Project No. FP6-004265

CoreGRID

European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies

Network of Excellence

GRID-based Systems for solving complex problems

D.RMS.05 – Review of the Performance Evaluation and Benchmarking of Grid Scheduling Systems

Due date of deliverable: 31 August 2006
Actual submission date: 14 May 2007

Start date of project: 1 September 2004 Duration: 48 months

Organisation name of lead contractor for this deliverable:
Universität Dortmund/Technical University Delft

Revision: *final*

Project co-funded by the European Commission within the Sixth Framework Programme (2002–2006)		
Dissemination level		
PU	Public	PU

Keyword list: grid scheduling, grid performance evaluation, grid benchmarking

Contents

1 Executive Summary and Introduction

Grids are geographically distributed computational platforms composed of heterogeneous machines that users can access via a single interface, providing common resource-access technology and operational services across widely distributed and dynamic virtual organizations, i.e., institutions or individuals that share resources. This proves beneficial in many situations, for example when applications or user communities require more resources than locally available, or when work needs to be balanced across multiple computing facilities. With the industrial and scientific communities tackling increasingly larger problems, grid computing is becoming a common infrastructural solution, and is becoming capable of replacing traditional computational environments (e.g., parallel production environments) in terms of offered computational power.

Resource Management and Scheduling (RMS), which is the subject of WP6 of CoreGRID, is of paramount importance for the effective and efficient usage of grids. This report on grid benchmarking methods is meant to give an overview of the efforts of CoreGRID partners in WP6 in the research field of RMS. In Section 2 we first present a general impression of the current state of the field of grid evaluation and benchmarking. Section 3 discusses the grid benchmarking approaches of a number of CoreGRID partners. In order to get a more uniform view of the research of CoreGRID partners in the area of RMS, we have conducted a survey with a questionnaire, the text of which is contained in Section 4 and the results of which are included in Section 5. In Section 6 we formulate our conclusions.

This report can be seen as a follow-up and accompaniment of CoreGRID deliverable *Proposal of Multi-level Scheduling Methods* (D.RMS.04).

2 The Field of Grid Evaluation and Benchmarking

2.1 Difficulties in Grid Evaluation and Benchmarking

When compared to other large-scale computational environments (e.g., parallel production environments), grids raise very different challenges for the procedure and the adoption aspects of performance evaluation. First, the existing grid workloads comprise almost exclusively sequential jobs [1] with long duration, unlike both web (services) and parallel production workloads. Second, the grids' dynamic and large-scale nature means that the various existing approaches to tackle the performance evaluation problem in the area of parallel environments [47, 87] cannot be directly applied in grids. Third, a grid environment is composed of many separate components, each with its own performance and reliability issues. In addition to the operating systems, the cluster middleware, and the user middleware, there is an additional, grid-specific, middleware layer that needs to be taken into account.

Other problems also influence negatively research in the area of grid evaluation and benchmarking.

First, the actual adoption of an evaluation procedure as a benchmark is a community approach that requires the agreement of a sufficient number of grid stakeholders. However, the grid technology has evolved with rapid pace over the past decade. This rapid technological evolution has acted as a brake: the community has simply decided to wait until the technology stabilizes, before taking a decision. As a consequence, many of the current performance evaluation procedures, though valuable, lack the typical features of a good candidate for grid benchmarking. Specifically, they do not use realistic workloads [19], or use non-validated measurement data as input for the evaluation process [64], and thus cannot be used for reliable system comparisons and evaluations, cf. [16, 42, 21].

Performance evaluation and comparison require the existence of many workload traces from a variety of grid environments, which at the moment simply do not exist. The Grid Workload Archive effort described in Section 3.1.4 addresses exactly this issue, but is still in its infancy. The main difficulty in obtaining such traces is not only to obtain access to the data, but also that the required data may not exist. In a recent analysis of (incomplete) traces obtained from grid environments [1], the authors observe that these traces log partially or even not at all information regarding the actual job origins (e.g., when users are mapped randomly to pools of usage certificates), the resource consumption (e.g., when the local resource managers do not log the actual CPU, I/O, and bandwidth consumptions, or when information about jobs running across grid sites cannot be correlated), the job coupling/dependencies (e.g., when job batches or jobs belonging to an organization are not recorded as such), or the failures. To ameliorate the lack of grid traces, synthetic, that is, generated on the foundation of an appropriate model, workloads are used for evaluation purposes. The main and, in fact, very hard problem is to create a good model without having enough workload instances (i.e., real system traces). While there exist good models in the parallel processing community, there is no comprehensive workload model available for grids.

2.2 Motivating Scenarios

The common definition and proposed visions for grids go in the direction of a large-scale heterogenous computing platform with varying resource availability. This inherently dynamic and distributed nature is the root of the specific problem of evaluating grids: the sheer size and the dynamic behavior of grids renders difficult the evaluation of their performance. In this context, two questions need to be answered: "What is the actual scenario for which performance evaluation is done?" and 'What kind of performance objectives are sought after?".

Clearly, a single evaluation system will not be able to fulfill all needs. For example, performance evaluation in simulated systems can be done by restricting the environmental description to a few[1] parameters (the number of clusters and of machines, the machine's speed distribution etc.) and allows the analysis of long-term usage as well as non-typical configurations. Simulated systems are, however, restricted to whatever the simulation designer has considered, and their results should not be seen as actual performance values, but more as indicators towards them. In contrast, the use of an actual grid system allows the derivation of current system data on the performance, stability, and usability of a real installation. Still, long-term assessments are inherently difficult, due to the non-exclusive access to the system itself or its configuration. Moreover, the evaluation produces results that are difficult to reproduce, even in the same scenario. To avoid the disadvantages of the previous two scenarios, emulated systems come into place: here, a high-accuracy simulation is done, and performance evaluation is occurs just like in a real environment. This, of course, requires the representation of the simulated infrastructure to match as closely as possible the technical description of the system to be emulated, which leads to a trade-off between the achievable precision and the evaluation speed. Furthermore, the emulated environment needs to run itself on top of a large-scale distributed system. While the theoretically reachable precision of the evaluation results is very high, it is extremely difficult to prove the correctness of the emulation due to the combinatorial explosion of parameter values that can be varied.

We assume that all three approaches, simulation, emulation, and real system testing, are of significance in their domain. Thus, a performance evaluation system should ideally support all of them, and allow a comparison of results on a technical level.

Nevertheless, the applied workload and job models, as well as the underlying grid model, are crucial for the evaluation. It is clear that, in a scenario in which a scheduling and management strategy for grids is quantitatively analyzed, the applied workload and the examined grid configuration are highly dependent. If for instance the requested load extends the system's saturation level, more and more jobs will be queued over time: the wait time for users will increase over time to an unrealistic level, which destabilizes many performance objectives and, in the end, makes the results from such evaluations mostly useless. As a counter example, if the requested load is significantly lower than the saturation level, the scheduling problem degenerates to trivial job dispatching. Due to the strong dependence between a grid configuration and the applied workload, evaluation is very complex, as it is not possible to re-use the same workload for grid configurations which deviate largely in performance. One solution to the problem can be the dynamic adaption of workload

[1] The description of the simulation environment can also influence the evaluation, but this topic is out of the scope of this work.

generation based on the grid performance. However, such an approach has high impact on the performance objectives that can be assessed. We will investigate this problem in more detail in Sections 2.3 and 2.4.

2.3 General Aspects for Workload Modeling

Most research on workload modeling for single HPC parallel computers focus on the characterization of the overall features of workloads. Since the evaluation of scheduling strategies for parallel computers focus on the optimization of a global performance metric, like to minimize the *overall* response time, or the makespan or to increase machine utilization, a general descriptive model is often sufficient for workload modeling [68, 20]. Here, a collection of probabilistic distributions are sometimes suitable for various workload attributes (e.g. runtime, parallelism, I/O, memory). By sampling from the corresponding distributions, a synthetic workload is generated. The construction of such a workload model is done by fitting the global workload attributes to mathematical distributions.

In a grid environment the scheduling objectives depend more on the individual choice of the users. Here, some users may prefer the minimization of cost, while others accept a longer waiting time in favor of a better quality of service, e.g. more or faster processor nodes available. Therefore, a different knowledge of the workload characteristics is needed for a realistic evaluation of grid systems. Unfortunately, there is currently no actual grid workload trace publicly available, such that only assumptions can be made about the actual use of grids. For the time being, it can however be assumed that the current user communities from HPC sites are at the forefront of using grids. Thus, we argue that modeling techniques that have been employed for HPC traces can be (at least partly) applied also in the case of grids, and that existing workload traces taken from parallel computers at HPC sites may be useful as a first start for modeling grid workloads. Within the context of this assumption, the 17 HPC traces from the Parallel Workloads Archive[2] provide valuable modeling data. In this section we present the general aspects of HPC workload modeling.

2.3.1 User Group Model

While it is clear that the individual users' characteristics need to be emphasized in grid environments, the main challenge in the construction of a group and/or user model is to find a trade-off between two extremes: the summarization in a general probability model for all job submissions on the one hand, and unique models which are created for each user based on information about her past actions on the other. We further address the dimensions of the required modeling effort.

We call a set of users or a set of groups dominant if it covers the majority of jobs and is responsible for the majority of consumed resources (from hereon, *squashed area*, or *SA*). When the size of the dominant set of groups or users is reduced, e.g., less than 10, the detailed modeling approach may be used. In [65], the top-5 groups and the top-10 users form dominant sets, respectively, and unique models for each group and user are created. However, this approach does not scale for larger communities, e.g., using hundreds of distributions for different users. In this case, the approach suffers from two significant problems. First, there is usually not enough information available for all users, as some

[2] Available at `http://www.cs.huji.ac.il/labs/parallel/workload/`.

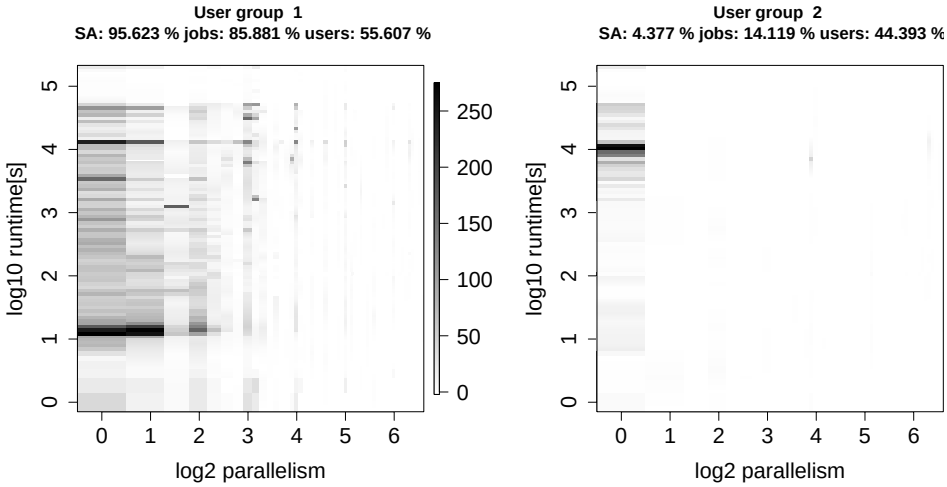

Figure 1: Dominant set of groups of size 2 for the KTH workload.

only have a few job submissions. Second, the overall number of parameters will be quite large so that the interpretability and scalability of the model is lost. As a consequence, a trade-off on the level of user groups is anticipated. That is, users are clustered into groups with similar but distinct submission features. This user group model allows to address the user submission behaviors while maintaining simplicity and manageability.

In [82] such a user group model has been proposed which clusters users into groups according to their job submissions in real workload traces. The analysis showed that for the examined workload, there exists a dominant set of groups of size 4. If the clustering would be even more pronounced, a dominant set of size 2 can be found, with the first group covering more than 95% of the squashed area (see Figure 1).

In the presented research work, the analysis and modeling was restricted to the job parameters *run time* and *number of requested processors* which were sufficient for single parallel computer scheduling. However, modeling on the level of these groups provides the possibility to assign additional workload features, e.g. necessary for grids, to these groups. Some examples of such additionally required features are discussed in Section 2.4.

