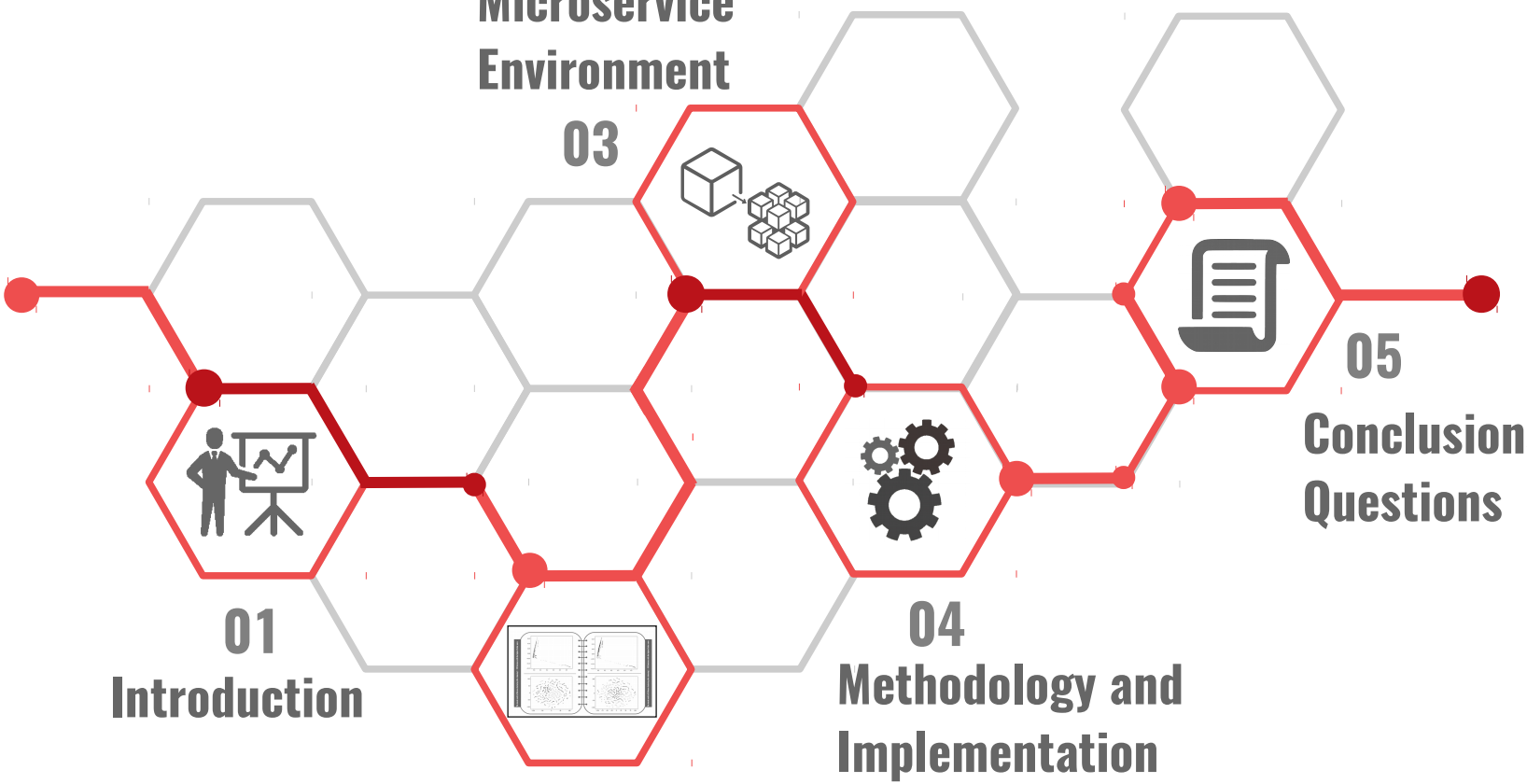
Anomaly detection in microservice systems using tracing data and Machine Learning

Iman Kohyarnejadfard
Prof. Daniel Aloise and Prof. Michel Dagenais

POLYTECHNIQUE
MONTREAL

May 2020

Agenda



01
Introduction

02
Previously on
Anomaly Detection

03
Microservice
Environment

04
Methodology and
Implementation

05
Conclusions &
Questions

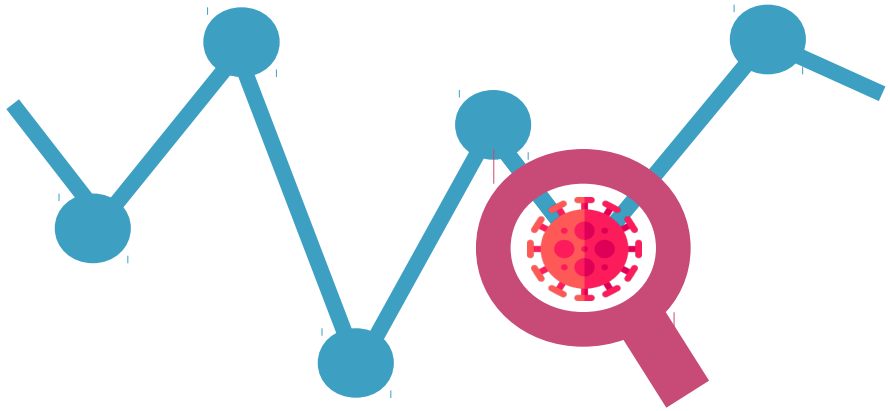
Anomaly Detection

Anomaly is something different which deviates from the common rule.

Anomalies are patterns in data that do not conform to a well defined notion of normal behavior.

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior.

Many anomaly detection techniques have been developed for various application domains.



Performance Anomaly

The most significant obstacles to the system to perform confidently and predictably

Sources

varying application load, application bugs, updates, and hardware failure

Workload

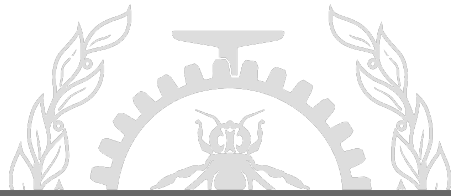
the application imposes continuous and more than expected average workload intensity to the system.

