International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 1
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *20 January 2024*

Autonomous AI Self-Healing Distributed Systems Using Deep Reinforcement Learning (DRL)

Anuraag Mangari Neburi

Vice President

ABSTRACT: The cloud-native and distributed systems of modernity create complex failures that can hardly be detected and recovered manually or through the rule of thumb. The current paper is a proposal of an Autonomous AI Self-Healing Distributed System based on Deep Reinforcement Learning (DRL). The structure integrates real time observability, artificial intelligence fault detection and a DRL based action engine to make autonomous choices and take autonomous action by selecting and executing the recovery actions. Controlled failure injection was used as a quantitative experimentation. The findings indicate that there are a great deal of improvement in Mean Time to Repair (MTTR), increased availability of the system, and there is also a low rate of false positive remediation when using the traditional ones. The results prove that DRL facilitates efficient, persistent, and self-reliant system resilience.

KEYWORDS: DRL, Autonomous AI, Distributed System, Self-Healing

I. INTRODUCTION

Microservice-based distributed systems along with the cloud platform are popular because they are scaled and flexible. They are very elaborate systems that are also likely to experience breakdowns easily. The devices used to monitor traditionally only produce an alert and still require human intervention to resolve the problem, further adding to the recovery time. The limitation that will be discussed in this paper is the proposal of an independent self-healer framework based on Deep Reinforcement Learning (DRL). The system is also capable of identifying failures and automatically making decisions and taking corrective measures. The purpose of the investigation is to quantitatively determine the effectiveness of DRL-based self-healing in enhancing the speed of recovery, the system availability, and the reliability of the functioning of the system.

II. RELATED WORKS

Self-Healing and Autonomic Systems

The idea of self-healing systems dates back to autonomic computing which was suggested to handle the increasing complexity of the IT systems in the contemporary world [6]. Autonomic systems are aimed at self-configuring, self-healing, self-optimizing and self-protecting. Early self-healing models concentrated on monitors that are guided by rules, and recovery responses which are predefined.

As an example, SHoWa proposed an autonomous web-based application programming structure that examines the performance metrics, finds anomaly by statistical correlation and takes automatic recovery measures [6]. The findings demonstrated correct anomaly detection under a very low performance overhead, which demonstrated that autonomous healing is possible within the production environment.

Nevertheless, such initial systems were very dependent on fixed rules and thresholds. These strategies are effective in familiar (patterns of) failure, but are ineffective in dynamic and large-scale settings where the failure changes with time.

Such restrictions were also found in self-healing applications to pervasive computing and handheld devices where recovery mechanisms that were implemented as middleware enhanced resilience but did not have adaptive intelligence [12]. Such systems could repair predetermined failures though were not able to learn about new failures or to make better decisions as time went on.

Self-healing has also been investigated to supply Industry 4.0 autonomy and coordination needs in industrial and manufacturing settings [11]. Autonomic cycles have been used to identify failures based on event logs, and activate healing activities.

Although they were effective, these methods once again relied on predetermined reasoning and was not utilizing the capabilities of learning-based decision-making to the full extent. The initial research on self-healing formed a well-built architectural basis but emphasized the necessity of adaptive and learning-based intelligence to work with complex and changing systems.

AI and AIOps for Detection

Manual monitoring was no longer an option as cloud computing, microservices, and large-scale distributed systems were growing in size. This led to the creation of the Artificial Intelligence to IT Operations (AIOps) that applies machine learning to automate the IT operations of detection and diagnosis [5][14]. The original research of AIOps was on log, metric, and KPI anomaly detection.

The approach of machine learning and deep learning in the detection of log anomalies has recorded good results in the

field of other studies. It was proved that online and constantly updated models increase detectors accuracy by significant ratio and decrease false alarms to the lowest level [13].

