QoS to run a distributed parallel application. The specifies his request and resource orchestrator starts the negotiation with the local scheduling systems of the compute resources and the with the network resource management system (NRMS) in order to find a suitable time-slot, where all required resources are available at the same time. Once a common time-slot is identified the orchestrator requires the reservation of the individual resources. Again, this reservation can be expressed as a Quality of Service and an SLA may be issued where the reservation is fixed. E.g. in the VIOLA project this is done by the MetaScheduling Service (MSS), which negotiates the time-slots with the different schedulers of the clusters and the NRMS and initiates the reservation of the nodes requested by the user At the time agreed upon the resources will then be available for the user to run his distributed application. Another use-case is a workflow spanning across several resources. The only difference to the use-case described before is the kind of temporal dependencies: While for the distributed parallel application the resources must be reserved for the same time, for the workflow use-case the resources are needed in a given sequence. Thus the orchestration service needs to negotiate the reservations such that one workflow component can be executed on the required resource after the preceding component is completed.

2.3 Grid Scheduler interoperation

As there is no single orchestrating service or Grid scheduler in a Grid spanning across countries and administrative domains we have to deal with multiple instances of independent Grid schedulers. Using resources from different domains requires co-ordination across multiple sites. There are two approaches either directly trying to negotiate with respective local scheduling systems or negotiation with the respective local orchestrator. The former solution requires local policies allowing a remote orchestrator to negotiate with local schedulers, which is in general not the case. In the second case there is one access point to the local resources, which then negotiates on behalf of the initiation orchestrator. As the second approach also has a better scalability than the first one the OGF Grid Scheduling Architecture Research Group (GSA-RG) decided to consider this approach for the definition of a Grid Scheduling Architecture (see Figure 1). For the communication between the different orchestration services or Grid schedulers a language and a protocol to create SLAs was selected to achieve the necessary interoperability while at the same time resulting in SLAs at the end of the negotiation process, that can be combined by the initiating orchestrator into one single agreement vis--vis his client.

3 Service Level Agreement Frameworks

This section introduces two existing frameworks for SLA specification and monitoring: Web Service Level Agreement (WSLA) developed by IBM (completed in March 2003) and Web Service Agreement (WS-Agreement) developed in a working group of the Open Grid Forum (OGF). While the WSLA specification [17] can still be downloaded at the WSLA web-page it was not widely adopted and seems to be superseded by WS-Agreement to some extent now.

