HPC4U and WS-Agreement

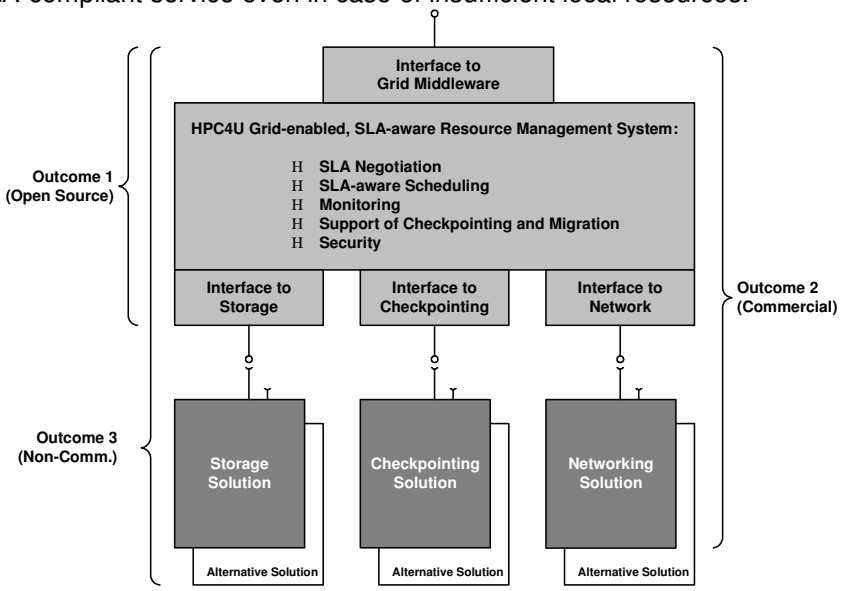
Matthias Hovestadt
Paderborn Center for Parallel Computing, University of Paderborn, Germany

A focal demand of commercial users for accepting Grid technology as regular business tool is the availability of contractually fixed service levels. In this context, Service Level Agreements (SLAs) have been introduced to Grid computing. SLAs are important instruments for defining all expectations and obligations within the business relationship between end-user and resource provider. This way, consumers are enabled to define their particular requirement profile. Such a profile does not only encompass resource requirements like type and amount of hardware resources. It also covers demands on quality of service or the specification of a deadline for job completion. Since an SLA corresponds to a regular business contract, it also defines the price for resource consumption as well as a penalty fee.

The goal of the EC-funded project HPC4U is the provision of a software-only solution for a fault tolerant cluster middleware for Internet Grids. This will be basically realized by means of checkpoint and restart of parallel processes, checkpointing and virtualization of storage subsystem, and failover functionality in network interconnect. The HPC4U system will be application transparent, so that recompilation is not necessary for benefiting from fault tolerance. This is of particular importance for commercial users, as the source code of commercial codes is usually not available.

HPC4U will allow users to negotiate for a particularly required level of Quality of Service, so that the user can depend on getting his results as agreed. By this, HPC4U introduces SLAs to commodity-based clusters providing Quality of Service (QoS). It will feature mechanisms like process and storage checkpointing to realize Fault Tolerance and to assure the adherence with given SLAs.

The HPC4U system will not only support SLA-compliant service to jobs coming from the Grid, it will also actively use the Grid for further enhancing its level of QoS. In case of resource outages the system may decide to use other Grid resources for resuming affected application. This way, HPC4U may provide SLA-compliant service even in case of insufficient local resources.



