Computers, Materials & Continua
DOI: 10.32604/cmc.2022.026663
Article

Tech Science Press

Aging Analysis Framework of Windows-Based Systems through Differential-Analysis of System Snapshots

Eun-Tae Jang[1], Sung Hoon Baek[2] and Ki-Woong Park[1,*]

[1]SysCore Lab, Sejong University, Seoul, 05006, Korea
[2]Department of Computer System Engineering, Jungwon University, Chungcheongbuk-do, 28024, Korea
*Corresponding Author: Ki-Woong Park. Email: woongbak@sejong.ac.kr
Received: 31 December 2021; Accepted: 02 March 2022

Abstract: When a Windows-based system is used for an exceedingly long time, its performance degrades, and the error occurrence rate tends to increase. This is generally called system aging. To investigate the reasons for system aging, various studies have been conducted within the range of the operating system kernel to the user application. However, finding an accurate reason for system performance degradation remains challenging research topic. In this study, system monitoring was conducted by dividing a system into 'before software installation,' 'after software installation,' and 'after software removal.' We confirmed that when a software installed in a system is removed, various system elements, such as storage and memory, are not restored to the level prior to the software installation. Consequently, we established a hypothesis regarding the performance degradation of a computer system owing to repeated software installation/removal operations, investigated the correlation between system aging and repeated software installation/removal operations, and proposed a system aging analysis framework for analyzing the reason behind system aging. In the proposed system aging analysis framework, we aim to forcibly age a Windows-based system by repeating the software installation/removal operation by utilizing the system forced aging module. The framework identifies the elements affecting system performance through a differential data analysis of the system time-series data extracted by the system performance extraction and system component snapshot modules. Consequently, the aging analysis framework presented in this study is expected to be effectively utilized as an index for studying system aging.

Keywords: System aging; virtual machine; system analysis

1 Introduction

When the system is utilized for a long time, its performance degrades and the error occurrence rate tends to increase over time, which is generally called system aging [1]. System aging is observed not only in personal computer/server-based computer systems but also in emerging computing systems such as cloud computing [2], mobile [3], and IoT enhanced with 5G network [4] systems, and there

has been a significant amount of research analyzing the reasons for system aging [5]. To determine the aging within the system kernel range, various related studies have been carried out within the range of the system kernel [6–11] to the user application [12], from kernel-based research on system aging based on the development of kernel tracing tools, the collection of parameters related to several kernel subsystems, and a statistical analysis of the collected data, to research hypothesizing that the reason for system aging is an element of the software design and errors in the software itself. However, finding an accurate reason for system aging is still regarded as extremely difficult research [13].

In this study, we identified that various system elements are not restored to their original states after removing an installed software. For such experiments, a system snapshot is taken after dividing the system into three statuses, before software installation, after software installation, and after software removal. The system snapshot can be taken at a high speed, discussed in our previous research [14]. Through the step-by-step system snapshot method, many significant changes were observed regarding various system elements, such as storage and memory, when comparing the system before and after installing and removing a software, respectively. As such, the system aging analysis framework proposed in this study aims to forcibly age a Windows-based system by repeating the software installation/removal operation utilizing the system forced aging module. The framework identifies the elements affecting the system performance through a differential data analysis [15] of the system time-series data extracted by the system performance extraction and system component snapshot modules. To achieve this result, the components of the system aging analysis framework proposed in this study consist of a system forced aging module, system performance extraction module, and system component snapshot module, which are presented in Fig. 1.

