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

Improving Workflow Execution through SLA-based Advance Reservation

Philipp Wieder
ph.wieder@fz-juelich.de
Central Institute for Applied Mathematics, Research Centre Jülich
D-52425 Jülich, Germany

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

Oliver Wäldrich, Wolfgang Ziegler
{Oliver.Waeldrich, Wolfgang.Ziegler}@scai.fhg.de
Fraunhofer Institute SCAI
D-53754 Sankt Augustin

CoreGRID Technical Report
Number TR-0053
December 29, 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

Improving Workflow Execution through SLA-based Advance Reservation

Philipp Wieder
ph.wieder@fz-juelich.de
Central Institute for Applied Mathematics, Research Centre Jülich
D-52425 Jülich, Germany

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

Oliver Wäldrich, Wolfgang Ziegler
{Oliver.Waeldrich, Wolfgang.Ziegler}@scai.fhg.de
Fraunhofer Institute SCAI
D-53754 Sankt Augustin

CoreGRID TR-0053

December 29, 2006

Abstract

In SOA-based Grid environments service provider and service consumer usually do not know each other. In order to establish a business relation they must inter alia (i) create a trust relationship, and (ii) set up mechanisms to create reliable, verifiable, and, at least in a commercial environment, also audible agreements with respect to the services requested, delivered and consumed. In this paper we will only briefly address (i) but concentrate on solutions for (ii) based on Service Level Agreements (SLAs). Therefore we will give an overview on the state-of-the-art of SLA usage in Grids, highlight possible obstacles for the deployment of SLAs, and present a detailed example of a service improving the execution of workflows through the use of WS-Agreement to negotiate advance reservation of resources to execute workflow components.

1 Introduction

Workflows have become a common way to describe and organise a sequence of processes, tasks, applications, or services with specific interdependencies to build a complex job e.g. to deliver results of a complex simulation. Today's more stable Grid environments seem to be suitable to allow usage of non-local resources to execute such a workflow if its resource demand exceeds local capacities. However, as the degree of control over resources changes drastically when using resources not belonging to the own administrative domain, additional considerations have to be made and measures to be taken to allow a reliable, efficient, and automatic processing of these workflows. In the following sections we concentrate on these aspects.

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

In service-oriented Grids service provider and service consumer potentially belong to different administrative domains. In order to enter in a business relation they must among other things (i) establish trust between them, and (ii) set up mechanisms to create reliable and verifiable (in a commercial environment also audible) agreements with respect to the services requested, delivered and consumed. In this paper we describe a research environment where trust is based on the X.509 certificates issued by Certificate Authorities of the co-operating institutions.

The recently increasing discussion on Service Level Agreements (SLA) reflects the beginning of broader use of SLAs as an instrument to provide and access reliable services. However, SLAs are still far from a regular day by day use for establishing agreements between service provider and service consumer. We will give an overview on the state of the art, highlight possible obstacles for the deployment of SLAs, and present a detailed example of a service improving the execution of workflows through the use of WS-Agreement to negotiate advance reservation of resources to execute the workflow components.

1.1 Related work

The solution proposed here uses WS-Agreement [2] as a model to formulate and manage SLAs. Specified by the Grid Resource Allocation Agreement Working Group (GRAAP) of the Open Grid Forum [10], the Web Services Agreement Specification version 1.0 will soon be officially released and is already been used in various implementations [15, 13, 4].

One of these implementations is the meta-scheduling environment developed within the VIOLA [20] project. Built around the *MetaScheduling Service (MSS)* [21] a WS-Agreement-based framework has been realised to provide co-allocation of compute and network services for MPI applications. Since co-allocation is just a special case of scheduling a workflow (one, where all tasks start synchronously), we decided to enhance this framework to process more general kinds of workflows.

1.2 Remainder of the Paper

The remainder of the paper is organised as follows. In Section 2 we give an overview of the different approaches, use-cases, and state of the art SLA technologies. As an example how SLAs can improve workflow execution we describe in Section 3 the VIOLA MetaScheduling Service, its application and experiences made. In Section 4 we present results of experiments we carried out to evaluate our approach. An overview about further developments for the SLA-based service orchestration in Section 5 concludes the paper.

