The Many JVMs of jtreg

Jonathan Gibbons

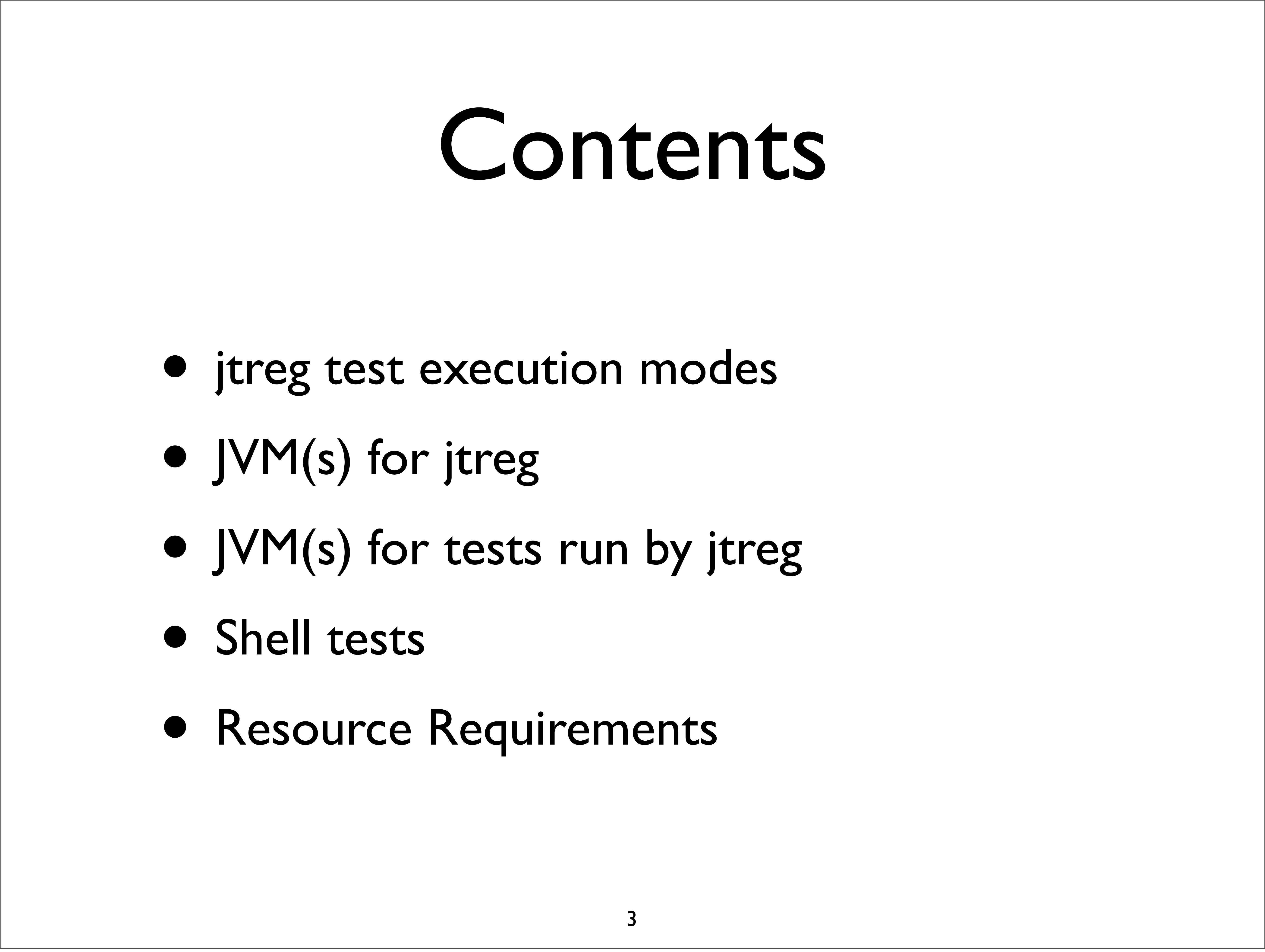
Purpose

- To explain the various JVMs created in the course of a jtreg test run
- To explain which options affect which JVMs

jtreg is the OpenJDK Regression Test Framework
See: http://openjdk.java.net/jtreg

Testing JDK is more than just testing the Java API.
It requires being able to test the various tools available on the JDK platform, and being able to test JDK under various extreme conditions. It is also desirable to be able to run tests “fast”.

All together, that implies the need to create and use a number of JVMs in the course of a test run, and any time you work with multiple JVMs, it becomes a potential source of confusion as to how to configure the various JVMs.



Contents

- jtreg test execution modes
- JVM(s) for jtreg
- JVM(s) for tests run by jtreg
- Shell tests
- Resource Requirements

We'll review the standard test execution modes supported by jtreg, and then go on to examine the JVMs required to run jtreg itself, and the tests.

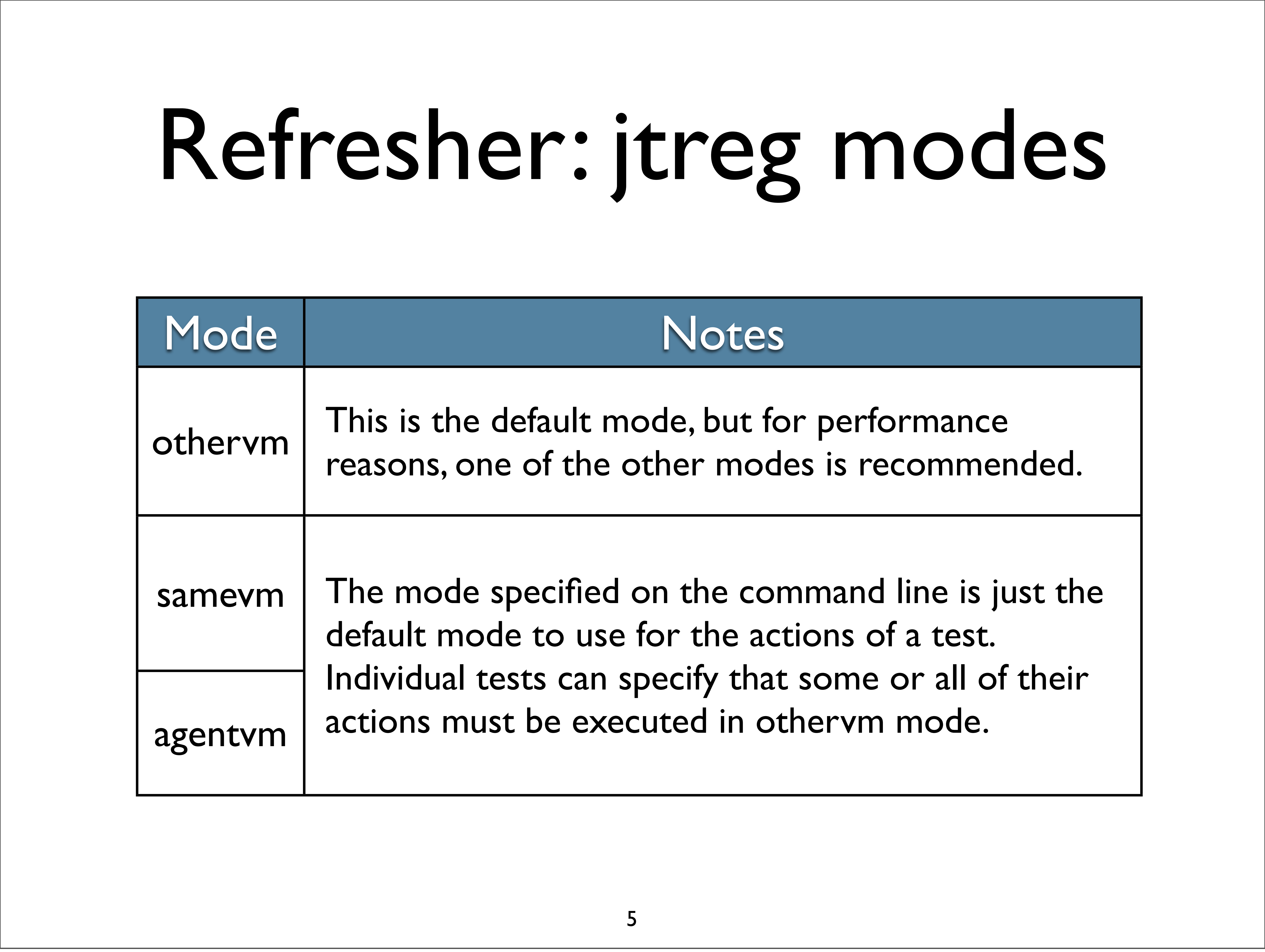
Shell tests provide for extensible test behavior, but create their own set of problems.

