System performance anomaly detection using tracing data analysis

Iman Kohyarnejadjfard
Prof. Daniel Aloise and Prof. Michel Dagenais

POLYTECHNIQUE
MONTREAL
01 Introduction

02 Methodology and Implementation

03 Results

04 Current work

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.

The figure retrieved from: https://pngtree.com
Motivation

1. Online Anomaly Detection in system
2. Discriminate between normal and anomalous processes
3. Improve the accuracy of detection methods
4. Develop an automatic anomaly detection framework
5. Develop a performance anomaly prediction framework
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.
Why system calls?

01 System Call is a program signal for requesting a service from the system kernel.

02 System calls can represent low-level interactions between a process and the kernel in the system.

03 System call traces generated by program executions are stable and consistent during program’s normal activities so that they can be used to distinguish the abnormal operations from normal activities.

04 System call streams are enormous, and suitable to use in machine learning. A single process can produce thousands of system calls per second.

05 We can use three different representations of system calls: n-grams of system call names, histograms of system call names, and individual system calls with associated parameters.

06 System call sequences can provide both momentary and temporal dynamics of process behavior.
**Methodology**

- The methodology is based on collecting streams of system calls produced by all or selected processes on the system, and sending them to a monitoring module.
- Machine learning algorithms are used to identify changes in process behavior.
- The methodology uses a sequence of system call count vectors or sequence of system call duration vectors as the data format which can handle large and varying volumes of data.
The benchmarking tool is run on virtual machines with different configurations and varying load on resources; LTTng is used to record the different tracing data streams.

Trace compass is used to read tracing data, create tables of system calls and construct the initial vectors to use in the machine learning part.
Indexes instead of names
## Read Trace

<table>
<thead>
<tr>
<th>System Call Name</th>
<th>Index</th>
<th>Time stamp</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td>1</td>
<td>t₀</td>
<td>d₀</td>
</tr>
<tr>
<td>open</td>
<td>2</td>
<td>t₁</td>
<td>d₁</td>
</tr>
<tr>
<td>open</td>
<td>2</td>
<td>t₂</td>
<td>d₂</td>
</tr>
<tr>
<td>read</td>
<td>0</td>
<td>t₃</td>
<td>d₃</td>
</tr>
<tr>
<td>open</td>
<td>2</td>
<td>t₄</td>
<td>d₄</td>
</tr>
<tr>
<td>newlstat</td>
<td>317</td>
<td>t₅</td>
<td>d₅</td>
</tr>
<tr>
<td>close</td>
<td>3</td>
<td>t₆</td>
<td>d₆</td>
</tr>
<tr>
<td>newlstat</td>
<td>317</td>
<td>t₇</td>
<td>d₇</td>
</tr>
<tr>
<td>newstat</td>
<td>316</td>
<td>t₈</td>
<td>d₈</td>
</tr>
<tr>
<td>close</td>
<td>3</td>
<td>t₉</td>
<td>d₉</td>
</tr>
<tr>
<td>mmap</td>
<td>9</td>
<td>t₁₀</td>
<td>d₁₀</td>
</tr>
<tr>
<td>newstat</td>
<td>316</td>
<td>t₁₁</td>
<td>d₁₁</td>
</tr>
</tbody>
</table>

**Frequency of each system call and Total duration of each system call in this window**

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>⋯</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₃</td>
<td>d₀</td>
<td>d₁+d₂+d₄</td>
<td>d₆+d₉</td>
<td>⋯</td>
<td>0</td>
<td>d₈</td>
<td>d₅+d₇</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>2</th>
<th>2</th>
<th>⋯</th>
<th>0</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₃</td>
<td>0</td>
<td>d₂+d₄</td>
<td>d₆+d₉</td>
<td>⋯</td>
<td>0</td>
<td>d₈+d₁₁</td>
<td>d₅+d₇</td>
</tr>
</tbody>
</table>

**Length of Vector = 318**
Use Case 1

The open source MySQL synthetic benchmarks tool, Sysbench, with OLTP test (OnLine Transaction Processing) in complex mode.

A virtual machine with different workloads, such as:
I. (CPU problem) Setting the VM CPU cap too low (e.g., 1 CPU core, while running 8 threads of MySQL)
II. (Memory problem) Setting the memory cap too low (e.g., 256 MB memory, while the MySQL table is of size 6 GB)

Sliding window = 10k system calls
overlapping size = 100 system calls

18k normal and anomalous samples
Use Case 2

The Chrome UnderStress 1.0 chrome extension is used to open, close, and refresh many light and heavy pages in Chrome with configurable speed.

Faults are simulated by running this Chrome extension on the VMs with different amount of CPU and memory resources.

Sliding window = 10k system calls
overlapping size = 100 system calls

18k normal and anomalous samples
Learning Module

Supervised, unsupervised, and semi-supervised techniques can be used to detect performance anomalies in processes, depending on the volume of available labeled data.

first, we assume that enough labeled training samples are available.
Learning Module

Supervised, unsupervised, and semi-supervised techniques can be used to detect performance anomalies in processes, depending on the volume of available labeled data.

01 SUPERVISED

02 UNSUPERVISED

First, we assume that enough labeled training samples are available.

Since providing labeled data for the whole data distribution is not always possible, we propose to use an unsupervised approach.
first, we assume that enough labeled training samples are available.

Since providing labeled data for the whole data distribution is not always possible, we propose to use an unsupervised approach.

However, unsupervised approaches usually present worse classification performance than supervised methods in practice given that no priori information is exploited. Therefore, in order to introduce a form of supervision into the unsupervised approach and improve the detection performance, we propose a semi-supervised approach.
**1 Reduce Sparsity**
Sparse matrices are common in machine learning. They occur in some data collection processes or when applying certain data transformation techniques like one-hot encoding or count vectorizing.

