On the Dynamic Resource Availability in Grids

Alexandru Iosup*, Mathieu Jan†, Ozan Sonmez*, Dick H.J. Epema*

*Faculty of Electrical Engineering, Mathematics, and Computer Science,
Delft University of Technology, The Netherlands
(A.Iosup, O.O.Sonmez, D.H.J.Epema)@tudelft.nl
†LRI/INRIA Futurs, University of Paris South, France
Mathieu.Jan@lri.fr

Abstract—Currently deployed grids gather together thousands of computational and storage resources for the benefit of a large community of scientists. However, the large scale, the wide geographical spread, and at times the decision of the rightful resource owners to commit the capacity elsewhere, raises serious resource availability issues. Littie is known about the characteristics of the grid resource availability, and of the impact of resource unavailability on the performance of grids. In this work, we make first steps in addressing this twofold lack of information. First, we analyze a long-term availability trace and assess the resource availability characteristics of Grid’5000, an experimental grid environment of over 2,500 processors. The average utilization for the studied trace is increased by almost 5%, when availability is considered. Based on the results of the analysis, we further propose a model for grid resource availability. Our analysis and modeling results show that grid computational resources become unavailable at a high rate, negatively affecting the ability of grids to execute long jobs. Second, through trace-based simulation, we show evidence that resource availability can have a severe impact on the performance of the grid systems. The results of this step show evidence that the performance of a grid system can rise when availability is taken into consideration, and that human administration of availability change information results in 10-15 times more job failures than for an automated monitoring solution, even for a lowly utilized system.

I. INTRODUCTION

Large-scale computing environments, such as the current grids CERN LCG [1], NorduGrid [2], TeraGrid [3] and Grid’5000 [4] gather (tens of) thousands of resources for the use of an ever-growing scientific community. At such scale, a significant part of the system resources may be at any time out of the users’ reach due to distributed resource ownership, scheduled maintenance, or unpredicted failures. Many of today’s grids comprise computing resources grouped in clusters, whose owners may share them only for limited periods of time. Often, many of a grid’s resources are removed by their owner from the system, either individually or as complete clusters, to serve other tasks and projects. Furthermore, grids encompass the problems of any large-scale computing environment, with the additional problem that their middleware is relatively immature, which increases further the resource unavailability rate. However, resource availability, and, most importantly, its impact on the performance of large-scale computing environments have yet to be analyzed. To address this gap, in this work we answer two questions.

The first question we address is: What are the characteristics of the resource (un)availability in large-scale environments? In Section II, we present detailed availability results at the resource, the cluster, and the system levels. Several other studies characterize or model the availability of environments such as super- and multi-computers [5], clusters of computers [6], [7], meta-computers (computers connected by a wide-area network, e.g., the Internet, also called desktop grids) [8], [9], and even peer-to-peer (file-sharing) systems [10], [11]. Our analysis is the first based on long-term availability traces from a multi-cluster grid environment. We further model four aspects of grid resource availability: the time when resource failures occur, the duration of a failure, the number of nodes affected by a failure, and the distribution of failures per cluster. The modeling results show that grid computational resources become unavailable at a high rate, and that nodes have rapidly increasing chances of failing with their uptime, negatively affecting the ability of grids to execute long jobs (even single-processor).

The second question we answer is: What is the performance impact of dynamic resource availability? We answer this question in Sections III and IV. First, we adapt traditional performance indicators, such as utilization for instance, to account for variable resource availability. Then, we show through trace-based simulation of a large-scale environment that the performance when taking into account availability is much better than when availability is not considered. In the first part of our simulations, we contrast the performance of a steady (available at all times) environment to that of a system in which resources fail. We continue our investigations by making various assumptions about the amount of availability information that is available to the resource manager, from perfect information to completely inaccurate. Our results also show that having more availability information is critical to achieve better system performance. During the experimental work for this part, we simulate Grid’5000 based on both workload and availability traces. To the best of our knowledge, ours is the first study to combine real information for both the jobs and the computational resources of a grid, for simulation purposes.

