Optimised planning and scheduling of grid resources

A. Quinte, W. Jakob, K.-U. Stucky, W. Süß

Institute for Applied Computer Science, Forschungszentrum Karlsruhe GmbH,
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
E-mail: {alexander.quinte, wilfried.jakob, uwe.stucky, wolfgang.suess}@iai.fzk.de
phone: +49 (0)7247 82 5739, *fax:* +49 (0)7247 82 2602

Abstract

This article will present the concept and implementation of a resource management system (RMS). The central component of the RMS is the resource broker GORBA that plans resource allocation by a combination of heuristic processes and evolutionary algorithms. For resource planning, schedules are generated, which distribute the grid jobs to the grid resources in a defined time window. A test environment with extensive visualisation options was developed for GORBA, which will be presented in detail. Using this test environment, benchmark runs were carried out, which are needed to evaluate and further develop GORBA. Automated resource planning and the graphic visualisation options facilitate the usability of a grid environment.

1 Introduction

The possibilities of grid computing are still used by a rather small number of users. A grid user usually has to acquire detailed knowledge in this field and to deal with uncomfortable command line-controlled tools. Many potential grid users are put off by the learning effort and complexity. Of course, this problem is not new and considerable efforts have been undertaken to make grid computing and in particular the grid middleware more user-friendly by the development of suitable interfaces and management systems. Automated resource allocation to grid applications, as an important part of a grid, shall be the subject of this article. The task of a resource management system (RMS) is to distribute jobs of grid applications in a reasonable manner to resources available in the grid. From a grid user's point of view, execution of his jobs should take place at optimum costs and/or time. An RMS should therefore use planning and optimisation. Furthermore, it should do this in a transparent way, requiring just the least possible knowledge about the grid, its middleware, and resources from the grid user. Other requirements on an automated RMS are a good and cost-efficient load distribution from the provider's point of view and the capability of managing errors in order to ensure a defect-free and stable operation.

This paper will present a concept and optimisation strategies used for scheduling by a resource management system (RMS). Here, an RMS will be described, with particular attention being paid to the visualisation options of the RMS. Presently, the RMS is being tested in a test environment. This test environment shall be outlined and results of benchmark runs will be presented.

2 Resource brokering for complex application

2.1 Architecture of the resource management system

The Institute for Applied Computer Science is currently developing a resource management system for grid applications. Figure 1 shows the architecture of the RMS. Its central task consists in establishing a simple connection between grid users and a grid environment. The core of the RMS is the application manager that consists of the components of the workflow manager, job manager, and resource broker.

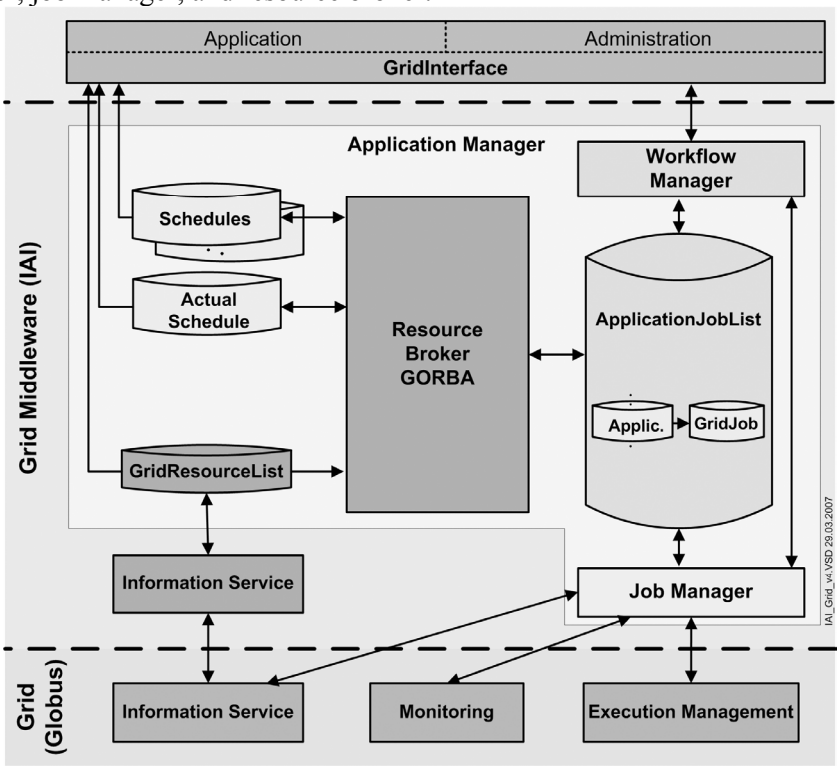

Figure 1: Resource management system with GORBA

The user specifies applications in the form of workflows and transmits them to the workflow manager of the RMS via a grid interface. The workflow manager analyses and processes the incoming jobs. By an information service, the resources available in the grid are acquired and entered into a list of available resources (GridResourceList). The resource broker allocates resources to the grid applications by generating an optimised allocation plan and transferring it to the job manager for execution in the grid. A detailed description of grid applications, resource allocation, schedules, and the resource broker will be given below.