Faults in system resources and Components

affect application performance at a high cost

Software bugs, operator errors, hardware faults, and security violations

system failures

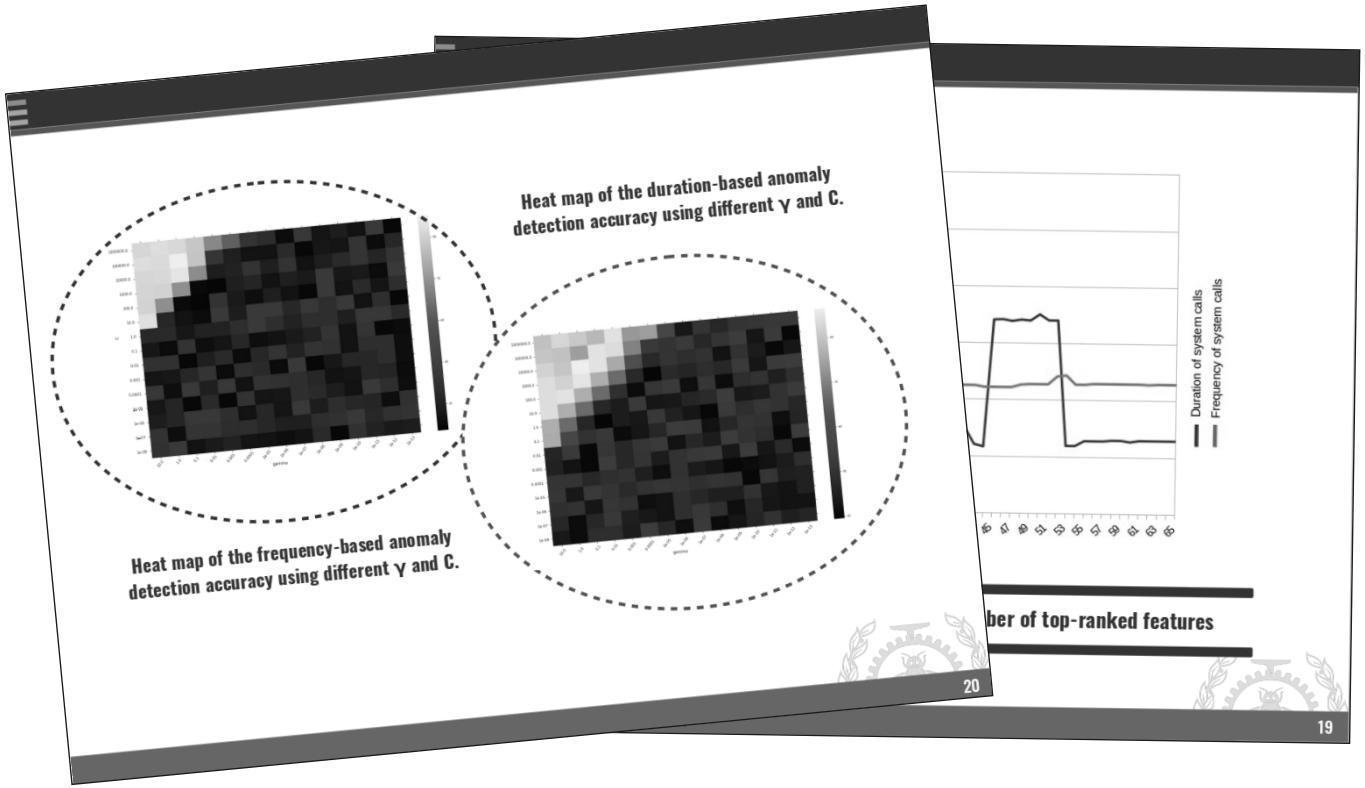


Previously on Anomaly Detection

May 2019

01 SUPERVISED

- LTTng was used to monitor the processes running on a system and extract the streams of system calls.
- The system calls streams are split into short sequences using a sliding window strategy.
- Finally, a multi-class support vector machine approach is applied to evaluate the performance of the system and detect the anomalous sequences.

