Performance Degradation Analysis of a Supercomputer

Domenico Cotroneo, Flavio Frattini, Roberto Natella, Roberto Pietrantuono
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Università degli Studi di Napoli Federico II
Via Claudio 21, 80125, Naples, Italy
{*cotroneo, flavio.frattini, roberto.natella, roberto.pietrantuono*}*@unina.it*

***Abstract*—We analyze performance degradation phenomena due to software aging on a real supercomputer deployed at the Federico II University of Naples, by considering a dataset of ten months of operational usage. We adopted a statistical approach for identifying when and where the supercomputer experienced a performance degradation trend. The analysis pinpointed performance degradation trends that were actually caused by the gradual error accumulation within basic software of the supercomputer.**

***Keywords*-Software Aging; Software Rejuvenation; High Performance Computing; Trend Analysis; Granger Causality**

I. Introduction

Characterizing and understanding performance issues in real-world computer systems, and in particular software aging phenomena, is of great importance to devise effective strategies against performance degradations and failures, such as software rejuvenation [1]–[3]. This is especially true for supercomputers, which are exposed to potential performance issues due to the huge volume and long-running nature of their workload, the large number of computational nodes involved, and their inherent complexity.

In this paper we describe our experience with the SCoPE supercomputer deployed at the Federico II University of Naples (http://www.scope.unina.it). The aim of our study, even if preliminary, is to provide a better understanding of performance degradation issues in supercomputers due to software aging. This both compensates for the lack of detailed studies on software aging in supercomputers and supports the definition of suitable software rejuvenation strategies for this kind of system.

The analysis faces a set of issues that arise when seeking for degradation phenomena within multiple sources of information and in the presence of several users and resources. A primary difficulty is related to the workload: large computer systems are typically used by many and diverse categories of users that submit a wide variety of workload patterns. Performance is notoriously related to the amount and type of work submitted in a given time period, and many studies showed that the workload indeed influences software aging phenomena [4]. Most of the experiments on software aging in real systems, with few exceptions, have been conducted in a controlled environment, and thus characterize the relation between a stable and well-defined workload with some aging indicators over a time period of hours [3], [5]–[7]. In a real setting, the analysis is much more complex: the workload is highly variable, and what administrators observe is not easily interpretable as a performance increase or decrease caused by aging issues. Hence, when considering a real scenario, the analyst has to figure out which degradation behaviors are simply due to a variation of the workload, and which ones are instead due to aging. Besides the impact of the workload, the analysis of supercomputers is made complex by the existence of several dynamics that can occur at the same time on the different nodes of the system. Different software aging phenomena can overlap to (and/or mask) each other, greatly hampering the detection of performance degradation trends. What the administrator sees at high level may not suffice to identify aging trends for the overall system. S/he may misjudge some behaviors as a degradation problem and/or overlook real software aging trends at a finer grain.

Our analysis is based on a statistical approach for localizing performance degradation trends at a fine grain, i.e., identify when and where the system experienced a performance degradation trend. The approach is aimed at guiding the analyst in the investigation of huge amounts of performance data, and to enable to focus problem diagnosis and failure prevention strategies on specific parts of the system that are affected by performance degradation. The empirical analysis of SCoPE data shows that supercomputers can actually be affected by software aging, and that aging phenomena can be local to specific parts of the system and occur only in specific periods of usage. We pinpointed some performance degradation trends that were actually caused by the gradual error accumulation within basic software of the supercomputer. These results encourage further research on software rejuvenation strategies for supercomputers.

The paper is organized as follows. After an overview of related work in Section II, we describe the approach in Section III. Section IV presents the results of the empirical analysis that are discussed in Section V.

II. Related work

The analysis of performance degradation aiming at discovering software aging phenomena is discussed in several papers, although most software aging studies are on memory consumption [8]. On one hand, studies analyze software aging effects arising in a controlled environment. In [6], [7], we conducted a workload-dependent analysis on Linux and on the JVM respectively, that was generalized later in [4] in a workload-based stress-testing method for software aging analysis. In many studies of this kind, web applications and web servers are considered as case study: examples are in [3], where authors analyze the performance degradation of the Apache web server, and in [5], where authors analyze dependability of Service-Oriented Architectures, pointing out software aging issues in controlled stress-test experiments.