2.3.2 Submission Patterns

The users of grids have their own habits to request resources and to submit jobs, which is referred to as *patterns*. Here, we take the daily cycle as an example. The daily cycle could refer to the habit of submitting more jobs during day time than night, and to the considerably distinct submission distributions during the day and the night. Figure 2 shows the daily arriving patterns of jobs, for the KTH workload. There is an obvious daily cycle:

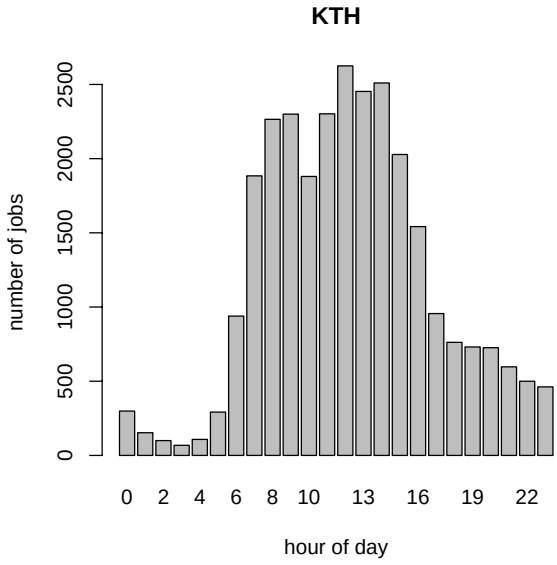

Figure 2: Job arrivals during the daily cycle.

most jobs arrive during the day and only a few of them at night. Obviously, these patterns might blur in grid environments because of users living in different timezones [32]; however, they are still important to the local sites (and the local schedulers).

Similar patterns can be found through the week, e.g., users tend to submit more jobs during the week-days than during the week-end, or year, e.g., an outstanding increase in the number of job submissions may be observed during several months of the year [65], or during short periods [18].

These effects can be described by classical statistical methods. For example, Downey [24] modeled the daily cycle using combined Poisson distributions; Calzarossa [12] found that an eight-degree polynomial function is a suitable representation of all the analyzed arrival processes. However, this does not necessarily hold because of dependencies within the workload (see [38, 33]), e.g. sequential dependencies.

Therefore, temporal modeling is an important aspect. For example, one of these temporal effects is *repeated submission* [38], namely, users do not submit one job once but several similar jobs in a short time frame. Even if the successively appearing jobs are disregarded, temporal relations can still be found, as shown in [33]. It can be seen that the successors of jobs with a large parallelism value also tend to require more nodes.

Such temporal characteristics are useful for the many grid scheduling scenarios, including resource reservation and load balancing. The application of various techniques, e.g.,

stationary and non-stationary analysis as well as stochastic processes, provides a good representation of the temporal relations in users' submissions. In [81], correlated Markov chains are used to model the temporal characteristics of job sequences. The idea to correlate the Markov chains is that since the job parameters are correlated, the transformations of their corresponding Markov chains are related as well. In [71], a model based on Markov chains is used for the number of jobs consecutively submitted by a user during a single submitting session.

Besides that, analysis has shown that users also tend to adapt to the performance of the available system. That is, users may change their job submissions according to the available online information, e.g. system states and quality of services as shown in [39, 40]. Therefore, it is reasonable to model the users' submissions with the considerations of such feedback behaviors. Thus, the workload generation should be coupled to the system with a feedback loop.

In many cases, the explicit feedback tags are missing; therefore it is not feasible to determine whether feedback factors do affect job delivery. For example, if a user seldom delivers jobs at noon, it might result from a regular lunch at this time, or has a real feedback implication: the user finds many waiting jobs at noon and then stops his or her submissions.

However, it is possible to elicit whether feedback factors affect a job's profile (like parallelism and runtime), since the job profiles can be compared along different situations of influential factors. To this end, the correlations between the feedback factors and the job attributes should be analyzed.

2.4 Grid-specific Workload Modeling

In this section we present the grid-specific workload modeling requirements. Due to the lack of publicly available traces[3] of real grids operation, we restrict our presentation to the main characteristics that could become subject of near-future modeling.

2.4.1 Types of Applications

Grid jobs may have a complex structure which may be handled only with advanced support from the submission middleware. From this point of view, we consider two types of applications that can run in grids, and may be included in generated grid workloads: i. *unitary applications*, which include sequential or parallel (e.g. MPI) applications and at most require taking the job programming model into account (launching additional naming or registry services, for example) and ii. *composite applications*, which include bags, chains or DAGs of tasks and additionally require special attention by the grid scheduler regarding inter-dependencies, advanced reservation and extended fault tolerance.

Note, in the remainder of this section we use the term application at some points. By this we understand a certain user problem that has to be calculated. In this sense, application and job are the same.

[3]There is, of course, one public trace of a HPC site participating in the EGEE/LCG production grid; however, due to the fact that only the batch system log is available, but no information whatsoever on the grid infrastructure layer, this workload degenerates to a standard HPC site trace.

2.4.2 Computation Management

Another grid-specific problem is the *processor co-allocation*, that is, the simultaneous allocation of processors located at different grid sites to single applications which consist of multiple components. Co-allocation models need to describe the application components and the possible resource co-allocation policies. To model the application components, we need to define the number of components (NoC) and the component size (CS), and furthermore must allow multiple configurations, such that sets of (Noc, CS) tuples or even ranges can be then specified. In practice, the typical configurations for processor co-allocations are selected such that they fill completely clusters of resources, to keep the inter-cluster communication low [9]; load-balancing across the number of sites can also be used for jobs requiring large numbers of resources [8]. Obviously, there are three possible resource co-allocation policies: 1. *fixed*, where each job has predefined resource preferences; 2. *non-fixed*, where jobs have no resource preferences at all and 3. *semi-fixed*, where only some job components require certain resources, whilst others can be dispatched at the scheduler's discretion. Experience with co-allocation in a real environment is described in [73, 57]. However, no statistical data regarding the use of co-allocation by real communities of users is publicly available.

In addition, *job flexibility*, that is, the (in)ability of a job to run on a changing number of resources, raises many more problems in grids than in traditional parallel environments. Flexibility models need to describe the flexibility type and (possibly) the number and dynamics of computing resources that may be allocated to a job. There are four possible flexibility types: rigid, moldable, evolving, and malleable [43]. To model the application flexibility, at least one job shape (cf. [21], a tuple comprising the minimum and maximum number of computing resources, the configuration constraints, e.g., n^2 processors, and the resource addition / subtraction constraints) must be defined per job. Statistical data for moldable jobs for a real supercomputing community is given by Cirne and Berman [21]; experiments with moldable applications in a real environment have been presented by Berman et al. [5].

Finally, one has to consider that, in production grid environments, there often exists a certain *background load*: many processing resources are shared with the grid by a local community, and may have local jobs running outside the grid resource management. Also, it is expected that usage of resources must differ greatly depending on the *project stage* of a certain user community which generates the usage. Considering a long-term project, there might be a startup and a transition phase, in which infrastructure and application test are produced, an execution phase, which contains the main body of work for the project, and an ending phase, in which jobs with characteristics and submission patterns totally different from the previous stages might appear. From such a projects' point of view, the modeler needs to be able to characterize each individual stage.

2.4.3 Data Management

We now discuss the modeling requirements of data management. Grid jobs need input data for processing, and generate output data to make the computed results persistent. The data needs to be present before[4] the job can start, and the stored results must be fetched after

[4]There also exist I/O models that introduce remote data access with read-ahead and/or caching mechanisms, but these are out of the scope of this work.

the job has finished, or streamed as they are produced. Hence, the modeler needs to specify at least the identifiers of the input and of the output files. For composite applications (see Section 2.4.1), it is also necessary to specify data dependencies between the jobs, that is, which of a job's output files represent another's input, and which input files are shared by several jobs.

Similarly to specifying an estimated computation time or size for their applications, it would be desirable that users specify an estimation of the needed input and output space within the job description. Also similarly to the estimated/actual runtime discrepancies, the information specified by the user may not be reliable and available, e.g., the user provides imprecise estimations or the job determines result data sets during runtime. We argue that such information can be added easily, as many applications have well-studied I/O requirements, both when running individually, or when running in batches [83].

For many applications, data is obtained from remote large-scale storage devices, usually with very different access times than the locally available data. Additionally, unexpected difficulties can occur regarding the access time for files which appear to be locally available, i.e., files might seem to be accessible on a local filesystem but essentially have been moved to tertiary storage. This is especially the case for HSM[5] systems, where the restoration of files can take a long time. In this case, a model should provide detailed information about the source and destination storage, for the input and output files, respectively.

Sometimes, the same file is replicated on several storage sites in the grid. Modeling this aspect can be reduced to specifying lists of input and/or output files locations for each unique file identifier. Note that the information in the list may need to be combined with information on the data storage device.

2.4.4 Network Management

When introducing data management into workload models for grids, it is obvious that also networks between sites have to be considered. The available bandwidth for transfers of input or output data is limited and thus has an impact on the data transfer time. This can influence the decision which site is used for a certain computation and whether data has to be replicated to this site in order to run the job. There are also other application scenarios in which network management is a critical feature, like the management of bandwidth in Service Level Agreements between remote resource allocations, e.g. for parallel computation, large-scale visualization, or consistent data streaming from experimental devices [45]. Therefore, the end-to-end bandwidth between different nodes in the grid must be described and managed.

Ideally, the total bandwidth of every end-to-end connection would be known and dedicated reservations could be enforced. However, this is often not supported: in IP networks, end-to-end connections are virtual (since the packet route can change) with a maximum weakest-link-in-chain bandwidth. Hence, in many realistic scenarios often no precise information about the service quality between two ends is available. However, there are means which can ameliorate this situation. For example, the NWS system [89] measures and records the available bandwidth between two nodes periodically. This data is then used to predict the expected average bandwidth in the future based on historic patterns. In cases

[5]Hierarchical Storage Management.

where reservation of bandwidth is not feasible, there are still possibilities to shape network traffic. However, abiding agreements on an certain QoS level cannot be settled normally. Regarding network latency, which is important for applications requiring large numbers of small network packets (e.g. streaming), the situation is akin.

Besides that, there is always a certain amount of background (not grid workload-related, that is) traffic on a network, which lowers both bandwidth and latency. However, due to the lack of reservation capabilities, the impact of background traffic is not predictable at all. Even when predictions expect high network availability and the known future utilization is low, a single data-intensive file transfer can suddenly produce a high, previously unexpected network load.

On the whole, realistic network depiction in workload models is difficult and will have to be subject to further research; first steps into the direction of grid-specific data staging and network modeling have been taken in the SDSC HPSS work [78] and Tan et. al [84]. However, it would be useful that a grid performance evaluation system provides support for network resources and consequently for network related workload requirements in order to have a testing platform for future models. Such an extension would include the addition of network information and requirements to jobs as well as evaluated grid configurations on which the workload is executed.

2.4.5 Locality/Origin Management

Another requirement for some evaluation scenarios in grids is the realistic modeling of the origin of job submissions. While some may argue that grid workload is created decentralized and on a global scale, many usage scenarios still need information about the locality of job creation. A typical example of such a scenario is the collaboration between HPC centers which want to share their workload to improve quality of service to their local user community. While the sites may agree on sharing the workload, it is quite common that certain policies or rules exist for this sharing (balancing between local and "foreign" users, for instance).

Other examples can be conceived, in which the submitting user plays a role, as he may belong to a certain virtual organization and may have subsequently special privileges on certain grid resources [58]. Support for these scenarios can be helpful for P2P grids, where resource access is mostly user-centric and not dependent on a particular site policy.

2.4.6 Failure Modeling

Due to the natural heterogeneity of grids and their sheer size, failures appear much more often than in traditional parallel and distributed environments, occur at infrastructure, middleware, application, and user levels, and may be transient or permanent. Furthermore, different fault tolerance mechanisms can be selected by the user, for each job or workflow in the workload. Hence, the modeler must use a dynamic grid model, combined with a realistic failure model. Then, she must take into account the fault tolerance mechanisms, i.e., the failure response mechanism selected by the user or employed automatically by the system. Furthermore, experiments such as comparing two middleware solutions may require deterministic failure behavior.

Failures in the grid infrastructure may be caused by resource failures (e.g. node crashes), or by other resource outages (e.g. maintenance). To model resource failures, the traditional availability metric, *mean-time to failure* [59], the length of failure (*failover duration*), and the percentage affected from the resource, must be specified for each resource. To model other resource outages, the following parameters must be specified: outage period, announced and real outage duration, percentage affected from the resource affected, and (optional) the details of the failures, e.g., which resources or resource parts did fail [16].

Failures in the grid middleware may have various causes. One source of errors is the scalability of the middleware; another is due to the middleware configuration: according to the survey in Medeiros et al. [70], 76% of the observed defects are due to configuration problems. For modeling purposes, starting points could be static mechanisms like mean-time to failure, and the length of the failures, again.

Regarding *failures in grid applications*, it has been observed by Kondo et al. [64] that jobs submitted during the weekend are much more error-prone. Therefore, an application failure model should contain a *fault inter-arrival time* distribution.

User-generated failures can be modeled similarly to the distribution of the jobs' inter-arrival time. Faults due to user-specified job runtimes have been a topic of interest in parallel workload modeling, other issues like missing or expired credentials and disk quota overrun [19, 26, 55], invalid job specifications [63] or user-initiated cancellations [20] are other sources of user-generated failures.

To respond to the numerous sources of failure, various *fault tolerance schemes* may be applied in grid, and possibly need to be modeled (see for example [53]); the technique type then needs to be specified for each job or workflow in the workload, coupled with the specific technique parameters.

