Large Scale Testing in an Agile World

Delivering an enterprise quality IBM SDK, Java Technology Edition 7.0

Important Disclaimers

Introduction to the speaker – William Smith

- Based in IBM's Java Technology Centre, Hursley, UK
- 12 years' experience working on IBM's Java implementation
  - Class Library development - graphics, font, globalization
  - Unit Test and System Test
  - Consumability tools development - monitoring, diagnostics
- Current role: software engineer in the Java Service team, working on customer PMRs
- will.smith@uk.ibm.com

Agenda

- Overview of Java in IBM

- Challenges of testing a complex product at multiple sites
  - Introduction to the Java SDK
  - What drives our testing: applications, platforms, releases
  - The types of tests we run

- Practical large scale multi-platform testing: **Test automation**.
  - Scheduling test execution
  - How to categorize and analyze test failures (real and false)
  - Reporting test status

Why is Java important to IBM ?

- Java^{tm} – the language and the runtime – is critically important to IBM...
  - Provides fundamental infrastructure to ***hundreds*** of products in IBM's software portfolio
  - Delivers a standard development and runtime environment for
    - IBM customers
    - IBM product development teams
    - Independent Software Vendors (ISVs) supporting IBM server platforms (AIX, z/OS, IBM iSeries)

- IBM invests in in the performance, reliability and serviceability of the Java runtime
  - Benefits all the applications and products that use Java runtime

What does our testing need to achieve?

- Ensure the delivery of a high quality runtime for
  - Hundreds of IBM products
  - Thousands of customer and ISV applications

- Testing also needs to encompass “off the shelf” Java applications
  - Open source applications
  - Hadoop distributed processing framework
  - Lucene search engine library
  - Tomcat, Geronimo application servers
  - Scripting languages: JRuby, Jython, Scala, Clojure, Groovy, ...

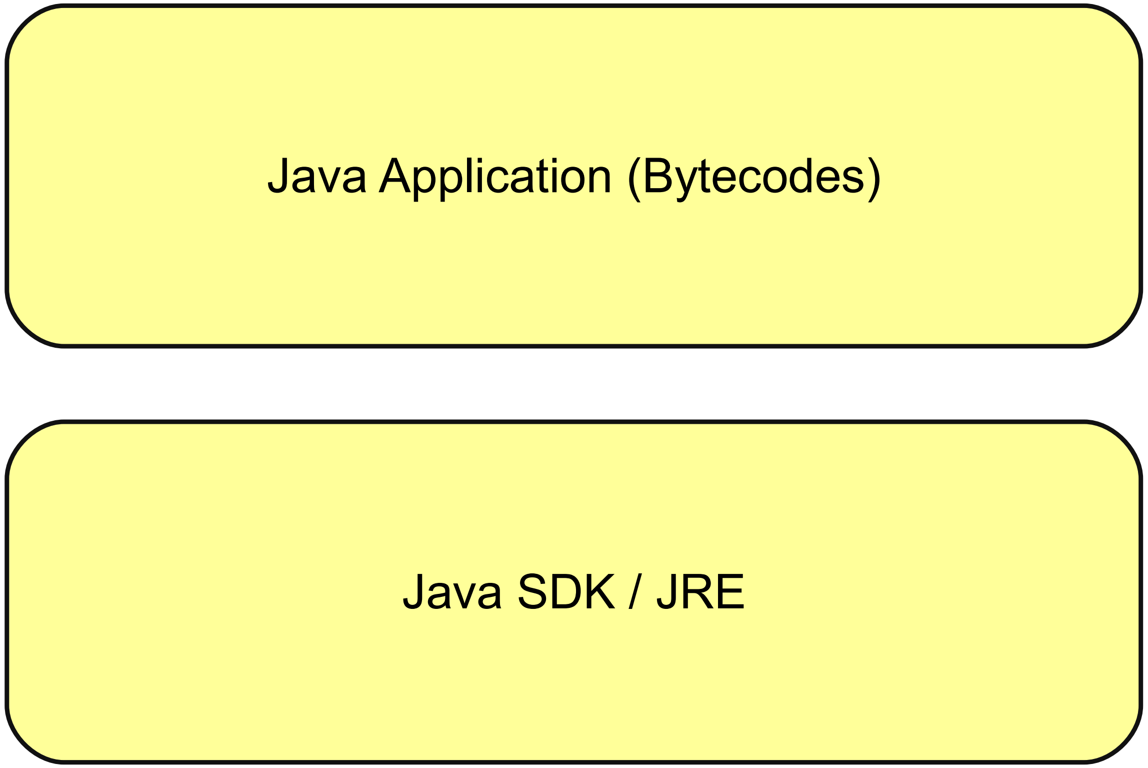
- Provide coverage across the whole Java API

IBM SDK, Java Technology Edition Version 7.0

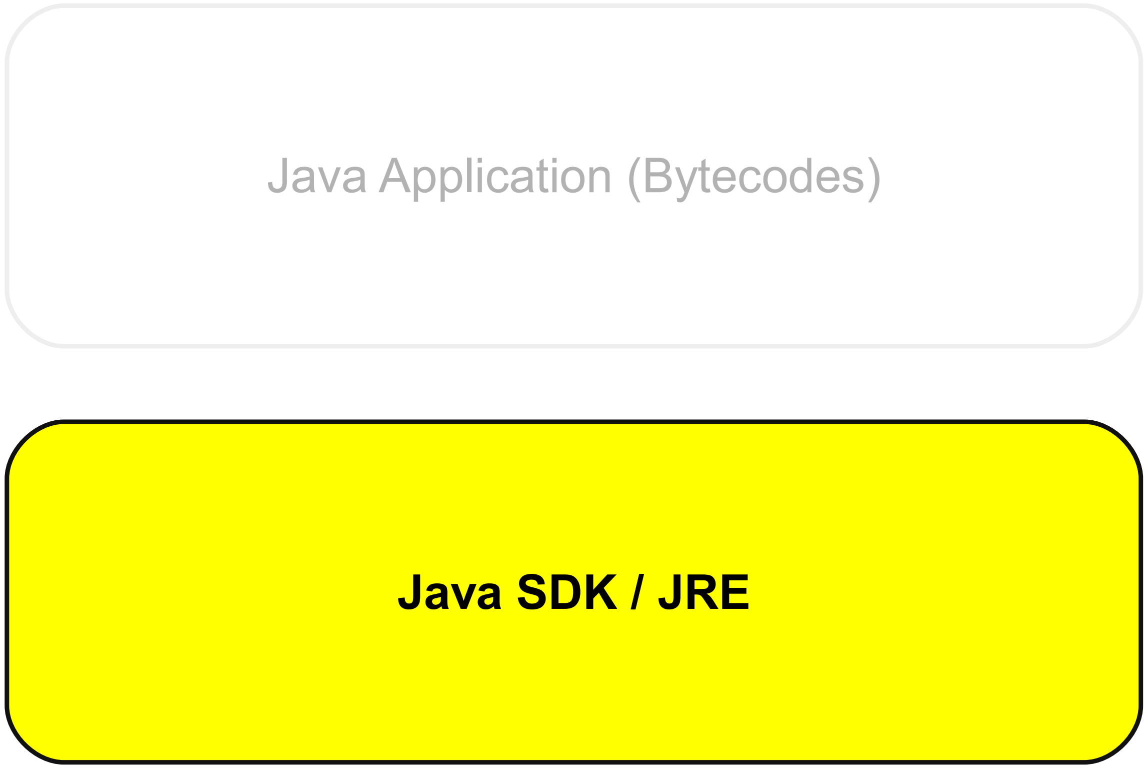
- General Availability September 2011
  - Improved throughput
  - Faster startup
  - Smaller footprint
  - Introduces Balanced GC
  - Added soft Real Time Java capabilities
  - Improved monitoring and diagnostics
- Operating systems
  - AIX, Linux, z/OS, Windows, Solaris
- Platforms
  - Power, System z, Intel, AMD, SPARC
- Latest refresh: Java 7 SR5, June 2013
  - Stability, Security fixes and features

What is the Java SDK? … the JRE?

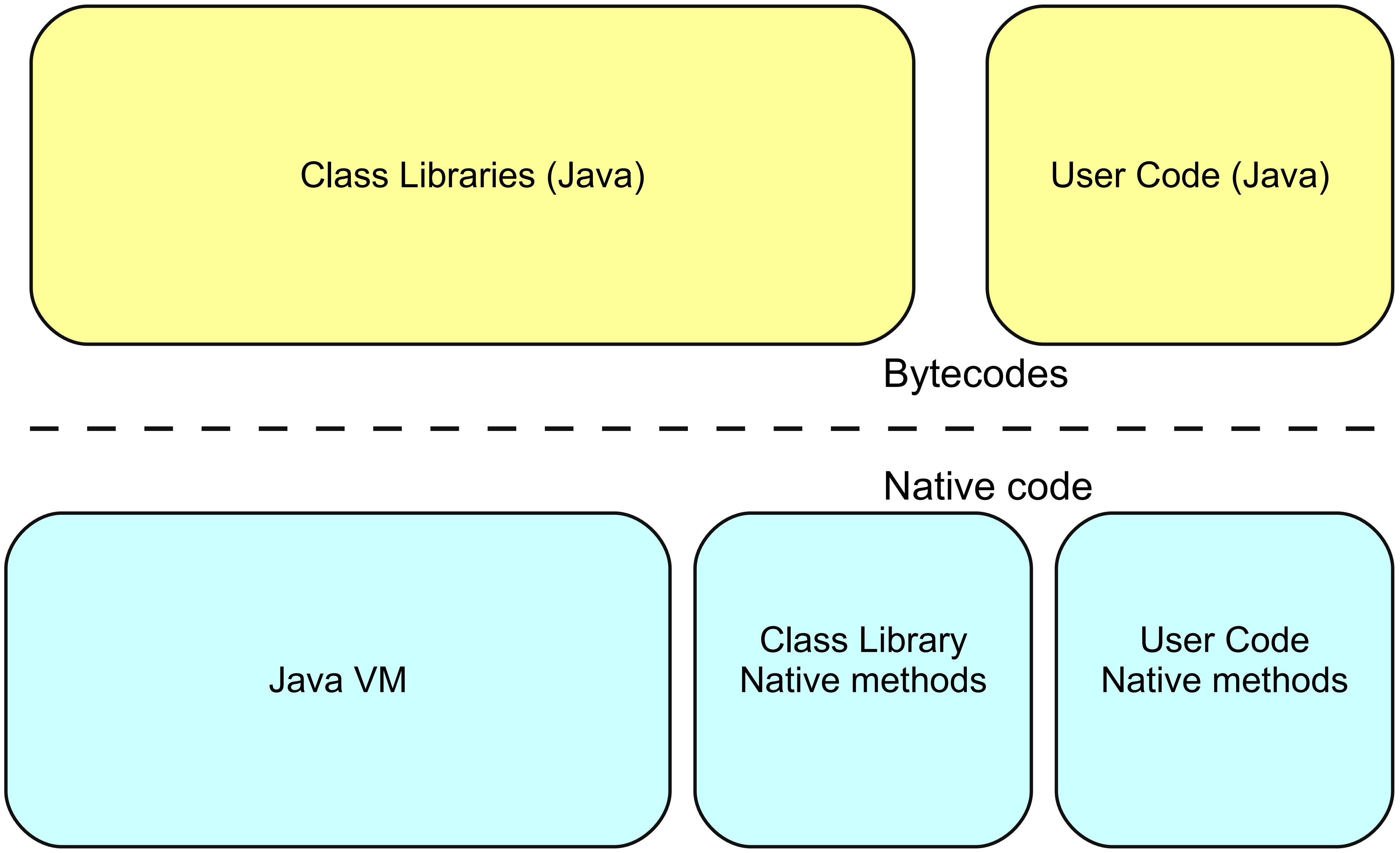
Java Software Development Kit
=
Java Runtime Environment + developer's tools like the javac compiler