2 Service Level Agreements - State of the art

2.1 Technology Overview

Increasing effort has been put into research and development related to SLAs over the last years, contributions coming from various domains. For the creation of SLAs in Web Services-based business environments Web Service Level Agreements [14] (WSLA) has been proposed as a language to specify SLAs, a system to monitor the compliance of a provided service with a service level agreement, and a workload management system that prioritises requests according to the associated SLAs. A WSLA document contains one *Parties* section; it may contain multiple *Service Definitions* and one *Obligations* section. WSLA supports a distributed model of monitoring the SLA.

Another approach is SLAng [12] being developed at the University College London. SLAng essentially comprises an XML schema which can be used stand-alone for SLA definitions or together with WSDL or BPEL4WS [18]. SLAng includes QoS metrics to describe EPRs of the parties involved by providing contractual information and technical QoS information including the metrics using *Service Level Specification* (SLS). As the metrics are hard-coded in the XML-scheme SLAng is not very flexible. A framework for monitoring SLAs has not yet been defined.

In addition the Grid domain has addressed SLAs over the last years. A framework integrated into the Globus Toolkit has been proposed in [7]. It is based on the SLA negotiation protocol defined within the Service Negotiation and Access Protocol (SNAP) [6] which is based on the General-purpose Architecture for Reservation and Allocation (GARA) [9]. The first implementations used a proprietary format for specification of SLAs. However, it is expected that WS-Agreement will be used in later versions of the Globus Toolkit.

The European NextGRID project believes that the existing approaches to create SLAs do not sufficiently cover business aspects [17]. SLAs should therefore also contain non-functional terms. The authors propose to view the

service from different perspectives that distinguish between the customer's and the service provider's view. The shared view is defined by the SLA and will principally contain business terms. SLAs are only to be expressed in terms of business level objectives (BLOs). Non functional terms are used to build the business relationship between customer and provider and help providing a differentiating factor between service providers.

WS-Agreement as proposed by the OGF provides a domain-independent and standard way to establish and monitor Service Level Agreements. We decided to use WS-Agreement as a mechanism for advance reservation of resources for workflow execution for several reasons, the most important being: (i) is the result of the only active standardisation effort for a framework supporting interoperable SLA specification, (ii) it is used or considered to be used in many other projects, (iii) it is extensible and adaptable to arbitrary domains due to pluggable term languages, and (iv) due to the possibility to define guarantee terms and business values it might be used in business or service oriented environments thus allowing an smooth migration from research application to business use. As it is the foundation for our approach we describe it in greater detail in Section 2.3.

2.2 Defining Service Level Objectives and Penalties

Service-level agreements are intended to provide advance knowledge about a certain quality of a service prior to its use. Thus, the definition of the expected or required service-level is an integral part of all SLA approaches. While the SLA will also include additional information like e.g. technical data, provider and consumer information, the actual definition of the guaranteed agreement terms is one of the main aspects for using SLAs [8, 13].

Service Level Objectives (SLO) describe the condition over available service terms that must be met. Such objectives can consist of simple conditions for single service attributes (e.g. the guarantee of a minimum bandwidth for a network link). However, also complex service level objectives can be conceived that might require combined, complex conditions (e.g. the combined processing power of several processors and the available main memory must meet minimal requirements).

While these SLOs alone already allow the specification of guaranteed quality requirements, it has to be considered that a single SLA might contain several of these objectives from which only some may apply under certain conditions. Thus, the notion of qualifying conditions or rules can be found under which a certain SLO applies. The definition of several SLOs with corresponding qualifying conditions allows the modelling of complex requirements in SLAs.

While the service-level objectives provide us with the ability to define guaranteed quality of service, it is often necessary to identify the importance of these guarantees. The situation might occur that one party is not able to fulfil a guarantee, either during the negotiation towards an SLA or after a commitment. Thus, a business value might be associated with the requested service level objective to allow either the trade-off between several objectives or the identification of penalties for violating a guarantee. The impact of an SLO violation might differ depending on its importance.