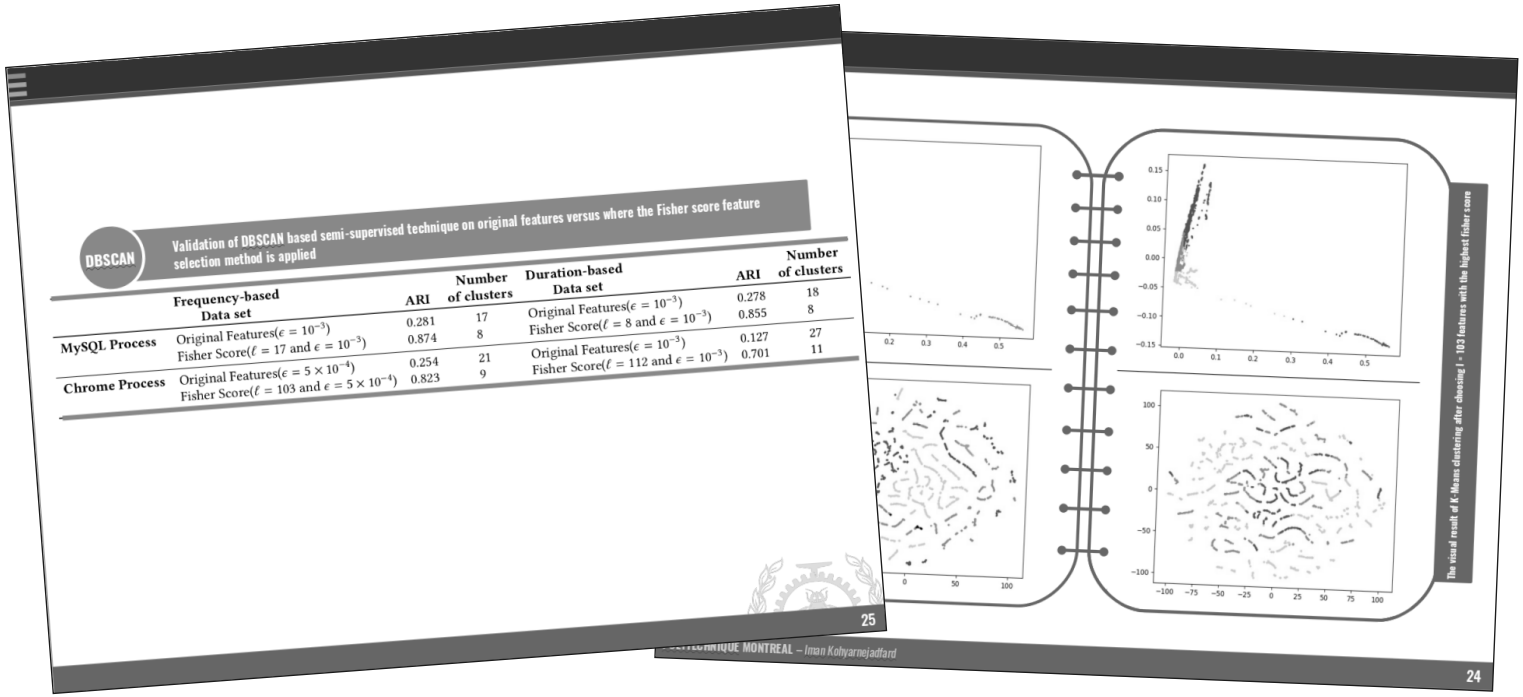


Previously on Anomaly Detection

01 Dec. 2019 02

SEMI SUPERVISED

- This method benefits from both supervised and unsupervised learning techniques to distinguish between normal and anomalous behavior.
- It removes the need of providing a huge labeled dataset.
- We select the most discriminative features from a small set of labelled data by means of the iterative Fisher Score feature selection method.
- In the sequel, the DBSCAN clustering algorithm is applied to group the remaining data into the sought number of classes.



DBSCAN Validation of DBSCAN based semi-supervised technique on original features versus where the Fisher score feature selection method is applied

Frequency-based Data set		ARI	Number of clusters	Duration-based Data set		ARI	Number of clusters
MySQL Process	Original Features($\epsilon = 10^{-3}$)	0.281	17	Original Features($\epsilon = 10^{-3}$)		0.278	18
	Fisher Score($\ell = 17$ and $\epsilon = 10^{-3}$)	0.874	8	Fisher Score($\ell = 8$ and $\epsilon = 10^{-3}$)		0.855	8
Chrome Process	Original Features($\epsilon = 5 \times 10^{-4}$)	0.254	21	Original Features($\epsilon = 10^{-3}$)		0.127	27
	Fisher Score($\ell = 103$ and $\epsilon = 5 \times 10^{-4}$)	0.823	9	Fisher Score($\ell = 112$ and $\epsilon = 10^{-3}$)		0.701	11

The visual result of K-Means clustering after choosing 1 - 103 features with the highest fisher score

25

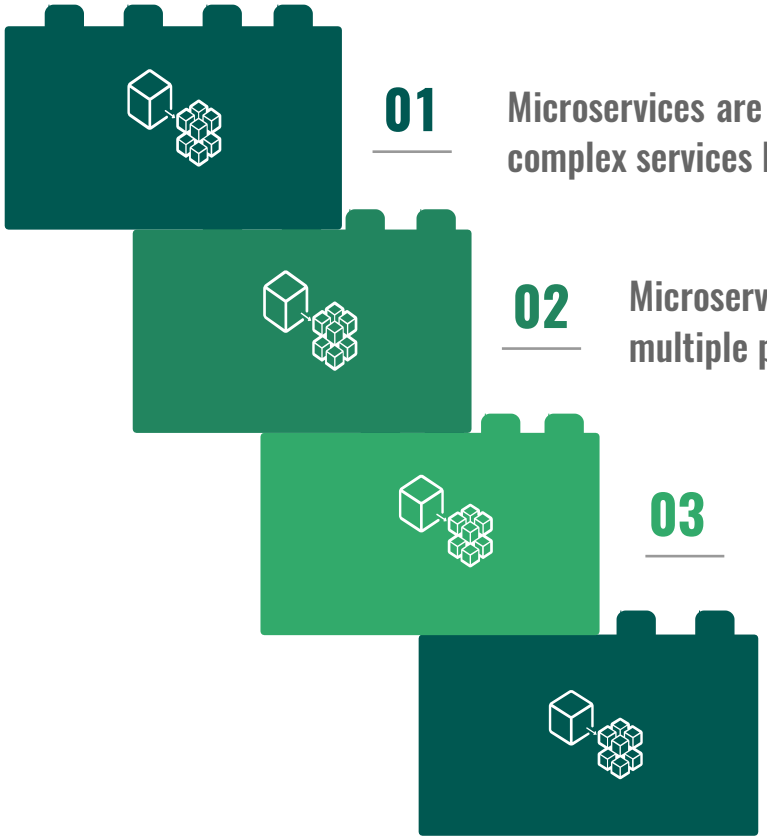
POLYTECHNIQUE MONTREAL – Iman Kohyarnejadfard

24

Previously on Anomaly Detection

The concept of DevOps and agile approaches like microservice architectures and Continuous Integration becomes extremely popular since the need for flexible and scalable solutions increased.

Microservice-based applications



01 Microservices are small services that are interconnected with many other microservices to present complex services like web applications.

02 Microservices provide greater scalability and make distributing the application over multiple physical or virtual systems possible.

03 Microservices architecture tackles the problem of productivity and speed by decomposing applications into smaller services that are faster to develop and easier to manage; if one microservice fails, the others will continue to work.

