Towards SLA-supported Resource Management

Peer Hasselmeyer[1], Bastian Koller[2], Lutz Schubert[2], Philipp Wieder[3]

[1] C&C Research Laboratories, NEC Europe Ltd., 53757 Sankt Augustin, Germany, hasselmeyer@ccrl-nece.de
[2] Höchstleistungsrechenzentrum Stuttgart, Allmandring 30, 70550 Stuttgart, Germany, {koller|schubert}@hlrs.de
[3] Research Centre Jülich, 52415 Jülich, Germany, ph.wieder@fz-juelich.de

Abstract. Achievements and experiences in projects with focus on resource management have shown that the goals and needs of High Performance Computing service providers have not or only inadequately been taken into account in Grid research and development. Mapping real-life business behaviour and workflows within the service provider domain to the electronic level implies focusing on the business rules of the provider as well as on the complexity of the jobs and the current state of the HPC system. This paper describes an architectural approach towards a business-oriented and Service Level Agreement-supported resource management, valuable for High Performance Computing providers to offer and sell their services. With the introduction of a Conversion Factory the authors present a component that is able to combine the Service Level Agreement, the system status, and all business objectives of the provider in order to address the business needs of service providers in the Grid.

1 Introduction

Current solutions for resource management of High Performance Computing (HPC) environments were mainly developed neglecting the business needs and goals of service providers. The introduction of Service Level Agreements (SLAs) [1, 9] provided an instrument to express business-related terms, but it became clear that configuring the system only according to an SLA does not solve the problem of automatic resource configuration at all.

The main purpose of SLAs is to define certain Quality of Service (QoS) parameters in a way that appropriate service levels can be maintained during interaction with customers. SLAs are therefore an important tool for automatic business enactment and QoS management, in particular monitoring the performance of services and detecting violations. By using SLAs, the customer has a document which states certain quality properties, in most cases bound to penalties for the service provider failing to deliver this quality. But the creation of the SLA, as well as the configuration of the system, is bound to different aspects. The authors aim specifically at developing an architecture for SLA-supported resource management which takes three major aspects into account: (a) The complexity of the job which is bound to (b) the availability of resources on the service provider's system, and all based on (c) the respective *Business Level Objectives* (BLOs) [7, 10] of the provider.

Nowadays, the usage of SLAs requires translation of SLA terms. Those terms are mostly stated as high level technical to low level technical terms and they can be used to derive the configuration of the system. SLA-specific configurations have to be based on the Business Level Objectives of the service provider while at the same time they have to respect the current state of the system resources (e.g. the load of the system). This state information is referred to as *infrastructure knowledge* throughout the paper.

A number of approaches towards this goal have already been proposed. Inter alia projects like TrustCoM [12] and NextGRID [11] use SLAs within the service provisioning process, the latter e.g. for automatic generation of SLAs with respect to Business Level Objectives which are translated into policies. Based on this idea, we describe how infrastructure configuration can be automated taking into consideration SLAs as well as the service provider's environment.

The paper starts by describing a business scenario and presents an overview how SLAs are currently handled at a service provider. Section 3 details the relationship between SLA contents and resource configuration. The architectural approach proposed in this paper is presented in Section 4, introducing the different components of this system as well as explaining a generic mapping process. Section 5 provides details on the usage of this approach for Service Level Agreement negotiation as well as for system configuration purposes. Finally, the concluding section discusses limitations and advantages of the proposed architecture, specifically related to potential upcoming implementations of the architecture introduced here. Additionally, references to ongoing research are provided for areas that are not in the focus of our work, but which are of essential importance once the proposed architecture is implemented.

2 Business Scenario

2.1 Description

The approach presented in this paper is based on actual business requirements which we shall exemplify in the context of a scenario taken from the TrustCoM project. In this scenario, an aircraft manufacturer intends to re-design the seats of an existing air plane. As this task involves complex calculations, a third-party HPC service is contracted to take over task-specific computations (i.e. to execute a job). In the given example, the HPC provider allows customers to run computational jobs at a specific Quality of Service. For this kind of providers, configuration and maintenance of the underlying HPC infrastructure is a particularly complex issue – specifically when performed autonomously and when the respective task is unknown to the service provider, i.e. no a priori configuration information exists to be reused.