The hybrid approaches to performing a combination of tools such as PCA and neural networks improved the detection of log anomalies and reduced the pseudo-positives in the actual systems like Hadoop and microservice benchmarks [15]. Similarly, more elaborate models of deep learning such as GRU-GAN have also been proposed to detect abnormalities of KPI in a highly dynamic data center environment, and have proven to be better than the older unsupervised models [17].

Despite these being made, most AIOps solutions stop at detection and generation of alerts. Root cause analysis (RCA) and remediation are characterized by leaving it to the human operators. AIOps datasets and competitions of the public were advantageous in the refinement of benchmarking and model generalization, and were largely concentrated on the detection and localization challenge rather than the autonomous recovery [14]. The studies of microservice self-adaptation substantiated this weakness since most solutions focused on the monitoring step and applied reactive and centralized approaches [16].

There is a good indication of such findings that there is gap between intelligent detection and intelligent action. Though one of the effective ways of problem identification is the use of AI-based monitoring systems, the control is seldom performed to the full extent to define and take corrective actions on its own. This is a weakness that pushes the utilization of reinforcement learning that can learn the optimal actions by receiving feedbacks on the system environment.

Reinforcement Learning and DRL

Reinforcement Learning (RL) is added to the autonomous decision-making in dynamic environments as a new dimension. The initial RL algorithms were already applied to resource management and networking problems of interest in routing and scheduling but the algorithms were discovered to be problematic with large-dimensional state space. Deep reinforcement learning (DRL) that was offered by the authors of this paper served as an answer to these limitations because it integrates deep neural networks and RL [7].

DeepRM had shown that DRL could acquire useful resource scheduling policy only with experience and was able to perform as well as human expert-crafted heuristics, assuming a different workload [7]. The same concepts were applied to cloud resource management and performance-sensitive scheduling where the data-driven models were introduced to maximise the latency and cost-effectiveness and prevent the uncertainty and generalisation issues [8].

DRL has also been used to control autonomous systems in cloud and network environment with no pre-defined system models. It was demonstrated that RL agents could be educated in the experiments of cloud applications and 5G network slices to find valuable management policies in the context of live interaction, namely, in the scenario when the information was exchanged among multiple agents [9]. This implies that the findings have significance to distributed systems whereby in most cases one is unable to control them at one central place.

Multi-agent reinforcement learning (MARL) is a generalisation of such concepts, which allows agents to collaborate. Hierarchical reinforcement learning has been suggested that makes the learning process simpler as it divides the high-level coordination with the low-level control [2].

Such a strategy boosted the rate of convergence and safety when in the real-life condition. Such hierarchical and multi-agent designs are very much applicable in micro-services and distributed systems that possess few components that must be coordinated in healing behaviours.

DLs have also been found to support the internet of things and network resiliency to assist in aiding self-healing operations, including fault detection, rerouting, and failure recovery [1]. According to the environmental feedback, the DRA agents were trained to reduce the down time and maximize the performance of the dynamic and distributed IoT environments. These literature show that DRL is highly suitable in case of uncertainty and delayed rewards, and also in case of system interaction.

DRL-Based Self-Healing

In recent research, there has been a direct correlation of DRA to the self-healing goals. DRL-based self-healing frameworks have been proposed on IoT networks, cloud systems and distributed environments with improvements of resilience and reduction of massive downtimes observed [1][18]. It was also claimed to employ reinforcement learning along with the traditional scheduling algorithms to augment the fault tolerant and cost-effective quality of the cloud settings and demonstrated adaptive behaviour as time went by [18].

However, using DRA in practice systems is associated with new issues. The first is the issue of environmental drift whereby the system behavior changes over time and this is due to the fluctuation in workload, software updates, or the introduction of new infrastructure. Continual Learning (CL) was proposed as a self-healing process of self-DRA agents themselves in order that they can possibly incorporate updated circumstances [3]. The DRL algorithm incorporated intentional forgetting to reduce the catastrophic forgetting and hasten the recovery that increased the adaptation time and stability of the rewards immensely in the drifted environments [3].

