

Resource Management for Early Production Grids

Erik Elmroth*[†] Åke Sandgren[†] Johan Tordsson*

June 13, 2003

Abstract

This contribution presents the ongoing development of a resource manager for use in early production grids. Even though our main focus is to develop a stable brokering facility for current production grids, we also address features needed in further improved resource managers for future enhanced grid infrastructures. The primary target environment is the NorduGrid platform, comprising around 20 parallel systems in 5 countries, available for production grid jobs 24 hours a day. Application characteristics considered include serial, parallel, and coordinated multi-resource jobs running in sequence or in parallel, all types in either interactive or non-interactive mode. The brokering process aims to minimize the time to delivery for each individual job and is based on a number of new features including reservation capability, information about currently used or reserved capacity, benchmark-scaled time predictions, and queue adaptation capability. We present the basic motivations for all these features and discuss various issues regarding their implementations in the current grid environment.

Keywords: Grid resource manager, resource broker, NorduGrid, production grid, Globus

*Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden.

[†]High Performance Computing Center North (HPC2N), Umeå University, SE-901 87 Umeå, Sweden.

1 Introduction

We present the ongoing development of a resource manager for use in early production grids. Even though our main focus is to develop a stable brokering facility for current production grids, we also address features needed in further improved resource managers for future enhanced grid infrastructures. Our primary target is the NorduGrid platform [11, 15], comprising around 20 parallel systems in 5 countries, available for production grid jobs 24 hours a day.

Application characteristics considered include serial, parallel, and co-ordinated multi-resource jobs running in sequence or in parallel, all types in either interactive or non-interactive mode. The brokering process aims to minimize the time to delivery for each individual job and is based on a number of new features including reservation capability, information about currently used or reserved capacity, benchmark-scaled time predictions, and queue adaptation capability. We present the basic motivations for all these features and discuss various issues regarding their implementations in a current grid environment.

The main task for a resource manager in a Globus environment [6, 8] is basically to match the requirements of one or more applications with a set of resources. This task requires a large number of subproblems to be solved and a long list of criterias to be taken into account.

The problem of deciding which resources to use for a particular application is made significantly harder by the lack of global control. We cannot assume that this resource manager has unique control over the grid resources. On the contrary, the typical situation is that resources are allocated through a large number of different resource brokers, and also by other users sharing the same resources but without making use of the grid infrastructure. Therefore, the resource broker must be based on information about resources and utilization rather than trying to gain control over how the resources are utilized.

Despite this difficulty, it is vital that the resource manager has capability to optimize the scheduling for different types of jobs, and at the same time, make the user feel that the resource allocation and scheduling process is fair, given the rules and policies agreed on. One should keep in mind that this basic requirement has shown to be hard to satisfy already with a traditional scheduler on a single parallel computer system, with global control.

There exist a number of other projects working on related issues, some trying to cover the complete resource brokering problem, other focusing on more specific features, such as to enable resource advance reservations, negotiation for service level agreements, allocation of resources for interactive use, scheduling algorithms etc. For examples, see [1, 2, 3, 4, 7, 13].

The outline of the manuscript is as follows. In Section 2, we briefly describe the NorduGrid and SweGrid production grids. The characteristics of the applications considered are classified in Section 3. Section 4 outlines the main steps of the resource management procedure and Section 5 elaborates on important considerations for improving the resource selection. Section 6 presents the features being implemented in this resource management system, and Section 7 gives some concluding remarks. Finally, Section 8 gives a short list of references.

2 The NorduGrid and SweGrid Platforms

The current development of this resource manager is performed in the NorduGrid grid environment [11, 15]. NorduGrid was initiated in the year 2000 as a grid testbed, aimed to develop Grid middleware and applications in the Nordic countries, extensively using modern software utilities and tools.

Today, NorduGrid has matured into a 24-7 production grid facility including around 20 parallel computer systems in Sweden, Norway, Denmark, Finland, and Japan. It includes some true high-end resources, such as a 240 processor Linux cluster at HPC2N, Umeå University and a 400 processor Linux cluster at NSC, Linköping University. The software for NorduGrid is based on the Globus toolkit, extended with a NorduGrid package freely available for download from the NorduGrid website. Most of the resources are using OpenPBS [12] with different schedulers as the local batch system.

