The importance of complete data sets for job scheduling simulations

Dalibor Klusáček and Hana Rudová

Faculty of Informatics, Masaryk University
Botatanická 68a, Brno, Czech Republic
`{xklusac,hanka}@fi.muni.cz`

Abstract. This paper has been inspired by the study of the complex data set from the Czech National Grid MetaCentrum. Unlike other widely used workloads from Parallel Workloads Archive or Grid Workloads Archive, this data set includes additional information concerning machine failures, job requirements and machine parameters which allows to perform more realistic simulations. We show that large differences in the performance of various scheduling algorithms appear when these additional information are used. Moreover, we studied other publicly available workloads and partially reconstructed information concerning their machine failures and job requirements using statistical and analytical models to demonstrate that similar behavior is also expectable for other workloads. We suggest that additional information about both machines and jobs should be incorporated into the workloads archives to allow proper and more realistic simulations.

Key words: Grid, Cluster, Scheduling, MetaCentrum, Workload, Failures, Specific Job Requirements

1 Introduction

Large computing clusters and Grids have become common and widely used platforms for the scientific and the commercial community. Efficient job scheduling in these large, dynamic and heterogeneous systems is often a very difficult task [32]. Development or application of an efficient scheduling algorithm requires a lot of testing and evaluation before such solution is applied in the production system. Due to several reasons, such as the cost of resources, reliability, varying background load or the dynamic behavior of the components, experimental evaluation cannot be usually performed on the real systems. Many simulations with various setups that simulate different real-life scenarios must be performed using the same and controllable conditions to obtain reliable results. This is hardly achievable in the production environment.

Usually, workload traces from the Parallel Workloads Archive (PWA) [6] or Grid Workloads Archive (GWA) [4] are used as the simulation inputs. However, these data do not contain several parameters that are important for realistic simulations. Typically, very limited information is available about the Grid or cluster resources such as the architecture, the CPU speed, the RAM size or the resource specific policies. However, these parameters often significantly influence the decisions and performance of the scheduler. Moreover, no information concerning background load, resource failures, or specific users' requests are available. In heterogeneous environments, users often specify some subset of machines or clusters that can process their jobs. This subset is usually defined either by the resource owners' policy (user is allowed to use such cluster), or by the user who requests some properties (library, software licence, execution time limit, etc.) offered by some clusters or machines only. Also, the combination of both owners' and users' restrictions is possible. When one tries to create a new scheduling algorithm and compare it with current approaches such as EASY Backfilling [28] or algorithms used in, e.g., PBSpro [15], LSF [33] or Moab [2], all such information and constraints are crucial, since they make the algorithm design much more complex. If omitted, resulting simulation may provide misleading or unrealistic results as we show in Section 6.

So far, we have been able to collect complete real-life data set from the Czech national Grid infrastructure MetaCentrum [23] that covers many previously mentioned issues, such as machine parameters and supported properties, specific job requirements or machine failures. Using this complete data set [16] we were able to perform more realistic simulations. We have studied behavior of several objective functions that cover typical requirements such as the average job slowdown, the average response time, or the average wait time. We have compared schedule-based algorithms involving Local Search [32] which we have been developing for couple of years [18, 19], as well as queue-based solutions such as FCFS or EASY and Conservative Backfilling. In our experiments, we have focused on two scenarios. The first (BASIC) scenario does not involve machine failures. Moreover, all jobs can be executed on any cluster (if enough CPUs are available), representing the typical amount of information available in the GWA or the PWA workloads. The second (EXTENDED) scenario uses additional information available in the MetaCentrum data set such as machine failures or additional cluster and job properties that define the job-to-cluster suitability (specific job requirements). As observed during the experiments (see Figure 2), the differences in the values of objective functions between these two scenarios are often large.

While the effects of machine failures on the cluster [20, 34, 27] or the Grid [20, 12] performance are widely discussed, we are not aware of similar works that would also cover the effects of specific job requirements. Therefore, inspired by our own interesting results, we have decided to perform more research and analysis of existing workloads. When it was possible, we tried to recover additional information "hidden" in the available data covering both machine failure intervals and job requirements. When informations were insufficient we carefully generated synthetic machine failures using a statistical model. Once created, these "extended" workloads were compared through experiment with their original simpler versions. As expected, we often discovered the disproportions in the values of objective functions similar to the differences for the MetaCentrum data set. This supports our idea that scheduling algorithms should be evaluated using complete data sets.

The paper is organized as follows. First, we define the studied problem. Next, we discuss the PWA and GWA workloads, known failure traces and characteristics of considered workloads. The model used to create extended workloads is introduced and considered scheduling algorithms are described. We provide the detailed experimental evaluation with discussion of results and conclude our paper with a short summary.

2 Problem Description

In this section we describe the investigated job scheduling problems, starting with the simpler BASIC and followed by the EXTENDED problem. These problems are specified by characteristics of considered machines and jobs. We also define the optimization criteria considered for the evaluation of generated solutions.

2.1 BASIC Problem

The system is composed of one or more computer clusters and each cluster is composed of several machines. So far, we expect that all machines within one cluster have the same parameters. Those are the number of CPUs per machine and the CPU speed. All machines within a cluster use the Space Slicing processor allocation policy [9] which allows the parallel execution of several jobs at the cluster when the total amount of requested CPUs is less or equal to the number of CPUs of the cluster. Therefore, several machines within the same cluster can be co-allocated to process the given parallel job. On the other hand, machines belonging to different clusters can not be co-allocated to execute the same parallel job.

Job represents a user's application. Job may require one (sequential) or more CPUs (parallel). Also the arrival time and the job length are specified. There are no precedence constraints among jobs and we consider neither preemptions of the jobs nor migrations from one machine to another. When needed, the runtime estimates are precise (perfect) in this study.

2.2 EXTENDED Problem

This scenario extends the BASIC problem with some more features that are based on the characteristics of the MetaCentrum Grid environment. First of all, each cluster has additional parameters that closely specify its properties. These parameters typically describe the architecture of the underlying machines (Opteron, Xeon, . . .), the available software licenses (Matlab, Gaussian, . . .), the operating system (Debian, SUSE, . . .), the list of queues allowed to use this cluster (each queue has a maximum time limit for the job execution, e.g., 2 hours, 24 hours, 1 month), the network interface parameters (10Gb/s, Infiniband, . . .), the available file systems (nfs, afs, . . .) or the cluster owner (Masaryk University, Charles University, . . .). We expect that all the machines within one cluster have the same parameters.

Corresponding information is often used by the user to closely specify job's characteristics and requirements. Those are typically the time limit for the execution (given by the queue or user), the required machine architecture, the requested software licenses, the operating system, the network type or the file system. Users may also directly specify which cluster(s) is suitable for their jobs. In another words, by setting these requirements, user may prevent the job from running on some cluster(s). In real life, there are several reasons to do so. Some users strongly demand security and full control and they do not allow their jobs (and data) to use "suspicious" clusters which are not managed by their own organization. Others need special software such as Matlab or Gaussian which is not installed everywhere. Some clusters are dedicated for short jobs only (2 hours limit) and a user wanting more time is not allowed to use such cluster, and so on. All these requests are often combined together. In the EXTENDED problem all such requirements have to be included into the decision making process to satisfy all specific job's requirements. If no suitable machine is found, the job has to be cancelled. Clearly, the specific job requirements cannot be used when the corresponding cluster parameters are not known. Without them, consideration of "job-to-machine" suitability is irrelevant. Therefore, whenever the term *specific job requirements* is referenced in this paper, it means that both additional job and cluster parameters are applied, decreasing the number of suitable clusters for the job execution.