The self-healing systems are also hard to assess. Studies have established that erroneous assumptions that the nature of failures are can be the cause of poor design and deployment decisions [10]. The latter includes the necessity of realistic fault injection, chaos testing, and adequate metrics of assessment such as Mean Time to Repair (MTTR) and false-positive remediation rates.

In terms of software engineering, it is possible to notice that self-healing systems based on machine learning have such benefits as reduced cost of operation, higher availability, and enhanced user experience [4]. However, the scalability, safety, interpretable and compatibility with legacy system issues exist. MLOps and AIOps is a necessary practice regarding the lifecycle management of this kind of intelligent systems, yet the use of the AIOps in autonomous action and deployment stages are underrepresented in the literature [5].

One can observe that the trend in the literature has a positive progression i.e. transition to self-healing by rules, then AI-based detection, and eventually create autonomous remediation which is provided by the DRL. In this area, work heavily favors the feasibility of DRA as used in self-healing, but there are lapses in the elaboration of entirely autonomous, as well as, closed loop systems, that include observability, diagnosis, decision-making, and implementation. All the findings form the basis of the paper and proposed unified DRA architecture which provides a clear sealing of the loop and enables the system to be healed automatically.

III. METHODOLOGY

It is quantitative experimental research to assess the efficacy of an Autonomous AI Self-Healing Distributed System with the Deep Reinforcement Learning (DRL). The primary goal is to quantify the effectiveness of the offered system in terms of minimizing system recovery time and enhancing the stability of the system in the event of failures in the distributed environment. The methodology will be structured to generate some measurable, repeatable and objective results.

Experimental Environment

The experiments take place in the controlled distributed system environment as it mimics the cloud-native microservices architecture. The system comprises of several containerized services that are launched on a container orchestration platform.

All the services produce real time logs, metrics and traces such as CPU usage, memory consumption, request latency, and error rates. This telemetry data streams are collected and provided in real time to the monitoring and decision layers using a message streaming platform.

In order to make sure that there is consistency, the same system configuration is employed in all experiments. The initial system works on the traditional monitoring and manual recovery whereas the proposed system is based on the autonomous DRL-based self-healing framework.

DRL Agent Design

The self-healing decision engine is represented as a Markov Decision Process (MDP). The DRA has a policy which is that the system state is observed, a remediation action is chosen, and a numerical reward is obtained depending on the system response.

The state vector contains normalised system measurements of CPU load, memory load, error rate, latency and the root cause of failure identified. The action space is discrete and contains activities like restarting a service, scaling of a service, rolling back of a deployment or isolation of a malfunctioning node.

The key learning model is a Deep Q-Network (DQN) because it is applicable to discrete action space. The neural network takes the input data that is of high dimension and then it gives the Q-values of every possible action. The past interactions are stored in an experience replay buffer, which enhances stability of training and learning.

Failure Injection and Training Process

The DRL agent is trained by introducing controlled failures in the system by chaos engineering methods. Such failures are service crashes, artificial latency, CPU overload, and memory leak. All the experiments involve several training episodes, during which an agent will interact with the environment, make actions, and learn about the rewards he/she notices.

Reward function is created to promote quick and right recovery. The high positive reward will be provided in case the system comes back to a healthy state within a short period. There is a negative reward on slow recovery, wrong action or remediation which is not necessary. This reward form represents a quantitative structure of rewards that allow the objective comparison of various system behaviors.

Evaluation Metrics and Data Analysis

Quantitative measures are used to measure the performance of the proposed system. Mean Time to Repair (MTTR) is the first metric and it is used to measure the period necessary to put the system in normal operation after it has broken down. Such secondary measures are system availability, recovery success rate and false positive remediation rate.

