Analysis of Software Aging in a Blockchain Platform

Douglas Dias
Department of Computing
Federal Rural University of Pernambuco
Recife, Brazil
douglas.dias@ufrpe.br

Fumio Machida
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
machida@cs.tsukuba.ac.jp

Ermeson Andrade
Department of Computing
Federal Rural University of Pernambuco
Recife, Brazil
ermeson.andrade@ufrpe.br

***Abstract*—Blockchain platforms have gained popularity in recent years and integrated with other digital technologies like Internet of Things (IoT) and Artificial Intelligence (AI) for multiple-business purposes. Software aging is a common issue in many long-running software systems, but little has been experienced in the context of blockchain platforms. To narrow this gap, this work aims to characterize potential software aging issues in the Cardano blockchain platform that is considered the largest cryptocurrency adopting proof-of-stake. By performing statistical analysis on the measurement data of the Cardano blockchain deployed in two environments with different configurations, we found a symptom of software aging through memory degradation that was confirmed by the Mann-Kendall test. By analyzing the running processes, we identify the *cardano-node* (the main process of the platform) as the process possibly responsible for such degradation.**

***Index Terms*—Blockchain, Cardano, Memory degradation, Software aging.**

I. Introduction

Software permeates our lives increasingly, including in critical areas such as safety and health, which motivates not only the development of more efficient and secure systems, but also the constant monitoring of their operation for long periods. In the long run, software systems can suffer from gradual performance degradation and resource exhaustion that are known as software aging. Software aging is a phenomenon empirically observed in many software systems that progressively degrades performance and eventually causes a system failure [1]. While it is not always possible to prevent its occurrence, its effects can be limited, its damage can be reversed, and its causes can be understood [2]. However, for blockchain technologies, which are behind platforms for trading cryptocurrencies and executing smart contracts [3], there are still very few works addressing this phenomenon and its implications in such environments.

Although blockchains have been strongly associated with cryptocurrencies since 2008, which was the year in which Satoshi Nakamoto published Bitcoin's famous white paper [4], the blockchain itself is nothing more than a data structure that enables the secure recording of transactions. However, the implementation of Bitcoin in 2009 and its eventual popularization demonstrated the possibilities for such technology and influenced the emergence of several projects with the most diverse applications [5]. Medicalchain, for example, is a London-based company that uses blockchain to store patients' medical history [6]. It aims to provide patients with the power to control their health-related data (e.g.: appointment or exam history) and allow them to share their records with medical organizations privately and securely. BlockPharma, on the other hand, uses blockchain to track medications and prevent counterfeits [7]. Over the past few years, different areas are producing solutions built using blockchain technology, showing its relevance and value. Therefore, it is essential to analyze the reliability aspects of these platforms, especially in long-running instances.

Although there are several blockchain platforms (e.g., Hyperledger Fabric, Ethereum, Tezos and Corda), Cardano stands out for being one of the largest and most important blockchains that use a proof-of-stake consensus mechanism. Therefore, this work aims to investigate the existence of software aging symptoms and their possible causes in a Cardano platform through an experimental evaluation of the execution of a *cardano-node*. In order to investigate the software aging, we connected a *cardano-node* to the Cardano's mainnet to perform synchronization tasks considering different environments and workloads. We collected various system performance metrics (e.g., memory/swap/CPU usage, response time, network, among others) and applied the Mann-Kendall test [8] with Sen's slope estimator [9] to confirm the symptoms of software aging on the *cardano-node*. In addition, we identified processes suspected of causing software aging. The results revealed that, on one hand, when the *cardano-node* was deployed on a computer with minimal requirements, it only supported low workload. It also was prematurely terminated at about forty hours after the start of the experiments due to memory degradation. On the other hand, when the *cardano-node* is deployed on a computer with the recommended requirements, it supported all workloads (low, medium and high). However, it showed indication of memory degradation for all of them. Our findings, as well as the analysis approach, can be useful for users or developers in the decision-making process to tackle software aging problems encountered in blockchain environments.

This paper is organized as follows. Section II briefly describes Cardano. Section III details the related work. Section

IV details our experiments. Section V shows the results of the experiments and statistical analysis. Finally, Section VI presents the conclusions and briefly describes future work.

II. Cardano

Cardano is one of several open source blockchain platforms available [10]. It was developed with the goal of implementing a sustainable and balanced ecosystem that facilitates the needs of users as well as other systems. Cardano has attractive features such as scalability, interoperability, decentralization, and low energy consumption. When compared to other platforms like Bitcoin or Ethereum, that use proof-of-work consensus mechanism, Cardano consumes less power per node as it employs proof-of-stake for consensus, which leads to a smaller carbon footprint. [11].