Finally, machine failures are considered in the EXTENDED scenario. It means that either one or more machines within a cluster are not available to execute jobs for some time period. Such failure may be caused by various reasons such as the power failure, the disk failure, the software upgrade, etc. However, we do not differentiate between them in this study. As a result of the failure, all jobs that have been — even partially — executed on such machine are immediately killed. Once the failure terminates, machine is restarted and becomes available for the job processing.

2.3 Evaluation Criteria

The quality of the generated solutions can be reflected by various types of optimization criteria. In both scenarios the following objective functions are considered: the avg. response time [9], the avg. slowdown [9] and the avg. wait time [5]. In addition, the total runtime of the scheduling algorithm [19] is measured as a standard evaluation criteria. If machine failures are used we also count the total number of killed jobs. The avg. response time represents the average time a job spends in the system, i.e., the time from its submission to its termination. The avg. slowdown is the mean value of all jobs' slowdowns. Slowdown is the ratio of the actual response time of the job to the response time if executed without any waiting. Avg. wait time is the time that the job spends waiting before its execution starts. As pointed out by Feitelson et al. [9], the use of response time places more weight on long jobs and basically ignores if a short job waits few minutes, so it may not reflect users' notion of responsiveness. Slowdown reflects this situation, measuring the responsiveness of the system with respect to the job length, i.e., jobs are completed within the time proportional to the job length. Wait time criterion supplies the slowdown and response time. Short wait times prevent the users from feeling that the scheduler "forgot about their jobs". All preceding objectives consider successfully executed jobs only, since killed jobs are not included. On the other hand, the total number of killed jobs is always monitored when machine failures are used. Finally, the total runtime of the scheduling algorithm measures the total CPU time needed by the scheduling algorithm to develop the solution for a given experiment.

3 Existing Sources of Workloads and Failure Traces

Two main publicly available sources of cluster and Grid workloads exist. Those are the Parallel Workloads Archive (PWA) [6] and the Grid Workloads Archive (GWA) [4]. There are two major differences between them. First of all, PWA maintains workloads coming from one site or cluster only (with few exceptions), while each workload in the GWA covers several sites. Second, the Grid Workloads Format (GWF) [13] is an extension to the Standard Workloads Format (SWF) [1] used in the PWA, reflecting some Grid specific job aspects. For example, each job in the GWF file has an identifier of the cluster where the job was executed. Moreover, the GWF format contains several fields to store specific job requirements. However, none of the six currently available traces uses them. These archives also often lack detailed and systematic description of the Grid or cluster resources, where the data were collected. Beside the real workloads, various models for generating synthetic workloads were proposed and implemented [31, 22, 8].

Traces of different kinds of failures related to the computer environment are collected in several archives such as the Repository of Availability Traces (RAT) [25] or the Computer Failure Data Repository (CFDR) [3]. For our purposes, the most suitable is the Failure Trace Archive (FTA) [20], that — among others — currently stores two Grid and cluster failure traces. Those are the Grid'5000 and the LANL traces. Remaining Grid or cluster related traces are either incomplete (PNNL) or were not yet converted from their original "raw" formats (EGEE, NERSC, HPC2, HPC4)[1]. FTA contains description of nodes but does not contain the information about jobs.

The complete MetaCentrum data set is publicly available at `http://www.fi.muni.cz/~xklusac/workload`. It contains trace of 103,620 jobs that includes specific job requirements as well as description of 14 clusters (806 CPUs) with the information about machine architecture, CPU speed, memory size and the supported properties. Also, the list of available queues including their priorities and associated time limits is provided. There is the trace of machine failures

[1] In December 2009.

and the description of temporarily unavailable machines that were reserved or dedicated for special purposes. The average utilization of MetaCentrum varies per cluster with overall utilization being approximately 55%. In this work, we simulate neither reserved nor dedicated machines and we focus strictly on the problem involving machine failures and specific job requirements. Therefore, the overall machine utilization has decreased to approximately 43% in our experiments[2].

Using these data sources we have selected three candidate workloads that have been used for the evaluation. Certainly the MetaCentrum workload was used as our base data set. Next, two more workloads were selected and carefully extended to obtain all information necessary for the EXTENDED problem. Methodologies used to generate such workloads are described in the next section. The SWF workload format does not contain information about job execution site, which is needed when generating extended workloads. Therefore, we were left with the GWA that contains six workloads now. However, three of them contain only sequential jobs, thus three candidates remained[3]: Grid'5000, DAS-2 and Sharcnet. Sadly, we had to eliminate Sharcnet since it does not provide enough information to generate the workload for the EXTENDED problem (see Section 4.2).

Grid'5000 is an experimental Grid platform consisting of 9 sites geographically distributed in France. Each site comprises one or several clusters, there are 15 clusters in total. The trace contains 1,020,195 jobs collected from May 2005 till November 2006. The total number of machines is not provided with the trace, therefore we had to reconstruct it from the job trace and information about machine failures available in the Grid'5000 failure trace. Then, we were able to determine the probable number of machines for each cluster. Totally, there has been approximately 1343 machines (2686 CPUs). Grid'5000 has a low average utilization being only 17%. On the other hand, there is a publicly available failure trace for Grid'5000 in the FTA, which is very convenient for our experiments. Sadly, all fields corresponding to specific job requirements are empty in the workload file.

DAS-2 (Distributed ASCI Supercomputer 2) workload trace comes from a wide-area Grid composed of 200 Dual Pentium-III nodes (400 CPUs). The Grid is built out of 5 clusters of workstations located at five Dutch Universities. Trace contains 1,124,772 jobs collected from February 2005 till December 2006. The workload has a very low utilization of approximately 10%. There is no failure trace available and the workload trace contains no specific job requirements.

Finally, Table 1 presents the main characteristics of PWA, GWA, FTA and MetaCentrum archives.

[2] Machines dedicated or reserved for special purposes are considered as 100% utilized.

[3] If all jobs are sequential (non-parallel), then all scheduling algorithms considered in this paper follow more or less the FCFS approach.

Table 1. Main characteristics of PWA, GWA, FTA and MetaCentrum archives.

	PWA	**GWA**	**FTA**	**MetaCentrum**
job description	Yes	Yes	No	Yes
machine description	Partial	Partial	Yes	Yes
failures	No	No	Yes	Yes
specific job requirements	No	Partial	No	Yes

4 Extending the BASIC problem

In this section we describe the main methods used to generate the synthetic machine failures and the specific job requirements. Using them, the Grid'5000 and the DAS-2 workloads were enriched towards the EXTENDED problem.