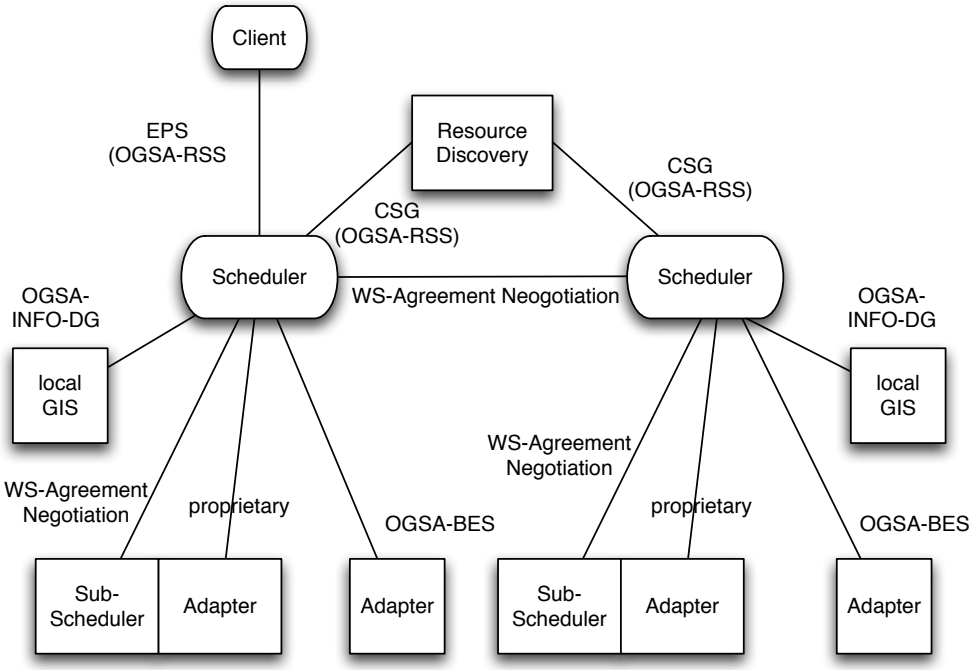

Figure 1: Grid Scheduler interoperation including OGSA components

3.1 WS-Agreement

The Web Services Agreement Specification [15] from the Open Grid Forum (OGF) describes a protocol for establishing an agreement on the usage of Services between a service provider and a consumer. It defines a language and a protocol to represent the services of providers, create agreements based on offers and monitor agreement compliance at runtime. An agreement defines a relationship between two parties that is dynamically established and dynamically managed. The objective of this relationship is to deliver a service by one of the parties. In the agreement each party agrees on the respective roles, rights and obligations. A provider in an agreement offers a service according to conditions described in the agreement. A consumer enters into an agreement with the intent of obtaining guarantees on the availability of one or more services from the provider. Agreements can also be negotiated by entities acting on behalf the provider and / or the consumer. An agreement creation process usually consists of three steps: The initiator retrieves a template from the responder, which advertises the types of offers the responder is willing to accept. The initiator then makes an offer, which is either accepted or rejected by the responder. WS-AgreementNegotiation which sits on top of WS-Agreement furthermore describes the re/negotiation of agreements. An agreement consists of the agreement name, its context and the agreement terms. The context contains information about the involved parties and metadata such as the duration of the agreement. Agreement terms define the content of an agreement: Service Description Terms (SDTs) define the functionality that is delivered under an agreement. A SDT includes a domain-specific description of the offered or required functionality (the service itself). Guarantee Terms define assurance on service quality of the service described by the SDTs. They define Service Level Objectives (SLOs), which describe the quality of service aspects of the service that have to be fulfilled by the provider. The structure of an agreement is depicted in Figure 2. The Web Services Agreement Specification allows the usage of any domain specific or standard condition expression language to define SLOs. The specification of domain-specific term languages is explicitly left open.

3.2 Web Service Level Agreement (WSLA)

WSLA [16] is a framework developed by IBM for specifying and monitoring Service Level Agreements (SLA) for Web Services. The framework is able to measure and monitor the QoS parameters of a Web Service and reports violations to the parties specified in the SLA. In a Web Service environment, services are usually subscribed dynamically and

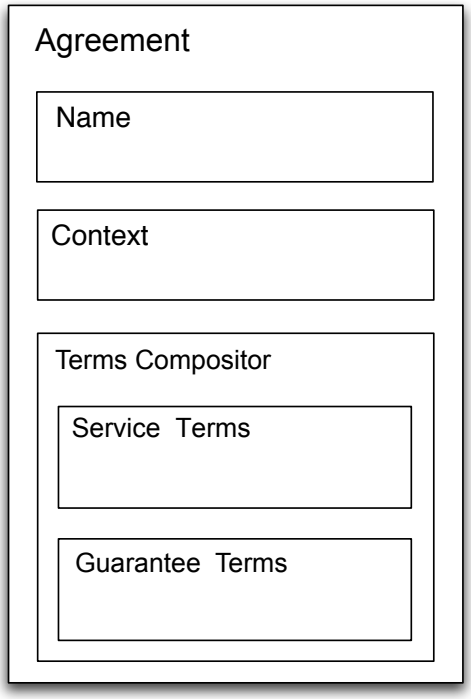

Figure 2: Structure of an Agreement

on demand. In this environment, automatic SLA monitoring and enforcement helps to fulfil the requirements of both service providers and consumers. WSLA provides a formal language based on XML Schema to express SLAs and a runtime architecture which is able to interpret this language. The runtime architecture comprises several SLA monitoring services, which may be outsourced to third parties (supporting parties) to ensure a maximum of objectivity. The WSLA language allows service customers and providers to define SLAs and their parameters and specify how they are measured. The WSLA monitoring services are automatically configured to enforce an SLA upon receipt.

The SLA management life-cycle of WSLA consists of five distinct stages:

1. Negotiation / Establishment: In this stage an agreement between the provider and the consumer of a service is arranged and signed. An SLA document is generated.
2. SLA Deployment: The SLA document of the previous stage is validated and distributed to the involved components and parties.
3. Measurement and Reporting: In this stage the SLA parameters are computed by retrieving resource metrics from the managed resources and the measured SLA parameters are compared against the guarantees defined in the SLA.
4. Corrective Management Actions: If an SLO has been violated, corrective actions are carried out. These actions can be to open a trouble ticket or automatically communicate with the management system to solve potential performance problems. Before all actions regarding the managed system the Business Entity of the service provider is consulted to verify if the proposed actions are allowable.
5. SLA Termination: The parties of an SLA can negotiate the termination the same way the establishment is done. Alternatively, an expiration date can be specified in the SLA.

