Paper Review

“ PerfScope: Practical Online Server Performance Bug Inference in Production Cloud Computing Infrastructures ”

1. Summary

This paper introduces a practical online performance bug inference tool to help the developer in a production cloud computing infrastructures. PerfScope can narrow down the search scope of the bug-related functions to a smaller percentage.

Online performance bugs have the following properties: hard to reproduce and unexpected. Perfscope does not require production-run bug reproduction by achieving online bug inference to diagnose bugs. Perfscope analyzes a window (e.g. five minutes) of recent system call traces collected on the faulty server pinpointed by a component-level fault localization tool. The reason for using system calls to diagnose problems is that system calls are easy to get with low cost and system calls can be strongly related to buggy executions. (e.g. while loop not end).

The implementation of PerdScope includes an offline function signature extraction scheme and online bug inference algorithms. Low-level system metrics (e.g. execution time, CPU usage) often vary significantly among different platforms so system calls are a good metric. System call entry includes the timestamp, syscall name, and sys_exit. By computing, we can pair syscalls with functions. An a-priori method is used to find a robust function signature. Starting at the basic event type (e.g. a single system call) and building up with more complex sequences. (shown in Figure 2 in the paper). (sys_write, *, sys_read) is the result where * stands for the middle procedure. Online bug inference uses an unsupervised learning approach because it is suitable for unexpected bugs. First, apply a top-down hierarchical clustering and then perform outlier detection with each cluster to identify abnormal execution units. Finally, map these abnormal execution units to bug-related application functions and rank them based on an abnormality degree metric.

Thanks to the low overhead, PerdScope shows great performance in evaluation (1.8% average overhead). I/O and locking operations were the most common system call generating problems that can be correctly handled by PerfScope.

The state-of-art research includes Triage depends on checkpoint replay, which is more intrusive. X-ray are based on binary instruments which cannot diagnose the bug caused by software. Source code offline debugging cannot detect runtime bugs and is an invasion of privacy. Both user-level and kernel-level tracing produced 280% overhead.

2. Advantages

+ Online performance bug inference tools are costless and powerful. By using system calls, which are powerful enough to reflect problems as well as low cost. And can diagnose runtime problems other than offline tools.

Feiran Qin qinfr@shanghaitech.edu.cn 2019533161

+ The robust application function signature extraction approach both works broadly and accurately.
+ Signature is extracted offline which can take full advantage of existing data sets.

3. Disadvantages

– The detection is limited since its only source is syscalls.
– I think sometimes the syscall can be complex, (e.g. when the robustness of a program shows, try-catch for example).
– For distributed systems, can we also think of the communication between nodes.

4. Brainstorming

Some tools like Copilot use open-source codes on Github as their data set and do GPT-2 for coding guidance. An idea comes to my mind that uses the data sets to predict which part of a code goes wrong since we may guess which code he copies or references from Github. We can collect syscalls and guess if he loops without an end or uses a common but wrong data structure.

Feiran Qin qinfr@shanghaitech.edu.cn 2019533161