2.4.7 Economic Models

There is a lot of discussion on the connotation of access to grid resources not being free of charge [60, 28, 10]. Especially the support for commercial business models will include support for economic methods in grids. Therefore, it is clear that the allocation of jobs to resources may incur cost in certain grid scenarios. Adopting cost has many implications to the allocation of jobs to grid resources: providers will require the implementation of pricing policies for the access to resources. To the same extend, users will need support for managing budgets for job executions and preference constraints on how jobs should be executed (e.g., price vs. performance). First economic models have been published by Ernemann et al. [28] and Buyya et al. [10].

While it is not the task of an evaluation system to tackle the technical implications of economic models, like whether cost occurs in virtual credits or actual money, it can be conceived that there are requirements to model budget information for either jobs, users or virtual organizations. This is even necessary if grids are modeled in which users or groups have a certain quota on resources; a precondition to optionally support budget constraints in the evaluation system.

Another step could be the support for different optimization goals that are economy-related. For instance, users may prefer a cheaper (in terms of cost) execution of a job in contrast to an early execution. This, however, requires the extension of the performance metrics to include cost-related parameters, in a possibly parametric fashion. For example,

in Ernemann et al. [28], grid users provide parametric utility functions, and the systems performs automated request-offer matching.

3 Grid Scheduling Performance Evaluation and Benchmarking Approaches

In this section we present an overview of the grid benchmarking and performance evaluation work of a number of CoreGRID partners in the Virtual Institute on Resource Management and Scheduling. Each of the subsections of this section details the work performed at an individual partner.

3.1 Co-allocation and the Design of a Grid Scheduler (TUD)

In the Parallel and Distributed Systems group at Delft University of Technology (TUD), the research in grid benchmarking and evaluation consists of the following four elements:

1. Simulations of co-allocation policies;
2. Experimentation with the co-allocating grid scheduler KOALA developed at TUD;
3. The development of the GrenchMark benchmarking tool;
4. And the development of a Grid Workloads Archive.

We will now elaborate on each of these four topics in turn.

3.1.1 Simulations of Co-Allocation Policies

Over the last decade, clusters and distributed-memory multiprocessors consisting of hundreds or thousands of standard CPUs have become very popular. Compared to single-cluster systems, multicluster systems made up of multiple, geographically distributed clusters can provide a larger computational power. Instead of smaller groups of users with exclusive access to their single clusters, larger groups of users can share the multicluster. This potentially leads to lower job turn-around times and higher system utilizations, and makes larger job sizes (in terms of the number of processors) possible by allowing jobs to use processors in multiple clusters simultaneously, that is, to employ (*processor*) *co-allocation*. Of course, co-allocation entails wide-area communication, which may increase application runtimes and so may reduce the potential benefits of co-allocation. We have designed and studied with simulations the performance of scheduling policies for processor co-allocation in multicluster systems, on which we have reported in a number of papers, see, e.g., [6, 7, 4]. Among the parameters we varied in these simulations are the distributions of the number and sizes of the components of jobs (e.g., the parts of jobs that are executed in separate clusters), the job service times, whether it is up to the scheduler toplace job components or whether the users prescribes specific clusters, and the cluster sizes. The performance metrics we use are the mean response time and the maximal utilization.

An example of a multicluster system is the Distributed ASCI Supercomputer (DAS) [22, 3], which is an important motivation for the work at TUD and which is currently in its

third generation. In particular, on the DAS the feasibility of running parallel applications across wide-area systems has been amply demonstrated [2, 61, 76, 4]. Of course, also in computational and data grids [48, 44], co-allocation can be employed, for more types of resources than only processors. In the most general setting, grids and grid resources are very heterogeneous; we restrict ourselves at TUD to homogeneous multiclusters such as the DAS and to processor co-allocation. Showing the viability of processor co-allocation in such systems may be regarded as a first step in assessing the benefit of co-allocation of various types of resources in more general grid environments.

Our two main conclusions are that processor co-allocation may be beneficial, at least when the overhead due to the wide-area communication is not too high, and that the restrictions to job-component sizes and the number of job components markedly improve the performance of co-allocation.

3.1.2 Experimentation with the Koala Grid Scheduler

Resource management in grids is performed by grid schedulers, which operate at a higher level compared to local (multiprocessor or cluster) schedulers. There are important differences between grid schedulers and local schedulers that complicates co-allocation of resources even more. These differences include:

1. Grid schedulers do not own resources themselves, and therefore do not have control over them; they have to interface to information services about resource availability, and to local schedulers to schedule jobs.

2. Grid schedulers do not have a full control over the entire set of jobs in a grid; local jobs and jobs submitted by multiple grid schedulers have to co-exist in a grid.

3. The set of available resources in grids is highly dynamic, and resources may come and go, either by being disconnected or by failing.

We have designed and implemented a co-allocating grid scheduler called KOALA, which is aimed at addressing the co-allocation problem presented above, and which has been operational in the DAS system since september 2005. The co-allocation problem is addressed through a set of policies built in KOALA; new policies can be added or modified without affecting KOALA operation. To allocate jobs across multiple sites, two placement policies are used: the Close-to-Files (CF) policy and the Worst Fit (WF) policy. CF addresses the problem caused by co-allocation of delaying the start of job execution because of long input file transfers. The file transfers are minimized by selecting execution sites for job components (subtasks of a job) on or close to sites where their input files are located. On the other hand, the WF policy simply places job components on sites with the largest numbers of idle processors. In doing so, WF through co-allocation, balances the number of idle processors across the grid. The placement policies extensively use KOALA Information Service to locate and monitor resources availability. In the context of a CoreGRID fellowship with the University of Muenster, we have adapted KOALA to be able to accept the submission of, and to schedule efficiently, so-called Higher Order Component (HOC) jobs [25, 27].

To guarantee the simultaneous availability of processors at a start of a job execution, the Incremental Claiming Policy (ICP) is used in the absence of support of advance processor

reservation by local resource managers. ICP is used because processors may not to be available anymore near the job start time. In such scenario and without delaying job start time, ICP tries to make processors available for job components by finding processors at other sites or, if permitted, by forcing processor availability through preemption of running jobs. Also, KOALA deals with the dynamicity of the grid resources and reliability problems of some grid components through its fault tolerance mechanisms to ensure that grid jobs are completed successfully. The major contributions of the work on KOALA are: 1) a reliable co-allocating grid scheduler, 2) a grid scheduler that brings co-allocation to different application types, 3) co-allocation policies, and 4) an alternative mechanism to advance processor reservation when absent in local resource managers. More details on KOALA and its policies can be found in [72, 74].

3.1.3 The GrenchMark

The current evolution of grids hinges on proving that they can run real applications, from traditional sequential and parallel applications to new, grid-only, applications. As a consequence, there is a clear need for generating and running workloads comprising grid applications for demonstration and testing purposes. To address these problems, we have built GrenchMark, a framework for synthetic workload generation and submission. With GrenchMark[6] , we try to find a common ground for grid performance analysis, and to offer the performance-oriented Grid community a tool that can help to bring together Grid performance evaluation approaches, towards the goal of building standard Grid benchmarks.

The GrenchMark package is *extensible*, in that it allows new types of grid applications to be included in the workload generation, *parameterizable*, as it allows the user to parameterize the workloads generation and submission, and *portable*, as its reference implementation is written in `Python`. GrenchMark uses a comprehensive workload model, that comprises notably composite jobs (e.g., batches, DAG-based), and co-allocation, that is, the simultaneous allocation of resources located in different grid sites to single applications which consist of multiple components. For a complete description of the workload model available through GrenchMark, we refer to [54, 56, 57]. GrenchMark has been used in a variety of scenarios [57], to run over 200,000 test jobs, for: application performance analysis, *what-if* analysis, functionality testing and demonstration, periodic system testing, stress-testing, grid settings comparison, and development and testing of peer-to-peer protocols.

3.1.4 The Grid Workloads Archive

There is an ever increasing gap between the grid research results, which show how grids can support larger and more diverse workloads, and the demand of real users, of which very little is known. This crucial lack of knowledge hampers not only the development of new solutions, but also prevents the evaluation of existing grid middleware, and limits the finding of relevant research directions. To bridge this gap, we have started the Grid Workloads Archive (GWA) project, at the same time a workload data exchange and a community center for the grid resource management and scheduling scientists[7].

[6] GrenchMark. [Online] `http://grenchmark.st.ewi.tudelft.nl/`. Feb 2007.
[7] The Grid Workloads Archive. [Online] `http://gwa.ewi.tudelft.nl`. Feb 2007.

In the flavor of the Parallel Workloads Archive[8], The GWA collects grid workloads (traces) from various contributors, and presents them to the public in a standard format. The data is anonymized before publication, that is, no user, site, cluster, or executable names present in the GWA will be identical to the originals. Each trace is accompanied by a clear identification of the contributor, and data copyright/sharing information. We are currently on the process of publishing traces from the CERN LCG, the DAS, the GLOW, Grid3, Grid'5000, the NorduGrid, and the TeraGrid, which we have recently analyzed [1].

The GWA offers various tools to parse and to analyze trace data, from start-up scripts to analysis tools of moderate and high complexity. There GWA also offers various community tools: a Wiki/forum environment, trace issues reporting and tracking, and a mailing list.

3.2 Performance Evaluation and Prediction (CNRS)

An important issue for the research on complex systems such as grids is to validate the obtained results. This validation constitutes a scientific challenge by itself since we have to validate models, their adequation to reality *and* the algorithms that we design inside these models. Whereas mathematical proofs establish soundness *within* such a context, the overall validation must be done by experiments. A successful experiment shows the validity of both the algorithm and the modeling at the same time. But, if the algorithm does not provide the expected result or performance, this might be due to several factors: a faulty modeling, a weak design, or a bad implementation.

In addition to this idea of validating the whole (modeling, design and implementation) in our research we are often restricted by a lack of knowledge: the systems that we want to describe might be too complex; some components or aspects might be unknown or the theoretical investigations might not yet be sufficiently advanced to allow for provable satisfactory solutions of problems.

We think that an experimental validation is a valuable completion of theoretical results on protocol and algorithm behavior.

The focus of algorithmic research being upon performance, the main experiments we are concerned with are performance evaluation. To our opinion, such experiments should fulfill the following properties:

reproducibility: Experimental settings must be designed and described such that they are reproducible by others and must give the same result with the same input. A brief look into the literature shows that for performance experiments in our scientific community this is not as straightforward and by no means obvious.

extensibility: A report on a scientific experiment concerning performance of an implementation on a particular system is only of marginal interest if it is simply descriptive and does not point beyond the particular setting in which it is performed. Therefore, the design of an experiment *must* target possible comparisons with passed and (in particular) future work, extensions to more and other processors, larger data sets, different architectures and alike. Several dimensions have to be taken into account: scalability, portability, prediction and realism.

[8]The Parallel Workload Archives makes various workload traces from real parallel production environments available at `http://www.cs.huji.ac.il/labs/parallel/workload/`.

applicability: Performance evaluation should not be a goal *in fine* but should result in concrete predictions of the behavior of programs in the real world. However, as the set of parameters and conditions is potentially infinite, a good experiment campaign must define realistic parameters for platforms, data sets, programs, applications, *etc.* and must allow for an easy calibration.

revisability: When an implementation does not perform as expected, it should be possible to identify the reasons, be they caused by the modeling, the algorithmic design, the particular implementation and/or the experimental environment. Methodologies that help to explain mispredictions and to indicate improvements have to be developed.

Experimental validation of grid systems is a particularly challenging issue. Such systems will be large, rapidly changing, shared and severely protected. Naive experiments on real platforms will usually not be reproducible, while the extensibility and applicability of simulations and emulations will be very difficult to achieve.

These difficulties imply that the study phases through modeling, algorithm design, implementation, tests and experiments. The test results will reveal precious for a subsequent modeling phase, complementing the process into a feedback loop.

As we already have emphasized above, we believe that an experimental component for research in computer science and especially in the case of distributed and grid systems is crucial. We have to validate all components of a given setting (modeling, design, implementation) despite the imprecision about the elements and their interactions, which hinder complete predictions. In addition, the dynamics of the target platforms prevents reproducible experiments and thus makes algorithm comparisons very difficult.

As a result, several experimental paradigms are used in the community depending on the desired level of realism and the accepted amount of efforts to setup the environment. *Simulations* are rather easy to setup, but the results sometimes lack of accuracy and applicability. On the other hand, *in situ* experimentations naturally alleviate model validation issues, but require inordinate amount of time and energy to establish stable development and evaluation environments. *Emulation* solutions aim at constituting a tradeoff between the two extremes, being easier to setup and use than real platform while providing more accurate information than simulations.

Concerning simulation, we use the SimGRID toolkit which enables the simulation of distributed applications in distributed computing environments for the purpose of both algorithmic studies and software development.