4.1 Machine Failures

Since both MetaCentrum and Grid'5000 data sets contain real machine failure traces, only the DAS-2 workload has been extended by the synthetic machine failures. First of all, the original DAS-2 workload was analyzed to determine the first time when each cluster was used by some job. As is shown in the Figure 1 (top left), only the DAS2/fs0 cluster was used from the beginning, four remaining clusters started to execute jobs approximately three months later. We have used this observation to generate four "initial cluster failures" that cover the time before the cluster was (probably) operational.

Next, the synthetic machine failures were generated. Five main parameters had to be determined. First of all, the total number of failures (F) has been established. Then, for each failure, four parameters have been chosen: failure duration, failure arrival time and the cluster and its machine that will exhibit this machine failure.

When solving these problems we were inspired with the model proposed by Zhang et al. in [34] and findings in [27, 12, 26]. We also used three available failure traces (MetaCentrum, Grid'5000, LANL) to get the necessary statistical data. As discussed in [27], failure rates are roughly proportional to the number of nodes (N) in the system. Therefore, the total number of failures (F) was computed as:

$$F = N \cdot D \cdot AFC \tag{1}$$

where D is the duration of DAS-2 workload in hours, and the AFC (Average Failure Count) is the average number of failures per machine per hour. While N and D were known, AFC had to be selected. Since we have no information concerning the real failure rates in DAS-2, we used known failure traces to compute the AFC values. Figure 1 (top middle) shows the AFC values for known failure traces. Grid'5000 shows suspiciously high AFC value, which is probably caused by the fact that some reported failures are rather "false alarms" than

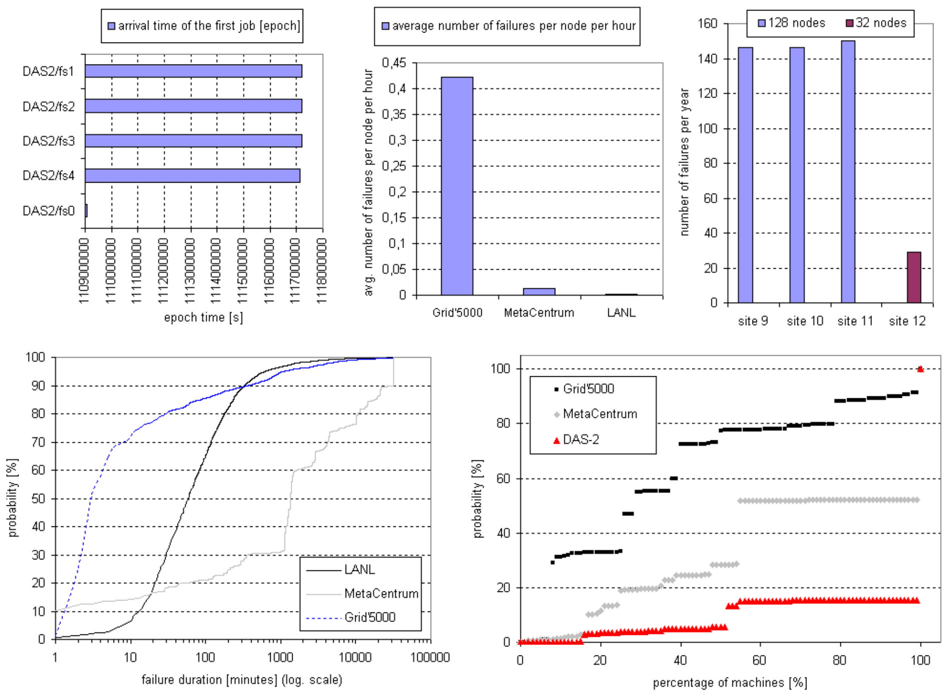

Fig. 1. The cluster start times in DAS-2 (top left), the average number of failures per node per day (AFC) (top middle), the number of failures per year in LANL (top right), the CDFs of failure durations for Grid'5000, LANL and MetaCentrum (bottom left) and the CDFs of "suitability distribution" of jobs onto clusters (bottom right).

actual failures as discussed in [20]. In MetaCentrum, the AFC value is much more smaller while the low failure rates of LANL result in the lowest observed AFC. Since the large amount of failures in Grid'5000 is suspicious we have chosen LANL's and MetaCentrum's AFC values as two possible inputs into our failure generator. This resulted in two different failure traces for DAS-2. In the remaining text, DAS-2-L represents DAS-2 workload with failure trace generated using the LANL-based parameters while DAS-2-M represents solution based on the MetaCentrum parameters.

Next, the remaining parameters were generated using the model [34] of Zhang et al. This involved the use of Weibull distribution [14] to generate the inter-arrival times between failures. Sahoo et al. [26] discussed that there are strong temporal correlations between failure arrivals. Using the model of Zhang et al., this behavior was simulated by including so called "failure bursts", i.e., multiple failures on one cluster appearing in (almost) the same time. Failure durations for DAS-2-L and DAS-2-M were generated using the Weibull distribution. Parameters of the distribution were selected by fitting the shape of Weibull

Cumulative Distribution Function (CDF) [14] to the original CDFs of LANL and MetaCentrum failure durations that are shown in Figure 1 (bottom left)[4]. These CDFs represent the probability that the duration of machine failure will be less than or equal to x minutes.

The distribution of failures between clusters was done using the observations of LANL's failure distribution pattern shown in Figure 1 (top right) which has been closely discussed in [27]. Here, clusters with the same hardware and age have their failure rates roughly proportional to the number of machines within the cluster. This indicates that failure rates are not growing significantly faster than linearly with the cluster size [27]. Figure 1 (top right) shows this behavior for sites 9, 10, 11 and 12 in LANL. According to the available data, all DAS-2 clusters are based on the same hardware, therefore we have used the same linear distribution of failures in our failure generator.

Finally, the distribution of failures on the machines within one cluster was analyzed. Several authors show that such distribution is not uniform for a given cluster [26, 10]. However, our own analysis of MetaCentrum failure trace showed that this is not always true. In MetaCentrum, some clusters had rather uniform failure distribution while for others it was highly unbalanced, showing significantly different shapes per each cluster. Since we have no reliable information about the type or shape of the desired distribution, we have decided to use simple uniform distribution in this case.

4.2 Specific Job Requirements

As far as we know there is no available model to simulate specific job requirements. Moreover, the only workload we are aware of that contains such information is the MetaCentrum workload. Our goal was to recreate such information for both DAS-2 and Grid'5000 workloads. Since it would be highly unreliable to simply transform known MetaCentrum pattern on different workloads, we have decided to use more realistic and conservative approach when establishing these requirements. Our approach is based on the analysis of the original DAS-2 and Grid'5000 workloads. In both of them each job contains identifier of the type (name) of the application that was used to execute the job as well as the identifier of the target cluster where it was finally executed [13]. Using this information, we could easily reveal the actual mapping of applications (jobs) on the clusters. To be more precise, we constructed a list of clusters where jobs having the same application identifier were executed. Next, during the simulation the application identifier is detected for each job and the corresponding clusters from the list are taken to be the only suitable execution sites for the job. Since we have no other information concerning job requirements, we used this mapping as the model of specific job requirements. Resulting CDFs based on such distributions are shown for the Grid'5000 and the DAS-2 workloads in Figure 1 (bottom right) together with the known distribution of MetaCentrum. Here, each CDF represents the

[4] The CDF of LANL is smoother since it was reconstructed from higher number of known failure durations. The x-axis is in log. scale.

probability that the job will be executable on at most $x\%$ of available machines. As it has been briefly mentioned in Section 3, this approach is not applicable for the Sharcnet workload, where the number of job identifiers is almost the same as the number of jobs in the workload. Thus, similar statistics did not make any sense and Sharcnet has not been used.

