Memory Degradation Analysis in Private and Public Cloud Environments

Ermeson Andrade*, Fumio Machida†, Roberto Pietrantuono‡, Domenico Cotroneo‡
*_Department of Computing_, _Federal Rural University of Pernambuco_, Recife, Brazil, ermeson.andrade@ufrpe.br
†_Department of Computer Science_, _University of Tsukuba_, Tsukuba, Japan, machida@cs.tsukuba.ac.jp
‡_University of Naples Federico II_, Naples, Italy, {roberto.pietrantuono, cotroneo}@unina.it

Abstract—**Memory degradation trends have been observed in many continuously running software systems. Applications running on cloud computing can also suffer from such memory degradation that may cause severe performance degradation or even experience a system failure. Therefore, it is essential to monitor such degradation trends and find the potential causes to provide reliable application services on cloud computing. In this paper, we consider both private and public cloud environments for deploying an image classification system and experimentally investigate the memory degradation that appeared in these environments. The degradation trends in the available memory statistics are confirmed by the Mann-Kendall test in both cloud environments. We apply causal structure discovery methods to process-level memory statistics to identify the causality of the observed memory degradations. Our analytical results identify the suspicious processes potentially leading to memory degradations in public and private cloud environments.**

Index Terms—**Causality analysis, Memory degradation, Public cloud, Private cloud, Software aging**

I. Introduction

Long-running artificial intelligence (AI) applications such as real-time image analysis or video recognition systems can benefit from the support of cloud computing infrastructure. As for such an infrastructure, deployers could use public clouds, which utilize a shared infrastructure, or private clouds, which utilize an organization's own infrastructure. In some contexts, private clouds are preferred since they reside on a company's own infrastructure, typically firewall protected and physically secured. Since private clouds are owned by the organization, there is no sharing of infrastructure, no multi-tenancy issues, and zero latency for local applications and users.

One of the concerns in the use of cloud computing for AI services is the software reliability issue during a long-time operation. Software aging in cloud computing systems has been experienced and analyzed in many studies [1]–[4]. An image classification system running on a public cloud system also can suffer from memory-related software aging [1]. While the previous works conducted statistical analysis on the measurement data and confirmed the existence of the degradation trends [2], [5], [6], the causality of software aging observed in cloud systems has not been investigated in depth. In particular, for public cloud systems, the entire software architecture and its resource management mechanism are not fully open to users, leading to a challenge in identifying root causes.

In order to investigate the causality of software aging in cloud computing environments, we conduct a set of experiments targeting an image processing system deployed on private or public cloud systems and statistically analyze the processes' memory consumption contributing to software aging. For the private cloud system, We build a CloudStack environment, while for the public system we employ Google cloud platform. On the cloud systems, image classification tasks are processed continuously for 72 hours with three different workload intensities (i.e., low, middle, and high). We collect several system metrics and apply the Mann-Kendall test [7] with Sen's slope estimate [8] to confirm the trends in the memory consumption both in the public and private cloud systems. Next, we delve into the potential causes of memory degradation issues by investigating the causality relation between processes' memory consumption and the observed memory degradation. The results show statistically significant memory degradation trends in both public and private clouds. The causality analysis results indicate the processes most likely to be causally related to the memory degradation for private and public cloud environments. We find that the software aging in private cloud is workload-sensitive, while the software aging in public cloud is not sensitive to workloads. Our findings as well as the causality analysis method can be useful for users or developers in the decision-making process of counteracting various software aging issues encountered in cloud computing environments.

The rest of the paper is organized as follows. Section II describes the related work for software aging analysis. Section III clarifies the research questions. Section IV details our experiments. Section V shows the results of the experiments and statistical analysis. Section VI presents the conclusions and briefly introduces future work.

II. Related Work

In the past decade, we have witnessed growing attention to software aging on cloud-based systems. According to the recent survey report [3], almost one hundred papers in the last ten years address cloud-computing-related software aging issues. Software aging problems in cloud systems are mainly concerned about its adverse impacts on user-perceived quality of services (e.g., availability, reliability, and performance), resulting in loss of users and the market.