On the other hand, there are studies that analyzed software aging in production environments. One of the first work in this direction is in [1], later extended in [2], where authors present an analysis that takes into account some system-level workload parameters, such as the number of CPU context switches and page faults. Their results confirm that software aging trends are related to the workload states. Nevertheless, these papers are focused on a single machine, while the analysis of software aging in supercomputers is pretty unexplored. Examples of work a bit closer to ours are the studies analyzing clustered systems, such as [9]–[11] in which, however, the focus was on exploring optimal rejuvenation policies (e.g., cluster failover strategies), typically by model-based analyses. In such studies, software aging issues in supercomputers are only hypothesized, but they do not provide evidence about software aging in this kind of system. Instead, previous studies on supercomputers aimed at characterizing their performance [12], [13] and failure distribution [14], [15], without focusing on their performance and reliability in the long-term and on software aging issues. The lack of a characterization of software aging phenomena, in turn, makes difficult to devise and to evaluate software rejuvenation strategies for supercomputers.

III. Data Analysis Approach

The goal of the approach is to point out performance degradation trends in supercomputer performance data. We seek for performance decreases since they often represent the onset of software aging phenomena, and can suggest where and when to perform more in-detail analyses to find and fix aging problems. The method is based on a combination of statistical hypothesis testing techniques.

In our analysis, we face an important issue: the performance data are not obtained from a controlled experiment, but they are collected during the actual usage of the supercomputer "in the wild", i.e., in production. This implies that variations in performance data can be due to many factors including, but not limited to, software aging problems. Another factor that impacts on performance is represented by the *workload* of the system: for instance, when the system is overloaded, its performance can be severely reduced. Also, given that data are collected over a large period of time, the supercomputer may experience events during this period that could impact on its performance, such as scheduled and unscheduled maintenance activities, and periods of inactivity occurring between periods of intense activity (e.g., inactivity between two different projects).

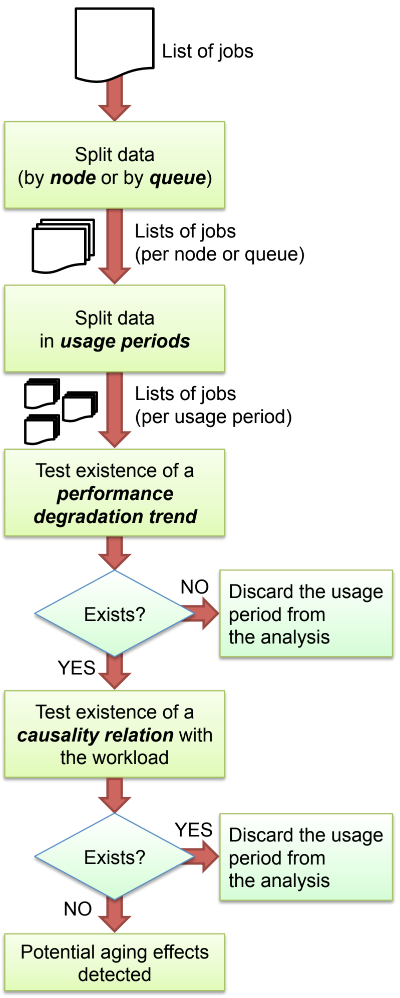

List of jobs
Split data (by **node** or by **queue**)
Lists of jobs (per node or queue)
Split data in **usage periods**
Lists of jobs (per usage period)
Test existence of a **performance degradation trend**
Exists?
NO
Discard the usage period from the analysis
YES
Test existence of a **causality relation** with the workload
Exists?
YES
Discard the usage period from the analysis
NO
Potential aging effects detected

Figure 1. Overview of the analysis.

In our approach, we adopt a sequence of data processing steps as depicted in Figure 1 and described in the following.

Step 1. We first divide the dataset into several subsets. Each subset includes performance data from a specific *node* or *scheduling queue* of the system (the *space* dimension). We do so since software aging phenomena may not be observable if considering the whole supercomputer; in fact, software aging can affect only a specific part of the system (e.g., a set of nodes) or a specific type of jobs (e.g., jobs with special features that are assigned to a specific scheduling queue). We thus obtain a distinct dataset for each distinct node or distinct queue in the initial dataset.