Concerning emulation, we work in the ACI *Masse de Données* GRID-Explorer at designing an environment to emulate heterogeneity. GRID-Explorer is indeed an homogeneous cluster designed to perform grid experiment. As distributed systems are mainly heterogeneous it is required to transform this homogeneous environment into an heterogeneous one. Our tool, called Wrekavoc [14], is doing that by degrading the performance of nodes and the network. We target mainly the CPU power, the available memory, the network bandwidth and latency. The scientific issues are how to control the degradation, how scalable is the proposed solution, and how the emulated environment compared with regard to a real one. With Wrekavoc we have a tool that:

- helps in testing algorithms designed for heterogeneous environments,

- controls the heterogeneity,
- provides reproducible experiments,
- allows quantitative comparison of algorithms and real applications.

Concerning real-world platform and *in situ* experimentations, we are involved in GRID5000 [15] which aims at designing a distributed instrument for grid experiments. More precisely, nine sites spread across France (Lille, Rennes, Orsay, Bordeaux, Lyon, Grenoble, Toulouse, Nice and lastly Nancy), are hosting clusters linked together by a high-speed backbone. These nine clusters form a grid on which reproducible experiments can be conducted. We are responsible of the Nancy site for acquiring, maintaining and providing to the community the cluster and the experimental environment. With GRID'5000 we have a platform for experiments in real life condition:

- Address critical issues of Grid system/middleware: programming, scalability, fault tolerance, scheduling, etc.
- Address critical issues of Grid Networking: High performance transport protocols, QOS
- Port and test applications.
- Investigate original mechanisms: P2P resources discovery, Desktop Grids.

3.3 Evaluation Methodology in Grid Scheduling (UniDo)

The IRF at the University of Dortmund has worked on different topics in scheduling and especially in Grid resource management and scheduling. Earlier work included theoretical analysis of online and offline job scheduling which was based on worst-case analysis and competitive factors for machine utilization, makespan, response time and weighted response time criteria [79, 80].

As theoretical analysis is difficult to apply for Grid scenarios, extensive work has been done on different aspects in Grid scheduling by using simulations [77]. To this end, several simulations environments had been created to facilitate the evaluation. All of these simulation tools used discrete event-based simulations. Areas of research interest are machine utilization, average weighted response time for mixed workloads with parallel applications in an online scenario.

The evaluation was conducted by the use of different set of workloads that had been adapted for the various specific research aspects like cost models, communication overhead, dependencies etc. [30, 29] For assuring realistic workload setting, most workloads had been derived from existing real life workload traces, most of which are available through the standard workload archive by Feitelson. To this end, workloads have been used that had been directly derived from the traces, as well as newly created workloads based on the statistical features of existing workloads. For reference purposes, some workloads included pure randomly generated profiles.

The simulated Grid configurations had been defined according to the size of the real machine configurations that were the source for the workload traces. However, to provide

comparability between different workloads as well as to allow larger Grid configurations to be simulated, we developed strategies to scale workloads which included the duplication of jobs from traces as well as the transformation of existing jobs to occupy larger node sets [34]. These operations were analyzed to provide realistic statistical features to include jobs with larger requirements as well as more job submissions.

For better understanding and modeling user behavior and their corresponding workload creation, the IRF worked on the analysis of existing workload traces. This included the analysis for dependencies between jobs within workloads, the submission pattern of individual users as well as the identification of user groups with similar behavior [82, 81]. Based on this analysis, methods have been provided to support better modeling of the submission behavior of these user groups. These approaches included Markov-chains to model the different submission processes.

Recent research included e.g.: negotiation models for market-like Grid environments [35, 31, 67, 66], multi-criteria optimization, inclusion of data requirements and management. Accordingly the workloads are enriched by additional information to support the evaluation in these research areas. However, as existing machines do not provide many of the necessary aspects and consequently there are no workload traces that include information about cost, utility functions, data requirements. Consequently, the workload extensions are currently modeled based on first assumptions that will need to be verified and adapted when more information from real environments are available. As none of the currently available simulation framework allows the consideration of all of these research topics, dedicated simulation tools have been and are developed at the IRF.

3.4 Grid scheduling in GridSim (MU)

3.4.1 Introduction

In the research at MU we want to design and investigate advanced scheduling techniques for planning various types of jobs in Grid environment [85]. Our solution should deal with common problems of job scheduling in Grids like heterogeneity of jobs and resources, dynamic runtime changes such as new users and jobs arriving or various failures in Grid system.

We have already developed a centralised Grid scheduler [62] which uses advanced scheduling techniques for schedule generation. We focused on local search based algorithms [49] and some dispatching rules [75]. This scheduler is implemented in GridSim [11] simulation toolkit. GridSim is Java based simulation toolkit which allows to simulate the topology of the Grid environment with entities such as users, resources, schedulers etc.

The scheduler was tested in both static and dynamic situation. Static situation means that all jobs are known in advance while dynamic situation means that jobs appear in the system during simulation so the generated schedule is changing through time as some jobs are already finished while the new ones are arriving.

Up to now we have developed an experimental Grid environment with simple Grid scheduler using GridSim simulation toolkit. The simulated Grid environment consists of Grid users, computational resources and centralised scheduler. This scheduler uses simple dispatching rules to create initial schedule and then does some optimalization using local search based algorithms both in static and dynamic situation.

3.4.2 Scheduler description

The scheduler communicates with Grid users by message passing. Users send descriptions of their jobs to the scheduler which creates schedule according to these information. In dynamic situation this schedule may change in time as some jobs are already finished while new ones appear.

We use various dispatching rules for initial schedule generation according to the type of job and corresponding optimalization criteria. Local search based algorithms are then used for schedule optimalization. Scheduler now works with three types of jobs.

- Simple jobs, only computational length is set with no other requirements or constraints. The scheduler optimises makespan. Initial schedule is generated by EST (Earliest Start Time) dispatching rule. Further optimalization can be done by hill climb search, tabu search or simulated annealing.
- Jobs with release date and due date, scheduler optimises total tardiness. Initial schedule is generated by ERD (Earliest Release Date) or EDD (Earliest Due Date) dispatching rules. Further optimalization can be done by hill climb search.
- Simple workflows with linear (chain) structure of subtasks, scheduler optimises makespan. Initial schedule is generated by EST dispatching rule. Further optimalization can be done by hill climb search respecting the precedencies.

3.4.3 Experimental results

The scheduler was tested in both static and dynamic situations. Not only makespan or total tardiness were measured but also the time necessary for schedule generation. The results showed that local search based algorithms significantly improve the initial schedule but are much more time consuming in contrast to dispatching rules.

3.4.4 Future work

As a part of the CoreGRID project, we would like to explore scheduling algorithms for jobs with specified due dates to be scheduled on cluster of computers. We would like to compare and extend the solution techniques and the scheduling models developed at MU with the work of Ranieri Baraglia from ISTI-CNR who also concentrate on solution of this type of problems.

We want to extend our current work at MU to make existing Grid model and Grid scheduler more complex with additional functionality. We plan to use GridSim 4.0 and include its network simulation capability into our model.

The scheduler should manage various types of complex jobs such as parallel applications, DAG workflows, preemptive and priority jobs, etc. Other very important functionality of the scheduler should be fault tolerance i.e., detection and proper reaction to errors appearing in Grid system e.g., resource failures, network failures or job losses.

3.5 The Grid Scheduling Simulator (PSNC)

The Grid Scheduling Simulator (GSSIM) has been developed by the Poznan Supercomputing and Networking Center (PSNC) on the basis of the well known GridSim and SimJava

toolkits. The main motivation behind GSSIM is to provide generic, flexible, and highly customizable Grid simulation framework. The goal of this framework is to allow researchers and developers to easily test various scheduling algorithms at different levels of the whole Grid scheduling architecture. In particular, GSSIM enables flexible definition of scheduling policies for both, Grid resource brokers and local resource providers. GSSIM design and set of defined generic interfaces help to move algorithms between the simulator and real systems minimizing number of required source code modifications.

Going into more detail, GSSIM added value, compared to GridSim consists of the following features:

- Workload generation management
- Resources generation management
- Grid resource broker interface and plugins
- Grid resource provider interface and plugins
- Execution time calculation interface and plugins

Additionally, some errors and missing issues were implemented and added to GridSim. The GSSIM main concept and its relation to GridSim and SimJava is illustrated in Figure 3.

Workload and resource generation. GSSIM enables a high customization of a workload. In the configuration file (in XML format) users can define basic parameters of the workload and specify probabilistic properties such as distribution of intervals between jobs, jobs' execution time, resource requirements etc. Generated workloads support the format of Feitelson's Parallel Workloads Archive but in order to take more complex parameters into account they also contain XML-based job descriptions. A language (XML schema) used to describe jobs can be replaced by other if needed. Currently, the GRMS job description schema [50] is used for that purpose. Generated workload can be used many times to ensure that input data is the same in many experiments. The similar mechanism is used for resources, however, currently XML resource descriptions are defined by hand.

Plugin mechanism. Another crucial added value of GSSIM is possibility of plugging of various scheduling algorithms. This functionality is available for three components.

Every implementation of algorithm at the Grid resource broker level must implement the *Grid Scheduling Interface*. It contains one major method: *scheduleJobs*. Knowledge that may be needed by the scheduling algorithm consists of the following elements:

- Job descriptions (resource requirements, data, workflow)
- User preferences
- Reservation requests
- Resource parameters (static and dynamic)
- Reservation offers
- Predicted times

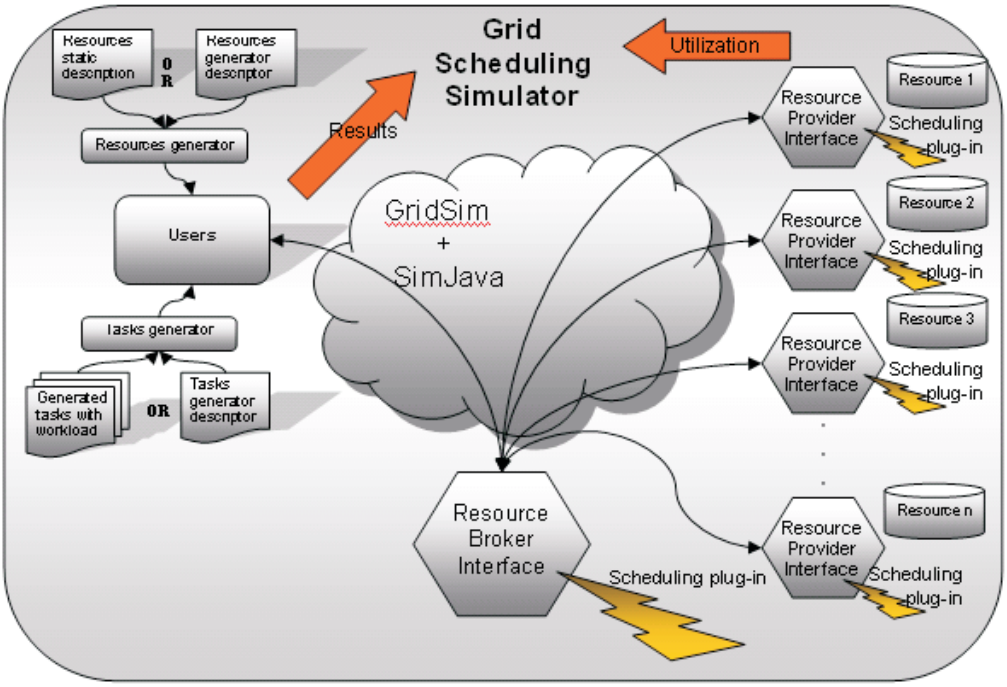

Figure 3: General concept of GSSIM

- Job queue
- Job registry (information about scheduled jobs).

Developer of a plugin must obtain the needed information from these elements, schedule jobs, and return a scheduling decision which is also defined by the interface.

Since GSSIM aims at simulating the whole Grid environment, we cannot assume that a resource broker has a direct control over all resources. Therefore, access to all resources is controlled by local resource providers. GSSIM provides possibility to implement an arbitrary scheduling policy of every single resource provider. This feature can be very useful for a simulation of influence of local queuing systems policies on the overall performance. The same as for a Grid resource broker, a plugin developer must use a proper interface: the *Resource Provider Interface*. In more detail, there are two types of resource provider interfaces in GSSIM. The first provides functionality similar to basic existing local queuing systems such as managing jobs in queues, submitting, canceling jobs etc. The second allows a resource broker to request an advance reservation of resources from resource providers.

The third type of plugins allow developers to customize the way the job execution time is calculated. This is very useful for simulation of applications for which performance models are known. It helps to take into account a diversity of applications used in Grids.

