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

1 Introduction

Current solutions of resource management for HPC providers were mainly developed without respecting the needs and business goals of the different service providers. The introduction of Service Level Agreements (SLAs) [1, 6] marked a first step into the right direction, but it became clear that configuring the system only according to an SLA does not solve the problem of automatic resource configuration at all. By using SLAs, the customer has a document which states a certain degree of quality properties, in most cases bound to penalties for the service provider failing to reach 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 bound to (b) the availability of resources on the service provider's system, and all based on (c) the respective *Business Level Objectives* (BLOs) [4] of the provider. Nowadays, the usage of the SLA implies the need for translation of the SLA terms, that are mostly in high level technical to low level technical terms, that can be used to develop the configuration for the system. It is of importance, that an SLA-specific configuration has to be based on the Business Level Objectives of the service provider while at the same time it has to respect the current status of the system resources (e.g. the status of the job queue).

A number of approaches towards this goal have already been proposed, e.g. automatic generation of SLAs with respect to Business Level Objectives which are translated into policies in NextGRID. In current IP projects like NextGRID [7] and TrustCoM [8] SLAs are used within the service provisioning process. Based on this idea, we describe in this paper how infrastructure configuration, taking into consideration the context of SLA usage and the service providers environment, can be automated.

The paper starts with describing a business scenario and giving 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 the mapping process. Section 5 provides details about 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.

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 as used in the IST project TrustCoM. In this scenario, an aircraft manufacturer is assumed to intend redesigning the chairs of an existing aeroplane. As such a task involves complex calculations, a third-party High Performance Computation (HPC) service is contracted to take over task-specific computations. In the given case, the HPC provider allows customers to deploy computational jobs maintaining a specific Quality of Service. For providers such as this, configuration and maintenance of the underlying HPC infrastructure is a particularly complex issue – specifically if to be performed autonomously and if the respective task is unknown by the service provider, i.e. no configuration information may be reused.

2.2 Limitations of existing solutions

Existing solutions usually rely on the HPC provider making a contract with the aircraft manufacturer. As, like mentioned before, the provider has had no experience with a job like that before, he/she has to undertake a complexity analysis of the task. The authors of this paper are aware of the fact that these complexity analysis will be hard to automate and therefore only sketch an idea of "semi"-automation of this analysis by using a database that stores already calculated complexities, but has to calculate "unknown" tasks though. Based on this complexity information, an administrator has to derive the resource requirements for this task – taking into consideration not only complexity, but also current system parameters like size and status of the job queue, etc.

Having the complexity received from an *Analyst*, the provider's *Administrator* usually can (manually) figure out what requirements this job has with regard to his resources in order to be successfully fulfilled. In our example of redesigning the aeroplane's chairs, this job should be executed within three days, and the assumed complexity is $O(n \log n)$. The provider would use his knowledge about jobs with this complexity and calculate the requirement of having say 22 nodes available per day. 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. Having an HPC provider, a BLO could for instance be "*maximise workload on this machine*". BLOs are usually set by a liable person from the respective business party, in our case the *Manager* of the service provider. This implies additional effort to gain influence from the Manager with respect to the BLOs when negotiating a contract. Furthermore the Administrator needs to gain knowledge about the current system status, the upcoming workload and potential conditions from side of the aircraft manufacturer. With this information one could e.g. calculate if the "*22 nodes free a day*" requirement (see above) can be really fulfilled or if one can predict a bottleneck within the valid time period for this job based on system information like the current state of job queue. Aligned to this worked-out facts the Administrator can configure the resource.

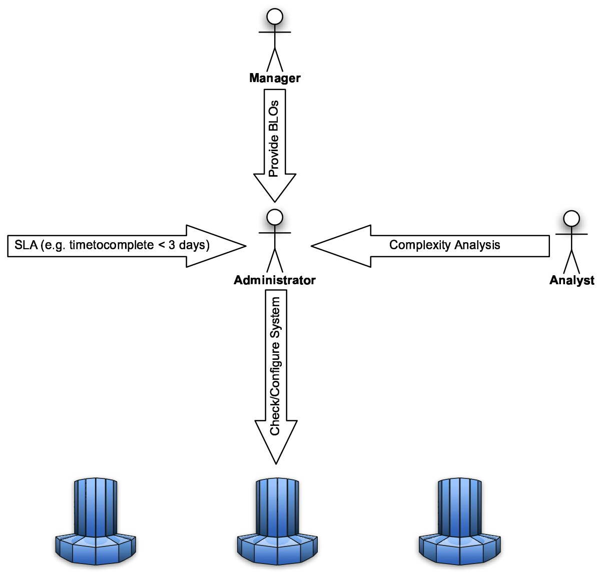

Fig. 1. A human-centric approach to resource configuration at service providers

Current solutions require involvement of different parties to collect the necessary information to fulfil the job according to the requirements of a customer.