2.2 Implementation of the resource management system

The central part of the RMS is the resource broker GORBA (Global Optimising Resource Broker and Allocator), the concept of which will be presented in detail below. For the development of GORBA, a special test environment was set up, by means of which the behaviour of GORBA can be analysed in a simulated grid environment under defined conditions. A typical test run comprises the read-in of all jobs and resource data, the generation and evaluation of schedules, and the simulated execution of the schedule generated. For simulation, a special test version of the job manager was developed, which simulates the function of a real job manager.

Figure 2 shows the schematic representation of the test environment of GORBA. The jobs required for the test runs are supplied in the form of XML files (ApplicationJob). In addition, all information on the resources of the simulated test grid is stored in XML files (GRDL). The results produced by GORBA are schedules that may be represented in a graphically processed form. Other graphic visualisation options give a survey of the application jobs, resources, and general statistical data. A graphical user interface allows for a simple operation of the test environment. This includes the read-in of all data necessary for the test runs, the execution of test runs, and the representation of the results. For the automation of more complex test runs, macros can be defined. The test runs do not only serve as a functional test, they are also aimed at improving the planning and optimization algorithms contained in GORBA.

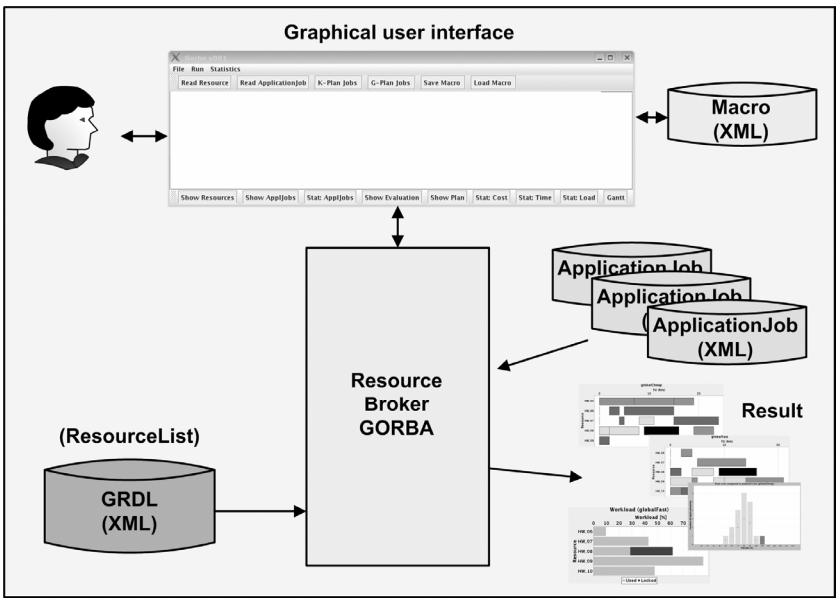

Figure 2: GORBA embedded in a test environment

Following extensive testing of GORBA in this test environment, GORBA will be applied first in a real grid environment under the CampusGrid project of the Forschungszentrum Karlsruhe [2]. The visualisation options developed in the test environment will also be used during the real operation of GORBA.

2.3 Grid application workflow

Usability and acceptance of a grid environment will largely depend on how the user is supported in specifying his grid application. A grid application is represented by a workflow that determines the order of processing the application tasks by predecessor-successor relations. A workflow, also called an application job in the GORBA context, consists of elementary grid jobs that are described by the combination of various resource requirements. The resources are hardware and software resources that execute the tasks arising within the framework of the application. Resource requirements for grid jobs are specified by the grid users and attached to grid jobs. They give all information necessary to find suitable resources for a job among all available grid resources. Resource requirements must be very flexible and allow the user to specify just a resource type, to determine all parameter values of the required resource, or to describe a detail level somewhere in between. The less specific the resource is given by the user, the more planning alternatives are obtained for the resource broker. The current version of GORBA expects

static workflows that represent directed acyclic graphs (DAG). The workflow manager determines the relevant information from the workflow and supplies this information to GORBA for planning and resource allocation.

