On Grid Performance Evaluation
using Synthetic Workloads

Dick H.J. Epema[1], Carsten Franke[2]*, Alexandru Iosup[1], Alexander Papaspyrou[2], Lars Schley[2], Baiyi Song[2], and Ramin Yahyapour[2]

[1] Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology, The Netherlands
`{D.H.J.Epema,A.Iosup}@tudelft.nl`

[2] Information Technology Section, Robotics Research Institute
Dortmund University, Germany
`{Carsten.Franke,Alexander.Papaspyrou,`
`Lars.Schley,Song.Baiyi,Ramin.Yahyapour}@udo.edu`

Members of the CoreGRID European Virtual Institute on Grid Resource
Management and Scheduling

Abstract. Grid computing is becoming a common technology platform for solving large scale computing tasks. However, a number of major technical issues, including the lack of adequate performance evaluation approaches, hinder the Grid's further development. The requirements herefore are manifold; adequate approaches must combine appropriate performance metrics, realistic workload models, and flexible tools for workload generation, submission, and analysis. In this paper we present an approach to tackle this complex problem. First, we introduce a set of grid performance objectives based on traditional and grid-specific performance metrics. Second, we synthesize the requirements for realistic grid workload modeling, e.g. co-allocation, data and network management, and failure modeling. Third, we show how GRENCHMARK, an existing framework for generating, running, and analyzing grid workloads, can be extended to implement the proposed modeling techniques. Our approach aims to be an initial and necessary step towards a common performance evaluation framework for grid environments.

1   Introduction

Grid computing facilitates the aggregation and sharing of large sets of heterogeneous resources spread over large geographical areas. This proves beneficial in many situations, for example when applications require more resources than locally available, or when work needs to be balanced across multiple computing facilities [15]. With the industrial and scientific communities tackling increasingly larger problems, grid computing is becoming a common infrastructural

---

* Born C. Ernemann.

solution, and is starting to replace traditional parallel environments. However, key features of grids are still ardent research subjects, e.g., sophisticated resource planning strategies or the adaptation of existing applications to grids. Many of these features require in-depth knowledge of the behavior of grids, and realistic performance evaluation and comparison of existing and new approaches.

Grid performance evaluation raises very different challenges for the procedure and the adoption aspects. Also, the motivation of an evaluation may have a major impact on the approach that is taken during the evaluation itself. The various existing approaches to tackle the performance evaluation problem in the area of parallel environments [26, 49] cannot be directly applied in grids, due to the grids' dynamic and large-scale nature. Other grid-oriented approaches, though valuable, either do not use realistic workloads [9] or use non-validated measurement data as input for the evaluation process [35], and cannot be used for reliable system comparisons and evaluations, cf. [7, 23, 11]. Furthermore, the actual adoption of an evaluation procedure as a benchmark is a community approach which requires the agreement of a sufficient number of grid stakeholders; this hinges on the existence of one or more established procedures, currently lacking in grids.

Besides that, performance evaluation and comparison require the existence of workload traces within a grid, which at the moment simply do not exist. To ameliorate the lack thereof, synthetic, generated on the foundation of an appropriate model, that is, workloads are used for evaluation purposes. The main and, in fact, very hard problem is obviously to create a good model without having any workload instances (real system traces, that is). While there exist good models in the parallel processing community, there is no comprehensive workload model for grids available.

This paper aims to provide a starting point for grid performance evaluation from a practical point of view: a selection of requirements, objectives, and guidelines (including both well-known metrics from parallel workload modeling and newer, more grid-specific measurement gauges) is suggested to give an overview of what could be considered within a Grid performance evaluation system, and steps towards a common framework for adoption in real-world environments for the purpose of verification, analysis and benchmarking are shown. Our main contribution is thus twofold: (1) our approach is the first to deal programmatically with different critical grid modeling issues like co-allocation, job flexibility, data management, and failures; and (2) we gauge our approach as a standardization effort, by providing the necessary theoretical framework, and an early toolset to work with it.

