

Java Concurrency Stress tests

jcstress

Виталий Бибаев

Java Memory Model

- JSR 133 (2004 год). Реализация в Java SE 5
- Описывает необходимые и достаточные условия, при которых изменения сделанные в одном потоке видны в других
- Вводит абстракцию happens-before
 - $X \text{ hb } Y \Rightarrow$ Весь код после Y видит все изменения до X
- Примеры happens-before
 - Запуск потока happens-before код в потоке
 - Запись в final поля happens-before все что после конструктора
 - И многие другие

jcstress

- Разработчик – Алексей Шипилев
- Состояние – в разработке, причем API может меняться
- Можно разделить на две части:
 - Тестовый фреймворк
 - Тесты
- 2 сценария использования:
 - Запуск своих тестов
 - Запуск всех тестов для текущей конфигурации
- В планах написать очень много тестов на примитивы синхронизации и коллекции

jcstress

- Экспериментальное средство для исследования корректности работы многопоточного кода в JVM
- Используется для тестирования JVM, библиотеки, железа
- Особенности:
 - Тесты вероятностные
 - Тесты нужно запускать на процессоре хотя бы с двумя CPU
- API – набор аннотаций
- Чтобы собрать требуется JDK 9

Пример 1

```
57 @JCStressTest
58
59 // These are the test outcomes.
60 @Outcome(id = "1, 1", expect = Expect.ACCEPTABLE_INTERESTING, desc = "Both actors came up " +
61 "with the same value: atomicity failure.")
62 @Outcome(id = "1, 2", expect = Expect.ACCEPTABLE, desc = "actor1 incremented, then actor2.")
63 @Outcome(id = "2, 1", expect = Expect.ACCEPTABLE, desc = "actor2 incremented, then actor1.")
64
65 // This is a state object
66 @State
67 public class SampleOne {
68     int v;
69     @Actor
70     public void actor1(IntResult2 r) { r.r1 = ++v; }
71
72
73
74     @Actor
75     public void actor2(IntResult2 r) { r.r2 = ++v; }
76 }
77
78
79
```

Пример 1. Результат

[OK] jcteststest.SampleOne

(JVM args: [-Dfile.encoding=windows-1251, -server])

| Observed state | Occurrences | Expectation | Interpretation |
|----------------|-------------|------------------------|---|
| 1, 1 | 1 276 254 | ACCEPTABLE_INTERESTING | Both actors came up with the same value: atomicity failure. |
| 1, 2 | 65 895 017 | ACCEPTABLE | actor1 incremented, then actor2. |
| 2, 1 | 33 603 849 | ACCEPTABLE | actor2 incremented, then actor1. |

[OK] jcteststest.SampleOne

(JVM args: [-Dfile.encoding=windows-1251, -client])

| Observed state | Occurrences | Expectation | Interpretation |
|----------------|-------------|------------------------|---|
| 1, 1 | 1 552 309 | ACCEPTABLE_INTERESTING | Both actors came up with the same value: atomicity failure. |
| 1, 2 | 63 581 434 | ACCEPTABLE | actor1 incremented, then actor2. |
| 2, 1 | 39 557 587 | ACCEPTABLE | actor2 incremented, then actor1. |

[OK] jcteststest.SampleOne

(JVM args: [-Dfile.encoding=windows-1251, -server, -XX:-TieredCompilation])

| Observed state | Occurrences | Expectation | Interpretation |
|----------------|-------------|------------------------|---|
| 1, 1 | 1 973 833 | ACCEPTABLE_INTERESTING | Both actors came up with the same value: atomicity failure. |
| 1, 2 | 66 926 143 | ACCEPTABLE | actor1 incremented, then actor2. |
| 2, 1 | 39 815 104 | ACCEPTABLE | actor2 incremented, then actor1. |

API

- State – разделяемое состояние
- Actor – код который, выполняется в отдельном потоке (воздействует на State)
- Arbiter – код который выполняется после завершения всех Actor методов
- Outcome – описание ожидаемого результата
- Result – объект для сохранения результата (IntResult1, ByteResult2)
- JCTestTest(Mode) – маркер, что класс является тестом
- Signal – вызов метода, который должен прервать работу @Actor
- + Вспомогательные аннотации для документации

API

- Expect
 - ACCEPTABLE
 - ACCEPTABLE_INTERESTING
 - FORBIDDEN
 - UNKNOWN
- Mode
 - Continuous
 - Termination

Пример 2. Mode = Termination

```
62 @JCStressTest(Mode.Termination)
63 @Outcome(id = "TERMINATED", expect = Expect.ACCEPTABLE, desc = "Gracefully finished.")
64 @Outcome(id = "STALE", expect = Expect.FORBIDDEN, desc = "Test hung up.")
65
66 @State
67 public class Termination {
68     int v;
69
70     @Actor
71     public void actor1() {
72         while (v == 0) {
73             // spin
74         }
75     }
76     @Signal
77     public void signal() { v = 1; }
78
79 }
```

Пример 2. Результат

[FAILED] jcteststress.Termination

(JVM args: [-Dfile.encoding=windows-1251, -server, -XX:+UnlockDiagnosticVMOptions, -XX:+StressLCM, -XX:+StressGCM])

| Observed state | Occurrences | Expectation | Interpretation |
|----------------|-------------|-------------|----------------------|
| STALE | 1 | FORBIDDEN | Test hung up. |
| TERMINATED | 2 | ACCEPTABLE | Gracefully finished. |

Messages:

Have stale threads, forcing VM to exit for proper cleanup.

[FAILED] jcteststress.Termination

(JVM args: [-Dfile.encoding=windows-1251, -server, -XX:-TieredCompilation, -XX:+UnlockDiagnosticVMOptions, -XX:+StressLCM, -XX:+StressGCM])

| Observed state | Occurrences | Expectation | Interpretation |
|----------------|-------------|-------------|----------------------|
| STALE | 1 | FORBIDDEN | Test hung up. |
| TERMINATED | 0 | ACCEPTABLE | Gracefully finished. |

Messages:

Have stale threads, forcing VM to exit for proper cleanup.

Резюме

- Результаты зависят от железа
- Результаты зависят от JDK (версии/сборки)
- Тесты вероятностные, поэтому нужно запускать многократно
- Запуск имеющихся тестов = проверка JDK/железа
- Запуск своих тестов = тестирование своего кода
- Можно использовать как справочник по JMM

Полезные ссылки

- JSR 133: <https://jcp.org/en/jsr/detail?id=133>
- Описание jcstress: <https://wiki.openjdk.java.net/display/CodeTools/jcstress>
- Исходники jcstress: <http://hg.openjdk.java.net/code-tools/jcstress/>
- Видео OpenJDK TestFest Russia 2013 <https://www.youtube.com/watch?v=4p4vL6EhzOk>
- Примеры тестов: <http://hg.openjdk.java.net/code-tools/jcstress/file/tip/jcstress-samples/src/main/java/org/openjdk/jcstress/samples>
- Roadmap: <http://hg.openjdk.java.net/code-tools/jcstress/file/04bd0247edbb/ROADMAP>

Спасибо за внимание

Эта презентация: <https://goo.gl/aPQpBz>