Scalable Computing: Practice and Experience
Volume 12, Number 2, pp. 164–177. http://www.scpe.org
ISSN 1895-1767

USING SERVICE LEVEL AGREEMENTS IN A HIGH-PERFORMANCE COMPUTING ENVIRONMENT

ROLAND KÜBERT *AND STEFAN WESNER †

Abstract. The concept of Service Level Agreements (SLAs) has come to attention particularly in conjunction with Grid computing. SLAs allow for a controlled collaboration between partners, that is service providers and their customers. SLAs have originated in the telecommunication industry but have found broad uptake in the Grid computing community, especially regarding the topics of their automated negotiation, general management and legal implications. High-performance computing (HPC) providers, however, have not been taking up SLAs yet, even though they often provide not only supercomputer resources but at the same time access to cluster or grid resources. This might be due to the fact that SLAs are mostly regarded on a per-job basis, which is not consistent with the contractual model that HPC providers currently use. In this work, we analyze how HPC providers can implement SLAs and what benefits this brings, proposing the usage of SLAs on a long-term basis. We further present a software that has been developed to simulate the scheduling of jobs at an HPC provider site using service levels and analyze how different distributions of service levels between the submitted jobs influence machine usage and average waiting times of jobs. Finally, we present how an HPC provider can practically implement SLA-based scheduling.

1. Introduction. For High Performance Computing resources scheduling of jobs is still realized in most cases using simple batch queues. While batch queues like OpenPBS [2], TORQUE [5] or others offer a quite comprehensive set of functionality for placing jobs in appropriate queues and optimizing the load of the cluster systems, also across sites, there is no mapping from business level requirements down to the low-level specifications. Low-level specifications are typical elements of a job description, for example desired number of CPUs, maximum wall- or run-time. The provision of such low-level properties requires a high level of expertise of the user and can only be specified if the target platform is predetermined as different node and CPU architectures require different values. Additionally the number of queues is limited and therefore requirements have to be mapped to a particular queue. While advanced reservation, allowing a predefined start time, can be specified the drawback of potentially significantly reduced efficiency due to fragmentation of the schedule is not mapped to potential business penalties such as dynamically adapted pricing for such requests depending on the concrete loss in efficiency.

If an HPC provider wants to offer its services as utilities and aims to map different possible flavors of the services on different queue structures, the following problems can occur:

- Typically the use of certain queues is mapped to Unix credentials and groups. So all users of a certain group can or cannot use e.g. the express queue. However, depending on time of day or load situations, the "express queue service" might not be available to the same group of users all the time.
- While queues for specialized nodes (for example with graphics processing units (GPUs) or high memory nodes) are underutilized and normal nodes are oversubscribed there is no way to allow clients and providers to agree on special "discounts" for them. An automatic movement from normal to premium node queues would require interaction with accounting services.
- Customers might want to differentiate quite fine-grained about the treatment of their jobs. In such cases nowadays manual movement of jobs within the queues to "prioritize" them might be agreed beyond existing queue structures and group memberships. Such manual interactions cannot scale.
- Not all elements describing the Quality of Service (QoS) or Quality of Experience (QoE) can be mapped on queue properties and parameters. The overall service covers a wider range of properties such as the availability of a certain compute environment, application versions and licenses, proper treatment of data or specific configurations of the cluster system such as "require logical partition to isolate from other users".

This limitation that only a few functional parameters can be specified when submitting a job (also reflected in standards like the Job Service Description Language (JSDL) [1]) means that there is basically no way for the user to express his requirements on a QoS or QoE level. As an HPC provider is offering a service to users that is potentially replaceable with services from other providers, it is necessary to bridge the gaps between complex demands from the user side and simple services that are currently offered in order to ensure continued service usage.

*High Performance Computing Center Stuttgart, University of Stuttgart, Stuttgart, Germany, E-mail: kuebert@hlrs.de
†High Performance Computing Center Stuttgart, University of Stuttgart, Stuttgart, Germany, E-mail: wesner@hlrs.de

As a result a significant amount of work has been spent on realizing SLA frameworks allowing to mutually agree on the terms of the service between provider and consumer. However, while these frameworks cover well the necessary steps to realize SLAs also as a legally binding agreement, the concrete content of such an SLA and more important how these terms can be guaranteed and provided from the service provider side are not adequately addressed.

So far we have the possibility for the consumer to express the requirements and agree the terms with the provider but

- terms within the SLA are not on the desired business level but mimic the low-level properties of the underlying queuing systems and
- the agreement process is typically detached from the underlying infrastructure such as current load situation of different resources, priority and importance of the consumer in a Customer Relationship Management (CRM) system and the accounting and billing services.

Consequently there is a gap between the demand of defining business level SLAs and their implementation using available methods and tools for the management of them on different type of computing facilities ranging from commodity of the shelf (COTS) clusters over specialized compute systems to cloud computing and storage systems.

Management systems on the provider side between the SLAs agreed with the consumer and the concrete physical resources need to interact with a range of different elements within the providers IT infrastructure and must look beyond individual SLAs to optimize the overall operation of all resources within a HPC computing service provider.

This work is structured as follows: section 2 analyzes related work, section 3 examines business models for the provisioning of HPC services, section 4 elaborates on benefits for HPC providers when implementing SLAs and section 5 describes the advantage of long-term SLAs against typically used short-lived, per-job contrast. Section 6 transfers the theoretical work from the previous section to a more practical setting by analyzing the impact of SLAs on resource usage and job waiting time and section 7 addresses the issue of implementing SLA management in an HPC environment. Finally, section 8 gives and outlook on how the previous findings can be transferred from HPC to cloud services and section 9 concludes the work.

2. Related work. There are various approaches to the usage of service level agreements for job scheduling. While they differ in many respects - detail of the presentation, assumed parameters, implementation level, etc. - they all share the fact that they treat SLAs as agreements on a per-job basis. That means that, for each job to be submitted, a unique SLA is established before the job can be submitted; an exception to this is the work of Kübert and Wesner, who propose to use service level agreements as long-term contracts [17]. In [31], Yarmolenko et al., after having identified the fact that SLA-based scheduling is not researched as intensively as it could be, investigate the influence of different heuristics on the scheduling of parallel jobs. SLAs are identified as a means to provide more flexibility, increase resource utilization and to fulfill a larger number of user requests. Parameters either influence timing (earliest job start time, latest job finish time, job execution time, number of CPU nodes) or pricing. They present a theoretical analysis of scheduling heuristics and how they are influenced by SLA parameters and do not investigate how the heuristics might be integrated into an already existing setup. The same authors identify in [23] the need to provide greater flexibility in service levels offered by high-performance, parallel, supercomputing resources. In this work they present an abstract architecture for job scheduling on the grid and come to the conclusion that new algorithms are necessary for efficient scheduling in order to satisfy SLA terms but that little research has been published in this area. MacLaren et al. come to a similar conclusion, stating that SLAs are necessary in an architecture supporting efficient job scheduling [20].