The remainder of the paper is organized as follows. Section 2 presents three different evaluation scenarios. Section 3 analyzes a set of typical performance objectives which are commonly used in grids. Sections 4 and 5 focus on the features and requirements for modeling workloads for grids. While the modeling aspects in Section 4 contain strong links to existing work from the High-Performance Computing (HPC) community, Section 5 discusses in detail a selection of six grid-specific aspects: computation management, data management, locality/origin

management, failure modeling, and economic modeling. Section 6 describes the GRENCHMARK system, its current status, and the foreseen extensions towards the additional requirements presented in this paper. The discussion in Sections 4 and 5 acts as a guideline for added functionality to the GRENCHMARK framework. We conclude with a brief summary and a preview of our future research in this area, in Section 7.

2 System 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: 1. What is the actual scenario for which performance evaluation is done? and 2. 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[3] 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 achieveable 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 (a) support all of them and (b) allow a comparison of results on a technical level.

---

[3] Of course, the description of the simulation environment can also influence evaluation, but this discussion is out of the scope of this work.

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 destabilises 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 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 4 and 5.

3 Performance Metrics

The evaluation of the Grid performance highly depends on the given scenario (see Section 2), and the provider and user objectives. However, some typical standard evaluation metrics exist that can be applied in most cases. In this section we shortly present many of these metrics, and propose a selection of metrics for general purpose use.

Although we base the evaluation of grid systems on the seminal work in the context of parallel production environments by Feitelson et al. [24, 23], our notation is in some cases modified according to the standards defined for operational research scheduling by Graham et al. [27]. For an overview of this notation we refer to [41]. Rooting our work in these approaches enables us to build on established results, and to have a good base of comparison with previous performance evaluations.

Within the Grid we assume m machines[4] and a job system τ. Within the system, each job $j \in \tau$ can further be divided into tasks $k \in j$. The number of jobs in the system is $\mid \tau \mid$; the number of tasks for job j is $\mid j \mid$. Sometimes, such tasks are modeled as individual jobs that are connected by precedence constraints; especially then, the tasks of a job j are not executed in parallel.

Each job j and all its corresponding tasks $k \in j$ are released at time r_j. Grids typically work in an online scenario, that is, r_j is not known in advance for most jobs and tasks. As they arrive, jobs are scheduled to run, that is, a suitable set of resources is allocated for the future job run. For rescheduling capabilities, we define the *final schedule* for a period of time T as the schedule of all jobs arriving during time 0 and time T in which no job can be further rescheduled.

_____

[4] Note that this term is used loosely and may specify any type of resource.

3.1 Time-, Resource- and System-Related Metrics

Within the final schedule S, the task $k \in j$ is completed at time $C_k(S)$. Hence, job j leaves the system at time $C_j(S) = \max_{k \in j} C_k(S)$. The processing time of task $k \in j$ is p_k. Hence, the processing time p_j of job j can be calculated by $C_j(S) - \min_{k \in j}(C_k(S) - p_k)$. Besides the maximum lateness $L_{max} = \max_{j \in \tau}(C_j(S) - d_j)$, which may be used as an analysis criterion (and needs to be minimised for grid systems), the number of tardy jobs $\text{TJ} = \sum_{j \in \tau \wedge C_j > d_j} 1$ also is of interest, as it provides information about the number of user requests that could not be fulfilled.

The resource consumption RC_k of a task is defined by the product of the corresponding processing time and the used machines ($\text{RC}_k = p_k \cdot m_k$). Consequently, we can define the resource consumption of a job by $\text{RC}_j = \sum_{k \in j} \text{RC}_k$ and of a whole schedule by $\text{RC}(S) = \sum_{j \in \tau} \text{RC}_j$. Using this total resource consumption we can also define the utilization U of the available machines, see Equation 1. The resource provider usually[5] selects as objective the maximization of the system utilization.

$$U = \frac{RC(S)}{m \cdot (\max\limits_{j \in \tau} C_j(S) - \min\limits_{j \in \tau}(C_j(S) - p_j))} \tag{1}$$

With task execution failures being common in grids (see Section 5.6), jobs may fail during execution, and be run several times before they successfully complete. We therefore define the true resource consumption $\text{RC}_{\{}k,j\}^{true}$ and the true utilization U^{true} as corollaries respectively, so that also the failed job's consumption of resources is measured. The sum of the resource consumption of such faulty jobs is defined as the waste metric $\text{WASTE} = U^{true} - U$, which gives a hint on the dynamic reaction to failures of the grid system and is to be minimized by the resource owner.