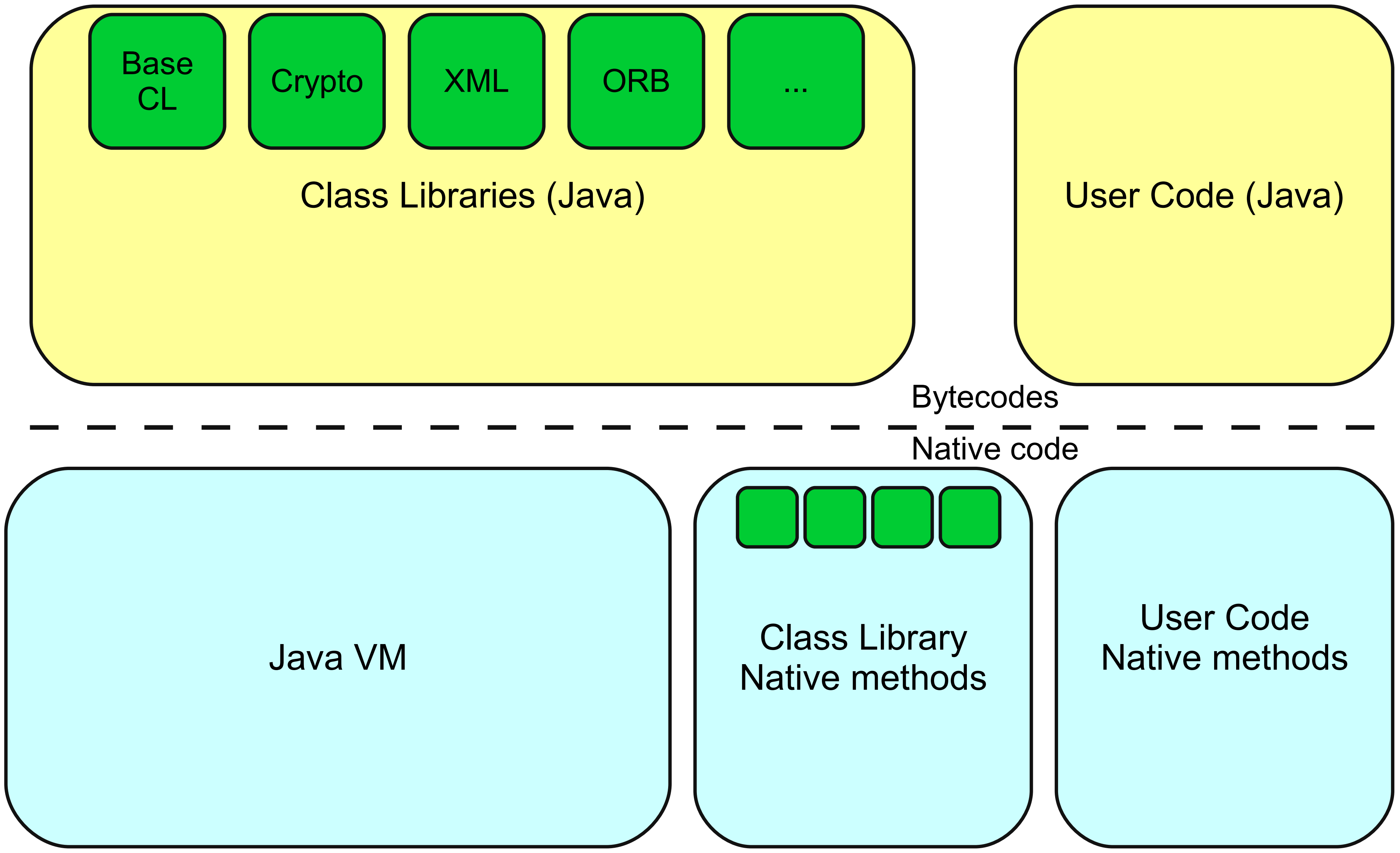
The Java runtime is the software component the JTC delivers



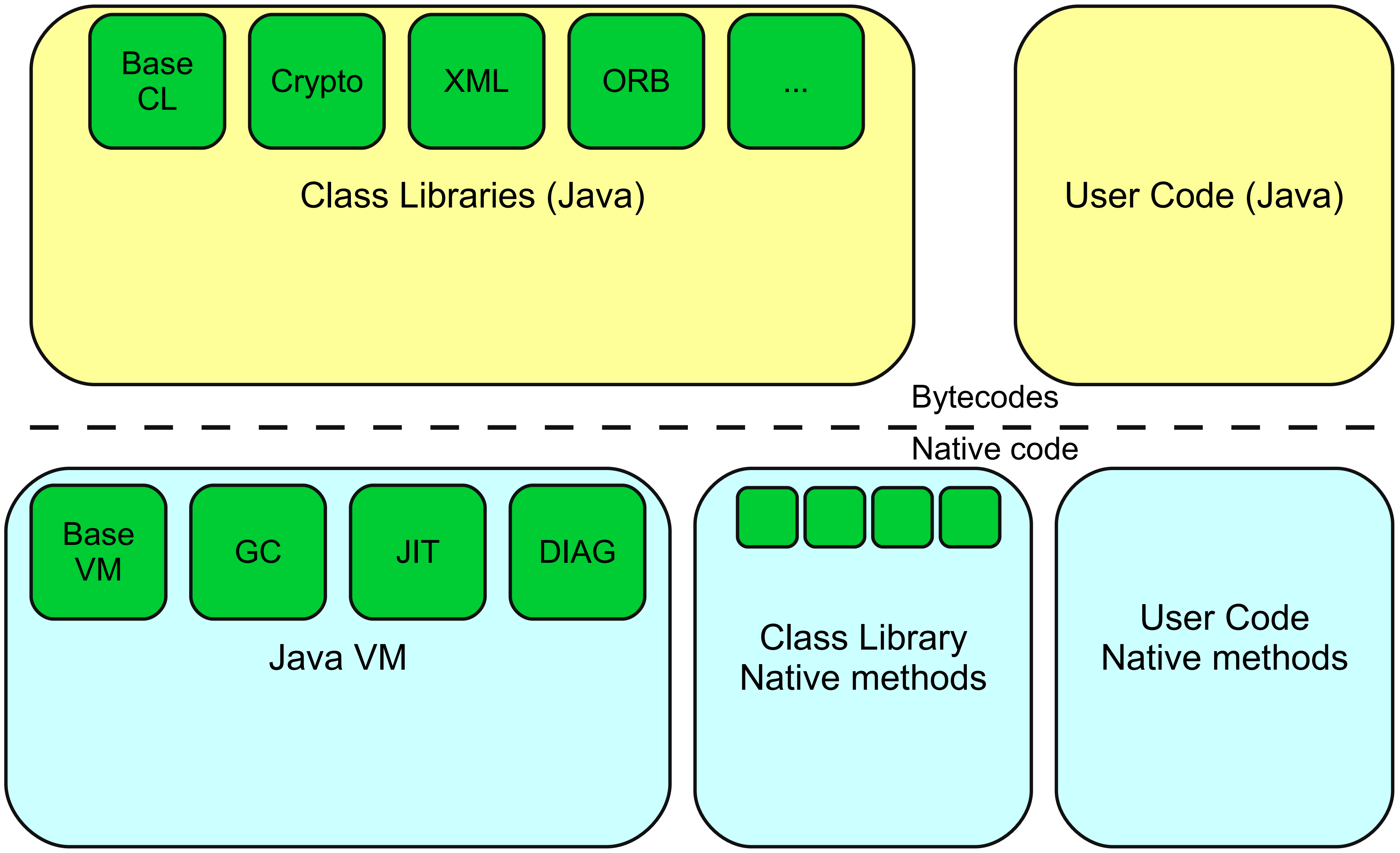
What's inside the Java runtime?