SLAs that express a job's deadline as central parameter for deadline-constrained job admission control have been investigated by Yeo and Buyya [32]. The main findings were that these SLAs depend strongly on accurate run time estimates, but that it is difficult to obtain good run time estimates from job traces.

Djemame et al. present a way of using SLAs for risk assessment and management, thereby increasing the reliability of grids [7]. The proposed solution is discussed in the scope of three use cases: a single-job scenario, a workflow scenario with a broker that is only active at negotiation-time and a workflow scenario with a broker that is responsible at run-time. It is claimed that risk assessment leads to fewer SLA violations, thus increasing profit, and to increased trust into grid technology.

Dumitrescu et al. have explored a specific type of SLAs, usage SLAs, for scheduling of grid-specific workloads using the bio informatics BLAST tool with the GRUBER scheduling framework [8]. Usage SLAs are

characterized by four parameters: a user's VO and group membership, required processor time and required disk space. The work analyzes how suitable different scheduling algorithm are. Additionally, it comes to the conclusion that there is a need for using good grid resource management tools, which should be easy to maintain and to deploy.

Sandholm describes how a grid, specifically the accounting-driven Swedish national grid, can be made aware of SLAs [24]. It is presented how the architecture can be extended with SLAs and it is stated the greatest benefit would be achieved by insisting on formally signed agreements.

A comprehensive overview of resource management systems and the application of SLAs for resource management and scheduling is given by Seidel et al. [26]. The connection of service level and resource management to local schedulers is clearly shown as a gap in nearly all solutions.

In summary, it can be said that isolated aspects of the usage of SLAs have partly been investigated in detail: scheduling algorithms and heuristics, abstract architectures, parameters which are to be used as service levels, SLA negotiation etc. Gaps, however, can be easily identified: the analysis of the "big picture", that is the composition of individual aspects of SLA usage into a complete system and the integration of SLAs and SLA management with local resource management. This is not only true for the "traditional" field of high performance and grid computing but can also be extended to the field of cloud computing. Furthermore, SLAs are solely treated on a per-job basis, the analysis of SLAs as long-term contracts is not covered.

3. HPC Service Provisioning Business Models. The increased importance of modeling and simulation in many academic disciplines and similarly in industry over the last years is one of the drivers leading to an increased diversity of the user community for High Performance Computing service providers. Another element is the hardware development towards multi- and manycore computing systems or GPGPUs making parallelism a commodity at any desktop. As a result the user community of a typical HPC service provider ranges from entry level users that have just started parallel programming up to high-end users able to use very large computing systems with many hundred teraFLOPS up to the petaFLOP scale.

Another dimension for classifying users is their typical workflow in using the HPC resources. Depending on the research discipline the demand for computing resources can be rather constant or may vary quite significantly over time. A user with a constant demand is interested in a service that delivers a guaranteed level of capacity of the computing system and costs that are lower as for a self-operated computing system. A user with a largely varying computing demand e.g. because he is relying on experimental data demands for an elastic computing system. Additionally all users change their use profile over time e.g. during development cycles of their simulation software more short runs are requested whereas during production phases long running jobs are the major use case.

Beside different job types in terms of scale or duration the expected HPC offer is also determined by the use case. Users performing open research & development typically operates based on grants with a duration of several months and years. The condition associated with this kind of access that is typically without costs is to publish all results obtained at the end of the grant period. The conditions of this kind of offer are typically defined by the provider for a large class of users and not negotiated bi-laterally. For example there could be a defined profile for university users, federal project users and large scale research projects detailing the service level, which resources can be accessed and what kind of applications and tools are available. If two different users receive a grant of the same type from the options above, their access conditions are completely identical.

Another offer type is a production level offer with bi-lateral agreed conditions and obligations defined in a legally binding contract and negotiated on a case by case basis. Such an offer is typically applied for commercial users that are willing to accept higher costs for computing services in order to receive special access conditions, user tailored environments (e.g. a dedicated file system or queue). Obviously a wide spectrum of offers in-between these two different extremes are possible such as special agreed conditions for a group of users.

3.1. Provider defined SLAs. Provider defined SLAs or service offers use the available hardware resources as the starting point for the offer definition. The typical model applied by many HPC service providers is to use a reduced amount of production hours due to maintenance or loss in utilization in the scheduling system (for example due to very large computing jobs) per year e.g. 5000-6000 hours instead of the theoretically possible 8760 hours ($365 * 24$). Additional resources such as application licenses or special hardware are considered similarly.

Based on the available resources and preallocated shares of a system (e.g. 30% for European Research projects) for the different system parts and the intended use model of the system an appropriate set of queues

are defined for the system. The queue design allows to control the major purpose of the system as capacity or capability machine, if large scale debugging or scalability test sessions are possible (queues for jobs of large size for short job duration and reasonable waiting time) or if large jobs should be prioritized. Additionally a set of constraints are defined on how certain external servers supporting the use of the big computing system should be used. For example that compilation of software should only be done on login servers, analysis of result data on dedicated post-processing servers.

Based on the definition of the environment that is offered to the users very different models are still possible controlling fair access to the computing system. One model deployed widely is to operate the resource in a competitive manner allowing all users to place their jobs in the different queues and schedule jobs driven by queue and user priority. Such a model has potentially the problem of overbooking and undetermined waiting times in queues but realize a quite high level of utilization. The quality of the service experienced by the users will vary and depends on the arrival process of competing jobs in the queues by other users.

Alternative approaches are to offer dedicated access to a share of the machine for a certain user group or project for a a given time period (several months or even a year) with more certainty in terms of waiting time or predictability of job launch but with significantly reduced utilization of the overall system.

The major characteristic of provider defined SLAs is the resource driven viewpoint aiming to define a very small set of offers for different user groups. Furthermore it is assumed that user demands are (1) well understood by the HPC provider, (2) are defined by the community or project the user belongs to and (3) the demand is not depending on the project phase or type of work currently done. These kind of SLAs are not negotiated but are policies or use models that need to be signed up by the users during the application procedure for accessing the resources. These conditions apply for all compute jobs submitted during the whole project lifetime.