2.2 Limitations of existing solutions

Existing solutions commonly rely on the HPC provider to negotiate a contract with a consumer, in case of the example with the aircraft manufacturer. As the provider has no experience with that particular job, he has to perform a complexity analysis of the task. The authors of this paper are aware of the fact that such complexity analyses are hard to automate. We therefore only sketch an idea of "semi"-automation of such analyses by using a database that stores already calculated complexities (see Section 4). Unknown tasks have to be calculated manually. From the complexity information, an administrator has to derive the resource requirements for this task – taking into consideration not only complexity but also provider-specific knowledge about previous executions of a job (if available).

Having the complexity received from an *Analyst*, the provider's *Administrator* can usually (manually) map the requirements of the job to the provider's resources. As an extension to our example of re-designing the air plane's seats, let us assume that the job belongs to a (provider-defined) complexity class C. The provider would use his knowledge of jobs with this complexity and calculate the requirement of having, say, 64 nodes available for 24 hours to execute this kind of job.

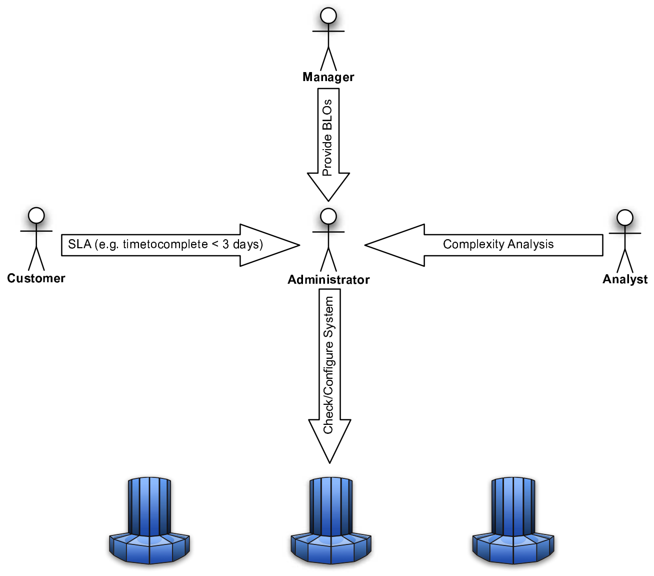

Fig. 1. A human-centric approach to resource configuration at service providers

It is important to mention that this kind of calculation reflects only the theoretical approach towards the manual mapping. In real business cases, such a calculation is also influenced by the HPC providers policies (mainly the Business Level Objectives). Business Level Objectives are abstract goal definitions of the respective business party, generally defined by a *Manager* before any business is conducted. Having an HPC provider, a BLO could for instance be "*maximise workload on this machine*". The BLOs influence translation from SLAs to configuration parameters, as in addition do information about the system and potential constrains from the customer, who may, for the sake of the example, request the job to be finished within three days. With this information the Administrator can determine whether the "*64 nodes*" requirement can really be fulfilled within the valid time period for this job or, depending on the provider's Business Level Objectives, whether he actually wants to fulfil it. Based on his calculations, the Administrator can than configure the resources.

Current solutions require involvement of different parties to collect the necessary information to execute the job according to the requirements of a customer. Fig. 1 shows the different parties and tasks they carry out as described in this section. It can be seen that this approach lacks automatic mechanisms to carry out the previously outlined tasks.

3 Dependency of Resource Configuration on SLAs

In order to configure his resources in a sensible and efficient way, a service provider has to take different facts into account. One set of aspects is defined by Service Level Agreements. But SLAs are only one part of the whole picture. The configuration of a system depends on other aspects as well: the Business Level Objectives of the service provider, the complexity of the job and the current system status. In the following we discuss how far SLAs influence the configuration of the system.

Terms within SLAs can usually not be used directly for configuring the affected resources. A Service Level Agreement can consist of a set of abstract terms which, unfortunately, mean different things to different providers. The term "performance", for example, is defined differently by different parties and it is therefore calculated and provided in different ways depending on the respective infrastructure. By going from abstract terms to the infrastructure layer, we need a mapping of high level terms to a low (technical) level (resulting in an *Operational Level Agreement* (OLA)). This mapping is inevitable as the service provider has to understand what he needs to provide in order to fulfil the conditions of an SLA in terms of processing power, processing time, etc.

We foresee an approach that integrates infrastructure and business-specific knowledge to allow automatic and autonomous conversion of SLA terms into configuration information. As discussed below, such a conversion process may also support the management of the infrastructure during operation.