Table 2 summarizes the origins of all extensions of the original workloads. Presented generator of machine failures and specific job requirements can be downloaded at `http://www.fi.muni.cz/~xklusac/generator`.

Table 2. Origin of machine failures and specific job requirements for the EXTENDED problem.

	MetaCentrum	**Grid'5000**	**DAS-2**
machine failures	original data	original data	synthetic DAS-2-M
			synthetic DAS-2-L
specific job req.	original data	synthetic by workload analysis	synthetic by workload analysis

5 Scheduling Algorithms

Scheduling was performed by simulated centralized scheduler [32] that managed target clusters using different algorithms. We have used FCFS, EASY backfilling (EASY) [28] and Conservative backfilling (CONS) [7, 29] optionally optimized with a Local Search (LS) algorithm [18, 19]. EASY backfilling is an optimization of the FCFS algorithm, focusing on maximizing the system utilization. When the first (oldest) job in the queue cannot be scheduled because not enough processors are available, it calculates its earliest possible starting time using the runtime estimates of running jobs. Finally, it makes a reservation to run the job at this pre-computed time. Next, it scans the queue of waiting jobs and schedules immediately every job not interfering with the reservation of the first job. While EASY makes reservation for the first job only, Conservative backfilling makes the reservation for every queued job. These reservations represent an execution plan. We call this plan *the schedule* [11]. This schedule is updated whenever a new job arrives or some job completes its execution. Moreover, it allows us to apply advanced scheduling algorithms to optimize the schedule. This is the goal of the LS optimization procedure. LS maintans the schedule and optimizes its quality. New jobs are added to the schedule using CONS, i.e., they are placed to their earliest starting time. LS is run periodically and it consists of several iterations. In each iteration, random waiting job is selected and removed from the schedule and a new reservation is chosen randomly either on the same cluster or on a different one. Other reservations are updated with respect to this new assignment. Typically, when the original reservation is cancelled, later reservations can be shifted to the earlier start times. Analogically, new reservation can collide with existing reservations. If so, these are shifted to the later start times. Then, the *new schedule* is evaluated using the *weigth* function W which is defined by Equation 2.

$$\begin{aligned}
w_{sld} &= (sld_{previous} - sld_{new})/sld_{previous} \\
w_{wait} &= (wait_{previous} - wait_{new})/wait_{previous} \\
w_{resp} &= (resp_{previous} - resp_{new})/resp_{previous} \\
W &= w_{sld} + w_{wait} + w_{resp} \qquad (2)
\end{aligned}$$

W is a sum of three decision variables w_{sld}, w_{wait} and w_{resp} which are computed using the avg. job slowdown, avg. job wait time and avg. job response time of the previous and new schedule. They express the percentage increase or decrease in the quality of the new schedule with respect to the previous schedule. A positive value represents an improvement while a negative means that the new schedule represents a worse solution. Obviously, some correction is needed when the $wait_{previous}$ or $resp_{previous}$ is equal to zero but it is not presented to keep the code clear[5]. The final decision is based on the W value. If the W is greater than 0, then the new schedule is accepted, otherwise it is rejected and the schedule returns to the previous state. Iterations continue until the predefined number of iterations or the given time limit is reached. When applied, LS is executed every 5 minutes of simulation time. Here we were inspired by the actual setup of the PBSPro scheduler [15] used in the MetaCentrum which performs priority updates of jobs waiting in the queues with a similar periodicity. The maximum number of iterations is equal to the number of currently waiting jobs (schedule size) multiplied by 2. The maximum time limit was set to be 2 seconds which is usually enough to try all iterations. At the same time, it is still far less than the average job inter-arrival time of the densest DAS-2 trace (50 seconds). Since LS uses random function in each of its iteration, all experiments involving the LS algorithm have been repeated 10 times using different seeds, their results have been averaged and the standard deviation computed.

FCFS, EASY Backfilling and Conservative Backfilling are usually applied to schedule jobs on one cluster only. Since our data sets are all multi-cluster, algorithms have been extended to allow multi-cluster scheduling. This extension is very simple. FCFS, EASY and CONS simply check each cluster separately, finding the earliest possible reservation. If multiple choices to execute the job appear, the fastest available cluster is selected. If all available clusters have the same speed, the first choice is taken[6].

Next extension defines algorithms' reactions in case of a machine failure or restart. The simplest extension is made in FCFS. Here, machine failure or restart simply changes the set of CPUs to be used by the FCFS. Similar case applies for EASY, CONS and LS. However, machine failure may collide with existing reservations made by EASY, CONS or LS. If so, different actions are taken for

[5] By definition, slowdown is always greater or equal to 1.

[6] Clusters are ordered according to the total number of CPUs in descending order.

EASY, CONS and LS. EASY checks whether the reservation of the first job is still valid. If not, it creates a new one. Since CONS and LS make reservation for every job it is more probable that collisions will appear. In our implementation, all jobs having reservations on the cluster where the machine failure occurred are re-backfilled using CONS. Other jobs' reservations are not changed since the total re-computation of all reservations for all clusters is very time consuming as we have have observed in our initial tests. If there are many machine failures, some highly parallel jobs may not be able to get a reservation, since there is not enough CPUs available in the system. If so, these jobs are canceled and removed from the queue since their huge wait times would distort the simulation results. Jobs killed due to a machine failure are not resubmitted. Upon a machine restart, both FCFS and EASY try to use new CPUs immediately. CONS and LS behave somehow different since they have a reservation for every job at that moment. All reservations on the cluster where the machine restart appeared are recreated to utilize the restarted machine. Again, only the jobs having a reservation on such cluster are re-backfilled to minimize the algorithm's runtime. Reservations of jobs on remaining clusters are not changed. It may result in a temporally unbalanced distribution of jobs, since the re-backfilled jobs may not utilize all CPUs, especially if many machines restarted at the same moment. Such CPUs can be potentially suitable for jobs having reservations on different clusters. However, this imbalance is only temporal as new job arrivals or LS optimization will quickly saturate the available CPUs.

The inclusion of specific job requirements is very simple. All scheduling algorithms will allow job's execution on some cluster(s) if and only if the cluster(s) meets all specific job requirements.

