A fault injection platform for learning AIOps models

DD2482
Automated Software Testing and DevOps

Essay

Afruz Bakhshiyeva
Luis Jira

We certify that generative AI, incl. ChatGPT, has not been used to write this essay.
Using generative AI without permission is considered academic misconduct.

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

1 Introduction

Web-service outages with an estimated cost between $1000,000 and $1 million have jumped from 28% to 47% between 2019 and 2021, according to the Uptime Institute's 2022 Outage Analysis Report [2], and outages with root causes in operations top the list. This indicates a dire need for better failure prevention and response in IT operations world-wide. In this essay, we will explain how learning-data for Machine-Learning (ML) models used in IT operations can be automatically generated using a newly developed fault-injection platform for applications.

Large-scale applications are omnipresent in today's life. From social media networks to video conferencing, these web-based applications are tightly integrated into our everyday lives. In recent years, the software engineering practices for them have gravitated towards microservice architectures. This move aims to reduce the complexity of each service and foster quicker feature development.

While microservice architectures successfully reduce development complexity, they drive up operational complexity. As an example, an application failure can now have a long dependency chain spanning many microservices. These dependency chains are complex and time-consuming to understand. Thus, it takes longer to find and fix the root cause of a failure.

A modern micro-service-based application can have hundreds of services running on hundreds of machines. Each of these services could receive multiple updates per day. Manually updating these services would overload an IT operations team, which is why automation tools have been introduced in the past. However, as mentioned before, the most common causes for outages lie in operations. Just performing these updates would occupy a team of operations engineers. To lessen this load, automated deployment tools have been implemented in the past. Additionally, the DevOps role and culture has penetrated the industry. Here, the open sharing of experiences and tight coupling between software developers and operations engineers leads to greater shared knowledge and better enterprise resiliency [5].

2 AIOps

The official definition of Artificial Intelligence for IT Operations (AIOps) by Gartner, who coined the term in 2016, is:

> AIOps platforms utilize big data, modern machine learning and other advanced analytics technologies to directly and indirectly enhance IT operations (monitoring, automation, and service desk) functions with proactive, personal and dynamic insight. AIOps platforms enable the concurrent use of multiple data sources, data collection methods, analytical (real-time and deep) technologies, and

presentation technologies. [9]

As mentioned in the blog post, starting in 2014, IT operations leaders noticed the discrepancy between the complex systems being built by IT operations teams and the rudimentary tools they were using themselves. At the same time, the pressure on IT operations teams to allow for faster, more agile software deployment while providing higher levels of reliability and efficiency grew.

Additionally, the industry was gravitating towards microservice architectures. As mentioned before, these reduce development complexity while increasing operational complexity.

Combined, these changes and challenges necessitated a change in the operation of IT operations.

Here, the field of AIOps comes in. By investigating the use of Artificial Intelligence (AI) for the management and improvement of IT services [10] it aims to simplify and streamline processes in IT operations. Possible use cases range personal assistants specialized for operations engineers through failure forecasts and failure recovery recommendations.

The data needed for such recommendations and forecasts is already processed in IT operations environments today. AIOps uses existing data sources, like application log events, traditional IT monitoring and application and network performance anomalies. This data is then processed by a Machine-Learning model. This model can distinguish between common events, that need little or no attention, and uncommon events that need immediate attention from the IT operations team.

A challenge for AIOps is obtaining the necessary vast amount of training data required for these models. Especially, since event logs and performance anomalies are often highly dependent on the application, they can not always be generalized. We will discuss one solution to this problem in section 3.

3 A fault injection platform for learning AIOps models

3.1 Problems

Modern IT environments are faced with a perplex combination of cost upsurge of persistent web-service outages, operational big data availability and increasingly complex systems. [2] [4]

With ever-evolving intricacies and abstractions of erratic components of modern day IT environments, jobs of Ops engineers are more complicated than ever.

The overwhelming size of observable data and dynamic nature of IT environments further complicates assessment of application resiliency. Consequently, this leads to the deceleration of production issue resolution. Agile development practices

perform frequent software releases, putting the preceding methodology of Ops engineers out-of-date, and thus making their job more difficult. [8]

3.2 State-of-the-art

In an attempt to combat or at least ease some of these issues, several efforts have been made in recent years.

With regard to web-service outages, bug localization techniques have been improved, however, ways to enhance operational issue resolution for critical production uses should to be investigated more. New microservice architectures simplified software development, but increased complexity for operations engineers.