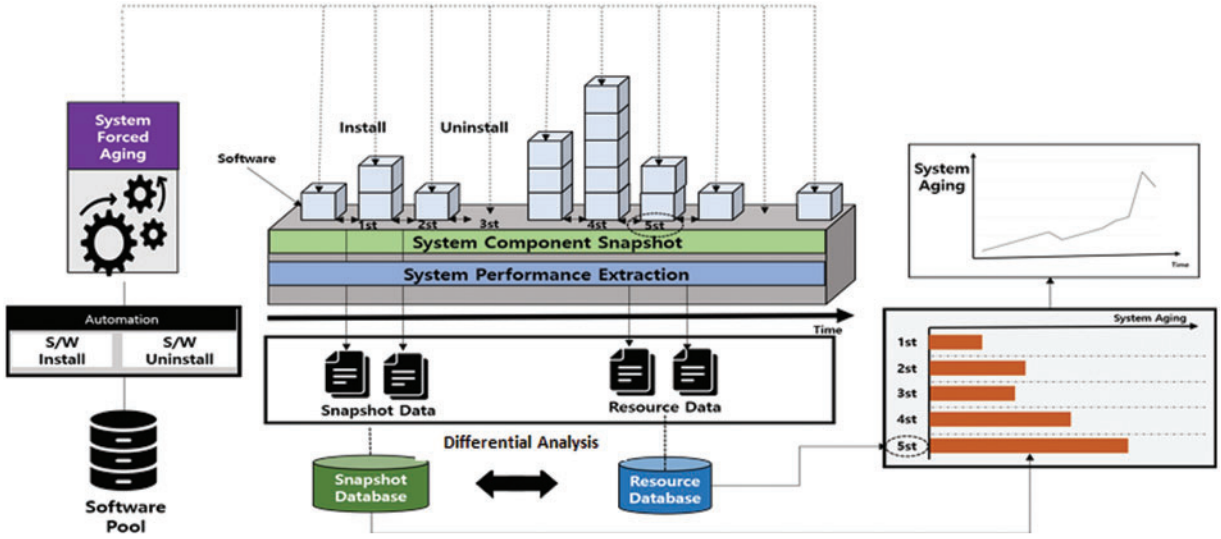

Figure 1: Overall system architecture of system aging analysis framework

In the system aging analysis framework, the system forced aging module was established to effectively repeat the software installation/removal in the Windows™ system. The system was forcibly aged by building the software pool composed of approximately 50 general-purpose software programs with different functions based on the Windows™ operating system inside the system forced aging module and by automating and repeating the installation/removal of the software programs. Furthermore, by building the system performance extraction module; measuring the usage and overhead of the CPU, memory, and storage, which are closely related to the system performance based on a constant

time; and recording the time-series data, a change in the system performance in terms of the software installation/removal operation was observed. Moreover, the system aging analysis framework extracts information related to the system elements that can affect the system performance based on a constant time interval and stores them as time-series data by utilizing the system component snapshot module [14]. The time-series data extracted by the system component snapshot module include the number of files and directories in the system, list of software programs installed in the system, and the registry information. Information of the process that is currently utilizing the memory is included as well. The time-series data extracted from the registry identifies whether the related registry information remains even if it belongs to software that has already been removed. This could be achieved by extracting the differential data from the registry information about the software installed in the system, which identifies if the software removed from the system operates as a process and consumes system resources by extracting information regarding the process utilizing the memory. The contributions of this research are as follows. First, the proposed system aging analysis framework investigates the correlation between the repeated software installation/removal operation most widely conducted in the system. It then analyzes the reason for the system aging through a differential analysis of the data extracted from the repeated software installation/removal operations without using or monitoring the system for a lengthy period of time for experiments. Second, to effectively conduct the software installation and removal operation in a Windows™ system, the software installation and removal operation was automated by building a software pool composed of approximately 50 different general-purpose software programs, which forced system aging. Third, studies that analyzed the reason for system aging were mostly conducted in a Unix-based open-source system. The system aging framework applied in this study analyzes the reason for system aging in the most universally utilized Windows™ operating system.

The remainder of this paper is organized as follows. In Section 2, studies related to system aging are analyzed, their limitations are investigated, and the differences of the present study are deduced. In Section 3, the change in the system is investigated when the software is installed and removed, and the system aging analysis framework design based on the results is described. In Section 4, whether the repeated software installation/removal can affect the system performance is identified through a differential analysis of the time-series data extracted based on the system aging analysis framework described in Section 3. Finally, concluding remarks are provided in Section 4.

2 Related Work

This section reviews the previously conducted research to investigate the reasons for system aging related to our proposed framework. System aging is studied not only for personal computers and server-based computer systems but also for mobile systems. Furthermore, more research is underway on various topics ranging from system kernels to user-level software. Based on an analysis of existing studies, the limitations of the existing studies will be commented on, and their differences from this study will be discussed.

2.1 System Aging in Computer System

2.1.1 Process Management and File System Analysis of Linux™ System

Cotroneo et al. [13], who analyzed the Linux™ system kernel to find the reason for system aging, researched the phenomenon in which software continuously working for a long time undergoes a performance degradation and its error occurrence rate increases over time. For these experiments, the kernel tracing tool was utilized and the reason for the aging of the Linux™ operating system was

investigated through a collection of parameters related to various kernel subsystems and statistical analysis. Based on the results, the main reason for the aging in Linux™ was found to be the process management file system. However, this research suffers from a disadvantage in that the software has to execute for a long time to mimic the actual environment where the system aging takes place.

2.1.2 Analysis of Correlation between Resource Management of the Operating System and System Aging

Gargl et al. [16], who analyzed the correlation between resource management of the operating system and system aging, concluded that the main reason for software aging, in which more frequent errors occurred in software that is used for a longer time, is the exhaustion of the memory swap space of the operating system along with the increase in the resource usage of the operating system. However, in this study, it was not experimentally proven that the memory swap space exhaustion and resource usage increase of the operating system affect the software error.

