Wikiprint Book

Title: jpf-racefinder

Subject: Java Path Finder - projects/jpf-racefinder

Version: 1

Date: 02/22/2013 06:50:44 AM

Table of Contents

jpf-racefinder **3**
Repository 3
How to install 3
How to use 3

jpf-racefinder

This is a jpf extension to detect a data race precisely to address the problem of Java relaxed memory model problem.

This work is done at University of Florida and the original homepage for jpf-racefinder (previously called Java RaceFinder) is here.

Repository

The Mercurial repository for jpf-racefiner is at http://babelfish.arc.nasa.gov/hg/jpf/jpf-racefinder.

How to install

1. Download and install jpf-core first. (You may need jpf-symbc for modular extension of jpf-racefinder.)
1. Add jpf-racefinder into your site.properties.
1. Apply CHANGES to jpf-core.

1) Modify jpf-core/src/main/gov/nasa/jpf/jvm/ThreadInfo.java as follows

```
// src/main/gov/nasa/jpf/jvm/ThreadInfo.java:
// at executeInstruction() method,
   if (logInstruction) {
     ss.recordExecutionStep(pc);
   }
   // By KyungHee
   else {
               String listener = JVM.getVM().getConfig().getProperty("listener");
               if ( listener != null && listener.contains("edu.ufl.cise.jrf.listener.JRFListener") )
                       ss.recordExecutionStep(pc);
   }
  // END By KyungHee
```

2) Copy jpf-file/*.java to jpf-core/src/classes/java/util/concurrent/atomic/.

```
AtomicIntegerArray.java
AtomicLongArray.java
AtomicReferenceArray.java
```

3) Rebuild jpf-core using ant.

1. Build jpf-racefinder using ant

How to use

The sample configuration file for jpf-racefinder is in *jrf.jpf*. The UNIX shell script to launch jpf-racefinder is in *bin/jrf*. To test, run

```
bin/jrf simple.SimpleRace
```

This will report two races as follows,

```
====================================================== system under test
application: simple/SimpleRace.java

====================================================== search started: 10/17/10 4:17 PM



JRF results

====================================================== data race #1
```

```
edu.ufl.cise.jrf.util.HBDataRaceException
        at THREAD      (simple.SimpleRace$Thread2@ from "Thread t1 = new Thread2();" at simple/SimpleRace.java:26 in (main
        to MEMORY      (simple.SimpleRace.x from [CLASS_LOADER])
        in INSTRUCTION (getstatic)
        of SOURCE      ("assert (x==1);" at simple/SimpleRace.java:45)

_____________________________________________________ analyze counter example
data race source statement : "putstatic" at simple/SimpleRace.java:35  : "x = 1;" by thread 1
data race manifest statement : "getstatic" at simple/SimpleRace.java:45: "assert (x==1);" by thread 2

Change the field "simple.SimpleRace.x from [CLASS_LOADER]" to volatile.
Change the field "simple.SimpleRace.done from [CLASS_LOADER]" to volatile.

_____________________________________________________ advice from acquiring history

===================================================== data race #2
edu.ufl.cise.jrf.util.HBDataRaceException
        at THREAD      (simple.SimpleRace$Thread2@ from "Thread t1 = new Thread2();" at simple/SimpleRace.java:26 in (main
        to MEMORY      (simple.SimpleRace.done from [CLASS_LOADER])
        in INSTRUCTION (getstatic)
        of SOURCE      ("while(!done) { /*spin*/ }" at simple/SimpleRace.java:44)

_____________________________________________________ analyze counter example
data race source statement : "putstatic" at simple/SimpleRace.java:36  : "done = true;" by thread 1
data race manifest statement : "getstatic" at simple/SimpleRace.java:44: "while(!done) { /*spin*/ }" by thread 2

Change the field "simple.SimpleRace.done from [CLASS_LOADER]" to volatile.

_____________________________________________________ advice from acquiring history

_____________________________________________________ frequency of advice
[2times] Change the field "simple.SimpleRace.done from [CLASS_LOADER]" to volatile.
[1times] Change the field "simple.SimpleRace.x from [CLASS_LOADER]" to volatile.

_____________________________________________________ statistic
JRF explored "0" more states through HB abstraction out of "12".
Total HB entries = 29*201
JRF takes 1sec to find 2 equivalent races with 4 counterexample traces.
The maximum counter example path length is "6".
JRF-E takes 0sec in 4 races analysis.

===================================================== results
no errors detected

===================================================== statistics
elapsed time:       0:00:01
states:             new=9, visited=6, backtracked=14, end=2
search:             maxDepth=5, constraints=0
choice generators:  thread=9, data=0
heap:               gc=7, new=302, free=35
instructions:       3242
max memory:         81MB
loaded code:        classes=75, methods=1040

===================================================== search finished: 10/17/10 4:17 PM
```

To detect data race in your program (`MyExample.java` under src/examples), you just need to change "`simple.SimpleRace`" into "`MyExample`".