GWD-I (Proposed Informational)

Karl Czajkowski, USC Information Sciences Institute
Stephen Pickles, University of Manchester
Jim Pruyne, Hewlett-Packard Laboratories
Volker Sander, Forschungszentrum Jülich

Grid Resource Allocation Agreement Protocol WG January 25, 2006

Usage Scenarios for a Grid Resource Allocation Agreement Protocol

Status of this Memo

This memo provides information to the Grid community relating to the agreement of resource allocations on a Grid, and protocols that might be used for achieving this. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

This document describes a set of usage scenarios of the Grid Resource Allocation Agreement Protocol (GRAAP). The intention is to derive the required capabilities of GRAAP from this document. Specifically, we describe and analyze a number of general usage scenarios; this is followed by the detailed presentation of the specific use case of the RealityGrid project.

This document is a product of the Grid Resource Allocation Agreement Protocol (GRAAP) Working Group of the Global Grid Forum. This document is intended to provide input into the GRAAP-WG's technical work: the design of standard mechanisms to reserve resources on the Grid.

Contents

1 Introduction

The Grid Resource Allocation Agreement Protocol (GRAAP) Working Group addresses the protocol between resource consumers, such as an end-user or a Super-Scheduler (Grid Level Scheduler), and local Schedulers necessary to reserve and allocate resources in the Grid. This document clarifies the role of the protocol by describing a set of usage scenarios that must be supported by the upcoming protocol standard.

2 Scope and Background

A common requirement for service provisioning in a Grid environment is to negotiate access to, and manage, resources that exist within different administrative domains than the requester. It is therefore essential to arrange some resource usage agreement between requester and service provider. Higher-level services such as a Super-Scheduler (Grid Level Scheduler) might act as intermediaries between an end-user and a local scheduler adding an additional layer as well as serving to broaden the scope of use requests. Services might thus be composed of different levels and the acquisition of a service by some end-user requires the transitive access to all agreements.

2.1 Service Level Agreements (SLAs)

The Service Negotiation and Acquisition Protocol [SNAP] introduces Service Level Agreements to model this complex environment. An SLA allows clients to understand what to expect from resources without requiring detailed knowledge of competing workloads or resource owners' policies. This concept holds whether the managed resources are physical equipment, data, or logical services. Given that each of the resources in question may be owned and operated by a different provider, establishing a single SLA across all of the desired resources is not possible. Management functions are decomposed into different types of SLAs which can be composed incrementally; an elegant solution to this problem. Three types of SLAs are proposed for this:

- **Task service level agreements** (TSLAs) in which one negotiates for the performance of an activity or task. A TSLA is, for example, created by submitting a job description to a queuing system. The TSLA characterizes a task in terms of its service steps and resource requirements.
- **Resource service level agreements** (RSLAs) in which one negotiates for the right to consume a resource. An RSLA can be negotiated without specifying for what activity the resource will be used. For example, an advance reservation takes the form of an RSLA. The RSLA characterizes a resource in terms of its abstract service capabilities.
- **Binding service level agreements** (BSLAs) in which one negotiates for the application of a resource to a task. For example, an RSLA promising network bandwidth might be applied to a particular TCP socket, or an RSLA promising parallel computer nodes might be applied to a particular job task. The BSLA associates a task, defined either by its

TSLA or some other unique identifier, with the RSLA and the resource capabilities that should be met by exploiting the RSLA.

These SLAs can be used to separate the complex service acquisition process into multiple phases, each negotiating the appropriate level of agreement. This modelling facilitates a distributed coordination of resources without any resource-resource trust relationship.

2.2 Advance Reservation

An RSLA delegates specific resource capacities to the requester that are held by the service provider. An RSLA can thus be viewed on what is often called “Advance Reservation”. The term “Advance Reservation” will be heavily used within the document. The following definition is intended to clarify its semantics:

> An *Advance Reservation* is a possibly limited or restricted delegation of a particular resource capability over a defined time interval, obtained by the requester from the resource owner through a negotiation process.

3 Scenarios

This section presents a list of scenarios in which the Grid Resource Allocation Agreement Protocol is intended to interface between resource consumer and provider. The intention of these scenarios is to clarify the context of GRAAP and to motivate particular requirements the protocol should fulfill.

3.1 The very long running service or application

This use case captures the notion that not all reservation requests must be for a finite duration. It will not always be possible to predict when a reservation will complete.