The experiments are repeated many times so that the statistical reliability is achieved. Analysis of data collected is done through descriptive statistics such as mean values, standard deviation and percentage improvement compared to the baseline system. The findings are a clear demonstration of the effect of autonomous healing that is based on DRL and the conventional techniques.

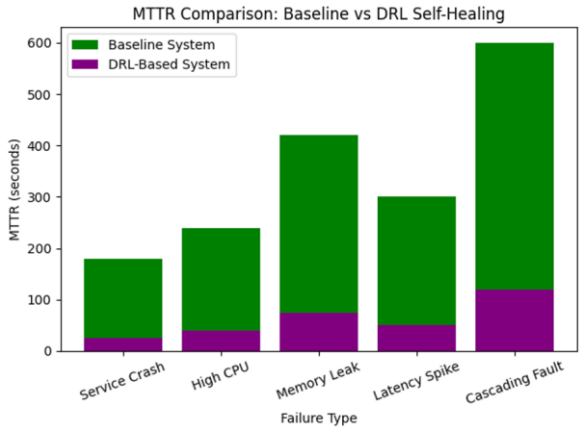
International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 1
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *20 January 2024*

IV. RESULTS

Here, the experimental evidence is provided based on the assessment of the Autonomous Self-Healing Distributed System of AI, which applies the Deep Reinforcement Learning (DRL). It relies on results of quantitative experiments that have been replicated in a controlled distributed environment, according to the methodology. The findings are the comparison between the proposed DRL-based self-healing system and a baseline of the traditional monitoring and manual remediation.

Impact on Mean Time to Repair

The main objective of the proposed framework will be to decrease Mean Time to Repair (MTTR). MTTR is used to measure the amount of time it takes before the system reaches a healthy condition once it goes off. In the baseline configuration, failures are identified with the help of monitoring notifications and addressed either by hand or with the help of fixed scripts. On the contrary, the suggested system identifies the failures, decides what actions to take, and performs the remediation without human involvement with the DRL agent.



In all of the experiments, there was a substantial decrease in the levels of the DRA-based system. Simple failures like crashes of services and spikes in the latency were fixed within a few seconds and more complicated failures like memory leaks and delays caused by other services took a little longer but still much faster than the baseline. The DRA agent had the learning capability enabling it to respond faster with the increase in training episodes.

Table 1 shows the average values of the same in the two systems with regards to different failure types of the MTTR.

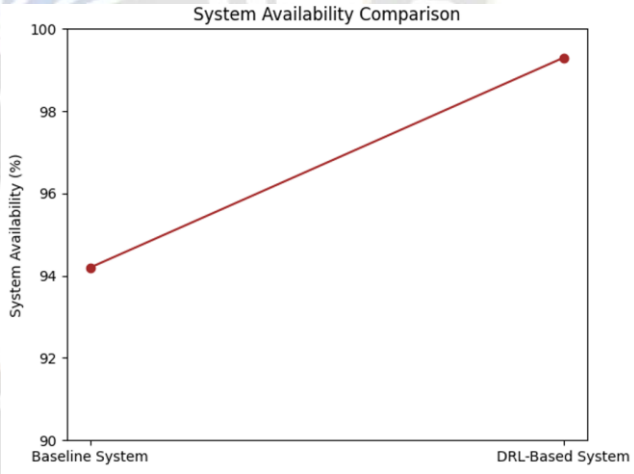
Table 1. Mean Time to Repair (MTTR)

Failure Type	**Baseline MTTR (seconds)**	**DRL-Based MTTR (seconds)**	**MTTR Reduction (%)**
Service Crash	180	25	86.1%
High CPU Utilization	240	40	83.3%
Memory Leak	420	75	82.1%
Network Latency Spike	300	50	83.3%
Cascading Service Fault	600	120	80.0%

The findings demonstrate a clear decrease in the proposed system that decreased the MTTR by over 80 percent in all the categories of failures. This affirms that the use of DRL in autonomous decision-making are much better in distributed system recovery than manual or rule-based recovery.