2.1.3 Analysis of the Reason for System Aging in Cloud Computing Environment

The necessity of stability, availability, and performance is increasing in recent applications that require managing the rapidly increasing demand while providing continuous service. For a cloud computing system, the necessity to provide access to large-scale data and a pool of computing resources is increasing, and in a study by Matos on software aging of Eucalyptus, a software framework was used to materialize a private cloud and hybrid type services. The results proved the occurrence of a high CPU usage of the virtual machine, an increased response time of applications operating in VM, a depletion of RAM, and a detrimental system dependence and decreased performance owing to a successful use of memory swapping [2,8,9]. In this study, research on the aging of software working in a virtual environment was conducted, and has a limitation in that the experiments were carried out only in a virtual environment.

2.2 System Aging in Mobile System

2.2.1 Software Aging Research in the Android™ System

Qiao et al. [17], who studied software aging in the Android™ system, claimed that the software aging phenomenon is widely observed in the Android™ mobile operating system and proposed empirical research regarding the aging-related bug in the Android™ operating system. The experiments were carried out by establishing various aging generation conditions in Android™ and introducing them to the system. Based on the results, it was concluded that the order of priority of the process and the background process affects the system aging in the Android™ operating system. The corresponding research also studied aging in the Android™ system [18–20]. However, a disadvantage occurred in that the verification was not conducted on establishing the aging generation conditions.

2.2.2 Analysis of the Reason for Mobile System Aging through Machine Learning

Huo et al. [21], who analyzed system aging through machine learning in Android™ systems, claimed that system aging is a phenomenon that frequently occurs in Android™ system and introduced a machine learning method to investigate the reason for this phenomenon. For the experiments, three machine learning algorithms, namely a decision hierarchy tree, support vector machine (SVM), and deep belief network (Bn), were applied and compared to analyze the reason for system aging in mobile systems. Z. Hao and J. Liu proposed a software aging detection-and-rejuvenation scheme targeting the Android™ platform based on the boundary equilibrium generative adversarial network [22] and state clustering technology, called GAN-ASD [3]. Their results showed that the proposed scheme

outperformed conventional rejuvenation schemes in terms of user experience and rejuvenation cost. The presented aging detection and rejuvenation schemes are a meaningful step forward for anti-aging of computer systems. Accordingly, the testbed construction to collect datasets and provide traceability toward the root elements of system aging presented in this study, could identify and improve critical software components causing system aging.

2.3 Limitations of the System-Aging Related Research

Based on a literature review on system aging, we identified that the performance degradation phenomenon, namely system aging, is common not only in personal computers and server-based computing systems but also in newly emerging computing systems, such as cloud computing, mobile, and IoT systems. Researchers studying system aging have hypothesized the close correlation between long-term usage of software and system performance degradation. Moreover, they studied the causes for system aging with an emphasis on long-term software usage [1,6,7,12], such as the exhaustion of the memory swap space of the operating system [9,13] and resource exhaustion [23,24] when using a software for a long time. Therefore, operating a software for long periods to establish experimental conditions and mimic actual system aging is rather difficult.

3 Framework Design for System Aging Analysis

To construct an analysis framework to prove the hypothesis that computer system performance degrades owing to repeated software installation/removal operations in a Windows™ system, this study presents a framework for tracing the cause of system aging. Specifically, a framework for analyzing the correlation between software installation/removal work and step-by-step system snapshot is presented in this study. The proposed framework is depicted in Fig. 2. The proposed system aging analysis framework mainly consists of the system forced aging module, system performance extraction module, and system component snapshot module. It aims to forcibly age the Windows-based system by repeating the software installation/removal operation by utilizing the system forced aging module. The framework traces the reasons for system aging through a differential data analysis of the collected data as well.

3.1 System Forced Aging Module Design

The system was designed to effectively automate the installation/removal of programs in a Windows™ system and to force the aging process of the Windows™ system using the corresponding module. A software pool is built inside the corresponding module, i.e., a type of software storage where approximately 50 general-purpose software programs operating in the system are stored. The representative software stored in the software pool is presented in the table inside of Fig. 2.

3.2 System Performance Extraction Module Design

The system performance extraction module is designed to identify whether the repeated installation/removal operations in Windows™ system affect system performance. As presented in Fig. 3, the identification is based on the system resource data collected by extracting and storing data on the overhead and performance of the CPU, memory, and storage. Because the differential analysis of the system performance after completing the software installation/removal operation is critical to trace the reasons behind system aging, this module is activated for each event of software installation/removal operation.