Our resource manager is also aimed for the SweGrid platform currently under buildup. SweGrid will complement NorduGrid with 6 major Linux cluster, each on the order of 100 processors, installed at six supercomputer centers in Sweden. SweGrid will initially be based on the NorduGrid software, and is planned to be available for production use during fall 2003.

3 Application Characteristics Considered

It is vital to take into account the different expectations on the level of service, depending on the nature of the applications. Does an application need constant access to certain resources, need co-allocation of different resources, etc, or is the only requirement to complete as many individual jobs as soon as possible? For example, high-throughput applications typically asks for a minimum time until the last job is completed, while interactive simulations may require constant access to some resources, coordinated with access to other resources. It is important that the resource manager provides support for the user to specify the required usage and that the manager is able to handle these differences in expectations appropriately. For this platform, it would be relevant to consider the following types of job characteristics:

1. Serial jobs (as single uni-processor jobs or multiple independent uni-processor jobs).
2. Parallel jobs running on *one* resource.
3. Coordinated jobs (of any of the two types above) running on multiple resources where, e.g., output from some jobs may be used as input in other.
4. Parallel jobs running on multiple resources, communicating between the resources using, e.g., a Grid-enabled MPI implementation [9].

For each of these, both interactive and non-interactive jobs can be considered.

4 Resource Management Procedures

In order to solve the general resource management problem, we can identify a number of major subproblems to be solved:

- Dynamic discovery & characterization of the resources.
- Resource selection.
- Immediate reservation (allocation).
- Advance reservation.
- Co-allocation/co-reservation (for simultaneous use of several resources and/or for coordinated use of resources in sequence).
- Adaptation.

The first five items specify the basic functionality needed to identify resources, check their capabilities, match them with the requirements, and to allocate them for different types of jobs. The reservation features extends the quality of service capabilities, and is vital for certain types of applications, e.g., requiring interaction or simultaneous access to several resources. Simultaneous access to several resources also requires co-allocation or co-reservation. However, in order to efficiently make use of new information, the reservation and co-allocation features are not sufficient, instead the system also must support adaptation. Adaptation may be used to improve the utilization of the resources and to better respond to an application's requirements, for example, when other reservations get cancelled, other programs terminate earlier than expected, shared resources get congested, etc.

Notably, these steps are not to be viewed as a true sequence of operations, as, e.g., some of them need to be mutually coordinated.

5 Resource Selection Considerations

The best choice of resources for a given application at a particular time depends on a number of parameters, several of which are difficult to determine or approximate. Below, we list some of the more fundamental issues to consider during this procedure:

- Actual performance or capacity limitations of the resources.
One of the most fundamental requirements is that the capacity and capability of the resources to be used fulfill the application's requirements, e.g., requirements on the amount of memory on the resource.
- Currently used and reserved capacity.

In order to approximate when an application can run on a specific resource one need to take into account the currently used and reserved capacity on that resource.

- Different local (e.g., scheduling) policies in different domains/resources.

The fact that the availability of different resources are restricted by different policies must be taken into account. For example, a large parallel system may have the restriction that jobs requiring large number of CPUs are only scheduled during nights or weekends; on other systems interactive usage may only be allowed during daytime, etc.

- Individual users' access rights (and possibly priorities) on different resources.

Different users may have different access rights to computers, software, software licenses, etc. In order to handle this, the resource manager needs to have capability for authorization filtering and priority determination.

- Location of code and data.

The fact that the code and input data initially is located at some specific resources and that the output data is expected to be stored at some specific resource has to be considered. Already in the trivial case with one application program to be executed on one CPU with one file of input data, one may have to consider moving the data to the program, the program to the data, or both the program and the data to a resource with higher capacity. In order to predict the time needed for the data movements, information about network performance and utilization is needed, and optimally it should be possible to reserve network capacity. Notably, one way to improve the total time to delivery for applications requiring large data sets is to extend the data replication capacity and improve the algorithms and methods for data scheduling.