6 Evaluation

All simulations were performed using the GridSim [30] based Alea simulator [17] on an Intel QuadCore 2.6 GHz machine with 2048MB of RAM. We have compared values of selected objective functions and the algorithms' runtime when the original (BASIC problem) and extended workloads (EXTENDED problem) have been used, respectively. As mentioned in Section 2, BASIC problem does not involve machine failures and specific job requirements while the EXTENDED does. In order to closely identify the effects of machine failures and specific job requirements on the values of objective functions, we have considered three different problems using the extended workloads. In EXT-FAIL only the machine failures are used and the specific job requirements are ignored. EXT-REQ represents the opposite problem, where the failures are ignored and only the specific job requirements are simulated. Finally, EXT-ALL uses both machine failures and specific job requirements. Using these setups, four different experiments were conducted for MetaCentrum and Grid'5000: BASIC, EXT-FAIL, EXT-REQ and EXT-ALL. Since DAS-2 has two variants of failure traces (DAS-2-L and DAS-2-M), there are six different experiments for the DAS-2 workload: BASIC, EXT-FAIL-L, EXT-FAIL-M, EXT-REQ, EXT-ALL-L and EXT-ALL-M, where "-L" or "-M" suffix specifies whether DAS-2-L or DAS-2-M failure trace has been used. Table 3 summarizes all data sets and problems we have considered in our experiments.

Table 3. Overall summary of workloads, problems and performed experiments.

	MetaCentrum	**Grid'5000**	**DAS-2**
BASIC	BASIC	BASIC	BASIC
EXTENDED	EXT-FAIL	EXT-FAIL	EXT-FAIL-L
			EXT-FAIL-M
	EXT-REQ	EXT-REQ	EXT-REQ
	EXT-ALL	EXT-ALL	EXT-ALL-L
			EXT-ALL-M

We start our discussion with the MetaCentrum workload where all information needed to simulate the EXTENDED problem were known, thus these results are the most reliable (see Figure 2). Next, we continue with the Grid'5000 (see Figure 3), where the EXTENDED problem was created using known data from the Failure Trace Archive (machine failures) and synthetically generated specific job requirements. Finally, Figure 5 presents results for the DAS-2 where additional data for the EXTENDED problem were generated synthetically. Resulting values of the avg. slowdown [9], the avg. response time [9], the avg. wait time [5] and the number of killed jobs are discussed for all experiments outlined in Table 3. Also the effects of inclusion of machine failures and specific job requirements on the total runtime of the scheduling algorithm [19] are discussed.

6.1 MetaCentrum Workload

Figure 2 shows the results for the MetaCentrum workload. As we can see the highest differences in the values of objective functions appear between BASIC and EXT-ALL experiments, which correspond with our expectations. In the case of BASIC, the differences between all algorithms are not very large for all considered objectives. On the other hand, when the EXT-ALL problem is applied, large differences start to appear for all criteria. It is most significant in the case of FCFS which generates the worst results among all applied algorithms. The values of FCFS are truncated for better visibility. EASY and CONS perform much better, while the best results are achieved by LS in most cases. Interesting results are related to the EXT-FAIL and EXT-REQ scenarios. As we can see, the inclusion of machine failures (EXT-FAIL) has usually asmaller effect than the inclusion of specific job requirements (EXT-REQ). Clearly, it is "easier" to deal with machine failures than with specific job requirements when the overall system utilization is not extreme. In case of a failure, the scheduler has usually

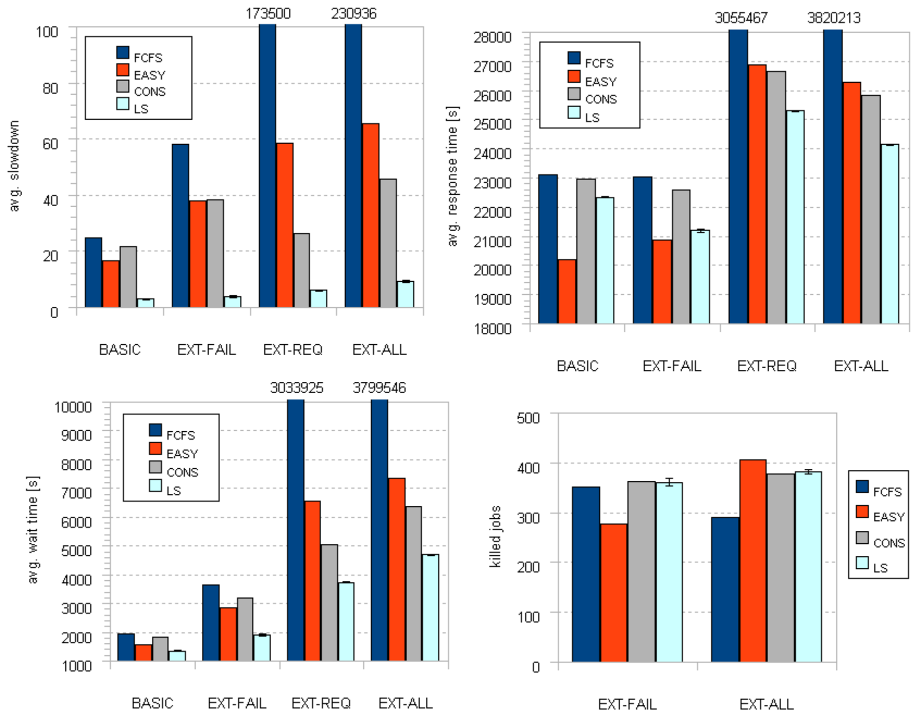

Fig. 2. Observed values of objective functions and the number of killed jobs for the MetaCentrum workload.

other options where to execute the job. On the other hand, if the specific job requirements are taken into account other possibilities may not exist, and jobs with specific requests have to wait until the suitable machines become available.

The comparison of EASY and CONS is interesting as well. Many previous studies have tried to analyze their behavior [7, 24, 29, 21]. A deep study of Feitelson [7] suggests that CONS is likely to produce better slowdown than EASY when precise runtime estimates are used and the system utilization is higher than approximately 50%. Similar behavior can be seen in the case of MetaCentrum, namely for EXT-REQ and EXT-ALL problems. Feitelson observed such large differences for workloads with at least 55-65% (Jann, Feitelson) or 85-95% (CTC) system utilization. For smaller system utilization, the performance of EASY and CONS was similar. However, the utilization of MetaCentrum is 43% on average. The reason for this behavior lies in the use of specific job requirements (EXT-REQ, EXT-ALL). As discussed, here jobs with specific requirements have to wait until the suitable machines become available. This in fact generates a higher system utilization on particular clusters, thus the benefits of CONS in this situation appear even for systems with a lower overall utilization.

If no specific job requirements are used (BASIC, EXT-FAIL), CONS produces worse or equal results than EASY in all cases.

Feitelson [7] suggests that EASY should generate better response time than CONS. This is clearly recognizable for BASIC and EXT-FAIL problems, while CONS is slightly better for EXT-REQ and EXT-ALL, but the difference is very small. Periodical application of LS optimization routine always improve the solution with respect to the CONS and — with an exception of the avg. response time in BASIC and EXT-FAIL — LS generates the best results among all algorithms. The standard deviation of different LS executions is very small. Concerning the total number of killed jobs, there is no clear pattern indicating the best algorithm. The total runtime of all scheduling algorithms grows with the problem complexity. Machine failures often collide with existing reservations which has to be recreated. When also specific job requirements are used, checks to identify suitable clusters must be performed for each job, increasing the total runtime of all scheduling algorithms. To sum up, the use of specific job requirements and machine failures significantly influence the quality of generated solution. In case of MetaCentrum, an experimental evaluation ignoring these features may be quite misleading. As was shown, the optimistic results for the BASIC problem are very far from those appearing when a more realistic EXT-ALL problem is considered.