Cardano has two networks into which third-party applications and systems can integrate: the testnet, which is used for integration tests, and the mainnet, which is the Cardano's core network. The networks are made up of nodes, called *cardano-nodes*, which are interconnected and work together to validate transactions and blocks through consensus. The *cardano-node* is the key component of Cardano as it sustains the network. The node is deployed through a container that runs on computers connected to the network and implements several components such as ledger, networking, consensus, storage, among others. The nodes communicate with each other through a set of IPC (Inter-Process Communication) mechanisms.

III. Related Work

Software aging is a phenomenon that plagues many long-running complex computing systems, which exhibit performance degradation or an increasing failure rate [12]. Due to its practical implications, both academia and industry have increasingly studied this topic, resulting in numerous scientific publications and registered patents [13]. As new technologies are created and adopted, the number of publications involving software aging also grows [14]. However, studies that aim to analyze software aging on blockchain platforms are still scarce.

Studies on software aging are broadly divided into model-based approaches [15] and measurement-based approaches [16]. Many researchers have investigated software aging in computing systems by exploiting stochastic models such as non-homogeneous Markov chain [17], continuous-time Markov chains (CTMC) [18] and stochastic Petri Nets (SPN) [19]. On the other hand, measurement-based approaches have been employed in many studies to empirically characterize software aging phenomena in computer systems from the earliest work [20] to the recent studies [16], [21]. In this approach, which is adopted in this work, measurement data are collected from the system under study to infer the trend of resource consumption or performance degradation caused by software aging . Nevertheless, in relation to blockchain platforms, there is only one work available in the literature that analyzes software aging, but the authors focused on a proof-of-work blockchain network called Hyperledger Fabric [22].

Differently, our work aims to investigate the existence of software aging on the Cardano platform by analyzing the memory degradation during the execution of a *cardano-node* connected to the mainnet. To the best of our knowledge, this is the first work to analyze software aging on a proof-of-stake blockchain network. Additionally, different environment configurations and workloads are considered. We also performed a process analysis to identify the process suspected of causing software aging.

IV. Experiments

The objective of the experiments is to investigate potential software aging issues on a Cardano platform under different environment settings and workloads. If any symptoms of software aging are identified, we also aim to analyze the main causes of such phenomenon. Through the experiments, we try to answer the following research questions.

RQ1: Does the execution of Cardano encounter any software aging issues?

RQ2: If so, what are the potential causes of Cardano software aging and how significant are they?

In the following, we explain the experimental setup, the stress tests, the metrics and the analysis techniques.

A. Setup

To perform the experiments, we set up two computers with *cardano-node* version 1.29.0 (one with the minimal configuration and the other one with the recommended configuration) and *cardano-cli* command line interface both obtained from the *cardano-node source code*[1]. The *cardano-node* is configured to be connected to the mainnet, which is the main network of the Cardano platform, composed of several other nodes running on various other devices. Each round of experiments is only started after the nodes were fully synchronized with the mainnet.

In each computer responsible for running the *cardano-node*, we created and ran scripts to collect data from the resources and running processes. We also created a load generator (Python program) to request the information about the last block of the blockchain through the command *getCardanoTip()*. The load generator is deployed on a client node and uses the *cardano-cli* to request the information. On the *cardano-node* side, a Python/Flask program is created that provides an endpoint responsible for receiving the requests from the client node. All requests are sent to the *cardano-nodes* over a wireless local area network. The adopted architecture is described in Figure 1. The specifications of the system components are presented below:

- Computer running the *cardano-node* on the minimum requirements (**Computer A**): Intel Core i5-4200U (3 MB cache, dual core @2.6 GHz, 4 threads), 8 GB RAM

[1]https://github.com/input-output-hk/cardano-node

1600Mhz DDR3, SSD 240GB, SATA, read 500MB/s, write 350MB/s, Ubuntu 20.04.3

- Computer running the *cardano-node* on the recommended requirements (**Computer B**): Intel Core i7-7700HQ, (6 MB cache, quad core @3.8 GHz, 8 threads), 16 GB RAM, 1TB HDD, SATA, 5400 RPM, Linux Mint 20.3
- Raspberry Pi 3 B+ (**Client node**), 1GB LPDDR2 SDRAM, 16GB SD, Raspberry Pi OS.

The version of the *cardano-node* we performed our experiments was 1.29.0. This version is the latest version of the Cardano platform whose requirements required at least 8GB of RAM for a *cardano-node* [23]. Therefore, Computer A (detailed above) meets the minimum requirements of such a version and represents the baseline of our experiments. The Computer B, on the other hand, meets the recommended requirements of this version. It also meets the requirements to be used as a server running a stake pool [24], maintaining and combining the stake of various stakeholders, processing transactions and creating new blocks.