Finally, we'll look at the resource requirements for all these JVMs.

Refresher: jtreg modes

Mode	Description
othervm	Each action of each test in run in its own JVM Maximum isolation ✓ Maximum cost ✗
samevm	Most actions of most tests are run in the same JVM Minimum isolation ✗ Minimum cost ✓
agentvm	Hybrid: reuse JVMs when possible Generally good isolation ✓ Generally low cost ✓

These modes are described in more detail in documents on the OpenJDK website, wiki. See the references at the end of these slides.



Refresher: jtreg modes

Mode	Notes
othervm	This is the default mode, but for performance reasons, one of the other modes is recommended.
samevm	The mode specified on the command line is just the default mode to use for the actions of a test. Individual tests can specify that some or all of their actions must be executed in othervm mode.
agentvm	

Othervm mode is the default mode, but it is also the slowest, because each Java action (@run applet, @run compile, @run main) starts up a new JVM. That is a lot of overhead if the test granularity is small.

Samevm mode requires that tests be more careful about cleaning up after themselves, which many do not do. The tests in the OpenJDK langtools repository can be run in “samevm” mode.

One problem with samevm mode is that if a test does not clean up after itself, it can prevent all following tests from passing.

In agentvm mode, jtreg will attempt to clean up a JVM after it has been used by an action of a test: if it can be cleaned up, the JVM is saved for later reuse; otherwise it is discarded. Thus, in the best case, the performance is close to that of samevm mode; in the worst case, it degrades to the performance of othervm mode. In practice, it is somewhere in between.

Starting jtreg

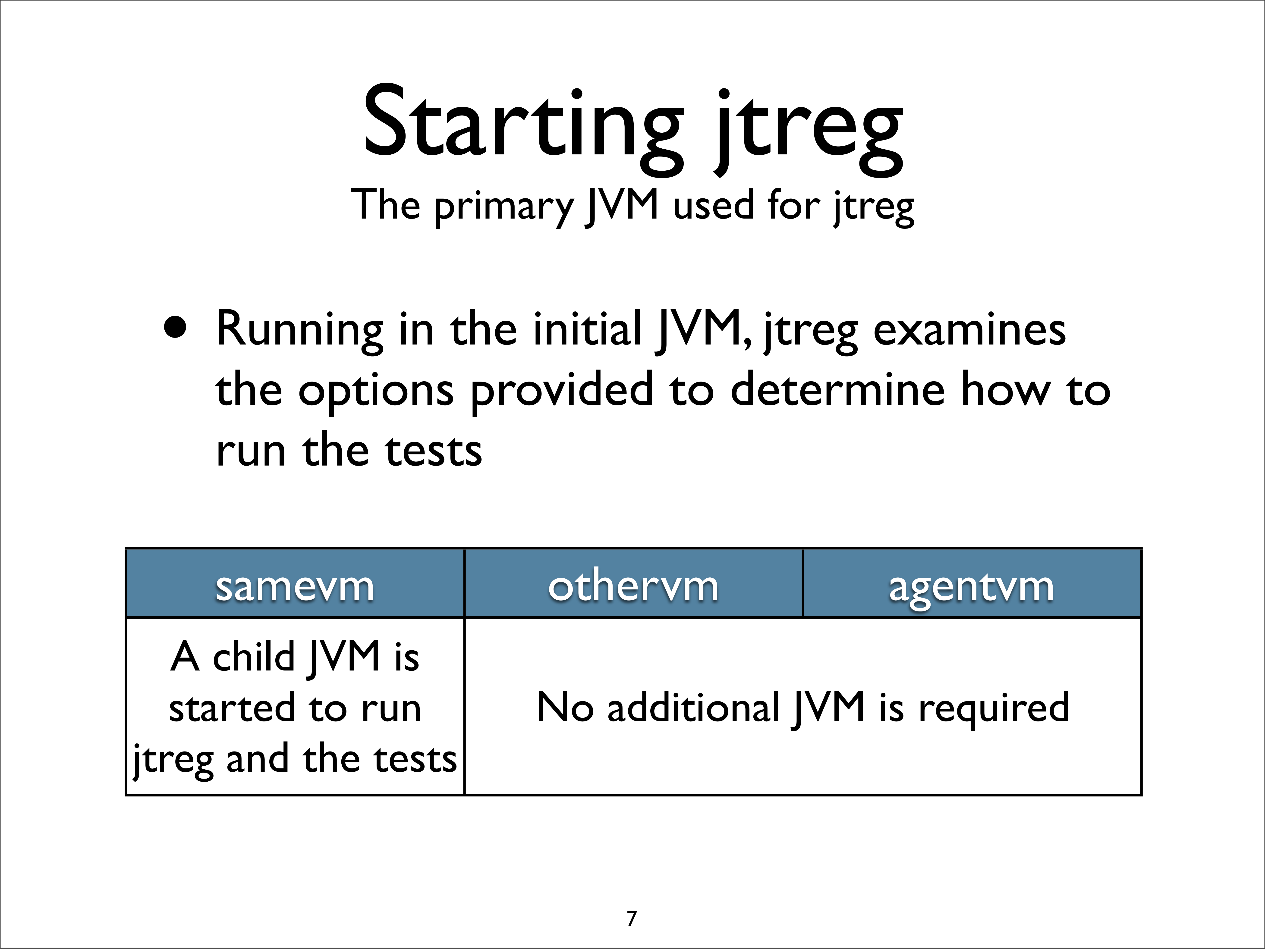
The initial JVM used for jtreg

jtreg script	java -jar jtreg.jar
Script analyzes options and environment variables for JDK and JVM options used to run jtreg JDK: ${JT_JAVA}, ${JAVA_HOME}, -jdk:*, java JVM Options: -J*	No analysis possible: jtreg uses host JVM

jtreg can be run either by using a utility script or by executing the jar file directly.