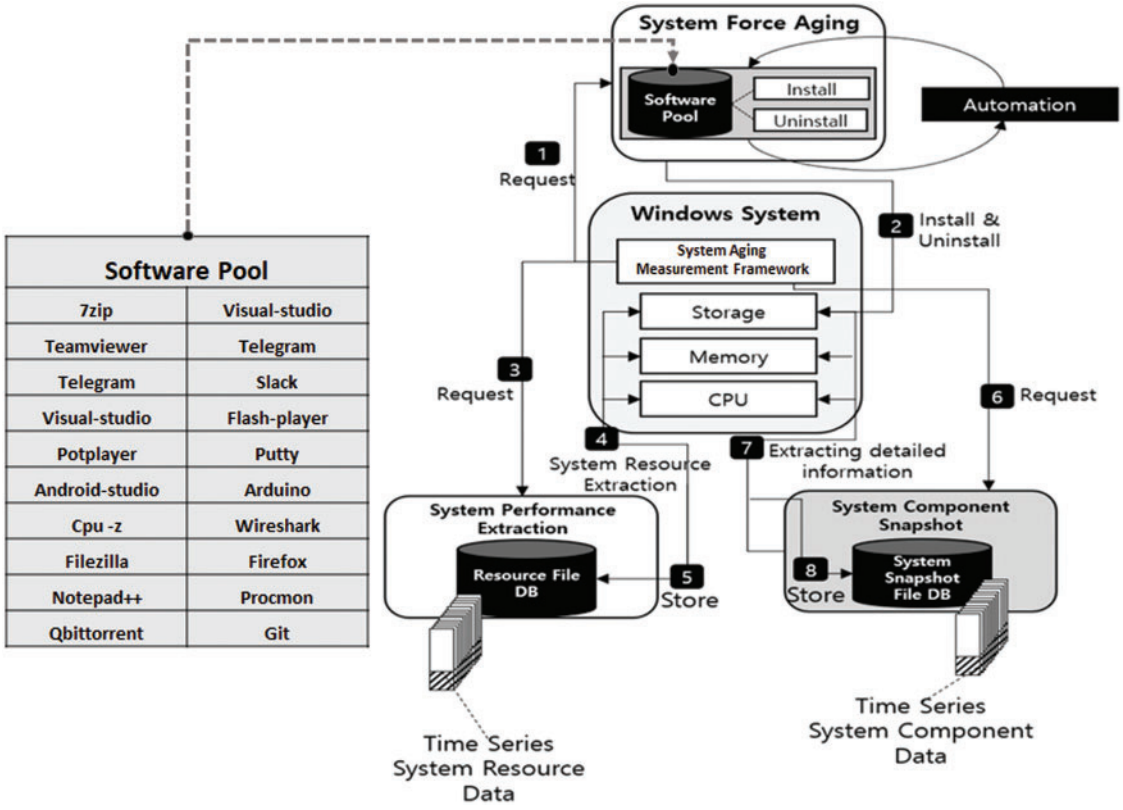

Figure 2: Overall operational flow system aging analysis framework

3.3 System Component Snapshot Design

As depicted in Fig. 4, the system component snapshot module observes the system state after performing the software installation/removal operation in the system, and is called last among the components of the system aging analysis framework for analyzing the reason behind system aging. The reason for calling the system component snapshot module last is because when this module is working in the system aging analysis framework component module, it generates the highest amount of overhead. Thus, if it operates prior to the system performance extraction module, it can affect the system resources measured using the system performance extraction module. The data collected in the system component snapshot module are composed of information that can affect system performance, such as dummy files of the removed software left in the storage, lists and numbers of processes working in memory, and registry data. The system components affecting system performance are extracted through differential analysis of the system performance data, which is correlated with the time-series among after completing the software installation/removal operations. The abovementioned measurement on system performance is conducted in the system performance extraction module, which is based on the *Sysinternals* [25] suite.