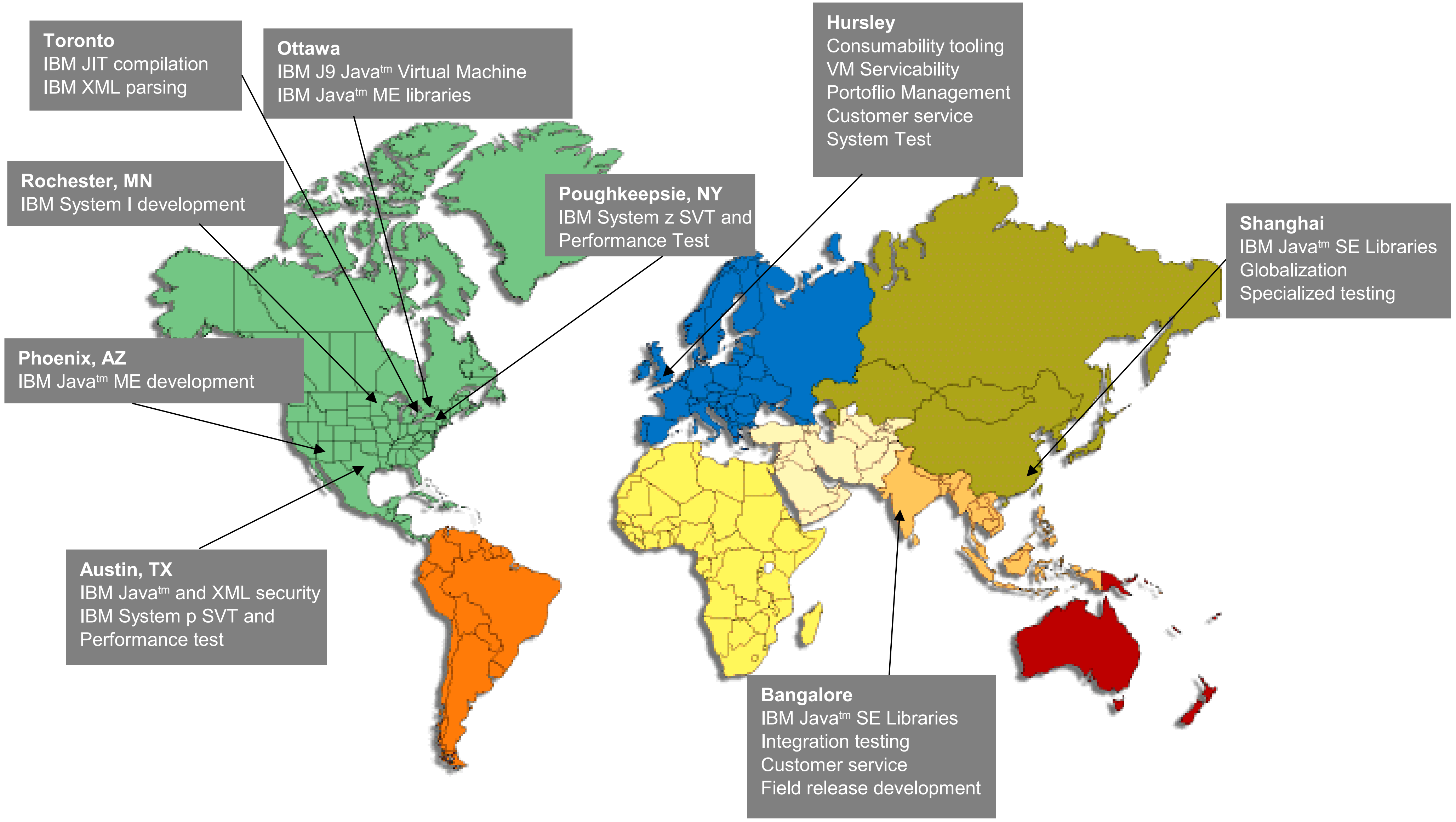
What's inside the Java runtime?



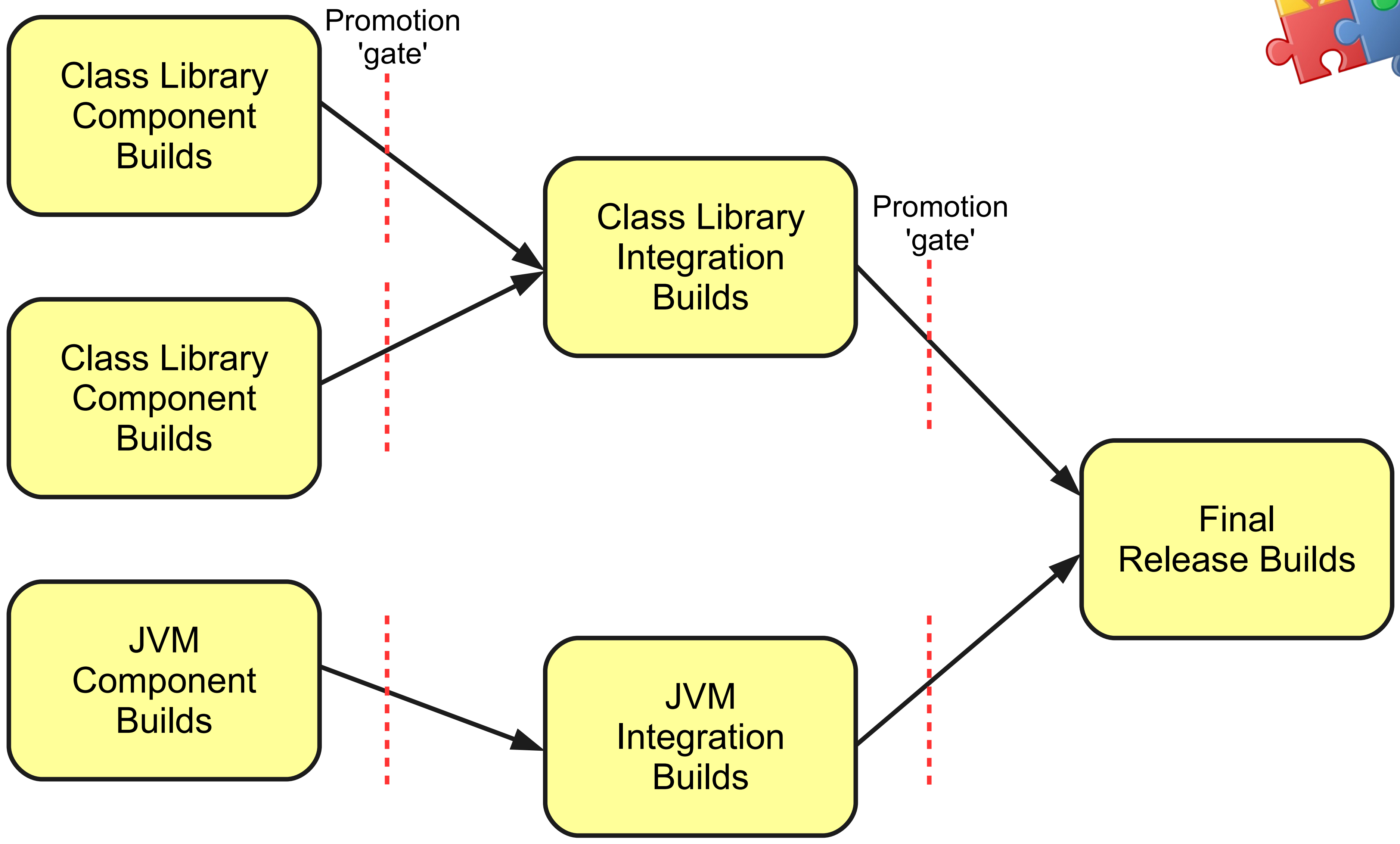
What's inside the Java runtime?



IBM Java Technologies Team – A World Wide Organization



How does the work flow fit together?



Build Process Summary

- 9 independent component builds (at least)

Build Process Summary

- 9 independent component builds (at least)
- Final Release Build runs on average 2 times daily

Build Process Summary

- 9 independent component builds (at least)
- Final Release Build runs on average 2 times daily
- Builds on 20 platforms (OS + architecture combinations)

Build Process Summary

- 9 independent component builds (at least)
- Final Release Build runs on average 2 times daily
- Builds on 20 platforms (OS + architecture combinations)
- Builds on 5 supported Java release streams

Build Process Summary

- 9 independent component builds (at least)
- Final Release Build runs on average 2 times daily
- Builds on 20 platforms (OS + architecture combinations)
- Builds on 5 supported Java release streams
- Typically 200 builds per day!

The Big Problem for Test – Scale

- 100s of builds tested every week

The Big Problem for Test – Scale

- 100s of builds tested every week
- 1000s of test machines

The Big Problem for Test – Scale

- 100s of builds tested every week
- 1000s of test machines
- 10000s of hours of tests run every day

The Big Problem for Test – Scale

- 100s of builds tested every week
- 1000s of test machines
- 10000s of hours of tests run every day
- *Millions* of testcases executed every month

The Big Problem for Test – Scale

- 100s of builds tested every week
- 1000s of test machines
- 10000s of hours of tests run every day
- *Millions* of testcases executed every month
  ...and they don't all pass!

Java application variety

- Java applications have changed a lot in 16 years...

Java application variety

- Heap size?

Java application variety

- Heap size from 32MB to 100s of GB

Java application variety

- Heap size from 32MB to 100s of GB
- Thread counts?

Java application variety

- Heap size from 32MB to 100s of GB
- Thread counts from 1 to thousands

Java application variety

- Heap size from 32MB to 100s of GB
- Thread counts from 1 to thousands
- Running on 1 to 128+ cores

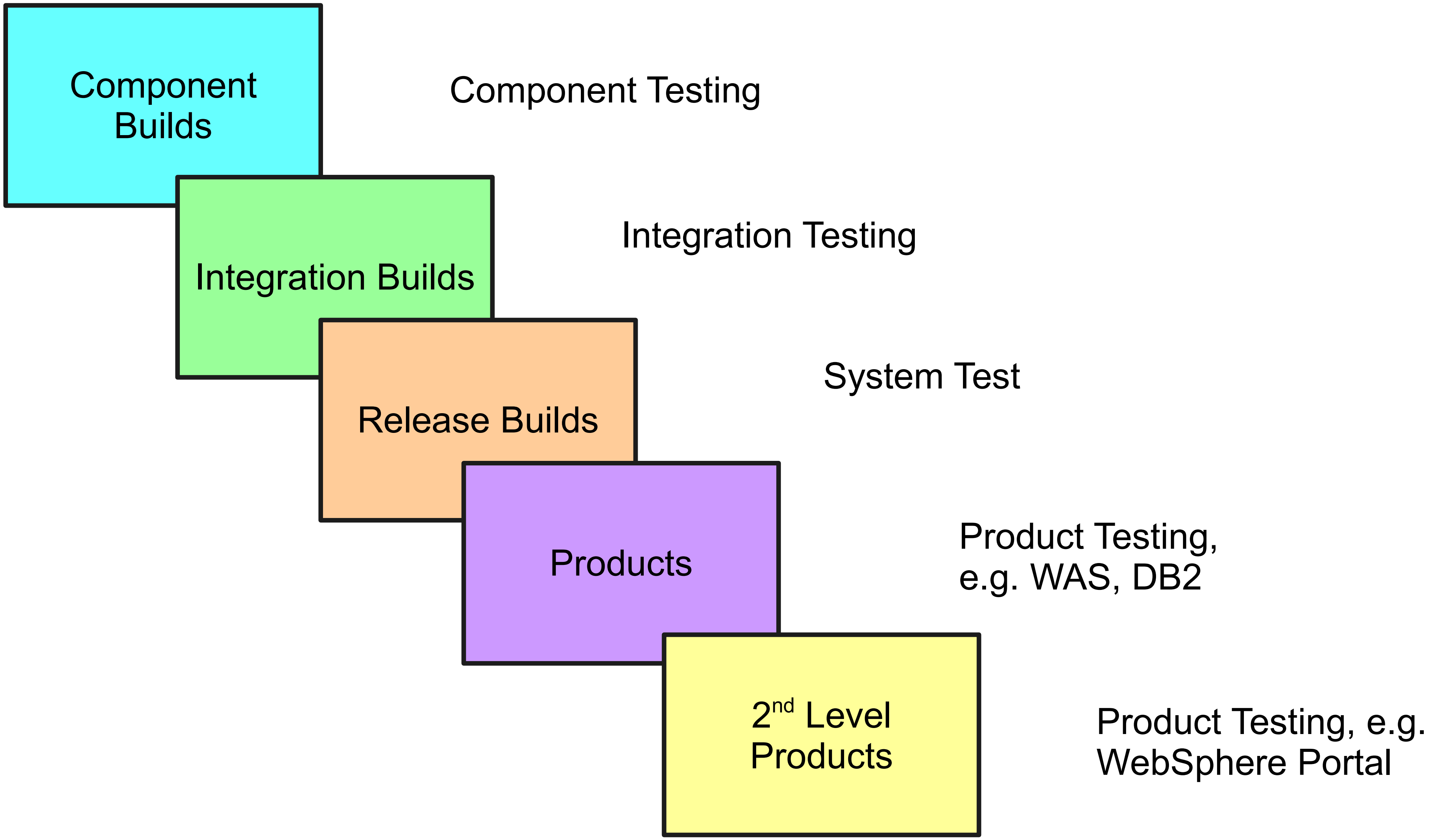
Java application variety

- Heap size from 32MB to 100s of GB
- Thread counts from 1 to thousands
- Running on 1 to 128+ cores
- 4 different Garbage Collector policies