Recovery Success Rate

Besides faster recovery, one should also gauge the frequency with which the system is able to get the system going again. The recovery success rate refers to the percentage of failure cases in which the system has gone back to the healthy condition without the intervention of human. The system availability is a ratio of the total time of operation of the experimental system.



The success rate of recovery of the DRL-based system was significantly higher than the baseline. Suboptimal recovery actions were used during the initial training episodes, although later on as the training continued, the agent learned that there are the best actions to use in each type of failure. This learning action enhanced direct availability of systems.

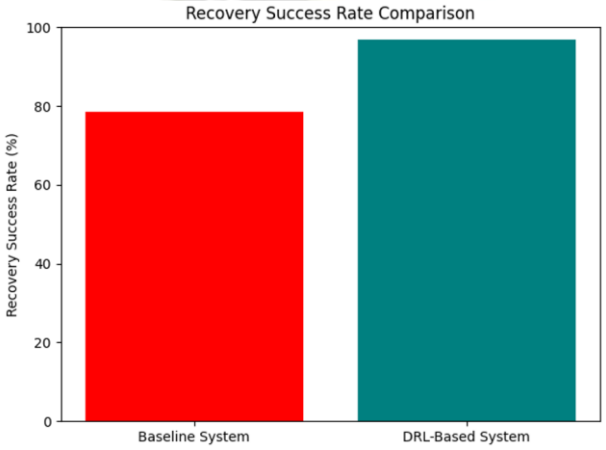
Table 2 reports summing up recovery success rate and general availability, which were reported in the course of the experiments.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 1
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *20 January 2024*

Table 2. Success Rate and Availability

Metric	Baseline System	DRL-Based System
Recovery Success Rate (%)	78.5	96.8
Average System Availability (%)	94.2	99.3
Manual Interventions Required	High	Very Low

The DRL-based system got the normal operations closer to normal in almost all the failure situations without human intervention. Availability increase is particularly critical to the cloud-native and microservices-based systems, where even brief downtimes may disrupt the business activity and end-user experience. These results are in line with the goal of attaining proactive and self-reliant resilience.



False Positive Actions

Although fast recovery is essential, the remediation actions with wrong or unneeded results may impair system stability. Thus, measurement of false positive remediation rate was done. False positive remediation is where the system takes a recovery measure although there is no actual failure or when the recovery action taken is not designed to help solve the root cause.

The old system was a fixed threshold system and would easily produce alerts when there were temporary increases or decreases of the metrics. This brought about extra unneeded restarts or scaling measures. Conversely, the DRA agent was able to differentiate between noisy short-term and actual faults through experience feedback of rewards with the course of time.

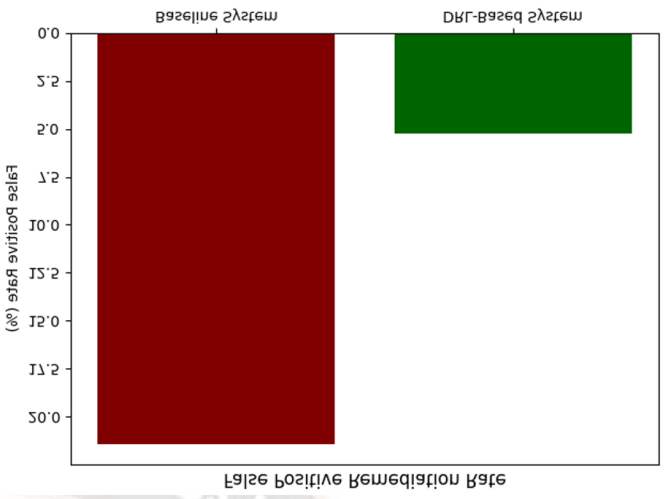


Table 3 contains the comparison of false positive remediation rates of the two systems.