The results of HPC4U will be a mix of open source and proprietary software embedded in three outcomes:

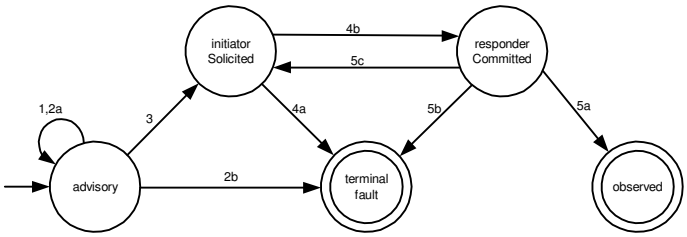
- First outcome includes the developed SLA-aware resource management system, the module for SLA negotiation, the SLA-aware resource scheduler, mechanisms for job migration over multiple administrative domains, security infrastructure and the interfaces to the check-pointing feature with networking and storage.
- Second outcome: Vertically integrated, ready-to-use HPC4U middleware, which contains all necessary fault-tolerance mechanisms such as check-pointing feature, storage, and networking. Due to the pre-existing – partly commercial – products of the HPC4U partners this product will not

be available as open source. It will demonstrate the functionality and the performance of the HPC4U work in real-world scenarios and applications.
- Third outcome: Vertically integrated, ready-to-use HPC4U middleware, consisting of non-commercial components only, i.e. Berkeley Checkpointing and LAM-MPI.

The above figure depicts the basic architecture of the HPC4U system. The resource management system is in the central position, since it provides an interface to the upper layer Grid middleware. At the lower level, the RMS interfaces the subsystems, offering fault tolerance and QoS in the domains of storage, process, and network. The interface to Grid middleware will both allow Grid middleware to negotiate on new SLAs as well as starting a negotiation process itself for finding suitable migration targets.

HPC4U focuses on supporting a negotiation process limited to static negotiation. Thus, the user will receive a template for an SLA from the system; he can fill in the template and send it back to the RMS. The RMS will decide whether this is acceptable or not. However, the user will be able to specify diffuse requests, stating that some parameters are acceptable for him within a certain range. If he has specified multiple diffuse parameters, the RMS will send the SLA back to the user with fixed parameters and will ask the user to re-confirm this specification.

HPC4U only uses the initiatorSolicited model, as responderSolicited is not useful in the scope of HPC4U. The following enumeration further explains the state transitions during the negotiation of an SLA.



1. The user initiates a negotiation by contacting the RMS with its credentials.
2. The RMS checks whether the user is allowed to use the system and responses either with a template for an SLA (2a) or refuses the request (2b).
3. The user uses the template and fills in his requirements regarding resource consumption and QoS aspects and sends the SLA back to the provider. The user might specify *diffuse requests* by giving ranges for certain values.
4. The provider checks the values in the SLA template. In case it cannot fulfill the SLA, it is rejected (4a); no counter-proposal is given. If the user specified diffuse requests, the provider fills in actual values replacing the ranges given by the user. The SLA is sent back to the user (4b).
5. If the proposal is ok for the user, it accepts it and the SLA is concluded (5a). Otherwise, the user might refuse it (5b) or make a new proposal, probably changing the acceptable ranges in the diffuse requests (5c).

HPC4U will categorize all SDTs into two categories: Resource specific SDTs and QoS specific SDTs. Resource specific SDTs categorize available resources by their type and functionality. QoS specific SDTs describe which QoS guarantees and mechanisms the HPC4U system will provide for the job. Resource specific SDTs represent the fundament of soft- and hardware components that the HPC4U system resides on. We distinguish between the types of basic components of a cluster system (e.g. Cluster:NumberOfNodes, Node:MainMemory, or Storage:Capacity). The guarantee terms for QoS parameters describe, which operations are guaranteed to the user even in case of failure. In contrast to resource specific SDTs, QoS specific SDTs affect the fault tolerance mechanisms that the HPC4U system will utilize to assure the agreed level of QoS. We distinguish between QoS specific SDTs which relate to the job and QoS specific SDTs which relate to resources.

Even if an implementation of server and client for this protocol has been finished, HPC4U does not focus on developing its own implementation of WS-Agreement. In contrast, we would like to use existing standard implementations that are able to follow new revisions of future versions. Currently we are investigating some existing implementations. Furthermore HPC4U is looking for cooperation with other projects working on SLA at level of Grid middleware, which are interested to use the services HPC4U can provide.