Description of the application job is done with XML and in line with the GADL (Grid Application Definition Language) developed by the Fraunhofer Society [7] and extended to meet the special requirements of GORBA. For instance, the application job contains additional specifications for planning, such as the earliest start of the application job, the latest end, the maximum costs, and a weighing factor for the assessment of the cost and time requirements in the evaluation function.

2.4 Resource description with GResourceDL

To generate and optimise an allocation plan, current information on the resources available is required. Consequently, the resource data required by GORBA have to be made available in a suitable form. For the description of the resource data, the GADL-defined GResourceDL (Grid Resource Definition Language) of the Fraunhofer Society is used [7]. Here, the resources are divided into six resource types (*hardware*, *hardwareClass*, *data*, *dataClass*, *software,* and *softwareClass*), which may be connected with each other by four different types of relations. Whereas the resource types *software*, *hardware,* and *data* represent concrete usable resources, the resource types *hardwareClass*, *dataClass,* and *softwareClass* are classifications of resources. Figure 3 shows an example with a *hardwareClass* (HC_01), four *hardware* resources (HW_01 to HW_04), three *softwareClass* (SC_01 to SC_03), and seven *software* resources (SW_01 to SW_07). The arrows indicate the relations of the resources. Black arrows represent the relation *depends* and the light grey arrows the relation *provides*. *depends* means that the resource at the beginning of the arrow is stringently dependent on the resource at the end of the arrow. *provides* means that the resource at the beginning of the arrow provides or contains the resource at the end of the arrow.

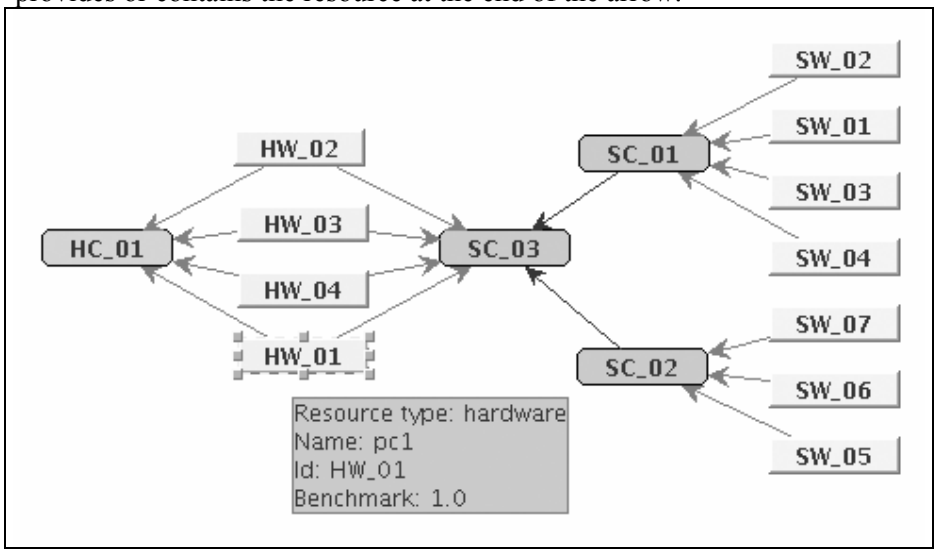

Figure 3: Example of a resource description with GResourceDL

In our example, the user, when defining a grid job, may specify the resource “HW_01” and, thus, enforce an execution on just this resource or he may specify the resource “HC_01” and, thus, leave it to the planning system on which of the four resources “HW_01“ to ”HW_04“ the grid job will be executed. Consequently, the freedom of planning can be influenced by the specification of a resource. For instance, if the user selects the resource “SC_01” for a grid job, he enforces a co-allocation, with a *software* resource

from “SW_01” to “SW_04” and a *hardware* resource from “HW_01” to “HW_04” being required for the execution of the grid job. Furthermore, the user may specify several resources for a grid job, e.g. an additional device.

With GResourceDL, a number of different properties of the resources can be described. Above all, information on the performances and mutual dependencies of the resources is of relevance to resource planning with GORBA.

GORBA is conceived for use in heterogeneous grid environments. The fact that some resources are available for the grid at certain times only is taken into account. To specify this, the GResourceDL for GORBA was extended by the indication of blocking times.