Although a prototype of GSSIM has been already used for experiments there are still many challenges and implementation efforts needed. First of all, development of various plugins and simulations of many Grid environments will help to evaluate the simulator and to introduce needed improvements. The major next steps and future plans foreseen to date are as follows:

- Advanced description and simulation of network topology and parameters
- Use of other job and resource description languages
- Simulation of multiple resource brokers at the same time
- Improvements of reporting and analysis tools

3.6 Grid Performance Modeling (UPC)

The modeling and prediction techniques presented have been evaluated and proposed in the scope of the UPC by the eNanos research group. They are based on statistical and data mining techniques that other authors, like Gibbons, Smith or Downey have proposed previously. We have carried out an analysis of the usage of prediction techniques in the backfilling scheduling techniques ant its derivate. We based all these studies on a previous workload analysis focused on finding out desirables workload properties for the performance prediction [52].

We carried out a deeper analysis of which is the impact of the runtime prediction in the EASY backfilling policy and its derivate: LXWF, SJF and SJL [51]. We evaluated two different types of errors: quantitative errors, adding an amount of time in the real runtime; and qualitative errors, changing completely the nature of the job, for example estimating that a job will be really large while it is short. Furthermore, we evaluated the impact of predicting with accuracy a given subset of jobs based on their nature, for example jobs

that use a reduced number of processors or jobs that use a high amount of memory. The main goal of this second study have been detecting if predicting with accuracy a given kind of applications the performance of backfilling increases substantially. We have designed new performance metric that provides a uniform index of satisfaction of the user about the submitted jobs. Our main goal have been designing a metric that tries to evaluate the Quality of Service that the user perceives, providing a uniformed way to evaluate all submitted jobs. It is not intended to be substitute of the other used metrics, rather this, it is intended to provide complementary information that we consider that it is not present in the other.

The experimental part of our work at UPC [52] is also closely related with the workload analysis [13][69]. The conclusions of our experiments have been contrasted and based with the analytical studies available for each of the workloads that we have used in our simulations:

- The San-Diego Supercomputer Center (SDSC) Paragon logs from 1995. In [88] can be found a comparasion between the Paragon '95 workload with the NASA Ames iPSC workload.
- The San-Diego Supercomputer Center (SDSC) Paragon logs from 1996.
- The Cornell Theory Center (CTC) SP2 log [41].
- The Swedish Royal Institute of Technology (KTH) SP2 log.
- The Los Alamos National Lab (LANL) CM-5 log

Initially we also used workloads synthetically generated that were based on models such as [23][46][69][36]. However, as the number of experiments to generate was considerably big we decided only to test our ideas with the mentioned 5 workloads.

All the experiments have been conducted using a C++ event-driven simulator that was implemented and used by Tsafrir et al. in the paper [86].

This modular simulator allows to implement several policies due to the core of the simulator that provides the event scheduling is independent to them. It also allows adding predictors modules. Therefore a given policy can easily use predictions for plan the schedule of the jobs.

The inputs for the simulations are basically divided in: a set of parameters that allows to choose which policy has to be used, which kind of estimation has to be used (in case that the policy requires one) and other parameters that we added for tune the synthetic estimations used in the study; and a workload trace that is based in the standard workload format (SWF) proposed by Feitelson et al. [17] [37].

As a result of these studies we concluded that qualitative errors in job runtime estimation have shown similar patterns as the other presents in the quantitative errors: adding qualitative errors to the short jobs has an important impact on the system performance. It has been demonstrated that adding qualitative errors on their runtime estimations rather results in dramatic drop of the performance. The backfilling scheduling policies are being affected by the jobs runtime estimation error. However, not all the errors are critical, adding similar errors to all the jobs has no real impact, neither adding errors to those jobs that are large or medium. On the other hand, estimation errors for short jobs are more critical. If the estimations error increases, it results in a substantial increase of slowdown. Furthermore,

qualitative errors in these jobs are more critical than the others, and predictors should try to avoid them.

Taking into account the results obtained in this study, we support that it make sense to design specialized predictors for determined job types, and use them instead of user runtime estimates as an input of backfilling scheduling policies. In general, these predictors should have to be as much accurate as possible for those jobs that are likely to be short, and try to avoid qualitative errors on them. On the other hand higher errors could be acceptable in the rest of the jobs.

Based on the presented evaluation for the impact of the prediction error in the backfilling policies, we designed a set of prediction techniques and they have been deployed in the systems. The prediction of the runtime and memory usage is be carried out using the statistical estimators of the mean, median or the linear regression. The historical information of the workload is used to estimate such values. However, when a prediction is required, not all the historical information is used for the computation. The entries of historical data used for the prediction are selected using a set of static templates (for example [user, group, number processors]). The more specific template that matches the job is used. Also, other prediction techniques based in data mining techniques have been used for estimate the run time, for instance C45, ID3 binary trees, X-Means or K-Nearest Neighbor algorithms.

4 The Questionnaire on the Evaluation of Grid Scheduling Systems

In order to get to know the work and the view of the partners in the CoreGRID Virtual Institute on Resource Management and Scheduling on grid scheduling and benchmarking, we have held a survey among all these partners by means of a questionnaire. In the first part, this questionnaire asks about the evaluation methods used by the partners, and in more detail, about the use of mathematical analysis, simulations, and experimentation. In the second part it asks about the performance metrics used and about the assessment of the impact of failures. Finally, the questionnaire asks what the partners think are currently the most important research problems in grid performance evaluation.

We present the full text of this questionnaire below.

```
Questionnaire for deliverable D.RMS.5 (review of Benchmarking Models)

Please note that all the data provided in this questionnaire will
be anonymized.

0) Introduction

        1. User data
               a. Name: _________________________

               b. Institution (please specify at least name and country):
                        _________________________
```

```
               c. Contact information:
                  email: _________________


1) Benchmarking and evaluation method

1. What evaluation method do you use? (more than 1 answer may apply)
[ ] mathematical analysis
[ ] simulations
[ ] experiments in a real test bed
[ ] experiments in a production environment

2. *If* you use mathematical analysis,
a. Which mathematical methods  do you use?
  [ ] Markov models
     [ ] graph-theoretic models
  [ ] numerical approximations
[ ] other, please specify:

b. How do you generate the workload for your model?
[ ] synthetic distributions
[ ] distributions derived from traces of real systems
[ ] actual data from traces of real systems

c. Does your model include (y/n)
[ ] sequential applications?
[ ] parallel applications?
[ ] workflows?
[ ] batch applications?
[ ] interactive applications?

d. What percentage of the jobs in your model are sequential?
[ ] 100
[ ] >75
[ ] about 50
[ ] < 25
[ ] 0

e. Which resources do you model?
[ ] processors
[ ] storage/data


3. *If* you use simulations,
a. Which simulation package do you use?
[ ] GridSim
```

```
[ ] MicroGrid
[ ] GangSim
[ ] SimGrid
[ ] Another grid simulation package, please specify:
[ ] A general simulation tool (e.g., Csim), please specify:
  [ ] Our own simulation program

b. How do you generate the workload in your model:
[ ] synthetic distributions
[ ] distributions derived from traces of real systems
[ ] traces of real systems

c. Does your model include (y/n)
[ ] sequential applications?
[ ] parallel applications?
[ ] workflows?
[ ] batch applications?
[ ] interactive applications?

d. What percentage of the jobs in your model are sequential?
[ ] 100
[ ] >75
[ ] about 50
[ ] < 25
[ ] 0

e. Which resources do you model?
[ ] processors
[ ] storage/data
[ ] network bandwidth


4. *If* you do experiments in a real system:
a. Please specify the system:
[ ] total number of processors
[ ] total number of clusters
[ ] total amount of storage capacity

b. What local resource manager is used?
[ ] PBS
[ ] SGE
[ ] Condor
[ ] LSF
[ ] other, please specify:

c. What Grid midddleware is used?
```

```
[ ] Globus
[ ] Legion
[ ] UNICORE
[ ] DIET
[ ] other, please specify:

d. Do you use a grid scheduler (y/n)? If so, which one:

e. How do you generate the applications for the workload for your testbed:
[ ] synthetic applications
[ ] real applications

f. Does your model include (y/n)
[ ] sequential applications?
[ ] parallel applications?
[ ] workflows?
[ ] batch applications?
[ ] interactive applications?

g. What percentage of the jobs in your workloads are sequential?
[ ] 100
[ ] >75
[ ] about 50
[ ] < 25
[ ] 0

f. How do you generate the arrivals of jobs in your workloads?
[ ] synthetic distributions (e.g., Poisson arrivals)
[ ] distributions derived from traces of real systems
[ ] traces of real systems

g. Of wich resources do you measure the performance?
[ ] processors
[ ] storage/data
[ ] network bandwidth


2) Benchmarking methodology

1. Which performance metrics do you use? (more than 1 answer may apply)
[ ] response time
[ ] weighted response time
[ ] wait time
[ ] weighted wait time
[ ] system utilization
[ ] overhead in the middleware
```

```
[ ] other, please specify:

2. Do you assess the influence of failures in your model/system (y/n)?


3) What do you regard as the biggest research problem in benchmarking
and evaluation of grid performance?
```

5 Survey Results

In this section we present the results of our survey. Throughout this section we use the terms "groups" and "respondents" interchangeably to denote the individual participants in the survey. The following members of the CoreGRID Virtual Institute on Resource Management and Scheduling have responded to the questionnaire of Section 4:

1. CETIC
2. TUD
3. EPFL
4. CNRS
5. UNIDO
6. MU BRNO
7. PSNC
8. MTA SZTAKI
9. UNICAL
10. UPC
11. UCY

5.1 Benchmarking and Evaluation Method

Figure 4 shows the survey results for the evaluation method used. A significant number of groups use simulations or experiments in real environments for experimental purposes: Over 80% of the respondents use simulations, and over 60% of the respondents experiment in real environments. The number of respondents with access to real production environments is less than the number of respondents that use mathematical analysis for experiments (the numbers are 27%, and 36%, respectively), which is surprising since grids (as large-scale, complex systems) are notoriously difficult to be subjected to analytical methods. The number of respondents that experiment in production environments is less than half the number of respondents that experiment in research-only environments.

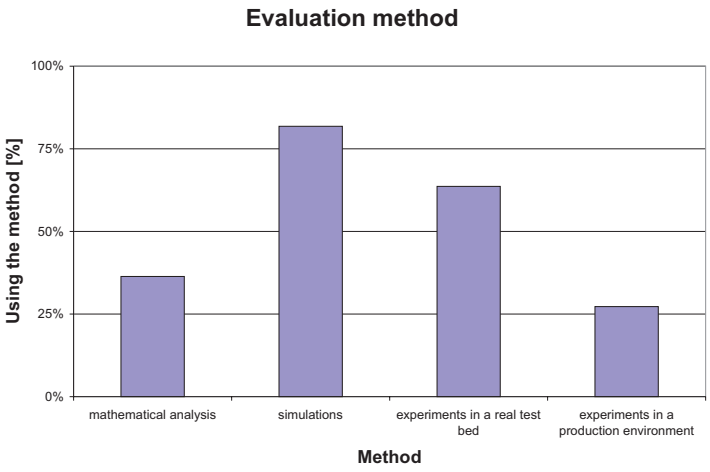

Figure 4: Survey results for the evaluation method used.

5.1.1 Mathematical analysis

Figure 5 shows the survey results for groups using mathematical analysis.

Three quarters of the respondents that employ mathematical analysis use Markov chains-based analysis, which also means that over a quarter of all respondents use Markov chains-based analysis. Numerical approximations are used by half of the respondents that employ mathematical analysis. Graph-theoretic models are used by a quarter of the respondents that employ mathematical analysis. There are no other mathematical analysis methods used by our respondents for their grid research.

According to their replies, half of our respondents that use mathematical analysis generate workloads based on distributions derived from traces of real systems, and half use actual data from traces of real systems (the two choices are non-exclusive). This is surprising, as grid workloads have not yet been reported until a couple of years ago, and grid traces have started to be made available only this year. All our respondents also use synthetic distributions to generate experimental workloads.

In their experimental workloads, all the respondents that use mathematical analysis use both sequential and parallel applications. Half of them use batches of jobs, and a quarter of them use workflow applications. No respondent mentioned the use of interactive applications in their analysis work.

While all our respondents that use mathematical analysis use sequential applications, the percentage of sequential jobs included in experimental workloads is low. Three quarters of the responses mentioned that up to 25% of their jobs are sequential, and the remaining quarter generate workloads with between 25% and 50% sequential jobs.

All our respondents model processors for their analysis work, with a quarter also modeling storage/data aspects.