6.2 Grid'5000 Workload

Figure 3 shows the results for the Grid'5000 workload. Similarly to the MetaCentrum workload, the highest differences appear between BASIC and EXT-REQ and EXT-ALL problems as can be seen in the case of the avg. slowdown and the avg. wait time. Again, due to the same reasons as before, the inclusion of specific job requirements (EXT-REQ) has a higher effect than the inclusion of machine failures (EXT-FAIL). The values of FCFS are often truncated for better visibility.

A closer attention is required when analyzing the average response time for EXT-FAIL and EXT-ALL problems in Grid'5000 shown in Figure 3 (top right). Initially, it is quite surprising that the average response time for EXT-FAIL and EXT-ALL is smaller than for the BASIC problem. The explanation is quite straightforward. In our simulations, whenever some machine fails, all jobs being executed on this machine are killed immediately. Moreover, such jobs are not resubmitted. As mentioned in Section 4.1, the failure rate in Grid'5000 is very high causing many premature job terminations (see Figure 3, bottom right). Therefore, long jobs are more likely to be killed than the short ones. Our experiments confirmed this expectations. The average length of a killed job in EXT-FAIL (FCFS) has been 60,690 seconds. However, the average length of all jobs in Grid'5000 is just 2,608 seconds. It means that especially long jobs are being killed in this case. Therefore, whenever machine failures have been used (EXT-FAIL, EXT-ALL), the average response time has been smaller than for the BASIC and the EXT-REQ problems, since many long jobs have been killed and

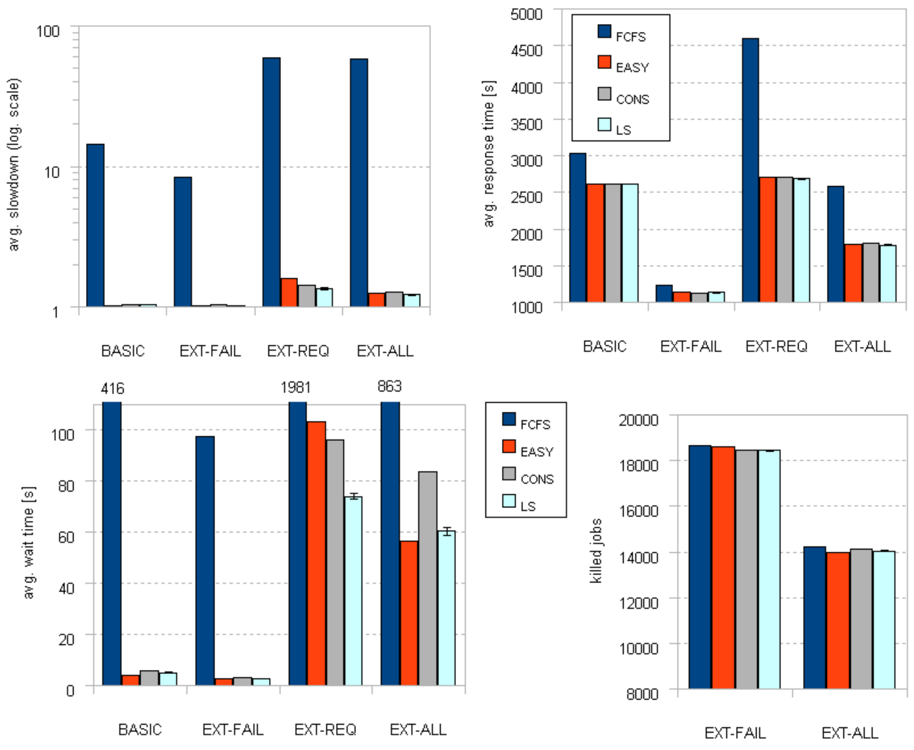

Fig. 3. Observed values of objective functions and the number of killed jobs for the Grid'5000 workload.

their long response times could not have been taken into account. The comparison of the avg. slowdown or the avg. wait time for BASIC and EXT-FAIL shows nearly no difference. Failures in Grid'5000 are usually very short as is shown in Figure 1 (bottom left). Therefore, they do not cause significant delays in job executions, although they appear very frequently. Moreover, the system utilization is very low (17%), so there is usually enough free CPUs to immediately start the execution of a newly incoming job.

As expected, FCFS did not perform well. In the Grid'5000 job trace, there are several jobs that request a large number of CPUs and only the largest clusters can be used. Since FCFS does not allow backfilling, smaller jobs in the queue have to wait until such large job has enough free CPUs to start its execution. It produces huge slowdowns for short jobs, although the overall utilization is only 15%. All remaining algorithms have been able to deal with such situations more efficiently, producing more or less similar solutions.

An interesting result is related to the Figure 3 (bottom right) showing the number of killed jobs. Here the total number of failed jobs for EXT-ALL is significantly lower than in the case of EXT-FAIL. This behavior is a combination of three factors and needs a closer explanation. First factor is related to the cluster selection process. If there are more suitable clusters to execute a given job then all scheduling algorithms applied in this paper select the fastest one. However, there are no information about clusters' speed in Grid'5000 and DAS-2, thus all clusters are considered to be equally fast. In such situation, all algorithms will choose the first suitable cluster (see Section 5), i.e., "the first fit" approach is applied. Since the Grid'5000 has a very small utilization (second factor) and clusters are always checked in a given order, most jobs are actually executed on the largest cluster. The third factor is the high failure rate in Grid'5000 (see Figure 1 top middle). The largest cluster exhibits 42% of all failures. When many jobs are executed on this single cluster, then the probability that machine failure will kill some job is rather high. For FCFS, there were 18668 killed jobs in the EXT-FAIL problem. 95.3% of them were killed at the largest cluster. Once the EXT-ALL problem is solved, specific job requirements cause that some jobs have to be executed on different clusters. As a side effect, the number of failed jobs decreases, as observed in Figure 3 (bottom right). To conclude, the combination of the "first fit" cluster selection policy together with high failure rate and low utilization may be very dangerous with respect to the number of killed jobs according to our observation. To prove this hypothesis we have developed a new version of the cluster selection policy, where — if multiple choices are available — a random cluster is selected using uniform distribution rather than the first one. We have used this policy in FCFS-LB (FCFS with Load Ballancing) scheduling algorithm and compared this solution with the original "first fit" FCFS. The results are shown in Figure 4. All experiments involving the FCFS-LB algorithm have been repeated 10 times using different seeds and their results have been averaged and the standard deviation computed. Concerning the number of killed jobs (right-