Fig. 1 pictures the different parties and tasks they carry out as described in this section. It can be seen that these solutions lack automatic mechanisms to carry out the previously outlined tasks.

3 Dependencies of Resource Configuration on SLAs

Configuring his/her resources in a sensible and efficient way, a service provider has to take into account different facts. With the introduction of Service Level Agreements one set of aspects is defined which has to be respected when configuring the system. But SLAs are only one part of the whole picture. The configuration of a system depends on more aspects: the Business Level Objectives of the service provider, the complexity of the job and the current system status. So in how far do SLAs influence the resource configuration of the system?

The introduction of Service Level Agreements leads to the need of taking into consideration that the agreed upon terms within SLAs usually cannot be used 1:1 for configuring the respective resource. A Service Level Agreement can consist of a set of abstract terms which, unfortunately, mean something different to individual providers. E.g. the term "performance" is defined different by different parties and therefore also calculated and provided in different ways, depending on the respective infrastructure. By going from the 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 calculation is inevitable as the service provider has to understand what he needs to fulfil the conditions of the SLA in terms of e.g. processing power, processing time, etc.

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

The main purpose of SLAs is to define certain QoS parameters in a way that such a service level can be maintained during its interaction with a specific customer. SLAs support therewith automatic business enactment and observation of the respective QoS parameters, i.e. monitoring the performance of a service provider and comparing it with previously agreed terms (evaluation). In addition SLAs have a great impact on the configuration of the system, in particular if service provision is to be dynamic and on-demand. Running projects like TrustCoM and NextGRID examine how Service Level Agreements can be created and used in a way that is sufficient for both customers and service providers. In these projects the SLA usage is restricted to pre-defined SLAs with respect to negotiation. This simplifies selection and creation of SLAs and enables the usage of databases to store matching configuration information. However, studies of real business use cases have pointed out that pre-defined configurations can only be used in a minor set of cases, 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.

Therefore we are aware that future approaches to automated resource management will have to concentrate in some parts on the "translation" of Service Level Agreements to system-understandable data. Furthermore these "translation services" have to ensure that the Service Level Agreements, when used for finding an agreement with the customer, are (on the service provider's side) observing the Business Level Objectives as they represent the service provider's business needs and goals.

Business Level Objectives are abstract goal descriptions and should therefore be used as guidelines for configuration, defined by the service provider before business takes place. Unfortunately, objectives like "*utilize my resources a hundred percent*" or "*get the maximum profit, while spending as little money as possible*", are at the moment of the creation of this paper not supported. However, we know of and are involved in research activities (in NextGRID, TrustCoM and the upcoming project BREIN [2]) which try to solve this issue. For the purposes of this paper, we assume the BLOs being 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, i.e. if and how the service provider can execute the requested jobs in time. Concluding we state that an environment where the SLA-specific configuration depends on the "base"-configuration realised by Business Level Objectives and the complexity analysis of a job represents a valuable approach towards SLA-supported resource management.

4 Mapping Service Level to Operational Level Agreements

The outlined scenario has high demands with respect to the HPC provider's capability of configuring his/her resources on-the-fly – as discussed in the preceding sections, the configuration is also dependent upon the Service Level Agreements defining the Quality of Service to be maintained. Related to our previous discussion, we will present in this section our architectural approach for 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 technical level terms as SLA content (e.g. "timetocomplete" 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. Operating under that condition it is inevitable to map those high level terms down to the technical layer, usable for calculating and defining the requirements to 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,

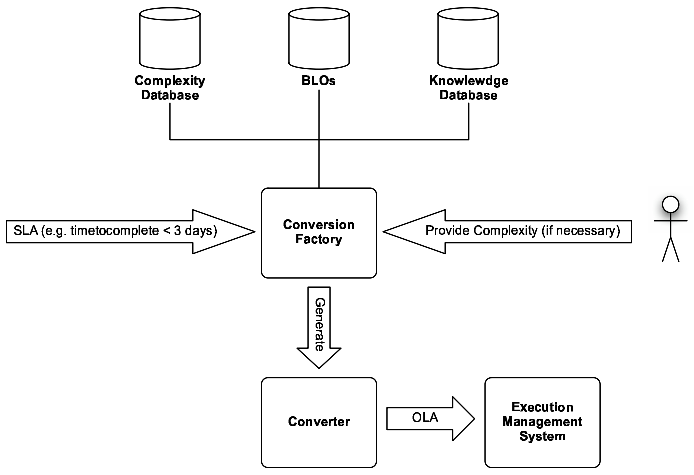
Complexity Database
BLOs
Knowlewdge Database
SLA (e.g. timetocomplete < 3 days)
Conversion Factory
Provide Complexity (if necessary)
Generate
Converter
OLA
Execution Management System