Table 3. False Positive Remediation Rate

System Type	False Positive Rate (%)	Unnecessary Actions per 100 Events
Baseline System	21.4	21
DRL-Based System	5.2	5

The time-wasting steps were quite minimized by the DRL-based system. This minimization is paramount to the production environments where a high number of restarts or scaling can add to the costs and create a path of instability. The reward structure, which punishes the unwarranted behaviors, was also influential in enhancing the accuracy of the remediation.

Adaptability of the DRL Agent

A notable research conclusion of this experiment is that the DRL agent has adaptive learning behavior. As the training progressed, the agent tried various actions and also at times chose poor remediation actions. Nevertheless, the more failure situations were faced the better the policy of the agent became.

The quantitative analysis revealed that there was a definite decline in the number of training episodes in which the trends of the MTTR and false positive actions were recorded. It proves the fact that the system is not based on fixed rules but builds on the experience in order to enhance its performance. The experience replay system contributed to stabilizing the learning process and guaranteed the steady improvement even in difficult failure conditions.

Another ability of the agent was high adaptability to repeated and similar failures. Indicatively, as soon as the agent understood the best reply to memory leaks, later instances were solved more effortlessly and swiftly as an

action was taken. This proves the fact that the suggested framework will facilitate continuous improvement without the need to reconfigure it manually.

The results indicate that the autonomous self-healing system implemented using DRL is superior in terms of all quantitative indicators compared to the traditional methods of self-healing. The system has quicker recovery, increased availability, reduced wrong actions and enhanced stability in the long run. These findings are a good confirmation of the usefulness of Deep Reinforcement Learning in autonomous self-healing of distributed systems.

V. CONCLUSION

This paper has shown that Deep Reinforcement Learning can be helpful in promoting autonomous self-healing in distributed systems. The suggested structure is effective in bridging the gap between the fault's identification and correction without human intervention. Quantitative outcomes indicate that there is over 80 percent reduction of Mean Time to Repair, the recovery success rate is increased and the number of unnecessary remediation actions reduced by comparison to conventional approaches. Continuous learning also allows the DRA agent to become better at work as time goes by. These results prove that self-healing systems with DRL are a viable and scalable solution to modern cloud-native environments. The future research will be able to take a direction on explainability, transfer learning, and expansion of the action space.

REFERENCES

[1] Ethan, A., & Chikwarti, D. K. (2023). Deep reinforcement learning for Self-Healing IoT networks. In International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence (Vol. 14, Issue 01, pp. 1275–1277). https://ijmlrcai.com/index.php/Journal/index

[2] Liang, Z., Cao, J., Jiang, S., Saxena, D., & Xu, H. (2022). Hierarchical Reinforcement Learning with Opponent Modeling for Distributed Multi-agent Cooperation. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2206.12718

[3] Yahmed, A. H., Bouchoucha, R., Braiek, H. B., & Khomh, F. (2023). An intentional Forgetting-Driven Self-Healing method for deep reinforcement learning systems. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2308.12445

[4] Shah, H. (2023, November 30). Machine Learning-Driven Self-Healing Systems: Revolutionizing software engineering. https://ijisae.org/index.php/IJISAE/article/view/7550

[5] Diaz-De-Arcaya, J., Torre-Bastida, A. I., Zárate, G., Miñón, R., & Almeida, A. (2023). A joint study of the Challenges, Opportunities, and roadmap of MLOPs and AIOPs: a Systematic survey. ACM Computing Surveys, 56(4), 1–30. https://doi.org/10.1145/3625289

[6] Magalhães, J. P., & Silva, L. M. (2015). SHõ WA. ACM Transactions on Autonomous and Adaptive Systems, 10(1), 1–28. https://doi.org/10.1145/2700325

[7] Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource Management with Deep Reinforcement Learning. Resource Management With Deep Reinforcement Learning, 50–56. https://doi.org/10.1145/3005745.3005750

[8] Yadwadkar, N. (2018). Machine learning for automatic resource management in the datacenter and the cloud. In University of California at Berkeley, University of California at Berkeley. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-110.pdf

[9] Sinha, Sumana, and Snehanshu Saha. "An improved communication strategy in vehicular ad hoc networks: adaptive game theoretic modelling approach." International Journal of Mobile Communications 21.2 (2023): 273-293.