Step 2. We further split the subsets, by considering the *time* dimension. In fact, nodes and queues tend to exhibit periods of intense usage, which are separated by periods of inactivity. For instance, a node becomes inactive when it undergoes a maintenance intervention, such as a reboot or a hardware upgrade. Another situation is represented by two different projects or teams that alternate in the usage of a specific resource. We denote with the term "*usage period*" a time period during which a node or queue is continuously used, and which is preceded and followed by periods of inactivity. Therefore, different usage periods are not necessarily related to each other, and different workloads and performance trends can take place during distinct usage periods. Therefore, we *separately analyze each usage period*, by splitting data in distinct subsets, each containing performance data for a specific usage period. In this way, we avoid to mix usage periods that are interleaved with a maintenance intervention or by an abrupt change of usage pattern. To split different usage periods, we look for periods with no workload: if an usage period is followed by a period of no workload (i.e., no job is being processed), then we assume that the usage period has ended.

Step 3. For each dataset obtained from the previous steps, we perform a *statistical hypothesis test* to evaluate whether there exists a *statistically significant trend* in performance data, that is, a gradual and enduring decrease of performance that is not due to random data variations, with a quantitative degree of confidence. To detect performance degradation trends, we adopt the *Mann-Kendall* test, which is a statistical procedure for detecting linear trends that has been adopted in many domains, including several studies on software aging and rejuvenation [1], [3]. The test evaluates the likelihood of the observed data under the assumption (*null hypothesis*) that there is no trend in the data; if this likelihood is very low (e.g., lower than 10%), then the null hypothesis is *rejected* and a trend is detected. When a trend is detected, we then estimate its *intensity* using the Sen's procedure [1], [3], which is robust (i.e., it does not assume normally distributed measurement errors) and insensitive to outliers. This procedure provides an estimation of the slope, along with a confidence interval in which the value of the slope lies with a given probability (e.g., the interval containing the actual value of the slope with 95% probability). Given that several hypothesis tests are being performed, and given that some of the tests can erroneously detect a trend, we need to assure that the false rejection rate of the overall set of the tests is below a reasonable limit. We therefore adopt the Benjamini-Hochberg procedure [16], which controls the false rejection probability of a set of tests (we set the maximum false rejection rate to 10% in our analysis) by tuning the thresholds adopted for rejecting the null hypothesis at each Mann-Kendall test.

Step 4. Finally, after that a performance degradation trend has been detected and estimated, we perform a joint analysis between performance data and workload data, in order to determine whether the performance degradation trend was caused by a variation of the workload or not. In fact, if a performance degradation occurs at the same time of a variation of the workload, it is doubtful whether the performance degradation has been caused by software aging or by the variation of the workload. To be on the conservative side, we opted to discard performance degradation trends that are likely to depend on workload variations. In such a way we avoid performance degradation trends that are not actually due to software aging: this is especially important when considering large systems with huge amounts of performance data, as in the case of supercomputers. To rigorously perform this filtering, we adopt the Granger causality test [17]. Given two time series X and Y, the test evaluates whether a variation of X is useful to forecast a later variation of Y (X *Granger-causes* Y). The test performs a regression analysis between the time series Y (in our case, a performance indicator), and lagged versions of both X and Y, and it detects a Granger causality relationship when X (in our case, a workload indicator) contributes to the regression model in a statistically significant way. In our analysis, we set the lag between the series being between 1 and the maximum job duration D (up to a few days, depending on the node or queue), since a performance variation that occurs later than $t + D$ is unlikely caused by a workload variation at t. In the case that there is no statistically significant relation of causality between workload and performance data (we adopt a significance level of 10%), we conclude that a performance degradation trend can potentially represent a software aging phenomenon that is worth to inspect more closely.