II. RESOURCE AVAILABILITY IN LARGE-SCALE COMPUTING ENVIRONMENTS

In this section, we present an analysis of resource availability in a large-scale multi-cluster grid: Grid’5000 [4]. Grid’5000 is an experimental grid platform consisting of 9 sites (grid VO) geographically distributed in France. Each site comprises one or several clusters, for a total number of 15 clusters and over 2,500 processors. Each cluster is made of a set of bi-processors nodes. Figure 1 shows the structure of Grid’5000. The number of processors per cluster is valid for 12 December, 2006.
A. Workload data analysis

We have analyzed availability traces recorded by all batch schedulers handling Grid’5000 clusters (OAR [12]), from mid-May 2005 to mid-November 2006. Altogether, this trace is made of more than half million of individual events that occurs on nodes. Each event in the trace represents a change in the status of nodes: either a node becomes available or unavailable. Note that most clusters of Grid’5000 were made available during the first half of 2005. However, availability information were only activated across the grid platform after mid-May 2005. In addition, note that we filter out the trace the impact of the reconfiguration system used in Grid’5000, which allows to reboot a set of nodes. In Table I, we summarize the content of the considered availability trace in this work, and the corresponding workload trace for this period. We refer the reader to the Grid Workloads Archive [13] for more details about the workload trace of Grid’5000.

In the remainder of this section, we first perform an analysis at grid and clusters level, that is by considering nodes from the whole platform and restricted to a specific cluster, respectively. Then, we perform an analysis at nodes level, that is considering all nodes across platform. The difference being that an node level analysis shows values for metrics of individual nodes, whereas a grid level analysis show values for the platform considered as a single entity.

Figure 2 shows the availability of resources in Grid’5000, at a grid level\(^1\). In average, resource availability in Grid’5000 at this level is 69\% \((\pm 11.4)^2\), with a maximum of 92\% and a minimum of 35\%. The mean time between failures (MTBF) of the environment is of 744\pm2631 seconds, that is around 12 minutes. Figure 3 shows the cumulative distribution function (CDF) of the different values of this MTBF for Grid’5000. At a cluster level, resource availability varies from 39\% up to 98\% across the 15 clusters. The average MTBF for all clusters is 18404\pm13848 seconds, so around 5 hours. As expected, this value is much higher than the MTBF at the grid level.

At a node level, our analysis shows that in average a node fails 228 times (for a trace that spans over 548 days). However, some nodes fail only once or even never according to our

\(^1\)May 2005 is not shown as availability information of clusters are starting to be recorded at different date during this month.

\(^2\)\(\pm\) stands for standard deviation.
in terms of number of events. Our results show a similar results as for the patterns of jobs: a daily peak during day hours (from 8am to 8pm) and a weekly pattern, less events occurring during weekends. In addition, we can clearly see the impact of the increasing size of Grid'5000 between 2005 and 2006 and, more importantly, the increase in its utilization.

Finally, we have also investigated the notion of groups of unavailabilities, which we called correlated failures. We define $T S ( \cdot )$ a function that returns the time stamp of an event. We therefore define correlated failures, with time parameter $\Delta$, as a set of failures (ordered according to increasing event time), in which for any two successive failures $E$ and $F$, $T S ( F ) \leq T S ( E ) + \Delta$. In short, we are interested to know how a single failure (either a node or a set of nodes) can affect other nodes? Note that we do not take into account the origin of the cluster for an individual failure to build a correlated failure. In our analysis, we vary $\Delta$ from 1 to 3600 seconds. However, we selected $\Delta = 60s$ as: 1) results for previous $\Delta$ (1, 10 and 30 seconds) show similar results and 2) this value is twice a commonly used value for timeout/delays in network operations (30 seconds). Figure 6 shows the CDF of the size of correlated failures for $\Delta = 60s$. Our analysis shows that in average the size of a correlated failure is 11.0±21.0, with a maximum of 339. This latter value is little less than the size of the largest cluster, which is made of 342 nodes. To confirm this value, we have analyzed the number of sites involved in a correlated failure. In average, this value is indeed of 1.06±0 with a maximum of 3 (for $\Delta = 60s$), that is to say that correlated failures generally do not expand beyond a site. To conclude about correlated failures, note that the number of correlated failures is 7473, for a total of around 85k failure events. Therefore, correlated failures represents less than 30% of the total number of failures events in the trace (around 300k).

