Parikshan: A Testing Harness for In-Vivo Sandbox Testing

Abstract

One of the biggest problems faced by developers testing large scale systems is replicating the deployed environment to figure out errors. In the recent years there has been a lot of work in record-and-replay systems which capture

In this work we present a testing harness for production systems which allows the capabilities of running test-cases in a sandbox environment in the wild at any point in the execution of an integrated application. The paper levarages, Linux Container Shells(LXCs) to launch test instances in forked VM's from running instances of a process. The LXC shell provides a sandbox environment, for safe execution of test-cases provided by the users without disturbing the execution environment. Test cases are initiated using user-defined probe points which launch test-cases using the execution context of the probe point. Our sandboxes provide a seperate namespace for the processes executing the test cases, replicate and copy inputs to the parent application, safetly discard all outputs, and manage the file system such that existing and newly created file descriptors are safetly managed.

We believe our tool provides a mechanism for practical testing of large scale multi-tier and cloud applications. In our evaluation provide a number of use-cases to show the utility of our tool.

1 Introduction

As application software grows and gets more complicated, testing large scale applications has become increasingly important. However, it is often impossible to recreate realistic workloads in an offline development environment for large scale multi-tier or cloud based applications. On the other hand, there has been an equally impressive increase in the scale of computing resources, and distributed scalability of infrastructure. This often allows for redundant computation, which can be used for testing purposes. Leveraging this abundance of resource we present a testing harness which allows the user to dynamically insert test cases in a production environment.

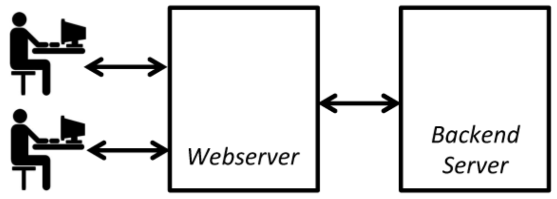

Figure 1: Workflow

Our tool called `Parikshan`[1] which allows capturing the context of application, and allows for a test-case to be run without effecting the sanctity of the actual application. This is achieved by using dynamic instrumentation mechanisms to initiate a VM from forking off an executed state and encapsulating the forked execution in a VM. The user can pre-define probe points for dynamically inserting test-cases (by default the entry and exit of each function) is considered a probe point. Since the test is executed in a VM it acts like a sandbox which restricts it from causing any perturbation to the state of the parent process

The key contributions of this paper are:

- Our tool provides a sandbox environment to execute test cases in the production environment. This allows for a safe and secure test harness which does not effect the production state, and allows the application to safely proceed in it's execution.
- We allow for dynamic insertion of the test case, and safetly capturing the context of the applica-

[1]Parikshan is the sanskrit word for testing

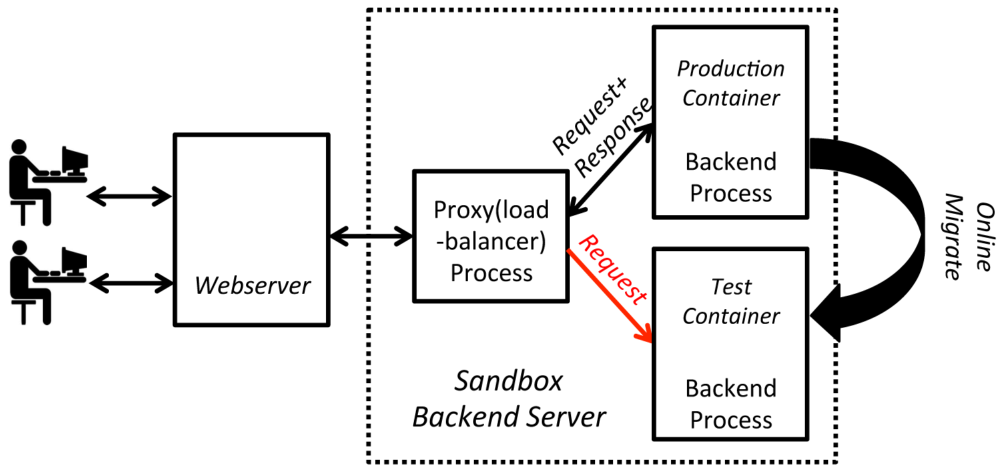

Figure 2: Workflow

tion. `Zero-Probe Effect` probe points are added to the application which can be activated to insert test cases using ptrace[?]. The use of dynamic instrumentation capability to add test cases in an application is an extension of our previous work of a dynamic instrumentation tool iProbe [?]

The authors previous work in in-vivo testing[?] explored testing in the wild by initiating test cases in the production environment and sharing the load across several instances of deployed application. This approach adds test-cases in predetermined functions before starting the execution of the process, and periodically executes them in the run-time environment based on a probabilistic function.

[?]

1.1 Contributions

The impact of sandbox testing can be seen in several different ways

- **Sandbox Live Testing** One of the key motivations leading to *Parikshan* is to provide a harness to allow the user to test real, live implementations.
- **Fault Tolerance**
- **Verification**
- **Integration Testing**

2 Related Work

There have been several existing approaches that look into testing applications in the wild. The related work can be divided in several categories:

- **Perpetual Testing** We are inspired by the notion of perpetual testing[?] which advocate that software testing should be key part of the deployment phase and not just restricted to the development phase.
- **Record and Replay**
- **Alpha-Beta Testing**

3 Design

In this section we begin with a system overview of `Parikshan`. We then explain how it inserts test cases, into the test harness, and finally we explain how a user can use the `Parikshan` api to insert test cases in the test harness.

3.1 System Overview

4 Triggering and Inserting Test Cases

5 Network Issues

There are several problems that can effect the execution of sandbox testing.

- **Stateful Connections**
- **Time Lag**

6 Implementation

7 Conclusion

References