4 Usage of SLAs for Resource Management and Scheduling

4.1 Schedulers with WS-Agreement implementations

A couple of projects already started implementing and using WS-Agreement for the definition of SLA. The projects are in different stages and not all build their implementation on the version 1.0 of WS-Agreement that has become a proposed recommendation in May 2007. However, it may be expected that all of the implementations move to this version within the next months.

4.1.1 VIOLA MetaScheduling Service (MSS)

Description: In the VIOLA project an optical testbed was implemented between multiple partners in Germany. The main goals were the test of advanced network architectures, development of software for user-driven dynamical provision of bandwidth and test of advanced applications e.g. in the Grid area. Applications were integrated into VIOLA to provide a realistic environment for testing new network architectures and equipment. The project ended April 2007 but the MSS will be further developed in a number of other projects.

Grid ecosystems: Grid applications in VIOLA were run on three Linux-based PC-Clusters, a SUN-Cluster and a Cray X-D1 with a total peak performance of 900 GFLOPS. Figure 3 depicts the current architecture which was developed from the VIOLA one. Middleware to support Grid usage by applications was developed in a sub-project of VIOLA. Grid resources are accessed by UNICORE, the Uniform Interface to Computing Resources, which provides a seamless and secure access to multiple compute sites. A single instance of a MetaScheduling Service integrated into the UNICORE middleware is able to perform co-ordinated CPU and network bandwidth reservation between the clusters in the Grid, enabling distributed applications on these systems [11].

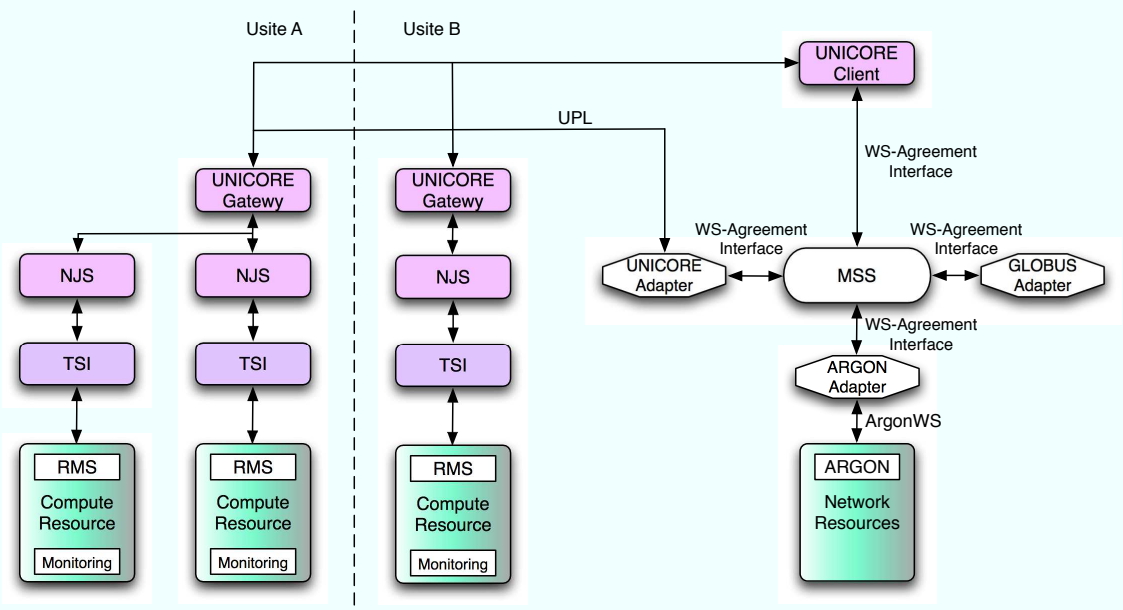

Figure 3: Architecture of the MetaScheduling Environment

WS-Agreement implementation: The VIOLA MetaScheduling Service MSS is responsible for negotiation of resource allocation with the local scheduling systems. It is implemented as a Web Service receiving a list of resources preselected by a resource selection service. The resource reservation is based on WS-Agreement. In the VIOLA project the MSS is utilised to co-allocate computational and network resources in a UNICORE-based Grid. Network resources are reserved through a WS-Agreement Interface with the Adapter of the NRMS ARGON [1]. Resource reservations are negotiated through adapters with local scheduling systems also using WS-Agreement. Furthermore,