The inclusion of such penalties is crucial for business relevant application scenarios to cover the liability for guarantees. A single SLA might be an important building block in a broader application scenario with complex dependencies and followup-cost if it is broken. As an example one might consider an SLA for a certain network bandwidth which is used in a complex application. Here, a consumer might create a set of SLAs with different providers that guarantee the availability of their resources for a certain time frame for a certain amount of money. The violation of the single SLA by the network provider might render the complete remaining SLAs useless for the consumer, while these independent and bilateral SLAs will require the consumer to pay for them. Therefore, it might be necessary to cover such risk with associating penalties for SLA violation.

For the sake of completeness, we now introduced SLOs and penalties which are considered key aspects for business oriented scenarios. In the following we do not further exploit SLOs and penalties but focus in the next section on the general foundation of supporting SLA management based on WS-Agreement. SLO optimisation and its trade-offs in penalties and risk management is beyond the scope of this paper and an important topic for future research.

2.3 WS-Agreement

The objective of the WS-Agreement draft specification defined by the GRAAP Working Group is to provide a domain-independent and standard way to establish and monitor Service Level Agreements. The specification comprises three major elements: (i) a description format for agreement templates and agreements, (ii) a basic protocol for establishing agreements, and (iii) an interface specification to monitor agreements at runtime.

A service defined in the agreement is specified as a *Service Description Terms* (SDT). Service description terms can be a reference to an existing service, a domain specific description of a service, or a set of observable properties of the service. Multiple SDTs describe different services that are provided within the same agreement. Dependencies between these SDTs can be described by using Guarantee Terms. Guarantee terms specify non-functional characteristics of a service in the service level objectives, an optional qualifying condition under which objectives are met, and an associated business value specifying the importance of meeting these objectives. Additionally, guarantee terms comprise a service scope, which defines a list of services a guarantee term applies to. Thus guarantee terms can be used for defining dependencies between different service description terms within an Agreement, or even to specify dependencies to existing agreements by using *Service References* to address related agreements.

Guarantee terms over multiple SDTs can basically be used to model a specific QoS requirement within a *Service Level Agreement*. For example a guarantee term that references multiple service description terms and specifies that these SDTs have to be executed in parallel, defines at least a co-allocation of the described services. On the other hand, guarantee terms that incorporate Service References can be used to model decisions based on the outcome of the related SLA. That is, a specific SDT or a set of SDTs of an agreement may only become active, if the related agreement was processed successfully. Of course, this can also be done for negative decisions.

Therefore guarantee terms are the key for describing workflows within WS-Agreement. Since the content of the Guarantee Terms is largely free per definition, one can consider every necessary extension for a workflow representation as a part of the Guarantee terms. Therefore the WS-Agreement framework can be used for describing workflows within one SLA, or even to compose workflows from multiple SLAs by using service references.

3 SLA based service provisioning for workflows

In this section we describe how SLAs can be used to improve the processing of distributed workflows through negotiation and advance reservation of the resources or services required to perform the different tasks of a workflow.

3.1 The workflow resource problem

There are many application scenarios in which not just a single resource is required but a set of resources with certain time dependencies. Assuming that multiple resources are provided by several resource providers, such workflows complicate the management of resources since resource access must be synchronised in advance to allow reliable workflow execution.

SLAs are one instrument which can be used to reserve resources in advance [15]. The time requirements and dependencies can be modelled in the SLA to guarantee the resource availability [16]. The dependency can pose additional importance of single SLAs in a workflow; as the different SLAs may rely on each other, the individual business value of a single SLA might be increased. As mentioned before, this can yield to the need for including higher penalties for individual SLAs.

A main problem for the management of workflows is the negotiation towards the matching SLAs. While the creation of a single SLA might be much easier as both parties can settle on a set of agreement terms, this becomes more complicate if the terms must match or correspond to other SLAs that are in negotiation. For instance, if three resources must be concurrently available in the same time frame, this time frame must be first identified. It then must be assured that all three SLAs are finally committed (or none of them) as it has to be prevented that the consumer gets an eventually unusable subset of SLAs.