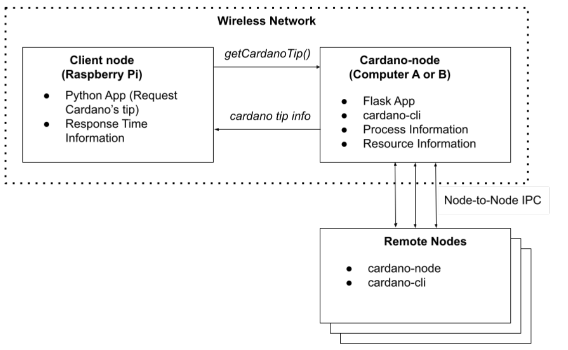


Fig. 1: System setup.

B. Stress Test

To accelerate the possible symptoms of software aging on each computer running the *cardano-node*, we ran stress tests using the load generator. We applied three workload intensities: low (synchronous requests sending at most 3 requests per second), medium (synchronous requests sending at most 8 requests per second), and high (asynchronous requests sending 20 requests every second). For each experiment, we ran the environment for 72 hours or until the *cardano-node* is interrupted due to an operating system crash or interruption. As a result, we have 3 long-running test results for each computer configuration, resulting in a total of 6 test results. However, Computer A, which only met the minimum requirements, only supported the low workload, consequently, we only obtained 4 test results at the end of the experiments.

C. Metrics and Analysis

For each experiment, we collected system monitoring data and analyzed aging indicators. As mentioned in [25], aging indicators refer to system variables that can be measured directly and can be related to the software aging phenomenon. For the aging indicators, several metrics were collected such as response time, CPU/memory usage, disk I/O, etc. In this study, however, we focused on investigating the presence of memory degradation as the main aging indicator, since other metric variations were negligible compared to memory-related indicators. Therefore, memory (RAM) and swap usage are the aging indicators used in this work.

For statistical analysis, we used the conventional Mann–Kendall test (MKT) to analyze the trends of aging indicators, and the Sen's slope estimate to calculate the magnitude of the trends. The Mann-Kendall test adopts two hypotheses: the null hypothesis, that there is no trend present in the data, and the alternative hypothesis, that there is a statistically significant increasing or decreasing trend in the data. If the *p-value* returned from the test is less than the significance level of 0.05, we reject the null hypothesis and accept the alternative hypothesis. This indicates that there is significant evidence that a trend exists. Once detected the presence of a trend, the Sen's slope estimate is obtained to measure the magnitude of the trend, so that a positive Sen's slope implies a positive trend, while a negative Sen's slope means a negative trend. Finally, the five running processes that used the most memory were collected and analyzed in order to identify the potential causes of aging of software.

V. Results

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

A. Preliminary Analysis

In this subsection, we present preliminary results regarding the execution of the experiments. Table I details the results related to RAM usage, while Table II presents the results related to swap usage. For each configuration, we computed the mean, standard deviation (SD), and confidence interval with 95% of confidence level.

TABLE I: Average memory usage.

Memory usage (%)					
Computer/Workload	**Mean**	**SD**	**Confidence Interval**		
			Lower	**Upper**	**SE**
Computer A / Low	95.4189	9.14528	95.05169	95.78612	0.18726
Computer B / Low	58.53823	6.48198	58.34771	58.72875	0.09718
Computer B / Medium	64.37922	4.95793	64.23321	64.52522	0.07447
Computer B / High	65.30512	3.94030	65.18756	65.42268	0.05996

TABLE II: Average swap usage

Swap usage (%)					
Computer/Workload	**Mean**	**SD**	**Confidence Interval**		
			Lower	**Upper**	**SE**
Computer A / Low	76.39333	30.4485	75.17071	77.61594	0.62348
Computer B / Low	0.00000	0.00000	0.00000	0.00000	0.00000
Computer B / Medium	0.10000	0.00000	0.10000	0.10000	0.00000
Computer B / High	0.03323	0.06981	0.03115	0.03532	0.00106

For the Computer A, under a low workload (3 synchronous requests per second), the average memory usage was 95.4% with the upper limit of the confidence interval of 95.7%. The value for the standard deviation was 9.1%, which denotes a dispersion in the data during the execution of the experiments. We remark that memory usage remained high throughout the execution of the experiments with *cardano-node* being prematurely terminated around the fortieth hour due to memory degradation. Since the maximum limit for Computer A was already reached when running the lowest workload, experiments for medium and high workloads were not possible on this environment. As described in the *cardano-node* system requirements, 8GB of RAM is indeed a minimum requirement for running Cardano version 1.29.0. However, the *cardano-node* termination after a few hours of experiments indicates that further investigation is needed to identify possible causes of this resource exhaustion.