the negotiation between the MSS and the UNICORE Client is based on WS-Agreement. When a UNICORE Client wants to make a reservation, it sends the resource request to the MSS as a WS-Agreement template. The MetaScheduling Service then negotiates a potential start time for the Job and requests reservation of the network and computational resources. After successful completion of this reservation the MSS sends an End Point Reference (EPR) of the created WS-Agreement back to the UNICORE Client.

4.1.2 AssessGrid openCCS

Description: AssessGrid is a European project, which started in April 2006. AssessGrid introduces risk management and assessment to Grid computing to facilitate a wider adoption of Grid technologies in business and society. Risk assessment helps providers to make decisions on suitable SLA offers by relating the risk of failure to penalty fees. Similarly, end-users get knowledge about the risk of an SLA violation by a resource provider that helps to make appropriate decisions regarding acceptable costs and penalty fees. A broker is the matchmaker between end-users and providers. The broker provides a time / cost / risk optimised assignment of SLA requests to SLA offers. AssessGrid introduces a framework supporting risk assessment and management for all three Grid actors (end-user, broker and resource provider).

Grid ecosystems: AssessGrid is designed to supply Next Generation Grids with additional components to help SLA concepts become a widely accepted tool for commercial Grid usage. All results of the AssessGrid project shall be demonstrated in real environments and in close interaction with customers. AssessGrid uses a distributed Resource Management System (RMS) called Computing Center Software (CCS or openCCS). CCS provides interfaces to UNICORE and Globus Toolkit 4 [9].

WS-Agreement implementation: In CCS a Grid resource broker provides transparent access to Grid resources by querying resource providers on behalf of end-users. For the end-user the broker acts as a pre-selector of Grid resource providers. The WS-Agreement protocol is currently being implemented for openCCS. The major negotiable SLA parameters in openCCS are: General parameters like number of nodes, amount of memory, job runtime, Deadline for job completion, Policies on security or migration, and Fault tolerance requirements. In the AssessGrid project the RMS is enhanced by risk assessment and risk management functionalities. A negotiator module is responsible for negotiating SLAs with external contractors using the WS-Agreement-Negotiation protocol. The negotiator defines the price, penalty and risk in the SLA according to policies. The risk is included in the SLA as an additional attribute. The Risk Assessor evaluates current monitoring information as well as aggregated statistical information to assess the risk for an SLA violation. WS-Agreement is used in the communication of the user with the broker, the user with the provider and the broker with the provider. The user interface modifies the SLA template of the broker or provider based on the user prerequisites such as hardware architecture and available libraries and sends it back to the broker or provider in order to receive SLA offers. Offers are presented to the end-user by the broker with price, risk of failure and penalty attributes. The end-user then either agrees or rejects such an offer.

4.1.3 ASKALON

Description: ASKALON [2] is a Grid project of the Distributed and Parallel Systems Group at the University of Innsbruck. The main goal is to simplify the development and optimisation of applications that can utilise a Grid for computation. It is developed as a Service Oriented Architecture (SOA).

Grid ecosystems: ASKALON is used to develop and port scientific applications as workflows in the Austrian Grid project. The developers designed an XML-based Abstract Grid Workflow Language (AGWL) to compose job workflows. A resource manager remotely deploys software e.g. by using the Globus Toolkit middleware with the GridFTP protocol and the Globus Resource Allocation Manager (GRAM).

WS-Agreement implementation: Service-level agreements can be made with the Grid resource for a specified timeframe by using the GridARM Agreement package. GridARM ensures that a defined capacity and capability is available in the agreed timeframe including parameters like number of CPUs, etc. It maximises the resource and client utility.

The agreement management consists of two parts: The AgreementNegotiator and the AgreementService. The AgreementNegotiator works as an agreement factory service. During the agreement negotiation process with the client, multiple agreement offers are created based on the information provided by the client as an AgreementTemplate. The client can accept one or more of the offers or reject all of them. The AgreementService manages particular agreements. After the negotiation process is finished, all interaction addressing e.g. agreement access and updates is done by interacting with the AgreementService using an EPR. In ASKALON, the client (consumer) always creates agreement templates and is therefore always the agreement initiator. The provider creates one or more offers which are accepted or rejected by the client.