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

– the *Complexity* of the job,
– the *Knowledge* of former job executions (stating something like: "*for this job, I need the following system resources*"), and
– the Service Level Agreement itself.

The resulting set of OLAs will be the basis of the system configuration via the *Execution Management System* (EMS). The EMS is actually not part of our proposed architecture but the entity responsible to consume and interpret 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. We foresee the process of creating the Converter instance as being influenced by the Business Level Objectives from the service provider, the Complexity and the infrastructure Knowledge. Business Level Objectives are the representation of the goals of the service provider. In an ideal world, the service provider's main goal would be to satisfy the Customer's needs as best as possible. But in the business world, both Customers as well as service providers have some internal goals in mind, when starting a business relationship. To pursue these goals, but also to achieve a successful business relation, the service provider will have to take into account his respective BLOs while negotiating and executing a business transaction. Such a BLO could e.g. be "*maximise the workload on the machines*" or "*give higher priority to Customer X*".

Since we focus on the SLA aspects, the Converter is required for (a) checking availability, (b) configuring the system and (c) monitoring the status. Thus a

[1] The function of the EMS is analogue to that of the OGSA Execution Management Services which "are concerned with problems of instantiating and managing, to completion, units of work" [3].

conversion unit needs to perform the principally same task multiple times, in order to support the business transaction in best order. In particular monitoring is relationship dependent (as it is different for each individual SLA).

Therefore we highly recommend the introduction of a "Conversion Factory" that may instantiate individual conversion units, respecting the particular needs for a specific business relationship. For this purpose the Conversion Factory implements interfaces necessary to query the different databases to retrieve information about potential pre-calculated complexities or former business transactions. In particular we see the need for a Knowledge Database (storing information about the system requirements for the different complexities, depending on experiences made by running similar jobs in the past), a Complexity Database (storing previous calculated complexities of jobs, e.g. we can query it for the aircraft manufacturer job to retrieve, if this job has been analysed before, the complexity of $O(n \log n)$) and a BLO Database (storing the Business Level Objectives of the service provider, like "*maximise profit means a workload of 90 % on a special machine*"). It is obvious that we cannot expect the Complexity Database to be filled with all possible job-complexity pairs like "for Job A we need algorithms/workflows with complexity $O(n)$", "for Job B we need we need algorithms with $O(n \log n)$", etc. This implies the need for providing mechanisms for complexity calculation of so far "unknown" jobs. Generally, it is almost impossible to predict the behaviour (and hence complexity) of an unknown algorithm. Potential workarounds consist in letting the job run with a reduced set of requirements to estimate the time requirements (in relationship to the system's capabilities) - however, this neglects either the relationship between data complexity and time requirements, or between time and computational effort. As such, human intervention will most likely be required in these cases which, however, do not diminish the improvements as suggested by our conversion unit.

Whilst the Knowledge Database and the Complexity Database are exploited to reduce the mapping complexity and therefore speed up this process, it is the purpose of the Database of BLOs to provide definitions how to reflect the priorities of the service provider over the job requirements. As mentioned before, the BLOs represent what policies need to be taken into consideration when generating the configuration information. Such priorities could provide a direct influence on this process, e.g. with statements like "*jobs from companies X,Y,Z get higher/lower priority*", etc.

Please note that, as opposed to the Knowledge and Complexity Databases, which are (in an ideal situation) automatically populated with information, the service provider wants to retain control of the definition of his/her own BLOs. Therefore we do not consider automation of BLO definition sensible.

The Converter therefore is created based on the SLA, the Complexity and the infrastructure knowledge and 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 the monitored data from the EMS converted to SLA level terms to check the status of the executed service.

3. **Configure**. Once a SLA is negotiated, this interface will be used to convert the SLA specific parameters into valid configuration information usable as input to the EMS basing on the pre-arranged conversion mechanisms.

Once instantiated, the Converter is available until either the negotiation process fails or the execution of a job is finished.