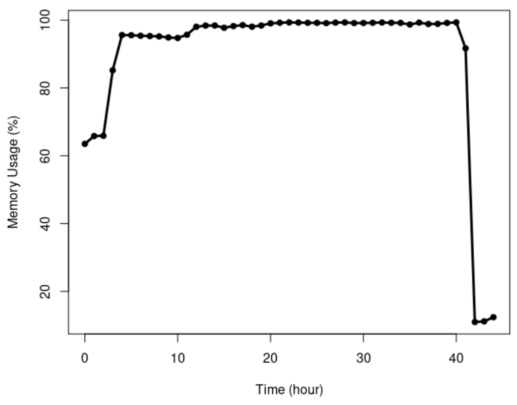
For the Computer B, the average memory usage show variations according to the workload. That is, there is a direct influence between workload and memory usage. The difference is significant when comparing the differences between workloads, especially low workload (3 requests per second) and medium workload (8 requests per second). Additionally, it is worth noting that the confidence intervals do not overlap for the workloads (low, medium and high), meaning that the means are statistically different.

The swap usage of Computer B for all workloads was very low. Although, for the medium workload case, the Computer B consumed more swap than the high workload case, it did not vary over time and had 0.1% usage since the beginning. Additionally, the confidence interval shows that the lower and upper bounds are equal to the mean, and the standard deviation is equal to 0. Therefore, for the medium workload, we consider that the execution of the *cardano-node* did not influence the swap usage. Under high workload, the swap showed a slight increase over time, but extremely low. Regarding the Computer A, the results showed a considerable consumption of swap, especially considering that swap is used only when the system runs out of available memory, which explains the *cardano-node* interruption mentioned above.

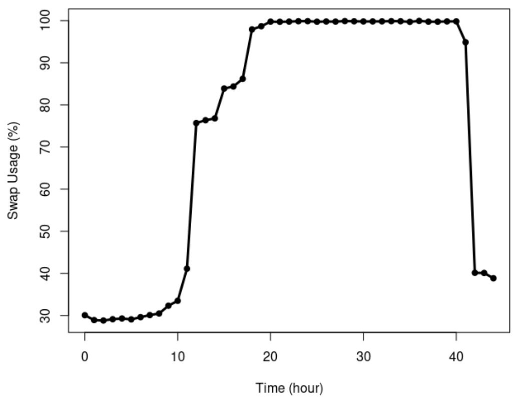
B. Analysis of Potential Software Aging

In this subsection, the research questions presented above are examined. First, we analyze the degradation trends in available memory (RQ1). Then, if suspicion software aging is confirmed, our next goal is to find out what are the potential processes causing such memory degradation (RQ2). Note that for Computer A only the low workload was used for experiments, since the others were not supported by the adopted environment. For Computer B, on the other hand, low, medium and high workloads were used.

1) Computer A: Figure 2 presents the memory and swap analysis for Computer A. As it can be seen, there is a rapid growth for the memory usage after the third and fourth hour of the experiment. From this point, the memory consumption drops and increases slightly, but remains high throughout the experiments (above 90%). The swap usage, on the other hand, start to increase slowly, so that after the tenth hour, it increases rapidly reaching its maximum around 20 hours.



(a) Memory



(b) Swap

Fig. 2: Memory and swap usage for Computer A.

After the fortieth hour, both memory and swap start to drop as a consequence of the termination of the *cardano-node* by the OOM Killer[2]. Inspecting the issues on the *cardano-node* repository, we found that *cardano-node* is known to have aging-related problems, due to memory leak[3][4].

Table III presents the results for the Mann-Kendall test and the Sen's slope, considering up to the fortieth hour. That is, before the *cardano-node* termination. As the results show, the trend is upward for both memory and swap usage, since the slopes are positive. Besides, the *p-values* are less than 0.05, which indicates the existence of a trend for both memory and swap usage. Therefore, the results reveal a statistically significant degradation of available memory and swap for the low workload, suggesting the suspicion of software aging.

To further investigate the underlying causes of the suspected aging phenomenon, we performed a process analysis for the

[2]The Out of Memory (Out Of Memory) Killer is a process that the Linux kernel employs when the system is critically low on available memory. It analyzes the running processes and terminates one or more of them to free up memory to keep the system running.