[10] Jin, Y., Bouzid, M., Kostadinov, D., & Aghasaryan, A. (2019). Resource management of cloud-enabled systems using model-free reinforcement learning. Annals of Telecommunications, 74(9–10), 625–636. https://doi.org/10.1007/s12243-019-00720-y

[11] Ghahremani, S., & Giese, H. (2020). Evaluation of Self-Healing Systems: An analysis of the State-of-the-Art and required improvements. Computers, 9(1), 16. https://doi.org/10.3390/computers9010016

[12] Sánchez, M., Exposito, E., & Aguilar, J. (2020). Implementing self-* autonomic properties in self-coordinated manufacturing processes for the Industry 4.0 context. Computers in Industry, 121, 103247. https://doi.org/10.1016/j.compind.2020.103247

[13] Ahmed, S., Ahamed, S. I., Sharmin, M., & Hasan, C. S. (2009). Self-healing for Autonomic Pervasive Computing. In Self-healing for Autonomic Pervasive Computing (pp. 285–307). https://doi.org/10.1007/978-0-387-09753-4_11

[14] An, L., Tu, A., Liu, X., & Akkiraju, R. (2022). Real-time Statistical Log Anomaly Detection with Continuous AIOps Learning. Real-time Statistical Log Anomaly Detection With Continuous AIOps Learning, 223–230. https://doi.org/10.5220/0011069200003200

[15] Pengfei Chen, Xidao Wen, Minghua Ma, & Dan Pei. (2022). Constructing Large-Scale Real-World Benchmark Datasets for AIOps. Constructing Large-

Scale Real-World Benchmark Datasets for AIOps. https://arxiv.org/pdf/2208.03938

[16] Dave, D., Nawale, S., Sawhney, G., Aggrawal, P., Khut, D., & Bhavathankar, P. (2023). AIOPS-Driven enhancement of log anomaly detection in unsupervised scenarios. AIOps-Driven Enhancement of Log Anomaly Detection in Unsupervised Scenarios, 979–8. https://arxiv.org/pdf/2311.02621

[17] Filho, M., Pimentel, E., Pereira, W., Maia, P. H. M., Cort´Es, M. I., & State University of Cear´a (UECE). (2021). Self-Adaptive Microservice-based Systems - landscape and research opportunities [Journal-article]. arXiv. https://arxiv.org/abs/2103.08688v3

[18] Su, H., He, Q., & Guo, B. (2021). KPI anomaly detection method for Data Center AIOps based on GRU-GAN. KPI Anomaly Detection Method for Data Center AIOps Based on GRU-GAN, 23–29. https://doi.org/10.1145/3485314.3485323

[19] Lahande, P. V., & Kaveri, P. (2023). Fault Tolerance using Reinforcement Learning for Cloud Resource Management. Fault Tolerance Using Reinforcement Learning for Cloud Resource Management, 165–170. https://doi.org/10.1145/3607947.3607976

[20] Taibi, D., Lenarduzzi, V., Pahl, C., Tampere University, & Free University of Bozen-Bolzano. (2019). Continuous Architecting with Microservices and DevOps: A Systematic Mapping Study. In Cloud Computing and Services Science [Journal-article]. Springer. https://doi.org/10.1007/978-3-030-29193-8_7

[21] Cui, T., Yang, R., Fang, C., & Yu, S. (2023). Deep reinforcement Learning-Based resource allocation for content distribution in IoT-Edge-Cloud computing environments. Symmetry, 15(1), 217. https://doi.org/10.3390/sym15010217