B. Availability Model

In this section, we build a model for resource availability in multi-cluster grids. Our model considers four aspects: 1) the time when resource failures occur, 2) the duration of a failure, 3) the number of nodes affected by a failure and 4) the distribution of failures per cluster. Compared to traditional resource availability models [5], [15], [7], ours adds the necessary link between the failures and the clusters where they occur.

We begin by summarizing the basic statistical properties illustrated in the previous section. Table II shows the minimum, maximum, mean, and median values, and, for completion, the 1st and the 3rd quartiles of the availability Grid’5000 trace. The results for inter-arrival time and for size (rows A and C, respectively), show that the ratio between the mean and the median is relatively homogeneous across clusters. This indicates that a single distribution parameters (with the exception of scale) could be used across all clusters for a good fit with the inter-arrival time data and with the size data, respectively. Depending on the ratio between the mean and the median of the duration of failures, there are two main classes of clusters: class 1 with a ratio of about 1:1 (clusters c1 and c8), and class 2 with a ratio of about 1:6-9 (the remaining clusters). This may indicate the need for separate distributions for each of the classes, or for a distribution with more degrees of freedom, e.g., hyper-exponential or hyper-gamma. However, class 1 contains only clusters where only few jobs have been submitted over the duration considered for this study. Therefore, we choose to disregard this class, and use a single distribution to model the failure duration.

We first perform a graphical analysis of fitting the availability data. This allows us to eliminate from the modeling process the distributions which clearly do not fit to the data. Figure 7(a) shows the graphical analysis of the fit between the CDF of the logarithm of inter-arrival time of failures, for one cluster. Clearly, the exponential distribution is not a good fit. The other distributions yield reasonably close results, with the Weibull distribution looking especially promising. Figure 7(c) leads to similar conclusions. Figure 7(b) shows the graphical analysis of the fit between the CDF of the logarithm of failure duration, for one cluster. The Weibull, log-normal and even normal distributions look promising. We therefore select for detailed model fitting the normal, the log-normal, the Weibull, and the gamma distributions.

We now attempt to fit statistical distributions to our availability data. We use the following distributions: normal, log-normal, exponential, Weibull and gamma. For details regarding each of these distributions, e.g., their probability and their CDFs, we refer to [16]. We fit the above mentioned distributions using the Maximum Likelihood Estimation (MLE) method. Then, we perform goodness-of-fit tests to assess the quality of the fitting for each distribution, and to establish a best fit for each of the model parameters. For each distribution $d$, we formulate the hypothesis that the Grid’5000 availability data comes from the distribution $d$, whose parameters are found during the fitting process (the null-hypothesis of the goodness-of-fit test). We use the Kolmogorov-Smirnov test (KS-test [17]) for testing the null-hypothesis. The KS-test statistic, $D$, estimates the maximal distance between the CDF.
TABLE II
SUMMARY OF THE BASIC STATISTICAL PROPERTIES OF THE LOGARITHMIC GRID’5000 AVAILABILITY DATA (LOG(ITEM) FOR EVERY ITEM IN DATA).
VALUES HIGHER THAN 10000 HAVE BEEN REPORTED AS “>10k”. OMITTED MAX VALUE ROWS THAT CONTAIN ONLY “>10k” VALUES.

| | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A. Inter-arrival time between consecutive failures [s] | 25 | 21 | 18 | 16 | 14 | 12 | 10 | 8 | 6 | 4 | 2 | 0 | 2 | 4 |
| 1st Qu. | 3513 | 1165 | 4160 | 9648 | 3513 | 1500 | 1640 | 2000 | 1709 | 1005 | 1509 | 533.5 | 601 | 2356 | 901 |
| Median | 7258 | 3388 | 1540 | 1207 | 1800 | 1207 | 2400 | 1709 | 1005 | 1509 | 533.5 | 601 | 2356 | 901 |
| Mean | 6527 | 4608 | 2281 | 2187 | 4927 | 5399 | 5202 | 5734 | 5239 | 4640 | 4864 | 34923 | 3178 | 5369 | 3041 |
| 3rd Qu. | >10k | 8702 | 3475 | 3600 | >10k | 9824 | 9570 | 9783 | 9703 | 8156 | 8142 | 6432.8 | 4397 | 8582 | 3495 |
| B. Failure duration [s] | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Min. | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 1st Qu. | 791 | 122 | 195 | 151 | 97 | 159 | 126 | 235.3 | 247 | 116 | 152 | 86 | 125 | 106 | 175 |
| Median | 4475 | 263.5 | 125 | 303 | 100 | 163 | 157 | 1432 | 259 | 121 | 163 | 110 | 181 | 121 | 201 |
| Mean | 5022 | 1907.3 | 1047 | 2025 | 1655 | 553.7 | 772 | 1301.2 | 1103 | 418.3 | 560.6 | 1272 | 421 | 560.9 | 995 |
| 3rd Qu. | 9631 | 1832.8 | 1080 | 2445 | 679 | 169 | 479 | 1762.8 | 294 | 134 | 179 | 1157 | 339 | 162 | 1304 |
| C. Failure size [number of processors] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Min. | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 1st Qu. | 9.408 | 7.41 | 7.919 | 3.16 | 4.9 | 5.046 | 6.066 | 5.032 | 5.762 | 5.802 | 5.389 | 6.164 | 3.377 | 2.625 | 6.371 |
| Mean | 1.70 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Max. | 97 | 200 | 228 | 319 | 267 | 185 | 336 | 200 | 98 | 100 | 43 | 295 | 339 | 67 |

Fig. 7. Sample fit between the CDF of the logarithm of per cluster arrival data, and various statistical distributions: (a) fit of inter-arrival time of failures; (b) fit of failure duration; (c) fit of failure size.

Fig. 8. Generating resource failure data using the proposed availability model.

of the empirical distribution of the input data, and that of the theoretical distribution. The null-hypothesis is rejected if the D is greater than the critical value obtained from the KS-test table. The KS-test is robust in outcome (i.e., the value of the D statistic is not affected by scale changes, like using logarithmic values). The KS-test has the advantage over other traditional goodness-of-fit tests, like the t-test or the chi-square test, of making no assumption about the distribution of data\(^3\). The KS-test can disprove the null-hypothesis, but cannot prove it. However, a lower value of D indicates better similarity and a higher degree of similarity between the input data and data sampled from the theoretical distributions. We use this latter property to select the best fits. We find that the best fits for the inter-arrival time between failures, the duration of a failure, and the number of nodes affected by a failure, are the Weibull, Log-Normal, and Weibull, respectively. Table III shows the parameter values of the best fit of the best model for the Grid’5000 availability data per cluster and for the overall system, respectively. The results for inter-arrival time between consecutive failures are alarming: the shape parameter of the Weibull distribution is (high) above 1, which indicates an increasing hazard rate function (the frequency with which a system or component fails, provided that it has survived so far [14]). This indicates that the longer a computing node stays

\(^3\)Pearson’s chi-square test is applied to binned data (e.g., a data histogram). However, the value of the test depends on the how the data is binned.

\(^4\)However, note that these numbers depend on the date when clusters were made available to users.
In this section, we define and analyze the performance of a large-scale system when the dynamic availability of resources is or is not considered. Our results show that there is a big difference in the performance of the two cases, which prompts the investigation in Section IV.

A. Performance Metrics

The evaluation of grid performances depends on many factors, amongst which the system’s architecture, the workload, and also the system’s and the user’s objectives. For instance, resource providers may have as objective to maximize the number of jobs completed for a specific user. Another possibility is to maximize the utilization of the whole system. Similarly, users may have as objective completing as many jobs during a fixed time interval, or seeing the jobs being started with as little waiting time as possible. Several metrics have been traditionally used as a de-facto performance indicator of a grid, as they have often contrary impact on the performance of a system. However, in lack of availability-aware performance metrics, the performance results of systems with highly dynamic availability, e.g., grids, cannot be compared with those for other systems. This is especially true with the results of the cluster computing and of parallel production environments communities. We propose in the remainder of this section five availability-aware performance metrics, each adapted from a traditional metric.