IV. Results

The approach described in Section III is applied to performance and workload data collected at the SCoPE supercomputer over a period of 10 months. SCoPE is a batch system developed by the Italian Institute of Nuclear Physics (INFN) and the Federico II University of Naples for research activities and also as a Tier-2 resource of the Worldwide LHC Computing Grid (WLCG). It is made up of 512 servers, each equipped with 2 quad core CPUs and 32GB of memory, and Scientific Linux as operating system. For jobs queueing, job scheduling and resource management, SCoPE uses Maui/TORQUE. Our dataset is the list of jobs submitted to and executed by the system over a certain time period. For each job, we have the following information:

- **Job ID**: a unique identifier of the job in the system;
- **Queue**: the scheduling queue to which the job was submitted; jobs are allocated to different queues depending on their duration (e.g., *long* or *short* jobs), the specific organizations that use the system, and on the specific project they belong to;
- **Submission Time**: the time (in Unix timestamp) when the job was submitted to the system;
- **Start Time**: the time (in Unix timestamp) when the execution of the job started;

- **Completion Time**: the time (in Unix timestamp) when the execution of the job finished;
- **Node**: the node of the system in which the job ran;
- **CPU utilization**: average usage of CPU by the job.

To quantify the workload of the system by considering the *number of jobs submitted to the system at each day*, on the basis of their *Start Time*. To quantify performance, we consider the *average duration of jobs at each day*, that is, by computing the mean value of the duration of the jobs completed at each day, on the basis of their *Completion Time* and *Submission Time*. Even if these metrics only provide a partial indication of the workload and of performance, these metrics can be easily computed from the dataset that was available to us, and enable a preliminary analysis of performance degradations; we aim to extend the analysis in the future with more metrics, based on an extended dataset.

Table I provides some basic information about the dataset used for the analysis. During the analyzed period, several hundreds of thousands of jobs were submitted to the supercomputer from many different users; jobs were scheduled through 20 queues and evenly distributed over 292 nodes.

Table I
BASIC INFORMATION ABOUT THE DATASET USED FOR OUR ANALYSIS.

Number of jobs	277,249
Number of users	475
Number of nodes	292
Number of queues	20
Observation period	Oct. 1st, 2010 - July 31st, 2011
Number of queues with performance degradation trends	4
Number of nodes with performance degradation trends	8

The analysis of performance data of the system as a whole (i.e., without splitting data by node/queue and by usage period) did not point out any decreasing trends of the performance. Instead, we found some performance degradation trends on a small subset of nodes and queues in the supercomputer. This suggests that software aging phenomena are localized in a specific part of the system or for specific types of workload. We hypothesize that this can be due to *(i)* software aging issues (e.g., resource leaks) that are only present in the software of the affected nodes or queues, and/or *(ii)* specific types of jobs that trigger such issues. We focus the rest of the analysis on the nodes that exhibited a performance degradation trend (i.e., a significative increase in the average duration of the jobs).

Table II reports the estimated intensity of the detected trends and the usage periods in which these trends were detected. Figure 2 shows workload (plot at the top of each subfigure) and performance data (plot at the bottom of each subfigure) from some of these nodes.

Table II
PERFORMANCE DEGRADATION TRENDS DETECTED BY THE APPROACH.

Node	Estimated slope [hours/day]	Confidence Interval [hours/day]	Period [days]
wn157	0.4070	[0.1128, 0.6060]	[149, 170]
wn167	0.2065	[0.0237, 0.4002]	[148, 171]
wn168	0.2347	[0.0240, 0.4008]	[148, 171]
wn182	0.5746	[0.0000, 1.0341]	[156, 180]
wn197	0.3755	[0.0000, 0.9070]	[156, 187]
wn211	0.6977	[0.2374, 1.2665]	[156, 181]
wn241	0.1043	[0.0103, 0.2044]	[118, 195]
wn260	0.0860	[0.0036, 0.1719]	[117, 195]

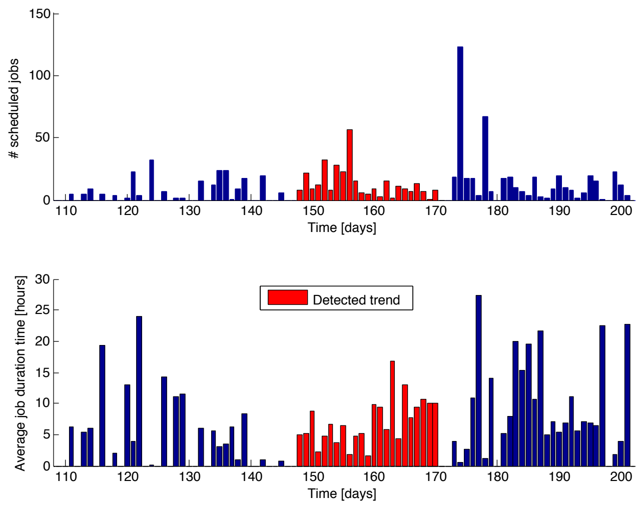
The average slope of the trends is 0.3358 hours/day, that is, the average job duration increased by a fraction of hour at each day, over a period of time that ranges between 21 and 78 days. While such trends are clearly visible for some nodes, in other cases only a rigorous statistical approach is able to detect the increasing trend of the average daily job duration. An interesting finding is that the detected trends at different nodes occurred at overlapped periods of time: all of them include the time period between day 149 and day 170. Moreover, the performance data seem to have followed similar patterns, such as those showed in Figure 2. This observation suggested us that a common root cause was behind these performance degradation trends in the SCoPE supercomputer, either in user software or in the basic software infrastructure of the supercomputer.