If the script is used, some simple analysis of the script’s arguments is done, to determine the JDK used to run jtreg, and any JVM options to be passed to that JDK.

If the jar file is run directly, it is up to the user to determine the JDK to use and any JVM options that may be required.



Starting jtreg

The primary JVM used for jtreg

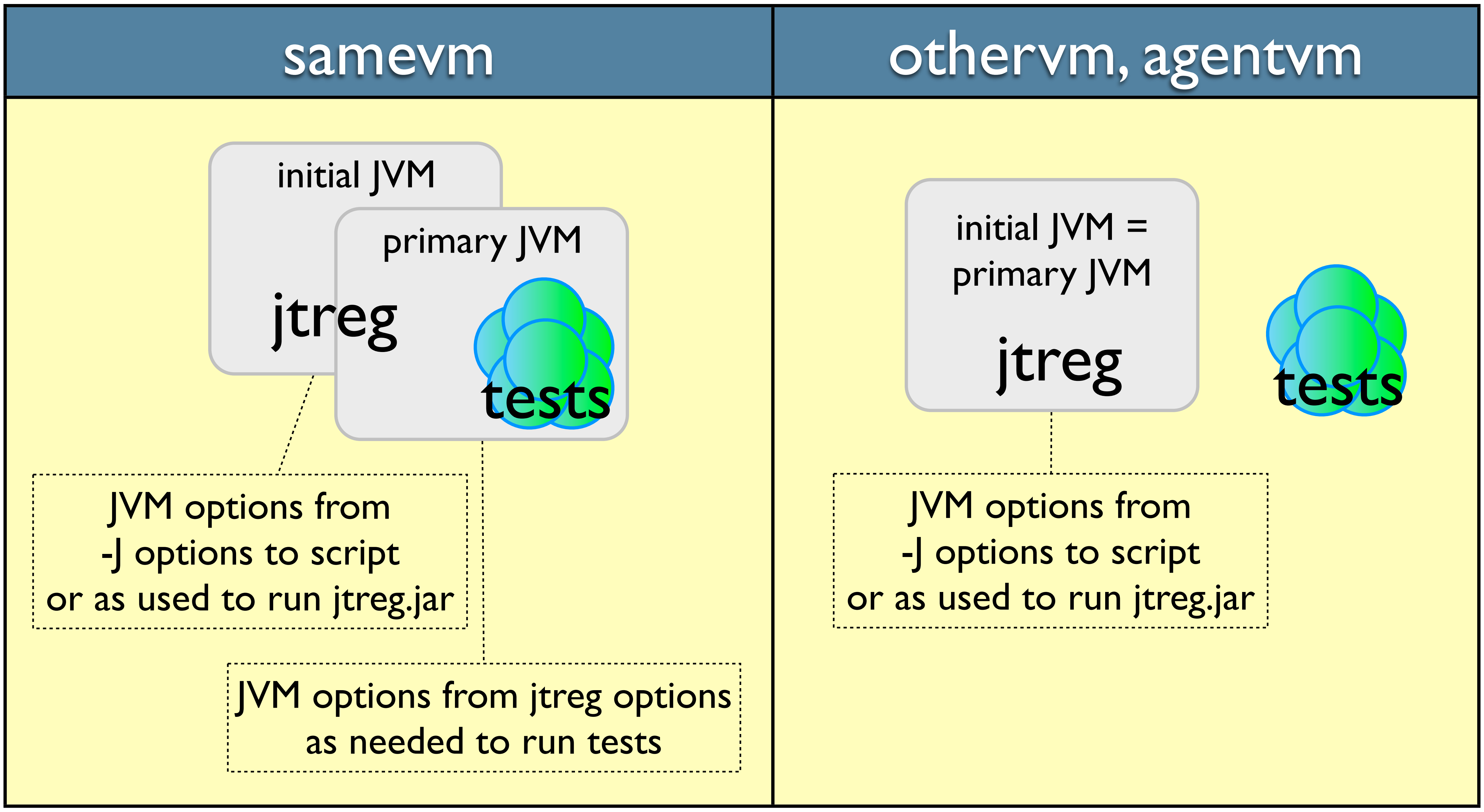
- Running in the initial JVM, jtreg examines the options provided to determine how to run the tests

samevm	othervm	agentvm
A child JVM is started to run jtreg and the tests	No additional JVM is required	

When jtreg is started, it does an initial scan of the options, to determine how the tests are to be run.

If the default test execution mode is “samevm”, jtreg will determine whether or not the current JVM can be used to run the tests. In general, this is unlikely, and so jtreg runs itself in a child JVM with the required characteristics.