First, we consider utilization, which is defined as the percentage of resources consumed by the system users, from the total resources present in the system, over a period of time. The ideal utilization value is 100%. However, due to resource fragmentation and other reasons, a utilization of 60-70% is considered high for systems that run parallel jobs [18]. For large-scale systems in which resources are not always available, computing utilization raises major practical difficulties, as the resource availability is usually not rigorously and accurately recorded.

We also consider the traditional metrics of wait time and response time. Note that in multi-cluster environments, jobs may spend time in several levels of queues, and computing the actual wait time of a job becomes a non-trivial task.

Finally, we consider the normalized throughput and the normalized goodput metrics. The throughput traditionally characterizes the number of jobs finished during a time interval, e.g., one day. Higher throughput values are considered better. The goodput characterizes the amount of resources consumed by all jobs towards their completion (this excludes the amount of time spent waiting in queues or for data to arrive). In both cases, in order to be able to compare grids of different sizes, we normalize these metrics, that is, we divide them by the number of processors in the system.
B. Models of Availability Information

The performance of a grid resource manager depends on the availability of the resources it manages. However, it is not the actual availability of computing nodes, but the information regarding it that the resource managers have to use. We therefore introduce below four models of grid availability information, from complete lack of to perfect:

1. Systems with Steady Availability (SA).
   This model assumes that all resources are online at all time. Many resource management results are readily available for these steady systems [19], [20].

2. Systems with Known Availability (KA).
   This model assumes a system with dynamic resource availability. However, the information regarding availability is perfect (complete and on-time). We are interested to understand what is the impact of perfect availability information on grid performance.

   This model assumes a system with dynamic resource availability. It also assumes that the most recent resource availability information is available from a monitoring system, which samples periodically the grid for individual computing nodes’ availability. If the monitoring period is high, the monitoring information can be stale; if it is low, the monitoring overhead is unbearable for the grid. We are interested to understand what is the impact of the information staleness.

   This model is similar to the AMA model, but assumes that the availability information is provided by the (human) system administrator at fixed, but relatively large intervals: 1 week or 1 month for instance. We are interested to understand what is the impact of human intervention.

IV. PERFORMANCE EVALUATION

In this section, we first present our experimental setup for our simulation. Then, we present our results for the previously introduced metrics (see section III-A) with our different models of availability information (see section III-B).

A. Experimental Setup

We have developed a custom trace-driven discrete event simulator which operates under the assumptions of identical processors for all grid nodes, and of FCFS policy for each cluster. We have simulated the Grid’5000 [4] platform, as shown by Figure 1, based on its availability trace (see section II-A) as well as the associated job trace during this period (June 2005 until October 2006). The workload trace is the GWA-7-2 from the Grid Workload Archive [13].

In our simulations, jobs may fail due to two reasons. First, a job fails when the scheduler has inaccurate information about the number of available (i.e., alive) processors in the system. Therefore, the scheduler wrongly considers that there are enough number of idle processors for the job. We call this situation a job submission failure. Second, a job fails when at least one processor used by this job crashes. We call this situation a job execution failure. We do not consider jobs that can cope with this situation. The time that the failed job has spent on used processors is taken into consideration for our performance analysis.

B. Results

Figures 9(a), 9(b), and 9(c) present the comparison of utilization, throughput and goodput-cpu time, respectively, in a system with steady availability (SA) and in a system with known availability (KA). As one may expect, the performance of all metrics, when taking into account availability information, is much better, compared to the case where it is not (see Table V for average values). The reason behind this is obviously that a more precise number of resources are taking into account by schedulers, leading to less job submission failures.

Further, we compare the performance results of two systems with automated monitoring of availability (AMA), with sampling rates of 60 seconds and 1 hour respectively. The first sampling rate reflects a real grid monitoring setting (e.g. Ganglia [21] for instance), whereas the second one represents a reasonably long sampling rate. Table V shows that different monitoring intervals do not lead to any relative performance degradation on the considered metrics. This can be interpreted as, in a system with considered resource availability characteristics and under low utilization, submission failures do not have a large impact of the performance of a grid. Moreover, we can also claim that resource failures do not cause that many job failures when the utilization of the system is low (see Table VI). As the comparison results are similar with KA (see Table V), we do not present the related graphs.