2.5 Resource planning with GORBA

As indicated by its name, GORBA (Global Optimising Resource Broker and Allocator) represents a solution for the optimisation of grid job planning and resource allocation. It was described in detail in a number of publications, e.g. in [1]. GORBA belongs to the class of planning systems [3]. This means that GORBA plans for the presence and the future. A time window for the execution on a grid resource is allocated to all grid jobs. Good and reliable planning requires certain information about the application jobs and resources. For example, the runtimes of the individual grid jobs have to be available with sufficient accuracy and the performance and availability of resources have to be known for the complete planning duration. Allocation planning with GORBA is aimed at a homogeneous working load of grid resources and at fulfilling the requirements specified by the users in the application jobs, as listed in section 2.3.

The GORBA problem of resource allocation is a scheduling problem. It is comparable to industrial job shop scheduling and belongs to the class of NP-complete problems. This means that there is no (exact) solution within polynomial time. On the other hand, an exact solution is not really required, as the corresponding plan in most cases will not be executed completely, due to events like new jobs or resource break-down. What is really needed is a mechanism for permanent replanning, which delivers nearly optimal or at least good and feasible solutions. For GORBA, a two-step planning mechanism has been developed, which utilises approved heuristics from job shop scheduling like *job with the shortest execution time first* or *job which is closest to due time first*. Both are simple and fast local optimisers. They are used to seed the initial population (set of start solutions) of the global optimiser GLEAM (General Learning and Evolutionary Algorithm and Method), an evolutionary algorithm (EA) [4]. EAs are known to be a powerful general optimisation technique which can deliver at least nearly optimal solutions for NP-complete problems. On the other hand, they converge slowly when they approach an optimum. The common solution of this drawback is a hybridisation with local search methods in order to obtain the best of both worlds: a global and fast search. Hybridisation is done here in three ways: firstly, by seeding the start population, secondly, in the process of resource selection, as will be described later, and thirdly, by local improvement of offspring generated by evolution. The last mechanism is also known as Memetic Algorithms which have proved their usefulness in many applications [5].

Two different gene models have been applied, which have in common that the grid job execution sequence is determined by the evolution. The first one (GM1) leaves the selection of a resource from a set of alternatively useable ones to the evolution and the second one (GM*2*) uses one of the following simple heuristic strategies instead: use the *fastest* or *cheapest* available resource *in general* or let the *application job priority* decide which one to use. As it is not known a priori which of these three strategies performs best

for a given planning task, the selection of one of them is left to the evolution. This means that the resource selection strategy is co-evolved together with the schedules.

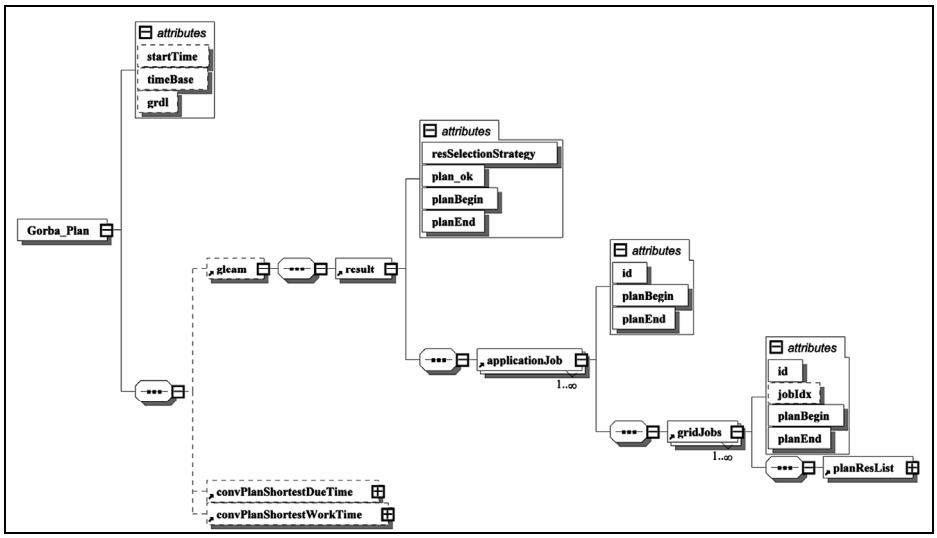

Figure 4: XML scheme of the allocation plans

3 Results and visualisation

The results of GORBA are allocation plans stored in XML files. For this purpose, an XML scheme (Figure 4) was developed. The allocation plan contains the specification of the time base (attribute *timebase*) and a reference to the description of the resources (attribute *grdl*) with which planning is made. The element *gleam* describes an allocation plan generated by GLEAM. The element *result* contains various attributes and a list of the application jobs. Each application job has its starting time and end time as attributes and a list of its grid jobs as element. Via attributes and the element *planResList*, the start and end times and the resources, respectively, are allocated to each grid job.