4.1.4 Community Scheduler Framework

Description: The Community Scheduler Framework (CSF) [4] is an open source implementation of an OGSA-based metascheduler. It is developed by Platform Computing Inc. and the Jilin University (China). The Community Scheduler Framework (CSF) is a set of modules that can be assembled to create a metascheduling system that accepts job requests and executes them using available Grid compute services. The CSF classes can be extended to provide domain specific schedulers and support different kinds of Grid deployment models. Grid level scheduling algorithms can for example include scheduling across multiple clusters within a VO, co-scheduling across multiple resource managers, scheduling based on SLAs and economic scheduling models.

Grid ecosystems: CSF supports the Globus Toolkits Grid Resource Allocation and Management (GRAM) service. It is included in Globus Toolkit 4.0.

WS-Agreement implementation: CSF claims to support the WS-Agreement specification, however we are not aware of an implementation using WS-Agreement so far. The reservation service implements the AgreementFactoryType as defined in the OGSI-Agreement specification to support reservations based on agreement terms. A reservation can be created by agreement by using a template agreement file in the Globus installation directory. The values of this agreement XML file must be customised.

4.1.5 AgentScape

Description: The Intelligent Interactive Distributed Systems (IIDS) group of the Vrije Universiteit Amsterdam develops the AgentScape framework that provides mobile agents access to computing resources on heterogeneous systems across the Internet [10].

Grid ecosystems: AgentScape provides mobile agents co-ordinated resource usage across multiple domains. The negotiation of resource access for applications is based on WS-Agreements. An application can acquire time-limited contracts for resources. A mediator called domain co-ordinator (DC) in AgentScape represents multiple autonomous hosts and communicates with the mobile agent on behalf of these nodes. Agents can negotiate their options with DCs of multiple domains, being able to select the DC that provides the best offer.

WS-Agreement implementation: The WS-Agreement based negotiation infrastructure of AgentScape allows agents to negotiate terms of conditions and quality of service of resource access with domain co-ordinators. Hosts providing resources are aggregated into virtual domains. The DC represents the hosts within a virtual domain in the negotiation process. The WS-Agreement interaction model is extended to allow a more sophisticated negotiation. In this extended negotiation model, hosts provide an agreement interface to their DC. The DC aggregates templates offered by hosts into composed templates and makes these available to agents. Agreement requests mad by agents based on composed templates are received by the DC. The DC then negotiates an agreement with the hosts with the requested resources. The additional accept/reject interaction sequence allows agents to enter into negotiations with multiple providers and compare received offers. The AgentScape negotiation architecture defines a set of resources that can be requested and used by agents including CPU time, communication bandwidth, amount of memory, disk space, web services that the agent is allowed to access and the number of calls of a web service that the agent is allowed to do. After the negotiation phase, a host manager monitors and controls the resource usage to ensure that agreements are met.

4.1.6 CATNETS

Description: CATNETS is a project of several universities and research centers across Europe with the objective to determine the applicability of a decentralized economic self-organization mechanism for resource allocation in application layer networks (ALN), which include Grid systems. The name CATNETS is based on an economic self-organization approach of a free market , the Catallaxy . CATNETS simulates the ALN environment by an economy, where the resources are for example processor time or storage space, while the economic actors are computers or web services. The application service and compute resource allocation of Application Layer Networks is broken down into two types of interrelated markets: A Grid resource market, where computational and data resources are traded and a service market where application services are traded. These services provide particular application functionality, e.g. query execution or molecule docking. In these separate markets complex services buy basic services, which buy raw resources. In this Catallaxy approach, the market is self-organizing which means that no centralized broker is required.

Grid ecosystems: In the prototype implementation the middleware is implemented as a set of simple, specialised agents using the light-weighted agents platform of the Decentralised Information Ecosystem Technologies (DIET) project. The agents provide for example access to markets, negotiations, object discovery and communication. The management of local resources is based on the WS-Resource Framework offered by Globus Toolkit 4. Middleware is further implemented using JXTA technology.