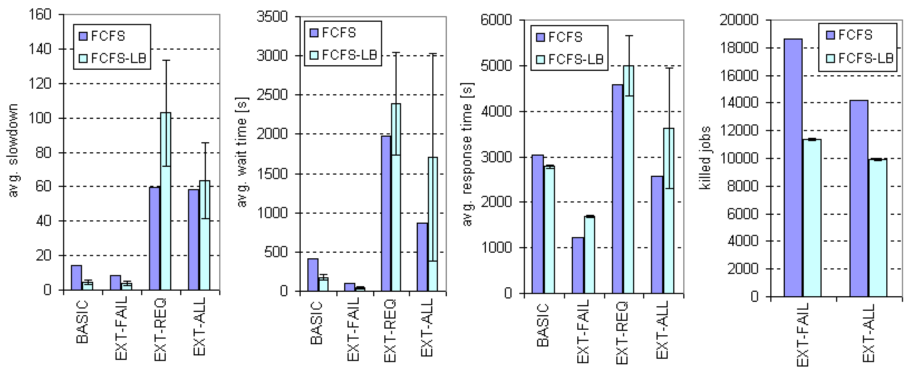

Fig. 4. Observed values of objective functions and the number of killed jobs for the FCFS and FCFS-LB in Grid'5000.

most graph) we can see that FCFS-LB works much better than FCFS since jobs are uniformly spread over available clusters. Also the avg. job slowdown and avg. job wait time is slightly better for BASIC and EXT-FAIL. On the other hand, as soon as specific job requirements are considered (EXT-FAIL, EXT-ALL) FCFS-LB produces worse results on average. Closer inspection shows, that the actual performance depends on the seed used in the random number generator, as can be seen from the large values of standard deviations. Clearly, simple solution such as FCFS-LB is not sufficient for more complex problems. We will try to fully understand this phenomena in the future since it is beyond the focus of this paper.

6.3 DAS-2 Workload

Figure 5 shows the results for the DAS-2 workload. As before, the highest differ-

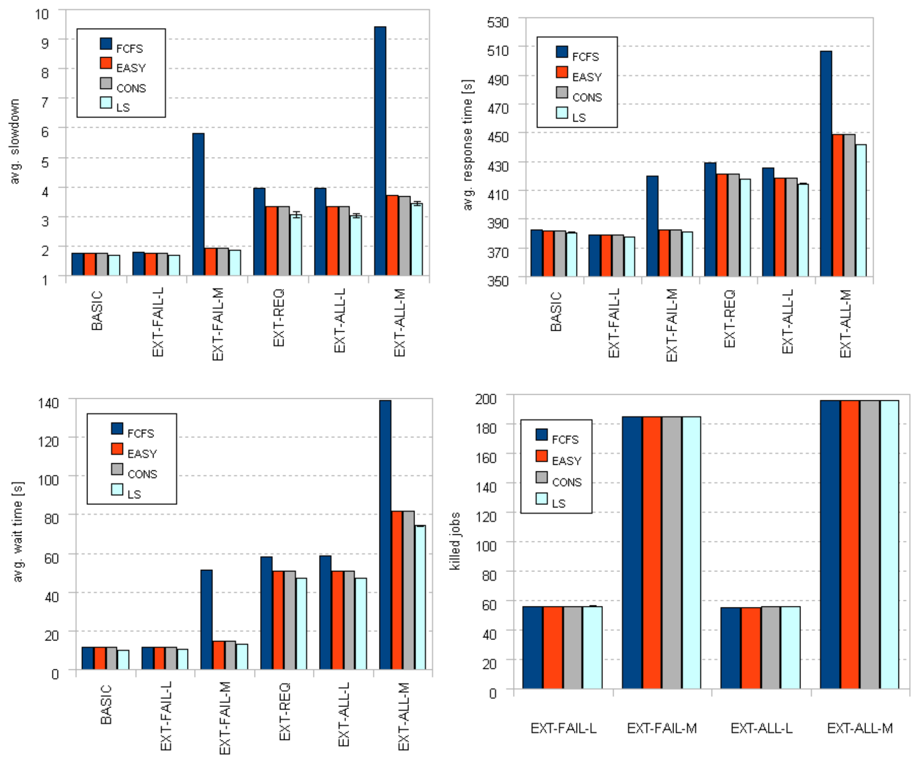

Fig. 5. Observed values of objective functions and the number of killed jobs for the DAS-2 workload.

ences in the values of objective functions appear between BASIC and EXT-ALL-L/EXT-ALL-M problems. For BASIC, no matter what scheduling algorithms is applied, the avg. slowdown, the avg. response time and the avg. wait time are always nearly the same. Again, the poorest results are related to the application of FCFS. The disproportion between EXT-FAIL-L and EXT-FAIL-M observed for FCFS can be still reduced by any other more complex algorithm. Similarly to the MetaCentrum workload, LS generates the best results in all cases. As soon as EXT-ALL-L or EXT-ALL-M is applied, the differences between scheduling algorithms become more visible. Especially for the EXT-ALL-M, the use of more complex scheduling algorithms start to make sense. Although the absolute differences of selected objective functions between BASIC and EXT-ALL-M or EXT-ALL-L are not very large, still the application of machine failures and specific job requirements results in different behavior even for so lowly utilized system (10%) as the DAS-2 is. When the higher failure rate of the MetaCentrum workload is used to generate the failures (EXT-ALL-M) the resulting values of the avg. wait time and the avg. response time are worse than the corresponding values for the workload with LANL-based failures (EXT-ALL-L). Also, the number of killed jobs is higher when the MetaCentrum-based failures are used (EXT-FAIL-M, EXT-ALL-M) which is expectable. Again, the use of specific job requirements (EXT-REQ) has usually higher effect than the use of machine failures only (EXT-FAIL-L, EXT-FAIL-M).

The total runtime of the scheduling algorithm follows the expectations — the more complex problem is considered and the more sophisticated algorithm is applied, the higher the algorithm runtime is.

6.4 Summary

In this section we have shown that the use of complete data sets has significant influence on the quality of generated solutions. Similar patterns in the behavior of scheduling algorithms have been observed using three different data sets. First of all, the differences between studied algorithms are usually small for the BASIC problems, but the situation changes for the EXTENDED problems. Here, an application of more intelligent scheduling techniques may lead to significant improvements in the quality of generated solutions, especially when the system utilization is not very low. In such case, the optimization provided by Local Search (LS) algorithm may outperform all remaining algorithms as observed for MetaCentrum data set. LS operates over the schedule of job reservations that is generated by CONS. When the system is lowly utilized, such schedule is often empty since jobs are executed immediately after their arrival. Then, LS has a little chance for optimization and its performance is very close to the original CONS algorithm as observed in Grid'5000 and DAS-2 cases. In addition, specific job requirements may have higher impact than machine failures. So far, the differences between the EASY and CONS solutions were likely to appear in systems with high utilization [7]. As observed in the MetaCentrum experiment, the application of specific job requirements can significantly decrease the threshold of the system utilization when the differences between algorithms are likely to appear. The inclusion of machine failures may have severe effect on the values of objective functions as observed mainly in Grid'5000 having a very high failure rate. In this case, another objectives such as the total number of killed jobs must be taken into account to explain otherwise "confusing" results. Moreover, low utilization, high failure rates combined with scheduler's selection policies could bring unexpected problems such as high numbers of killed jobs ("first-fit" job allocation). Trying to solve this problem using some form of load balancing may help, but other objectives can be easily degraded due to highly unstable performance as observed for FCFS-LB. Here, immediate solutions such as the simple load balancing do not work very well, since many other factors interact together. These observations support our idea that complete workload traces should be collected, published and used by the scientific community in the future. They will allow to test more realistic scenarios and they will help to understand the complicated behavior of real, complex systems.

