Paper Review: X-ray: Automating Root-Cause Diagnosis of Performance Anomalies in Production Software

Sohail Ahmed Shaikh

February 18, 2019

1 Summary:

This paper aims to aid automatic diagnosis of root causes of performance anomalies related to misconfiguration and input errors. This is done by a technique called performance summarization which executes binaries, records events and analyzes it to infer potential root causes. The authors also evaluate the performance of the system in several tools as well as it's performance overhead.

2 Description:

Debugging performance problems is a daunting task because of complexity of systems, scarcity of error messages and lack of knowledge of the systems and /or interactions in the system. While there are several tools that can detect the bottlenecks in systems, monitor the events when performance degrades or reveal symptoms, inferring the root cause from these is a fairly difficult task. This problem is also exacerbated by misconfiguration issues.

The authors propose to solve this by introducing X-ray, a tool to perform root cause analysis. The scope of the project is limited to performance degradation which is caused by configuration settings and anomalous requests (or inputs). The authors justify this decision by referring to the statistically high rate of human and configuration errors in deployed systems.

The basic working of X-ray is as follows: X-ray can be divided into an online phase and offline phases. The online phase records the system and has a low overhead. The offline phases are more compute and overhead intensive (required for root cause analysis) and done outside a production environment.

X-ray relies on monitoring the execution of the application binary (which is referred as binary instrumentation in the paper) instead of source code or log messages. The users of the system need to specify the configuration files and input sources for applications. An interval of software execution is recorded and analysed based on a selected cost metric such as latency, CPU, file system usage or network usage. X-ray utilizes a modified version of ConfAid - a tool that reports potential root cause of a program. ConfAid works by assigning a taint id to registers and memory addresses when data is read from configuration files. As the program executes the taint id's are propagated to other locations in the address space which depends on control flows. All variables having a taint id may not cause performance degradation hence a weight is associated with each taint id to represent the strength of relationship between the taint values and the root cause. When a failure event is detected, all the root causes associated with that particular flow of the program are ordered by weight of the taint ids, which are then reported. X-ray calculates control flow taint of every basic block since no a-priori knowledge of the system is assumed (thus differing

from ConfAid). Also, in multi-threaded programs to handle the issue of path explosion, alternative paths in control flow within only a single thread are examined. As opposed to ConfAid, X-ray is an online tool.

Additionally, X-ray provides users the option to analyze performance during specific time periods or a specific causal path to debug particular issues. This enables users to control the scope of analysis. It also allows comparing execution of two similar operations to explain the difference in performance.

The offline module of X-Ray consists of replaying the determinstic recording of the system. Because this analysis is heavyweight, it is performed outside the production environment. Identifying potential root causes of performance degradation is done by performance summarization. X-Ray essentially pinpoints why performance anomalies have occurred without requiring source code by utilizing PIN - which is a widely used tool developed by Intel to instrument x86 binaries.

The authors have modified the linux kernel and glibc library to implement determinstic record and replay. The reason for not implementing this within PIN and instead in the kernel is because record and replay has a high overhead if done outside the kernel. The memory management for the PIN tool is done by the kernel module the authors implemented, to avoid different results while recording and replaying the system.

Finally, the authors evaluate the performance of X-ray on four diverse servers and conclude that X-ray ranks true root causes in majority of the cases with very less average overhead. The evaluation was done on Apache, Postfix, PostgreSQL and lighttpd by reproducing 16 different issues and the average overhead was found to be 2.3% with good accuracy.

3 Strong points:

1) It's different from previous research which reveal what events occur when performance degrades. X-ray goes a step further in associating these events to root causes.
2) X-ray can estabilish causality within application components without requiring the developers to annotate the code.
3) Performs block level analysis so it can identify root causes even when requests are dissimilar.
4) It does not require schemas from the users.
5) It is able to record executions and replay them with very less average overhead of 2.3%. Hence it can be run on production environments.
6) Does not require source code and hence is not intrusive.
7) Accuracy of X-ray is not highly affected by the nature of the exact heuristics that are used. Hence it has fairly stable results.

4 Weak points:

1) It is platform dependent, since, the recording module has been implemented in Linux Kernel.
2) It cannot track the taint inside the OS as it does not consider dependence between system calls since system calls are not monitored.
3) X-ray is limited to configuration and program inputs. Other root causes such as program bugs cannot be detected.
4) Offline execution is very slow as compared to online execution. Thus, root cause analysis will not be real time.
5) It can be deployed in a production environment as it has a low overhead of around 2.3%. However, this evaluation is based on a very small number of applications and may not scale well.

5 Improvements:

1) System calls may be monitored to identify system call parameters that could degrade the performance. However, this needs to be carefully evaluated as the overhead of the system might increase and system calls of unrelated tasks might be recorded.