WS-Agreement implementation: WS-Agreement is used in the implementation of both the service market and the resource market. CATNETS defines separate bidding language for the service and the resource market, which are used by agents to submit bids for services or resources. These languages are mapped onto WS-Agreement via domain-specific schemes. The offers are encoded in XML using WS-Agreement and JSDL. In the resource market basic services can submit sell orders to the order books with WS-Agreement and the resource services can submit buy orders to the order books. After submission of all bids to the auctioneer, the allocation and the corresponding prices are determined, which results in an agreement. The activity on the service market is quite similar. The WS-Agreement implementation of CATNETS is technically integrated into the Triana workflow engine which allows visualisation of Agreement Templates and Offers. It also enables the workflow to be paused until an Agreement Offer has been confirmed.

4.1.7 Job Submission Service (JSS)

Description: JSS [5] developed at the Umeå University is a broker aiming at identifying the set of resources that minimizes the Total Time to Delivery (TTD), or part thereof, for each individual job submission. In order to do this, the broker makes an a priori estimation of the whole or parts of the TTD for all resources of interest before making the selection. The TTD estimation includes performing a benchmark-based execution time estimation, estimating file transfer times, and performing advance reservations of resources in order to obtain a guaranteed batch-queue waiting time. For resources not providing all information required or a reservation capability, less accurate estimations are performed. On the Grid resource, an authorization callout mechanism is used to validate that i) the requested reservation identifier exists and ii) the reservation actually was created by the same user that submits the job. JSS is currently used e.g. in NorduGrid and Swegrid.

Grid ecosystems: JSS with integrated WS-Agreement is currently available for two Grid middleware environments: JSS provides support for Globus Toolkit 4 and NorduGrid/ARC. Integration with another Grid middleware only consists of writing the above described authorization plug-in.

WS-Agreement implementation: JSS is using a WS-Agreement implementation that was done at the Ume University. The implementation originally was planned to be based on Cremona (an early WS-Agreement implementation by IBM), which turned out not to be feasible as IBM didnt give access to the source code. The GT4/WSRF-based implementation is rather straightforward, with createAgreement requests forwarded by the AgreementFactory to a decisionMaker plugin, which interacts with the local resource management system. One such plugin is used to create advance reservations, and hence interacts with the batch system scheduler. JSS currently support the Maui scheduler, although others (supporting advance reservation) easily can be added. JSS also implement s a two-phase mechanism,

where reservations will be released shortly after their creation, unless they are confirmed. This is implemented using WS-ResourceLifetime, as each Agreement is modelled as a WS-Resource. The terms used to negotiate advance reservations are

- number of CPUs,
- duration,
- earliest allowed start time,
- latest allowed start time,
- malleability flag.

The semantics of a malleable reservation is that the local scheduler may modify the reservation start time freely, as long as the starttime stays in the [earliest, latest] allowed start time window. Previous research demonstrates that this behaviour reduces the utilization drop induced by using advance reservations. JSS has however, due to lack of support in the local scheduler, not been able to implement malleable reservations, although the system is prepared to support it. Reservations are created by the job submission service during resource brokering. When the job is submitted to the selected resource, an identifier for the created reservation is included in the job description.

4.2 Scheduler planning WS-Agreement implementations

The Grid Scheduling Architecture research group (GSA-RG) of the Open Grid Forum and the task 1 (Definition of a Grid scheduling architecture) defined in the CoreGRID Institute on Resource Management and Scheduling address the overall architecture of scheduling in the Grid, the components and their interaction. In the GSA-RG a number of projects and groups are represented that have developed and maintain most of the systems described before. One of the outcomes of the work in the GSA-RG is considering WS-Agreement for the communication between Grid schedulers. The developers of two other systems also participating in the GSA-RG decided to join the effort of implementing WS-Agreement to provide interoperability with other Grid schedulers and to create a testbed to perform a number of demos and experiments.

4.2.1 Grid Resource Management System

Description: The Grid Resource Management System (GRMS) [8] is an open source metascheduling system, which is developed at the Poznan Supercomputing and Networking Center, Poland. It is a part of the Gridge Toolkit. GRMS is a metascheduling system, which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. The main goal of GRMS is to manage the process of remote job submission to various batch queuing systems, clusters or resources. GRMS supports dynamic resource selection, mapping and scheduling.

Grid ecosystems: The Gridge Toolkit components were tested with different version of the Globus Toolkit and other Grid middleware solutions. GRMS Service is a web service implemented in Java running on an Apache Tomcat and Axis server as the SOAP engine. The GRMS interfaces are described by the Web Services Description Language (WSDL). GRMS can be used in conjunction with various queuing systems, such as Condor, PBS and LSF.

