race conditions
“Race conditions arise in software when separate computer processes or threads of execution depend on some shared state. Operations upon shared states are critical sections that must be mutually exclusive. Failure to obey this rule opens up the possibility of corrupting the shared state.”

Wikipedia.org
race conditions

\[
\begin{align*}
    a &= 0; \\
    b &= 3; \\
    c &= a + b; \\
    a &= 7; \\
    b &= 5; \\
    d &= a - b; \\
\end{align*}
\]
“Race conditions have a reputation as difficult to reproduce and debug, since the end result is non-deterministic, and highly dependent on the relative timing between interfering threads. Problems occurring in production systems can therefore disappear when running in debug mode, when additional logging is added, or when attaching a debugger [...]. It is therefore better to avoid race conditions by careful software design rather than attempting to fix them afterwards.”, Wikipedia
race conditions

\[ \text{synchronized (lock)} \]

- \( a = 0; \)
- \( b = 3; \)
- \( c = a + b; \)

\[ \text{synchronized (lock)} \]

- \( a = 7; \)
- \( b = 5; \)
- \( d = a - b; \)
Flavia Rainone - Principal Software Engineer
investigating race conditions...

Where?
investigating race conditions

• study and verify the code

\[
\begin{align*}
  a &= 0; \\
  b &= 3; \\
  c &= a+b; \\
  a &= 7; \\
  b &= 5; \\
  d &= a-b;
\end{align*}
\]
investigating race conditions

- write simple tests
  1. check for good coverage
     
     $ mvn cobertura: cobertura
  
  2. run with many threads
  
  3. run repeatedly
tests with multiple threads

```java
public class MultiThreadedNioTcpChannelTestCase extends NioTcpChannelTestCase {

    @Before
    public void setThreads() {
        super.setNumberOfThreads(5);
    }
}

public class MultiThreadedNioTcpChannelTestCase2 extends NioTcpChannelTestCase {

    private static final MAX_THREADS = 10;

    @Before
    public void setThreads() {
        int randomNumber = (new Random()).nextInt(10);
        return randomNumber < 2: 2: randomNumber;
    }
}
```
avoiding errors

HOW TO FIND THEM?

Image courtesy of David Castillo Dominici / FreeDigitalPhotos.net
Byteman is a tool which simplifies tracing and testing of Java programs. Byteman allows you to insert extra Java code into your application without having to rewrite or recompile anything. The code can be loaded either during JVM startup or even after your application has already started running.
byteman

- TestCase + Rules
- JUnit or TestNG
- @BMScript
- @BMRules
- @BMRule
- Helper
byteman - example

code example:

```java
package myPackage;

@RunWith(BMUnitRunner.class)
@BMScript(dir= "src/test/resources")
public class MyTestCase {

    @Test
    public void test() {}
}
```

```btm
RULE hello world
CLASS myPackage.MyTestCase
METHOD test
AT ENTRY
IF true
DO
    debug("Hello world!"
ENDRULE
```
byteman - example

```xml
<dependency>
    <groupId>org.jboss.byteman</groupId>
    <artifactId>byteman</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.jboss.byteman</groupId>
    <artifactId>byteman-bmunit</artifactId>
    <scope>test</scope>
</dependency>
```
Results:
Failed tests:
test(org.test.racecondition.MyTestCase)
Tests run: 675, Failures: 1, Errors: 0, Skipped: 0

[INFO]-------------------------
[INFO] BUILD FAILURE
[INFO]-------------------------
public void setMode(final Mode newMode) {
    ...  
    final ArrayList<Runnable> tasks = new ArrayList<Runnable>(4);  
    synchronized (this) {
        ...  
        internalSetMode(newMode, tasks);  
        ...  
    }
    doExecute(tasks);  
}