As grid systems are belonging to several stakeholders, measuring the fairness of use is becoming an interesting point. A possible, but rather simple metric for measuring resource use fairness is the average wait time deviation [44], as defined in Equation 2; here, the objective is to minimize the $AWDT$ for each grid stakeholder.

$$AWTD = \frac{1}{\mid \tau \mid} \sqrt{\sum_{j \in \tau}(WT_j)^2 - (\sum_{j \in \tau} \frac{WT_j}{\mid \tau \mid})^2} \tag{2}$$

---

[5] In some cases (when certain users or user groups are willing to pay for the utilisation of a machine or have an affiliation to a certain organisation, etc.), the utilization might be of less importance.

3.2 Workload Completion and Failure Metrics

In nowadays grids, the ability to complete the execution of a given workload can be even more important than the speedup obtained through this execution[6]. Grids require the redefinition of the *application failure* notion: a grid application which was not able to finish successfully within its budget generates an application failure event upon the first detection of its inability to complete successfully. For example, an application fails if its requested computation resources cannot be found, because of having a deadline assigned, but exceeding it, or because of running out of credits (even during execution). Using this notion, *fault tolerance* becomes postponing the application failure as much as possible, while there are realistic chances of finishing the application, possibly to the point where the application finishes successfully. In this section we describe metrics pertaining to workload completion and failures.

We propose as a metric the workload completion (WC), computed as given by Equation 3. This helps to identify the limitations of the grid system, and its maximization should be used as a major objective both by the user and by the resource owners. However, the workload completion has limitations from the resource owners' point of view, as jobs with a smaller number of tasks have a higher influence on this value. As complementary metrics, we propose the task completion (TC), given by Equation 4, and the enabled task completion (ETC), given by Equation 5. Note that in the latter the enabled tasks are those tasks which can be run, after their previous tasks dependencies have been completed. The resource owners' objective is to maximize the enabled task completion. If the TC and the ETC metrics differ greatly, special care must be taken by the resource owners to fulfill critical tasks (tasks which are present in the dependency lists of many other tasks), for example by automatically launching redundant runs of these tasks.

$$\text{WC} = \frac{\sum\limits_{j \in \tau \wedge j \ completed} 1}{\mid \tau \mid} \tag{3}$$

$$\text{TC} = \frac{\sum\limits_{j \in \tau \wedge k \in j \wedge k \ completed} 1}{\sum\limits_{j \in \tau} \mid j \mid} \tag{4}$$

$$\text{ETC} = \frac{\sum\limits_{j \in \tau \wedge k \in j \wedge k \ completed} 1}{\sum\limits_{j \in \tau \wedge k \in j \wedge k \ enabled} 1} \tag{5}$$

We further propose as a metric the system failure factor (SFF), as the ratio between the number of failures observed for all the tasks that were started and the number of tasks that were started. Note that SFF is equal to $1 - ETC$

---

[6] Note that the jobs executed in grids may be much more complex than the jobs executed in traditional parallel environments, e.g., workflows vs. batches of jobs.

for a system with no retry capabilities, but may vary greatly otherwise. The SFF metric may be an effective performance evaluator for the ability of the grid system to detect and correct failures in the system, e.g., if a resource becomes unavailable, repeatedly sending jobs to it for execution would increase the number of observed failures, and prove a lack of dynamic response to partial system failures. The objective of the resource owner is minimize the value of the SFF metric. Note that it is possible to have a high value for the waste metric and a small value for the system failure factor at the same time, for instance when a few tasks fail, but their failure occurs or is observed after the tasks have been running for an extensive period of time. Besides this system-oriented metric, the *expected task failure*, that is, the probability that a task will fail upon being run, may be used to evaluate the performance of grids where the availability of resources is very dynamic [35].