3.2. User negotiated SLAs. User or user group negotiated SLAs start from the particular demand of the research or commercial project. These demands expressed on the user domain level reflecting the workflow and processes applied by the users need to be mapped on the concretely available resources at the provider side. As such user tailored SLAs are as of now implemented by manually changing the configuration of the HPC provider environment such offers are limited to a small set of customers.

If part of such a user negotiated SLA are specific guarantees or dedicated access to a decent fraction of the system the utilization of the system is reduced and is in contradiction to the major goal of the HPC service provider to achieve the highest possible level of utilization. Consequently such special offers must be associated with an appropriately increased price reflecting the loss in efficiency. As an example consider the demand to start compute jobs spanning across 50% of the overall system with a guaranteed start time of less than one hour. This means that upon submission of such a job under this SLA all jobs running longer than one hour blocking the part of the system to be used must be terminated and re-started after the large job. The already used computing time until the last checkpoint is lost and needs to be added to the costs of the large job.

As the loss of such a model can be quite significant and if too many user negotiated access offers have been agreed the implementation might be impossible (or the risk of violating the SLA would become too high) such offers are typically limited to a small fraction of the system in the range of 10-20%.

4. Benefits of SLAs for HPC service provisioning. The current operation model for high-end computing resources is conceptually still the same as fifty years ago where users placed a set of punching cards at the registry desk. The only difference is that users now can submit their compute jobs to a set of different queues and instead of the human operator the scheduling system is picking the jobs from the different queues depending on defined policies aiming for an optimized load of the system partially reaching 99% utilization. The major shortcoming of this approach is that the optimization strategy defined by the queues and the scheduling system policies is oriented towards a global optimization rather than an individual service offer.

If a user needs a special service (e.g. guaranteed start time of a job during a demonstration, interactive visualization or exhibition) beyond regular job submission the negotiation is typically done directly with the system operator and the performed steps are mostly done manually.

The availability of multi-core CPUs will lead to compute nodes with 32 cores and more in the near future, the rise of GPU-based computing with several hundred “cores” per card allows a reasonable number of applications to run on a single node. This is particularly true if the application is not targeting for a high-end simulation e.g. in the area of Computational Fluid Dynamics (CFD) domain with a very fine-grained mesh but more on exploring the problem space. Other examples are cases where the full simulation has been done before and now only small changes in geometry are done interactively demanding much less intensive computing to reach

a stable state again as it is based on the previously achieved results.

Driven by the availability of cloud service providers and emerging products such as the Amazon Cluster Compute Instances also high-end computing service providers change their offers to be more *elastic* and realize a more *dynamically changing* infrastructure having certain queues available only during specific time periods or realizing a dynamic allocation of resources to logical partitions depending on the load situations or specific time bound agreements.

The predominant use of high-end computing services will continue to be highly scalable technical simulations demanding a large number of compute nodes for exclusive use. However additional use cases have emerged driven by changes on the hardware level and competition with cloud service providers in particular for small scale simulations. The exclusive access for a user to one single node might even for compute intensive applications become a relic of the past. This substantially more complex management model for HPC service providers that cannot rely anymore on a quite homogeneous user behavior and long running jobs demands for a more complex management solution for operating their resources. The challenge is to integrate the demands of policies from different levels such as business policies (e.g. users with highly scalable and long running jobs should experience a preferred treatment) with more short term policies reacting on the current load situation (e.g. reducing prices or accepting more small jobs to fill gaps in the current schedule) and the demand of the users on a per-job basis.

The following sections aim to cover in examples the three major use cases driving the need for an SLA-guaranteed HPC service provision. Abstracting from concrete cases three different cases can be identified:

4.1. Interactive Validation. In many areas simulations have already replaced real experiments or physical prototypes during the development process. However at certain control points in the process simulation results have to be verified using physical prototypes. Within the IRMOS project augmented reality techniques are used to overlay real experimental data like a smoke train in the wind channel with a visualization of trace lines from the corresponding simulation. This "hybrid" prototype allows experts to directly compare the behavior of the real prototype with the results of the simulation. Such a design review session typically involving several people of a development team spread around the globe demands a fixed availability of the wind-tunnel, the computing resources, the visualization resources, the corresponding network resources and all involved experts, for example via video-conferencing.

In such a scenario simulation data will be generated continuously by a simulation running on a compute resource that is directly connected to the visualization resources. The current configuration of the wind channel like the air speed will be communicated as boundary condition for the simulation, thus the same parameters for both will be used while the experiment is running. This requires a coordinated and automated provision of the resources involved in the overall setting.

Such a scenario cannot rely on batch queue-based access as the computing and simulation part is just one piece in the overall setting. The demand for a co-ordinated availability also opens questions on how penalties are applied if one of the pieces in the overall setting is failing. For example if the compute resources are not provided as promised in time and the wind tunnel cannot be used the costs for it still accumulate. This applies also the other way around if the wind tunnel is not available or fails to communicate the boundary conditions for the simulations or the network connection is not delivering sufficient bandwidth.

As the resources needed for the full scenario are provided by different organizational entities the different quality levels needed by each individual contributor need to be put in a formal SLA, covering the terms of service as well as the agreed penalties in case of failures.

4.2. Guaranteed Environment. As outlined in [30] beside quality constraints there is also a demand to ensure a certain environment or other procedural constraints such as data handling, security policies or environmental properties (version of the operating system, available Independent Software Vendor (ISV) applications, etc.).

This is especially necessary for simulations performed as part of an overall design cycle for a complex product such as a car or airplane. A software environment is frozen for a full development cycle in order to ensure reproducible simulation results. This fixed environment is typically ranging from operating system over certain versions of numerical libraries up to application codes. A typical approach to address this requirement is to have beside a paper-based SLA agreed for a design cycle period a dedicated computing resource with the requested environment.

Advances in virtualization technologies as well as the possibility to apply different boot images in diskless cluster environments allow a more flexible treatment. Using such technologies a potentially unlimited number

of predefined images, or even user-defined images, might be provided. As not all environments can be provided on all compute resources there must be a negotiation process between the user and the provider where a certain environment is demanded (e.g. expressed in a certain SLA bundle such as "Silver") and a corresponding reply about the conditions for the different options from the provider side is delivered.

The increased flexibility would allow to offer customized environments not only to large customers asking for resources for a long time period but also for users looking to meet their peak demands with outsourcing avoiding tedious customization activities of the environment reducing the entry gap.

4.3. Real-time Constraint Simulations. With the increasing role of simulations in design processes for complex products the demand to have a time-boxed simulation where results need to be delivered in time have emerged. This might be a set of simulations exploring a parameter space as input for a meeting of engineers the other day deciding on the focus for the future (long running simulation jobs). Another possibility is if the results of one single simulation (or a set of simultaneously running simulations) is the input to support an expert in taking a decision.