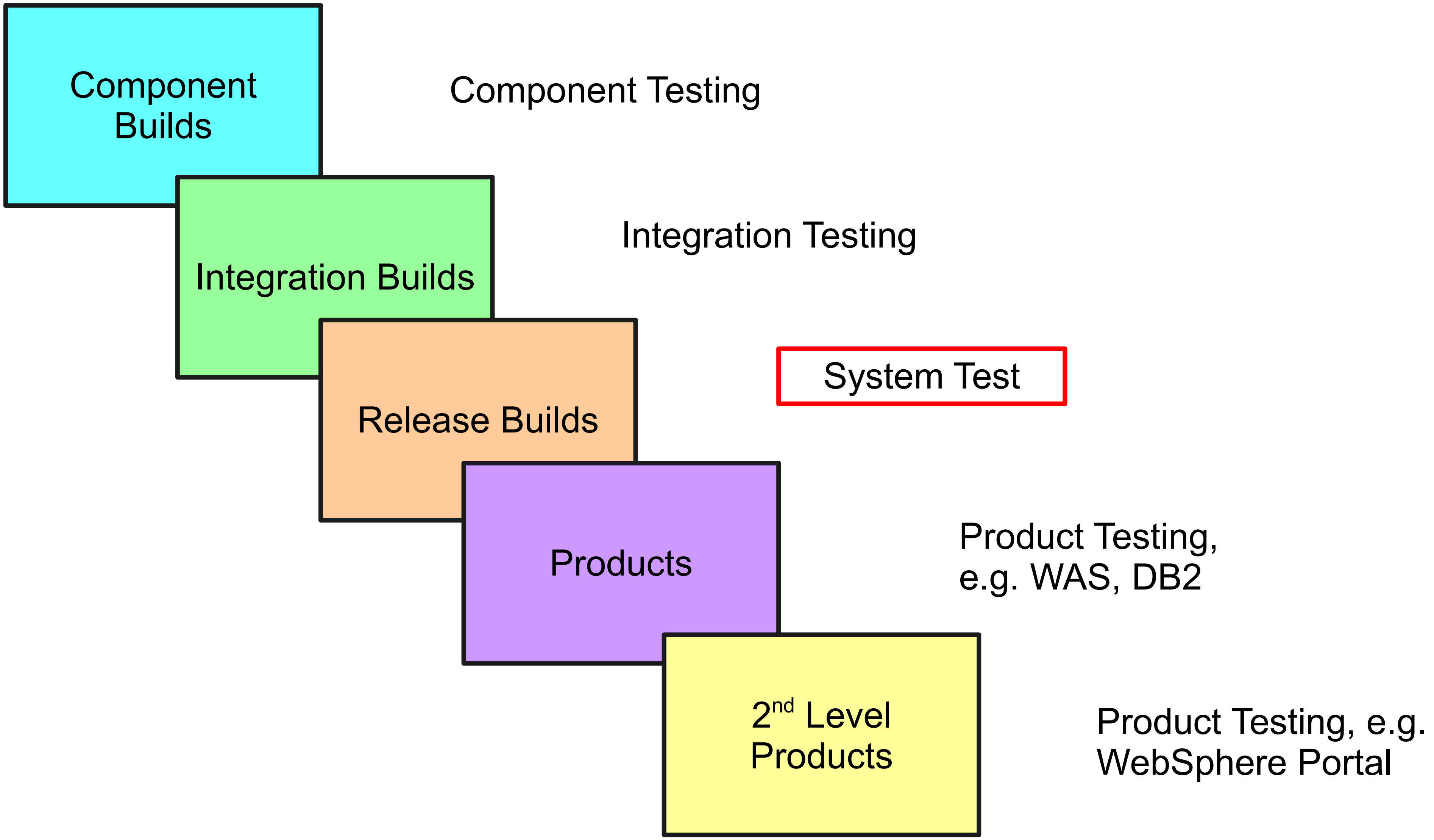
Java application variety

- Heap size from 32MB to 100s of GB
- Thread counts from 1 to thousands
- Running on 1 to 128+ cores
- 4 different Garbage Collector policies
- 1000s of command line option combinations

What kind of testing do we do?



What kind of testing do we do?



Why is System test different?

What kind of testing do we do?

Usage Patterns

Jython

IBM Application Testing

Scala

IBM Features

Project Coin

Apache Harmony

Derby

Customer Scenarios

Application Scenarios

Load Testing

New tests each release

JRuby

JSR292 (java.lang.invoke)

NIO.2

> 200 Test Modes

Eclipse

Mauve

20 Platforms

Regression Testing

What kind of testing do we do?

Usage Patterns

Jython

IBM Application Testing

Scala

IBM Features

Project Coin

Apache Harmony

Derby

Customer Scenarios

Application Scenarios

Load Testing

New tests each release

JRuby

JSR292 (java.lang.invoke)

NIO.2

> 200 Test Modes

Eclipse

As much as possible!

Mauve

20 Platforms

Regression Testing

What are the challenges?

- Challenges
  - Provide ***continual verification*** of the quality of ***every build*** during development
  - Large matrix of combinations of command line options gives >200 Test Modes
  - 10000's of testcases
  - The Java runtime is a complex, dynamic system... some defects can be highly intermittent and difficult to reproduce
  - Tests and their environments need to be reproducible and deployable on multiple test machines

What are the challenges?

- Challenges
  - Provide ***continual verification*** of the quality of ***every build*** during development
  - Large matrix of combinations of command line options gives >200 Test Modes
  - 10000's of testcases
  - The Java runtime is a complex, dynamic system... some defects can be highly intermittent and difficult to reproduce
  - Tests and their environments need to be reproducible and deployable on multiple test machines

- How can we optimize the time and effort spent doing this?

What are the challenges?

- Challenges
  - Provide ***continual verification*** of the quality of ***every build*** during development
  - Large matrix of combinations of command line options gives >200 Test Modes
  - 10000's of testcases
  - The Java runtime is a complex, dynamic system... some defects can be highly intermittent and difficult to reproduce
  - Tests and their environments need to be reproducible and deployable on multiple test machines

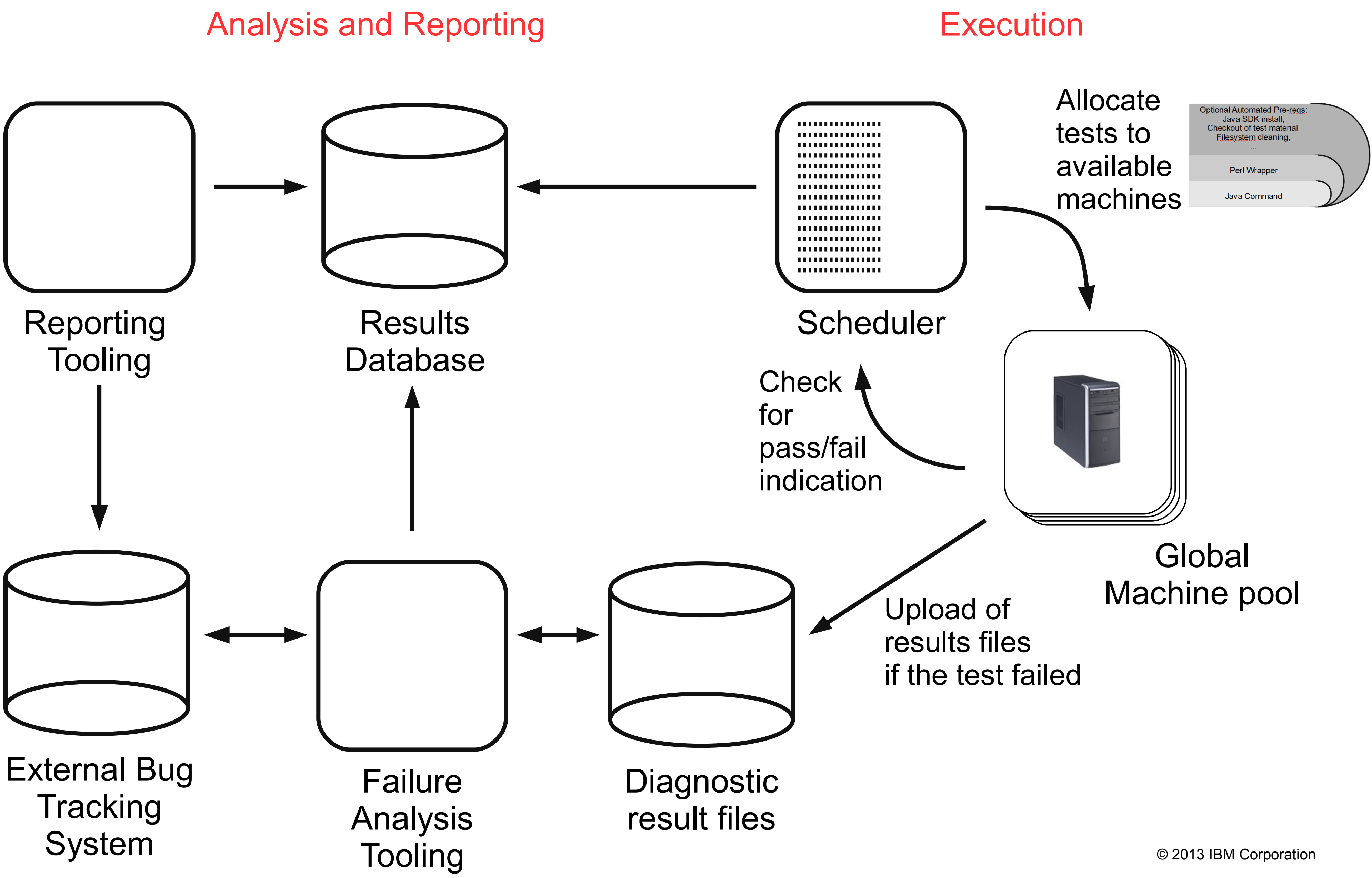
- How can we optimize the time and effort spent doing this?