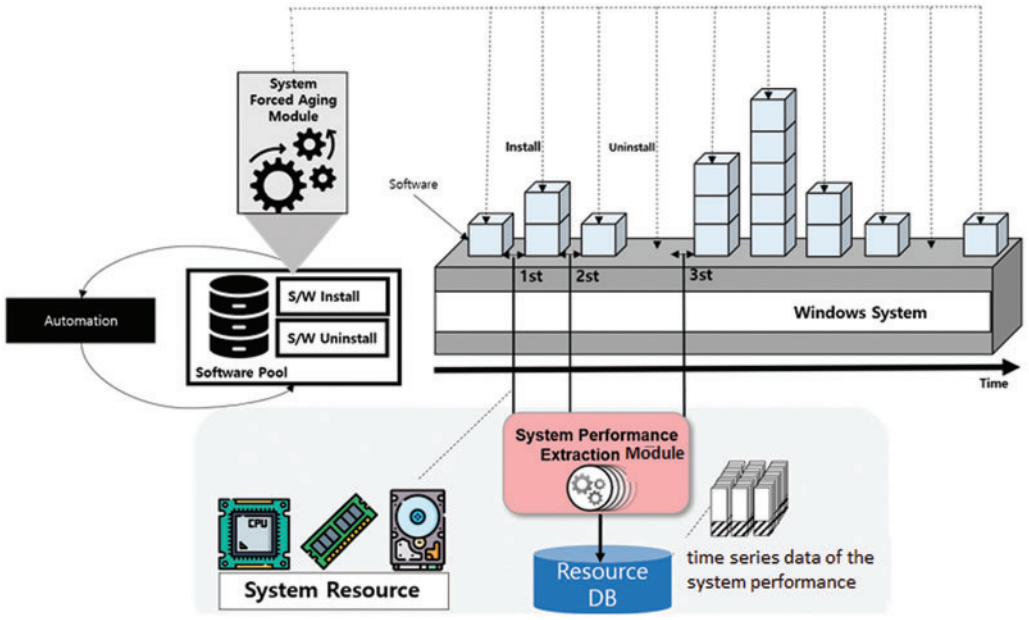

Figure 3: Operational flow of system performance extraction module

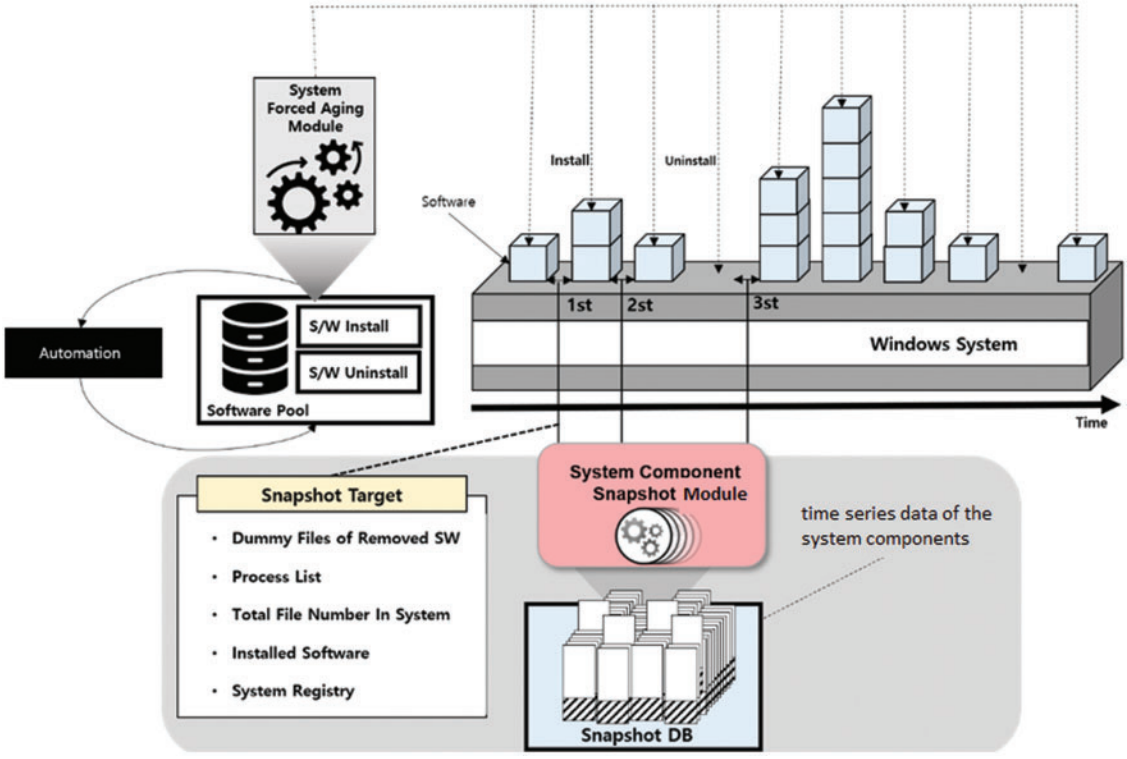

Figure 4: Operational flow of system component snapshot module

4 Evaluation

4.1 Experimental Conditions

To analyze the aging in the Windows™ system using the system aging analysis framework proposed in this study, the experiments were carried out by producing a virtual machine (VM) based on VMware, the experimental conditions of which are listed in Tab. 1. The operating system of the host server used in the experiments is Windows™ 10. For the CPU, an Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz was utilized and 32 GB of memory was used. The virtual machines used in the experiments applied Windows™ 10, and 2 processor cores, 4 GB of memory, and a 60-GB hard disk were utilized. Moreover, the system aging analysis framework tool proposed in this study was developed using Python 2.7.