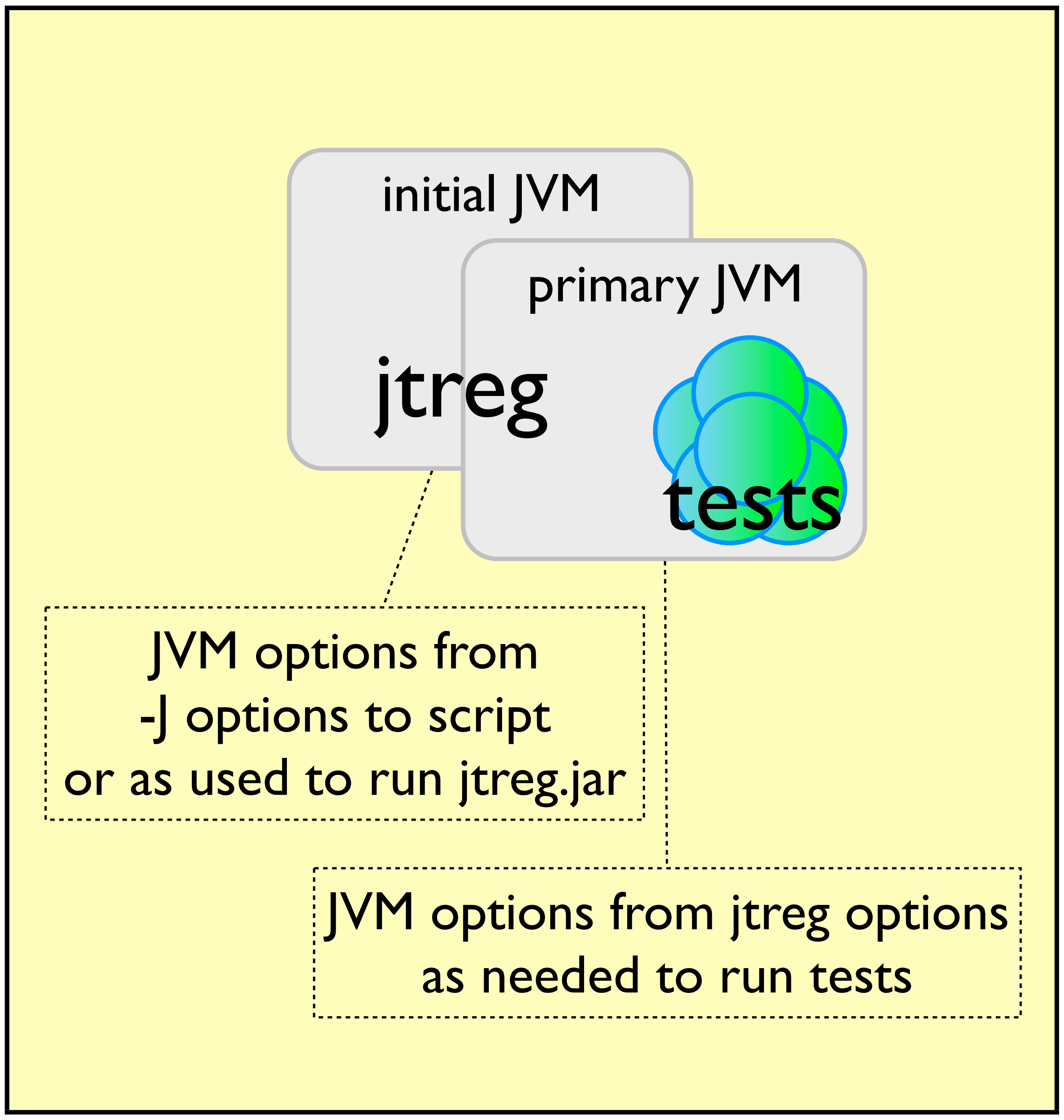
jtreg JVMs



This means that in samevm mode, there are typically two JVMs active: the initial JVM, and then the primary JVM used to actually run the jtreg and the tests.

But, in othervm and agentvm mode, jtreg can run the tests from within the initial JVM. The tests will be run in additional JVMs that are created as needed.

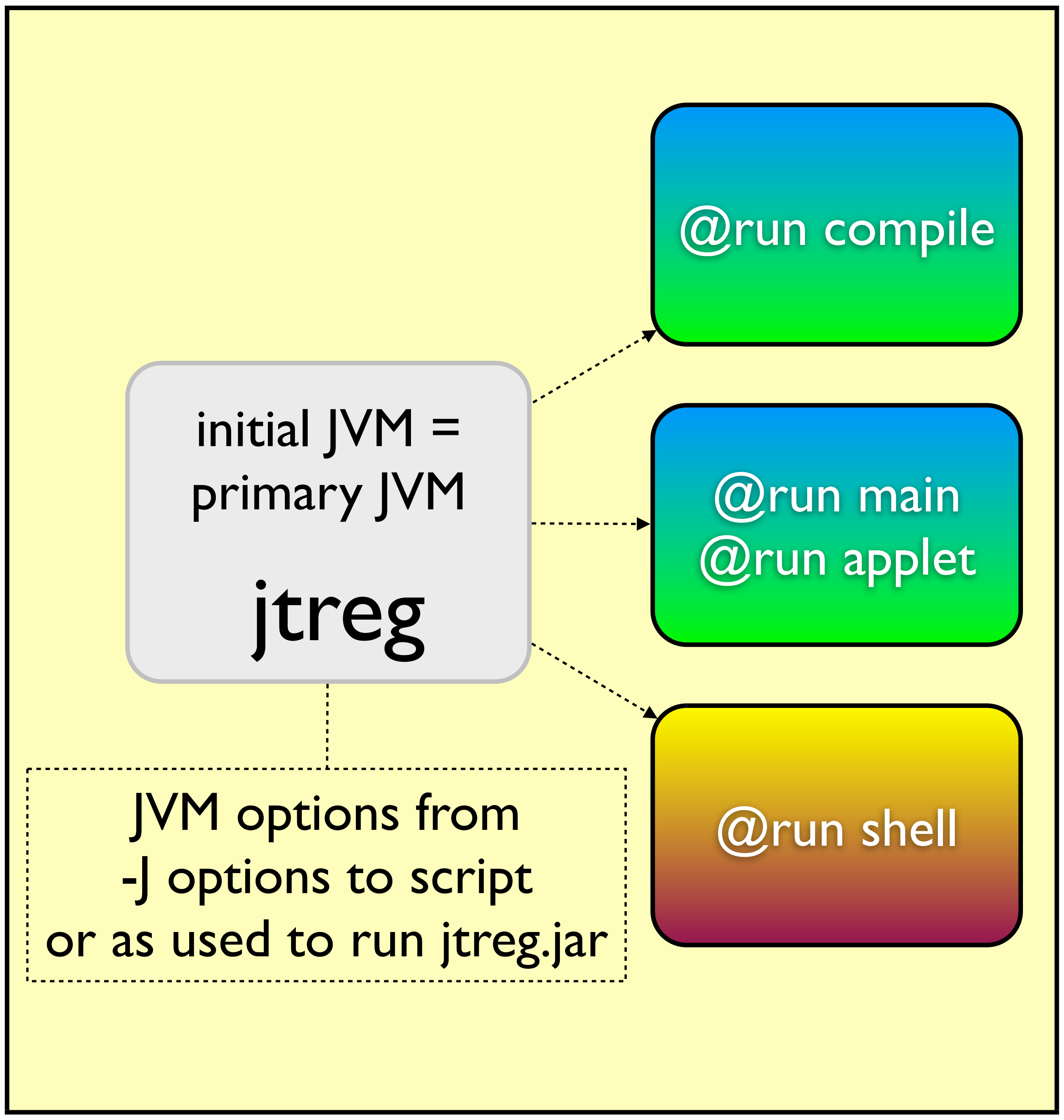
Test JVMs: samevm



- By default, all actions run in the same JVM
- Inherent limitations on JVM and JVM options
  - Compile JDK = Test JDK
  - -javaoptions not allowed

In samevm mode, no additional JVMs are necessary, because jtreg will run the tests in the same JVM that it has already started for itself.