6 Implementation Issues

This resource manager includes a range of features that address the issues discussed in the previous section. In the following we review some of these.

Total time to delivery. When discussing the time for completing a job, we distinguish between the actual execution time, the waiting time in queue, and the time needed for pre- and postprocessing, i.e., the time for downloading software and input files, and uploading the output files. Hence, the total time to delivery is the total time from the job is submitted to the time when the output files are stored where requested. Typically, the job of the resource manager is to select the resource that minimizes this quantity. For this, and other reasons presented below, it is important to be able to predict also the time needed for the pre- and postprocessing operations.

Reservations. The implementations of features for reservation, i.e., for the user to be able to specify when a certain job should run, is currently depending

on the support provided by the local scheduler in the underlying batch system of each resource. Currently, our main focus for reservations is on batch systems using the MAUI scheduler as this scheduler has the support required locally for making reservations [10, 14]. The reservation process is a two-way negotiation, where the resource manager’s request is first processed by the scheduler. The request can, e.g., be a fixed starting time, a time interval, a requested latest time for completion etc. The scheduler returns a suggested action for the job, e.g., a suggested start time or time for completion, and keeps the corresponding time slot temporarily reserved. Finally, the resource manager has to accept or reject this reservation, possibly after interaction with the user.

For the reservation features to work properly, it is vital to have a good prediction of the time needed for the download of code and input data, since the download must be completed before the reserved starting time.

Currently used and reserved capacity. What information that can be made available about currently used and reserved capacity also depend on the underlying scheduler. Also here, MAUI provides good support about queued jobs, while other schedulers may give less information. The main reason to ask for the queue information is to enable an approximation of the expected time before the job will be scheduled, if submitted, to a specific resource. Notably, also with good information about the currently used and reserved capacity, the time when a new submitted job would actually start depends also on things such as: the algorithm used by the scheduler of that local batch system; the difference between the expected and actually required time of the jobs in queue; and other submissions possibly being made after the queue information is given but before our job is submitted. Despite these difficulties, the information about currently used and reserved capacity is important to improve the resource selection decision.

However, with good reservation capabilities, a better approximation of the expected time for start of the execution is given by requesting a reservation as soon as possible. Then also the local scheduling policy would be encountered for, and other nearly simultaneous submissions would not overrule the estimation. Hence, the reservation can naturally be used for the negotiating of binding service level agreements.

Interactive jobs. Both immediately allocated jobs and queued jobs can be specified to run as interactive jobs. This is basically a request for the standard input and output to be directed to the user’s terminal, where the request is made.

Benchmark-scaled time predictions. The traditional scenario, where the user specifies the requested wall-clock time and some required system characteristics, such as memory requirements is hardly useful in a heterogeneous environment where the resource manager tries to optimize the utilization. First, we have the fact that the performance of the computers processors easily can vary

by a factor of ten or more. Second, computers with similar peak performance numbers may perform quite differently on different types of applications, e.g., floating point vs. integer computations or compute intense vs. memory intense applications.

Hence, we allow the user to specify not only the expected wall-clock time required, but also on what system this expectation is valid and which benchmark that best characterizes the performance expected of the application. Based on this information, we scale the wall-clock time requested by the different systems mutual performance difference on the benchmark in question.

The list of benchmarks will evolve over time, but should include widely understood benchmarks, such as Spec, HP Linpack, STREAMS, Pallas MPI, NAS benchmarks, etc. [5].

Besides improving the total time to delivery for individual jobs, this will make it possible to improve the total throughput on faster systems.

Adaptation. The implementation of true adaptivity is still regarded a future task, but a “cancel and resubmission” feature is included to provide adaptability for significant changes in the queues on different resources. Hence, the user can ask the resource manager to continue to look for better upcoming alternative resources after the job has been submitted. If found, the first submission is cancelled and a resubmission is made to the resource with a shorter total time to delivery.