On the other hand, if three resources necessary to perform the three tasks of a workflow must be available in a sequential order at different times also three SLAs have to be created. This time, it must be assured that the time dependencies between the tasks of the workflow are reflected in the SLAs leading to the reservations. Figure 1 depicts this situation. The MetaScheduling Service has to assure that $t_5 >= t_3 >= t_2$ in order to respect the dependencies. At the same time the MSS has to negotiate the individual start times as close as possible to the end times of the respective previous tasks in order to minimise the total duration of the workflow execution (which is a prevalent objective scheduling jobs and workflows).

3.2 Negotiating resource usage

The negotiation required for the resource reservations (as presented in Section 3.1) is based on SLAs describing the requirements of the workflow's tasks and the dependencies between them. The SLAs are derived from the resource

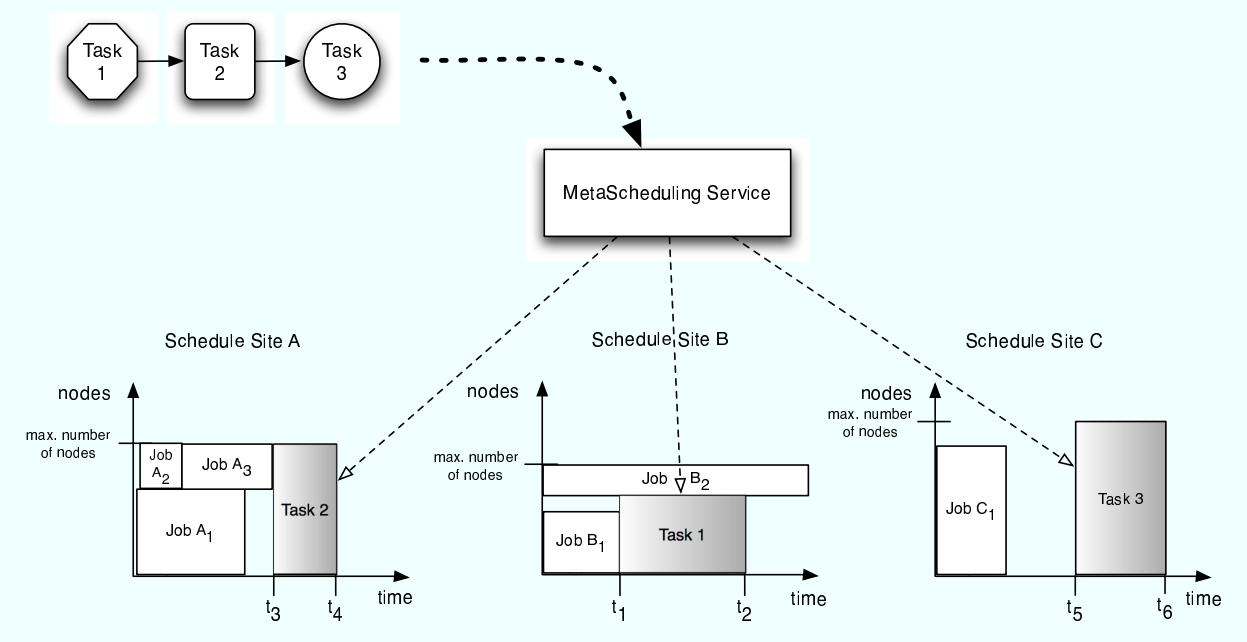

Figure 1: Mapping of a workflow to three resources respecting the time dependencies

requests specified by the user, the format of which depending on the Grid middleware used. In the case of Globus, for example, the user includes resource requirements into a WS GRAM job description, in the case of UNICORE they are embedded into an Abstract Job Object (AJO). These requests are then to be mapped to the respective SDTs. For the specification of the SLA for a workflow WS-Agreement is used (see Section 2.3). In the following description of the negotiation process we refer to the UNICORE environment of the VIOLA project. However, as the MSS is able to communicate with a specific GRAM plug-in in the same way, the process is the same for Globus environments. The following steps are executed to establish an agreement between client and MSS:

1. The UNICORE client requests an agreement template from the MSS.
2. The MSS delivers an EPR of the agreement to the client.
3. The client fills in the template with the workflow-specific details. With respect to the description of resource requests JSDL [3] is used within the SLA and a library to map the UNICORE-specific terms to JSDL.
4. The client sends the completed agreement template back to the MSS.
5. The MSS starts to negotiate the reservation of the resources for the workflow tasks with the respective local scheduling systems of the resource providing sites according to the SLOs described in the agreement. The negotiations are also based on individual SLAs. This time the MSS plays the role of the client vis-a-vis the resource providers.
6. Once all individual SLAs between the MSS and the different resource providers are in place the MSS accepts the agreement proposed by the client. Otherwise the client is informed that the MSS can not accept the agreement. In this case the user is notified and may modify the requirements for his workflow to adopt to the situation and repeat the negotiation from the beginning.

In case of one or more individual agreements with resource providers fail, the MSS will cancel all other accepted agreements. This is due to the fact that the current version of WS-Agreement does not allow modification of accepted agreements thus all agreements have to be negotiated from the beginning.

3.3 MSS Implementation

As mentioned in Section 3.2 the German VIOLA project develops among other components a meta-scheduling environment providing resource reservations based on WS-Agreement. The immediate objective of this development is to co-allocate computational and network resources in a UNICORE-based Grid, but we designed the environment to support arbitrary types of resources and to be integrated into different Grid middleware systems. The main integration effort to access other middleware, like e.g. Globus, is to implement the client-side interface of the MetaScheduling Service. Since it is beyond the scope of this paper to explain the system in detail we refer to [21] for a complete architectural description of it and to [19] for the definition of the negotiation protocol currently implemented.

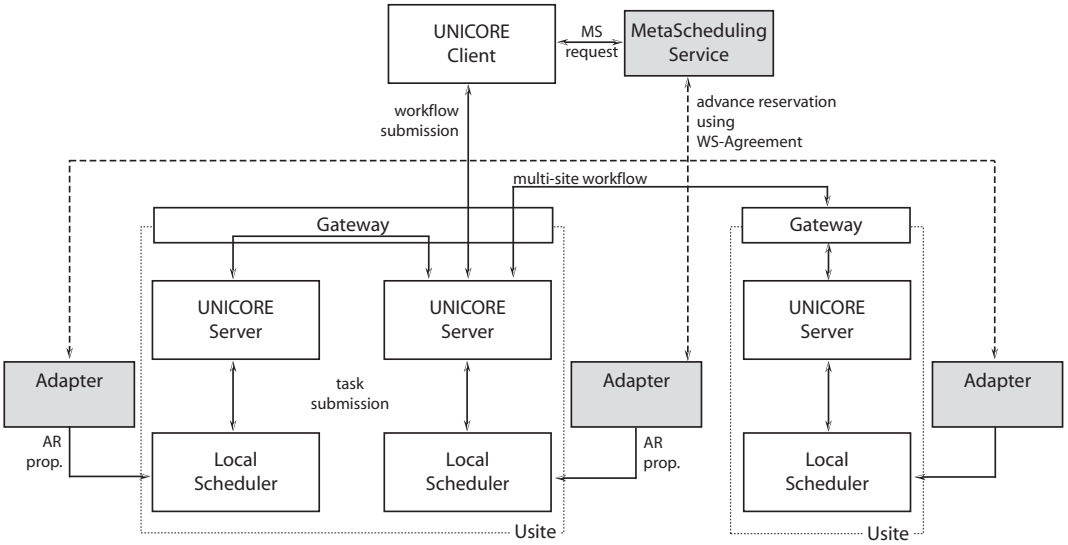

Figure 2: High-level meta-scheduling architecture

Figure 2 sketches the basic architecture of the meta-scheduling environment and its integration into the UNICORE Grid middleware. The VIOLA Meta-Scheduling Service communicates with a client application using WS-Agreement, it receives the workflow-specific resource requests as described in Section 3.2 wrapped into a Meta-Scheduling (MS) request, and it returns reservations for all of these resources. To interact with varying types of scheduling systems we use the adapter pattern approach. The role of an Adapter is to provide a single interface to the Meta-Scheduling Service by encapsulating the specific interfaces of the different local scheduling systems. Thus the Meta-Scheduling Service can negotiate resource usage by exploiting a single interface independent of the underlying resource type. To achieve this, the Meta-Scheduling Service first queries local scheduling systems for the availability of the requested resources and then negotiates the reservations across all local scheduling systems. These, in order to participate in the negotiation process, have to be capable and willing to let the MetaScheduling Service reserve resources in advance by offering data about job execution start and stop times, and provide at least partial access to their local schedules, e.g. by publishing information about available free time slots.