The studies on software aging problems are broadly divided into model-based and measurement-based approaches. Many researchers have investigated software aging in cloud computing systems, or more generally in virtualized systems, by exploiting stochastic models such as stochastic Petri nets (SPN) and stochastic reward nets (SRN) [9]–[12], continuous-time Markov chains (CTMC) [13], [14], semi-Markov processes (SMP) [15], [16], as well as combinatorial models such as reliability block diagrams (RBD) [17] and dynamic fault trees (DFT) [18]. In these works, the cloud architectures – including virtual machines (VMs), virtual machine monitor (VMM), and physical host(s) - and the associated rejuvenation strategies are modeled with the aim of computing the optimal time for rejuvenation and of fine-tuning the adopted rejuvenation techniques.

On the other side, measurement-based approach has been employed in many studies for empirically characterizing software aging phenomena in cloud computing systems. In our previous work [1], we use several statistical methods to characterize the software aging in an image classification system deployed on a Google Cloud Computing. In [2], various regression models are used to characterize the software aging on the Eucalyptus cloud computing. A similar statistical analysis is also conducted for software aging analysis in IBM cloud controller systems [6]. Nevertheless, there are only a few works that provide a statistical analysis for identifying the root causes of software aging, and none of them focus on cloud-based systems. In [19], the authors study how software aging impacts different versions and vendors of Android. The study also provides the insight that bloated Java containers are a significant contributor to software aging. Alonso *et al.* [20] develop a framework based on Aspect Programming to monitor the resources used by every application component. The approach is based on the idea of injecting their solution into J2EE architectures at runtime to determine the component root cause of software aging.

The statistical analysis method and the aging indicators considered in this paper are similar to the existing studies. However, in contrast to the existing studies, we also conduct the causality analysis to process memory consumption to find the suspicious processes contributing to memory degradations.

III. Research questions

The objective of our experimental study is to investigate the potential software aging problem in AI services running on private and public cloud computing environments. As an example of an AI application, we consider a real-time image classification system using machine learning. Since the root cause of software aging is the primary interest, we plan the experiments along with the following research questions.

RQ1: *Does an image classification system running on public and private cloud environments encounter any software aging problems?*

RQ2: *What are the potential causes of software aging if any software aging phenomena are observed in the cloud systems?*

RQ3: *How software aging manifestations and causalities are different between private and public cloud?*

To answer the above questions, we conduct the experiments and statistical analysis that are detailed in the next section.

IV. Experiments

In this section, first we explain the cloud-based systems used in our experiments. Next, experimental campaign of our stress tests are detailed. Then, the collected metrics are explained. Finally, the statistical analysis methods are introduced.

A. Setup

Following our previous study that showed memory degradation in a cloud system [1], we use the same image classification program to further investigate the potential causes of software aging in cloud systems. We use MNIST handwritten digit dataset [21] and assign digit recognition tasks for the image classifier. The image classifier is implemented by Python with Keras [22] that is used to build a neural network for classifying handwritten digits. The constructed neural network consists of three layers with 64 filters each, using the Rectified Linear Units (ReLU) for activation. The output layer is a Dense layer with 10 nodes and Softmax activation function. Dropout with probability of 0.2 was used on the fully connected layers. We compiled the model with categorical_crossentropy loss and the adam optimizer. Note that we do not focus on the accuracy of the classifier, but rather look at the degradation of system performance during the continuous operation of classification tasks. In the experiments, input images are generated on a client device and are sent periodically to a virtual server that contains the image classifier. The virtual server is deployed either on a public cloud or a private cloud. For the public cloud system, we use a VM instance from Google's cloud infrastructure. For the private cloud system, we construct a CloudStack platform in a local environment and create a VM instance for executing the image classifier. We do not execute any other VM instances on the same physical machine. In the both cloud systems, the client device acts as a workload generator, which is a program written in Python 3, to generate samples using the MNIST test dataset. The specifications of the system components are given as follows:

- Client device: Apple MacBook Air 11-in, Intel Core i5 1.60GHz, 4 GB, 64 GB, Mac OS X Lion 10.7.
- Google virtual machine: n1-standard-1 (1 vCPUj, 3.75 GB), Debian GNU/Linux 10 located in us-central1-a.
- CloudStack server: Apache CloudStack 4.13.1.0, AMD A8-5500 Quad Core 3.2GHz, 7 GB, 500 GB, CentOS Linux 7.
- CloudStack virtual machine: Medium instance, 1 GB, Ubuntu 20.04.2.0

B. Stress tests

In order to accelerate potential software aging phenomena in cloud systems, we plan to execute stress tests using the workload generator. We apply three different workload intensities for the image classification system: low workload (one image is sent every second), medium workload (one image is sent every 0.5 seconds) and high workload (one image is sent every 0.1 seconds). For the sake of comparative study, we also run the systems with no workload. For each experiment, we execute the system for 72 hours and measure several system metrics. As a result, we have 4 long-running test results for both cloud systems, resulting in 8 test results in total.