private void internalSetMode(Mode newMode, ArrayList<Runnable> taskList) {
    final Mode oldMode = mode;  
    if (oldMode == Mode.REMOVE) {
        if (state.compareTo(Substate.REMOVING) >= 0)  
            throw new IllegalStateException("Service already removed");  
        ...  
    }
    ...  
    mode = newMode;  
}
void transition(final ArrayList<Runnable> tasks) {
    Transition transition;
    do {
        ...
        transition = getTransition();
        if (transition == null) {
            return;
        }
        switch (transition) {
            ...
            case STOPPING_to_DOWN: ...
            case DOWN_to_REMOVING: ... // call RemoveTask() and other stuff
            case REMOVING_to_REMOVED: ...
            ...
            state = transition.getAfter();
        } while (tasks.isEmpty());
        notifyAll(); // notify waiters that a transition occurred
    }
private static final ServiceName serviceName = ServiceName.of("service");

@Test
public void test() throws Exception {
    final Future<ServiceController<?>> serviceStart = ...;
    final ServiceController<?> serviceController = serviceContainer.
        addService(serviceName, Service.NULL).install();
    assertController(serviceController, serviceStart);
    serviceController.setMode(Mode.REMOVE);
    boolean failed = false;
    try {
        serviceController.setMode(Mode.ACTIVE);
    } catch (IllegalStateException e) {
        failed = true;
    }
    assertTrue(failed);
}
example 1 – race condition

```java
service.setMode(Mode.REMOVE)

service.setMode(Mode.ACTIVE)
```

service removal
example 1 - rules

```java
RULE setMode(Mode.ACTIVE) to removing service
CLASS
METHOD
AT ENTRY
IF
DO

ENDRULE

RULE service REMOVING
CLASS
METHOD
AT EXIT
IF
DO

ENDRULE
```
example 1 - rules

RULE setMode(Mode.ACTIVE) to removing service
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD setMode
AT ENTRY
IF true
DO

ENDRULE

RULE service REMOVING
CLASS
METHOD
AT EXIT
IF
DO

ENDRULE

src/test/resources/org/jboss/msc/test/SetActiveOnRemovingServiceTestCase.btm
example 1 - rules

```java
RULE setMode(Mode.ACTIVE) to removing service
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD setMode
AT ENTRY
IF $1 == org.jboss.msc.service.ServiceController$Mode.ACTIVE
DO

ENDRULE

RULE service REMOVING
CLASS
METHOD
AT EXIT
IF
DO

ENDRULE
```

`$0, $1, $2,...`

`$0: this`

`$1, $2,...: method parameters`
example 1 - rules

```java
RULE setMode(Mode.ACTIVE) to removing service
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD setMode
AT ENTRY
IF $1 == org.jboss.msc.service.ServiceController$Mode.ACTIVE
DO

ENDRULE

RULE service REMOVING
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD transition
AT EXIT
IF $0.state.toString().equals("REMOVING")
DO

ENDRULE
```

src/test/resources/org/jboss/msc/test/SetActiveOnRemovingServiceTestCase.btm
example 1 - rules

```java
RULE setMode(Mode.ACTIVE) to removing service
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD setMode
AT ENTRY
IF $1 == org.jboss.msc.service.ServiceController$Mode.ACTIVE
DO
  # hold setMode until service is in REMOVING state
  waitFor("service REMOVING")
ENDRULE

RULE service REMOVING
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD transition
AT EXIT
IF $0.state.toString().equals("REMOVING")
DO
  # after service B enters REMOVING, wake setMode(Mode.ACTIVE)
  signalWake("service REMOVING", true)
ENDRULE
```
example 1 - rules

```java
RULE setMode(Mode.ACTIVE) to removing service
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD setMode
AT ENTRY
IF $1 == org.jboss.msc.service.ServiceController$Mode.ACTIVE
DO
    # hold setMode until service is in REMOVING state
    debug("wait for service enter REMOVING");
    waitFor("service REMOVING");
    debug("proceed with setMode(Mode.ACTIVE)")
ENDRULE

RULE service REMOVING
CLASS org.jboss.msc.service.ServiceControllerImpl
METHOD transition
AT EXIT
IF $0.state.toString().equals("REMOVING")
DO
    # after service B enters REMOVING, wake setMode(Mode.ACTIVE)
    debug("signalling setMode(Mode.ACTIVE)");
    signalWake("service REMOVING", true);
    debug("signalled setMode(Mode.ACTIVE)")
ENDRULE
```
**synchronization API**