SLAs have strong impact on the configuration system, in particular if service provision has to be dynamic and on-demand. TrustCoM and NextGRID currently examine how Service Level Agreements can be created and used in a way that is adequate for both customers and service providers. In these projects

SLAs are not negotiable. This simplifies selection and creation of SLAs, by using e.g. a protocol and SLA representation like WS-Agreement [1], and enables the use of databases to store (static) configuration information. However, studies of real business use cases have shown that pre-defined configurations can only be used in a limited set of scenarios, as the configuration of the service provider's system does not only depend on the Service Level Agreements, but also on the job type(s) and the current workload.

Although, at the time of writing, Business Level Objectives like "*utilise my resources a hundred percent*" or "*get the maximum profit, while spending as little money as possible*", are not supported by available resource management systems, current research activities to solve this problem are on-going in NextGRID, TrustCoM, and the upcoming project BREIN [2]. For the purposes of this paper, we assume that BLOs are represented as documents in a database and that they can be retrieved easily. Additionally, a certain degree of knowledge of the service provider's infrastructure is required, e.g. whether and how the service provider can execute the requested jobs in time. In summary, a configuration system that builds on a "base" configuration represented by BLOs and the complexity analysis of a job seems like a promising approach towards SLA-supported resource management.

4 Mapping Service Level to Operational Level Agreements

The outlined scenario has high demands on the HPC provider's capability of configuring its resources on-the-fly. Related to discussion in the previous section, this one presents our architectural approach to mapping Service Level Agreements to Operational Level Agreements.

4.1 An Architecture for SLA-supported Resource Management

Current research in the Service Level Agreement area deals mainly with high-level application-specific terms as SLA content (e.g. "time-to-complete" as used in NextGRID). These terms are used since the customer should not know and in most cases does not want to know the details of the service provider's configuration. It is therefore mandatory to map those high level-terms to the low-level technical layer usable for calculating the requirements on the system.

Our architecture (see Fig. 2) is an approach to automate the process of configuring a service provider's system based on:

– the BLOs of the service provider,
– the *complexity* of the job,
– the *configuration knowledge* of former job runs (stating something like: "*for this job, I need the following system resources*"), and
– the Service Level Agreement itself.

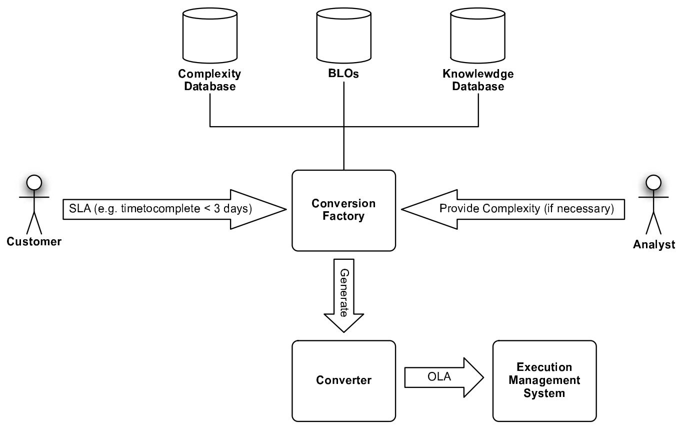

Fig. 2. An architecture for SLA-supported resource management

The resulting set of OLAs will be the basis for the system configuration. The translation of OLAs to actual system configurations is performed by the *Execution Management System* (EMS). The EMS is not part of our proposed architecture but the entity responsible for consuming and interpreting the Operational Level Agreements.[1]

The proposed architecture consists of the *Conversion Factory*, the *BLO Database*, the *Knowledge Database*, the *Complexity Database*, and the *Converter*. The main component of our proposed architecture is the Converter. The Converter instance is created based on the Business Level Objectives of the service provider, the complexity of the job and the infrastructure knowledge. Business Level Objectives represent the goals of the service provider. In an ideal world, the service provider's main goal would be to satisfy the customer's needs. But in the business world, both customers as well as service providers have additional internal goals. All goals need to be observed when negotiating and executing a business transaction.

The job of the Converter is to provide the information to (a) check resource availability, (b) configure the system and (c) monitor the status. Thus a Converter needs to perform basically the same task multiple times depending on changes of the provider's infrastructure, in order to support the business transaction at the best. In particular monitoring is dependent on the business relation (as it is different for each individual SLA).