[3]https://github.com/input-output-hk/cardano-node/issues/769

[4]https://github.com/input-output-hk/cardano-node/issues/460

TABLE III: Mann-Kendall test and Sen's slope for Computer A.

MK Test and Sen's Slope - Computer A			
Metric	***p-value***	**Slope**	**Trend**
Memory	2.09e-08	1.36e+04 KB/h	Growth
Swap	4.14e-12	4.30e+04 KB/h	Growth

TABLE IV: Description of the processes for Computer A.

Top 5 memory-consuming processes	
PID	**Description**
13416 (cardano-node)	The *cardano-node* process.
1634, 1163 (gnome-shell)	GNOME Desktop Environment Graphical Shell.
2225133, 23306 (nautilus)	GNOME's default file manager.

Cardano plataform. We created a Python program to collect information about the processes running on the *cardano-node* every hour. This program uses the `ps` command with `eo` to retrieve process information such as Process Identification Number (PID), Resident Set Size (RSS), and Virtual Set Size (VSZ). Figure 3 lists the five processes that consumed the most memory on Computer A, while Table IV presents the description of such processes. The results reveal that the *cardano-node* process consumed on average more than 80% of the available memory, while the four other processes with the highest consumption, all of which are part of the operating system (in this case, Ubuntu), used little memory. Therefore, we can conclude that the execution of *cardano-node* largely contributed to the memory exhaustion that occurred during the experiment, indicating that it may be the potential responsible for the software aging.

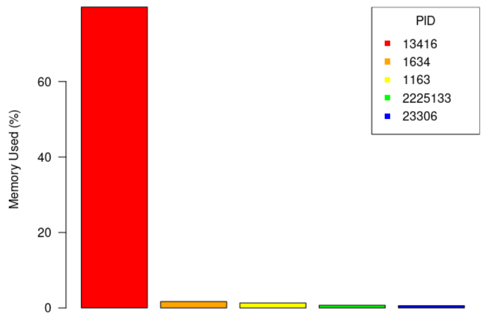


Fig. 3: Top 5 memory-consuming processes for Computer A.

2) Computer B: Figure 4 presents the memory analysis for Computer B, considering the low workload. The results show a large increase in memory consumption from the fourth hour of the experiment. But, before that, the memory usage experienced a small increase in the first hour of the experiment, followed by two hours of relative stability. After the fourth hour, the behavior was repeated for some time. That is, a slight increase in the following hours, followed by a period of stability. At the thirtieth hour, a sharp growth is observed that lasted until approximately the fortieth hour. After that, memory usage was stabilized. For swap usage, on the other hand, there was no variation, remaining at zero percent from the beginning to the end of the experiment.

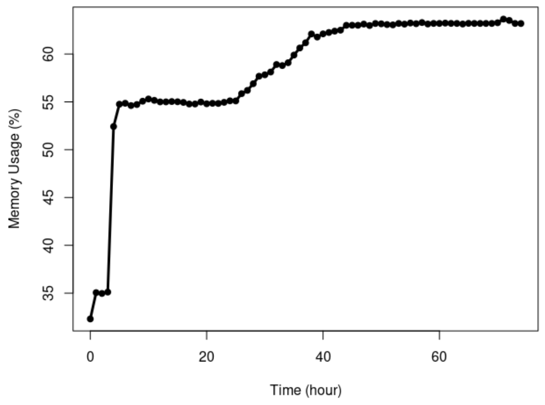


Fig. 4: Memory usage for Computer B under low Workload.

Table V presents the results for the Mann-Kendall test and Sen's Slope, taking into account Computer B under low workload. As the test result shows, the memory usage has a tendency to degrade, since the *p-value* is less than 0.05, which indicates potential software aging. Furthermore, the Sen's Slope is positive, indicating an upward trend. On the other hand, swap usage has no trend at all, since the *p-value* is zero. Figure 5 lists the five processes that consumed the most memory on Computer B for the low workload, while Table VI presents the description of such processes. Again, the *cardano-node* led by far among the most memory-consuming processes, using about 8GB of the 16GB available on the computer. The other processes are part of the operating system and consumed little memory, the same characteristics present in the execution of the experiment on the Computer A.

TABLE V: Mann-Kendall test and Sen's slope for Computer B under low workload.

MK Test and Sen's Slope - Computer B			
Metric	***p-value***	**Slope**	**Trend**
Memory	2.22e-16	3.30e+04 KB/h	Growth
Swap	0	0 KB/h	None

TABLE VI: Description of the processes for Computer B under low workload.