Table 1: Experimental environment to analyze the aging in the Windows™ System

Host	
Operating system	Windows™ 10
CPU	Intel(R) Core i7-8700 3.20 GHz
Memory	32.00 GB
Virtual machine (User)	
Operating system	Windows™ 10
Processor cores	2
Memory	4.00 GB
Hard disk	60 GB
System aging analysis framework tool	
Development language	Python 2.7

4.2 Experiments

4.2.1 Identification of the System Components Changed through Repeated Software Installation/Removal

The experiments were carried out with the purpose of observing the variables in the system when repeated software installation/removal operations are conducted in a Windows™ system utilizing the proposed system aging framework. For the experiments, 10 software programs were installed in the system using the system forced aging module, the system was rebooted, and the 10 software programs were removed. Afterward, the system was rebooted, and the variables that had changed in the system were observed using the system performance extraction module and system component snapshot module of the system aging analysis framework. Based on the results, it was identified that the repeated software installation and removal operations in the Windows™ system leaves numerous dummy files in the Windows™ system storage. These dummy files are either a part of the corresponding software or are used files despite the software having already been removed. The experimental results of the dummy files on the repeated software installation/removal operation are shown in Fig. 5. It was confirmed that when the software is installed and removed, a considerable number of dummy files are left in storage. The results showed that when the 10 software programs are installed and removed, approximately 20,000 dummy files are stacked up in the storage.

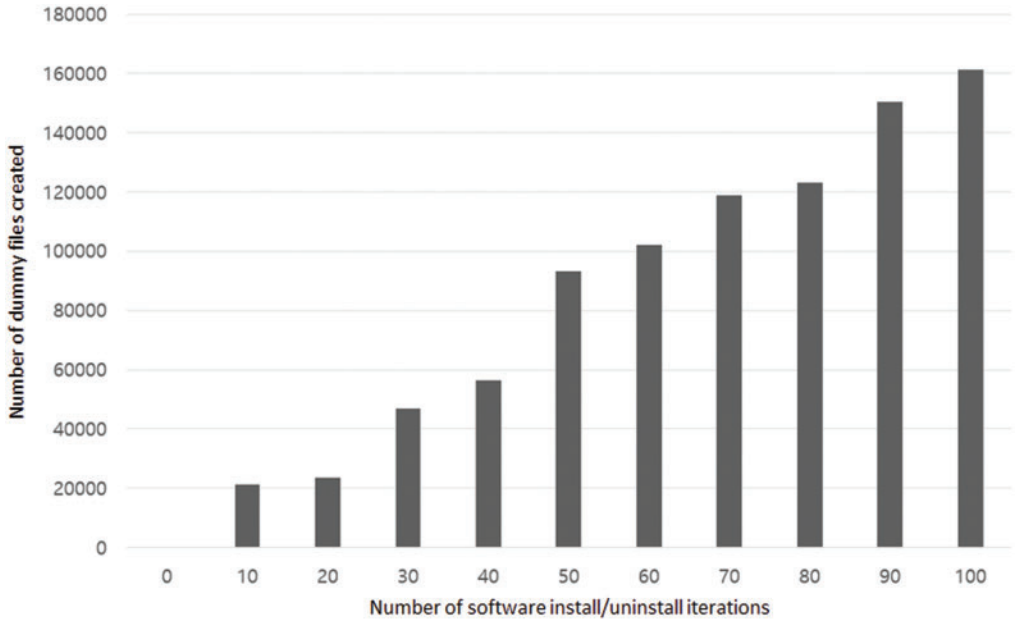

Figure 5: Measurement of the number of dummy files due to repetitive software installation/removal

4.2.2 Measurement of System Performance on the Software Dummy Files owing to the Repeated Software Installation/ Removal

After establishing the appropriate experimental conditions, an experiment was designed to observe the changes in system performance degradation as the number of dummy files in the Windows™ system increases. The dummy files used in the experiment are presented in Fig. 5. Based on the previous tests, we identified that the software dummy files are stacked up in the storage as the software installation and removal operations are repeated in the Windows™ system. Furthermore, we confirmed that the memory usage gradually increases with more dummy files in the system storage. Tab. 2 reports the experimental results regarding the change in system performance when the number of the dummy files increases in the system. One reason for this result is that an increase in the number of files leads to an increase in I/O transactions owing to the antivirus program and index daemon driven by the Windows™ system.