Test JVMs: othervm



initial JVM = primary JVM

jtreg

JVM options from -J options to script or as used to run jtreg.jar

@run compile

@run main
@run applet

@run shell

- By default, all actions run in a new JVM or process
- Characteristics depend on type of action

compile	main/applet	shell
-compilejdk -vmoptions -javacoptions	-testjdk -vmoptions -javaoptions	options available in env variables

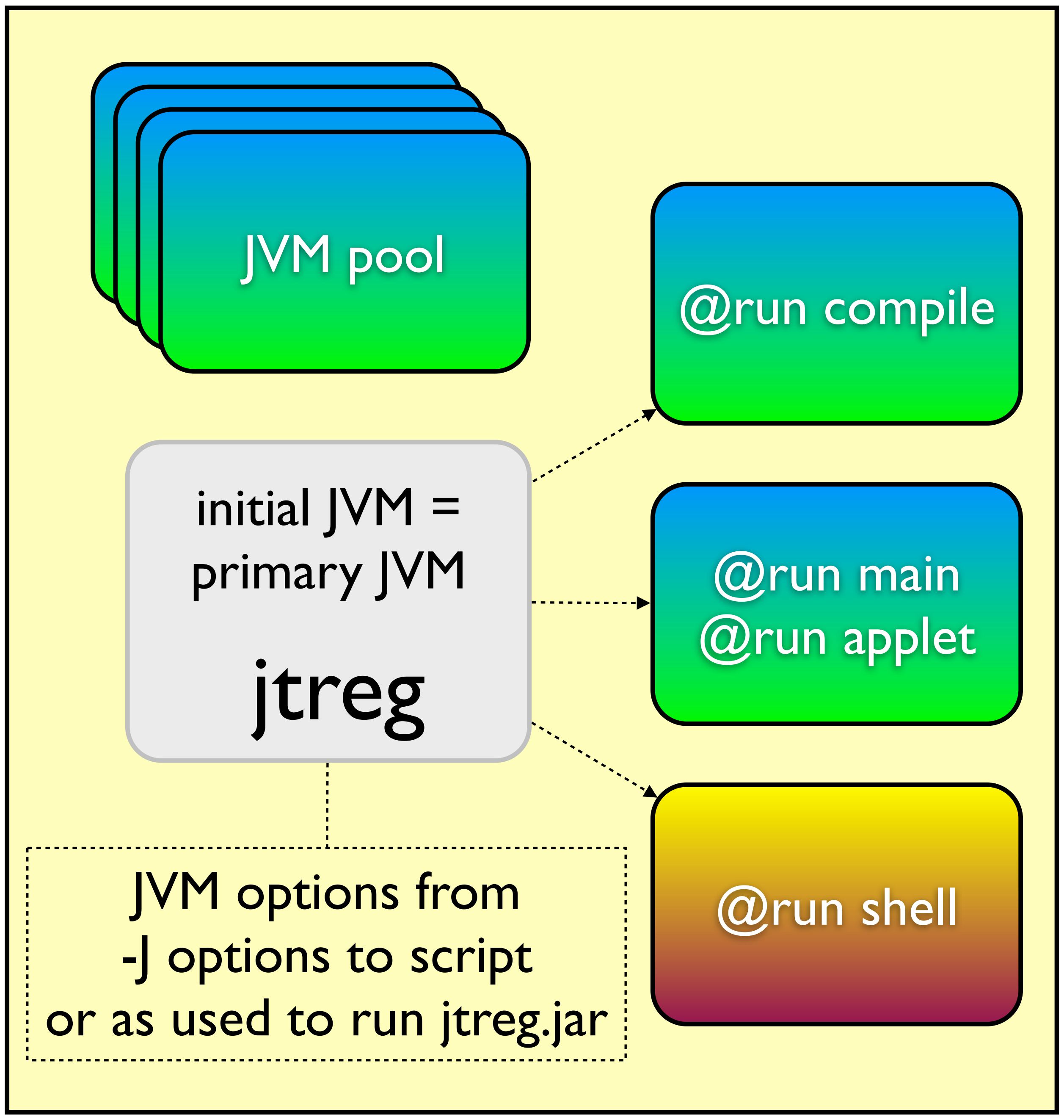
Wednesday, March 13, 2013

In othervm mode, most actions (@run main, @run compile, @run applet) will be initiated in a fresh new JVM.   The VM options for these child JVMs are determined from the command line options.

Options specified with –vmoption:* are applied to all JVMs that are created.   In addition, options specified with –javaoption:* are applied to JVMs used to run tests (@run main, @run applet) as compared to compiling the tests (@run compile).

Shell actions (@run shell) will be run in a new process, which may in turn create new JVMs. It is the responsibility of the author of the shell script to propagate the necessary options onto the JVMs created when the shell script is run.

Test JVMs: agentvm



- By default, Java actions run in a JVM with the right characteristics
- Characteristics depend on type of action

compile	main/applet	shell
-compilejdk -vmoptions -javacoptions	-testjdk -vmoptions -javaoptions	options available in env variables
reuse: yes	reuse: if main	reuse: N/A

For the most part, agentvm mode is similar to othervm mode.

The main difference is that jtreg maintains a “pool” of JVMs with different characteristics (JDK, JVM options, current directory, etc).

Using machines from a pool removes the overhead of setting up a new JVM for each action. It also allows HotSpot to optimize the frequently used code.