On the other hand, simply incorporating AI into Ops tools would be dependent on the supply of data retrospectively, where relevancy, correctness, and completeness of the data can't be guaranteed.

3.3 IBM's solution

Last year, IBM's research team went on a mission to solve these issues. The research team published their work on an automatic fault injection platform, enabling and optimizing data generation for AI models and the simulation of user-provided faults. The platform supports many operating systems, including virtual. Fault injection is integrated together with data collection, which satisfies both demands of developers and researchers. This integration aids automated learning, which in turn optimizes operational tasks in production. Their design also covers most common types of fault injections. User-friendliness, simulation of complicated fault situations and data generation optimization are some of the advantages of this platform.

Multiple AI functionalities are also one of the strengths of this platform. The performance of one such functionality will be explored more in the Subsection 3.5.

In the upcoming three subsections, we are going to take a closer look at building blocks of this platform and their functions.

3.4 Microservices of the platform

The platform is built out of three microservices, programmed in Python, executed on a Kubernetes® cluster (see Figure 3.1) as described in a more detail below.

3.4.1 Fault-Injection Microservice

The fault-injection microservice enables the modification of configurations for authentication by communicating to API endpoints for different operations. It supports fault-injection in both remote clusters, such as Kubernetes® and Red

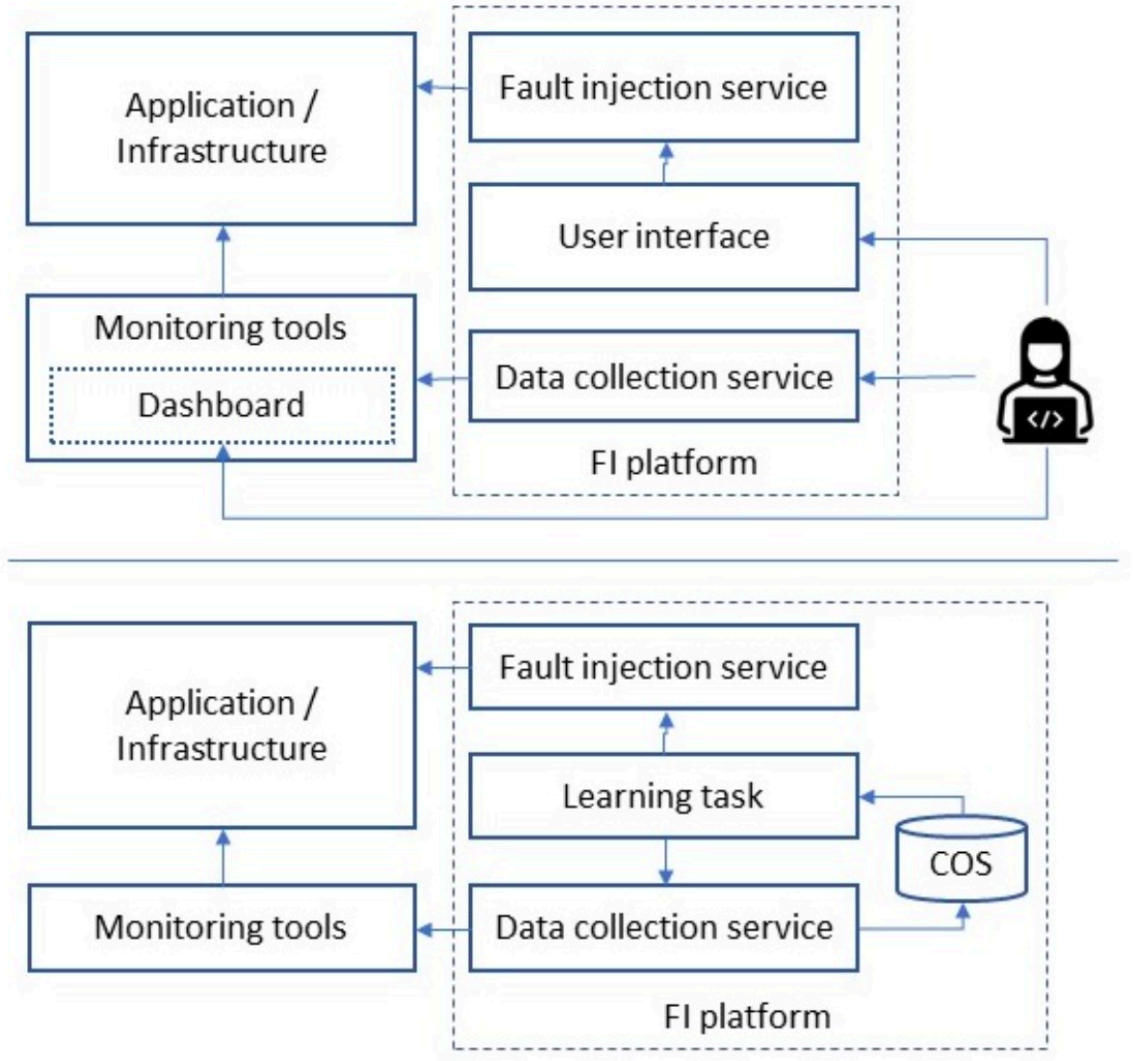

Figure 3.1: Fault injection platform infrastructure with manual exploration (above) and automated learning (below)

Hat OpenShift®, and virtually [13] [12]. Fault injection or removal, listing of current, historical, and all supported fault types, and their specifications, as well as information on target resource types and required parameters are all operations that the microservice supports.

The microservice calls previously implemented logic and/or third-party logic for fault injection, like chaos toolkit [14]. Thus, the end-user access to specific faults is simplified through the microservice's API. Diverse full-coverage fault integration is made possible due to the vast fault type variety and volume in the existing fault injection ecosystem, because external libraries can be easily integrated into the fault-injection microservice.

Saturation, Kubernetes® resource type, network and HTTP errors in Istio®-enabled [1] application errors are all types of faults covered by the present

implementation.

JSON dictionaries document the faults list passed to the API. Type, resource, and other relevant information for each fault is provided there.

Basic fault simulation is enabled through the selection of duration and delay in order to test more sophisticated scenarios. For instance, a memory consumption list with increasing values can be used to simulate a memory leak scenario. Common fault types are pre-configured and can be injected freely and easily.

3.4.2 User Interface Microservice