- Our Solution
  - Continual execution of tests 24x7
  - Develop tools and services to maximize throughput and minimize manual effort

Challenges – a closer look

- Test Execution
  - Size and nature of the product
  - Diversity of supported environments
  - Distributed organization (team and hardware)
  - Optimizing use of computing resources
  - People costs: submitting and running tests
- Analysis and Status Reporting
  - People costs: analyze and categorize failures
  - Reporting partial test status

Test Automation – Overview



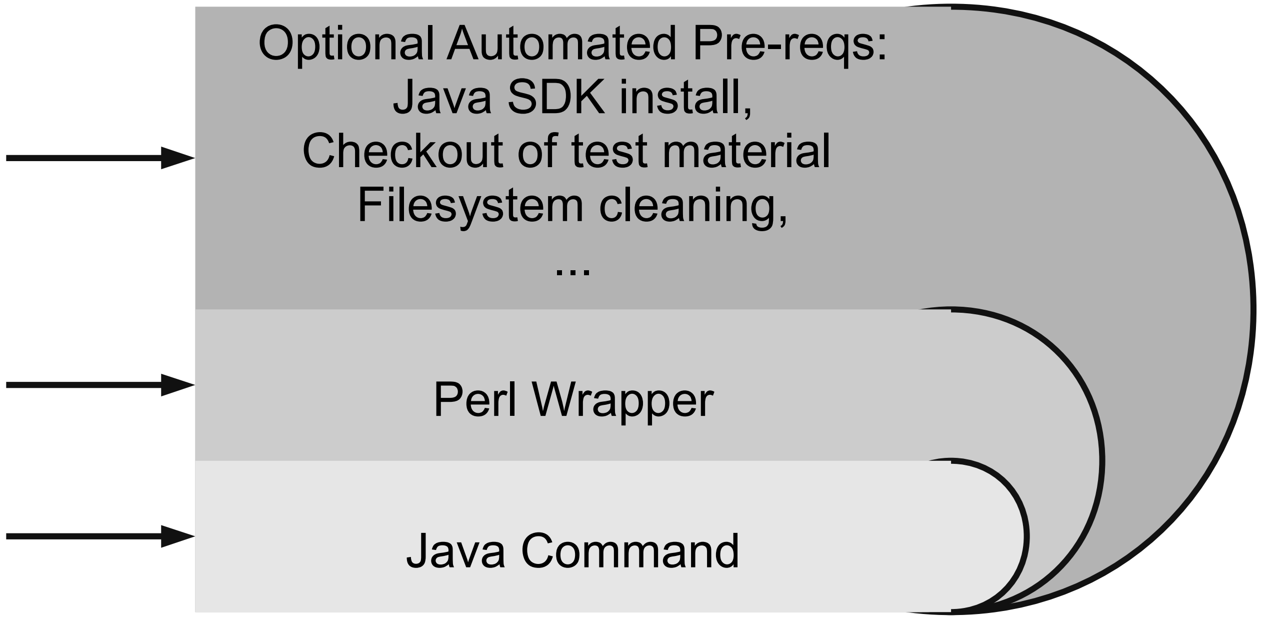
Analysis and Reporting
Execution
Reporting Tooling
Results Database
Scheduler
Allocate tests to available machines
Optional Automated Pre-reqs: Java SDK install, Checkout of test material Filesystem cleaning, ...
Perl Wrapper
Java Command
Check for pass/fail indication
Global Machine pool
Upload of results files if the test failed
External Bug Tracking System
Failure Analysis Tooling
Diagnostic result files

Introduction to 'The Onion' – Layers of Automation

Designing a test framework without becoming tied to one test environment

Individual testcases can be initiated at each of the 3 levels shown by arrows.

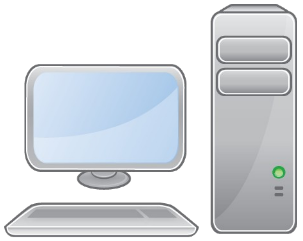
This facilitates any mode of operation, from fully-automated regression testing, all the way down to customising the Java command for defect debugging



Extra arguments can be added in at any level, passed down to the appropriate command lines.

Large, monolithic, test environments are not flexible enough to be passed between teams and do not lend themselves to a quick defect turnaround.

Challenge #1 – Size and nature of the product



- The Java runtime isn't an app! It's more like part of an OS
- Applications drive the runtime in many different ways
- Java provides an ***extremely*** broad API
  - **>7000** java.* and javax.* classes in rt.jar alone
  - Many use cases – far too many for one team to write

- Our Solution:
  - Adopt third party test suites in addition to writing our own tests
  - Design a test framework capable of accommodating these test suites

Solution – Adopt third party test suites

Jython

Scala

Apache Harmony

Derby

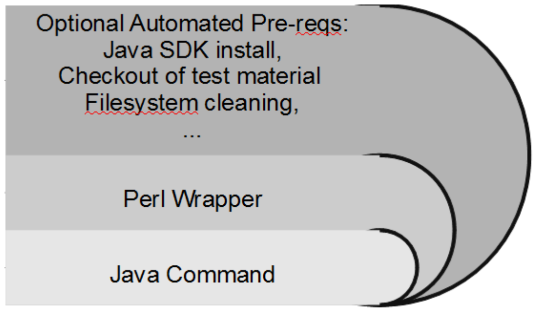
JRuby

IBM Application Testing

Eclipse

Mauve

Solution – Core of 'The Onion': the Java command



- The raw test command:

```
java <options> <class>
```

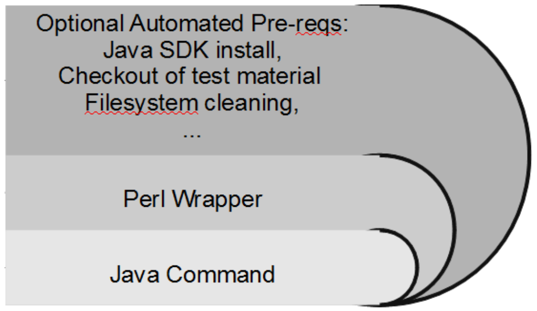
- `<options>` are varied to change the JIT configuration, GC policy or other JVM settings
- Other options include
  - triggering diagnostics
  - increase the likelihood of certain internal operations occurring
- Simple model
  - but in an automated system we need something else checking the outcome of the test

Challenge #2 – Diversity of environments

- 20 platforms (OS + architecture combinations)

- Our Solution:
  - Design the automation framework to accommodate this diversity

Solution – Next layer of 'The Onion': the Perl Wrapper



- A test harness written in Perl wraps the Java testcase
- Handle Platform variety – platform specific configuration
- Watch the Java process for unexpected behaviour
- Determine if the test has passed
- Collect diagnostic and output files, upload to central server
- Requirements naturally led to use of a cross-platform scripting language

Challenge #3 – Distributed organization

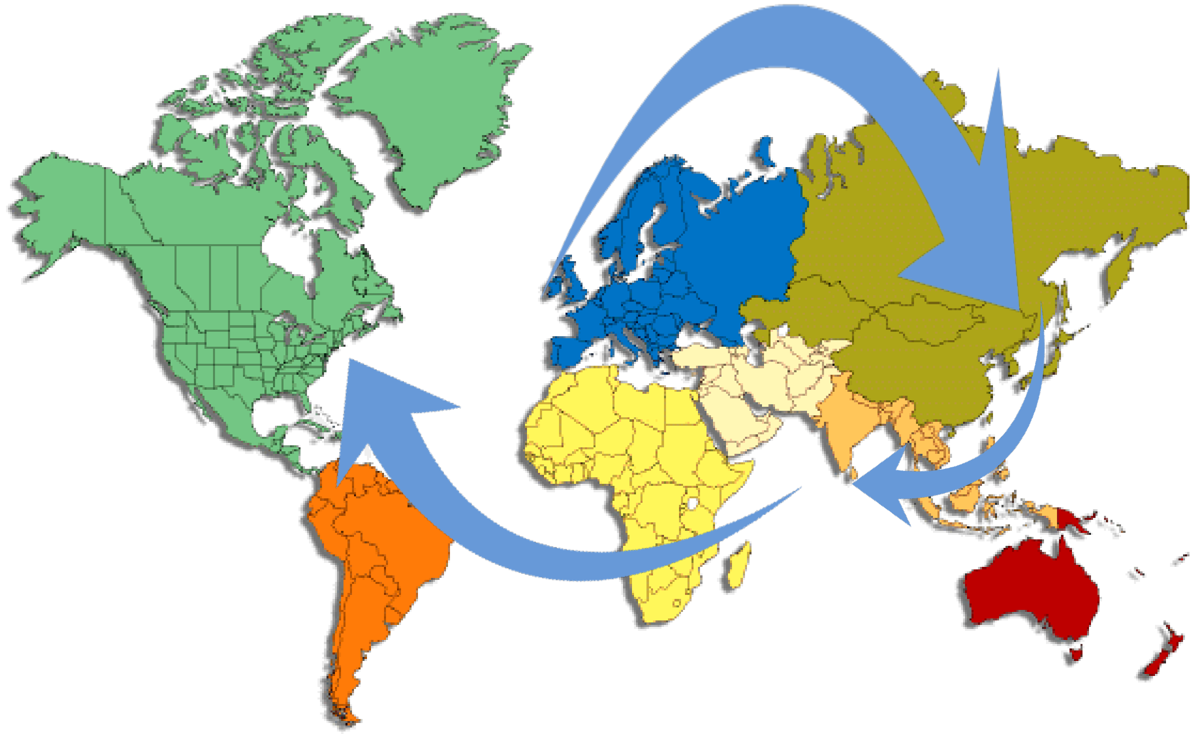
- Distributed organization: team **and** hardware
- Need resources available when and where required
- This really happens, every day:

Challenge #3 – Distributed organization

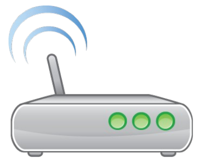
- Distributed organization: team **and** hardware
- Need resources available when and where required
- This really happens, every day:
- **Tester in UK** needs to execute a **test written in China** on an **SDK which was built in India** on a **test machine in New York**

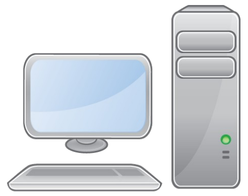
Challenge #3 – Distributed organization

- Distributed organization: team **and** hardware
- Need resources available when and where required
- This really happens, every day:
- **Tester in UK** needs to execute a **test written in China** on an **SDK which was built in India** on a **test machine in New York**



Challenge #4 – Optimizing use of computing resources

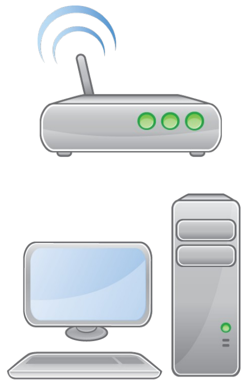




- Network
  - Bandwidth is finite; speed is as slow as the slowest link
  - Test machines are geographically distributed
  - Java SDK development builds are ~500MB
    - 15-20+ minutes download time
  - Need to distribute the test materials also

- CPU
  - Idle CPU cycles are ***wasted*** CPU cycles
  - Even if no new build is available, we will re-run tests
  - Why?