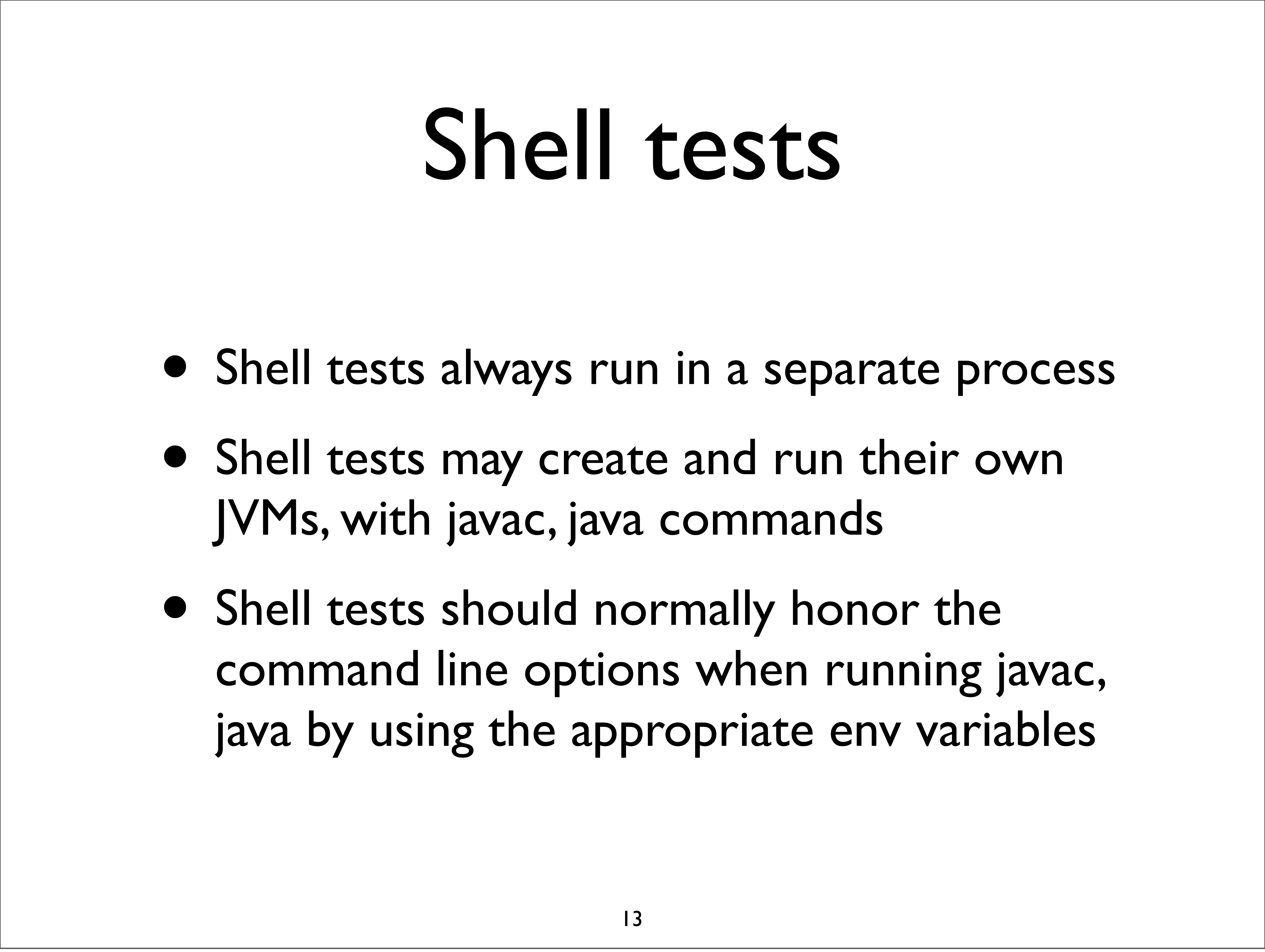
Test JVMs: agentvm

- jtreg maintains a pool of available JVMs
- JVMs are created as needed for a given set of characteristics (JDK, JVM options, etc.)
- When no longer needed, JVMs are reset and returned to the pool
- If a JVM cannot be reset, it is discarded

Instead of always creating a fresh new JVM for each action, jtreg checks to see if a JVM with the right characteristics is available in a shared pool. If so, that JVM is taken from the pool and used; otherwise a new JVM with the right characteristics is created.

When the action has completed, jtreg will attempt to “clean up” after the test and reset it to a standard state. It is successful, the JVM is returned to the pool for later reuse. Otherwise, if the JVM cannot be reset for whatever reason, the JVM is simply discarded.

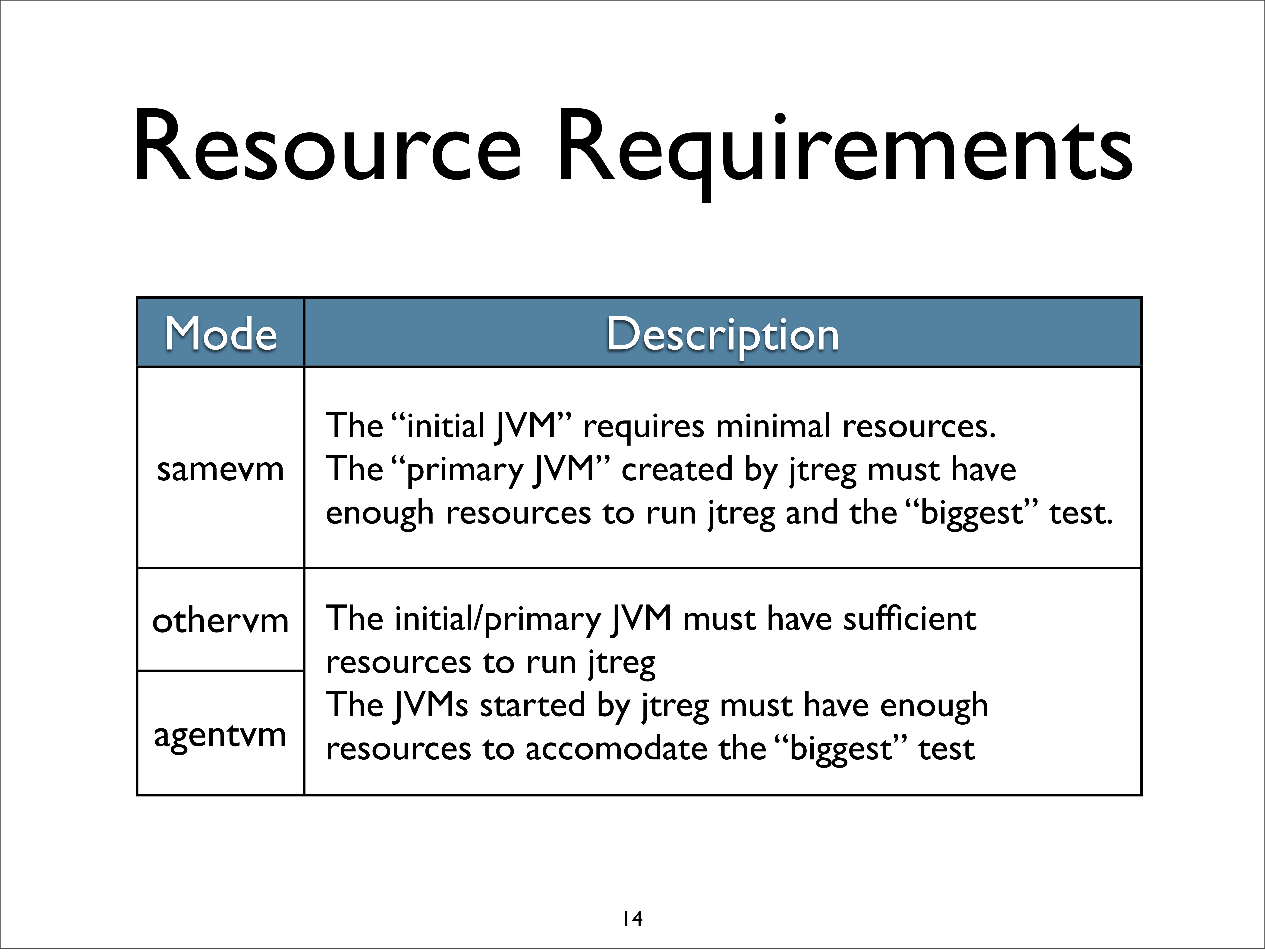
If a test specifies that an action should be executed in an unshared JVM (e.g. with /othervm) then a fresh new VM will be created for that specific action.



Shell tests

- Shell tests always run in a separate process
- Shell tests may create and run their own JVMs, with javac, java commands
- Shell tests should normally honor the command line options when running javac, java by using the appropriate env variables

Shell actions (@run shell) will be run in a new process, which may in turn create new JVMs. It is the responsibility of the author of the shell script to propagate the necessary options onto the JVMs created when the shell script is run.