The user interface microservice aims to provide an end-user friendly interface with the platform (see Figure 3.1). The user interface gathers necessary information such as what pods and/or services exist in a provided context from Kubernetes based on the end-user provided credentials. After fault selection has been made by the user, the user interface automatically provides the user with the list of suitable targets for easy selection.

3.4.3 Data Collection Microservice

Lastly, the data collection microservice is necessary to automatically collect the generated fault data. Logs and alerts from monitoring systems, like Instana, Mezmo, are collected for further processing by the AIOps learning algorithms [3] [11]. The default API calls conceal the structure and configurations of external calls from end-users, enabling manual and automated gathering and incorporation of multi-sourced operational data.

For complete automation of all three components, the learning algorithm collects the gathered data from shared cloud storage, which is where the data collection microservice saves its output (see Figure 3.1).

3.5 Experience

In a microservice-based application, services triggers sequences of calls between each other. This communication is useful information for operations engineers when troubleshooting a failure scenario. For example, an error in one service can propagate along this graph and potentially affect many services along the way. In this case, the communication graph would help to quickly find the root cause of the error.

While
Kubernetes® already provides the ability to trace inter-service communication. But the majority of production applications are not managed by Kubernetes. To detect the failure propagation path in a situation without Kubernetes, mostly only the logs will be available. The authors of the paper tested their platform in such a scenario to detect the causal relations between errors and then be able to learn the communication graph and error propagation model. For this they used

two publicly available benchmark microservice applications, DayTrader [7] and TrainTicket [6], with 5 and 33 services respectively.

To create the relevant learning data, an availability fault is automatically and sequentially injected into each microservice in a staging environment. The resulting log data is converted into time series data and causal learning is applied on it.

For the smaller benchmark, DayTrader, their model could accurately predict connected edges in the communication graph. Whereas, for the larger benchmark, TrainTicket, only 54% of the edges were predicted correctly. This translates to 73% of connected edges being detected as such.

The learned error propagation model was then tested as part of a fault localization algorithm. Here, the authors saw an that all injected faults were correctly localized for DayTrader and 94% for TrainTicket.

4 Reflection

We see AIOps as a necessary solution to the increasing complexity in IT operations. Especially, since the requirements for reliability are ever-increasing. The IBM research paper introduces a valuable enabler for AIOps. Due to the highly individual nature of application monitoring metrics, AIOps learning data can not be generalized. Here, the proposed fault-injection platform provides the necessary technology to automatically incorporate learning data from the staging environment of a deployment pipeline into the AIOps workflow. Only through this automation can AIOps be used to its full potential. Otherwise, the Machine-Learning models would likely be generated with outdated monitoring data and have no idea how to react to a failure of a newer application version.