3.3 Metrics Selection

Given the number of proposed metrics, the selection of an appropriate subset is still an open question. Recent works by Feitelson et al. show that *all* quantitative metrics should be reported and considered for a representative systems evaluation [21, 22]. Therefore, a scheduling performance evaluation could be done after considering the detailed resource consumption report, and the following aforementioned metrics: the system utilization **U**, the workload completion percentage **WC**, the enabled task completion **ETC**, the wasted resources **WASTE**, the system failure factor **SFF**, and the average wait time deviation **AWTD**. Besides that, we also consider the response time **AWRT**, the wait time **AWWT**, the slowdown **AWSD**, (all average), all used in their weighted versions, by which all jobs with the same resource demand have the same influence on the schedule quality. Without the weighting mechanism, a job using few machines would have the same impact on these metrics as a job that uses many machines, see Schwiegelshohn et al. [43]. To prevent this effect, bounds can be imposed for these metrics, e.g., bounded slowdown [24]. Specific time-based summaries of the consumption report and the nine metrics are sometimes needed, e.g., for normal, peak, and clear months, or for week days and week-end. Different providers will then be able to weight those metrics according to their system use scenario.

In some cases, metrics need also be computed per user or per user group, in addition to metrics for the full system. This may be needed, for example, for grids where the machine providers have different commercial relationships to different grid participants, and therefore specific objectives for different users or user groups [2]. An early example is the fair-share utilization concept used in the Maui scheduler [31], where separate policies are defined for different users and groups.

4   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 [37, 10]. 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[7] provide valuable modeling data. In this section we present the general aspects of HPC workload modeling.

4.1   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 [36], 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 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.

---

[7] Available at http://www.cs.huji.ac.il/labs/parallel/workload/.
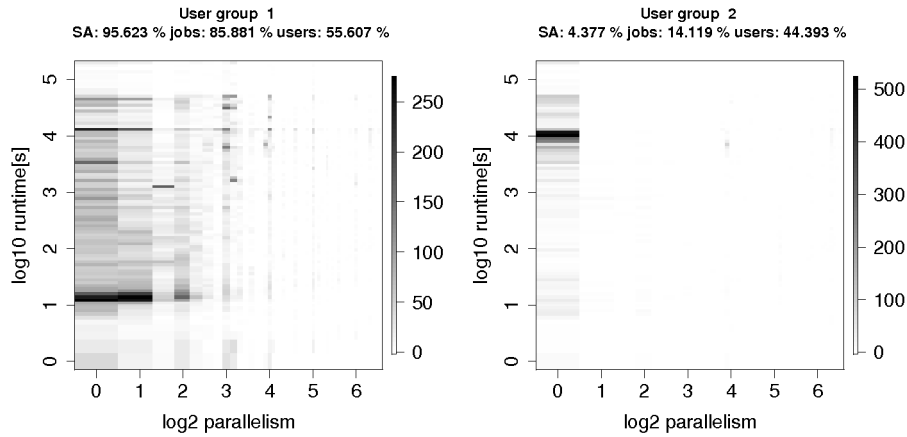
Fig. 1: Dominant set of groups of size 2 for the KTH workload.

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 [46] 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 Fig. 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 5.

4.2 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. Fig. 2 shows the daily arriving patterns of jobs, for the KTH workload. There is an obvious 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 [16]; however, they are still important to the local sites (and the local schedulers).
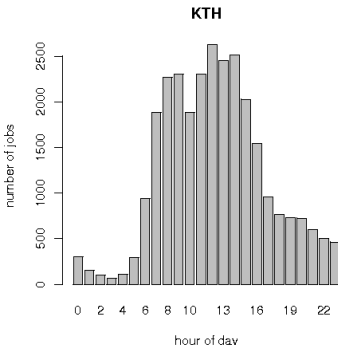
Fig. 2: Job arrivals during the daily cycle.

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 [36], or during short periods [8].

These effects can be described by classical statistical methods. For example, Downey [12] modeled the daily cycle using combined Poisson distributions; Calzarossa [6] 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 [18, 17]), e.g. sequential dependencies.