In an allocation plan, the necessary grid resources are allocated to each grid job and starting and end times are specified.

To support the user, a variety of visualisation options were created in the test environment. Apart from the purely textual representations of allocation plans, graphical representations exist in the form of Gantt charts. The properties of allocation plans are illustrated well by this representation. Figure 5 shows a comparison of two allocation plans. The top allocation plan was generated with a heuristic method in the first planning step, the bottom plan was generated with GLEAM in the second planning step. The improvement of GLEAM planning as compared to heuristic planning is clearly obvious. The GLEAM plan is much more compact and has a shorter total runtime.

Moreover, statistical data, such as the working load or the frequencies of exceeding cost and time limits, can be represented in the form of bar charts (Figure 6). In the top chart, the percentage values represent the degree of compliance with the requirements made by the user. For application jobs with more than 100%, the user requirements were exceeded. In the given example, it can be seen easily that two application jobs exceed the time limits, while one application job exceeds the costs. The chart for the working load represents the blocking times, during which the resource is not available, as black bars.

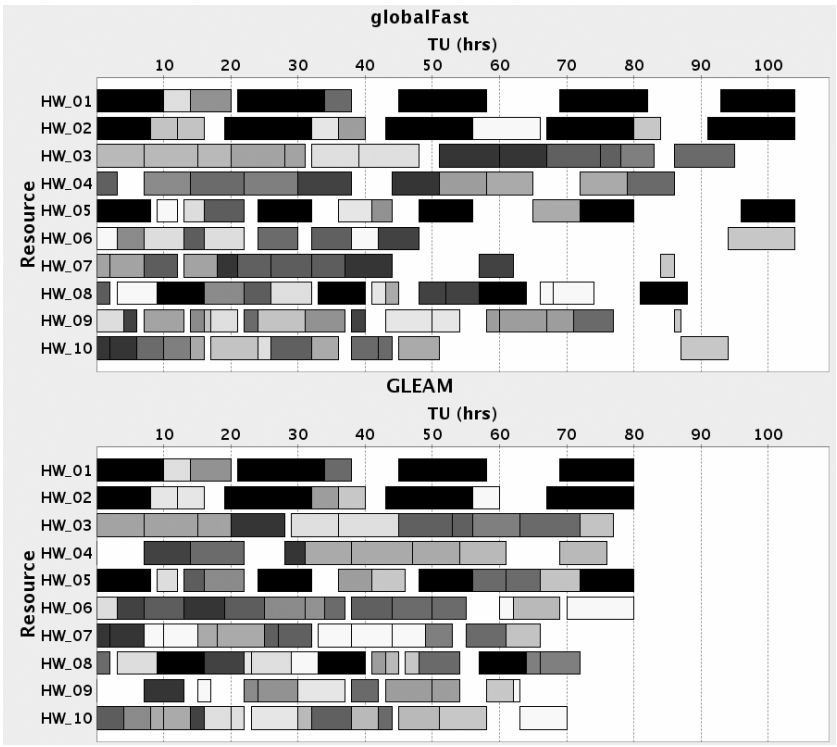

Figure 5: Representation of allocation plans as Gantt charts: result of heuristic planning (top) and result of planning with GLEAM (bottom). Black bars represent blocking times of the resources. Bars of the same grey colour are grid jobs of an application job

Another type of visualisation allows for the dynamic representation of the execution of an allocation plan by the job manager. The Gantt chart in Figure 7 shows the progress of the execution of the allocation plan. The bottom part of Figure 7 displays the status of the individual grid jobs of the application jobs. The different grey colours mark complete, current, executable, and waiting grid jobs. New plans caused by new application jobs, the outage of resources, or delays are updated dynamically.

4 Benchmarks

Above all, the test environment is intended to facilitate the execution of benchmark runs for GORBA, the objective being to further develop the algorithms to improve performance and obtain better solutions. By means of synthetic benchmarks, characteristic properties and diversity can be influenced specifically. Two parameters were defined for the benchmarks to describe their characteristics. One parameter is the mutual dependence of the grid jobs, the other determines the degree of freedom in the selection of resources [6].

