Technical Disclosure Commons

Defensive Publications Series

August 14, 2019

Reducing test runtimes by saving and restoring machine state

Vishal Sethia

Thomas Knych

Reducing test runtimes by saving and restoring machine state

ABSTRACT

Most tests of software for consumer electronic devices involve setup operations prior to actual execution of the test. The setup operations can be time-consuming and are often common across test nodes. This disclosure describes techniques that enable the setup operations to be performed once prior to the start of test execution. The results are shared across multiple test nodes. The techniques thereby reduce test runtimes, and increase the scale and volume of testing.

KEYWORDS

- Device testing
- Software testing
- Test configuration
- Test node
- Directed acyclic graph
- Virtual machine
- Machine state

BACKGROUND

Most tests of software for consumer electronic devices involve setup operations prior to the actual execution of the test. The setup operations can be time-consuming and are often common across test nodes.

DESCRIPTION

This disclosure describes techniques that enable the setup operations to be performed once prior to the start of test execution. The results are shared across multiple test nodes. The techniques thereby reduce test runtimes, and increase the scale and volume of testing.

A directed acyclic graph (DAG) that models machine configuration, setup requirements, test configuration, etc., is determined such that a given machine state is associated with a given machine configuration.

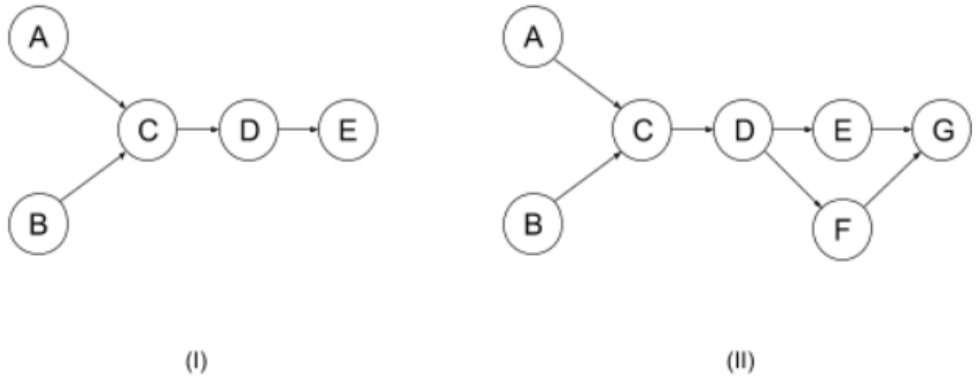

Fig. 1: Examples of directed acyclic graphs representing test nodes

Fig. 1 illustrates examples of directed acyclic graphs that represent test nodes. Each node in the example DAGs of Fig. 1 represents setup operations, e.g., install a certain app, create a user, store a certain file, etc. A node in the DAG can also represent virtual machine configurations. The order of nodes in the DAG determines the sequence of operations.

For example, in Fig. 1-(I), if node A represents virtual machine startup, node B represents virtual machine setup, and node C represents the installation of an app, the installation of the app (node C) can occur only after virtual machine startup (node-A) and setup (node-B) has been completed.

A test node comprises one or more test cases. An advantage of representing a test node as a DAG is that the dependencies of the test case on machine configuration and common setup requirements are clearly spelled out. Test cases comprising a test node are executed once the

machine configuration and common setup, as specified in its corresponding graph, are completed.

The DAG is used to identify test nodes that share machine configuration and setup requirements. For example, a set that includes multiple test nodes is described by the DAG of Fig. 1-(I), while a different set comprising other test nodes is described by the DAG of Fig. 1-(II). Upon identifying a set of test nodes that share a common DAG, virtual machines are instantiated with the configuration of the DAG, common setup operations are performed as specified in the DAG, the virtual machine state is persisted, and all test actions are executed in parallel. The test actions restore corresponding virtual machines and have an instance that matches their configuration requirements and has had all necessary setup actions already completed. The test actions only perform test execution, realizing an overall speed-up of test execution times, since common (or redundant) work of virtual machine startup and setup is performed only once for many test nodes.

In this manner, the techniques of this disclosure enable the identification of common requirements of machine configuration and test setup that are shared across multiple test nodes. In performing a multi-node test, such common work is done once, and test nodes are provided with an identical machine state that can be used immediately to execute respective test cases. The techniques thereby reduce test runtimes, and increase the scale and volume of testing, since redundant setup operations are executed only once per invocation rather than being repeated for every test node. The techniques also ensure that machine state is restored so long as the same configuration is used across different test nodes.

The described techniques can be utilized, e.g., for virtualization test systems that are used to test software execution on various mobile and wearable devices, e.g., a large matrix of devices

with different configurations, or for software testing of any type of application, e.g., cloud-based file storage, document editing, etc. The techniques can be offered as a cloud-based service for application developers and allow such developers to create scalable test infrastructure without having to write custom code. For example, such a service can include a specification for the developer to express machine configurations, common setup actions, and corresponding associations of test nodes.

CONCLUSION

Most tests of software for consumer electronic devices involve setup operations prior to actual execution of the test. The setup operations can be time-consuming and are often common across test nodes. This disclosure describes techniques that enable the setup operations to be performed once prior to the start of test execution. The results are shared across multiple test nodes. The techniques thereby reduce test runtimes, and increase the scale and volume of testing.

REFERENCES

[1] Krsul, Ivan, Arijit Ganguly, Jian Zhang, Jose AB Fortes, and Renato J. Figueiredo. "VMPlants: Providing and managing virtual machine execution environments for grid computing." In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, p. 7. IEEE Computer Society, 2004.

[2] https://skia.org/dev/testing/automated_testing, accessed on Feb. 11, 2019.