Top 5 memory-consuming processes	
PID	**Description**
4085 (cardano-node)	The *cardano-node* process.
1356 (cinnamon-replace)	Command that starts/restarts Cinnamon, which is a window manager.
862 (Xorg-core)	One of Xorg processes, which is a system software and protocol that provides a base to GUIs.
152020, 1787000 (cinnamon-screensaver)	The default screensaver in a GNOME desktop.

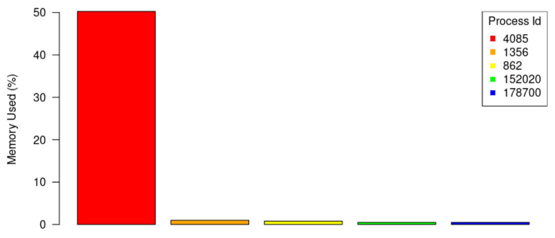


Fig. 5: Top 5 memory-consuming processes for Computer B under the low workload).

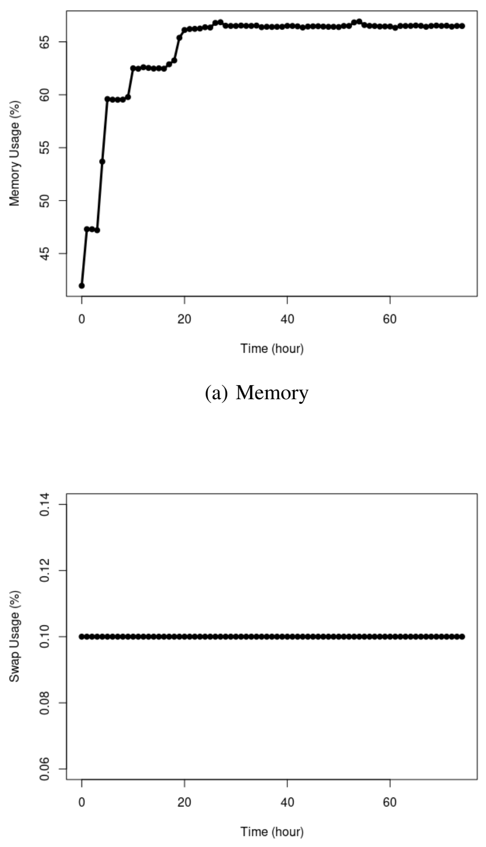
Figure 6 presents the memory and swap results for Computer B, considering the medium workload. The rapid increase in memory usage (see Figure 6a) during this experiment is remarkable when compared to the previous experiment (see Figure 4a). In addition to reach a higher average memory usage by the end of the first hour, about 47.5% versus 35% in the low workload case, the 60% memory usage threshold was reached in the tenth hour. Note that under the low workload this did not happen until just before the fortieth hour. For low and medium workloads, memory usage was more stable, just above 60%. However, for the medium workload case, the utilization was higher than 65%, a value that was not reached during the low workload experiment. The results related to the swap is presented in Figure 6b. Despite not being zero, the swap usage did not change during the experiments, which indicates that the small use of 0.1% is not a consequence of the execution of the *cardano-node*.

Table VII shows the results for the Mann-Kendall test and the Sen's Slope estimate. Similar to the low workload case, there is a degradation trend in the memory usage, which is confirmed by the *p-value* less than 0.05. Furthermore, the value of Sen's Slope estimate is positive, indicating an upward trend and highlighting the occurrence of memory degradation. Therefore, these results show indications of potential software aging. For the use of swap, there is no trend at all, as the *p-value* and the Sen's Slope estimate are equal to zero. By analyzing the five processes that consumed the most memory on Computer B under medium Workload (see Figure 7 and Table VIII), we can conclude that the *cardano-node* was, again, the process that consumed the most. Additionally, just like in the low workload experiment, the other processes with the highest memory usage are part of the operating system, but they consumed very little compared to *cardano-node*.

TABLE VII: Mann-Kendall Test and Sen's Slope for Computer B under medium workload).

Mann-Kendall Test and Sen's Slope			
Metric	***p-value***	**Slope**	**Trend**
Memory	1.307e-13	4.18e+03 KB/h	Growth
Swap	0	0 KB/h	None

For the experiment on Computer B under high workload, whose memory and swap usage results are shown in Figures 8 a and b, the increase in memory usage was even faster than in the experiment under medium workload, reaching more than 60% of memory usage before the end of the third hour. Although the growth in memory usage slowed after 60%, it continued to increase followed by periods of relative stability, as in the medium workload case. However, it reached higher memory usage, approaching 70%. Additionally,



(b) Swap

Fig. 6: Memory and swap usage for Computer B under medium workload.

