Framework for Continuous and Targeted Tracing of Software Execution

Abstract

Reasoning about software quality, potential bugs, code coverage, and software dynamic behavior (e.g., execution hot spots) is not an easy task. Existing approaches for doing so predominantly rely on static analysis. However, static analysis is inherently imprecise. The alternative to static analysis is dynamic analysis, which takes as input execution traces of software under test (e.g., operations on memory locations). However, generation of execution traces is not an easy task due to complexity of the process for generating traces and associated probe effects. It can also introduce a significant overhead, making it hard to integrate with continuous engineering techniques (i.e., generating and recording traces on each commit).

The goal of this project is to design and implement a framework for targeted tracing of software execution that integrates well with continuous engineering processes. The practical part of the work aims to produce execution trace of software under test, in a format and with details, that is suitable for various analyses. It will consider complementing existing, low-overhead tools (e.g., DynamoRio) with new functionalities. Considering that frequent generation of execution traces (e.g., on every commit) is computationally expensive, this project will aim to create an approach that enables targeted tracing – tracing only parts of the software that is of interest for a particular analysis. The targeted tracing is the main research contribution of this work. This solution should not only enable tracing of different parts of software (or tracing with arbitrary granularity), but will also be able to track changes in software versions and decide about parts of software that need to be re-traced (through tests selection and tracing configuration).

The main practical benefit of this approach will be the ability to maintain up-to-date execution trace of software that corresponds to the source code version under development (i.e., the last commit). The approach will profit from targeted tracing configuration and reasoning about code changes to minimize the overhead when generating execution traces. At the same time, the approach will minimize the probe effects that tracing has on software behavior.

The basic technology for tracing will be based on existing software tracing binary instrumentation tools (e.g., DynamoRIO). It will be necessary to modify one of these tools so they can record not only raw tracing information (e.g., access to memory locations), but also to extract debug symbols according to executed instructions. Furthermore, it will be necessary to create an external execution engine of this framework in a language more suitable for doing complex analysis (e.g., Python or Java). The framework should be configurable with options for targeting specific functions and threads for execution tracing, with the possibility of online configuration (i.e., during the execution). The framework should write the execution trace in a configurable form of JSON files. Finally, the whole approach should be packed into a Docker container and be ready for a server deployment (e.g., over Jenkins or Bamboo), for plug & play integration with the existing continuous engineering and DevOps practices.