public void waitFor(Object identifier)
public void waitFor(Object identifier, long millisecsWait)
public boolean waiting(Object identifier)
public boolean signalWake(Object identifier)
public boolean signalWake(Object identifier, boolean mustMeet)
public boolean signalThrow(Object identifier)
public boolean signalThrow(Object identifier, boolean mustMeet)

org.jboss.byteman.rule.helper.Helper
private volatile int totalConnections;

public NioTcpChannel accept() throws IOException {
    if (isConnectionFull())
        return null;
    ++totalConnections;
    if (totalConnections >= getHighWaterMark()) {
        suspendAccepts();
        setConnectionFull(true);
    }
    ...
}

private void undoAccept() {
    totalConnections--;
    if (isConnectionFull()) {
        setConnectionFull(false);
        Thread thread = waitingThread;
        waitingThread = null;
        if (thread != null)
            unpark(thread);
        resumeAccepts();
    }
}
public void awaitAcceptable() throws IOException {
    final Thread thread = Thread.currentThread();
    for (;;) {
        while (isConnectionFull()) {
            final Thread nextThread = waitingThread;
            waitingThread = thread;
            try {
                // if it's still set after we've registered...
                if (isConnectionFull())
                    park(this);
            } finally {
                if (nextThread != null)
                    unpark(nextThread);
            }
            if (thread.isInterrupted())
                throw new InterruptedIOException();
        }
        ...  
        If (!isConnectionFull())
            return;
    }
}
private static class AcceptableAwaiter implements Runnable {

    private AcceptingChannel<? extends ConnectedStreamChannel> acceptingChannel;

    public AcceptableAwaiter(AcceptingChannel<? extends ConnectedStreamChannel> c) {
        acceptingChannel = c;
    }

    public void run() {
        try {
            acceptingChannel.awaitAcceptable();
        } catch (IOException e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    }
}
@Test
public void test() throws Exception {
    AcceptingChannel<? extends ConnectedStreamChannel> server = ...;
    while (true) {
        SocketChannel channel = SocketChannel.open();
        channel.socket().bind(new InetSocketAddress(...));
        channel.connect(new InetSocketAddress(...), SERVER_PORT));
        ConnectedStreamChannel connectedChannel = server.accept();
        if (connectedChannel == null)
            break;
        if (!channel.finishConnect())
            break;
        channels.add(connectedChannel);
    }
    final Thread acceptableWaiterThread = new Thread(
        new AcceptableWaiter(server));
    acceptableWaiterThread.start();
    for (ConnectedStreamChannel channel: channels)
        channel.close();
    acceptableWaiterThread.join();
}
example 2 – race condition

create connections until isConnectionFull()

NioTcpServer.awaitAcceptable()

close connections
example 2 – race conditions

create connections until isConnectionFull()

NioTcpServer.awaitAcceptable()

isConnectionFull() isConnectionFull()
RULE awaitAcceptable checks connection is full
CLASS org.xnio.nio.NioTcpServer
METHOD awaitAcceptable
?
IF true
DO
  debug("awaitAcceptable checked that connection is full"),
  signalWake("connectionFull", true)
ENDRULE

RULE test close channels
CLASS org.xnio.nio.test.UnsetConnFullDuringAwaitTestCase
METHOD test
?
IF true
DO
  debug("waiting for awaitAcceptable"),
  waitFor("connectionFull"),
  debug("proceeding with closing channels")
ENDRULE
RULE awaitAcceptable checks connection is full
CLASS org.xnio.nio.NioTcpServer
METHOD awaitAcceptable
AFTER INVOKE isConnectionFull
IF true
DO
  debug("awaitAcceptable checked that connection is full"),
  signalWake("connectionFull", true)
ENDRULE