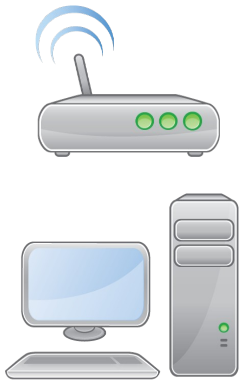
Challenge #4 – Optimizing use of computing resources



- Network
  - Bandwidth is finite; speed is as slow as the slowest link
  - Test machines are geographically distributed
  - Java SDK development builds are ~500MB
    - 15-20+ minutes download time
  - Need to distribute the test materials also

- CPU
  - Idle CPU cycles are ***wasted*** CPU cycles
  - Even if no new build is available, we will re-run tests
  - Why?
  - To keep the office warm?

Challenge #4 – Optimizing use of computing resources



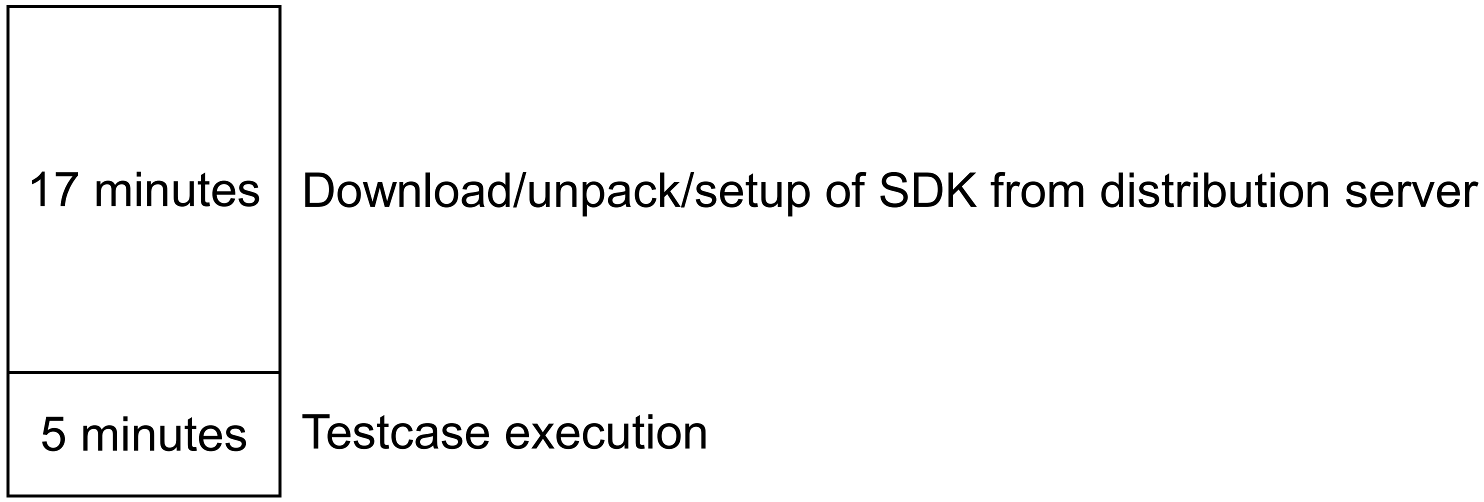
- Network
  - Bandwidth is finite; speed is as slow as the slowest link
  - Test machines are geographically distributed
  - Java SDK development builds are ~500MB
    - 15-20+ minutes download time
  - Need to distribute the test materials also

- CPU
  - Idle CPU cycles are ***wasted*** CPU cycles
  - Even if no new build is available, we will re-run tests
  - Why?
  - To keep the office warm?
  - Detect intermittent failures... 1 in 100 failure rate is not unusual

Effect of a distributed product team on test execution throughput

5 minutes | Testcase execution

Effect of a distributed product team on test execution throughput

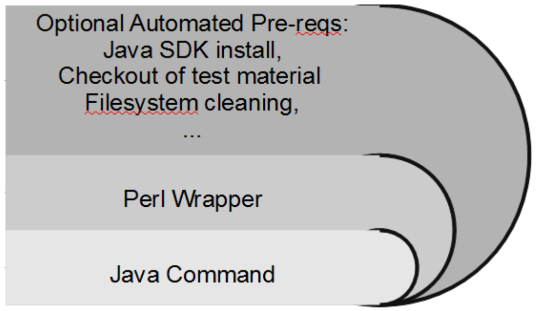


Effect of a distributed product team on test execution throughput

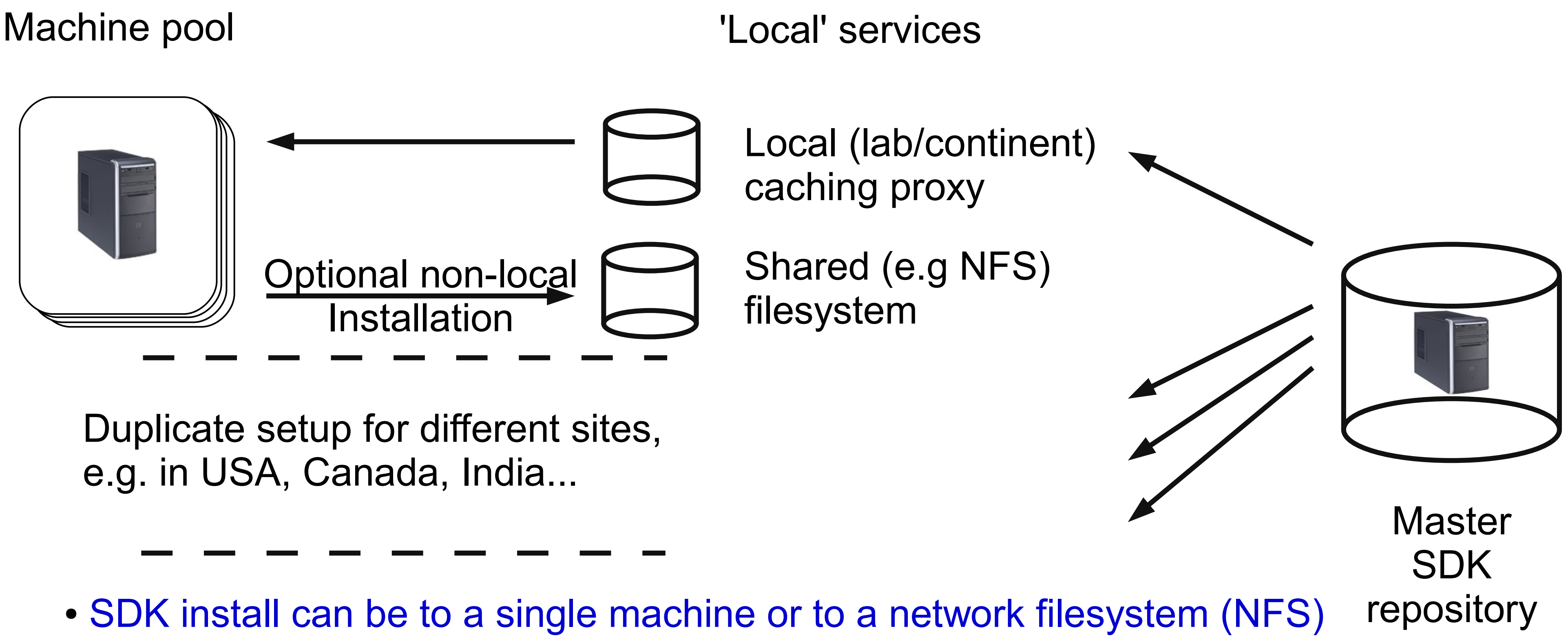
22 minutes

340% reduction in throughput

Our Solution – Outer layer of 'The Onion'



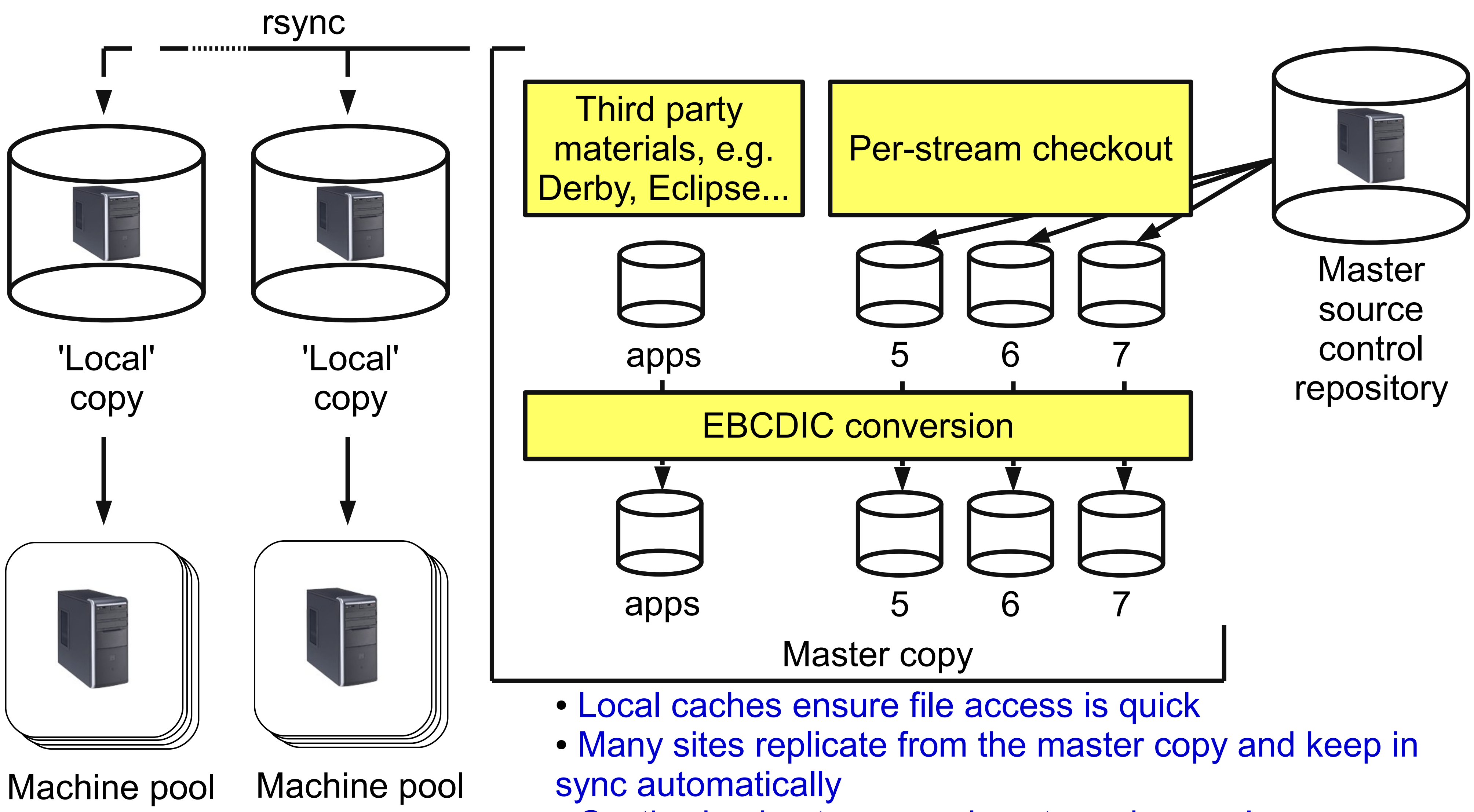
Prerequisites: Installing and Configuring a Java SDK on a test system