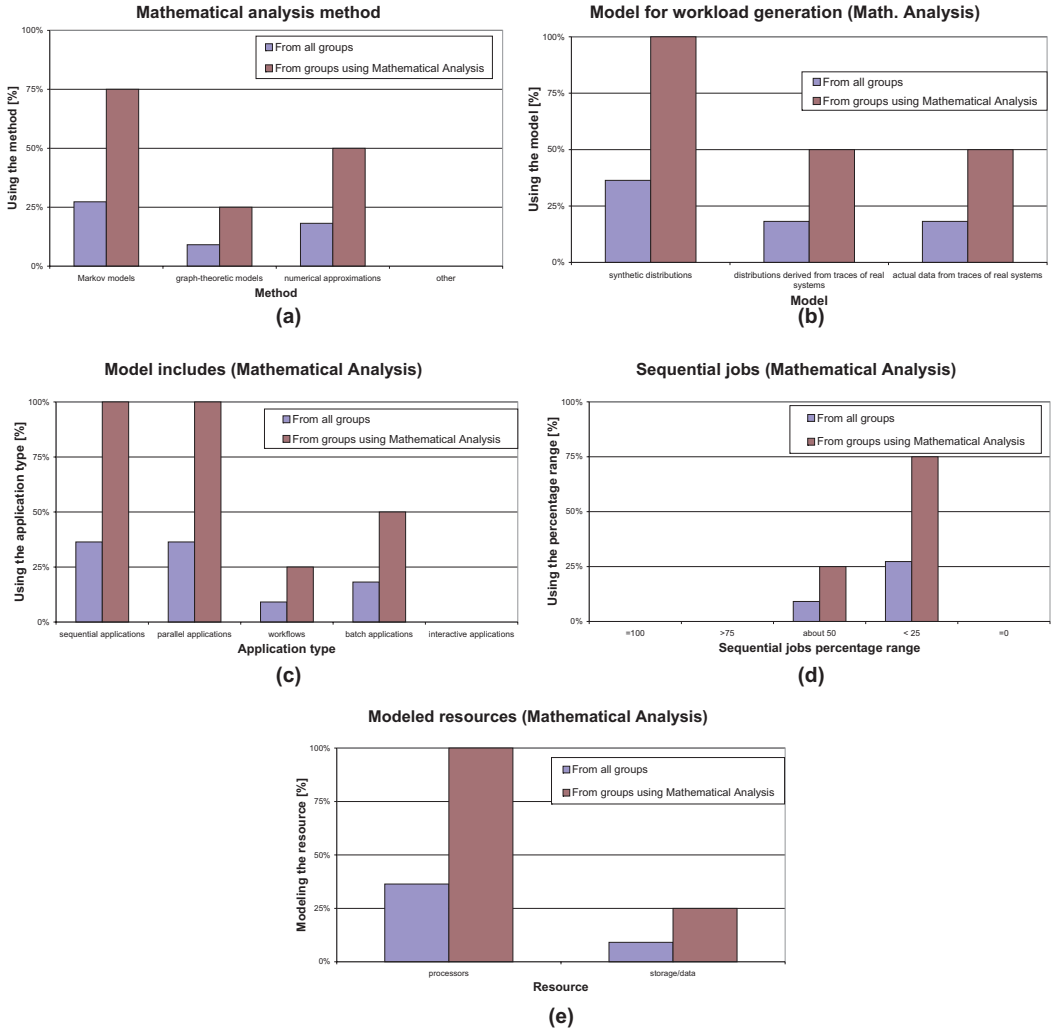

Figure 5: Survey results for groups using mathematical analysis: (a) mathematical analysis method; (b) model for workload generation; (c) types of applications included in the experimental workloads; (d) percentage range for sequential jobs in the experimental workloads; (e) modeled resources.

5.1.2 Simulation

Figure 6 shows the survey results for groups using simulations.

Over three quarters of our respondents that use simulation use their own simulation package. Other commonly mentioned simulation packages are GridSim and SimGrid, which are used by about a fifth of the respondents that use simulation, respectively. About a tenth

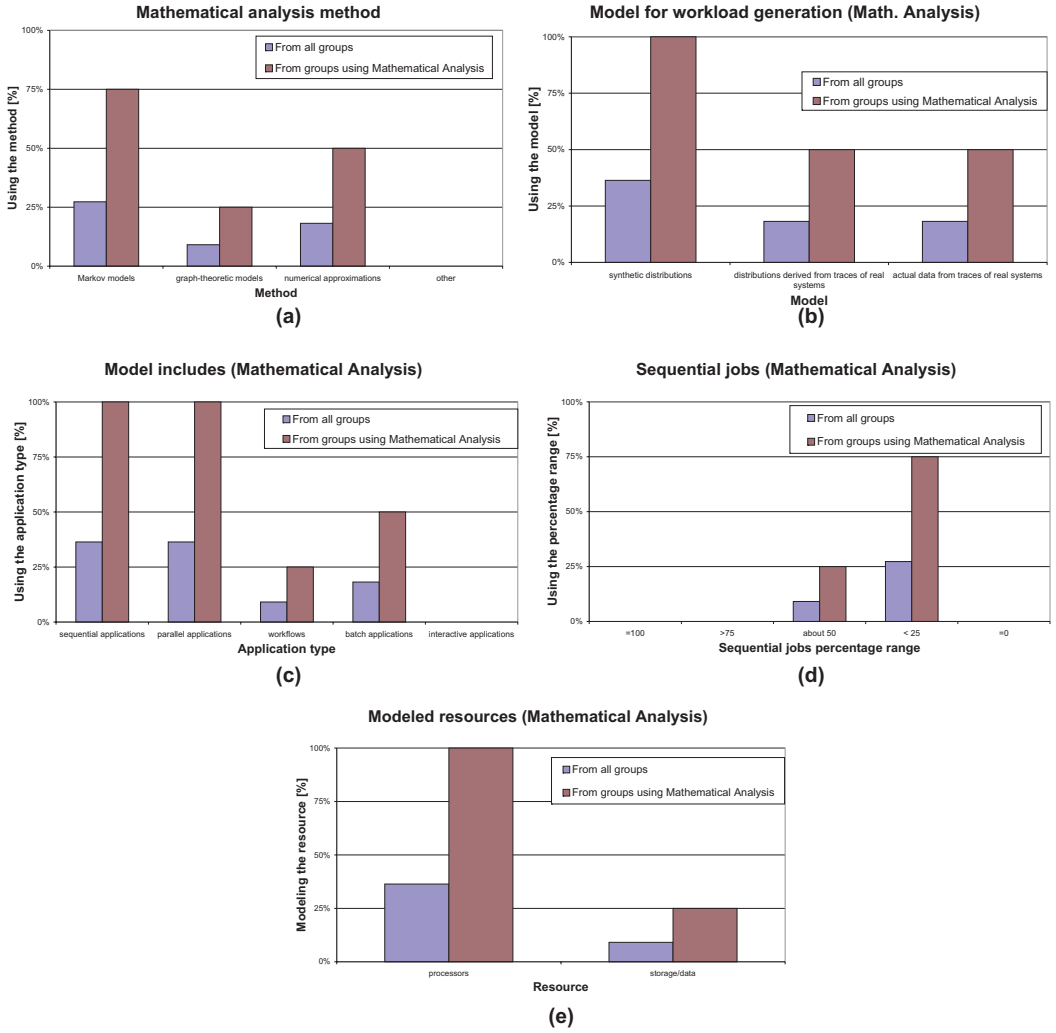

Figure 6: Survey results for groups using simulation: (a) simulation package; (b) model for workload generation; (c) types of applications included in the experimental workloads; (d) percentage range for sequential jobs in the experimental workloads; (e) modeled resources.

of the respondents that use simulation use another grid simulation package (e.g., the EASY simulation by Tsafir and Feitelson), or even a general simulation tool (e.g., CSim) on which they build their own simulator, respectively.

Similarly to the replies obtained for users of mathematical analysis, roughly half of our respondents that use mathematical analysis generate workloads based on distributions derived from traces of real systems, and half use actual data from traces of real systems (the two choices are non-exclusive). Almost all our respondents also use synthetic distributions

to generate experimental workloads.

Also similarly to the replies obtained for users of mathematical analysis, all the respondents that use simulation use sequential applications. Over 88% of them use parallel applications. Roughly half (44%) of them use batches of jobs, a third use workflow applications, and a tenth use of interactive applications in their analysis work.

The percentage of sequential jobs included in experimental workloads is much higher for respondents that use simulation, compared to that of the respondents that use analytical methods. Three quarters of the responses mentioned that up to 50% of their jobs are sequential, with roughly half of them generating workloads with between 25% and 50% sequential jobs. A fifth of the respondents that use simulation employ workloads with 75% or more sequential jobs.

All our respondents model processors for their analysis work, a half model network bandwidth aspects, and a fifth model storage and data.

5.1.3 Experiments in real environments

Figure 7 shows the survey results for groups experimenting in real environments.

Only one respondent had access to more than one grid. The average number of clusters to which our respondents that experiment in real environments have access is 9, but the distribution of values is highly skewed to lower numbers. The number of clusters ranges from 1 to 40, with the median value being 4. The average number of processors to which our respondents that experiment in real environments have access is 1717, but again the distribution of values is highly skewed. The number of processors ranges from 10 to 10432, with the median value being 300. The average size of disk storage for respondents that experiment in real environments is just above 1TB (though only 50% of our respondents that experiment in real environments indicated their disk size).

The respondents that experiment in real environments use local resource management ensured by PBS, SGE, and Condor, in 30%, 20%, and 10% of the cases, respectively. A fifth of the respondents that experiment in real environments use another local resource management tool, e.g., OAR or LoadLeveler.

Globus dominates the used grid middleware (with 60%), followed by UNICORE (with 20%). Other grid middleware such as XtremWeb, GridSolve, EGEE, and LCG, is available in 20% of the cases (one mention for each of these grid middleware packages).

Half of our respondents use a grid scheduler, with Koala, ISS, LCG-2 Broker, eNANOS, UNICORE-MetaSchedulerService, gLite Broker, and GTbroker receiving one mention each.

A surprisingly high number of experimenters employ real applications in their workloads: 70%. Just below a third of our respondents use synthetic applications in their workloads.

The respondents that experiment in real environments generate their workloads based on distributions derived from traces of real systems, on actual data from traces of real systems, or on synthetic distributions in 20%, 30%, and 30% of the cases, respectively.

Over three quarters of the respondents that experiment in real environments use sequential applications, and/or parallel applications (80% of the responses for each of the two). Just below of a third of them use batches of jobs, and/or workflow applications (30% of the responses for each of the two). A tenth use interactive applications in their analysis work.

The percentage of sequential jobs included in experimental workloads is lower for respondents that experiment in real environments, than for respondents that use simulation.

Over half (60%) of the responses mentioned that up to 50% of their jobs are sequential. No respondent confirmed the use of 100% sequential jobs.

Similarly to the simulation case, most of our respondents model processors for their analysis work, a third model network bandwidth aspects, and a fifth model storage and data.

5.2 Benchmarking Methodology

Figure 8 shows the survey results for the performance metrics used. System utilization and response time are the metrics of choice for a large majority of our respondents (91% and 73%, respectively), followed by wait time (45%) and middleware overhead (36%). By comparison, only 27% of the respondents assess the influence of failures in their model/system. It is also important that in 45% of the cases the respondents consider other metrics for their work, e.g., workload makespan, micro-benchmark metrics, (bounded) slowdown, application runtimes starvation, and backfilling metrics, etc.

5.3 Research Problems in Grid Evaluation and Benchmarking

A number of research problems in grid evaluation and benchmarking emerge from the answers of our respondents:

1. Performing realistic benchmarks, especially in production and highly dynamic environments;
2. Lack of information on and access to real grid workloads;
3. Lack of good Grid performance models (job-level workload model, modeling user behavior);
4. The comparability of evaluation results between different approaches (repeatability of experiments, accounting for background load in experiments, different models, different workloads, incompatible descriptions and interfaces to make grid workloads and simulation environments interoperable);
5. Assessing fairness of grid scheduling policies;
6. Assessing reliability of grid environments.

6 Conclusions

In this report we have given an overview of the work in performance analysis and benchmarking of Grid scheduling systems in the CoreGRID Virtual Institute (VI) on Resource Management and Scheduling (RMS), based on texts contributed by the partners in this VI and on the results of a questionnaire. In particular, we have asked the partners about the most important current research questions in the area. Based on the material presented in this report, and in particular in Sections 3 and 5, we can formulate the following conclusions:

1. Comparing the contributions of the partners in Section 3 and the research problems stated in Section 5.3, we conclude that the partners in the CoreGRID VI on RMS are working on the right research topics, especially on research problems 1, 2, 3 and 6 stated in Section 5.3.

2. There is a decent amount of benchmarking work done in real environments, which is important for the motivation of the research and for the applicability of the results. Only after research results have been validated in such environments can they possibly be transferred to practice.

3. In addition to the work in real environments, in CoreGRID there is also still work on modeling and simulation of grid scheduling systems going on, which is fundamental to the understanding of the performance of grids.

4. There is still much work to be done in the area of benchmarking and performance evaluation of grid scheduling systems in view of the important research problems stated in Section 5.3. In particular, much more collaboration is warranted with respect to research problem 4 in that section to make for a better comparability of the methods employed and the results obtained.