WS-Agreement implementation: GRMS developers are active in the OGF GSA-RG and plan to extend GRMS to support the evolving proposal for a Grid Scheduling Architecture.

4.2.2 GridWay

Description: GridWay [7] is a metascheduler developed by the Distributed Systems Architecture Group at the University of Madrid, Spain. GridWay performs job execution management and resource brokering transparently to the end user. It furthermore adapts job execution to changing Grid conditions by providing e.g. fault recovery mechanisms, dynamic scheduling, on-request migration and opportunistic migration.

Grid ecosystems: GridWay provides interfaces to remote resources through Globus GRAM and therefore supports all remote platforms and resource managers (for example fork, PBS and LSF) compatible with Globus. GridWay supports the usage of the Job Submission Description Language (JDSL) as well as GridWay Job Template files.

WS-Agreement implementation: The GridWay metascheduler could be extended or used as a building block for more complex architectures that implement SLAs or advanced reservation. Gridway developers are active in the OGF GSA-RG and plan to support the evolving proposal for a Grid Scheduling Architecture.

4.3 Scheduler not yet using but considering SLA technology

4.3.1 eNanos Resource Broker

Description: In the eNanos project [6] of the Barcelona Supercomputing Center a general purpose OGSI-compliant Grid resource broker [12] is developed and maintained.

Grid ecosystems: The eNanos Grid resource broker is implemented on top of Globus Toolkit (GT) and supports both GT2 and GT3. The broker focuses on resource discovery and management as well as dynamic policies management for job scheduling and resource selection. It utilises some of the Globus Toolkit services such as the Grid Security Infrastructure (GSI), the Grid Resource Allocation and Management (GRAM), Data Management Services (e.g. gridFTP), and the Information Services, Monitoring and Discovery System (MDS).

Service Level Agreement implementation; The eNanos Grid resource broker provides dynamic policy management and multi-criteria user requirements. The user multi-criteria file is an XML document is composed of requirements and recommendations and can be used in policy evaluation. A requirement (hard attribute) is a restriction for the resource filtering and a recommendation (soft attribute) can be used to create a resource ranking for policy evaluation. Extending the Grid resource broker to include an SLA component is currently under discussion.

4.3.2 Grid superscalar

Description: GRID superscalar [3] is a Grid programming environment developed and maintained at the Barcelona Supercomputing Center [13]. With GRID superscalar a sequential application composed of tasks of a certain granularity is automatically converted into a parallel application where the tasks are executed in different servers of a computational GRID. The aim of GRID superscalar is to reduce the development complexity of Grid applications to the minimum. GRID superscalar provides automatic deployment of tasks. It sends and compiles code in the remote workers and the master. It also provides automatic parallelisation between operations at runtime by utilising concepts such as dependence analysis, instruction windows, locality and prediction.

Grid ecosystems: The current version is built on top of Globus 2.4, Globus 4.0 ssh/scp. For file transfer, security, etc. the Globus functionality is used.

Service Level Agreement implementation; When a task is ready for execution the scheduler tries to allocate a resource. In this case the broker receives a request and checks if a resource fulfils the constraints of this task. If more than one resource fulfils the constraints, the resource with minimum file transfer and execution time is selected. Extending Grid superscalar to include an SLA component is currently under discussion.

5 Overview on the capabilities of the presented systems

Table 1 gives an overview on the capabilities of the different resource management and scheduling systems described in Section 4. The table lists the scope of the systems (local scheduling, MetaScheduling, Broker), the capability to orchestrate (co-allocate) resources, the support for advance reservation, the state of WS-Agreement implementation and the grid ecosystems supported by the systems.

RMS System	Local Scheduling	MetaScheduling	Broker	Orchestration	Advance Reservation	WS-Agreement	Middleware
VIOLA MSS		X		X	X	X	UNICORE, GT4 (plan)
CCS AssessGrid	(X)		X		X	X	GT4
ASKALON		X		X	X	X	GT4
CSF		X				X	GT4
AgentScape			X			X	n/a
CATNETS			X			X	GT4
JSS			X		X	X	GT4
GRMS		X			X	planned	GT2/3, GT4
GridWay		X		X	X	planned	GT4
eNanos			X			tbd	GT2, GT3
Grid Superscalar	X					tbd	GT2.4, GT4