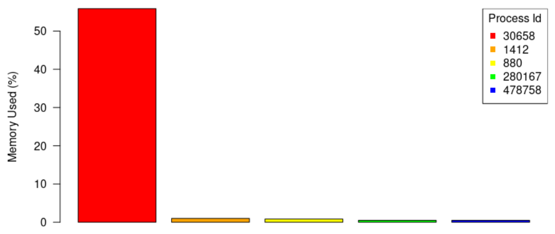
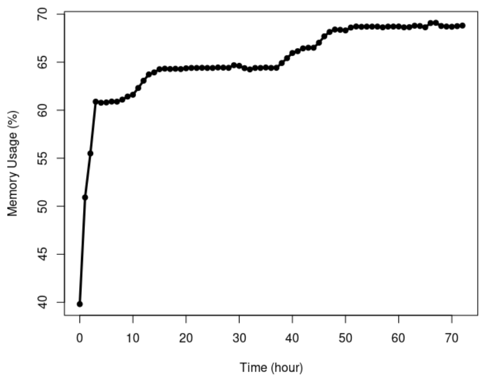


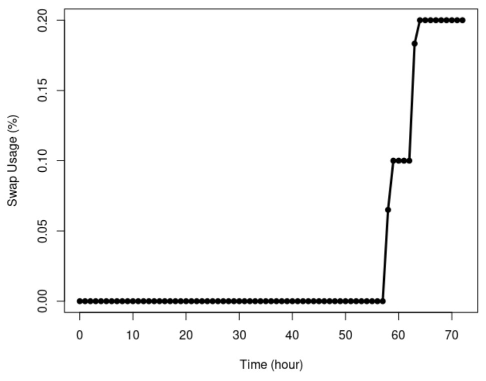
Fig. 7: Top 5 memory-consuming processes for Computer B under medium workload.

TABLE VIII: Description of the processes for Computer B under medium workload.

Top 5 memory-consuming processes	
PID	**Description**
30658 (cardano-node)	The *cardano-node* process.
1412 (cinnamon-replace)	Command that starts/restarts Cinnamon, which is a window manager.
880 (Xorg-core)	A process of Xorg, which is a system software and protocol that provide a base to GUIs.
280167, 478758 (mint-refresh-cache)	A process of Linux Mint's Update Manager.



(a) Memory



(b) Swap

Fig. 8: Memory and swap usage for Computer B under high workload.

for this experiment, swap usage (see Figure 8b) showed a small positive variation, starting to increase around the sixtieth hour, reaching a period of stability before growing again and reaching its maximum. However, the consumption was relatively low, reaching only 0.2% at the end of the experiment.

The results for the Mann-Kendall test and the Sen's slope estimate are shown in Table IX. They reveal a degradation trend for both memory and swap. In both cases, the *p-value* was less than 0.05, indicating the potential software aging. Furthermore, the value of Sen's slope estimate was positive, indicating an increasing trend. Figure 9 lists the five processes most consumed memory on Computer B for the high workload case, while Table X presents the description of such processes. Similar to other workloads, the processes were repeated for this case as well. The *cardano-node* was the process that consumed the most memory, while the other processes consumed little memory. In addition, the *cardano-node* process reached the highest memory usage considering all other workloads.

TABLE IX: Mann-Kendall test and Sen's slope for Computer B under high workload.

Mann-Kendall Test and Sen's Slope			
Metric	***p-value***	**Slope**	**Trend**
Memory	2.22e-16	2.04e+04 KB/h	Growth
Swap	2.22e-16	6.86 KB/h	Growth

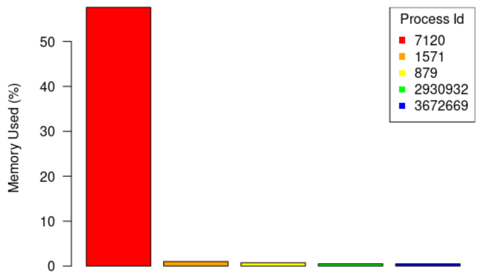


Fig. 9: Top 5 memory-consuming processes for Computer B under high Workload.

TABLE X: Description of the processes for Computer B under high workload.

Top 5 memory-consuming processes	
PID	**Description**
7120 (cardano-node)	The *cardano-node* process.
1571 (cinnamon-replace)	Command that starts/restarts Cinnamon, which is a window manager.
879 (Xorg-core)	A process of Xorg, which is a system software and protocol that provide a base to GUIs.
2930932, 3672669 (mint-refresh-cache)	A process of Linux Mint's Update Manager.