4.2 The mapping process

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

– **Complexity analysis of the job**. When a SLA (respectively an "offer") is sent to the Conversion Factory, the complexity of the job to be fulfilled has to be retrieved. This can either be done by querying the complexity database or a (human) analyst.
– **Knowledge Database lookup**. Having the complexity analysed (e.g. in our Scenario the complexity of the example job was $O(n \log n)$. The knowledge database represents the information about the infrastructure and the available resources: which resources can provide what and how. Furthermore it provides information about previous configurations of the system with respect to different job-complexities.
– **Application of BLOs to the SLA (Template)**. As the authors approach concentrates on the service provider domain, the customer is only implicitly of relevance (only through the SLA – which to maintain will be also in the service provider's interest). Therefore we foresee the process of creating the Converter Instance as being influenced by the Business Level Objectives from the service provider. Business Level Objectives are the representation of the goals of the service provider. In an ideal world, the Service Provider's main goal would be to satisfy the customer's needs as best as possible. But in the business world, both customers as well as service providers have some internal goals in mind, when starting a business relationship. To pursue these goals, but also to achieve a successful business relation, the service provider will have to take into account his respective BLOs while negotiating and executing a business transaction. Such a BLO could e.g. be "*maximise the workload on the machines*" or "*give higher priority to customer X*".

5 Usage

The above introduced architecture can be used to support negotiation of SLAs as well as for the 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 potential usage within a business interaction.

1. **Negotiation**. During the negotiation of a Service Level Agreement, the service provider has to figure out if he/she can fulfil a service request according to the SLA. Our architecture supports this by checking the availability of resources according to SLA and infrastructure. The mapping and Converter creation process takes place described in 4.2. Having instantiated the Converter, the Negotiation component of the service provider can now check the status of the job queue and with that the availability of the resources with respect to the SLA, by using the provided interface.
   Coming back to our scenario, the aircraft manufacturer asked for a job like redesign the airplane's chairs within three days. We assume that the complexity database has an entry for such jobs having the complexity of $O(n \log n)$. Querying the Knowledge database would give us back the information that $O(n \log n)$ means the need for 64 free nodes for a day. As "timetocomplete" is set with the value of three days, a possible, evenly distributed workload would consist of 22 nodes per day. At this stage, the BLOs and the status of the job queue have to be taken into consideration. Checking the current job queue, it gets clear that in the next two days 23 nodes will be available and at the third day 30, which leads to an amount of 76 free nodes in the next three days. Based on this information, the Negotiation component can decide to accept this job, if there would not have been enough free nodes, he/she would have had to reject the request or to send a counter-offer.
2. **Configuration**. Having found an agreement, the service provider has to configure his system according to the Service Level Agreement, the BLOs, etc. If the Converter was already instantiated during negotiation, it can be used to map the SLA details to the infrastructure-specific (technical) parameters, which will be sent to the Execution Management System in order to configure the systems. Remember that we had 23 free nodes on each of the first two days and 30 free on the last day. The allocation of the required 64 nodes takes place according to the BLOs of the provider. Imagine the aircraft manufacturer being a "high priority" business partner. This fact would lead to having his/her job run on all of the 23 nodes the first two days, and then on 18 nodes on the last day. In opposite a low priority would be handled different. Assume a high priority job from another Customer has to be fulfilled until the next day. This could lead to an initial reservation of 11 nodes on the first day, 23 on the second and 30 on the third day.
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 and therefore supports easy evaluation of the system and service state.
   In case of high priority jobs, the monitoring could also be used for violation prevention. If two nodes of the 23 assigned (cf. 2.) fail, the EMS could be notified of an upcoming violation. With that warning, the EMS can now readjust the system accordingly, e.g. by raising the number of nodes on the third day up to 20, to ensure a successful fulfilment of the SLA of this customer.

6 Conclusions

In this paper we introduced an architecture for SLA-supported resource management. This architecture reflects the business needs of the service provider and aims at a more automatic and autonomous resource configuration through the usage 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.

This paper describes the architecture and the processes necessary to execute the mapping between high level business and low level technical terms, but the actual mapping and the representation of BLOs and database contents within the proposed architetcure are subject of ongoing and future research. Although our approach calls for a sophisticated mapping the authors are confident that for example semantic techniques like [5] can 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. Online (to come up soon): `http://www.ist-brein.org`, 2006.
3. 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`.
4. J. O. Kephart and D. M. Chess. The Vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.
5. 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.
6. 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`.
7. The NextGRID project. Online: `http://www.nextgrid.org/`, 2006.
8. The TrustCoM project. Online: `http://www.eu-trustcom.com/`, 2006.