4 Results

In order to predict the gain in turnaround time for workflows with SLAs employing job dependencies in real world systems we have accomplished a series of experiments, where we compared the expected turnaround times for best effort jobs with those for advance reservation jobs. The test scenario is constructed as follows. As test environment we used a simple MSS setup consisting of three independent systems, each utilised with a basic load derived from real world log files [22] . Each SLA consists of 3 jobs where each job is scheduled on one system. The resource requirements of the jobs range from [6/8/8] to [48/64/64], where the numbers specify the number of requested nodes or CPUs for a job (workflow component) to be executed on one of the three systems. The runtimes of the jobs range from 60 minutes up to 240 min, within one test row all jobs have the same runtime. For best effort SLAs the first job is submitted to the resource management system and subsequent jobs are scheduled as soon its predecessor has

finished. For advance reservation SLAs the jobs are scheduled sequentially at submission time by using the first fixed fit strategy. Once the first job was scheduled the estimated end time of the job is determined and the next subsequent job is scheduled using the determined end time of its predecessor. The start times of a job must not change after it was scheduled. Figure 3 shows the results of our experiments.

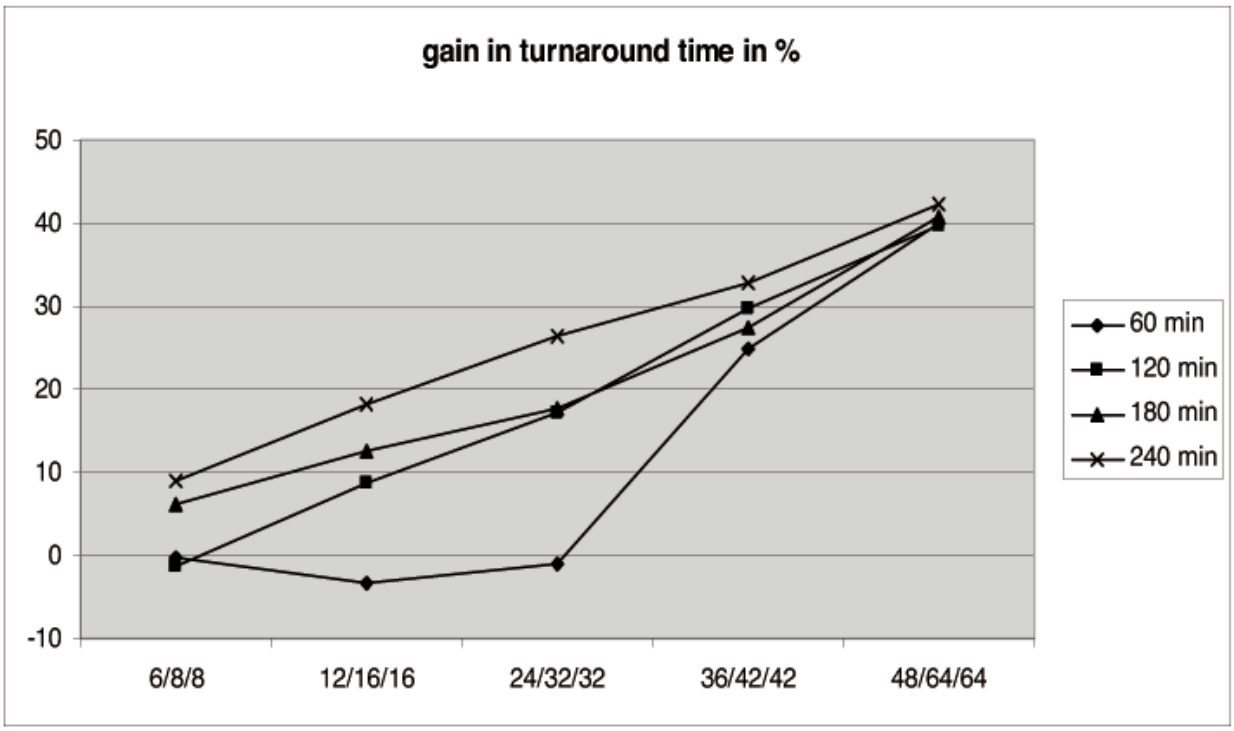

Figure 3: Gain in turnaround time using Advance Reservation

The results of the experiments proved true the expectation that a reduction of turnaround times is achievable when doing advance reservation of resources for a workflow. The achievements observed were gains in turnaround time of up to 45% depending on the number of CPUs and the time needed to execute the components of a workflow.

At a first glance surprisingly for workflows with low resource requirements and short runtimes it turns out, that the best effort scheduling strategy behaves better in our experiments than the advance reservation strategy. However, this results from the fact that the advance reservation jobs in our test scenario are scheduled at a fixed start time and therefore do not profit from potential backfilling possibilities, e.g. prior jobs that occupied the system resources finish earlier.

This problem can be solved easily if scheduling systems allow backfilling for advance reservation jobs until a requested time (here the estimated end time of the predecessor job) and furthermore provide the functionality to update this time later on. Given that this functionality is in place a Grid scheduler can be constructed employing the advance reservation strategy, where the turnaround time is at least as good as for best effort jobs, and the gain in the turnaround time might even increase compared to our test results.

5 Future Perspectives