Machine pool
'Local' services
Local (lab/continent) caching proxy
Optional non-local Installation
Shared (e.g NFS) filesystem
Duplicate setup for different sites, e.g. in USA, Canada, India...
Master SDK repository

- SDK install can be to a single machine or to a network filesystem (NFS)
- Local caching proxies (“staging”) reduce costly transcontinental transfers
- The source of the SDK can be the master build location, local proxy or NFS
- An asynchronous 'pre-stage' process pushes new builds to each proxy or NFS

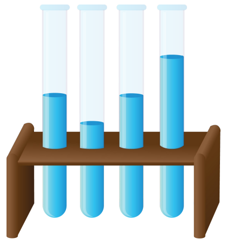
What about the Test material?

Scheduled push to make sure all locations run the current versions of tests



- Local caches ensure file access is quick
- Many sites replicate from the master copy and keep in sync automatically
- Costly checkout, conversion steps done only once

Challenge #5 – People costs: submitting and running tests



- Insufficient people to run tests manually (and it's boring!)

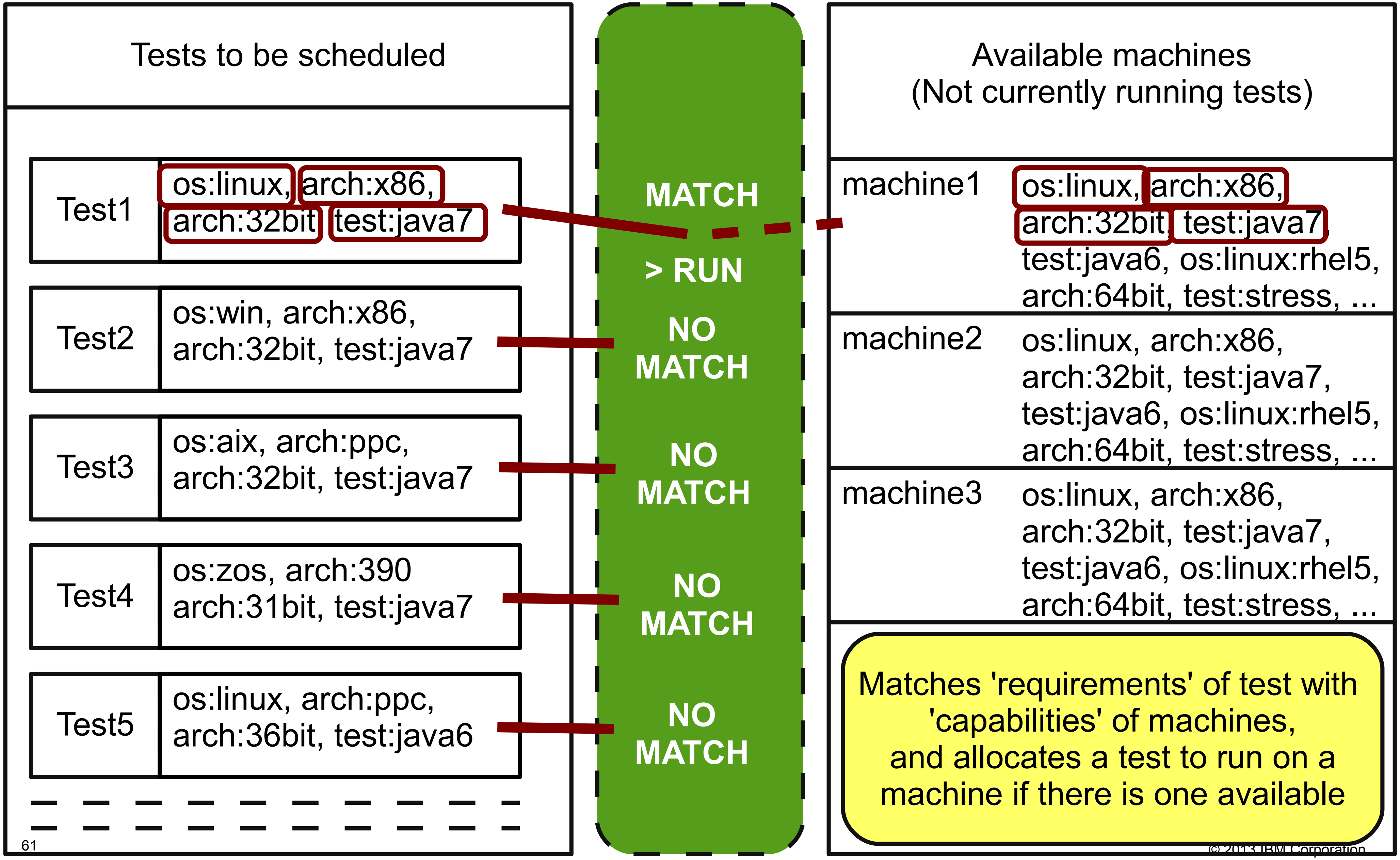
- Our Solution:
  - –Automated policy-based execution of testcases
  - –Continual triggering of testcases based on test machine availability

Solution – Test Execution Automation

The Scheduler

- Runs 24x7
- Assigns tests to idle machines based on
  - requirements of the test
  - the capabilities of the machines
- New runs of tests are submitted as new Java SDKs become available
- Over time, all supported environments are covered
- Optimizes machine usage

Test scheduling

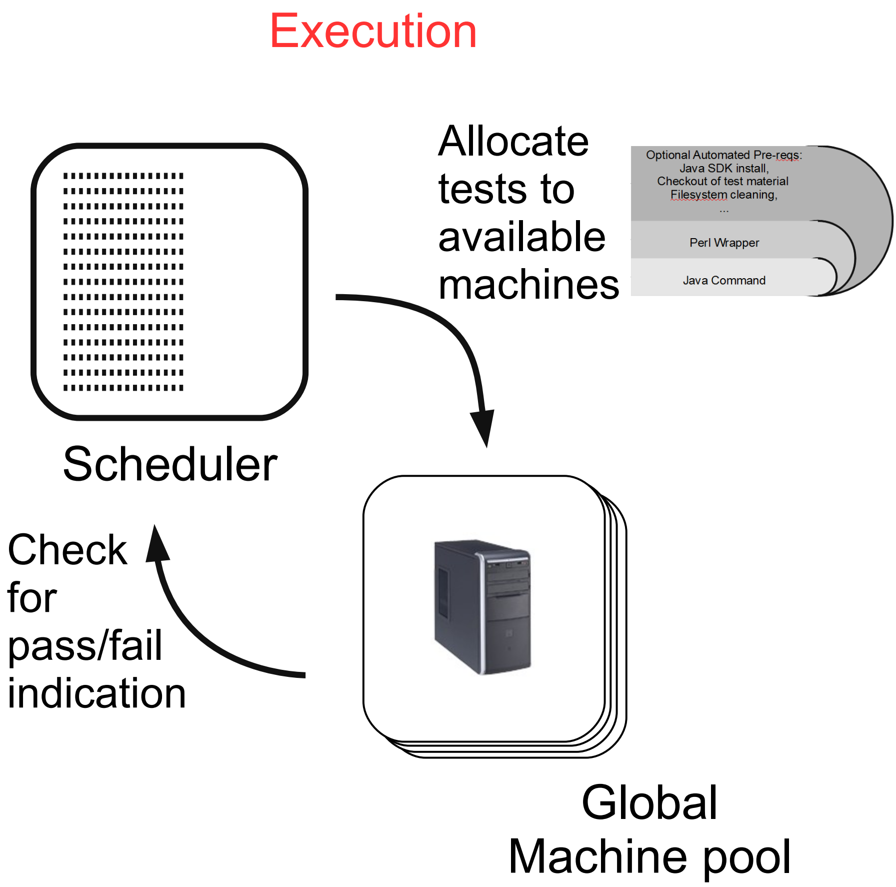


Tests to be scheduled	
Test1	os:linux, arch:x86, arch:32bit, test:java7
Test2	os:win, arch:x86, arch:32bit, test:java7
Test3	os:aix, arch:ppc, arch:32bit, test:java7
Test4	os:zos, arch:390 arch:31bit, test:java7
Test5	os:linux, arch:ppc, arch:36bit, test:java6