**2 Normalization**
The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

**3 Feature Selection**
It selects each feature independently according to their scores under the Fisher criterion, which leads to a suboptimal subset of features.

**4 Supervised Learning**
Supervised multi-class anomaly detection is done in this step using SVM with rbf kernel function.
Introduction

Methodology

Results

Current work

Conclusion

Unsupervised Model

Reduce Sparsity

Normalization

K-Means

DBSCAN

Unsupervised Model

Reduce Sparsity

Normalization

K-Means

DBSCAN

Unsupervised Model

Reduce Sparsity

Normalization

K-Means

DBSCAN
Semi-supervised Model

**Introduction**

**Methodology**

**Results**

**Current work**

**Conclusion**

- Feature Selection
- PCA
- Fisher Score
- KMeans
- DBSCAN
Semi-supervised Model

- Feature Selection
  - PCA
  - KMeans
  - DBSCAN

- Fisher Score

Number of Features
1. Kernel Tracing Data Extraction
   - LTTng Tracing
   - Read Trace Data (Extract streams of system calls)
   - Apply sliding window
     - Frequency of system calls
     - Duration of system calls

2. Preprocessing
   - Normalization
   - Reduce Sparsity
     - Fisher score feature selection

3. Clustering
   - DBSCAN clustering method

The architecture of the proposed Semi-supervised framework
Experimental results of the supervised method

MySQL dataset

Chrome dataset

SVM-based anomaly detection accuracy versus the different number of top-ranked features;
Heat map of the duration-based and frequency-based supervised anomaly detection accuracy using different parameters $\gamma$ and $C$ for the Mysql dataset

Heat map of the duration-based and frequency-based supervised anomaly detection accuracy using different parameters $\gamma$ and $C$ for the Chrome dataset
The performance of the proposed supervised anomaly detection approach

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MySQL Process</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency(ℓ=17)</td>
<td>0.928</td>
<td>0.989</td>
<td>0.968</td>
</tr>
<tr>
<td>Duration(ℓ=8)</td>
<td>0.937</td>
<td>0.988</td>
<td>0.978</td>
</tr>
<tr>
<td><strong>Chrome Process</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency(ℓ=103)</td>
<td>0.951</td>
<td>0.990</td>
<td>0.994</td>
</tr>
<tr>
<td>Duration(ℓ=112)</td>
<td>0.959</td>
<td>0.991</td>
<td>0.985</td>
</tr>
</tbody>
</table>
The visual result of dbscan clustering after choosing $l = 103$ features with the highest Fisher score

The visual result of K-Means clustering after choosing $l = 103$ features with the highest Fisher score
Validation of DBSCAN based semi-supervised technique on original features versus where the Fisher score feature selection method is applied

<table>
<thead>
<tr>
<th>Frequency-based Data set</th>
<th>ARI</th>
<th>Number of clusters</th>
<th>Duration-based Data set</th>
<th>ARI</th>
<th>Number of clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MySQL Process</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original Features ($\epsilon = 10^{-3}$)</td>
<td>0.281</td>
<td>17</td>
<td>Original Features ($\epsilon = 10^{-3}$)</td>
<td>0.278</td>
<td>18</td>
</tr>
<tr>
<td>Fisher Score ($\ell = 17$ and $\epsilon = 10^{-3}$)</td>
<td>0.874</td>
<td>8</td>
<td>Fisher Score ($\ell = 8$ and $\epsilon = 10^{-3}$)</td>
<td>0.855</td>
<td>8</td>
</tr>
<tr>
<td><strong>Chrome Process</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original Features ($\epsilon = 5 \times 10^{-4}$)</td>
<td>0.254</td>
<td>21</td>
<td>Original Features ($\epsilon = 10^{-3}$)</td>
<td>0.127</td>
<td>27</td>
</tr>
<tr>
<td>Fisher Score ($\ell = 103$ and $\epsilon = 5 \times 10^{-4}$)</td>
<td>0.823</td>
<td>9</td>
<td>Fisher Score ($\ell = 112$ and $\epsilon = 10^{-3}$)</td>
<td>0.701</td>
<td>11</td>
</tr>
</tbody>
</table>
Current and future work

01 Learning system performance
   Anomalies using tracing data analysis

02 Anomaly detection in microservice systems using tracing data and Machine Learning

03 Analyzing the root cause of anomalies in microservice systems
Anomaly detection in microservice systems using tracing data and Machine Learning

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

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

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.

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

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.
Anomaly detection in microservice systems using tracing data and Machine Learning

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.

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.
One of the essential parts of AIOps platforms is to detect the anomaly rapidly and decrease the reaction time before it leads to a service or system failure.

**Sources of anomalies**

1. **performance problems**
2. **component and system failures like outages**
3. **security incidents**
Microservices mostly use Representational State Transfer (REST) as a way to communicate with other microservices.
Microservices mostly use Representational State Transfer (REST) as a usual way to communicate with other microservices.

1. LTtng is the primary tool we use in the data collection step.

2. Traces in microservice architectures are composed of events (spans).

3. An event contains information describing the state, performance, and further service characteristics at a given time $t_i$.

4. Similar to structures for natural languages, we can convert the tracing data into sequential data to utilize and improve analysis methods developed in these domains.
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, response time, and timestamp.

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 timestep from the original sequence is different from the predicted one.
Thank you for your attention!

Questions?

Iman.kohyarnejadfar@polymtl.ca
https://github.com/Kohyar
References