Work on performance evaluation for workflow planning with SLAs will be continued considering the observations described above. Future work therefore will mainly focus on the areas:

- finishing the ongoing work on implementation and measurements
- moving to support a standard workflow description language (or a suitable and efficient mechanism to map different languages onto the one selected for the UNICORE environment)

- extending WS-Agreement to allow for a more flexible negotiation process and to support modifications of existing agreements.

Addressing the first aspect we will investigate the ongoing activities related to workflows in the CoreGRID [5] Institute on Grid Information, Resource and Workflow Monitoring Services. One of the institute's research group's is working on *Compatibility and Conversion of different Grid Workflow Description Languages* [11]. Based on the results we will decide whether to adopt a de-facto standard like BPEL (or BPEL4WS) or to integrate the mapping mechanism and to stay with the UNICORE workflow description language. In either case only a single procedure to convert a workflow description to a WS-Agreement will be necessary in the user's client, e.g. a UNICORE client, while allowing the user to select the most appropriate language to describe his workflow.

Experiences made so far show that the number of different resources needed for a complex workflow will often be higher than for a single application that is distributed across several resources, like in the co-allocation scenarios of VIOLA. The process of negotiating the reservation is based on WS-Agreement version 1.0 which does not support changes of the offer or later re-negotiation. Thus in case of a service provider not being able to match exactly the requirements of a client now the agreement is cancelled and has to be initiated from the beginning. Even worse, in case of a workflow negotiation including several service providers currently the whole workflow usually will not complete if one of the service providers is unable to deliver the service agreed upon and re-negotiation is not possible.

Extending WS-Agreement to allow for a more flexible negotiation process and to support modifications of existing agreements will help to overcome the potential waste of resources. The GRAAP working group recently started working on these extensions for the next versions of WS-Agreement. The group is gathering requirements and contributions, e.g. as drafted in [1] to define a standardised extension of the current protocol.

6 Acknowledgments

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] M. Aiello, G. Frankova, and D. Malfatti. What's in an Agreement? A Formal Analysis and an extension of WS-Agreement. In *Service-Oriented Computing – ICSOC 2005*, volume 3826 of *LNCS*, pages 424–436. Springer, December 12–15, 2005.

[2] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. WS-Agreement - Web Services Agreement Specification, September 20, 2005. <https://forge.gridforum.org/projects/graap-wg/document/WS-AgreementSpecificationDraft.doc/en/31>.