VI. Conclusion

This work investigated the symptom of software aging in the Cardano platform by analyzing memory degradation, considering various environments and workloads. We statistically confirmed potential software aging, accompanied by trends in memory usage degradation across all experiments, in addition to increases in swap usage for some cases. The results, together with the process analysis, are strong indications that the Cardano platform does indeed show symptoms of software aging, and that *cardano-node* is the main suspect. As future work, we plan to consider other blockchain platforms and aging indicators such as response time, power consumption, CPU utilization, etc. In addition, we plan to explore the effects of memory degradation on user-perceived metrics (e.g., throughput and response time) and possible rejuvenation solutions.

References

[1] M. Grottke, R. Matias, and K. S. Trivedi, "The fundamentals of software aging," in *IEEE 1st International Workshop on Software Aging and Rejuvenation*, 2008, pp. 1–6.

[2] D. Parnas, "Software aging," in *16th International Conference on Software Engineering*, 1994, pp. 279–287.

[3] M. D. Pierro, "What is the blockchain?" Computing in Science & Engineering, 2017, pp. 92–95.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review, 2008.

[5] W. Baiod, J. Light, and A. Mahanti, "Blockchain technology and its applications across multiple domains: A survey." Journal of International Technology and Information Management, 2021, pp. 78–119.

[6] IBM Blockchain Pulse, "Blockchain beyond cryptocurrency," https://www.ibm.com/blogs/blockchain/2019/12/blockchain-beyond-cryptocurrency/, 2019, accessed: 2022-07-23.

[7] Blockpharma, https://www.blockpharma.com/, accessed: 2022-07-23.

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

[9] 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.

[10] N. Deepa, Q.-V. Pham, D. C. Nguyen, S. Bhattacharya, B. Prabadevi, T. R. Gadekallu, P. K. R. Maddikunta, F. Fang, and P. N. Pathirana, "A survey on blockchain for big data: approaches, opportunities, and future directions," *Future Generation Computer Systems*, 2022.

[11] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85 727–85 745, 2019.

[12] 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 (JETC)*, vol. 10, no. 1, pp. 1–34, 2014.

[13] N. A. Valentim, A. Macedo, and R. Matias, "A systematic mapping review of the first 20 years of software aging and rejuvenation research," in *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2016, pp. 57–63.

[14] R. Pietrantuono and S. Russo, "A survey on software aging and rejuvenation in the cloud," *Software Quality Journal*, vol. 28, no. 1, pp. 7–38, 2020.

[15] F. Machida and P. R. Maciel, "Markov chains and petri nets," in *Handbook Of Software Aging And Rejuvenation: Fundamentals, Methods, Applications, And Future Directions*. World Scientific, 2020, pp. 93–126.

[16] R. Pietrantuono, J. Alonso, and K. Vaidyanathan, "Measurements for software aging," in *Handbook Of Software Aging And Rejuvenation: Fundamentals, Methods, Applications, And Future Directions*. World Scientific, 2020, pp. 73–90.

[17] Y. Bao, X. Sun, and K. S. Trivedi, "A workload-based analysis of software aging, and rejuvenation," *IEEE Transactions on Reliability*, vol. 54, no. 3, pp. 541–548, 2005.

[18] F. Machida and N. Miyoshi, "Analysis of an optimal stopping problem for software rejuvenation in a deteriorating job processing system," *Reliability Engineering & System Safety*, vol. 168, pp. 128–135, 2017.

[19] K. Vaidyanathan, R. E. Harper, S. W. Hunter, and K. S. Trivedi, "Analysis and implementation of software rejuvenation in cluster systems," in *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2001, pp. 62–71.

[20] S. Garg, A. Van Moorsel, K. Vaidyanathan, and K. S. Trivedi, "A methodology for detection and estimation of software aging," in *Ninth International Symposium on Software Reliability Engineering*. IEEE, 1998, pp. 283–292.

[21] E. Andrade, R. Pietrantuono, F. Machida, and D. Cotroneo, "A comparative analysis of software aging in image classifiers on cloud and edge," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[22] C. Melo, F. Oliveira, J. Dantas, J. Araujo, P. Pereira, R. Maciel, and P. Maciel, "Performance and availability evaluation of the blockchain platform hyperledger fabric," *The Journal of Supercomputing*, pp. 1–23, 2022.

[23] IOHK, "Cardano node releases," https://github.com/input-output-hk/cardano-node/releases, 2022, accessed: 2022-04-18.

[24] IOHK, "Stake pool minimum system requirements," https://iohk.zendesk.com/hc/en-us/articles/900001208966-Stake-Pool-Minimum-System-Requirements, 2022, accessed: 2022-08-23.

[25] 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.