Therefore, temporal modeling is an important aspect. For example, one of these temporal effects is *repeated submission* [18], 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 [17]. 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 [45], 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 [39], 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 [19, 20]. Therefore, it is reasonable to model the users' sub-

missions 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.

5   Grid-specific Workload Modeling

In this section we present the grid-specific workload modeling requirements. Due to the lack of publicly available traces[8] of real grids operation, we restrict our presentation to the main characteristics that could become subject of near-future modeling.

5.1   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.

5.2   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

---

[8] 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.

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 intercluster communication low [4]; load-balancing across the number of sites can also be used for jobs requiring large numbers of resources [3]. 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 [40, 30]. However, no statistical data regarding the use of co-allocation by real communities of users is publicly available.

Besides that, *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 [24]. To model the application flexibility, at least one job shape (cf. [11], 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 [11]; experiments with moldable applications in a real environment have been presented by Berman et al. [1].

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.

5.3 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[9] the job can start, and

---

[9] 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 stored results must be fetched after 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 5.1), 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 [47].

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^{10} 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.

5.4 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 [25]. 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

---

^{10} Hierarchical Storage Management.

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 [50] 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 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 [42] and Tan et. al [48]. 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.

5.5 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 [31]. 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.

5.6 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* [32], 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 [7].

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. [38], 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. [35] 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 [9, 13, 30], invalid job specifications [34] or user-initiated cancellations [10] 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 [28]); the technique type then needs to be specified for each job or workflow in the workload, coupled with the specific technique parameters.

5.7 Economic Models

There is a lot of discussion on the connotation of access to grid resources not being free of charge [33, 14, 5]. 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. [14] and Buyya et al. [5].

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. [14], grid users provide parametric utility functions, and the systems performs automated request-offer matching.

6   GRENCHMARK: A Framework for Grid Performance Evaluation

GRENCHMARK is a framework for synthetic grid workload generation, execution, and analysis of execution results. It is *extensible*, in that it allows new types of grid applications to be included in the workload generation without a change in the design, *parameterizable*, as it permits the user to parameterize the workloads generation and submission, and *flexible*, as it can be used for analyzing, testing, and comparing common grid settings. GRENCHMARK is currently developed at TU Delft[11].

6.1   Current Features

In our previous work we have shown how GRENCHMARK can be used to generate and submit workloads comprising unitary and composite applications, to replay-scale-mix traces from the Parallel Workloads Archive, and in general to analyze, test, and compare common grid settings [29, 30]. Therefore, we only point out prominent features, and invite the reader to consult our work.

GRENCHMARK offers support for the following workload modeling aspects. First, it supports unitary and composite applications as well as single-site and co-allocated jobs. Second, it allows the user to define the job submission pattern based on well-known statistical distributions. Third, it allows the workload designer to combine several workloads into a single one (mixing). This allows for instance the definition of separate user groups (see Section 4.1), further combined into a single grid workload. Furthermore, it supports the generation, storage, modification, replay and analysis of workloads based on user-defined parameters.

---

[11] A reference implementation is available from http://grenchmark.st.ewi.tudelft.nl/.

6.2 Extension Points

GRENCHMARK has been designed with an incremental approach in mind, and facilitates future extensions at many levels.

Based on the dynamics of the grid workload generation process, we identify two types of grid workload generation: statically-generated workloads, and dynamically-generated workloads. *Statically-generated workloads* do not change at execution time with modifications in the execution environment. Currently, GRENCHMARK incorporates rather simple models (statistical distributions for submission patterns, for example, without correlations to other parameters or feedback functionality). However, due to the extensibility, more complex notions such as temporal models and parameters correlation (see Section 4.2), data and network management (see Sections 5.3 and 5.4), or locality/origin management (see Section 5.5) can be easily adapted. To introduce support for *dynamically-generated workloads* into GRENCHMARK, the framework design needs to be extended with the ability to react to system changes for both workload generation and submission. Since the reference implementation already uses feedback for the submission process (for composite job submission and reacting to execution errors [30]), the implementation of such functionality seems feasible and, as such, we plan to address this issue in future work.