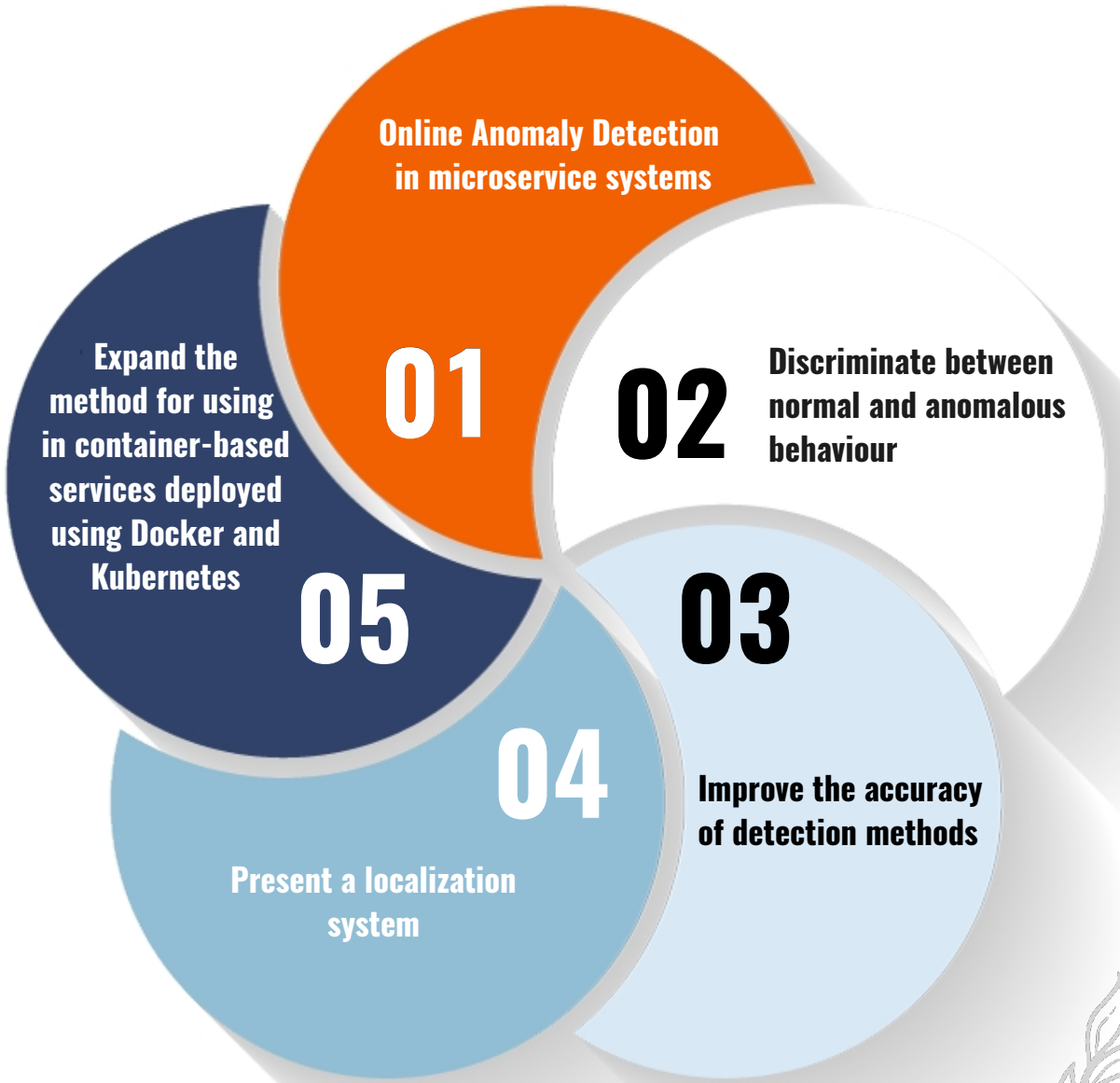
04 Each microservice can be written using different technologies, and they enable continuous delivery.

Microservice-based applications

Despite all these benefits, by increasing the degree of automation and distribution, application performance monitoring becomes more challenging because microservices are possibly short-lived and may be replaced within seconds.

Hence new requirements in the way of anomaly detection have emerged as these changes could also be the cause of anomalies.

Motivation


Online Anomaly Detection in microservice systems
01
02
Discriminate between normal and anomalous behaviour
03
Improve the accuracy of detection methods
04
Present a localization system
05
Expand the method for using in container-based services deployed using Docker and Kubernetes

Challenges

Defining a normal region that encompasses every possible normal behavior is very difficult.

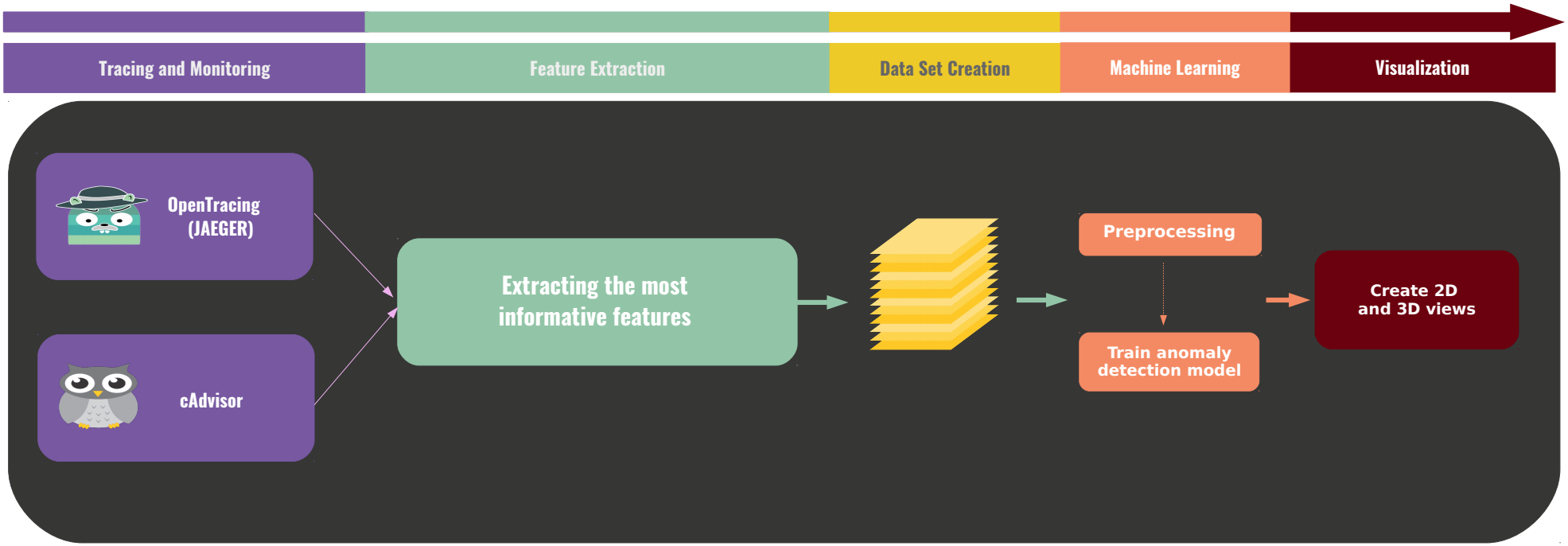
Normal behavior keeps evolving and the current notion of normal behavior might not be sufficiently representative in the future.

The exact notion of an anomaly is different for different application domains.

Availability of labeled data for training/validation of models used by anomaly detection techniques is a major issue.