C. Metrics

For each experiment, we collect system monitoring data and analyze aging indicators. The aging indicators refer to system variables that can be directly measured and can be related to the software aging phenomena [23]. Even though we collect both user-perceived and system-related aging indicators, user-perceived aging indicators proved to be irrelevant for our analysis, as they did not show any indication of aging. Therefore, in this work, we only consider system-related aging indicators. For system-related indicators, we primarily collect memory depletion indicators, specifically, the total available memory and the resident set size (RSS) of every process running in the system. Our data set also contains metrics related to the CPU usage and I/O statistics, but the observed variations are negligible compared to memory-related indicators. Hence, they are not further considered in the following analysis.

D. Analysis

For statistical analysis to detect potential software aging phenomena, we adopt the conventional Mann–Kendall test (MKT) [7] to analyze the trends of aging indicators, and the Sen's slope estimate [8] to calculate the magnitude of the trends. The Mann-Kendall analysis checks the null hypothesis (H0) that there is no trend in the time series data, while the alternative hypothesis (H1) indicates an upward or a monotonic downward trend in the data. If the p-value of the test is lower than the significance level (α =0.05), then there is statistically significant evidence that a trend is present in the time series data. Once detected the presence of a trend, the Sen's slope estimate is obtained to measure the magnitude of the trend. It is computed as the median of all pairwise slopes between each pair of points in the data set, so that a positive Sen's slope implies a positive trend, while a negative Sen's slope means a negative trend.

Beside aging trends detection and estimation, we run a causal analysis to identify which process is more likely to be causally related to the possible consumption of memory. The aim is to discover causal relations by analyzing statistical properties of the data, which is known as causal structure discovery (CSD). Algorithms in this area aim to infer possible cause-effect relations between sets of variables by means of repeated conditional independence tests between variables and then to represent the resulting relation by directed graphical causal models (DGCM). They therefore can be seen as methods for statistical estimation of parameters describing a graphical causal structure.

We adopt GFCI [24], a combination of Greedy Equivalence Search (GES) [25] and Fast Causal Inference (FCI) [26], which are two well known algorithms covering the two main approaches in this area: *score-based* and *constraint-based* algorithms. GFCI uses GES to build the graph and FCI to prune the graph and find the orientations between the nodes. GFCI has proved to be more accurate in many simulations than the original FCI algorithm. More details can be found in [27].

V. Results

In this section, we present the results of the experiments and the statistical analysis we performed to answer the research questions.

A. Memory trend analysis

To answer RQ1, first we analyze the trends observed in the available memory in public and private cloud systems. Figure 1 shows the available memory for the public cloud experiments during the 72 hours duration. For all the workload levels (low, middle and high) including the none workload case, the available memory sizes are decreasing over time. Table I summarizes the slopes and the p-value of the MKT for the four workload settings, confirming the presence of statistically significant trends, with medium and high workload being the most severe ones.

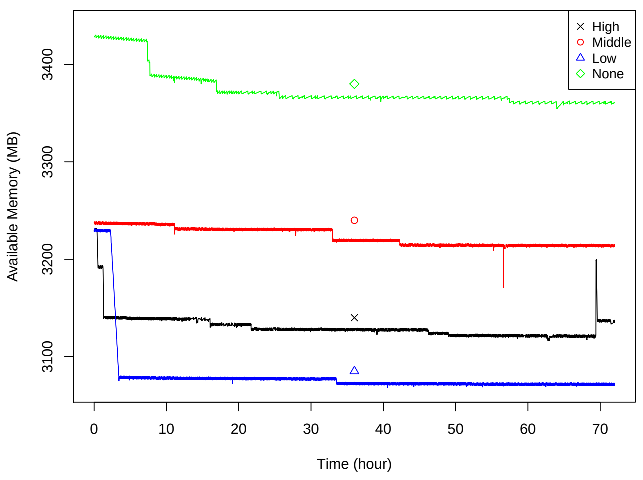

Fig. 1: Available Memory Analysis (Public Cloud).