We also performed an analysis on the types of jobs, by dividing the entire job dataset between CPU-bound and I/O-bound jobs, on the basis of CPU usage. This analysis, which is not shown for the sake of brevity, did not find a relation between the type of job and the presence of a trend, since performance degradation trends manifest both in CPU-bound and I/O-bound jobs. Therefore, the basic software infrastructure, rather than user software, seem to be the likely root cause of the performance degradation.

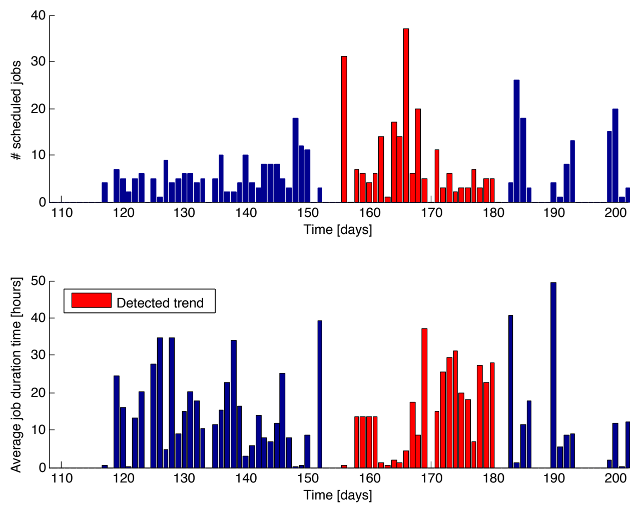
The Granger causality test pointed out that these performance degradation trends seem not to be related to variations of the workload. This result provides some confidence that the trends are not related to random variations of the workload, and that it is worth to inspect these trends more closely. Instead, Figure 3 shows an example of a performance degradation trend that seems to be related to workload variations and that was thus discarded from the analysis. Even if the average job completion time seems to be increasing, the workload exhibited similar variations, therefore casting doubt on the existence of a software aging phenomenon behind that performance trend.

V. DISCUSSION

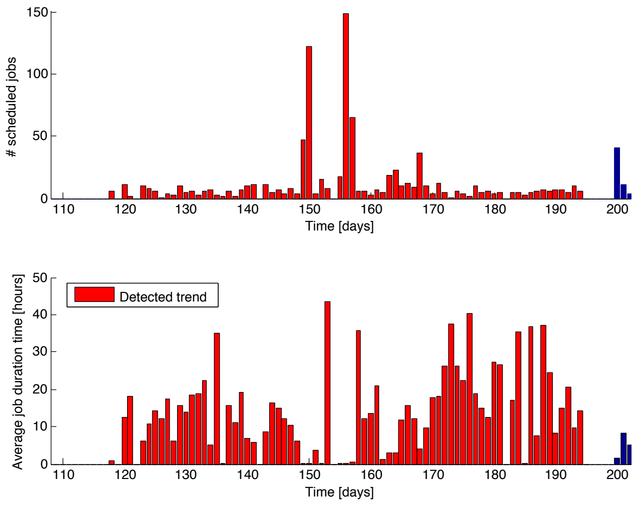
The results presented in Section IV show that supercomputer performance exhibit suspicious performance degrada-