RULE test close channels
CLASS org.xnio.nio.test.UnsetConnFullDuringAwaitTestCase
METHOD test
?
IF true
DO
  debug("waiting for awaitAcceptable"),
  waitFor("connectionFull"),
  debug("proceeding with closing channels")
ENDRULE
public void awaitAcceptable() throws IOException {
    final Thread thread = Thread.currentThread();
    for (;;) {
        while (isConnectionFull()) {
            final Thread nextThread = waitingThread;
            WaitingThread = thread;
            try {
                // if it's still set after we've registered...
                if (isConnectionFull())
                    park(this);
            } finally {
                if (nextThread != null)
                    unpark(nextThread);
            }
        } finally {
            if (nextThread != null)
                unpark(nextThread);
        }
        if (thread.isInterrupted())
            throw new InterruptedIOException();
    }
    ... 
    if (isConnectionFull())
        return;
}
RULE awaitAcceptable checks connection is full
CLASS org.xnio.nio.NioTcpServer
METHOD awaitAcceptable
AFTER INVOKE isConnectionFull 1
IF true
DO
    debug("awaitAcceptable checked that connection is full"),
    signalWake("connectionFull", true)
ENDRULE

RULE test close channels
CLASS org.xnio.nio.test.UnsetConnFullWhileAwaitTestCase
METHOD test

? IF true
DO
    debug("waiting for awaitAcceptable"),
    waitFor("connectionFull"),
    debug("proceeding with closing channels")
ENDRULE
RULE awaitAcceptable checks connection is full
CLASS org.xnio.nio.NioTcpServer
METHOD awaitAcceptable
AFTER INVOKE isConnectionFull 1
IF true
DO
  debug("awaitAcceptable checked that connection is full"),
  signalWake("connectionFull", true)
ENDRULE

RULE test close channels
CLASS org.xnio.nio.test.UnsetConnFullDuringAwaitTestTestCase
METHOD test
AT INVOKE org.xnio.channels.ConnectedStreamChannel.close
IF true
DO
  debug("waiting for awaitAcceptable"),
  waitFor("connectionFull"),
  debug("proceeding with closing channels")
ENDRULE
```
@Test
public void test() throws Exception {
    AcceptingChannel<? extends ConnectedStreamChannel> server = ...;
    while (true) {
        SocketChannel channel = SocketChannel.open();
        channel.socket().bind(new InetSocketAddress(...));
        channel.connect(new InetSocketAddress(...), SERVER_PORT));
        ConnectedStreamChannel connectedChannel = server.accept();
        if (connectedChannel == null)
            break;
        if (!channel.finishConnect())
            break;
        channels.add(connectedChannel);
    }
    final Thread acceptableWaiterThread = new Thread(
        new AcceptableWaiter(server));
    acceptableWaiterThread.start();
    for (ConnectedStreamChannel channel : channels)
        channel.close();
    acceptableWaiterThread.join();
```
RULE awaitAcceptable checks connection is full
CLASS org.xnio.nio.NioTcpServer
METHOD awaitAcceptable
AFTER INVOKE isConnectionFull 1
IF true
DO
  debug("awaitAcceptable checked that connection is full"),
  signalWake("connectionFull", true)
ENDRULE

RULE test close channels
CLASS org.xnio.nio.test.UnsetConnFullDuringAwaitTestCas
METHOD test
AT INVOKE org.xnio.channels.ConnectedStreamChannel.close
IF incrementCounter("run 'test close channels' once") == 1
DO
  debug("waiting for awaitAcceptable"),
  waitFor("connectionFull"),
  debug("proceeding with closing channels")
ENDRULE
public boolean createCounter(Object o)
public boolean createCounter(Object o, int count)
public boolean deleteCounter(Object o)
public int incrementCounter(Object o, int amount)
public int incrementCounter(Object o)
public int decrementCounter(Object o)
public int readCounter(Object o)
public int readCounter(Object o, boolean zero)
example 2 - rules

```
RULE clearCounters
CLASS org.xnio.nio.test.UnsetConnFullDuringAwaitTestCase
METHOD test
AT EXIT
IF TRUE
DO
    deleteCounter("run 'test close channels' once")
ENDRULE
```
private volatile int resourceCount = 0;