One important application area demanding for such an operation model is individualized patient treatment. For example in [25] a scenario for using simulations to validate different options to perform a bone implant for a specific patients is presented. In such cases the expert that needs to make a treatment decision has to ensure in advance of starting the simulation at a specific compute service provider that the results will be available in time before the treatment must be executed.

In such a scenario a negotiation with several providers would be started in parallel in order to make a case-by-case decision to which provider the job will be finally submitted. Such a loose binding to a specific provider would also require similarly to the scenario in the previous section a guaranteed or user-provided environment making the different providers interchangeable.

4.4. SLA Service Provision Benefits. From all the scenarios above it becomes clear that a much higher diversity of the offered services must be expected in the future. The requirements of the different scenarios on the provider's infrastructure are quite diverging. Additionally the consumer requirements are contradictory to the goal of the providers reaching a very high level of utilization of the provided resources.

As a result service providers will need to

- offer a mix of different services in order to combine the benefit of best-effort services (high utilization) with the benefit of special services (high value and price),
- offer a framework allowing consumers and providers to agree on the specific conditions for the service and
- actively manage their resources in a way that agreed SLAs are met, resources are most effectively used and any failures and incidents on the resource level are managed to avoid any impact on the agreed service levels.

The underpinning assumption presented in this section is that the provision of SLA controlled services is beneficial for consumers *and* providers. Consumer can negotiate guarantees and specific properties of the provided services as needed enabling new use models for high-end computing resources as outlined above. The provider perspective is clearly driven by business benefits to deliver as a part of the differentiation strategy specific products rather than aiming for a cost leadership approach. Consequently there is a clear need from the consumer side as well as a clear motivation from the provider side to deliver also in the HPC domain SLA based services. In other words the current model where the user needs to fully adapt to the provided environment and access model is changed to a model where the provider is offering certain possibilities or a kind of toolbox where the consumer can arrange the service offer according to their needs. Realistically this space of options needs to be discrete and limited allowing a management of the service offer from the provider side.

5. Using long-term service level agreements for job control. Service level agreements, when they are used for the scheduling of compute jobs, are normally assumed to be on a per-job basis. That means that an individual SLA only contains terms for one specific job and a new SLA needs to be established for each job (see for example [7], [31] and [32]). This may be ideal to investigate the influence of SLAs and parameters specified therein on the scheduling of jobs in an isolated environment but does not correspond with the reality of how contracts are handled at HPC providers. At HPC providers, users usually agree to a contract that specifies charges for computational times and storage for available machines [22]. Jobs are then submitted in accordance with the acknowledged charges which therefore can be thought of as a long-term contract. This contract is, however, missing a specification of service levels. There may be some service level-related parameters specified

- for example the availability of different machines to users and their characteristics, for example CPUs per compute node and memory size per node, but these are only specified in order to compute the amount finally billed to the user. By adding service levels to this contract, a long-term service level agreement is formed.

Long-term SLAs add the missing specification of service levels but keep the familiar contract behavior used by HPC providers intact. A simple specification of priorities, for example, might be realized through the following service levels:

Bronze Computational time is cheap, but there is no assurance on the scheduling of a job. This corresponds to the best-effort services provided today at HPC centers.

Silver Moderate prizing for computing time due to prioritized scheduling. Silver jobs can have timing guarantees and might preempt best-effort jobs. The increased prize is justified since guarantees on the job's scheduling are given.

Gold High-prized jobs that are only rarely used, for example for urgent computing when computations need to be started immediately.

In contrast to the current situation the possibility of providing different service levels allows users to potentially have multiple contracts in place in parallel. On job submission time, a user decides which contract to reference in the submission depending on the current requirements and conditions such as urgency of the simulation result, load situation of the provider(s) etc. This can be seen as a using the middle way between using SLAs on dynamic, per-job basis and solely having singular long-term contracts. As opposed to dynamic, per-job SLAs, this approach reduces the amount of negotiation as only few contracts are in place. Additionally, it avoids the problem that an urgent job cannot be submitted due to a failed negotiation. This approach is more flexible than having only singular contracts and allows users to choose necessary priorities depending on the prize they like to pay.

6. Simulating the scheduling of service level agreements. In the previous sections, we have elaborated on the benefits of the usage of SLAs for both the service provider, who can offer a much more diverse portfolio, and the customer, who can make use of this portfolio and will have certain guarantees assured from the service provider. As contracts that are established for typical HPC providers at the moment are of a long-term nature, it is suitable for the SLA-related contractual information to be of long-term nature as well. This immediately poses a straightforward questions: how can service providers know how the offering of service levels will work out? To put it another way, service providers, who are obliging themselves to various guarantees and may even be penalized if these are not met can not just switch from a best-effort scheduling approach to an implementation using service levels and hope that everything will "just work". Even though the provider is free to formulate service levels and to enter these with customers as it sees fit, the results are not foreseeable.

The provision of service levels in high performance computing is a novel approach, therefore the provider cannot rely on already existing data. Even though data exists that contains real-world workload traces and even models of these exist[1], these traces and models do not include service level data, so another way has to be found. This way needs to provide answers to different questions, for example:

- What distributions of service levels lead to a good quota of fulfillment, or, what service levels can be supported without the provider being liable to paying huge penalties?
- What different target functions can be applied and what does each one entail?
- How do different scheduling policies influence the fulfillment of contracts?
- What is the result on the provider's infrastructure, for example how much is a cluster used?

The right way to answer these questions in a way that does not put the provider in an experimental situation without its outcomes - possibly huge penalties - being foreseeable is to simulate the different service levels the provider envisions. The arrival, scheduling and computation of jobs at an HPC provider's site can be easily simulated with a discrete-event simulation, which represents a system as a chronological series of events.

Jobs arrive at a certain, discrete point in time and are scheduled. Time is advanced and the jobs are finally run on certain resources until they are finished, at which point in time they leave the system. All this can be modeled easily with different events, for example job arrival, computation start and computation end. Many tools for generic discrete-event simulations exist, for example SimJava [11], Tortuga [13] or SimPy [27], as well as tools for specialized applications, for example the GridSim Toolkit for the simulation of scheduling for parallel and distributed computing [28].

[1] The Parallel Workloads Archive at `http://www.cs.huji.ac.il/labs/parallel/workload/` is a good source for both of these.