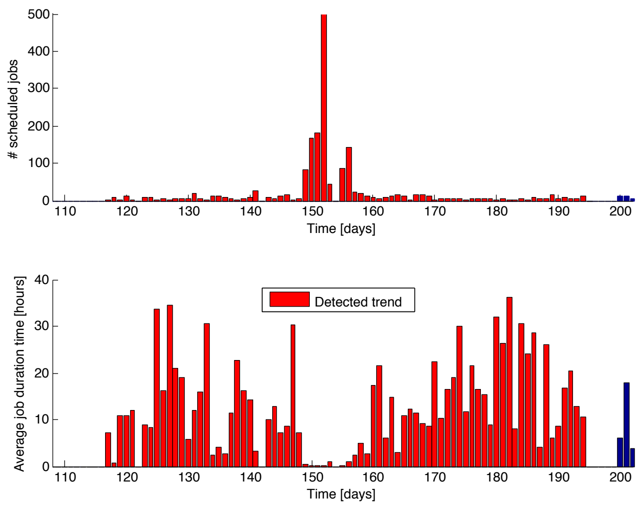
(a) Node wn168.



(b) Node wn211.



(c) Node wn241.



(d) Node wn260.

Figure 2. Performance degradation trends detected at a subset of nodes.

tion phenomena that can be classified as effects of software aging. Performance degradation was present in the system, even if it was not apparent when considering the system in its entirety. The analysis of the performance trend with node or queue granularity, instead, allowed us shining light on such issues. Moreover, by considering the absence of causality with the workload trend, we reduce the chance of false positives and support the hypothesis that the performance degradation is a symptom of software aging.

In our specific case, the similar trends found on several nodes are a signal that the software at that nodes were suffering from a common cause of performance degradation. Moreover, the lack of relationship between the type of jobs and performance degradation trends indicates that the underlying, shared software layer may be the root cause of performance issues. A more detailed analysis of job life-cycle, and a discussion with the SCoPE technicians, showed that the increase of job duration was due to the accumulation of errors in a distributed filesystem. In fact, read/write errors, due to filesystem corruption, caused the stop and the restart of several jobs, thus delaying their completion. The accumulation of stopped and restarted jobs, also caused the shortage of computing resources, such as memory, which delayed the execution of other jobs [18], [19]. These errors were more and more frequent with time, thus causing a gradual performance degradation of the nodes accessing that filesystem and hampering the proper execution of jobs. The problem lasted for about one month, until a maintenance intervention interrupted the performance degradation trend.

Another important finding of the analysis is that performance degradation trends occurred during a specific period, instead of being distributedd across the whole observation

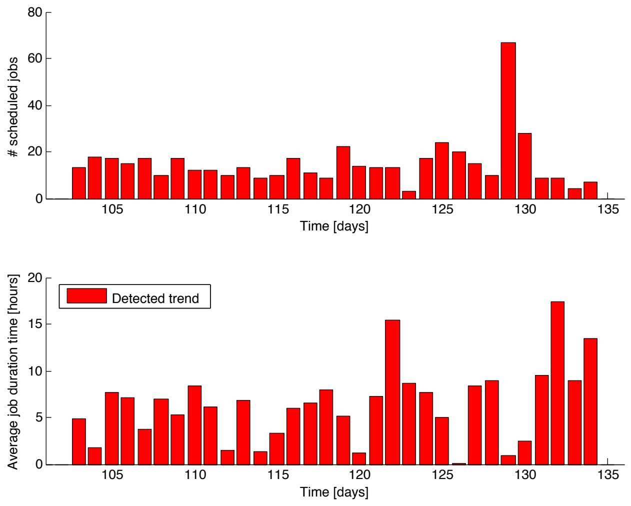


Figure 3. Example of performance degradation trend with a causal relationship with workload variations.

period. This suggests that highly variable environments, such as supercomputers, require measurement-based rejuvenation (i.e., triggering rejuvenation only when measurements exhibit a software aging trend) rather than pure time-based rejuvenation (i.e., rejuvenation is periodically triggered) to achieve better performance and reliability [8]: in such systems, the workload and maintenance activities may cause the abrupt occurrence or demise of performance degradation.

It should be noted that, in this early stage of our study, only the number of jobs has been considered as workload metric. More detailed information about jobs would enable the analysis of other influencing factors, such as the type of computations made by the jobs and their memory and I/O usage, and provide additional insights about the causes of performance degradations. Also, crossing performance data with failure reports can help at identifying the root cause of performance degradations, thus improving both the understanding of the issue and its prevention as well.