Resource Requirements

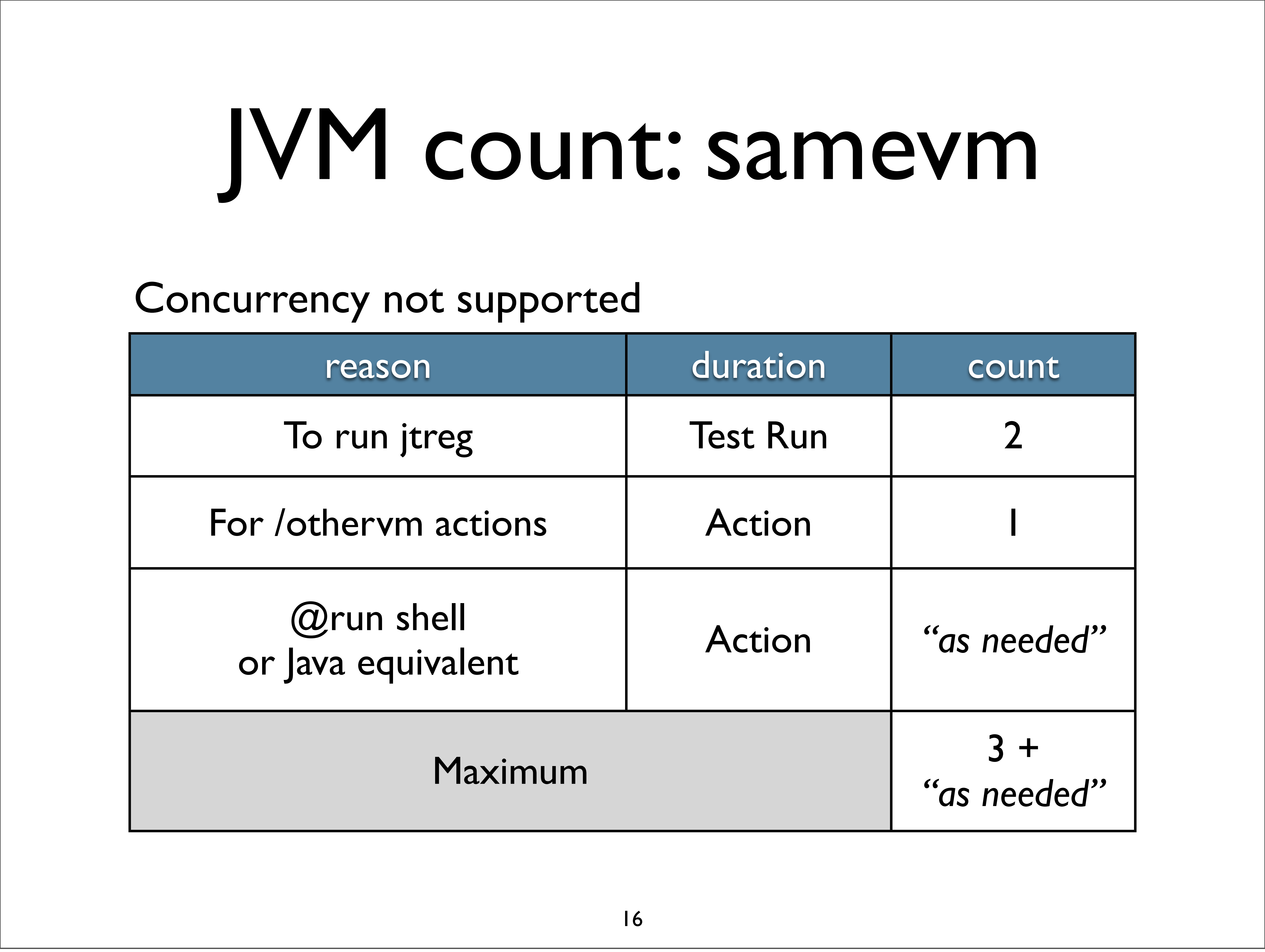
Mode	Description
samevm	The “initial JVM” requires minimal resources. The “primary JVM” created by jtreg must have enough resources to run jtreg and the “biggest” test.
othervm	The initial/primary JVM must have sufficient resources to run jtreg The JVMs started by jtreg must have enough resources to accomodate the “biggest” test
agentvm	

When considering the resource requirements for a test run, you need to understand the JVMs that will be (or may be) run and provision the JVMs accordingly.

Concurrent Test Execution

- jtreg supports concurrent test execution in agentvm and othervm modes
- Test actions still isolated into separate JVMs
  - Concurrent test execution not supported in samevm mode because of isolation issues
- Concurrent test execution affects total resource requirements

Concurrent test execution on a big multi-core machine can dramatically speed up test runs, but you need to be even more aware of the overall requirements for system resources that will be used.