For the experiments, four benchmark groups were set up, with a small and large degree of freedom for resource selection (*sR* and *lR*) combined with a small and large degree of dependence (*sD* and *lD*). Each of these groups comprised three benchmarks with 50, 100, and 200 grid jobs and 10, 20, and 40 application jobs, respectively, to represent different workloads. Their execution on the same inhomogeneous grid environment was simulated in the test environment.

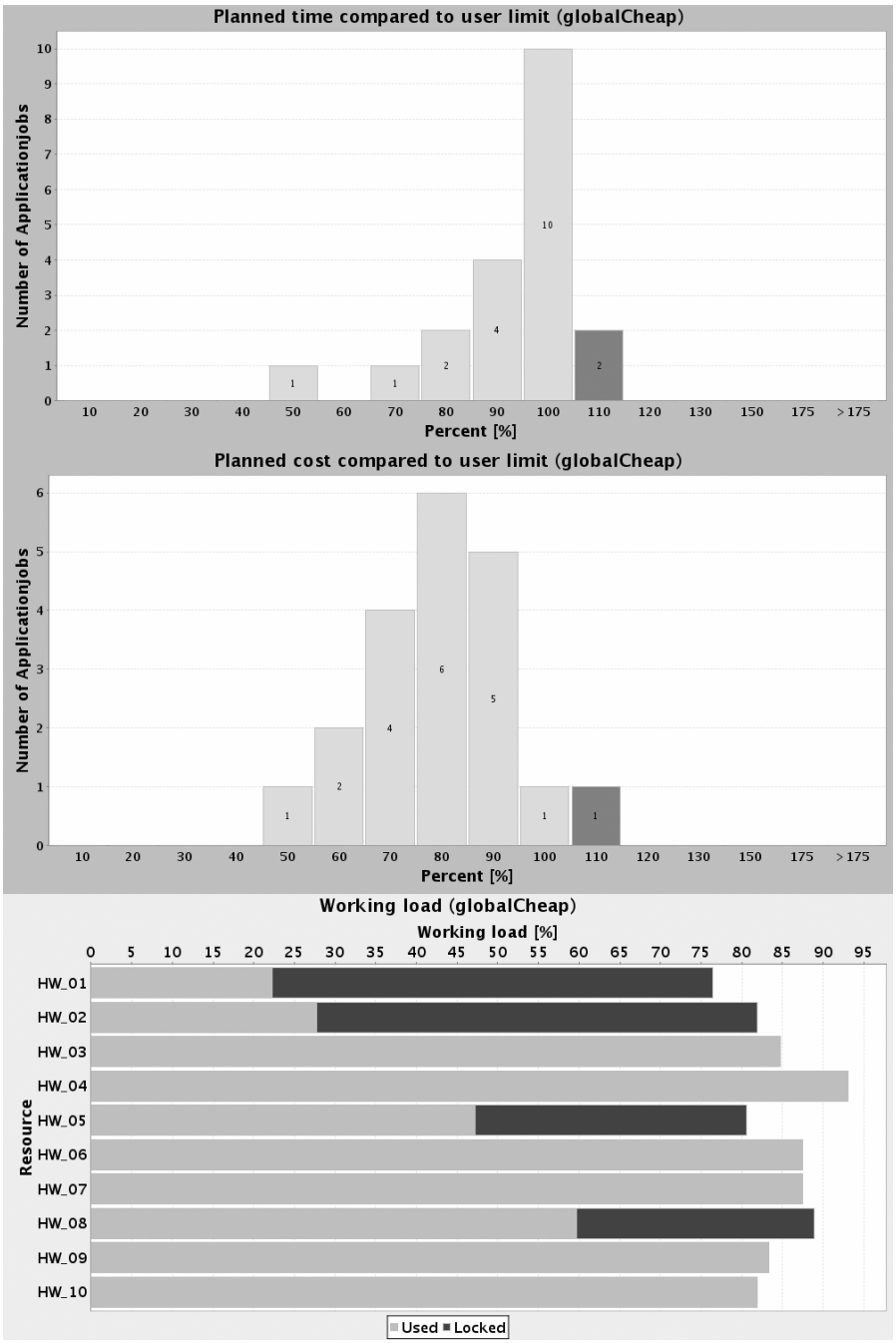

Figure 6: Graphical representation of the exceeding of time limits (top), the exceeding of cost limits (centre), and working load (bottom)

The requirements made on the application jobs with respect to costs and time were chosen in such a way that the heuristic planning phase could not solve them completely, i.e. some application jobs were too costly or too late. This introduces a simple quality measure: can the second evolutionary planning phase overcome these flaws and if so, to which extent? This can be measured easily by the *success rate* indicating the rate of application jobs which fulfil the given cost and time constraints. Each GLEAM run was limited to three minutes, because more time is not expected to be available to come up with a planning solution for practical applications. As GLEAM is, like all EAs, a non-deterministic method by nature, 100 runs were made for each benchmark in order to obtain a reasonable statistic statement. In this case, it is the average success rate.