Acknowledgments

This work was supported by the projects *TENACE* (PRIN n.20103P34XC) funded by the MIUR, and *ICEBERG* (IAPP n.324356) funded by the European Commission.

References

[1] S. Garg, A. van Moorsel, K. Vaidyanathan, and K. Trivedi, "A Methodology for Detection and Estimation of Software Aging," in *Proc. Int'l Symp. on Soft. Rel. Eng.*, 1998.

[2] K. Vaidyanathan and K. S. Trivedi, "A measurement-based model for estimation of resource exhaustion in operational software systems," in *Proc. Int'l Symp. on Soft. Rel. Eng.*, 1999.

[3] M. Grottke, L. Li, K. Vaidyanathan, and K. Trivedi, "Analysis of software aging in a web server," *IEEE Trans. on Reliability*, vol. 55, no. 3, 2006.

[4] A. Bovenzi, D. Cotroneo, R. Pietrantuono, and S. Russo, "Workload characterization for software aging analysis," in *Proc. Int'l Symp. on Soft. Rel. Eng.*, 2011.

[5] L. Silva, H. Madeira, and J. Silva, "Software aging and rejuvenation in a soap-based server," in *Proc. Int'l Symp. on Network Computing and Applications*, 2006.

[6] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "Software aging analysis of the linux operating system," in *Proc. Int'l Symp. on Soft. Rel. Eng.*, 2010.

[7] D. Cotroneo, S. Orlando, R. Pietrantuono, and S. Russo, "A measurement-based ageing analysis of the JVM," *Software Testing Verification and Reliability*, 2011.

[8] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "A Survey of Software Aging and Rejuvenation Studies," *ACM Journal on Emerging Technologies in Computing Systems*, 2013, to appear.

[9] L. Silva, J. Alonso, and J. Torres, "Using virtualization to improve software rejuvenation," *IEEE Trans. on Computers*, vol. 58, no. 11, 2009.

[10] A. Avritzer, A. Bondi, and E. Weyuker, "Ensuring system performance for cluster and single server systems," *Journal of Systems and Software*, vol. 80, no. 4, 2007.

[11] D. Wang, W. Xie, and K. Trivedi, "Performability analysis of clustered systems with rejuvenation under varying workload," *Performance Evaluation*, vol. 64, no. 3, 2007.

[12] U. Krishnaswamy and I. Scherson, "A framework for computer performance evaluation using benchmark sets," *IEEE Trans. on Computers*, vol. 49, no. 12, 2000.

[13] D. Feitelson, "Workload modeling for performance evaluation," in *Performance Evaluation of Complex Systems: Techniques and Tools*, 2002.

[14] N. Gottumukkala, R. Nassar, M. Paun, C. Leangsuksun, and S. Scott, "Reliability of a system of k nodes for high performance computing applications," *IEEE Trans. on Reliability*, vol. 59, no. 1, 2010.

[15] E. Heien, D. Kondo, A. Gainaru, D. LaPine, B. Kramer, and F. Cappello, "Modeling and tolerating heterogeneous failures in large parallel systems," in *Proc. Int'l Conf. for High Perf. Computing, Networking, Storage and Analysis*, 2011.

[16] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *J. Roy. Stat. Soc., Ser. B*, 1995.

[17] C. W. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica: Journal of the Econometric Society*, 1969.

[18] Rice University - Division of Information Technology, "Why Are My Jobs Not Running?" 2013, http://rcsg.rice.edu/rcsg/shared/scheduling.html.

[19] Italian Grid Infrastructure, "Troubleshooting guide for CREAM," 2013, https://wiki.italiangrid.it/twiki/bin/view/CREAM/TroubleshootingGuide.

[20] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proc. Int'l Conf. for High Perf. Computing, Networking, Storage and Analysis*, 2010.

[21] H. Okamura, S. Miyahara, and T. Dohi, "Dependability analysis of a transaction-based multi-server system with rejuvenation," *IEICE Trans. on Fundamentals of Electronics, Comm. and Computer Sciences*, vol. E86-A, no. 8, 2003.

[22] V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. Vaidyanathan, and W. P. Zeggert, "Proactive management of software aging," *IBM J. Res. and Dev.*, vol. 45, no. 2, 2001.