From the perspective of operating in a dynamic system, GRENCHMARK can already respond to the situation when the background load can be extracted from existing traces and, as such, is known, and offers adequate modeling capabilities. For handling the background load as a separate workload, an extension is still required. For a variable background load in a real environment (the most difficult case), the desired load could, for example, be controlled by coupling GRENCHMARK to existing monitoring services. Then, dummy jobs can be launched to ensure a fixed level of background load during all experiments, as in Mohamed and Epema [40]. However, the modeling itself remains an open issue.

Another important extension issue is the use of GRENCHMARK in different system scenarios (cf. to Section 2). We have already used GRENCHMARK in real environments. For simulated environments, the reference implementation needs to be extended to event-based simulation, which is work in progress.

Summarizing, GRENCHMARK provides a framework for Grid performance evaluation which already contains basic modeling techniques, but needs to incorporate more sophisticated modeling capabilities in order to generate and analyse Grid workloads.

7 Conclusion

In this work we have presented an integrated approach for generic grid performance evaluation. For this purpose, we have first presented several grid performance objectives, and made a selection for the general case. We have then combined traditional and grid-specific workload modeling aspects, and synthesized the requirements for realistic grid workload modeling. Finally, we have pre-

sented an existing framework for workload generation and analysis and pointed out extension points on both modeling and infrastructure issues.

In order to validate our work with experimental results, we are currently working on extensions to the GRENCHMARK framework to accommodate the changes detailed in Section 6.2 with the goal to have a powerful, yet extensible framework for workload modeling and analysis. We hope that this work will become the jump-start for a common performance evaluation framework for grid environments.

8 Acknowledgements

References

1. 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.
2. N. Beume, M. Emmerich, C. Ernemann, L. Schï£¡nemann, and U. Schwiegelshohn. Scheduling Algorithm Development based on Complex Owner Defined Objectives. Technical Report CI-190/05, University of Dortmund, January 2005.
3. 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.
4. 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.
5. 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).
6. 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.
7. 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.
8. 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.

9. 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.

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

11. 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.

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

13. 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.

14. 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.

15. Carsten Ernemann, Volker Hamscher, Uwe Schwiegelshohn, Ramin Yahyapour, and Achim Streit. On advantages of grid computing for parallel job scheduling. In *Proc. of the 2nd IEEE Int'l. Symp. on Cluster Computing and the Grid (CCGrid)*, pages 39–49, 2002.

16. 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.

17. 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.

18. Dror G. Feitelson. Packing Schemes for Gang Scheduling. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1162 of *Lecture Notes in Computer Science*, pages 89–110. Springer, 1996.

19. 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.

20. 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.

21. Dror G. Feitelson. Metric and workload effects on computer systems evaluation. *IEEE Computer*, 36(9):18–25, 2003.

22. Dror G. Feitelson. Experimental analysis of the root causes of performance evaluation results: a backfilling case study. *IEEE Transactions on Parallel and Distributed Systems*, 16(2):175–182, Feb 2005.

23. 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.

24. 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.

25. 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.

26. 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.

27. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 15:287–326, 1979.

28. 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.

29. A. Iosup, J. Maassen, R.V. van Nieuwpoort, and D.H.J. Epema. Synthetic grid workloads with Ibis, KOALA, and GrenchMark. In *Proceedigs of the CoreGRID Integrated Research in Grid Computing, Pisa, Italy*, November 2005.

30. 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).

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, 2nd edition, 2002.

42. 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.

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

44. Hongzhang Shan, Leonid Oliker, and Rupak Biswas. Job superscheduler architecture and performance in computational grid environments. In *SC*, pages 44–54. ACM, 2003.

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

46. Baiyi Song, Carsten Ernemann, and Ramin Yahyapour. User Group-based Workload Analysis and Modeling. In *Proc. of the 5th Int'l. Symp. on Cluster Computing and the Grid (CCGrid)*. IEEE Computer Society, 2005.

47. 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.

48. 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.

49. 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.

50. 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.