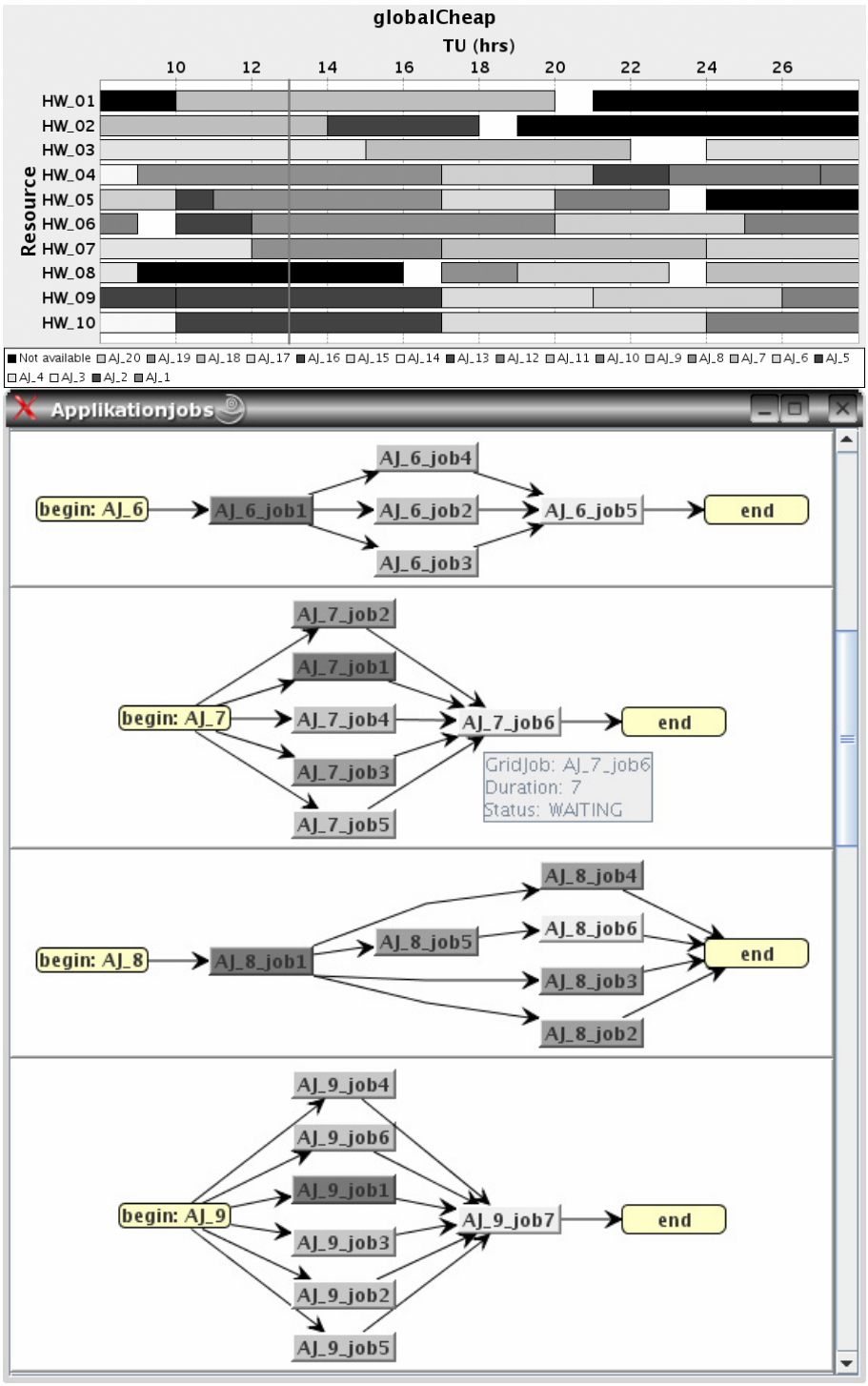

Figure 7: Dynamically updated representation of the execution of a schedule by the job manager