References

[1] Alexandru Iosup and Catalin Dumitrescu and Dick H.J. Epema and Hui Li and Lex Wolters. How are real grids used? The analysis of four grid traces and its implications. In *Seventh IEEE/ACM International Conference on Grid Computing (GRID)*, pages 262–269. IEEE Computer Society, Sep 2006.

[2] H.E. Bal, A. Plaat, M.G. Bakker, P. Dozy, and R.F.H. Hofman. Optimizing Parallel Applications for Wide-Area Clusters. In *Proc. of the 12th Int'l Parallel Processing Symp.*, pages 784–790, 1998.

[3] H.E. Bal et al. The Distributed ASCI Supercomputer Project. *ACM Operating Systems Review*, 34(4):76–96, 2000.

[4] S. Banen, A.I.D. Bucur, and D.H.J. Epema. A Measurement-Based Simulation Study of Processor Co-Allocation in Multicluster Systems. In D.G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Proc. of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2862 of *LNCS*, pages 105–128. Springer-Verlag, 2003.

[5] Francine Berman, Richard Wolski, Henri Casanova, Walfredo Cirne, Holly Dail, Marcio Faerman, Silvia M. Figueira, Jim Hayes, Graziano Obertelli, Jennifer M. Schopf, Gary Shao, Shava Smallen, Neil T. Spring, Alan Su, and Dmitrii Zagorodnov. Adaptive computing on the grid using apples. *IEEE Trans. Parallel Distrib. Syst.*, 14(4):369–382, 2003.

[6] A.I.D. Bucur and D.H.J. Epema. The Performance of Processor Co-Allocation in Multicluster Systems. In *Proc. of the 3rd IEEE/ACM Int'l Symp. on Cluster Computing and the GRID (CCGrid2003)*, pages 302–309. IEEE Computer Society Press, 2003.

[7] A.I.D. Bucur and D.H.J. Epema. Trace-Based Simulations of Processor Co-Allocation Policies in Multiclusters. In *Proc. of the 12th IEEE Int'l Symp. on High Performance Distributed Computing (HPDC-12)*, pages 70–79. IEEE Computer Society Press, 2003.

[8] Anca I. D. Bucur and Dick H. J. Epema. The performance of processor co-allocation in multicluster systems. In *Proc. of the 3rd IEEE Int'l. Symp. on Cluster Computing and the Grid (CCGrid)*, pages 302–309, 2003.

[9] Anca I. D. Bucur and Dick H. J. Epema. Trace-based simulations of processor co-allocation policies in multiclusters. In *Proc. of the 12th Intl. Symposium on High-Performance Distributed Computing (HPDC)*, pages 70–79, 2003.

[10] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. In *Special Issue of the Proceedings of the IEEE on Grid Computing*. IEEE Press, 2005. (To appear).

[11] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 14, Nov.-Dec. 2002.

[12] Maria Calzarossa and Giuseppe Serazzi. A characterization of the variation in time of workload arrival patterns. *IEEE Trans. Comput.*, C-34(2):156–162, Feb 1985.

[13] Maria Calzarossa and Giuseppe Serazzi. Workload characterization: A survey. *Proc. IEEE*, 81:1136–1150, 1993.

[14] L.-C. Canon and E. Jeannot. Wrekavoc a Tool for Hemulating Heterogeneity. In *15th IEEE Heterogeneous Computing Workshop (HCW'06)*, Island of Rhodes, Greece, April 2006.

[15] F. Cappello, F. Desprez, M. Dayde, E. Jeannot, Y. Jegou, S. Lanteri, N. Melab, P. Primet R. Namyst, O. Richard, E. Caron, J. Leduc, and G. Mornet. Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform. In *6th IEEE/ACM International Workshop on Grid Computing (GRID 2005)*, Seattle, WA, USA, November 2005.

[16] Steve J. Chapin, Walfredo Cirne, Dror G. Feitelson, James Patton Jones, Scott T. Leutenegger, Uwe Schwiegelshohn, Warren Smith, and David Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In Dror G. Feitelson and Larry Rudolph, editors, *Proc. of the 5th Int'l. Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, volume 1659 of *Lecture Notes in Computer Science*, pages 67–90. Springer, 1999.

[17] Steve J. Chapin, Walfredo Cirne, Dror G. Feitelson, James Patton Jones, Scott T. Leutenegger, Uwe Schwiegelshohn, Warren Smith, and David Talby. Benchmarks and standards for the evaluation of parallel job schedulers. *Job Scheduling Strategies for Parallel Processing*, vol 1659:pp. 66–89, 1999.

[18] Brent N. Chun, Philip Buonadonna, Alvin AuYoung, Chaki Ng, David C. Parkes, Jeffrey Shneidman, Alex C. Snoeren, and Amin Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In *Proc. of 2nd IEEE Workshop on Embedded Networked Sensors (EmNetsII)*, 2005.

[19] Greg Chun, Holly Dail, Henri Casanova, and Allan Snavely. Benchmark probes for grid assessment. In *Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[20] Walfredo Cirne and Francine Berman. A Comprehensive Model of the Supercomputer Workload. In *4th Workshop on Workload Characterization*, December 2001.

[21] Walfredo Cirne and Francine Berman. A model for moldable supercomputer jobs. In *Proc. of the 15th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 59–79, 2001.

[22] *The Distributed ASCI Supercomputer (DAS).*

[23] Allen B. Downey. Using queue time predictions for processor allocation. *3rd Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes In Computer Science; Vol. 1291:35 – 57, 1997.

[24] Allen B. Downey. A parallel workload model and its implications for processor allocation. *Cluster Computing*, 1(1):133–145, 1998.

[25] C. Dumitrescu, D.H.J. Epema, J. Duennweber, and S. Gorlatch. User-transparant scheduling of structured parallel applications in grid environments. In *Workshop on HPC Grid Programming Environments and Components and Component and Framework Technology in High-Performance and Scientific Computing (HPC-GECO+COMPFRAME)*. IEEE Computer Society Press, 2006.

[26] Catalin Dumitrescu, Ioan Raicu, and Ian T. Foster. Experiences in running workloads over grid3. In Hai Zhuge and Geoffrey Fox, editors, *Grid and Cooperative Computing (GCC)*, volume 3795 of *Lecture Notes in Computer Science*, pages 274–286. Springer, 2005.

[27] C.L. Dumitrescu, D.H.J. Epema, J. Dunnweber, and S. Gorlatch. Reusable cost-based scheduling of grid workflows operating on higher-order components. In *Proc. of the Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, pages 87–94. IEEE Computer Science, Dec 2006.

[28] C. Ernemann and R. Yahyapour. *Grid Resource Management - State of the Art and Future Trends*, chapter Applying Economic Scheduling Methods to Grid Environments, pages 491–506. Kluwer Academic Publishers, 2003.

[29] Carsten Ernemann, Volker Hamscher, Uwe Schwiegelshohn, Achim Streit, and Ramin Yahyapour. Enhanced Algorithms for Multi-Site Scheduling. In *Proceedings of the 3rd International Workshop on Grid Computing, Baltimore*, volume 2536 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2002.

[30] Carsten Ernemann, Volker Hamscher, Uwe Schwiegelshohn, Ramin Yahyapour, and Achim Streit. On advantages of grid computing for parallel job scheduling. In *CCGRID*, pages 39–49, 2002.

[31] Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. Economic scheduling in grid computing. In *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*, pages 128–152. Springer-Verlag, 2002.

[32] Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. Benefits of global grid computing for job scheduling. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, November 2004. IEEE Computer Society.

[33] Carsten Ernemann, Baiyi Song, and Ramin Yahyapour. Scaling of Workload Traces. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 166–183. Springer, October 2003.

[34] Carsten Ernemann, Baiyi Song, and Ramin Yahyapour. Scaling of workload traces. In *JSSPP*, pages 166–182, 2003.

[35] Carsten Ernemann and Ramin Yahyapour. *"Grid Resource Management - State of the Art and Future Trends"*, chapter "Applying Economic Scheduling Methods to Grid Environments", pages 491–506. Kluwer Academic Publishers, 2003.

[36] D. G. Feitelson. Packing schemes for gang scheduling'. *Job Scheduling Strategies for Parallel Processing*, Lect. Notes Comput. Sci. 1162:pp. 89–110, 1996.

[37] Dc. Dror G. Feitelson. Parallel workload archive - http://www.cs.huji.ac.il/labs/parallel/workload, 2006.

[38] D.G. Feitelson. Packing Schemes for Gang Scheduling. In D.G. Feitelson and L. Rudolph, editors, *Proc. of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1162, pages 89–110. Springer-Verlag, 1996.

[39] Dror G. Feitelson. The forgotten factor: facts on performance evaluation and its dependence on workloads. In B. Monien and R. Feldmann, editors, *Euro-Par 2002 Parallel Processing*, volume 2400, pages 49–60. Springer, 2002. Lecture Notes in Computer Science.

[40] Dror G. Feitelson. Workload Modeling for Performance Evaluation. In Mariacarla Calzarossa and Sara Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, volume 2459 of *Lecture Notes in Computer Science*, pages 114–141. Springer, 2002.

[41] Dror G. Feitelson and Larry Rudolph. Workload evolution on the cornell theory center ibm sp2. *Job Scheduling Strategies for Parallel Processing*, 1162:pp. 27–40, 1996.

[42] Dror G. Feitelson and Larry Rudolph. Metrics and benchmarking for parallel job scheduling. In Dror G. Feitelson and Larry Rudolph, editors, *Proc. of the 4th Int'l. Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, volume 1459 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 1998.

[43] Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn, Kenneth C. Sevcik, and Parkson Wong. Theory and Practice in Parallel Job Scheduling. In Dror G. Feitelson and Larry Rudolph, editors, *Proc. of the 3rd Int'l. Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, volume 1291 of *Lecture Notes in Computer Science*, pages 1–34, Geneva, April 1997. Springer-Verlag.

[44] I. Foster, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In *Int'l Workshop on Quality of Service*, pages 27–36, 1999.

[45] Ian Foster, Markus Fidler, Alain Roy, Volker Sander, and Linda Winkler. End-to-end quality of service for high-end applications. *Computer Communications*, 27(14):1375–1388, September 2004.

[46] H. Franke, J. Jann, J. E. Moreira, P. Pattnaik, , and M. A. Jette. An evaluation of parallel job scheduling for asci blue-pacific. *Supercomputing*, November 1999.

[47] Michael A. Frumkin and Rob F. Van der Wijngaart. Nas grid benchmarks: A tool for grid space exploration. In *Proc. of the 10th Intl. Symposium on High-Performance Distributed Computing (HPDC)*, pages 315–326, 2001.

[48] *The Global Grid Forum.* `www.gridforum.org`.

[49] Fred W. Glover and Gary A. Kochenberger, editors. *Handbook of Metaheuristics*. Kluwer, 2002.

[50] GRMS. `http://www.gridge.org/content/view/18/99/`.

[51] F. Guim, J. Corbalan, and J. Labarta. Impact of qualitative and quantitative errors of the job runtime estimation in backfilling based scheduling policies. Technical report, Architecture Computer Deparment - Technical University of Catalunya, 2005.

[52] Francesc Guim, Julita Corbalan, and Jesus Labarta. Analyzing loadleveler historical information for performance prediction. *Jornadas de Paralelismo 2005 and Granada*, 2005.

[53] Soonwook Hwang and Carl Kesselman. Gridworkflow: A flexible failure handling framework for the grid. In *Proc. of the 12th Intl. Symposium on High-Performance Distributed Computing (HPDC)*, pages 126–137, 2003.

[54] A. Iosup, D.H.J. Epema, C. Franke, A. Papaspyrou, L. Schley, B. Song, and R. Yahyapour. On grid performance evaluation using synthetic workloads. In E. Frachtenberg and U. Schwiegelshohn, editors, *Proc. of the 12th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP),*, Lecture Notes in Computer Science. Springer Verlag, Jun 2006.

[55] Alexandru Iosup and D.H.J. Epema. GrenchMark: A framework for analyzing, testing, and comparing grids. In *Proc. of the 6th IEEE/ACM Int'l. Symp. on Cluster Computing and the GRID (CCGrid)*, May 2006. (accepted).

[56] Alexandru Iosup, Dick Epema, Peter Couvares, Anatoly Karp, and Miron Livny. Build-and-test workloads for grid middleware: Problem, analysis, and applications. In *Proc. of the 7th IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGrid07)*. IEEE Computer Society, 2007. 14-17 May 2006, Rio de Janeiro, Brasil.

[57] Alexandru Iosup and Dick H. J. Epema. Grenchmark: A framework for analyzing, testing, and comparing grids. In *CCGRID*, pages 313–320. IEEE Computer Society, 2006.

[58] David Jackson, Quinn Snell, and Mark Clement. Core algorithms of the Maui scheduler. In Dror G. Feitelson and Larry Rudolph, editors, *Proc. of the 7th Int'l. Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, volume 2221 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2001.