The addition of service levels into these simulations doesn't change the inherent nature of the problem - a discrete-event simulation can still be used to simulate scheduling taking service levels into account. There is, however no readily extensible tool that can be used for the simulation of different, possibly new scheduling algorithms, especially not in conjunction with service levels. In order to not duplicate work that has already been solved in satisfactory way - however, a very common feature [14] - a simulation program or toolkit that is close to the required functionality should be used and extended. Alea [15] is a job scheduling simulator that is based on the popular GridSim toolkit; its basic functionalities are fulfilling the requirements, but Alea is not extensible, can generally not be readily used and does not provide any support for service levels. As an extension of the GridSim toolkit [28] , Alea's own code base is, however, more easily manageable. This led the authors to the decision to analyze Alea and extend it with the ability to simulate service levels. This was supported by the fact that Alea already supports different scheduling algorithms - as previously mentioned, unfortunately not related to service levels - can read different workload traces (for example from the above mentioned Parallel Workloads Archive) and even provides a simple visualization of results.

Alea works by reading different data files for workload traces, hardware resource description and machine failures. While the specification of failure data is optional, workload traces and resource descriptions are of course mandatory. Alea is implemented in an object-oriented approach and each entity read from a data file is a corresponding class instance. The communication between these classes is performed by a central scheduler component which is divided into a communication and a scheduling part. In order to enrich jobs specified in workload traces, an additional data file that specifies the service level for each job has been developed. This data file is read by a custom loader class, similar to the ones for workload traces and resource descriptions and the service level information is added to the job object. Figure 6.1 shows Alea's architecture with added components and communications represented with dashed lines.

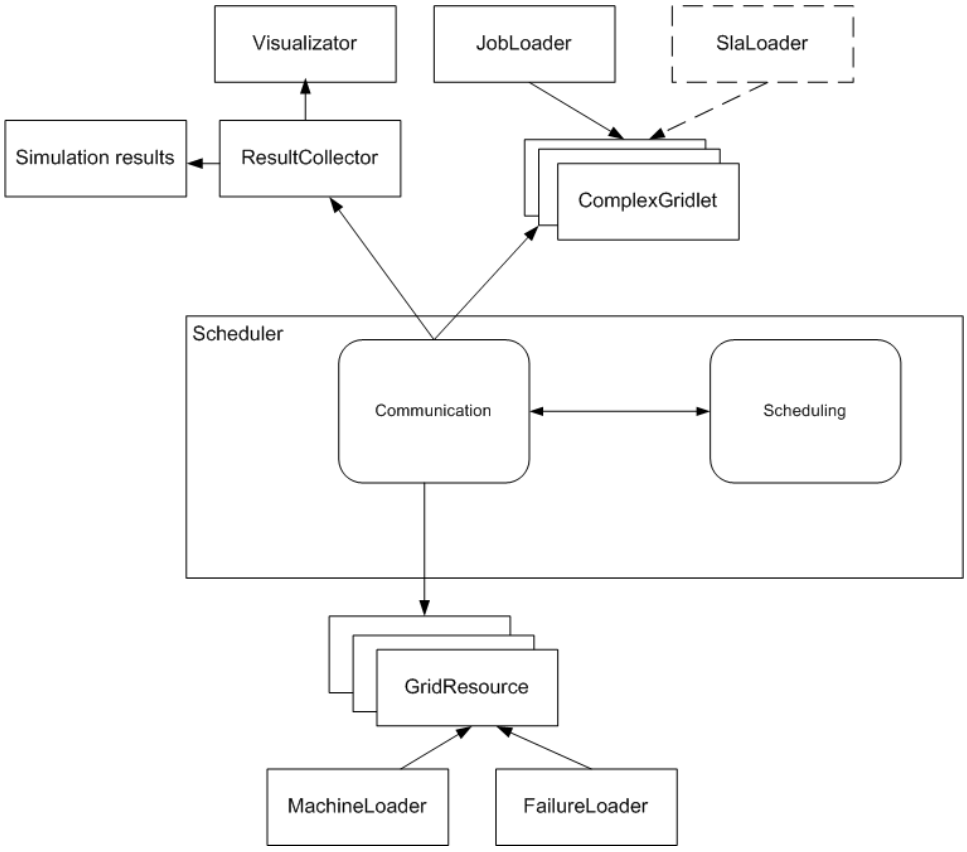

Fig. 6.1: The Alea architecture with the added SLA loader [15]

The workload formats that Alea uses are more or less standardized, so the decision was taken to not

change the format. Instead, a special file mapping jobs to service levels has been created. This file is only read if service level-scheduling is used, other algorithms which Alea supports are not touched. The scheduling supporting service levels is currently using a queue-based approach (Alea supports schedule-based approaches as well). Each newly arriving job is sorted into the queue according to its priority using a special *Comparator*[2].

An initial implementation of the work has already brought tangible results that can guide service providers to rough estimations on what distribution of service levels can be offered [16]. The initial simulation was performed using three different service levels - gold, silver and bronze which corresponded to urgent computing, prioritized scheduling and best-effort and it was investigated in how far different distributions of these service levels influence the average waiting time of jobs - in each service level and in total - and the machine usage.

Due to the workload distribution itself, the machine usage did not change. This is, however, not universal, as it is simple to construct example cases where the machine usage is changed. This is especially true if additional service parameters apart from pure prioritization are used, for example exclusive access to resources. Scheduling a job for exclusive use of a resource will then reduce the total machine usage if the job running exclusively will not use as many resources as would be consumed at the same time using non-exclusive access. The machine usage is most likely not the foremost factor to be considered when offering service levels, as it is of more importance to satisfy the service levels and only then can other parameters be taken into consideration.

An interesting question is: given a possible distribution of service levels, could the provider give any absolute guarantees ("X% of jobs submitted in the silver service level will wait less than y minutes") or is this impossible and the only guarantees possible are relative ("gold service level jobs are scheduled before silver service level jobs, which are scheduled before bronze service level jobs")? In order to answer this question, we have investigated different distributions of the three service levels and have looked at how each distribution changes the average waiting time of jobs and, as well, the average waiting times of jobs in each service level. The base case, best-effort scheduling, can be seen as 100% of jobs in the bronze service level.