Methodology

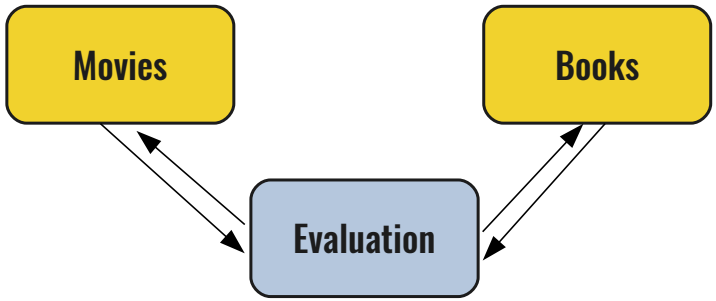
- The methodology is based on collecting streams of events produced by all or selected microservices, and sending them to the Data Processing Module.
- Machine learning algorithms are used to identify changes in services behavior.
- The methodology uses the performance metrics across microservices, containers, and nodes monitored resources.



Case Study

An open-source microservices-based application is developed for evaluating our proposed anomaly detection method.

It emulates an online movie and book encyclopedia that consists of 3 main services: Movies, Books and Evaluation services.



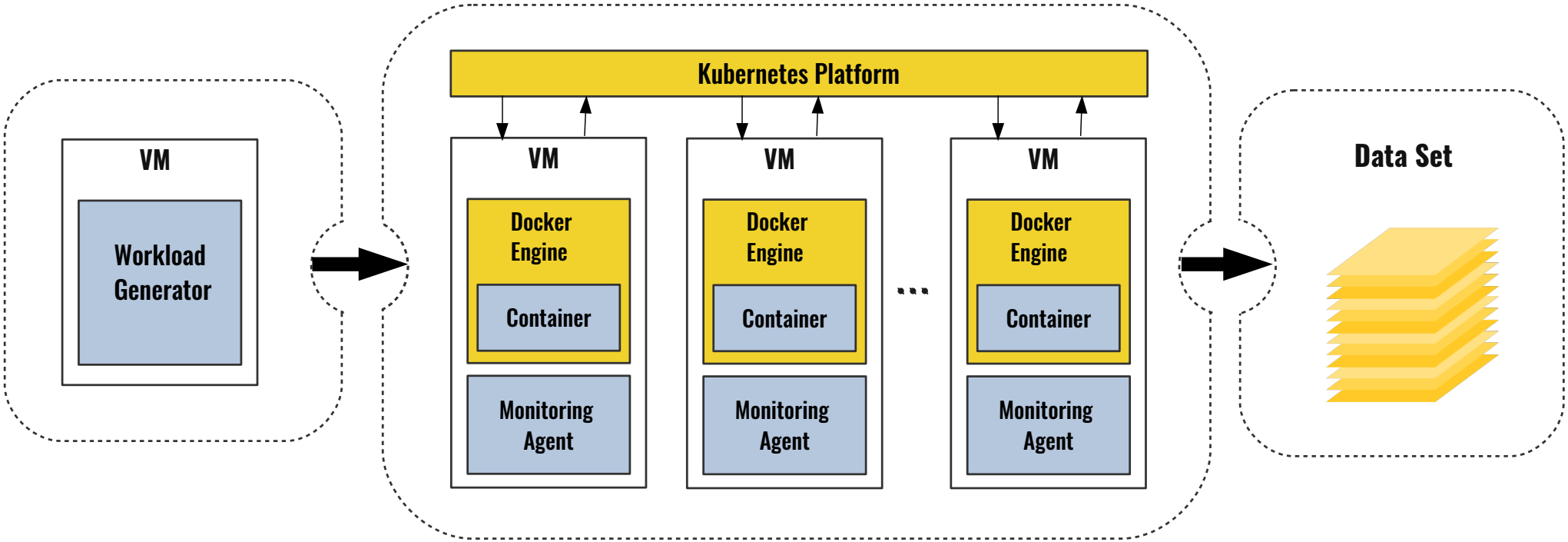
We deployed several instances of each service at the same time using Docker and Kubernetes.

The experimental environment consists of 12 VMs (nodes) in 3 servers.

The workload is generated by another application at different times.

Implementation of the target system in Microservice architecture

We deployed multiple instances of the microservices on several VMs:



Developing with Docker

- Containers are isolated workload environments in a virtualized operating system such as cloud.
- They are used to allocate computing resources, help to organize, migrate and develop microservices.
- Containers allow us to run an application and its dependencies in a resource-isolated process.
- Each component runs in an isolated environment and does not share memory, CPU, or the disk of the host operating system(OS).
- Our microservice-based application is composed of small services, each of which is in a container and runs its own process.
- Containers can communicate with each other through well-defined channels.

Developing with Docker

- Docker is a set of platform as a service products that uses OS-level virtualization to deliver software in packages called containers.
- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.

Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

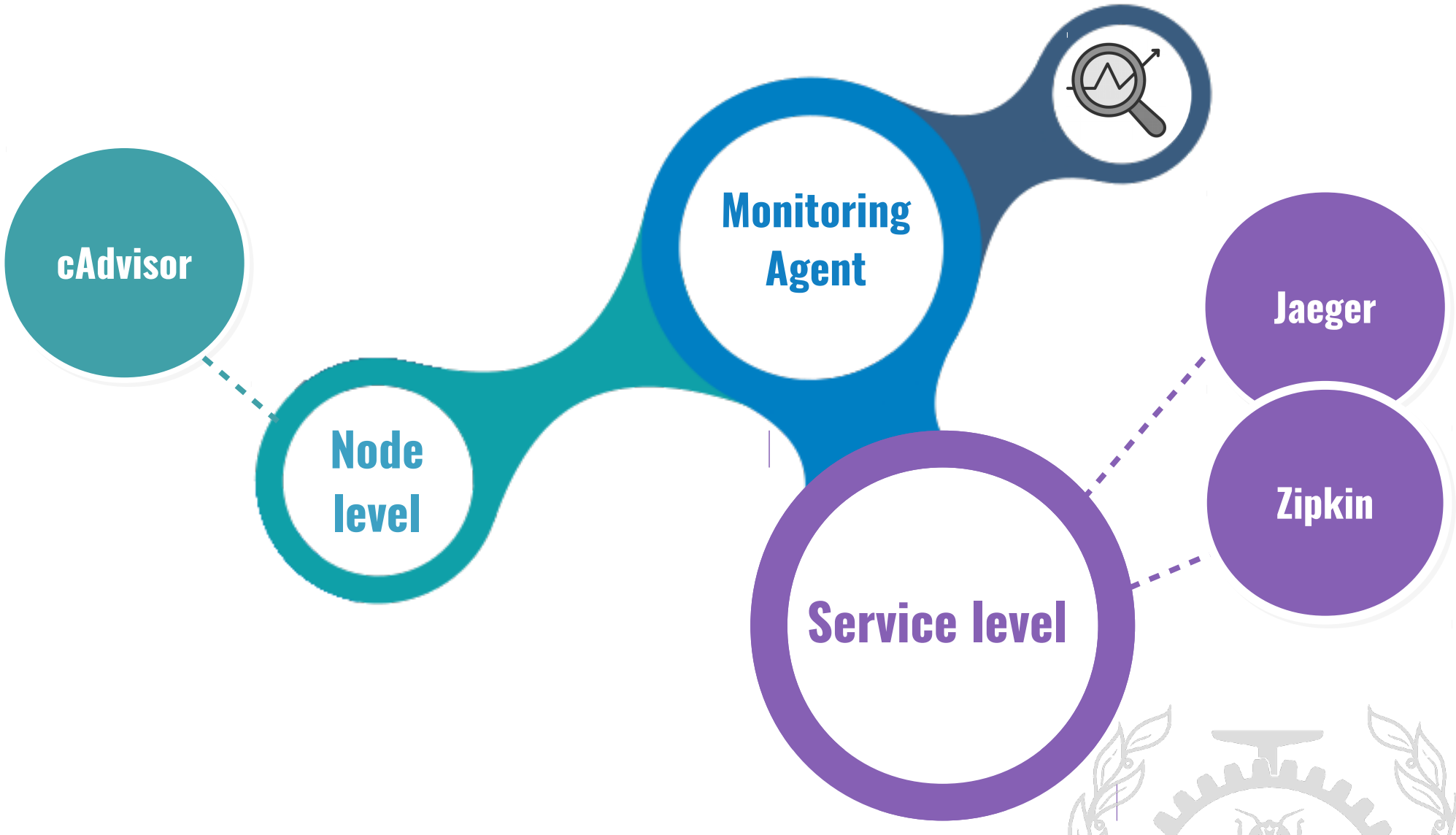
It has a large, rapidly growing ecosystem.

We use Kubernetes because its services, support, and tools are widely available.

The target system runs on a kubernetes platform which consists of 12 VMs.

Each VM has 1 CPU and 2 GB memory and VMs are connected through a 100 Mbps network.

A monitoring agent is installed on each of the VMs.



Fault injection

We consider injecting faults to our environment using two scenarios:

1. Injecting faults as intentional software bugs into the microservices code. Whenever a microservice is called, one of the following events may happen:
   - Pause the microservice for n milliseconds and then continue.
   - Calculate π with 50 bits of precision.
   - Perform the normal functionality

Fault injection

We consider injecting faults to our environment using two scenarios:

2 In the second scenario, the target is the containers in which the services are running.

CPU and memory problems can be simulated using Stress. It is a workload generator tool designed to subject your system to a configurable measure of CPU, memory, I/O and disk stress.

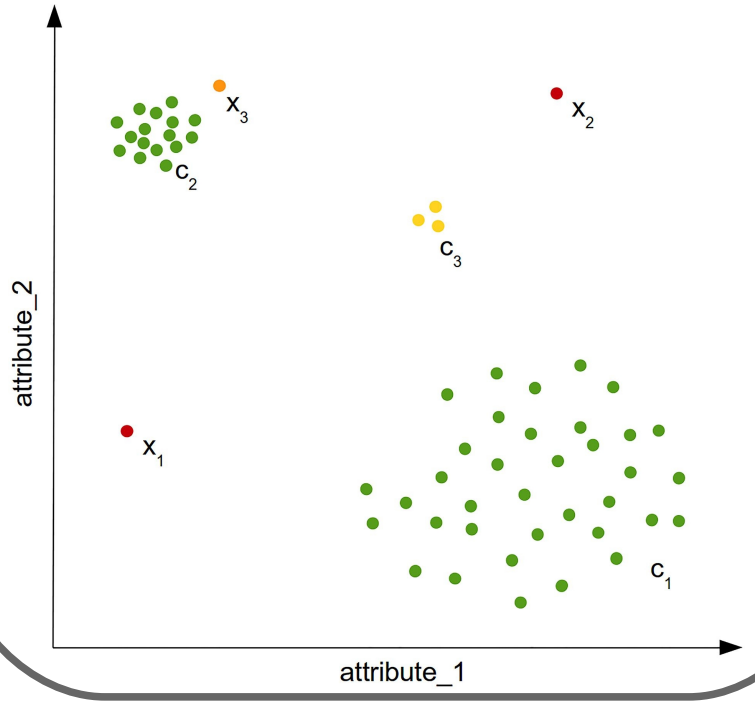
Network fault is also simulated using Pumba. It can do network emulation, simulating different network failures, like delay, packet loss/corruption/reorder, bandwidth limits and more.

Data Processing Module

Anomalies in data happen under various forms. Two main forms of anomalies are as follows:

Point anomalies

- Point anomalies are data points that are different from normal data.
- Two points X1 and X2 are significantly different from all other data, so they can be considered as anomaly.



Collective anomalies

- In this class of anomalies, we can not detect the individual data points as anomalies by themselves, however, their occurrence together as a collection may be an anomalous behaviour.

... ftp, http-web, ssh, smtp-mail, http-web, **ssh, buffer-overflow, ftp**, http-web, ftp, smtp-mail,http-web...