We therefore introduce a "Conversion Factory" that instantiates individual Converters. Each Converter is specific to the particular needs of a business relationship. For this purpose the Conversion Factory queries the different databases to retrieve information about potentially available, pre-calculated complexities

[1] The function of the EMS is analogous to that of the OGSA Execution Management Services which "are concerned with problems of instantiating and managing, to completion, units of work" [4].

or previous business transactions. In particular we see the need for a Knowledge Database, a Complexity Database, and a BLO Database. The Knowledge Database stores information about system requirements for different complexity classes, generated by running "similar" jobs in the past. In addition, the Knowledge Database in our approach represents an information service which also provides data about the current status of the system. Realisations of the architecture described may obviously choose to implement such a service separately or to exploit the respective service of the middleware in use. The Complexity Database stores previously calculated complexities of jobs; it could, as far as our scenario is concerned, contain the assignment of the complexity class C to t he aircraft manufacturer job. The BLO Database stores the Business Level Objectives of the service provider.

Obviously, the Complexity Database does not contain all possible job-complexity pairs. Complexity calculation of so far "unknown" jobs is therefore needed. Generally, it is impossible to predict the behaviour (and hence complexity) of an unknown application. Potential solutions include letting the job run with a reduced quantity of resources to estimate the time requirements (relative to the system's capabilities) – however, such an approach neglects for example the relationship between input data complexity and time requirements. Manual processing will therefore be needed in most cases. This does nevertheless not diminish the improvements brought about by our Converter.

While the Knowledge Database and the Complexity Database are exploited to reduce the mapping complexity and thus speed up this process, the purpose of the BLO Database is to provide information on the service provider's goals and their relative importance. As mentioned before, BLOs represent the policies to be taken into consideration when generating configuration information. Such priorities could directly influence configurations, e.g. with statements like "*jobs from companies X, Y, Z get higher/lower priority*".

Please note that, as opposed to the Knowledge and Complexity Databases which are ideally populated automatically, the service provider wants to retain control of the definition of its own BLOs. Automatic definition of BLOs does therefore not seem to be a sensible option.

As described, the Converter is created based on the SLA, the Complexity, and the infrastructure knowledge. It provides three interfaces:

1. **Availability**. This interface is used to check the availability of an offered service for negotiation purposes.
2. **Status**. This interface provides monitoring data from the EMS converted to SLA level terms to check the status of the service.
3. **Configuration**. Once an SLA is negotiated, this interface is used to convert the SLA-specific parameters into valid configuration information usable as input to the EMS based on the pre-configured conversion mechanisms.

Once instantiated, the Converter is available until either the negotiation process fails or the service provisioning period expired.

4.2 The mapping process

This section gives an overview of the processes which are executed by the Conversion Factory to create an appropriate Converter instance. The different processes executed during the mapping process are:

- **Complexity analysis of the job**. When an SLA (respectively an "offer") is sent to the Conversion Factory, the complexity of the job is retrieved. This can either be done by querying the Complexity Database or a (human) analyst.
- **Knowledge Database lookup**. The Knowledge Database contains information about the infrastructure and the available resources. In addition it provides information about previous configurations of the system with respect to different job complexities.
- **Application of BLOs to the SLA (template)**. As our approach concentrates on the service provider domain, the customer is only implicitly relevant (through the SLA – which to maintain is also in the service provider's interest). The process of creating the Converter instance is influenced by the Business Level Objectives of the service provider.

5 Usage

The proposed architecture can be used to support SLA negotiation as well as configuration of the service provider's system. This chapter will give a short overview of the capabilities of the Conversion Factory / Converter approach and its expected use within a business interaction.

1. **Negotiation**. During the negotiation of a Service Level Agreement, the service provider has to figure out whether it can fulfil a service request according to the SLA terms. Our architecture supports this by checking the availability of the requested resources according to SLA and infrastructure status. The mapping and Converter creation process takes place as described in Section 4.2. Using the Converter, the negotiation component of the service provider can check the status of the system and thus the availability of the resources with respect to the SLA, by using the provided interface (see Section 4.1).
   Coming back to our example scenario: The aircraft manufacturer asked for re-designing the air plane's seats within three days. We assume that the Complexity Database has an entry for the respective job, providing the Conversion Factory the job's complexity class C. A Knowledge Database query then translates C into a resource requirement of 64 nodes for 24 hours. At this stage, the BLOs and the system's status have to be taken into consideration. As "time-to-complete" is set to three days, and given that the respective HPC resources are available, the BLOs now determine the level of service provided and communicated to the customer as the job is accepted. In case the system's schedule (or any BLO) does not allow the execution of the job, the provider has to reject the request or create a counter-offer.