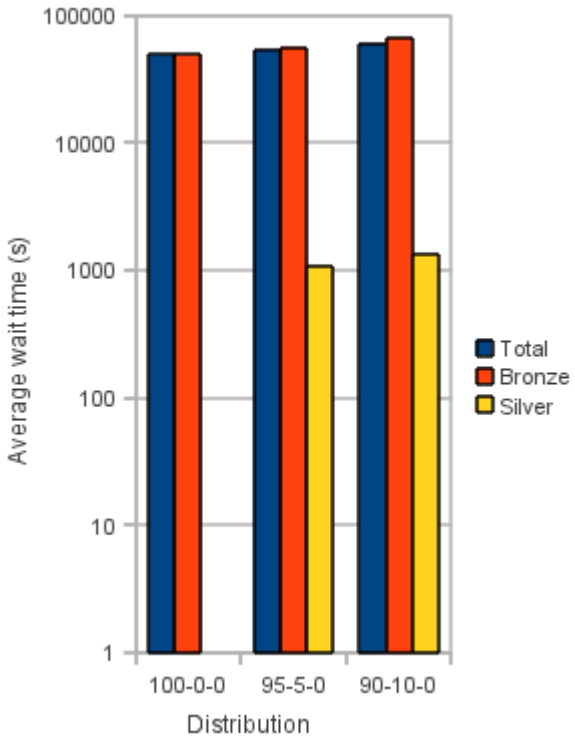

Fig. 6.2: Distribution of waiting times with only silver and bronze service levels

[2]See `http://download.oracle.com/javase/6/docs/api/java/util/Comparator.html`.

Figure 6.2 shows the base case on the left and two additional distributions of only the silver service level, first with 95% of jobs in the bronze service level and 5% in the silver service level (middle), then with 90% of jobs in the bronze service level and 10% in the silver service level (right). The average waiting time increases with rising number of silver jobs, of course, as bronze jobs are potentially queued longer; the same is true for the bronze level itself, but with a higher increase in the average waiting time – silver jobs have a short waiting time and therefore reduce the average case. Between 5% and 10% there is only very small increase in waiting time. The provider can therefore give guarantees that with a high probability will not be breached; he would, however, have to limit the amount of silver jobs a single user can have in the system as otherwise it is trivial to force the provider into a situation where it breaches SLAs, for example by submitting a high number of silver jobs. Even though the increase in waiting time is very small for the silver service level, the provider will not want to increase the number of silver level jobs too much, as the increase in waiting time for the best-effort bronze level will then reach a point which makes the provider unattractive to customers only using the bronze service level.

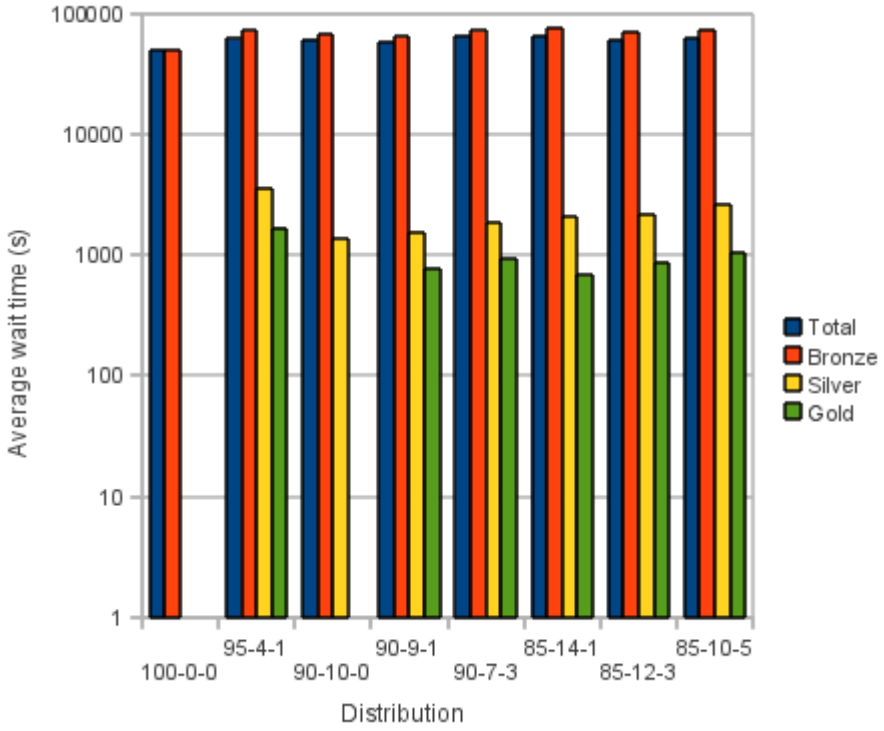

Fig. 6.3: Distribution of waiting times with gold, silver and bronze service levels

Figure 6.3 shows different distributions of service levels, taking all service levels (gold, silver and bronze) into account. The trend that can be seen here is similar to the one shown before: the average waiting increases moderately while the waiting time for the bronze service levels has a much bigger increase. Additionally, for a constant percentage of bronze jobs, the waiting time for both silver and gold jobs increases with increasing number of gold jobs. Once again the results hint at guarantees the provider might give: the average waiting time for gold jobs rises only twice above 1,000s, the average waiting time for silver jobs only twice about 2,500s.

As these results are simulated with fixed workload traces, a provider cannot be 100% sure these findings are transferable directly to an implementation of service-level based scheduling. Taking more complex considerations into account can, however, align the simulation closer to reality. Findings after an implementation - for example the number of jobs that users who have the possibility to submit to different service levels submit to each service level - can help refine the simulation with user behavior. Nonetheless, the simulation shows that a provider can at least simulate rough impacts of different distributions of service levels.

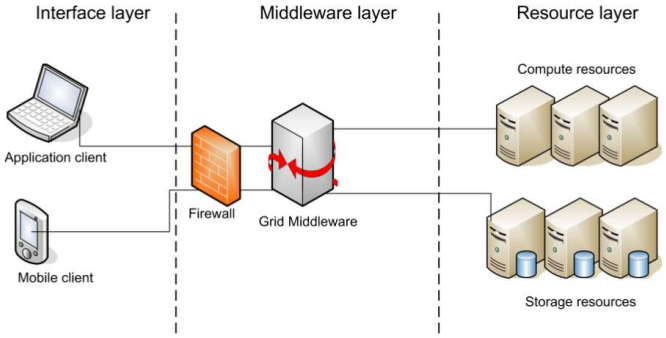

Fig. 7.1: Layered architecture

7. An integrated approach to service level management. The basic service level approach given above can be realized with current techniques, for example the usage of specialized priority queues; however, providing more complex service levels cannot be realized that easily but require the integration of service level management techniques across interface, middleware and resource layer.

Figure 7.1 shows the typical three-layered setup that is used by HPC providers. The left-hand sides shows clients of the HPC provider, either static or mobile[3]. The middleware layer is positioned between the client and the low-level resources and serves as a central entry point to the HPC provider's system and provides access through grid middlewares, for example the Globus Toolkit. The Grid middleware takes jobs submitted by the client and passes them on to low-level resources by means of a resource manager which employs a job scheduler in order to determine which jobs are placed on which resources.