Table 2: System performance analysis according to the number of dummy files

Number of dummy files	Memory usage (Max: 4096 MB)	CPU usage
10,000	2217 MB	3%
20,000	2474 MB	2%
30,000	2812 MB	4%
100,000	3482 MB	5%

4.2.3 Measurement of Change in System Overhead and Resource Usage owing to the Repeated Software Installation/ Removal Operation in the Windows™ System

The experiment described in this section aimed to identify if the repeated software installation/ removal operations in the Windows™ system affect the CPU, memory, or storage. In other words, this experiment examined the key components affecting system performance using the proposed system aging framework. Ten software programs were installed in the system using the system forced aging module. Then, the system was rebooted, and the 10 programs were removed. The system was rebooted again, and the system resource information was measured utilizing the system aging analysis framework as shown in Fig. 3. When software installation and removal were repeated in the Windows™ system, the initial CPU usage was approximately 3%. By contrast, when 100 software programs were installed and removed, the CPU usage was approximately 5.3%. This confirms that repeated software installation and removal degrades system performance in terms of average CPU utilization. Moreover, the total memory usage of the system was initially approximately 28%, but after 100 software installation/removal procedures, it was approximately 79.9%. This result confirms that repeated software installation and removal operations lead to a permanent memory consumption owing to the removed-but-left-behind processes. In terms of storage usage, approximately 38% of the total storage was initially used in the Windows™ system prior to the experiments. However, when the software installation and removal were repeated, the storage usage was observed to gradually increase. When 100 cycles of software installation/removal were conducted, the storage usage increased to approximately 74% despite the absence of installed software. Based on these experimental results, we confirmed that repeated software installation and removal operations causes unnecessary memory and storage resource consumption and can affect the overall performance in a Windows™ system.

5 Conclusions

In this study, we hypothesized that repeated software installation and removal operations have a closed relationship with performance degradation in a Windows™ system and presented a system aging analysis framework for tracing the origin of system aging through a differential system snapshot analysis. The system aging analysis framework proposed in this study was used to investigate the correlation between software installation/removal operations, namely the most commonly conducted operations in the system, and system aging. In addition, the factors affecting system performance were identified through differential system data extracted from the repeated software installation/removal operations, which originally required the computer to be used by a human and was time-intensive. Subsequently, system aging was accelerated efficiently by applying an automated software installation and removal scheme in the Windows™ system and building a software pool composed of approximately 50 different general-purpose software programs inside the system forced aging module. The experimental results derived using the proposed framework are as follows. First, based on a performance evaluation and multiple experiments, we confirmed that unremoved dummy files continued stacking up in the system storage and occupied a considerable amount of storage space as the software installation/removal operation was repeated in the Windows™ system. We identified that when one software program was removed approximately 60%–70% of the files remained in storage on average, which accounted for approximately 50%–60% of the installed software size. As the number of unremoved dummy files increases, the system memory overhead also increases; thus, the overall system performance degrades. Moreover, experiments were conducted to identify if a software already removed from the system works as a process in the memory and consumes system resources by collecting information related to the processes operating in memory. The initial number of processes working in memory was 128. After 100 software installation and removal operations, the number

of working processes was 129. However, we discovered that the process remnants of the removed software were causing an additional operation inside the Windows™ system process. Consequently, the proposed system aging analysis framework is expected to be effectively used as an index for studying system aging.

This research has one major limitation. We conducted this research with an emphasis on designing a framework for a system aging analysis and correlation analysis between system aging and repeated software installation/removal. Consequently, in the future, we will focus on an effective analysis of the extracted time-series data enhanced with a workload generation considering behavioral human-usage pattern to simulate system aging and present the effective reaction for the discovered system aging.

Funding Statement: This work was supported by the ICT R&D program of MSIT/IITP (Project No. 2021-0-01816, A Research on Core Technology of Autonomous Twins for Metaverse, 10%) and National Research Foundation of Korea (NRF) (Project No. NRF-2020R1A2C4002737, 90%) grants funded by the Korean government.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. Cotroneo, R. Natella, R. Pietrantuono and S. Russo, "Software aging analysis of the linux operating system," in *Proc. 2010 IEEE 21st Int. Symp. on Software Reliability Engineering*, San Jose, California, USA, pp. 71–80, 2010.

[2] P. Shruthi and G. Nagaraj, "An analysis of software aging in cloud environment," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 6, pp. 5985–5991, 2020.

[3] Z. Hao and J. Liu, "GAN-ASD: Precise software aging state detection for Android system based on BEGAN model and state clustering," in *Proc. 20th IEEE/ACM Int. Symp. on Cluster, Cloud and Internet Computing (CCGRID)*, Melbourne, Australia, pp. 212–221, 2020.

[4] D. Minovski, C. Åhlund, K. Mitra and I. Cotanis, "Anomaly detection for discovering performance degradation in cellular IoT services," in *Proc. 2021 IEEE 46th Conf. on Local Computer Networks*, Virtual Conference, pp. 99–106, 2021.

[5] W. Yan and N. Ansari, "Why anti-virus products slow down your machine?," in *Proc.18th Int. Conf. on Computer Communications and Networks*, San Francisco, CA, USA, pp. 1–6, 2009.

[6] S. Garg, A. van Moorsel, K. Vaidyanathan and K. S. Trivedi, "A methodology for detection and estimation of software aging," in *Proc. 9th Int. Symp. on Software Reliability Engineering (Cat No.98TB100257)*, Paderborn, Germany, pp. 283–292, 1998.