3.1.1 Example Usage

A user has developed a service that they wish to provide for the grid community at large. For example, it may implement some particular compute intensive algorithm. They do not have the resources necessary, so they wish to find a resource provider on which they can host their service. As a service, it must remain running as long as any clients of the service may wish to contact it. Typically, we cannot predict or even know when clients no longer exist for a service, so there is no way to specify a run-length or expected completion time.

3.1.2 Requirements for the Protocol

GRAAP is used in two stages here. First, the user who has developed the service must receive the capability to use a resource offered by some provider. The system here must provide for an infinite or "do not know" completion time or expected running length. For purposes of discussion, we consider infinite and “do not know” as equivalent because they each require the local scheduling system to maintain the reservation for all time. One choice of terminology may be preferred within the protocol to make it clear to the users how to make use of this functionality. Additionally, GRAAP is used by end-users to interface with this established service.

3.2 The variable demand long running service

As in the very long running service example, we assume that completion time for this service is not known. We do know, however, that the resource level required will change over time. Put another way, the peak demand for resources is significantly different than the mean demand. The reservation must be negotiated such that this profile of demand over time can be included in the agreed reservation.

3.2.1 Example Usage

Consider a service that processes results from a particular sort of experiment. When the experiment is run, the service will see a great deal of traffic, but at other times will have much lower demand. If we know the schedule for experiments to be run, we should create a resource reservation that matches this schedule and therefore prepares the service for these high loads. Alternatively, the demand could be on a regular, predictable schedule such as every second Friday when payroll is processed.

3.2.2 Requirements for the Protocol

The protocol must provide a method of specifying a variable and most likely periodic demand profile. If it is a deterministic demand profile, we have to map this profile on to the calendar. If it is probabilistic, we must map it with some probability and attempt to optimize the overall use of resources modulo the probability of over commitment.

3.3 The configurable application

Some reservations may have multiple acceptable resource levels or configurations. We therefore do not expect that a reservation negotiation can have only one possible outcome. It is possible that multiple alternative reservations could satisfy an application’s need so the protocol must permit alternatives to be selected among.

3.3.1 Example Usage

A user has an application that can be configured for multiple resource levels, but it cannot change once deployed. For example, a parallel computation that requires a power of 2 level of compute resources, but can be configured to use any power of 2. It must interact with the scheduling system in such a way that it can determine available resource levels, and allow the user to choose the most appropriate alternative based on their QoS (such as cost, desired turnaround time, etc.). With the reservation completed, the application can be configured and deployed with agreed upon resource level.

3.3.2 Requirements for the Protocol

This scenario requires the introduction of some sort of negotiation with the scheduler. We need to ask it what is available and what the QoS/SLA/economic/etc. parameters of various configurations are. Rather than simply making multiple requests, it would preferable if the scheduler could provide us with the set of options available to us. These options should be constrained by what we can say about the configurations the application is capable of handling.

3.4 The templated application

We realize that application development and application execution may be performed by different people. We therefore desired a method of describing an application’s resource demands in a generic, and parameterizable manner so that those running the application can negotiate a reservation without knowing the specifics of the application's requirements.

3.4.1 Example Usage

A user has developed an application, and wishes to provide that application for a community. The community need not be required to understand the details of resource levels and topologies required by the application. An example would be a grid service that has a needed topology in terms of the grid service hosting environment (which itself may have resource and topology requirements) and particular Internet connectivity as well as local connectivity to storage, etc. Community members only need to understand that the application performs a certain desired function. The original developer should be able to provide a template describing the application's resource requirements at a high-level so that others can create reservations for running the application, perhaps adding some parameters based on the resource levels they wish to consume for this run of the application.

3.4.2 Requirements for the Protocol

The protocol should support a sort of template for describing the application's requirements, but one which is written by the developer, not the submitter. The template should include the sort of things that a normal submission request would, but should include parameterization, and should be nameable. The name may take the form of a Grid Service such that the template is accessed in the same manner that other Grid Services are accessed. Ideally, the templates would be composable so that complex requests may be built out of smaller (sub-)requests. In this way, we build a library of logical resource requests on the Grid that can be used to build complex requirements. Note that this covers both: the ability of the resource description language to compose requests and the ability of the protocol to transport existing agreement information.

3.5 Monitoring of resource utilization and changing of reservation parameters

Reservation is not a one time thing. A reservation has a life-cycle and we wish to be able to track it through that life-cycle.

3.5.1 Example Usage

A reservation is created for a service. As it runs, we wish to monitor when it becomes active and what allocations have been performed. Further, progress of the application may lead us to want to change the reservation, and therefore the configuration of the application. We term these sorts of applications “adaptive applications”. This may be particularly important in an area such as weather forecasting where late results are useless results. Being able to react to changes in either the received level of service or, to modify the requested level of service based on availability facilitates an effective problem solving environment.