Table 1: Overview of the capabilities of the presented systems

6 Outlook

A large number of the researchers contributing to the developments described in this survey are affiliated with institutions that are members of CoreGRID; many of these researchers are members of research groups of the CoreGRID Institute on Resource Management and Scheduling (IRMS). The development of several of the systems described in this survey is influenced by the research of the IRMS and contributing to research objectives of the IRMS, e.g. the VIOLA MetaScheduling Service is extended within a CoreGRID fellowship to interoperate with the Intelligent Scheduling System (ISS) developed by partners of the Swiss Grid consortium. In accordance with one of the overall objectives of CoreGRID, the integration of European Grid research, the focus of the work in the IRMS is on integration and interoperability of the developments of the partners approaches and developments.

There are already a number of Schedulers either using an SLA implementation for providing dedicated QoS to the end-users applications. Others plan to provide an implementation of it in the near future, either for interoperability reasons and/or to provide a guaranteed level of service. Recent discussions in the OGF GRAAP working group and at the 5th Meeting of the CoreGRID Institute on Resource Management and Scheduling in Manchester point out that Service Level Agreements will play a major role in the domain of Grid Resource Management and Scheduling. WS-Agreement is now a proposed OGF recommendation for the expression and creation of SLAs. It might be expected, that more Grid level Schedulers or Brokers will adopt to using SLAs for two reasons: (i) interoperability with other Grid level Schedulers and Brokers and (ii) using a standardised interface for negotiating QoS between users and service providers.

7 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 #01AK605L. 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] ARGON - Allocation and Reservation in Grid-enabled Optic Networks. VIOLA Project Report, March 2006 <http://www.viola-testbed.de/>.

[2] ASKALON, 2007. 15 June 2007 <http://www.dps.uibk.ac.at/projects/askalon/>.

[3] Rosa M. Badia, Raul Sirvent, Jesus Labarta, and Josep M. Perez. Programming the grid: An imperative language based approach. In *Engineering the Grid*, January 2006.

[4] CSF - Community Scheduler Framework, 2007. 15 June 2007 <http://www.globus.org/grid_software/computation/csf.php>.

[5] Erik Elmroth and Johan Tordsson. An interoperable standards-based grid resource broker and job submission service,. In *Proceedings of the first IEEE Conference on e-Science and Grid Computing*, pages 212–220, USA, 2005. IEEE Computer Society Press.

[6] eNANOS Scheduler, 2007. 15 June 2007 <http://www.bsc.es/grid/enanos>.

[7] Gridway home page, 2007. 15 June 2007 <http://www.gridway.org/>.

[8] GRMS - Grid Resource Management System, 2007. available at the Gridge webpages: <http://www.gridge.org>, last visited: June 15, 2007.

[9] A. Keller. openCCS: Computing Center Software. Technical report, Paderborn Center for Parallel Computing, 2007.

[10] D.G.A. Mobach, B.J. Overeinder, and F.M.T Brazier. A ws-agreement based resource negotiation framework for mobile agents. *Scalable Computing: Practice and Experience*, 7 (1):23 – 36, 2006.

[11] Wieder Philipp, Oliver Wäldrich, and Wolfgang Ziegler. A meta-scheduling service for co-allocating arbitrary types of resources. In *Proceedings of the 6th International Conference, Parallel Processing and Applied Mathematics, PPAM 2005*,, volume 3911 of *LNCS*, pages 782 – 791, Poznan, Poland, September 2005. Springer.

[12] I. Rodero, J. Corbaln, R.M. R.M. Badia, and J. Labarta. enanos grid resource broker. In *Proceedings of the European Grid Conference 2005*, volume 3470 of *LNCS*, pages 111 – 121, Amsterdam, Netherlands, February 2005. Springer.

[13] GRID superscalar, 2007. 15 June 2007 <http://www.bsc.es/grid/gridsuperscalar >.

[14] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN, 2007. 15 June 2007 <http://www.viola-testbed.de/>.

[15] Web Service Agreement (WS-Agreement). GFD.107 proposed recommendation, available at <http://www.ogf.org/documents/GFD.107.pdf>.

[16] Web Service Level Agreements (WSLA) Project. Website, <http://researchweb.watson.ibm.com/wsla/documents.html/>, last visited: June 15, 2007.

[17] WSLA Language Specification, Version 1.0, January 2003, IBM Corporation, 2003. 15 June 2007 <http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf>.