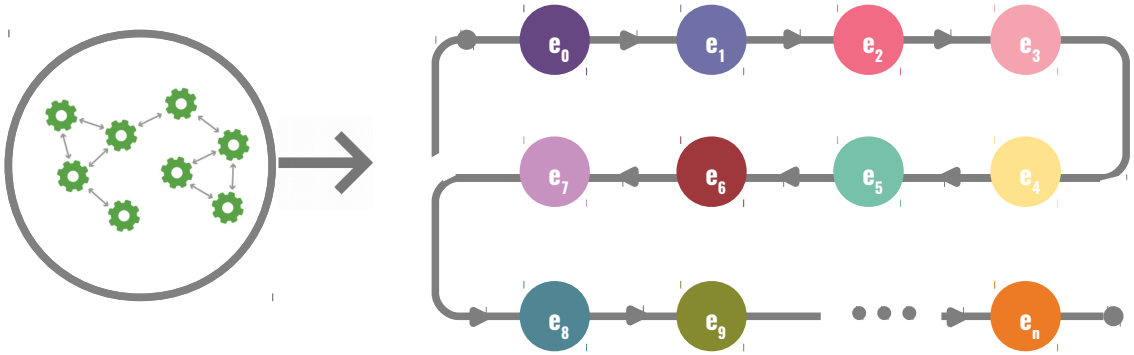
- where each individual event in the bolded sequence would not be anomalous, however, the collection together is a sign of a probable intrusion.

Point anomalies

- A variable number of instances of a microservice may run simultaneously inside a microservice-based application.
- Because the microservice application consists of various microservices, with teams working on each one, updates are committed irregularly and continuously.
- When an update is released, it may happen that one or more run the updated version and the other ones run the older version.
- Different versions of the service may behave differently in terms of performance, known as anomaly.
- Each instance will be able to output specific performance metrics that are accessible during the execution.

- One choice of design is that we perform classification in microservice instances level.
  - If we consider each instance (or container) at a given time as one point in our feature space, we will be able to investigate point anomalies in the system.
  - At this stage, we do not involve the relationships between services and only use the performance metrics of the nodes.

Collective anomalies



The model should be stable for current microservice instances and can be adaptable with changing the environment like changing or adding hardware.

Classic feature selection methods that don't consider specialized information in this field, are not used.

Communication information is obtained from OpenTracing: the detailed information of messages transferred between services is available.

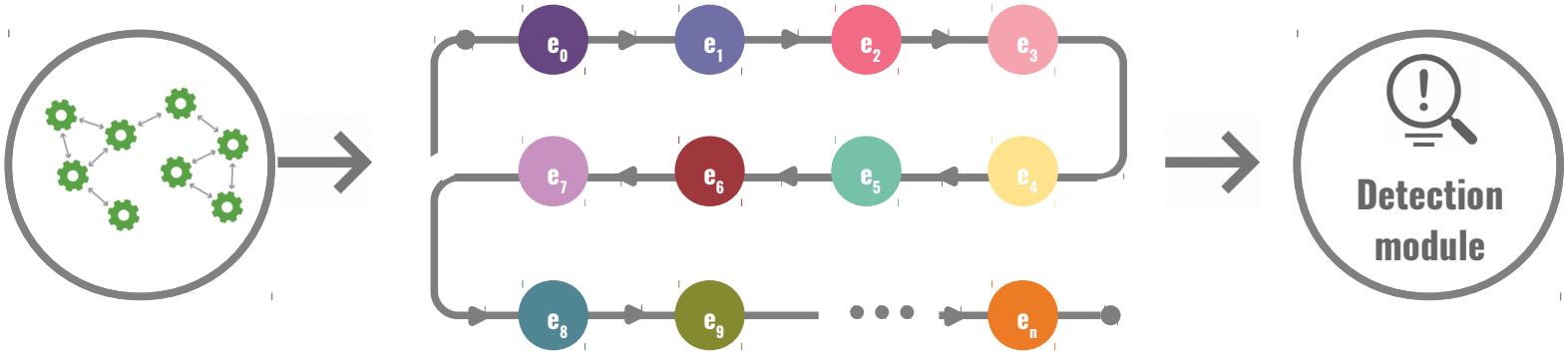
A trace T is represented as an enumerated collection of events sorted by the timestamps (e_0, e_1, ..., e_n).

Each event in the trace contains some attributes such as ID, parent ID, protocol, host address, return code, URL, function, response time, and timestamp.

The dataset A is obtained by collecting the representatives of the events (spans) during a time interval while some sequences of representatives are anomalous. Representatives are obtained by the combination of source address, destination address, function and etc.

We can even use the duration of representatives in a separate dataset B.

Collective anomalies



In the detection phase, we use this sequential information to make a prediction and compare the predicted output against the observed value.

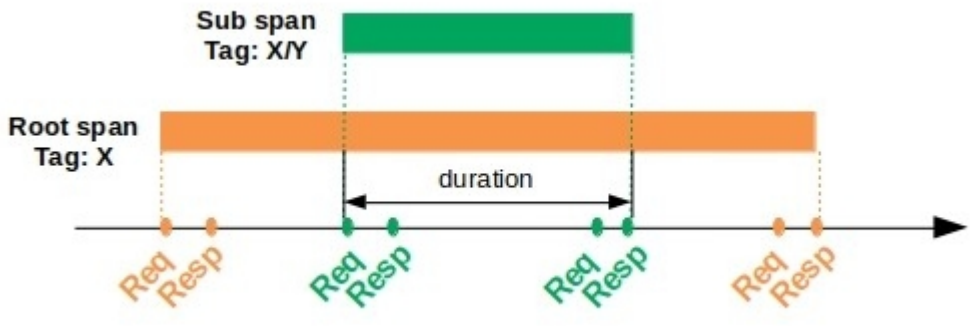
A limited number of events can be the result of an action. Therefore few of the possible events can appear as the next event in the sequence.

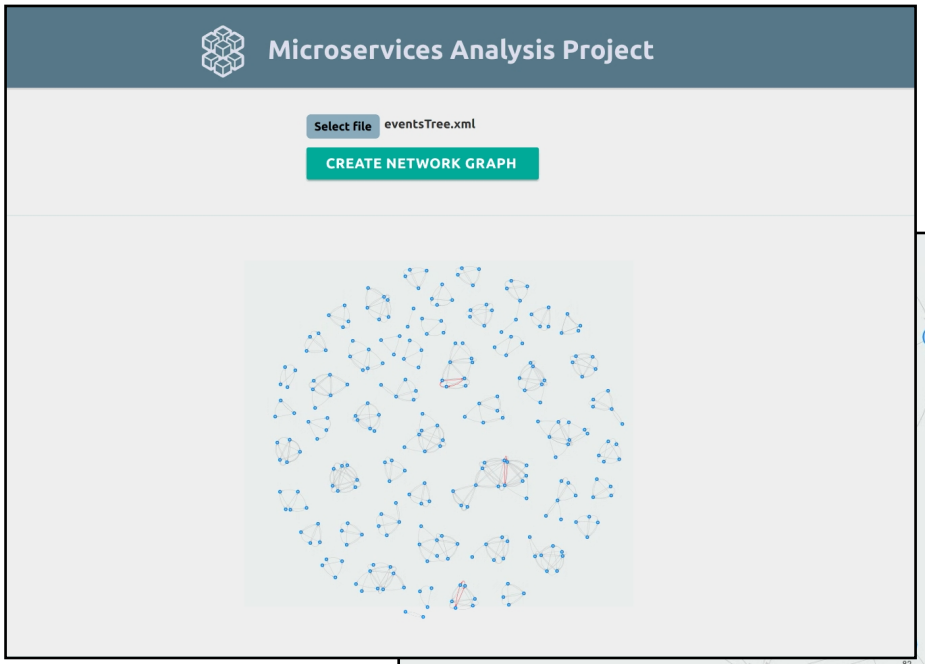
The LSTM network can be used in this part to learn the possible sequence of events and predict the next event.