7 Conclusion

In this paper, using the real-life based data from the Czech Grid MetaCentrum, we have demonstrated that machine failures or specific job requirements significantly affect the performance of various scheduling algorithms. Since the workloads in current archives miss to capture these features we have carefully extended selected existing workloads to show that they may exhibit similar behavior. Clearly, complete and "rich" data sets influence the algorithms' behavior and causes significant differences in the values of objective functions with respect to the simple versions of the problems. Especially, the effect of specific job requirements should not be underestimated. We suggest, that beside the common workloads from the GWA and the PWA, also the complete ones should be collected, published and applied to evaluate existing and newly proposed algorithms under harder conditions. As it was presented, existing base workloads may not clearly demonstrate the differences between trivial and advanced scheduling algorithms. When possible, detailed and standardized description of the original cluster and Grid environment should be provided as well, to assure that simulations will use correct setups. As a first step we provide the complete MetaCentrum data set for further open research.

Acknowledgments. We appreciate the gracious support of the Ministry of Education, Youth and Sports of the Czech Republic under the research intent No. 0021622419. We also appreciate the help of MetaCentrum, graciously supported by the research intent MSM6383917201. MetaCentrum team provided us the workload data and help us to interpret them correctly. The DAS-2 workload was kindly provided by the Advanced School for Computing and Imaging, the owner of the DAS-2 system, while the Grid'5000 workload and failure traces were kindly provided by the Grid'5000 team. We also appreciate the LANL failure traces graciously provided by the LANL team.

References

1. 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, *Job Scheduling Strategies for Parallel Processing*, volume 1659 of *LNCS*, pages 67–90. Springer, 1999.
2. Cluster Resources. *Moab workload manager administrator's guide, version 5.3*, January 2010. `http://www.clusterresources.com/products/mwm/docs/`.
3. The computer failure data repository (CFDR). `http://cfdr.usenix.org/`.
4. Dick Epema, Shanny Anoep, Catalin Dumitrescu, Alexandru Iosup, Mathieu Jan, Hui Li, and Lex Wolters. Grid workloads archive (GWA). `http://gwa.ewi.tudelft.nl/pmwiki/`.
5. Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. Benefits of global Grid computing for job scheduling. In *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 374–379. IEEE, 2004.
6. Dror G. Feitelson. Parallel workloads archive (PWA). `http://www.cs.huji.ac.il/labs/parallel/workload/`.
7. 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, 2005.
8. Dror G. Feitelson and Larry Rudolph. Metrics and benchmarking for parallel job scheduling. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1459 of *LNCS*, pages 1–24. Springer, 1998.
9. 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, *Job Scheduling Strategies for Parallel Processing*, volume 1291 of *LNCS*, pages 1–34. Springer Verlag, 1997.
10. Taliver Heath, Richard P. Martin, and Thu D. Nguyen. Improving cluster availability using workstation validation. *ACM SIGMETRICS Performance Evaluation Review*, 30(1):217–227, 2002.
11. Matthias Hovestadt, Odej Kao, Axel Keller, and Achim Streit. Scheduling in HPC resource management systems: Queueing vs. planning. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *LNCS*, pages 1–20. Springer Verlag, 2003.
12. Alexandru Iosup, Mathieu Jan, Ozan Sonmez, and Dick H. J. Epema. On the dynamic resource availability in grids. In *GRID '07: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pages 26–33. IEEE Computer Society, 2007.
13. Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, and Dick H. J. Epema. The grid workloads archive. *Future Generation Computer Systems*, 24(7):672–686, 2008.
14. Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 1. Wiley-Interscience, second edition, 1994.
15. James Patton Jones. *PBS Professional 7, administrator guide*. Altair, April 2005.
16. Dalibor Klusáček and Hana Rudová. Complex real-life data sets in Grid simulations (abstract). In *Cracow Grid Workshop 2009 Abstracts (CGW'09)*, 2009.

17. Dalibor Klusáček and Hana Rudová. Alea 2 – job scheduling simulator. In *Proceedings of the 3rd International Conference on Simulation Tools and Techniques (SIMUTools 2010)*, 2010.
18. Dalibor Klusáček and Hana Rudová. Efficient grid scheduling through the incremental schedule-based approach. *Computational Intelligence: An International Journal*, 2010. To appear.
19. Dalibor Klusáček, Hana Rudová, Ranieri Baraglia, Marco Pasquali, and Gabriele Capannini. Comparison of multi-criteria scheduling techniques. In *Grid Computing Achievements and Prospects*, pages 173–184. Springer, 2008.
20. Derrick Kondo, Bahman Javadi, Alexandru Iosup, and Dick Epema. The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems. Technical Report 00433523, INRIA, November 2009.
21. Jochen Krallmann, Uwe Schwiegelshohn, and Ramin Yahyapour. On the design and evaluation of job scheduling algorithms. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1659 of *LNCS*, pages 17–42. Springer Verlag, 1999.
22. Uri Lublin and Dror G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.
23. The MetaCentrum project. `http://meta.cesnet.cz/`.
24. Ahuva W. Mu'alem and Dror G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, 2001.
25. Repository of availability traces (RAT). `http://www.cs.illinois.edu/~pbg/availability/`.
26. Ramendra K. Sahoo, Anand Sivasubramaniam, Mark S. Squillante, and Yanyong Zhang. Failure data analysis of a large-scale heterogeneous server environment. In *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*, pages 772–784. IEEE Computer Society, 2004.
27. Bianca Schroeder and Garth A. Gibson. A large-scale study of failures in high-performance computing systems. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 249–258. IEEE Computer Society, 2006.
28. Joseph Skovira, Waiman Chan, Honbo Zhou, and David Lifka. The EASY - LoadLeveler API project. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1162 of *LNCS*, pages 41–47. Springer, 1996.
29. Srividya Srinivasan, Rajkumar Kettimuthu, Vijay Subramani, and P. Sadayappan. Selective reservation strategies for backfill job scheduling. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2537 of *LNCS*, pages 55–71. Springer Verlag, 2002.
30. Anthony Sulistio, Uros Cibej, Srikumar Venugopal, Borut Robic, and Rajkumar Buyya. A toolkit for modelling and simulating data Grids: an extension to GridSim. *Concurrency and Computation: Practice & Experience*, 20(13):1591–1609, 2008.
31. Dan Tsafrir, Yoav Etsion, and Dror G. Feitelson. Modeling user runtime estimates. In Dror G. Feitelson, Eitan Frachtenberg, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 3834 of *LNCS*, pages 1–35. Springer, 2005.
32. Fatos Xhafa and Ajith Abraham. Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, 26(4):608–621, 2010.

33. Ming Q. Xu. Effective metacomputing using LSF multicluster. In *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, pages 100–105. IEEE, 2001.
34. Yanyong Zhang, Mark S. Squillante, Anand Sivasubramaniam, and Ramendra K. Sahoo. Performance implications of failures in large-scale cluster scheduling. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 3277 of *LNCS*, pages 233–252. Springer Verlag, 2004.