TABLE I: Public cloud: Slope (KB/h) and p-value of MKT.

Workload	Slope (KB/h)	p-value
High	-3.06e+02	7.68e-24
Middle	-3.93e+02	8.64e-30
Low	-1.30e+02	7.78e-31
None	-3.68e+02	2.76e-29

On the other hand, Figure 2 shows the trend of available memory for the private cloud experiments during the 72 hours duration. In this case, only high workload scenario exhibits a clear decreasing trend in the available memory. Table II shows the slopes and the p-value of the MKT for the four workload settings. Again, we observe statistically significant degradation trend of available memory in the high-workload scenario. However, the other scenarios do not have much impact on memory degradation.

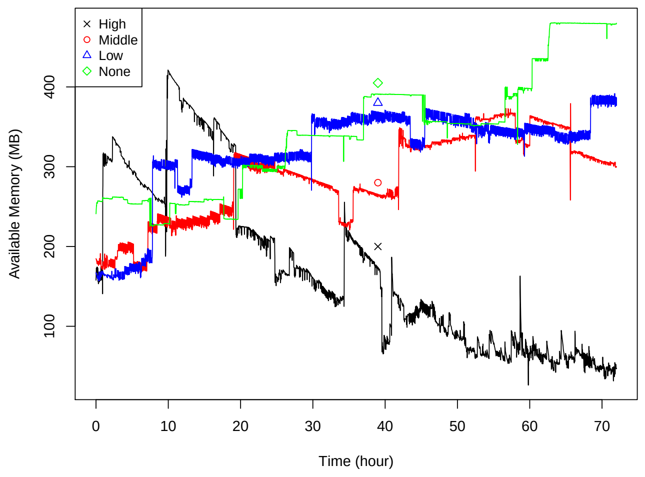

Fig. 2: Available Memory Analysis (Private Cloud).

TABLE II: Private cloud: Slope (KB/h) and p-value of MKT.

Workload	Slope (KB/h)	p-value
High	-4.37e+03	2.33e-24
Middle	2.43e+03	3.04e-14
Low	1.36e+03	8.48e-12
None	2.53e+03	9.35e-19

In summary, we observe statistically significant memory degradation trends both in public and private clouds. However, the trends are not consistent across the different workload intensities. In particular, for the private cloud system, the memory degradation trend is observed only for high workload scenario, implying that software aging may occur only at a certain level of workload intensity.

B. Causality analysis

Since we observed suspicious memory aging trends in both cloud systems in High workload cases, our next question is what are the potential causes of memory degradations (RQ2). To answer this question, we conducted the causality analysis on the process-level memory statistics. We run the GFCI algorithm introduced earlier to find possible causal relations between the processes' memory consumption (in terms of *resident set size*, RSS), which are the hypothesized causes, and the observed degradation (namely, the potential effect) measured as available memory. We considered the top-10 processes, namely the ones with the highest RSS during the experiment.

Tables III-V shows the cases in which a relation has been identified for the public cloud experiments. It shows the names and the PIDs (processes IDs) of the processes with the probability that there exists an arc (i.e., a causal relation) between the process and the total memory consumption (π_{Mem}). The values in the parentheses are the probabilities that there exists a relation, but the algorithm cannot determine what is the cause and what is the effect (i.e., it cannot assign a direction to the edge). We highlighed as boldface text those values of π_{Mem} for which the probability of the directed edge (between the process RSS toward the available memory) is greater than the probability of the undirected edge.

TABLE III: Process causal analysis for Public cloud with High workload.

Public Cloud, High workload		
Process name	PID	π_{Mem}
`google_clock_skew_daemon`	778	**0.64** (+0.064)
`tmux`	844	**0.81** (+0.032)

TABLE IV: Process causal analysis for Public cloud with Middle workload.

Public Cloud, Middle workload		
Process name	PID	π_{Mem}
`google_network_daemon`	774, 775	**1.0** (+0.0)
`systemd-journald`	422	**1.0** (+0.0)
`google_osconfig_agent`	651	**1.0** (+0.0)

TABLE V: Process causal analysis for Public cloud with Low workload.