Resource selection. Given the features listed above, the actual resource selection is a two step procedure, where we first identify which resources that fulfill the basic requirements, and then we identify which of these that gives the minimum time to delivery. Notably, the computational time is then based on the benchmark-scaled numbers and the pre- and postprocessing time on the best estimates determined from available information about locations and network capacities.

7 Concluding Remarks and Future Directions

We have presented the ongoing development of a resource manager for use in early production grid environments. These environments are not yet to be considered mature in all aspects, even though, e.g., NorduGrid provides stable working conditions 24 hours a day on large number of parallel computer systems. Hence, the surrounding infrastructure lacks some of the features required to tackle the long-term resource brokering goals. So, even though we currently address most of the issues listed in Section 5, there are still lots of research to be conducted and software to be developed in order to consider these problems finally solved.

Some other problems has not yet been addressed, e.g., relating to different local policies for different resources and differences in access rights or priorities among users. In the current NorduGrid environment this is not too critical

because of rather small such differences, but with a wider spectrum of policies, these questions become important. A related problem, also not yet considered, is to include individual user's (or groups of users') resource utilization limits in the resource allocation procedure.

A major and important task that requires significant additional support in the grid infrastructure is the possibility to reserve network capacity. Due to the lack of this infrastructure, this is yet to be considered a future direction of our work. Finally, we also like to stress the importance of evaluating the brokering features in real production use, e.g., where the resource selection and data replication features may interact for further reducing the total time to delivery.

References

- [1] G. Alosio and M. Cafaro. Web-based access to the grid using the grid resource broker portal. *Concurrency Computat.: Pract. Exper.*, 14(13-15):1145–1160, 2002.
- [2] J. Brooke and D. Fellows. Draft discussion document for GPA-WG – Abstraction of functions for resource brokers. Technical report, Global Grid Forum, 2003. <http://grid.lbl.gov/GPA>.
- [3] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In Feitelson D.G. et al., editors, *Job Scheduling Strategies for Parallel Processing*, volume 2537 of *Lecture Notes in Computer Science*, pages 153–183, Berlin, 2002. Springer-Verlag.
- [4] K. Czajkowski, S. Pickles, J. Pruyne, and V. Sander. Usage scenarios for a grid resource allocation agreement protocol. Technical report, Global Grid Forum, 2002. <http://www.fz-juelich.de/zam/RD/coop/ggf/graap/sched-graap-3.0-GGF7.pdf>.
- [5] N. Edmundsson, E. Elmroth, B. Kågström, M. Mårtensson, M. Nylén, Å. Sandgren, and M. Wadenstein. Design and Evaluation of a TOP100 Linux Super Cluster System. Technical Report UMINF-02.23, Revised, May 2003. *Submitted to Concurrency and Computation: Practice and Experience*.
- [6] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Int. J. Supercomputer Applications*, (2):115–128, 11 1997.
- [7] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Int. Workshop on Quality of Service*, 1999.
- [8] Globus. <http://www.globus.org>.

- [9] N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing*, 2003 (to appear).
- [10] J. MacLaren. Advance reservations state of the art. Technical report, Global Grid Forum, 2003. <http://www.fz-juelich.de/zam/RD/coop/ggf/graap/sched-graap-2.0.html>.
- [11] NorduGrid. <http://www.nordugrid.org>.
- [12] Portable Batch System. <http://www.openpbs.org>.
- [13] K. Ranganathan and I. Foster. Simulation studies of computation and data scheduling algorithms for data grids. *Journal of Grid Computing*, 1:53–62, 2003.
- [14] Supercluster.org. Center for HPC Cluster Resource Management. <http://www.supercluster.org>.
- [15] A. Wäänänen, M. Ellert, A. Konstantinov, B. Kónya, and O. Smirnova. An overview of an architecture proposal for a high energy physics grid. In J. Fagerholm, J. Haataja, J. Järvinen, M. Lyly, P. Råback, and V. Savolainen, editors, *Applied Parallel Computing*, volume 2367 of *Lecture Notes in Computer Science*, pages 76–88, Berlin, 2002. Springer-Verlag.