Figures 10(a), 10(b), and 10(c) present the comparison of utilization, throughput and goodput-cpu time in a system with human monitoring of availability (HMA). Intervals are set to 1-week and 1-month. Note, that we only plot the two months period where the differences are the more visible. Figures show that for the considered metrics, 1-week and 1-month intervals give similar results. However, note that considering fixed values for resource availability degrades the performance, compared to results obtained using other models.

Table VI presents the number of job completions and failures. The results imply that the HMA model leads to more job submission failures compared to the AMA model (see the job submission failure differences between these 2 models in Table VI). Of course, in a real system monitoring has the drawback of network overhead. However, our results also imply that with relatively long monitoring intervals, which would pose relatively low overhead on the network, same performance values can be attained. In addition, Table VI shows that the number of job submission failures is 10 to 15 times higher than the number of job execution failures. Thus, more work is required to overcome this limitation of current schedulers.

V. RELATED WORK

There exists a large number of studies that have considered the characteristics of system and workload (component) failures ([5], [10], [11], [7], [8], [9], [22]). From these, many refer to the systems of up to early 1990s ([15], [5]), are based on data sets spanning at most a few months ([8], [9]) or do not attempt to investigate the impact of these failures on the performance of their originating systems ([15], [10], [9], [22].
Similarly to this work, the studies in [23, 10], [11], [8], [9], [22] consider uncorrelated failures. Other studies have shown that for some systems there exist bursts of failures ([15], [5], [7]). Our work combines these approaches by analyzing errors at different levels of resource aggregation, e.g., from individual resource to complete grid. Only a few analyze systems of size ([22]) and purpose ([6], [7], [8], [22]) similar to the ones presented in this study.

The study most closely related to ours is [7], which analyze the node (un)availability through CPU failure, and its implication on the performance of large-scale clusters. Through simulation, and using a parallel production environment workload, they assess that the most important factor affecting performance is the failures arrival rate, which increases dramatically the job response time, and the work overhead.

Also closely related, [6] and [8] analyze the availability of desktop grids. They also give evidence that the performance of such a kind of system is around 70% of that of a cluster composed from equivalent resources, when the workload comprises parallel and sequential jobs, respectively.

VI. Conclusion

Currently deployed grids gather together thousands of computational and storage resources for the benefit of a large community of scientists. However, the large scale, the middleware immaturity, and at times the decision of the rightful resource owners to commit the capacity elsewhere, raise important resource availability issues. In this work, we have make first

Fig. 9. Performance of systems with SA and KA, over a sample period of 1 1/2 years: (a) Utilization; (b) Normalized throughput; (c) Goodput-cputime. Note: For readability, the vertical axis ranges are truncated.

**TABLE V**

<table>
<thead>
<tr>
<th>Availability model</th>
<th>Experiment</th>
<th>Avg. utilization (%)</th>
<th>Avg. normalized throughput</th>
<th>Avg. normalized goodput-cpu [s]</th>
<th>Avg. wait time [s]</th>
<th>Avg. response time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Steady Availability</td>
<td>12.1</td>
<td>0.48</td>
<td>10535</td>
<td>12913</td>
<td>15489</td>
</tr>
<tr>
<td>KA</td>
<td>Known Availability</td>
<td>16.6</td>
<td>0.86</td>
<td>14320</td>
<td>12494</td>
<td>14911</td>
</tr>
<tr>
<td>AMA</td>
<td>Monitoring '60s'</td>
<td>16.6</td>
<td>0.86</td>
<td>14320</td>
<td>12494</td>
<td>14911</td>
</tr>
<tr>
<td>AMA</td>
<td>Monitoring '1h'</td>
<td>16.6</td>
<td>0.86</td>
<td>13833</td>
<td>10214</td>
<td>12832</td>
</tr>
<tr>
<td>HMA</td>
<td>Intervention '1week'</td>
<td>15.9</td>
<td>0.81</td>
<td>13808</td>
<td>10173</td>
<td>12713</td>
</tr>
<tr>
<td>HMA</td>
<td>Fixed Availability</td>
<td>14.4</td>
<td>0.79</td>
<td>12491</td>
<td>7229</td>
<td>9793</td>
</tr>
</tbody>
</table>