Public Cloud, Low workload		
Process name	PID	π_{Mem}
`python3.7server.py`	770	**0.81** (+0.064)
`google_accounts_daemon`	1435	**0.93** (+0.032)

A description of the involved processes is given in Table VI. Several of these are pre-installed google daemons of the Google Compute Engine (GCE) instance, some of which (e.g., `google_clock_skew_daemon`, `google_osconfig_agent`) are related to the management of VMs. Nevertheless, `systemd-journald` and `tmux` are not directly related to the GCE instance. Inspecting bug repositories and issue trackers, we found conformation that aging problems exist for all the mentioned processes. `systemd-journald` and `tmux` have been reported to

TABLE VI: Description of the Public cloud processes appearing in the top-10 list of each experiment (Table III-V).

Public Cloud Processes	
PID (Process name)	*Description*
`778` (google_clock_skew_daemon)	Daemon to keep the system clock in sync after VM start and stop.
`844` (tmux)	Terminal Multiplexer.
`775,774` (google_network_daemon)	Daemon to handle the network setup.
`442` (systemd-journald)	It is a system service that collects and stores logging data.
`651` (google_osconfig_agent)	Part of the VM Manager.
`770` (python3.7server.py)	Our classification system.
`1435` (google_accounts_daemon)	Daemon to setup and manage user accounts.

suffer from memory leaks/bloating[1]. The google daemons have been reported to suffer from performance degradation problems that can be related to memory increase[2]. Our classification system also appears in the list, which may be ignored as a root cause because it is more related to the experiment rather than to actual problems in the system.

Tables VII-IX report the process analysis for the private cloud experiments. A description of the involved processes is given in Table X. In this case, the involved processes are, beside our classification task, the `tracker-store` of Linux, which is a file indexing tool also used by `Gnome` (which, in fact, appears in all the experiments). This process seems to be the main cause of memory depletion in the private cloud setting. Inspecting the bug repositories, we found that `tracker-store` is known to have aging-related problems, due to the constant indexing of the tracker, which would be fixed by disabling the tracker-storage service[3]. `gnome` processes use the tracker store service, and also have their own memory leak problems[4].

TABLE VII: Process causal analysis for Private cloud with High workload.

Private Cloud, High workload		
Process name	PID	π_{Mem}
`python3.7server.py`	1620	**0.74** (+0.23)
`gnome-shell`	1007	**0.65** (+0.23)
`tracker-store`	1633	**0.71** (+0.23)

The causality analysis results indicate that many well-known bugs can cause memory degradation with different manifestation patterns in cloud environments. For the public cloud environment, the root cause of memory degradation is mainly related to the Google daemon processes, while for the private cloud environment it is related to the file indexing process.

TABLE VIII: Process causal analysis for Private cloud with Middle workload.

Private Cloud, Middle workload		
Process name	PID	π_{Mem}
`gnome-initial-setup`	1223	0.41 (+0.51)
`update-notifier`	1286	**0.35** (+0.26)

TABLE IX: Process causal analysis for Private cloud with Low workload.

Private Cloud, Low workload		
Process name	PID	π_{Mem}
`gnome-shell`	926	**0.54** (+0.0)
`tracker-store`	1685	**0.26** (+0.26)
`evolution-alarm-notify`	1155	**0.58** (+0.13)

C. Comparative analysis

While we conducted the same experiments on the public and private cloud systems, the observed memory degradation trends and suspicious root causes look quite different. We further statistically investigate the difference of the trends between two cloud environments to provide an answer to RQ3. To find differences under the same workload, we checked for the confidence intervals (CI) of the slopes of each public and private environments. Table XI reports the 95% confidence intervals of the slopes identified by the Theil-Sen's procedure. When the slopes overlap, there is no significant difference between the cases. As can be seen, we do not find any overlaps between public and private cloud environments for all the workload conditions.

In terms of trends, the public cloud case experiences trends under all the workload settings and even under the "none" workload case (see Table I). Trends are of the order of 1.0e+02 KB/hour. This indicates that there might be an aging problem unrelated to the workload (which just emphasizes more the trend). This case may be related to known memory degradation issues from `systemd-journald` and `tmux`, or

[1]Example of `systemd-journald` aging-related issue: https://github.com/systemd/systemd/issues/9141.
Example of `tmux` aging-related issue: https://groups.google.com/forum/#!topic/tmux-users/WiSZy6ft1As.
[2]Example of `google_network_daemon` aging-related issue: https://github.com/GoogleCloudPlatform/compute-image-packages/issues/603.
Example of `google_osconfig_agent` aging-related issue: https://github.com/GoogleCloudPlatform/osconfig/issues/261.
[3]https://askubuntu.com/questions/1187191/tracker-process-taking-lot-of-cpu.
[4]The issue at https://github.com/GSConnect/gnome-shell-extension-gsconnect/issues/1028 reports that `gsconnect` causes `gnome-shell` to lag and possibly memory leak.

