Killing Zombies – Generating Realistic Trace Files

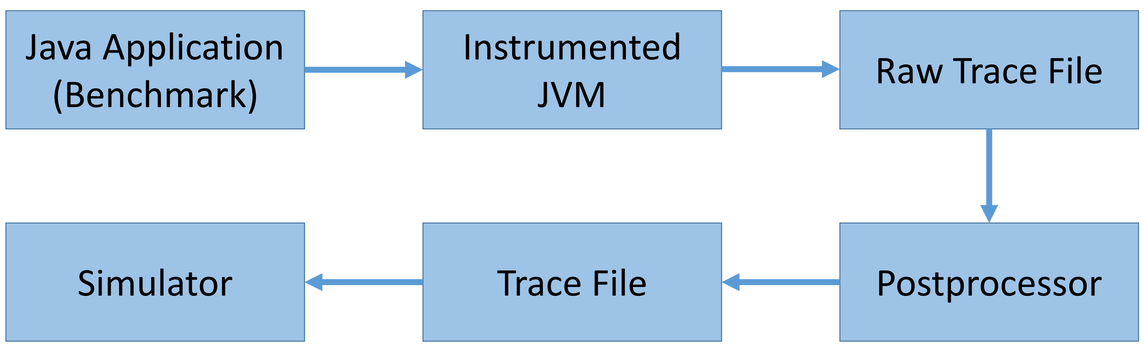
Johannes Ilisei, Kenneth B. Kent, Gerhard W. Dueck
University of New Brunswick, IBM Canada
Faculty of Computer Science
jilisei@unb.ca, ken@unb.ca, gdueck@unb.ca

Motivation

JVM implementations are large-sized projects that include ongoing improvements and additions of new features throughout many years. Working on these projects or experimenting with new algorithms can be a difficult and time consuming task.
Simulators are available that reproduce desired JVM operations. They can be used to implement and test new features in little time. Like a JVM, the simulator requires instructions in the form of input-files that contain operations. Trace files are generated with an instrumented JVM by capturing relevant operations. This project focuses on the generation of highly realistic trace files.

Project

The trace file generation process includes the following files and subsystems:



The Previous Approach

The previous trace file generator kept objects directly reachable from the roots as long as they were accessed. The last root set deletion was printed after the last access to an object. The result was an unrealistic root set representation.

The New Approach

In order to solve the root set problem, this project captures root operations directly on occurrence. The bytecode interpreter parses bytecode instructions and performs root additions and deletions.
The following table shows how many functions push and pop objects out of the total 334 interpreter-functions:

Push	Pop	Push and Pop
41	19	5

The Zombie Problem

The first test run revealed an unexpected problem: objects are accessed after they become garbage. We call them zombie objects. A very simple zombie example may look like this:

```
a T1 O42 S128 N3  //allocate Object 42
+ T1 O42          //add object 42 to the root set
- T1 O42          //remove object 42 from the root set
w T1 P40 #2 O42   //create a reference from object 40 to 42
```

After object 42 is removed from the roots, it is unreachable and therefore garbage. The write operation from object 40 to 42 may result into an error.

Reasons for Zombie Objects

Zombie objects exist because not all operations are captured during JVM execution.
These are the main reasons for zombie objects:

- Missing locking operations
- Immortal Objects
- Rootset operations outside the bytecode interpreter