[3] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) Specification v1.0. Grid Forum Document GFD.56, Global Grid Forum, November 2005.

[4] The Business Grid Project. Web site. 30 Mar 2006 <http://www.ipa.go.jp/english/softdev/sixth.html>.

[5] CoreGRID. Web site. 14 Aug 2006 <http://www.coregrid.net/>.

[6] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In *Job Scheduling Strategies for Parallel Processing (Proceedings of the 8th International JSSPP Workshop)*, volume 2537 of *LNCS*, pages 153 – 158. Springer, 2002.

[7] K. Czajkowski, I. Foster, C. Kesselman, and S. Tuecke. Grid Service Level Agreements. In J. Nabrzyski, J. M. Schopf, and J. Weglarz, editors, *Grid Resource Management*, pages 119–134. Kluwer Academic Publishers, 2004.

[8] A. Dan, K. Keahey, H. Ludwig, and J. Rofrano. Guarantee Terms in WS-Agreement. Technical report, Grid Resource Allocation Agreement Protocol (GRAAP) Working Group Meetings, 2004.

[9] I. Foster, M. Fidler, A. Roy, and V. Sander. End-to-end quality of service for high-end applications. *Computer Communications, Special Issue on Network Support for Grid Computing*, 2002.

[10] Grid Resousource Allocation Agreement Protocol Working Group. Web site. 07 Aug 2006, <https://forge.gridforum.org/sf/projects/graap-wg>.

[11] Compatibility and Conversion of different Grid Workflow Description Languages. Web site. 14 Aug 2006 <http://www.gridworkflow.org/snips/gridworkflow/space/Workflow+Description+-Languages/Compatibility+and+Conversion/>.

[12] D. Lamanna, J. Skene, and W. Emmerich. SLAng: A Language for Defining Service Level Agreements. In *Proceedings of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems (FTDCS'03)*, pages 100 – 106. IEEE Computer Society Press, May 2003.

[13] H. Ludwig, A. Dan, and B. Kearney. Cremona: An Architecture and Library for Creation and Monitoring WS-Agreements. In *ICSOC04, New York*. ACM, 2004.

[14] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. IBM, USA. <http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf>.

[15] H. Ludwig, T. Nakata, O. Wäldrich, Ph. Wieder, and W. Ziegler. Reliable Orchestration of Resources using WS-Agreement. In *Proc. of the 2006 International Conference on High Performance Computing and Communications (HPCC-06)*, volume 4208 of *LNCS*, pages 753–762. Springer, September 13–15, 2006.

[16] J. MacLaren, R. Sakellariou, K.T. Krishnakumar, J. Garibaldi, and D. Ouelhadj. Towards Service Level Agreement Based Scheduling on the Grid. Workshop on Planning and Scheduling for Web and Grid Services (in conjunction with ICAPS-04), 2004.

[17] P. Masche, P. Mckee, and B. Mitchell. The Increasing Role of Service Level Agreements in B2B Systems. In *2nd international conference on web information systems and technologies*, Setubal, Portugal, April 2006.

[18] S. Thatte (ed.). Business Process Execution Language for Web Services version 1.1. <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>.

[19] A. Streit, O. Wäldrich, Ph. Wieder, and W. Ziegler. On Scheduling in UNICORE – Extending the Web Services Agreement based Resource Management Framework. In *Proc. of Parallel Computing 2005 (ParCo 2005)*, Malaga, Spain, September 13–16, 2005. To appear.

[20] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN, 2006. Web site. 29 Mar 2006 <http://www.viola-testbed.de/>.

[21] O. Wäldrich, Ph.Wieder, and W. Ziegler. A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources. In *Proc. of the Second Grid Resource Management Workshop (GRMWS'05) in conjunction with the Sixth International Conference on Parallel Processing and Applied Mathematics (PPAM 2005)*, volume 3911 of *LNCS*, pages 782–791. Springer, September 11–14, 2006.

[22] Parallel Workloads Archive - San Diego Supercomputer Center (SDSC) SP2 log, 2006. Web site. 29 Mar 2006 <http://www.cs.huji.ac.il/labs/parallel/workload/logs.html>.