3.5.2 Requirements for the Protocol

Once submitted to the scheduler, users should be able to monitor the state of their requests. Monitoring could include receiving events on various state transitions (e.g. start, stop, suspend, etc.) or on-line monitoring of various parameters (queuing time, total run-time, total resource consumption, total network bandwidth used, etc.). The scheduler needs to both monitor these things as well as provide an on-line and likely post mortem report on these things. A user should be able to disconnect and re-connect later to request an update on the values. Here, the protocol should support mechanisms which can be used for authentication and authorization of future reservation or acquisition references.

After submitting a request, the user should be able to modify that request. For example, they may wish to reduce the requested resource levels to get the application started sooner. The user may want to modify an existing request rather than stop and re-submit the request so they do not lose any built up priority in a FIFO or other time-sensitive queue. Reservations must not be considered rigid objects. They should be malleable while in the waiting state, and, ideally, we can operate simply on the changes in the request without having to treat things as deletes and re-inserts. This procedure complies with solution proposed in the Advance Reservation document [GFD-E.5].

It is understood that changes to a reservation will take place under the same constraints, and using a similar method to which reservations are initially made. That is, a re-negotiation of the parameters of the reservation will be performed to make the changes using the same negotiation protocol used to create the initial reservation. The difference may be that policies MAY be in place to assure some alterations to the reservation. For example, the initial negotiation may result in a reservation that is guaranteed to allow cancellation or reduction in resource demands. This is not a requirement, however, and such guaranteed updates to reservation parameters must be specified in the policy associated with the reservation when it is agreed upon.

3.6 File Transfer Scenario

Grid applications often process remote data. Pre- and Post-staging are therefore relevant operations.

3.6.1 Example Usage

Consider a finite element application that is comprised of three phases. The pre-processing phase might be a result of a computer aided design (CAD) process in which the designer partitions the geometrical information into sub-domains and creates a finite element mesh for mathematical analysis. This process is done at some CAD workstation in the user's office. The second phase solves the matrix equations on some remote supercomputer. It therefore needs access to the mesh data. Finally, the post-processing is performed on the CAD workstation again to check the validity of the computed results.

3.6.2 Requirements for the Protocol

The workflow of the described scenario requires some basic synchronisation. The equation solver on the remote supercomputer requires the data to be accessible prior its can process. Similarly, the user must have some notification when the processing has ended and the data has been transmitted to its destination. These staging operations could either be performed in a best-effort network environment in which the synchronization mechanisms do not assume a particular bandwidth service level, or, when the network is capable of providing advanced services on demand, by additionally asking for a deadline file staging reservation. Here, deadline file staging assures that the data is available when the computing resource becomes available. When combined with advance reservations, the deadline file transfer would use a reservation-based approach to allocate the appropriate level of bandwidth

3.7 Co-allocation of several resources

Grid applications may use a collection of heterogeneous Grid resources located in various administrative domains. GRAAP must support these scenarios. This subsection specifies an example for this scenario and derives the issues associated with it.

3.7.1 Example Usage

Advanced computational problem-solving methods often rely on the ability to visualize and steer the application in real-time. In a Grid environment, these applications might be executed in a geographically dispersed environment, i.e. some supercomputing resource is serving a remote

VR-device such as a Holobench or an ImmersaDesk. Now consider that the supercomputer application processes some raw data served by some cache service. We receive a workflow which requires the coordinated allocation of several resources. Now suppose that this application is placed in an environment in which a community scheduler provides advanced services for scheduling these types of applications. The user will submit the workflow to the community-scheduler with the intention that this will take care about the appropriate---in terms of user specified constraints---allocation and execution. The super-scheduler must then coordinate the use of the appropriate cache machines, of the network resources used during the data transport, the selection and allocation of an appropriate compute server(s), and the tertiary storage system which might be used to store the result of the workflow. Of course, some of the listed resources provide their services through local resource management systems. In particular, network services may be provided by the use of a bandwidth-broker that allows a network path to be treated as a single resource, comparable to a VLL (Virtual Leased Line), while the compute resource candidates are operated by what is commonly named Distributed Resource Management (DRM) system.