[59] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,*. Wiley-Interscience, New York, NY, USA, May 1991. Winner of "1991 Best Advanced How-To Book, Systems" award from the Computer Press Association.

[60] Chris Kenyon and Giorgos Cheliotis. Architecture requirements for commercializing grid resources. In *Proc. of the 11th Intl. Symposium on High-Performance Distributed Computing (HPDC)*, pages 215–224, 2002.

[61] T. Kielmann, R.F.H. Hofman, H.E. Bal, A. Plaat, and R.A.F. Bhoedjang. MagPIe: MPI's Collective Communication Operations for Clustered Wide Area Systems. In *ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pages 131–140, 1999.

[62] Dalibor Klusacek. Planovani uloh v paralelnim a distribuovanem prostredi. Master's thesis, Faculty of Informatics, Masaryk University, Brno, 2006.

[63] George Kola, Tevfik Kosar, and Miron Livny. Phoenix: Making data-intensive grid applications fault-tolerant. In Rajkumar Buyya, editor, *GRID*, pages 251–258. IEEE Computer Society, 2004.

[64] Derrick Kondo, Michela Taufer, Charles L. Brooks III, Henri Casanova, and Andrew A. Chien. Characterizing and evaluating desktop grids: An empirical study. In *Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[65] Hui Li, David Groep, and Lex Wolters. Workload characteristics of a multi-cluster supercomputer. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing (JSSPP'04)*, pages 176–194. Springer-Verlag, June 2004.

[66] Jiadao Li and Ramin Yahyapour. Learning-based negotiation strategies for grid scheduling. In *To be published in Proceedings of the International Symposium on Cluster Computing and the Grid (CCGRID2006)*. IEEE Computer Society, 2006.

[67] Jiadao Li and Ramin Yahyapour. Negotiation strategies for grid scheduling. In *Proceedings of the First International Conference on Grid and Pervasive Computing*, Lecture Notes in Computer Science (LNCS 3947). Springer, 2006.

[68] Uri Lublin and Dror G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(20):1105–1122, 2003.

[69] Uri Lublin and Dror G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *J. Parallel and Distributed Computing*, 11:pp. 1105–1122, November 2003.

[70] Raissa Medeiros, Walfredo Cirne, Francisco Vilar Brasileiro, and Jacques Philippe Sauvé. Faults in grids: Why are they so bad and what can be done about it?. In Heinz Stockinger, editor, *GRID*, pages 18–24. IEEE Computer Society, 2003.

[71] Emmanuel Medernach. Workload analysis of a cluster in a grid environment. In Dror G. Feitelson, Eitan Frachtenberg, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Proc. of the 11th Int'l. Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, volume 3834 of *Lecture Notes in Computer Science*, pages 36–61. Springer, 2005.

[72] H.H. Mohamed and D.H.J. Epema. An Evaluation of the Close-to-Files Processor and Data Co-Allocation Policy in Multiclusters. In *CLUSTER 2004, IEEE Int'l Conf. on Cluster Computing*, pages 287–298, 2004.

[73] H.H. Mohamed and D.H.J. Epema. Experiences with the koala co-allocating scheduler in multiclusters. In *Proc. of the 5th IEEE/ACM Int'l Symp. on Cluster Computing and the GRID (CCGrid)*, May 2005.

[74] H.H. Mohamed and D.H.J. Epema. Experiences with the KOALA Co-Allocating Scheduler in Multiclusters. In *Proc. of the 5th IEEE/ACM Int'l Symp. on Cluster Computing and the GRID (CCGrid2005)*, pages 784–791, 2005.

[75] Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 2002.

[76] A. Plaat, H.E. Bal, R.F.H. Hofman, and T. Kielmann. Sensitivity of Parallel Applications to Large Differences in Bandwidth and Latency in Two-Layer Interconnects. *Future Generation Computer Systems*, 17:769–782, 2001.

[77] R.Yahyapour. *Design and Evaluation of Job Scheduling Strategies for Grid Computing*. Shaker Verlag, Aachen, Germany, January 2003.

[78] Wayne Schroeder, Richard Marciano, Joseph Lopez, and Michael K. Gleicher. Analysis of HPSS Performance Based on Per-file Transfer Logs. In *Proc. of the 16th IEEE Mass Storage Systems Symposium*, pages 103–115, San Diego, March 1999. IEEE Computer Society.

[79] U. Schwiegelshohn and R. Yahyapour. Analysis of first-come-first- serve parallel job scheduling. In *Proceedings of the 9th SIAM Symposium on Discrete Algorithms*, pages 629–638, January 1998.

[80] U. Schwiegelshohn and R. Yahyapour. Fairness in parallel job scheduling. *Journal of Scheduling*, 3(5):297–320, 2000.

[81] Baiyi Song, Carsten Ernemann, and Ramin Yahyapour. Parallel Computer Workload Modeling with Markov Chains. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Proceedings of the 10th Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science*, pages 47–62. Springer, October 2004.

[82] Baiyi Song, Carsten Ernemann, and Ramin Yahyapour. User Group-based Workload Analysis and Modeling. In *Proceedings of the International Symposium on Cluster Computing and the Grid (CCGRID2005)*. IEEE Computer Society, 2005.

[83] Douglas Thain, John Bent, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Miron Livny. Pipeline and batch sharing in grid workloads. In *Proc. of the 12th Intl. Symposium on High-Performance Distributed Computing (HPDC)*, pages 152–161, 2003.

[84] Mitchell D. Theys, Howard Jay Siegel, Noah B. Beck, Min Ta, and Michael Jurczyk. A Mathematical Model, Heuristic and Simulation Study for a Basic Data Staging Problem in a Heterogenous Networking Environment. In *Proc. of the 7th Heterogeneous Computing Workshop*, pages 115–122, Orlando, March 1998. IEEE Computer Society.

[85] Nicola Tonellotto, Philipp Wieder, and Ramin Yahyapour. A proposal for a generic grid scheduling architecture. Technical report, CoreGRID - Network of Excellence, 2005.

[86] Dan Tsafrir, Yoav Etsion, and Dror G. Feitelson. Modeling user runtime estimates. *In the 11th Workshop on Job Scheduling Strategies for Parallel Processing,Lecture Notes in Computer Science*, Vol.3834:pp. 1–35, 2006.

[87] G. Tsouloupas and M. D. Dikaiakos. GridBench: A workbench for grid benchmarking. In P. M. A. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Proc. of the European Grid Conference (EGC)*, volume 3470 of *Lecture Notes in Computer Science*, pages 211–225. Springer, 2005.

[88] K. Windisch, V. Lo, R. Moore, D. Feitelson, , and B. Nitzberg. A comparison of workload traces from two production parallel machines. *6th Symp. Frontiers Massively Parallel Comput.*, pages pp.319–326, 1996.

[89] Rich Wolski, Neil Spring, and Jim Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems*, 15(5-6):757–768, October 1999.

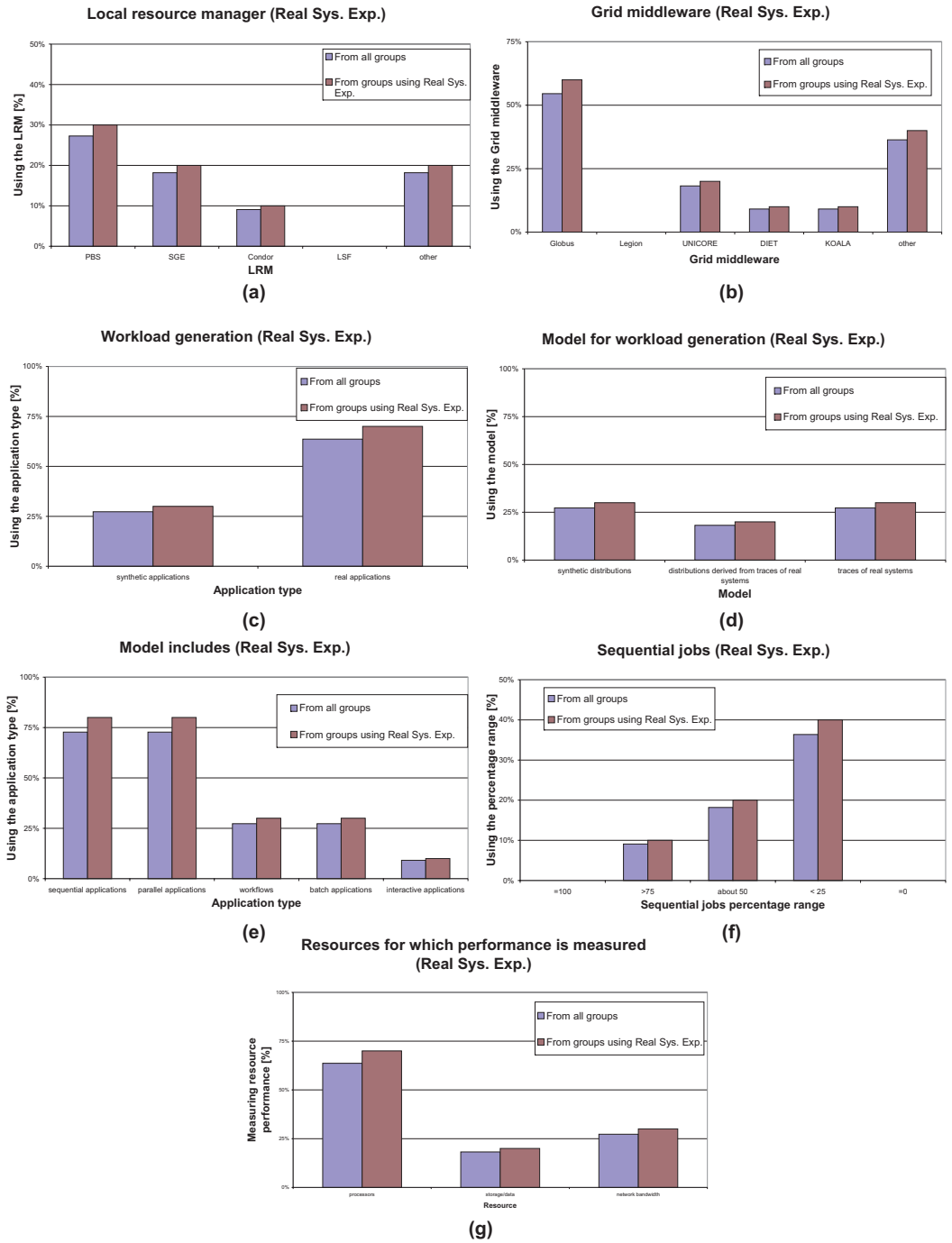

Figure 7: Survey results for groups experimenting in real environments: (a) local resource manager; (b) grid middleware; (c) source of applications (i.e., synthetic or real); (d) model for workload generation; (e) types of applications included in the experimental workloads; (f) percentage range for sequential jobs in the experimental workloads; (g) resources for which performance is measured.

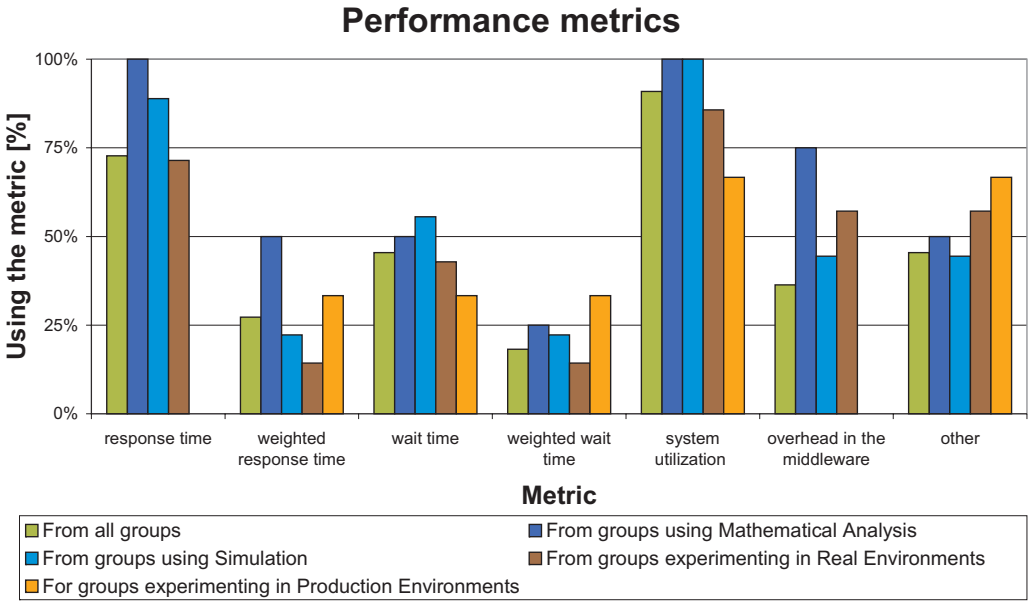

Figure 8: Survey results for performance metrics used.