7.1. Service level selection by the client. Enhancing the client with the ability to select service levels is very straightforward. This can be either done by changing the job submission client, adding SLA information to the message sent to the middleware or by integrating SLA functionalities into the application, if job submission is performed directly out of it.

7.2. Enhancing the middleware. SLA management on a middleware level has been investigated by various research projects and therefore different components and solutions already exist; the NextGRID project [21] aimed at providing a generic solution in the broader realm of a full architecture for next-generation grids while other projects focused on solutions for specific problems - the FinGrid project [9] on the financial industry, the IRMOS project [12] on the applicability of SLAs to the execution of real-time aware applications on Service-Oriented Architectures (SOAs) and the BEinGRID project [4] on a solution that can be adapted to different business cases. As these solutions often have the drawback of being very complex, a simpler solution is preferable, as it eases the amount of work necessary for installation, integration and maintenance.

Figure 7.2 is a diagram depicting how easily SLA management can be implemented on a middleware level and the underlying resource layer. The client thereby can either communicate with the SLA Manager, a central component on the HPC provider side responsible for SLA management, or submit jobs to the cluster front-end in the manner already explained.

The SLA Manager provides data regarding the long-term SLA contracts, for example contract information, accounting pertaining to contracts etc. It uses an internal SLA Repository for storing the contracts and other relevant information and is the central point that is queried by other components regarding SLAs. The cluster front-end, for example, on submission of a job, can query the SLA Manager for the validity of SLAs and can, after job completion, send accounting data to the SLA Manager.

The SLA functionality for the cluster front-end in the grid middleware can be realized in a non-intrusive way, for example through a policy decision point (PDP) that checks incoming requests and their SLA specification for validity. Incoming requests that do not contain SLA specifications can be mapped internally to a default SLA

[3]Mobile in this context should not be mixed with cellular phones and is understood as a nomadic user that is connecting from different locations without predefined IP addresses.

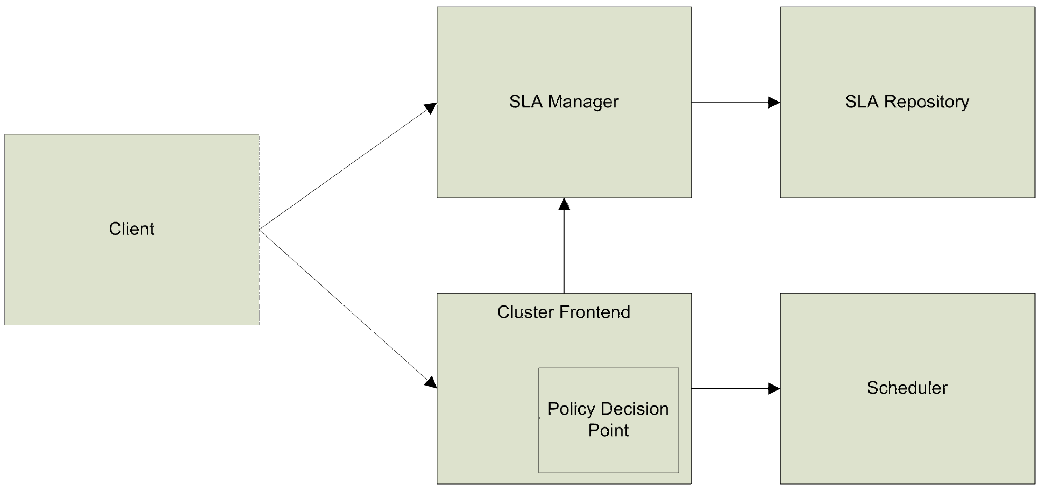

Fig. 7.2: High-level SLA management components

specifying a best-effort style service level, thereby realizing complete SLA-functionality and being backwards-compatible to clients.

7.3. Acknowledgement of SLAs. Honoring service levels of submitted jobs depends on the software used on this low level. Very simple schedulers, like the default scheduler supplied with the TORQUE resource manager, cannot honor service levels and need to be replaced by an SLA-enabled scheduler. This can be implemented by the provider itself, but this is a time-consuming and error-prone job. Rather, an SLA-enabled scheduler, for example the Moab Cluster Suite, should be used. It allows the formulation of quality of service levels for resource access, priority and accounting.

The providers main task is then to express the high-level SLAs offered to customers in such a way that the scheduler can implement them on the resource layer. Additionally, the incoming job requests have to be mapped to the corresponding service levels.

8. From high-performance to cloud computing. In the previous sections we have elaborated a concept to enhance "classical" high-performance computing with service levels through the use of long-term service level agreements. Cloud computing, in many terms similar to the previous scenarios, seems like a logical step for the provisioning of services and can be a sensible offer to provide for HPC providers besides their usual role. Even though the term cloud computing is not clearly defined, it can be seen as distributed computing with virtual machines. Virtualization allows for more flexibility, scalability and abstraction of the underlying resources. Accounting and billing are usage-dependent [3].

Cloud computing brings benefits both for consumers and providers. Virtualization allows the provider to use free resources for the execution for virtual machines as the underlying hardware is mostly irrelevant, although requirements specified by the user of course still need to be met. The usage of virtual machines means that the provider can offer a multitude of different environments tailored to customers, which was previously infeasible. Users might even be allowed to provide their own virtual machines, therefore giving them control over the complete environment.

Cloud computing began on a best-effort basis and many solutions provided today don't offer any more service [18] [29]. Service level agreements for cloud computing are, however, provided by some service providers, but they provide only minimal service levels [6] [19].

It has been shown that both Infrastructure as a Service and Platform as a Service - two types of cloud computing where the first one offers the concept described above and the second one offers a scalable, flexible but predefined environment to users - can benefit from service level agreements as well [10].

9. Conclusions. The preceding work has described an integrated approach to using service level agreements for the control of compute jobs which allows HPC providers to offer support for various quality of service levels. The approach was motivated through a discussion of business models for service provisioning in an HPC

environment and it has been presented how the usage of SLAs can be simulated before implementing it in a production infrastructure. Due to the proposed solution including both high-level SLA management and low-level resource management and job scheduling, service providers can take advantage of service level agreements through their complete infrastructure. We shortly concluded in how far this can be transferred to the recently introduced cloud computing paradigm.