3.7.2 Requirements for the Protocol

We can derive a set of requirements from the example listed above. Note that GRAAP is used for both: the communication between user and community scheduler and the communication between community scheduler and the service provider. First, and most essential, GRAAP must be able to interface with various resource management systems. This basically means that either the management systems itself are implementing GRAAP, or some service is in place which is adapting the protocol semantics of GRAAP to the provider specific interface. Furthermore, we recognize one of the major intentions of GRAAP: the standardization of the access to advance reservation capabilities. Whenever the bandwidth broker or the DRM is offering advance reservation, GRAAP should interface with these capabilities. There are several reasons why we claim that advance reservation is of benefit in this example, the major one is the following: The ability to perform advance reservations including deadline file transfers simplifies the determination of a combined schedule for the whole workflow, without unnecessarily occupying resources (such as disk space, because the data is there, but the job cannot be scheduled yet). However, as the application is steered, the actual resource requirements might vary over the time. The network demand, for example, might vary over the time and its actual characteristic might depend on both: the remote control and the processing. It is therefore essential to support the renegotiation of an existing reservation.

Co-allocation drives a requirement for a two-phase protocol for creating a reservation. In the first phase, the consumer and the supplier negotiate a reservation. In the second, they commit to put the reservation in place. To allocate resources across multiple sites, a user must negotiate with those sites independently. Only when an agreement has been formed with each of the required sites can the user proceed to commit to those agreements. Any commitments to individual sites may result in charges or agreements that are un-needed because other sites cannot be found. As stated above, cancelling of a reservation is not a guaranteed operation, so the two-phase protocol is the only method a consumer has of ensuring that reservations are not made, and resource allocated needlessly.

3.8 The "bottom-feeder" application

We typically think of a reservation requesting a required set of resources for an interval of time. In some cases, we may wish to permit applications to request whatever “left over” resources are not being used by other applications in the system.

3.8.1 Example Usage

A user has an application which can consume all available resources such as running SETI@home or a distributed.net factoring program. It can be assumed that this is a low priority task, but would like to consume any resources that are not otherwise allocated. This will by necessity be an adaptive application that can withstand unpredictable suspension and resource level changes.

3.8.2 Requirements for the Protocol

This scenario describes a somewhat unusual case of an advance reservation. How does a bottom-feeder get rolled into the schedule/calendar? How do we arbitrate between multiple bottom-feeders that could use the same resources? As with any adaptive application, how does the application become informed of changes in resource levels? To support this scenario, GRAAP must be able to propagate policy information, i.e. constraints and priorities, and to notify about changes on resource levels.

3.9 Complex Workflows

Grid applications may consist of several interdependent computational steps. Some steps might be executed in parallel, while others depend on the results of the former execution and have to be processed after the required input data has been computed.

3.9.1 Example Usage

Consider some complex simulation process which operates at multiple stages. Now consider that these stages can be structured as a directed acyclic graph which represents the workflow. We can thus model the timely order of the execution stages.

3.9.2 Requirements for the Protocol

The main issue here is that the protocol must support efficient synchronisation between each level of the directed acyclic graph. Furthermore, it should be possible handle this request as a composition of individual requests, each of it with a potentially associated advance reservation.

4 A Use Case: “RealityGrid”

The RealityGrid project (http://www.realitygrid.org) aims to predict the realistic behaviour of matter based on the properties of the microscopic components using diverse simulation methods (Lattice Boltzmann, Molecular Dynamics and Monte Carlo) spanning many time and length scales and the discovery of new materials through integrated experiments. A central theme of RealityGrid is the facilitation of distributed and collaborative exploration of parameter space through computational steering and on-line, high-end visualization.

4.1 Advance Reservation and Co-allocation Requirements

A typical RealityGrid scenario involves a large-scale simulation running on a massively parallel system at on site coupled to a high-end visualization system at another site with the steering and display interfaces running at one or more remote sites The simulation component periodically (or as demanded by the steerer component) emits “samples” for consumption by the visualization component, while grid middleware is responsible for the transfer of data between components.

The most pressing requirement for advance reservation in RealityGrid arises out of the need to co-allocate (or co-schedule) processors to run a parallel simulation code and multiple graphics pipes and processors on the visualization system. Co-allocation may be required now (either by a RealityGrid developer or by a scientist engaged in routine investigations) or at some more distant time in the future (for a scheduled collaborative session). We expect advance reservation to subsume both co-allocation scenarios.

The visualization resources (b) will usually be located on a different system to the computational resources (a). The two sets of resources ((a) and (b)) will often be located on systems owned and administered by different organisations, and the administration teams within the two organisations, if aware of each other’s existence at all, are unlikely to have established comprehensive Service Level Agreements. It is assumed that the end-user(s) will be able to access resources on both systems by presenting a single credential, through a single sign-on mechanism based on digital certificates such as GSI or the UNICORE security model.