Figure 8 compares the success rates obtained for the two different gene models described in section 2.5. Evolving the resource selection strategy (GM2) in most cases performs equally or better than evolving the resource selection directly (GM1). The reason is a larger search space for GM1, which results in a smaller improvement of the schedule within the given time frame. Other test runs which were allowed to converge showed that GM1 delivers better solutions measured in resource working load and application job cheapness and fastness. This was expected, as GM1 is more flexible in resource allocation. As the decision is made individually for each grid job,

usage of resources which would not have been considered when obeying one of the allocation strategies is possible. But this process requires more time and therefore GM2 is preferred according to the rule that the best plan is useless, if it comes too late. In all cases, including the poor case of *lRsD* for 200 grid jobs of GM1, the schedules from the heuristic phase were improved.

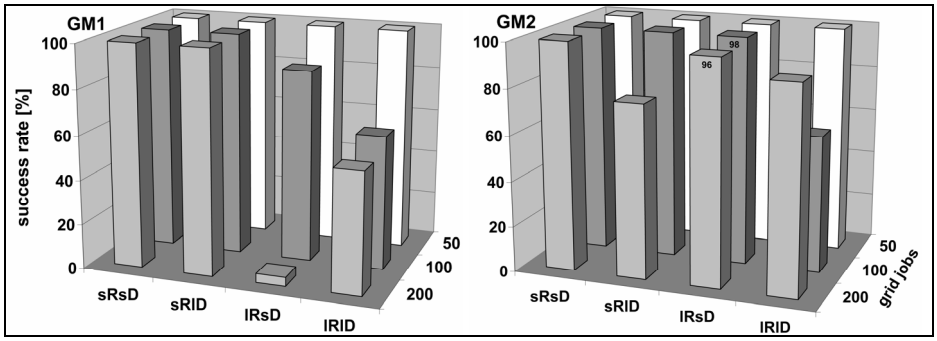

Figure 8: Success rates of the GLEAM planning phase using gene model GM1 on the left and GM2 on the right

5 Outlook

The evaluation of GORBA so far has produced very good results for most benchmark classes. To cover a maximum of applications, however, the optimisation methods implemented in GORBA have to be further developed and extended. Work also focuses on integrating GORBA in CampusGrid. CampusGrid is a project of the Forschungszentrum Karlsruhe [2], which is aimed at virtualising its heterogeneous computing environment. Previous components of the test environment, such as the visualisation options and workflow representation, will be used with GORBA in CampusGrid. New developments mainly refer to the job manager and the resource information services. Apart from the test environment, CampusGrid use will be another, this time real, platform for the further development of GORBA.

References

1. Jakob, W., Quinte, A., Süß, W., Stucky, K.-U.: Optimised Scheduling of Grid Resources Using Hybrid Evolutionary Algorithms. Proc. 6th Int. Conf. on Parallel Processing and Applied Mathematics, Poznan, PL, September, 2005, Springer, LNCS 3911, 2006, pp.406-413.

2. Schmitz, F., Schneider, O.: The CampusGrid test bed at Forschungszentrum Karlsruhe. Sloot, P.M.A. (Ed.), Advances in Grid Computing – EGC 2005, Amsterdam, NL, Feb. 14-16, 2005, Lecture Notes in Computer Science 3470, Springer, Berlin, 2005, pp. 1139-1142.

3. Hovestadt, M., Kao, O., Keller, A., Streit, A.: Scheduling in HPC Resource Management Systems: Queuing vs. Planning. Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP) at GGF8, Seattle, WA, USA, June 24, 2003, LNCS 2862, Springer, Berlin, 2003, pp.1-20.

4. Blume, C., Jakob, W.: GLEAM – An Evolutionary Algorithm for Planning and Control Based on Evolution Strategy. Conf. Proc. GECCO 2002, Vol. Late Breaking Papers, 2002, pp.31-38.

5. Jakob, W.: HyGLEAM – An Approach to Generally Applicable Hybridization of Evolutionary Algorithms. In: Merelo, J.J., et al. (eds.): Proceedings of PPSN VII, LNCS 2439. Springer-Verlag, Berlin, 2002, pp.527–536

6. Quinte, A., Suess, W., Jakob, W., Stucky, K.-U.: Construction of Benchmarks for a Fair Comparison of Grid Resource Brokers. To be published.

7. Hoheisel, A.: Grid Application Definition Language GADL 0.2, Internal report, 18. September 2002.