A proof of concept implementation of the above mentioned HPC scenario using three distinct service levels has been implemented, providing overarching service level support from the resource layer to the middleware layer. Following the trend to provide Infrastructure as a Service (IaaS) solutions, we are currently investigating how the general concept can be realized with the Gridway meta-scheduler which is compatible with the OpenNebula cloud toolkit and therefore would allow for the realization of an SLA-based cloud offering. This enables HPC providers to offer IaaS with distinct service levels, which is not possible at the moment.

In general, the offering of service levels can be a distinctive advantage for HPC providers as current contracts normally do not foresee the provision of service levels. Customers gain flexibility by having the possibility to choose between different service levels when submitting jobs. This also allows providers the option of offering previously unsupported service models, for example for urgent computing, which can generate a new revenue stream.

REFERENCES

[1] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, *Job submission description language (jsdl) specification, version 1.0*. `http://forge.gridforum.org/sf/go/doc12582?nav=1`. [Online, accessed 8-March-2010].

[2] Argonne National Laboratories, *OpenPBS Public Home*. http://www.mcs.anl.gov/research/projects/openpbs/.

[3] C. Baun, M. Kunze, J. Nimis, and S. Tai, *Web-basierte dynamische it-services*, (2009).

[4] BEinGRID Consortium, *BEinGRID project home page*. `http://beingrid/`, 2008.

[5] Cluster Resources Inc., *TORQUE Resource Manager*. http://www.clusterresources.com/products/torque-resource-manager.php.

[6] M. Corporation, *Download details: Windows Azure Compute SLA document*. `http://go.microsoft.com/fwlink/?LinkId=159704`, 2010. [Online, accessed 2-March-2010].

[7] K. Djemame, I. Gourlay, J. Padgett, G. Birkenheuer, M. Hovestadt, O. Kao, and K. Voss, *Introducing risk management into the grid*, in e-Science, IEEE Computer Society, 2006, p. 28.

[8] C. L. Dumitrescu, I. Raicu, and I. Foster, *Usage sla-based scheduling in grids: Research articles*, Concurr. Comput. : Pract. Exper., 19 (2007), pp. 945–963.

[9] FinGrid Consortium, *FinGrid project home page*. `http://141.2.67.69/`, 2008.

[10] G. Gallizo, R. Kuebert, K. Oberle, A. Menychtas, and K. Konstanteli, *Service level agreements in virtualised service platforms*, in eChallenges 2009, Istanbul, Turkey, 2009.

[11] F. Howell and R. Mcnab, *simjava: A discrete event simulation library for java*, in In International Conference on Web-Based Modeling and Simulation, 1998, pp. 51–56.

[12] IRMOS Consortium, *IRMOS project home page*. `http://irmos-project.eu/`, 2008.

[13] James Barkley et al., *tortugades – Tortuga is a software framework for discrete-event simulation in Java*. `http://code.google.com/p/tortugades/`.

[14] R. Katz and T. J. Allen, *Investigating the not invented here (nih) syndrome: A look at the performance, tenure, and communication patterns of 50 r & d project groups*, R&D Management, 12 (1982), pp. 7–20.

[15] D. Klusáček and H. Rudová, *Alea 2 – job scheduling simulator*, in Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010), ICST, 2010.

[16] R. Kübert, *Providing quality of service through service level agreements in a high-performance computing environment*, in PARENG 2011, 2-nd Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering, Ajaccio, France, Apr. 2011, Civil-Comp Press, pp. ***–***. To appear.

[17] R. Kübert and S. Wesner, *Service level agreements for job control in high-performance computing*, in IMCSIT, 2010, pp. 655–661.

[18] O. P. Leads, *Opennebula: The open source toolkit for cloud computing*. [Online; accessed 22-June-2010].

[19] A. W. S. LLC, *Amazon EC2 SLA*. `http://aws.amazon.com/ec2-sla/`. [Online; accessed 2-March-2010].

[20] J. MacLaren, R. Sakellario, K. T. Krishnakumar, J. Garibaldi, and D. Ouelhadj, *Towards service level agreement based scheduling on the grid*, in Proceedings of the 2 nd European Across Grids Conference, 2004, pp. 100–102.

[21] NextGRID Consortium, *NextGRID project home page*. `http://nextgrid.org/`, 2008.

[22] M. Resch, *Entgeltordnung fr die Nutzung der Rechenanlagen und peripheren Gerte des Hchstleistungsrechenzentrums Stuttgart (HLRS) an der Universitt Stuttgart*, 2008.

[23] R. Sakellariou and V. Yarmolenko, *Job Scheduling on the Grid: Towards SLA-Based Scheduling*, IOS Press, 2008.

[24] T. Sandholm, *Service level agreement requirements of an accounting-driven computational grid*, Tech. Report TRITA-NA-0533, Royal Institute of Technology, Stockholm, Sweden, September 2005.

[25] R. Schneider, G. Faust, U. Hindenlang, and P. Helwig, *Inhomogeneous, orthotropic material model for the cortical structure of long bones modelled on the basis of clinical ct or density data*, Computer Methods in Applied Mechanics and Engineering, 198 (2009), pp. 2167 – 2174.

[26] J. Seidel, O. Wldrich, P. Wieder, R. Yahyapour, and W. Ziegler, *Using sla for resource management and scheduling - a survey*, in Grid Middleware and Services - Challenges and Solutions, D. Talia, R. Yahyapour, and W. Ziegler, eds., CoreGRID Series, Springer, 2008. Also published as CoreGRID Technical Report TR-0096.
[27] SimPy Developer Team, *SimPy Simulation Package Homepage*. `http://simpy.sourceforge.net/`.
[28] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, and R. Buyya, *A toolkit for modelling and simulating data grids: an extension to gridsim*, Concurr. Comput. : Pract. Exper., 20 (2008), pp. 1591–1609.
[29] E. Systems, *Eucalyputs - your environment. our industry leading cloud computing software.* [Online; accessed 22-June-2010].
[30] S. Wesner, *Integrated Management Framework for Dynamic Virtual Organisations*, dissertation, Universität Stuttgart, Stuttgart, Germany, 2008.
[31] V. Yarmolenko and R. Sakellariou, *An evaluation of heuristics for sla based parallel job scheduling*, in Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, April 2006, pp. 8 pp.–.
[32] C. S. Yeo and R. Buyya, *Managing risk of inaccurate runtime estimates for deadline constrained job admission control in clusters*, in ICPP '06: Proceedings of the 2006 International Conference on Parallel Processing, Washington, DC, USA, 2006, IEEE Computer Society, pp. 451–458.

Edited by: Dana Petcu and Marcin Paprzycki
Received: May 1, 2011
Accepted: May 31, 2011