TABLE X: Description of the Private cloud processes appearing in the top-10 list of each experiment (Table VII-IX).

Private Cloud Processes	
PID (Process name)	*Description*
`1620` (python3.7server.py)	Our classification system.
`1007,926` (gnome-shell)	It is an user interface of the Gnome desktop.
`1633,1685` (tracker-store)	It is a file indexing and search tool for Linux.
`1223` (gnome-initial-setup)	Gnome Initial Setup.
`1286` (update-notifier)	It scans the computer for software updates.
`1155` (evolution-alarm-notify)	Calendar event notifications.

TABLE XI: Confidence Interval (CI) of the available memory trends. Underlined pairs indicate non-overlapping CIs.

Workload	Public Cloud	Private Cloud
High	[-3.42*e*+02; -2.58*e*+02]	[-4.87*e*+03; -3.97*e*+03]
Middle	[-4.41*e*+02; -3.48*e*+02]	[1.84*e*+03; 2.76*e*-03]
Low	[-1.48*e*+02; -9.46*e*+01]	[1.01*e*+03; 1.83*e*+03]
None	[-5.13*e*+02; -2.64*e*+02]	[3.11*e*+03; 3.89*e*+03]

more interestingly, it might be related to the VM processes we highlighted in Table VI, which are specific processes of the public cloud environment.

The private cloud case indicates a (severe) trend, of the order of 1.0+03 KB/hour, just for the high-workload case (see Table II). This indicates that the private setting is more sensitive to the workload and less sensitive to other processes. The impact of the heavy workload is mainly on the tracker-store file indexing tool, which is used by several services in the environment (see Table X). In short, the results reveal that the public cloud is not workload sensitive, while the private cloud is severely affected. It also indicates that the public cloud might suffer from an aging issue unrelated to the workload.

VI. Conclusion

This paper investigated the memory degradation observed in the private and public cloud environments deploying an image classification system. We confirmed both cloud environments suffer from memory degradation during 72 hours of experiments with high workloads. Through the causality analysis for the process's memory consumption, we identified the suspicious processes that are likely to be the root causes of the memory degradation. For the public cloud system (i.e., Google cloud platform), the memory degradation is found to be related to Google daemon processes. For the private cloud system using CloudStack, we conclude that the tracker-store file indexing process is the potential root cause. The analysis results indicate that the processes should be looked at first when trying to fix the software aging in memory. Although the analysis cannot provide certainty about the aging-causing processes, it significantly restricts the scope of the action (e.g., if a user/developer opts for rejuvenating, s/he can focus on a process-level rejuvenation starting from the highlighted processes). Deeper investigations are planned in the next future to also include further indicators in the root cause analysis for a more precise characterization (e.g., process-level I/O or CPU consumption, as well as caching/buffering/swapping indicators). We also plan to setup an online degradation detection mechanism to warn about suspicious processes and support prompt fault tolerance/proactive recovery actions.

Acknowledgment

This research was partially funded by CNPq - Brazil, grant 406263/2018-3. The work was also supported in part by the grant of University of Tsukuba Basic Research Support Program Type S.

References

[1] E. Andrade, F. Machida, R. Pietrantuono, and D. Cotroneo, "Software aging in image classification systems on cloud and edge," in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2020, pp. 342–348.

[2] J. Araujo, R. Matos, V. Alves, P. Maciel, F. Vieira de Souza, R. Matias Jr., and K. S. Trivedi, "Software Aging in the Eucalyptus Cloud Computing Infrastructure: Characterization and Rejuvenation," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 10, no. 1, pp. 11:1–11:22, 2014.

[3] R. Pietrantuono and S. Russo, "A survey on software aging and rejuvenation in the cloud." *Software Quality Journal*, vol. 28, pp. 7–38, 2020.

[4] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, "Aging-related bugs in cloud computing software," in *2012 IEEE 23rd international symposium on software reliability engineering workshops*. IEEE, 2012, pp. 287–292.