It is anticipated that the system (a) running the simulation will typically be a massively parallel system with a workload characterised by sustained heavy demand and therefore the allocation of resources is likely to be entrusted to a batch scheduling system. The characteristics of the visualization system on the other hand are likely to vary, with demand for graphics and CPU ranging independently from low to high, and there may or may not be a batch scheduling system in place. The resources (b) required on the visualization system include both graphics pipes and processor (CPU+memory) resources. In general, whatever system may exist for booking the graphics pipes is unlikely to be integrated with whatever system may exist for booking the processors.

We may therefore distinguish two cases:

- co-allocation of processors on the simulation system, and graphics pipes on the visualization system;
- processors on the visualisation system; where the visualization system does not run a batch scheduling system, co-allocation of processors on the simulation system and graphics pipes on the visualization system, relying on chance (or external arrangement) to acquire a sufficient share of CPU resource for visualization purposes.

The ability for a RealityGrid user to reserve processors and graphics pipes manually *without involving system administrators* would be useful now, and would remove a significant barrier to the routine use of computational steering. The ability for an agent to do the same will be important for the resource broker that will be developed in the later stages of the RealityGrid project.

4.2 Future Requirements for Advance Reservation

Based on current projections, the largest computationally-steered simulations that RealityGrid is likely to undertake will require bandwidth between the simulation and visualization systems of order 1 Gbps in order to achieve satisfactory interactivity. The bandwidth requirements between visualization systems are less demanding – 100 Mbps will be adequate for most purposes – but reasonably good latency and jitter characteristics are desirable. Thus the ability to make advance reservations of network bandwidth with certain quality of service characteristics and using the same protocols as for the reservation of processors are seen as desirable by RealityGrid.

RealityGrid's design philosophy is component based. In the computational steering scenario described above, there are two coarse-grained components, simulation and visualization, deployed on two different systems. However, RealityGrid is investigating finer-grained componentisation in which the simulation is composed out of a number of smaller communicating components, each of which must be deployed onto (possibly remote) computational resources at run-time. Thus RealityGrid will need robust mechanisms for co-allocating much more complex sets of resources, involving many advance reservation requests many .

RealityGrid has a significant work package devoted to performance control. The goal of this work package is to optimise the collective performance of the components comprising the RealityGrid application based on performance information collected at run time. Initially, the set of resources will be assumed to be fixed during execution, and it is by redistributing components across this set of resources that the performance control system hopes to achieve performance improvement. Ultimately, however, the ambition is to adapt the application to utilize new resources that become available during execution; this is likely to require rather specialist functionality of the advance reservation system such as the ability to renegotiate an existing reservation.

5 Security Considerations

This document is informational, and contains a set of use cases. As such, it does not address security considerations directly. However, the scenarios described in this document rely on the abilities of a convenient authentication and authorization environment. The negotiation process performed by GRAAP of course requires a mutual authentication between resource provider and consumer. However, requests might be composed of existing agreements. It is therefore important to reflect this composition in the related security framework. Co-allocation brings up other concerns as a user may have different identities in different domains, but wish to make allocations that span those domains. Policies about who has authority to see and alter reservation state will also be important. Policies describing the "ownership" of a reservation and listing rules for who is allowed to perform what operation on the existing agreement have to be considered.

6 References

[GFD-E.5] V. Sander and A. Roy, "Advanced Reservation API", Grid Forum Document (GFD), Experimental 5 (E-5)

[SNAP] K. Czajkowski, I. T. Foster, C. Kesselman, V. Sander, and S. Tuecke. "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002), Revised Papers, LNCS Vol. 2537, pages 153–183. Springer Verlag, 2002.

7 Author Information

Karl Czajkowski
Univa

karlcz@univa.com

Stephen Pickles
Manchester Computing
The University of Manchester
Oxford Road
Manchester M13 9PL
United Kingdom

stephen.pickles@manchester.ac.uk

Jim Pruyne
Hewlett-Packard
Laboratories
1501 Page Mill Rd.
Palo Alto, CA
United States of America

jim_pruyne@hp.com

Volker Sander
Zentralinstitut für Angewandte Mathematik,
Forschungszentrum Jülich GmbH,
52428 Jülich
Germany

v.sander@fz-juelich.de

Jon MacLaren (Final Editor)
302 Johnston Hall
Louisiana State University
Baton Rouge, LA 70803
United States of America

maclaren@cct.lsu.edu

Please contact Jon MacLaren if you have any comments regarding this document.

8 Intellectual Property Statement

9 Full Copyright Notice