**TABLE VI**

<table>
<thead>
<tr>
<th>Availability model</th>
<th>Experiment</th>
<th>Number of jobs submitted</th>
<th>Number of jobs completed</th>
<th>Number of job submission failures</th>
<th>Number of job execution failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Steady Availability</td>
<td>739164</td>
<td>739164</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>KA</td>
<td>Known Availability</td>
<td>739164</td>
<td>734588</td>
<td>0</td>
<td>4576</td>
</tr>
<tr>
<td>AMA</td>
<td>Monitoring '60s'</td>
<td>739164</td>
<td>734588</td>
<td>0</td>
<td>4576</td>
</tr>
<tr>
<td>AMA</td>
<td>Monitoring '1h'</td>
<td>739164</td>
<td>734588</td>
<td>0</td>
<td>4576</td>
</tr>
<tr>
<td>HMA</td>
<td>Intervention '1week'</td>
<td>739164</td>
<td>687956</td>
<td>46917</td>
<td>4291</td>
</tr>
<tr>
<td>HMA</td>
<td>Intervention '1month'</td>
<td>739164</td>
<td>683722</td>
<td>51208</td>
<td>4234</td>
</tr>
<tr>
<td>HMA</td>
<td>Fixed Availability</td>
<td>739164</td>
<td>671925</td>
<td>63180</td>
<td>4059</td>
</tr>
</tbody>
</table>

Fig. 10. Performance of the system with HMA, over a sample period of 2 months: (a) Utilization; (b) Normalized throughput; (c) Goodput-cputime. Note: For readability, the vertical axis ranges are truncated.
steps in analysis the scale and the characteristics of resource availability in grids.
First, we have analyzed a long-term resource availability trace from a multi-cluster grid, Grid’5000. Our analysis shows that the resource availability in grids varies greatly. We find that the MTBF is high: around 12 minutes at grid level, 5 hours at cluster level, and around 2 days per computing node. The duration of the computing nodes failures is 14 hours. We further find that when a failure occurs, it affects on average 10 or more computing nodes.

Second, we have created a grid resource availability model, which considers the time when resource failures occur, the duration of a failure, the number of nodes affected by a failure, and the distribution of failures per grid cluster. The results for the inter-arrival time between failures are alarming: the shape parameter of the Weibull distribution, our best fit, indicates an increasing hazard rate with strong effects on the ability of grids to execute long jobs (even single-processor).

Third, we have analyzed the performance impact of dynamic resource availability in grids. We have considered four resource managers with different levels of resource availability information, and we have simulated their use in Grid’5000, based on real traces for both the resource availability and the workload. Our simulations show that: considering resource availability is important when assessing the performance of a grid, and that human monitoring and intervention of the system leads to 10 times more job failures (both submission and execution) than that of an automated alternative.

As future work, we would like first to validate our resource unavailability model using other traces, especially as the Grid’5000 platform targets special use cases\(^3\). We also plan to investigate the effect of varying resource availability characteristics in the model, e.g., the interarrival time between consecutive failures for instance, on the system performance. Finally, to extend our contribution, we plan to study how our results can be applied to other large-scale computing environments, and in particular for parallel production environments.

ACKNOWLEDGMENTS
This work was carried out in the context of the Virtual Laboratory for e-Science project (www.v1-e.nl), which is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W), and which is part of the ICT innovation program of the Dutch Ministry of Economic Affairs (EZ). We gratefully acknowledge and thank Dr. Franck Cappello and the Grid’5000 team, for providing us with the grid traces used in this work. We also thank Shanny Anoep for his help with the experiments.

DATA AVAILABILITY
The data used in this article is available online as part of the Grid Workloads Archive:

http://gwa.ewi.tudelft.nl

\(^3\)Obtaining availability traces is difficult: resource owners prefer to show that their system works (workload traces), than that it does not (availability traces). We urge potential contributors to consider also the benefits that they will get from resource managers that react properly to resource unavailability.

REFERENCES