System performance anomaly detection using tracing data analysis

Iman Kohyarnejadfar
Prof. Daniel Aloise and Prof. Michel Dagenais

POLYTECHNIQUE
MONTREAL
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.
Motivation

Online Anomaly Detection in system

01

Discriminate between normal and anomalous processes

02

Develop a performance anomaly prediction framework

03

Develop an automatic anomaly detection framework

04

Improve the accuracy of detection methods

05
Challenges

01
Defining a normal region that encompasses every possible normal behavior is very difficult.

02
Normal behavior keeps evolving and the current notion of normal behavior might not be sufficiently representative in the future.

03
The exact notion of an anomaly is different for different application domains.

04
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 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.
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.
Our Use Case

The open source MySQL synthetic benchmarks tool, Sysbench, with oltp test in complex mode.

A virtual machine with different workloads, such as:
I. (CPU problem) Setting the VM’s CPU cap to too low (e.g., 1 CPU core, while running 8 threads of MySQL)
II. (Memory problem) Setting the memory cap to 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
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 machine learning part.
Indexes instead of names

- read
- write
- open
- close
- stat
- fstat
- lstat
- poll
- mmap
- eventfd
- munmap
- readv
- ...
- ...
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
# 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>3</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>

**Trace Compass**

- **Windowing Size = 10**
- **Windowing step = 2**

**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+d₃</td>
<td>d₀+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**
Preprocessing

1. **Scaling**
   It selects the same number of samples from each class without considering any order in vectors.

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. **Sparsity**
   Sparse matrices are common in machine learning. They occur in some data collection processes or applying certain data transformation techniques like one-hot encoding or count vectorizing.

4. **Fisher score**
   It selects each feature independently according to their scores under the Fisher criterion, which leads to a suboptimal subset of features.
We have analysed the created dataset in 3 steps: At first, we trained the SVM classifier using labelled data. In the second step, we did clustering on samples (unsupervised) and finally, we used parameters estimated in the classification step to cluster the samples.
Supervised Learning

- Raw Data
  - Reduce Sparsity
  - Normalization
  - Fisher Score

- SVM
The Fisher score for each system call in frequency-based approach

The Fisher score for each system call in duration-based approach
Unsupervised Learning

Projection of Dataset using PCA

Projection of Dataset using TSNE
Unsupervised Learning

- PCA
- Feature Selection
- Autoencoder
- KMeans
- DBSCAN
Semi-Supervised Learning

Introduction
Methodology
Results
Conclusion

Feature Selection

DBSCAN

Fisher Score

Autoencoder

Number of Features

The figure retrieved from: https://www.vectorstock.com
Supervised Learning accuracy versus different number of top-ranked features
Results

Heat map of the frequency-based anomaly detection accuracy using different $\gamma$ and C.

Heat map of the duration-based anomaly detection accuracy using different $\gamma$ and C.
Results

Accuracy of the supervised learning approach on multiple runs.

The performance of the proposed RBF based anomaly detection approach compared to the Sigmoid and polynomial based methods.
We gained $\text{ARI}=0.8471$ by utilizing DBSCAN clustering method with $\text{eps}=0.001$ and Fisher Score feature selection with (number of top-ranked features)$=3$.
Future Directions

1. Test the methodology on other use cases to find stable and accurate strategy.

2. Utilize other metrics and analysis such as critical path data extraction to improve the performance.

3. Apply the methodology for online anomaly detection.

4. Employ the extracted features in developing the anomaly prediction framework.
Thank you for your attention!

Questions?

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