However, the complexity and resource consumption of the approach was not discussed. In our opinion, a staging environment with the ability to simulate service failures under realistic load is a big resource commitment, that only a few companies will be willing to make. Additionally, it is not discussed how time-consuming the fault-injection and learning processes are. We believe that the added delay in staging environments will turn prospective users away, as it fundamentally disagrees with the fast and seamless deployment schema it tries to support.

On top of these shortcomings, the reported accuracies for benchmark applications showed quickly diminishing performance from 5 to 33 services. Neither of these benchmark applications is a truly large and complex microservice architecture. This leads us to question how useful the platform is. Since it only provides useful output for architectures that are simple enough to not need such infrastructure to begin with.

5 Conclusion

In this essay we saw the IT operations evolution towards complex-to-operate microservice architectures while requiring higher reliability, efficiency and faster deployment. While these changes seem to clash with each other, we discussed how AIOps helps to combat the incurred challenges.

By leveraging Artificial Intelligence to manage and improve IT operations, AIOps can provide targeted notifications to operations engineers. This increases efficiency in IT operations and, in turn, reduce failure recovery time and cost.

We then discussed a solution to the problem of learning-data generation for AIOps. Here, the bad generalization of application monitoring metrics is circumnavigated by automatically collecting data from a fault-injection platform during the staging process. This ensures, fresh learning-data specific to the current application is used to inform the Machine-Learning AIOps model about failure behavior and root causes.

References

[1] *A leading service mesh*. URL: https://istio.io/.

[2] Ascierto, Rhonda and Brown, Chris. *Critical update: Uptime Institute 2022 outage report*. June 2022. URL: https://uptimeinstitute.com/webinars/webinar-critical-update-uptime-institute-2022-outage-report.

[3] *Automated Observability: Automation and AI to prevent and remediate application issues fast*. Feb. 2023. URL: https://www.instana.com/.

[4] Barnaghi, Payam et al. "IoT Analytics: Collect, Process, Analyze, and Present Massive Amounts of Operational Data–Research and Innovation Challenges". In: *Building the Hyperconnected Society-Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*. River Publishers, 2022, pp. 221–260.

[5] Davis, Jennifer and Daniels, Ryn. *Effective DevOps: building a culture of collaboration, affinity, and tooling at scale.* " O'Reilly Media, Inc.", 2016.

[6] Fudanselab. *train-ticket*. [Online; accessed 11. Apr. 2023]. Apr. 2023. URL: https://github.com/FudanSELab/train-ticket.

[7] IBM. *DayTrader*. June 2022. URL: https://github.ibm.com/ocp-r2-demo/.

[8] Inukollu, Venkata Narasimha, Arsi, Sailaja, and Ravuri, Srinivasa Rao. "Security issues associated with big data in cloud computing". In: *International Journal of Network Security & Its Applications* 6.3 (2014), p. 45.

[9] Lerner, Andrew. *AIOps Platforms*. [Online; accessed 11. Apr. 2023]. Aug. 2017. URL: https://blogs.gartner.com/andrew-lerner/2017/08/09/aiops-platforms.

[10] Notaro, Paolo, Cardoso, Jorge, and Gerndt, Michael. "A systematic mapping study in AIOps". In: *Service-Oriented Computing–ICSOC 2020 Workshops: AIOps, CFTIC, STRAPS, AI-PA, AI-IOTS, and Satellite Events, Dubai, United Arab Emirates, December 14–17, 2020, Proceedings*. Springer. 2021, pp. 110–123.

[11] *Observability Data Pipeline & Log Analysis Solutions*. [Online; accessed 11. Apr. 2023]. 2022. URL: https://www.mezmo.com/.

[12] *Production-grade container orchestration*. URL: https://kubernetes.io/.

[13] *Red Hat OpenShift Enterprise Kubernetes Container Platform*. [Online; accessed 11. Apr. 2023]. URL: https://www.redhat.com/en/technologies/cloud-computing/openshift.

[14] Team, The Chaos Toolkit. *The Chaos Engineering Toolkit for developers*. [Online; accessed 11. Apr. 2023]. 2017. URL: https://chaostoolkit.org/.