2. **Configuration**. Having made an agreement, the service provider has to configure its system according to the Service Level Agreement, the BLOs, and the job complexity. In case the Converter was already instantiated during negotiation, it can be used to map the SLA terms to the infrastructure-specific (technical) parameters, which will be sent to the Execution Management System in order to configure the systems. This configuration is subject to the BLOs of the provider. Imagine the aircraft manufacturer being a "high priority" business partner and the objective of the service provider is to provide "*minimal response time for high priority business partners*". This BLO would cause the EMS to run the job as soon possible. In case of the customer being a "low priority" business partner (and assuming that the job is not a "rigid" [3] one), less than 64 nodes may be allocated for more than 24 hours to keep resources in reserve for higher priority customers.
3. **Execution (Monitoring)**. With the introduction of the status interface, we enable the service provider to use the Converter for monitoring during the execution of the service. Getting the data from the EMS, the Converter delivers the data mapped to SLA terms to the service provider's management system.

   In case of high priority jobs, monitoring could be used for violation prevention. If some of the allocated nodes fail, the EMS could be notified of an impeding SLA violation. With that warning, the EMS can adjust the system accordingly, e.g. by migrating (parts of) the job, to ensure the fulfilment of the SLA.

6 Conclusions

In this paper we introduce an architecture for SLA-supported, half-automatic resource management. The architecture reflects the business needs of the service provider and aims at a more automatic and autonomous resource configuration through the introduction of a Conversion Factory. To achieve this, Service Level Agreements are mapped to provider-specific Operational Level Agreements with the aid of formalised business goals (Business Level Objectives), complexity analyses, and knowledge of previous configurations.

We are aware that the approach presented in this paper implies the availability of sophisticated methods and algorithms to realise the logic of components like the Conversion Factory and the Converter. Also the formalisation of SLAs, OLAs, and BLOs is still subject of various research activities of varying maturity.

With respect to the classification of jobs the Γ model described in [5] and the related scheduling architecture [6] may serve as a starting point towards the realisation of the complexity analysis as well as the resource assignment and configuration knowledge provision capabilities needed. Furthermore, our approach calls for services to map SLAs to OLAs while observing the provider's Business Level Objectives. In this case the authors are confident that semantic techniques as e.g. presented in [8] will help to provide appropriate solutions.

Acknowledgements

This work has been supported by the NextGRID project and has been partly funded by the European Commission's IST activity of the 6th Framework Programme under contract number 511563. This paper expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this paper.

The developments presented in this paper were also partly funded by the European Commission through the IST programme under Framework 6 grant 001945 to the TrustCoM Integrated Project.

References

1. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement) Version 2005/09, September 2005.
2. The BREIN project. Website, `http://www.ist-brein.org`. To be available soon.
3. D. G. Feitelson and L. Rudolph. Toward convergence in job schedulers for parallel supercomputers. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1162 of *LNCS*, pages 1–26. Springer, 1996.
4. I. Foster, H. Kishimoto, A. Savva, et al. The Open Grid Services Architecture, Version 1.0, January 2005. `http://www.ggf.org/documents/GFD.30.pdf`.
5. R. Gruber, P. Volgers, A. De Vita, M. Stengel, and T.-M. Tran. Parameterisation to tailor commodity clusters to applications. *Future Generation Comp. Syst.*, 19(1):111–120, 2003.
6. K.Cristiano, R. Gruber, V. Keller, P. Kuonen, S. Maffioletti, N. Nellari, M.-Ch. Sawley, M. Spada, T.-M. Tran, Ph. Wieder, and Wolfgang Ziegler. Integration of ISS into the VIOLA Meta-scheduling Environment. In S. Gorlach and M. Danelutto, editors, *Proc. of the Integrated Research in Grid Computing Workshop*, pages 357–366. Università di Pisa, November 28–30, 2005.
7. J. O. Kephart and D. M. Chess. The Vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.
8. L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, Budapest, Hungary, May 20–24, 2003. ACM.
9. H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification, 2003. `http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf`.
10. Ph. Masche, P. Mckee, and B. Mitchell. The Increasing Role of Service Level Agreements in B2B Systems. In *Proc. of WEBIST 2006 – 2nd International Conference on Web Information Systems and Technologies*, Setúbal, Portugal, April 11–13, 2006. To appear.
11. The NextGRID project. Website, 24 June 2006 `http://www.nextgrid.org/`.
12. The TrustCoM project. Website, 24 June 2006 `http://www.eu-trustcom.com/`.