void resourceUntick(Object opened) throws NotOpenException {
    int old = resourceCount ++;
}

private void closeTick1() {
    int res = -- resourceCount;
}

private void finishPhase1() {
    // all our original resources were closed; now move on to stage two
    worker.shutdown();
    return;
}
private volatile int resourceCount = 0;

void resourceUntick(Object opened) throws NotOpenException {
    int old = resourceCount ++;
    if ((old & CLOSED_FLAG) != 0)
        throw new NotOpenException("Endpoint is not open");
}

private void closeTick1() {
    int res = -- resourceCount;
    if (res == CLOSED_FLAG) {
        // this was the last phase 1 resource.
        finishPhase1();
    } else 
        // else log a few messages
        ...
}

private void finishPhase1() {
    // all our original resources were closed; now move on to stage two
    worker.shutdown();
    return;
}
example 3

```java
private IoFuture<Connection> doConnect(...) throws IOException {
    boolean ok = false;
    resourceUntick("Connection to " + destination);
    try {
        final ConnectionProvider connectionProvider = ...;
        ...
        connections.add(connection);
        return futureResult.getIoFuture();
    } finally { if (! ok) closeTick1("a failed connection (1)"};
}

protected void closeAction() throws IOException {

    // Commence phase one shutdown actions
    int res = resourceCount;
    resourceCount = res | CLOSED_FLAG;
    if (res == 0)
        finishPhase1();
    else for (Object connection : connections.toArray())
        ((ConnectionImpl)connection).closeAsync();
}
```
private IoFuture<Connection> doConnect(...) throws IOException {
    synchronized (connectionLock) {
        boolean ok = false;
        resourceUntick("Connection to " + destination);
        try {
            final ConnectionProvider connectionProvider = ...;
            ...
            connections.add(connection);
            return futureResult.getIoFuture();
        } finally { if (! ok) closeTick1("a failed connection (1)"}; }
    }
}

protected void closeAction() throws IOException {
    synchronized (connectionLock) {
        // Commence phase one shutdown actions
        int res = resourceCount;
        resourceCount = res | CLOSED_FLAG;
        if (res == 0)
            finishPhase1();
        else for (Object connection : connections.toArray())
            ((ConnectionImpl)connection).closeAsync();
    }
}
private static class Connect implements Runnable {

    private boolean failed = false;
    private final Endpoint endpoint;

    public Connect(Endpoint endpoint) {
        this.endpoint = endpoint;
    }

    public void run() {
        final IoFuture<Connection> futureConnection;
        try {
            futureConnection = endpoint.connect(...);
            if (futureConnection != null)
                futureConnection.get();
        } catch (CancellationException e) {
            failed = true;
        }
    }

    public boolean hasFailed() {
        return failed;
    }
}
private static class CloseEndpoint implements Runnable {
    private final Endpoint endpoint;

    public CloseEndpoint(Endpoint endpoint) {
        this.endpoint = endpoint;
    }

    public void run() {
        endpoint.close();
        endpoint.awaitClosed();
    }
}
protected static Endpoint endpoint;

@Test
public void test() throws Exception {
  // create endpoint, auth provider, etc, create server
  endpoint = Remoting.createEndpoint("test", OptionMap.EMPTY);
  // create connect and close endpoint threads
  Connect connectRunnable = new Connect(endpoint);
  Thread connectThread = new Thread(connectRunnable);
  Thread closeEndpointThread = new Thread(new CloseEndpoint(endpoint));
  // execute and run threads
  connectThread.start();
  closeEndpointThread.start();
  connectThread.join();
  closeEndpointThread.join();
  // race condition means connect runnable may not fail
  assertTrue(connectRunnable.hasFailed());
}
example 3 – race condition

Thread 1: endpoint.close()

- If (resourceCount > 0)
  - true: closeAsync() on all connections;
  - false: finish close()

Thread 2: endpoint.connect(...)