[7] R. Matos, J. Araujo, V. Alves and P. Maciel, "Experimental evaluation of software aging effects in the eucalyptus elastic block storage," in *Proc. 2012 IEEE Int. Conf. on Systems, Man, and Cybernetics*, Seoul, Korea, IEEE, pp. 1103–1108, 2012.

[8] T. Hayashi and S. Ohta, "Performance degradation detection of virtual machines via passive measurement and machine learning," in *Artificial Intelligence: Concepts, Methodologies, Tools, and Applications*, IGI Global, U.S.A, pp. 2092–2110, 2017.

[9] F. Machida, J. Xiang, K. Tadano and Y. Maeno, "Aging-related bugs in cloud computing software," in *Proc. IEEE 23rd Int. Symp. on Software Reliability Engineering Workshops*, Dallas, TX, USA, pp. 287–292, 2012.

[10] Z. Mavuş and P. Angın, "A secure model for efficient live migration of containers," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 10, no. 3, pp. 21–44, 2019.

[11] J. Bouché, L. Atkinson and M. Kappes, "ShadowHeap: Memory safety through efficient heap metadata validation," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 12, no. 4, pp. 4–21, 2021.
[12] M. Grottke, L. Li, K. Vaidyanathan and K. S. Trivedi, "Analysis of software aging in a Web server," *IEEE Transactions on Reliability*, vol. 55, no. 3, pp. 411–420, 2006.
[13] D. Cotroneo, R. Natella and R. Pietrantuono, "Predicting aging-related bugs using software complexity metrics," *Performance Evaluation*, vol. 70, no. 3, pp. 163–178, 2013.
[14] S. Choi and K. Park, "Timeline-based container restore via a computationally efficient snapshot," in *Proc. 6th Int. Conf. on Next Generation Computing 2020*, Busan, Korea, pp. 277–279, 2020.
[15] D. Bae and J. Ha, "Performance metric for differential deep learning analysis," *Journal of Internet Services and Information Security*, vol. 11, no. 2, pp. 22–33, 2021.
[16] S. Garg, A. van Moorsel, K. Vaidyanathan and K. S. Trivedi, "A methodology for detection and estimation of software aging," in *Proc. Ninth Int. Symp. on Software Reliability Engineering (Cat No.98TB100257)*, Paderborn, Germany, pp. 283–292, 1998.
[17] Y. Qiao, Z. Zheng and F. Qin, "An empirical study of software aging manifestations in Android," in *Proc. 2016 IEEE Int. Symp. on Software Reliability Engineering Workshops (ISSREW)*, Ottawa, ON, Canada, pp. 84–90, 2016.
[18] C. Weng, J. Xiang, S. Xiong, D. Zhao and C. Yang, "Analysis of software aging in Android," in *Proc. 2016 IEEE Int. Symp. on Software Reliability Engineering Workshops (ISSREW)*, Ottawa, ON, Canada, pp. 78–83, 2016.
[19] D. Cotroneo, F. Fucci, A. K. Iannillo, R. Natella and R. Pietrantuono, "Software aging analysis of the Android mobile os," in *Proc. 2016 IEEE 27th Int. Symp. on Software Reliability Engineering (ISSRE)*, Ottawa, ON, Canada, pp. 478–489, 2016.
[20] M. Park, G. You, S. Cho, M. Park and S. Han, "A framework for identifying obfuscation techniques applied to Android apps using machine learning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 10, no. 4, pp. 22–30, 2019.
[21] S. Huo, D. Zhao, X. Liu, J. Xiang, Y. Zhong *et al.,* "Using machine learning for software aging detection in Android system," in *Proc. 2018 Tenth Int. Conf. on Advanced Computational Intelligence (ICACI)*, Xiamen, China, pp. 741–746, 2018.
[22] D. Berthelot, T. Schumm and L. Metz, "Began: Boundary equilibrium generative adversarial networks, Machine Learning (cs.LG)," 2017. [Online]. Available: https://arxiv.org/abs/1703.10717.
[23] C. Guo, H. Wu, X. Hua, D. Lautner and S. Ren, "Use two-level rejuvenation to combat software aging and maximize average resource performance," in *Proc. 2015 IEEE 17th Int. Conf. on High Performance Computing and Communications*, New York, NY, USA, pp. 1160–1165, 2015.
[24] D. Cotroneo, R. Natella, R. Pietrantuono and S. Russo, "Software aging and rejuvenation: Where we are and where we are going," in *Proc. 2011 IEEE Third Int. Workshop on Software Aging and Rejuvenation*, Hiroshima, Japan, pp. 1–6, 2011.
[25] M. Russinovich, *Sysinternal Suite*, Microsoft Press, 2022. [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysinternals-suiteM.