[5] E. Andrade, B. Nogueira, R. Matos, G. Callou, and P. Maciel, "Availability modeling and analysis of a disaster-recovery-as-a-service solution," *Computing*, pp. 1–26, 2017.

[6] H. Sukhwani, R. Matias, K. S. Trivedi, and A. Rindos, "Monitoring and Mitigating Software Aging on IBM Cloud Controller System," in *28th International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2017, pp. 266–272.

[7] H. B. Mann, "Nonparametric tests against trend," *Econometrica: Journal of the econometric society*, pp. 245–259, 1945.

[8] P. K. Sen, "Estimates of the regression coefficient based on kendall's tau," *Journal of the American statistical association*, vol. 63, no. 324, pp. 1379–1389, 1968.

[9] J. Xu, X. Li, Y. Zhong, and H. Zhang, "Availability Modeling and Analysis of a Single-Server Virtualized System with Rejuvenation," *Journal of Software*, vol. 9, no. 1, pp. 129–139, 2014.

[10] A. Rezaei and M. Sharifi, "Rejuvenating high available virtualized systems," in *5th International Conference on Availability, Reliability, and Security (ARES)*. IEEE, 2010, pp. 289–294.

[11] F. Machida, D. S. Kim, and K. S. Trivedi, "Modeling and analysis of software rejuvenation in a server virtualized system," in *Second International Workshop on Software Aging and Rejuvenation (WoSAR)*. IEEE, 2010.

[12] ——, "Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration," *Performance Evaluation*, vol. 70, no. 3, pp. 212–230, 2013.

[13] M. Myint and T. Thein, "Availability Improvement in Virtualized Multiple Servers with Software Rejuvenation and Virtualization," in *Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI)*. IEEE, 2010, pp. 156–162.

[14] T. Thein and J. S. Park, "Availability Analysis of Application Servers Using Software Rejuvenation and Virtualization," *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 339–346, 2009.

[15] F. Machida, V. F. Nicola, and K. S. Trivedi, "Job completion time on a virtualized server subject to software aging and rejuvenation," in *Third International Workshop on Software Aging and Rejuvenation (WoSAR)*. IEEE, 2011, pp. 44–49.

[16] ——, "Job Completion Time on a Virtualized Server with Software Rejuvenation," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 10, no. 1, pp. 10:1–10:26, 2014.

[17] M. Melo, P. Maciel, J. Araujo, R. Matos, and C. Araujo, "Availability Study on Cloud Computing Environments: Live Migration As a Rejuvenation Mechanism," in *43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2013.

[18] J. Rahme and H. Xu, "A Software Reliability Model for Cloud-Based Software Rejuvenation Using Dynamic Fault Trees," *International Journal of Software Engineering and Knowledge Engineering*, vol. 25, no. 09n10, pp. 1491–1513, 2015.

[19] D. Cotroneo, A. K. Iannillo, R. Natella, and R. Pietrantuono, "A comprehensive study on software aging across android versions and vendors," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3357–3395, 2020.

[20] J. Alonso, J. Torres, J. L. Berral, and R. Gavalda, "J2ee instrumentation for software aging root cause application component determination with aspectj," in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*. IEEE, 2010, pp. 1–8.

[21] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[22] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[23] K. S. Trivedi, M. Grottke, and E. Andrade, "Software fault mitigation and availability assurance techniques," *International Journal of System Assurance Engineering and Management*, vol. 1, no. 4, pp. 340–350, 2010.

[24] J. M. Ogarrio, P. Spirtes, and J. Ramsey, "A hybrid causal search algorithm for latent variable models," in *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, A. Antonucci, G. Corani, and C. P. Campos, Eds., 2016, pp. 368–379.

[25] D. M. Chickering, "Optimal structure identification with greedy search," *J. Mach. Learn. Res.*, vol. 3, no. null, p. 507–554, Mar. 2003. [Online]. Available: https://doi.org/10.1162/153244303321897717

[26] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search, 2nd Edition*, ser. MIT Press Books. The MIT Press, December 2001, vol. 1, no. 0262194406.

[27] C. Glymour, K. Zhang, and P. Spirtes, "Review of causal discovery methods based on graphical models," *Frontiers in Genetics*, vol. 10, p. 524, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2019.00524