- resourceUntick
- add connection to connections

If (resourceCount > 0)

Thread 1: endpoint.close()

- true: closeAsync() on all connections;
- false: finish close()
example 3 - rules

RULE Endpoint.closeAction
CLASS org.jboss.remoting3.EndpointImpl
METHOD closeAction
AT ENTRY
IF TRUE
DO
    waitFor("resourceUntick"),
ENDRULE

RULE doConnect calls resourceUntick
CLASS org.jboss.remoting3.EndpointImpl
METHOD doConnect
AFTER INVOKE org.jboss.remoting3.EndpointImpl.resourceUntick
IF TRUE
DO
    signalWake("resourceUntick", true)
ENDRULE
example 3 - rules

RULE Endpoint.closeAction
CLASS org.jboss.remoting3.EndpointImpl
METHOD closeAction
AT ENTRY
IF TRUE
DO
  waitFor("resourceUntick"),
ENDRULE

RULE doConnect calls resourceUntick
CLASS org.jboss.remoting3.EndpointImpl
METHOD doConnect
AFTER INVOKE org.jboss.remoting3.EndpointImpl.resourceUntick
IF TRUE
DO
  signalWake("resourceUntick", true),
  waitFor("connections to Array")
ENDRULE

RULE closeAction invokes Set.toArray
CLASS org.jboss.remoting3.EndpointImpl
METHOD closeAction
AFTER INVOKE java.util.Set.toArray
IF TRUE
DO
  signalWake("connections to Array", true)
ENDRULE
example 3 - rules

RULE Endpoint.closeAction
CLASS org.jboss.remoting3.EndpointImpl
METHOD closeAction
AT ENTRY
IF TRUE
DO
  waitFor("resourceUntick"),
ENDRULE

RULE doConnect calls resourceUntick
CLASS org.jboss.remoting3.EndpointImpl
METHOD doConnect
AFTER INVOKE org.jboss.remoting3.EndpointImpl.resourceUntick
IF TRUE
DO
  signalWake("resourceUntick", true),
  waitFor("connections to Array", 100)
ENDRULE

RULE closeAction invokes Set.toArray
CLASS org.jboss.remoting3.EndpointImpl
METHOD closeAction
AFTER INVOKE java.util.Set.toArray
IF TRUE
DO
  signalWake("connections to Array", false)
ENDRULE
Results:

Failed tests:
  test(org.test.racecondition.MyTestCase)

Tests run: 675, Failures: 1, Errors: 0, Skipped: 0

[INFO]-------------------------
[INFO] BUILD FAILURE
[INFO]-------------------------
to encourage...

GOOD PRACTICES

Flavia Rainone - Principal Software Engineer

Image courtesy of David Castillo Dominici / FreeDigitalPhotos.net
good practices

• Is Byteman running?

@RunWith(BMUnitRunner.class)
public class AbstractRaceConditionTest {

    private static final String RULE = "$0.instrumented = true";
    private boolean instrumented = false;

    @Test
    @BMRule(name="set instrumented to true",
        TargetClass = "AbstractRaceConditionTest",
        TargetMethod ="isInstrumented",
        Condition ="TRUE",
        Action = RULE)

    public void isInstrumented() {
        assertTrue(instrumented);
    }
}

Flavia Rainone - Principal Software Engineer
good practices

• verify whether the error exists and whether you understand the race condition
  – test first
  – fix the bug after!

• traditional test coverage
  – it is vital to have a good test regime in place in order to be able to identify race conditions
good practices

• Is my rule correct?

$ bmcheck [-cp classpath]* [-p package]* script1 [... scriptN]

• Is my rule being injected or executed?
  – -Dorg.jboss.byteman.verbose
  – -Dorg.jboss.byteman.debug
  – -Dorg.jboss.byteman.contrib.bmunit.verbose

• Can I use a debugger?
  – of course!
useful links

- Byteman home page
  www.jboss.org/byteman
- Repositories of the projects mentioned in the examples:
  github.com/xnio/xnio
  github.com/jboss-remoting/jboss-remoting
  github.com/jbossas/jboss-msc/