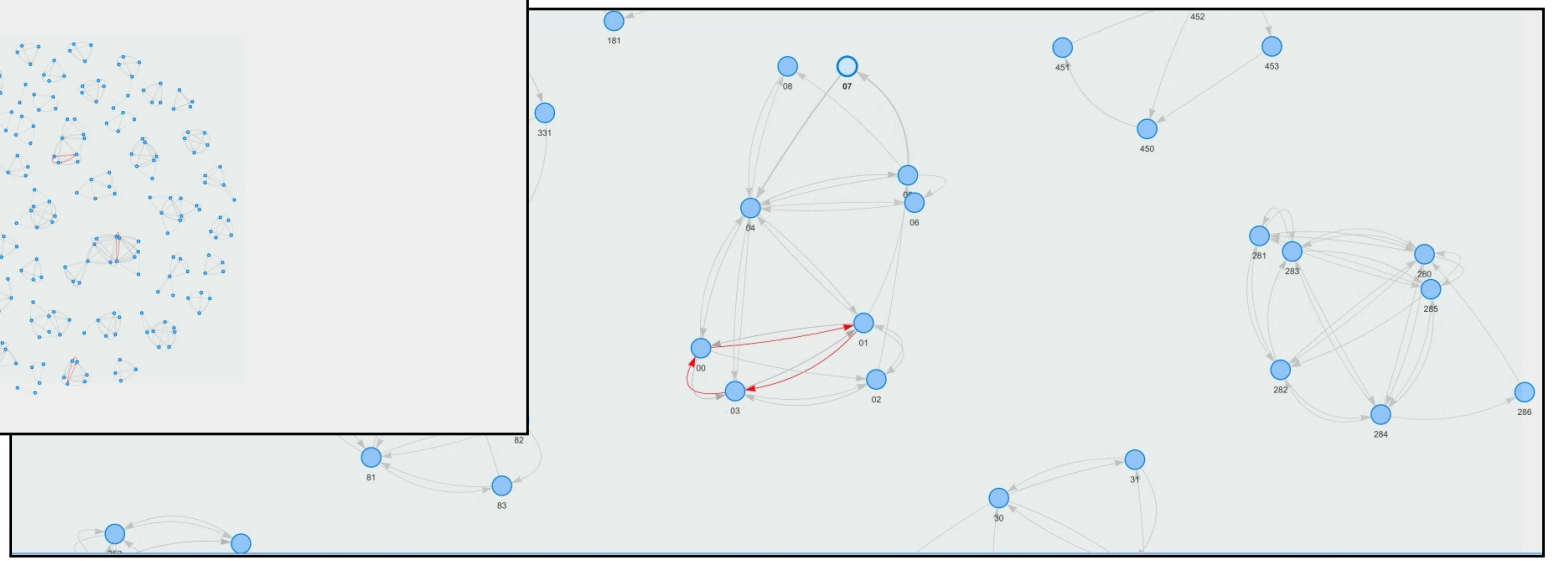
The anomaly is reported if the event observed in the next timestamp from the original sequence is different from the predicted one.

It also enables us to detect unexpected execution paths and helps us in root cause analysis.

Simultaneous work







Thank you for your attention!

Questions?

Iman.kohyarnejadfard@polymtl.ca
https://github.com/kohyar

References

[1] Z. Xu, X. Yu, Y. Feng, J. Hu, Z. Tari, and F. Han. A multi-module anomaly detection scheme based on system call prediction. In 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), pages 1376–1381, June 2013.

[2] A. Liu, C. Martin, T. Hetherington, and S. Matzner. A comparison of system call feature representations for insider threat detection. In Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, pages 340–347, June 2005.

[3] Michael Dymshits, Ben Myara, and David Tolpin. Process monitoring on sequences of system call count vectors. 2017 International Carnahan Conference on Security Technology (ICCST), pages 1–5, 2017.

[4] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. Deep learning for classification of malware system call sequences. In Byeong Ho Kang and Quan Bai, editors, AI 2016: Advances in Artificial Intelligence, pages 137–149, Cham, 2016. Springer International Publishing.

[5] Mathieu Desnoyers and Michel Dagenais. The lttng tracer : A low impact performance and behavior monitor for gnu / linux. In OLS Ottawa Linux Symposium, 2006.

[6] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg, 2006.

[7] Ulrich H.-G. Kre. Advances in kernel methods. chapter Pairwise Classification and Support Vector Machines, pages 255–268. MIT Press, Cambridge, MA, USA, 1999.

[8] Du, Qingfeng, Tiandi Xie, and Yu He. "Anomaly Detection and Diagnosis for Container-Based Microservices with Performance Monitoring." International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, 2018.

[9] Cao, Wei, Zhiying Cao, and Xiuguo Zhang. "Research on Microservice Anomaly Detection Technology Based on Conditional Random Field." Journal of Physics: Conference Series. Vol. 1213. No. 4. IOP Publishing, 2019.

[10] Nikiforov, Roman. "Clustering-based Anomaly Detection for microservices." arXiv preprint arXiv:1810.02762 (2018).

[11] Pahl, Marc-Oliver, and François-Xavier Aubet. "All eyes on you: Distributed Multi-Dimensional IoT microservice anomaly detection." 2018 14th International Conference on Network and Service Management (CNSM). IEEE, 2018.

[12] Nandi, Animesh, et al. "Anomaly detection using program control flow graph mining from execution logs." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016.