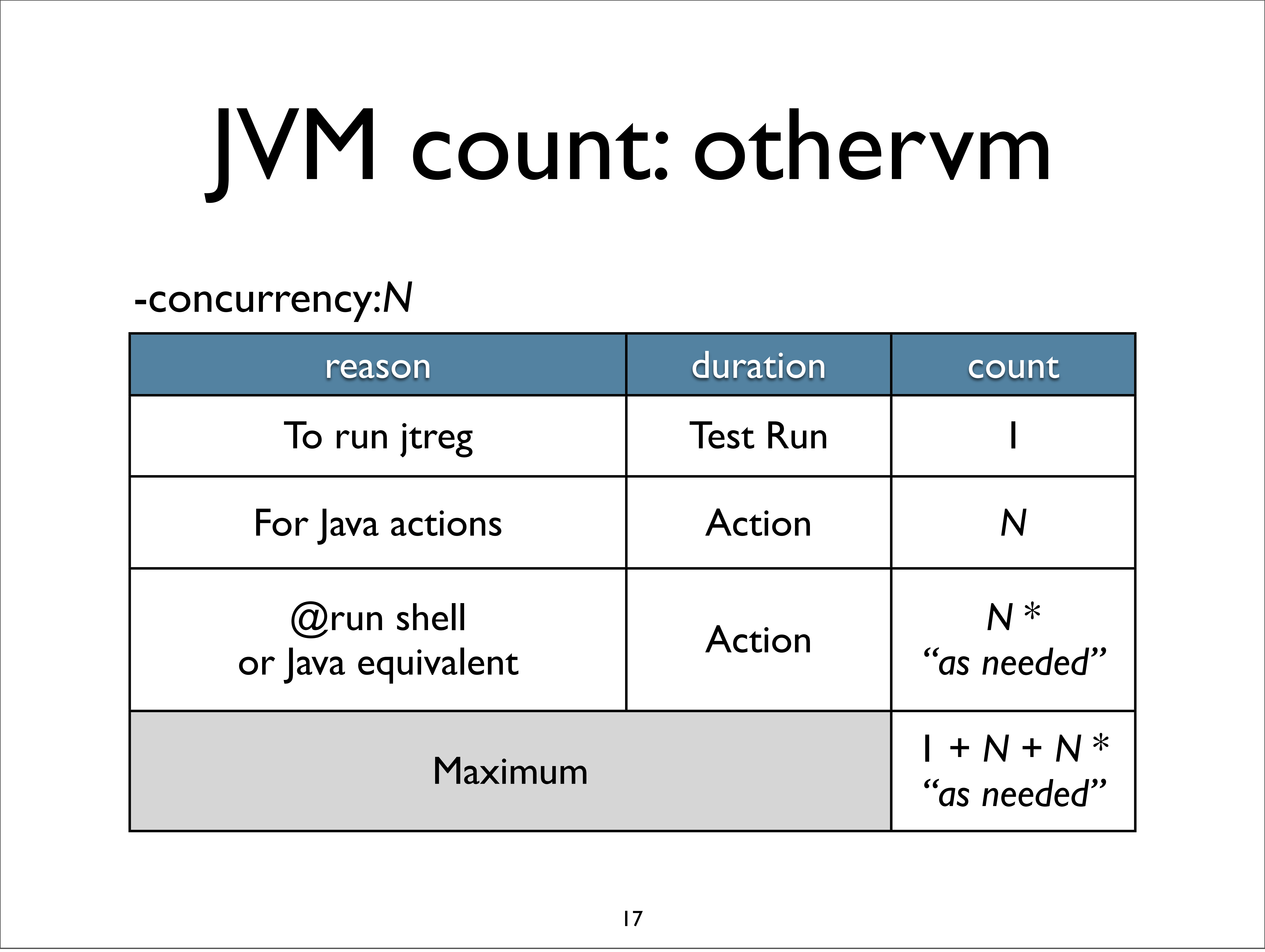


JVM count: samevm

Concurrency not supported

reason	duration	count
To run jtreg	Test Run	2
For /othervm actions	Action	1
@run shell or Java equivalent	Action	*"as needed"*
Maximum		3 + *"as needed"*

In samevm mode, there will likely be the initial JVM, and then the primary JVM used to run most of the tests. Another JVM will be transiently required for any action that are specified to require an unshared JVM. And finally, some tests, especially shell tests, may temporarily create their own JVM.

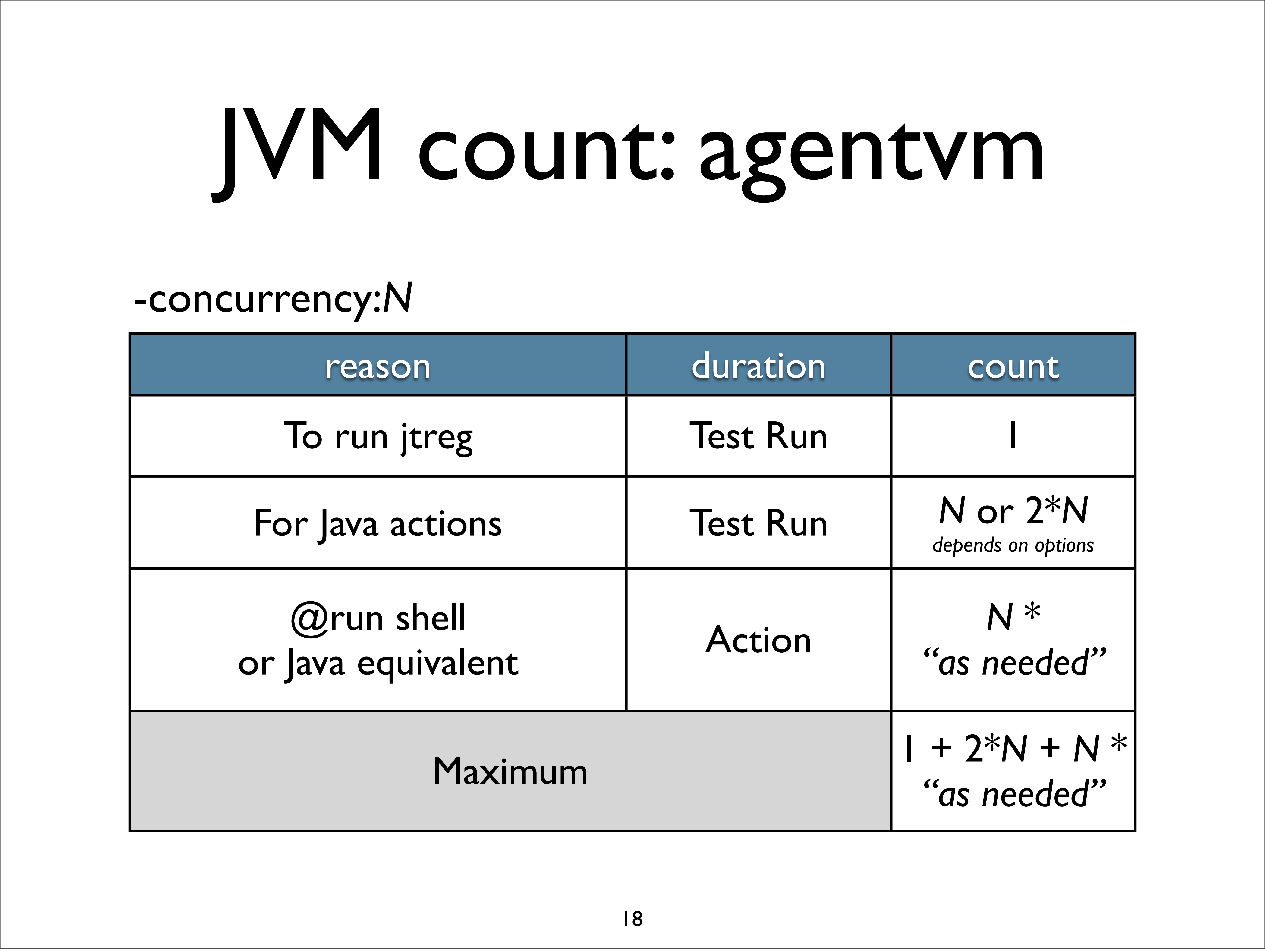


JVM count: othervm

-concurrency:*N*

reason	duration	count
To run jtreg	Test Run	1
For Java actions	Action	*N*
@run shell or Java equivalent	Action	*N* * *“as needed”*
Maximum		1 + *N* + *N* * *“as needed”*

In othervm mode, you need a single JVM to run jtreg, and then in general, for each of the threads being used to execute tests concurrently, you need to create the JVMs required to execute the test. If the concurrency level is set to N, most of the time there will be N+1 JVMs running. But, as always, some tests, especially shell tests, may create additional JVMs while the test is running.



JVM count: agentvm

-concurrency:*N*

reason	duration	count
To run jtreg	Test Run	1
For Java actions	Test Run	*N* or 2**N* *depends on options*
@run shell or Java equivalent	Action	*N* * *“as needed”*
Maximum		1 + 2**N* + *N* * *“as needed”*

The calculation for agentvm mode is somewhat similar to that for othervm mode, except that the JVMs in the shared pool have to be taken into account.

If the characteristics are the same for the JVMs required to compile and run the actions of a test, then the Java actions for the test can serially share the same JVM. But if characteristics differ (for example, specifying -javaoptions, or -compilejdk) then different JVMs will be serially required, with one being idle in the pool while the other is being used.

See Also

- http://openjdk.java.net/jtreg/
- http://openjdk.java.net/jtreg/vmoptions.html
- http://openjdk.java.net/projects/code-tools/jtreg/
- http://openjdk.java.net/projects/code-tools/jtreg/intro.html
- https://blogs.oracle.com/jjg/entry/jtreg_old_and_new

Happy reading.