Paper Review: PerfScope: Practical Online Server Performance Bug Inference in Production Cloud Computing Infrastructures

Sohail Ahmed Shaikh

February 12, 2019

1 Summary:

This paper describe Perfscope which is an online bug inference tool. It helps developers in tracking down performance bugs during production run without requiring the source code of the application.

2 Description:

Diagnosing performance bugs in a cloud infrastructure is a challenging task because reproduction of bugs outside the production environment is extremely difficult. Existing offline debugging tools will not work in this scenario. Also, performance bugs have little diagnostic information and they may not produce any error message. In such conditions searching for a specific problem among possibly thousands of functions is kind of a needle in a haystack problem. Perfscope attempts to address this.

Perfscope is triggered when a performance anomaly is detected by an existing anomaly detection tool and analyzes a 5 minute window of recent system calls. System calls are used because they can be easily collected via kernel tracing tools with less overhead and also works in a virtualized environment.

Perfscope has a signature driven approach. It extracts function signatures without requiring application source code. It does not use low level system metrics since they are platform dependent. Instead it utilizes closed frequent system call episodes which are largely stable under different workloads and environments. Also raw system call sequences are not used to represent the signatures of a function as they are sensitive to environment changes. The system call episodes are extracted using a frequent episode mining algorithm called A-priori method. The frequent episodes are further pruned for faster processing while extracting the episodes.

System call traces could be quite huge, hence they are segmented into execution units which are characterized using various features. A set of abnormal identified units are used as clues to identify functions that may have caused bugs. The abnormal execution units are identified using a light weight unsupervised learning approach. The advantage of unsupervised learning is that it does not require any labelled training data. Using this approach the execution units performing similar operations are clustered. Next, outlier detection is performed on each cluster to identify the abnormal execution units using Nearest neighbor algorithm. Outlier detection is not performed if the number of execution units in a cluster is small (eg < 4). Instead the entire small cluster is considered abnormal.

The abnormal execution units are then mapped to functions that resulted in a bug. For each execution unit a set of frequent closed system call episodes are extracted. The frequency of the functions occurring in these episodes are then ranked to give a list of potential bug related functions.

The authors also evaluated Perfscope using real performance bugs in some open source systems to demonstrate the accuracy and performance of the system and conclude that Perfscope can successfully identify true bug-related functions without imposing a heavy runtime overhead.

3 Strong points:

1) It does not require source code.
2) It is application agnostic.
3) It does not require bug reproduction in a production environment.
4) It imposes less overhead:1.8% on average.
5) It is non intrusive.

4 Weak points:

1) Functions having the same frequent episode set will not be distinguished correctly by PerfScope.
2) It works on a single server environment. Performance degradation due to poor system interactions such as poor I/O or congested network cannot be detected by PerfScope.