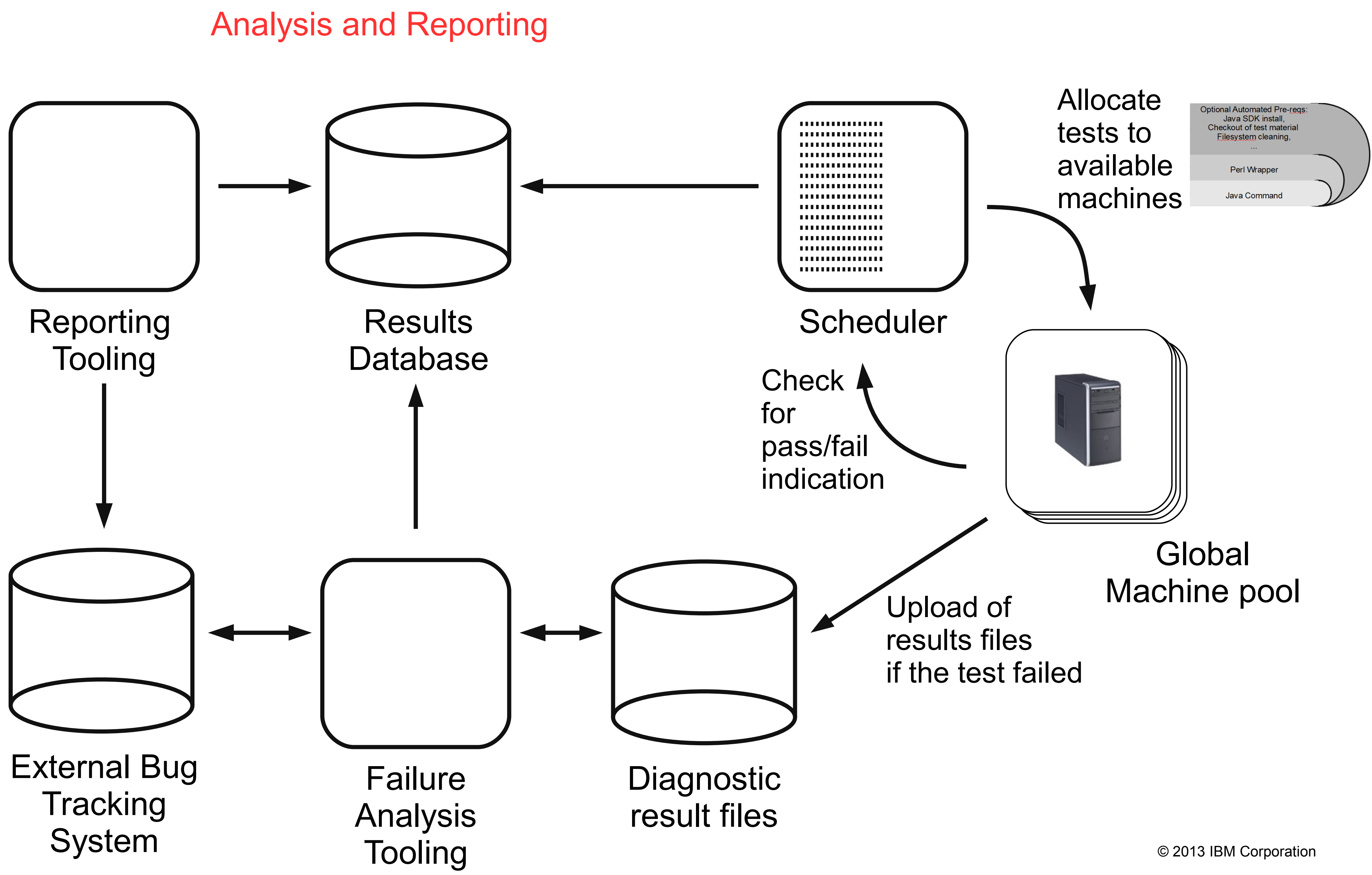
Available machines (Not currently running tests)	
machine1	os:linux, arch:x86, arch:32bit, test:java7, test:java6, os:linux:rhel5, arch:64bit, test:stress, ...
machine2	os:linux, arch:x86, arch:32bit, test:java7, test:java6, os:linux:rhel5, arch:64bit, test:stress, ...
machine3	os:linux, arch:x86, arch:32bit, test:java7, test:java6, os:linux:rhel5, arch:64bit, test:stress, ...

Matches 'requirements' of test with 'capabilities' of machines, and allocates a test to run on a machine if there is one available

Test Automation – Overview


Execution
Allocate tests to available machines
Optional Automated Pre-reqs: Java SDK install, Checkout of test material Filesystem cleaning, ...
Perl Wrapper
Java Command
Scheduler
Check for pass/fail indication
Global Machine pool

Test Automation – Overview



Remember this..?

- *Millions* of test cases executed every month

Remember this..?

- *Millions* of test cases executed every month
  ...and they don't all pass!

Challenge #6 – People costs: analyse and categorise failures

- Hundreds, sometimes ***thousands***, of failures to analyse, debug, categorise
- A single bug may manifest itself in many test failures
- Need to reduce rediscovery of
  - known failures (distributed team)
  - infrastructure problems (distributed hardware)

- Our Solution:
  - Automated failure analysis, using regex-style string pattern matching

Solution – Automated failure analysis

```
AutoTriageRuleFile v0.1

TITLE:
        {
        JIT Crash compiling ReadTask.prepareBuffers()
        }
DESCRIPTION:
        {
        vmState=0x0005ff09 –
        Method_being_compiled=sun/nio/ch/WindowsAsynchronousSocketChannelImpl$ReadTask.prepareB
        uffers()V
        }
TRIGGER:
        {
        + PLAT ##win_x86-32##
        + STRM ##Java7##
        }
CHECK:
        {
        + ##axxonJobOutput.txt##
        ##Method_being_compiled=sun\/nio\/ch\/WindowsAsynchronousSocketChannelImpl##
        }
ACTION:
        {
        PROBLEM_TRACKER:7189
        }
```

Solution – Automated failure analysis

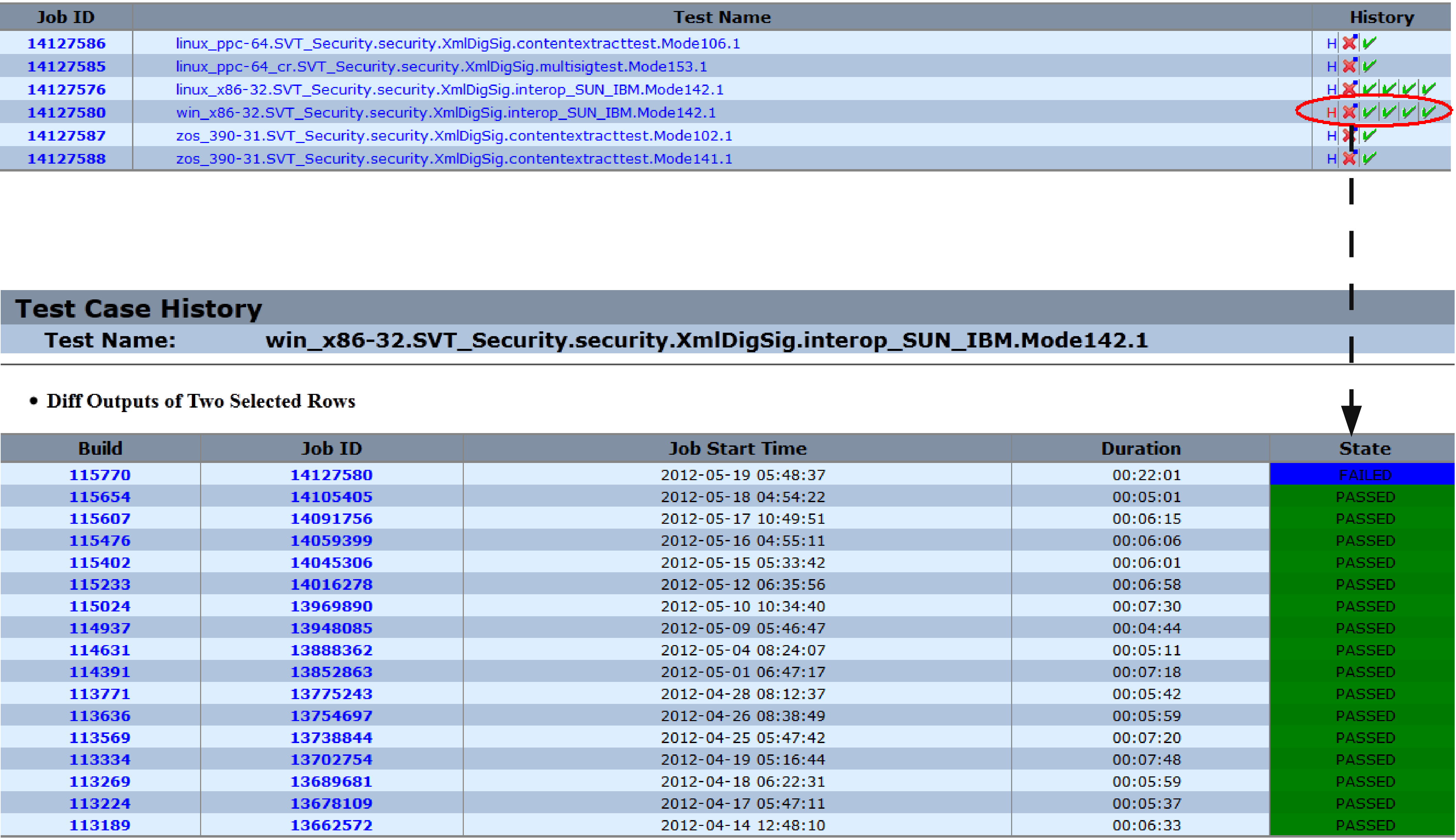
```
AutoTriageRuleFile v0.1

TITLE:
        {
        AcceptPendingEcception not thrown
        }
DESCRIPTION:
        {
        Multiple calls to .accept() on a channel does not throw the expected AcceptPendingException
        }
TRIGGER:
        {
        + STRM ##Java7##
        + TEST ##nio2##
        }
CHECK:
        {
        + ##axxonJobOutput.txt## ##Second call to \.accept\(\) did not throw an
        AcceptPendingException##
        }
ACTION:
        {
        PROBLEM_TRACKER:4159
        }
```

Challenge #7 – Reporting partial test status

- Reporting status for the final build is easy, but during development it is fluid
- Up to 200 builds per day
- 10000s of tests
- Insufficient hardware to run every test on every daily build

- Our Solution
  - Custom reporting tools showing the most recent results for each test and each build

Testcase result history



Job ID	Test Name	History
14127586	linux_ppc-64.SVT_Security.security.XmlDigSig.contentextracttest.Mode106.1	H ✗ ✓
14127585	linux_ppc-64_cr.SVT_Security.security.XmlDigSig.multisigtest.Mode153.1	H ✗ ✓
14127576	linux_x86-32.SVT_Security.security.XmlDigSig.interop_SUN_IBM.Mode142.1	H ✗ ✓ ✓ ✓ ✓
14127580	win_x86-32.SVT_Security.security.XmlDigSig.interop_SUN_IBM.Mode142.1	H ✗ ✓ ✓ ✓ ✓
14127587	zos_390-31.SVT_Security.security.XmlDigSig.contentextracttest.Mode102.1	H ✗ ✓
14127588	zos_390-31.SVT_Security.security.XmlDigSig.contentextracttest.Mode141.1	H ✗ ✓

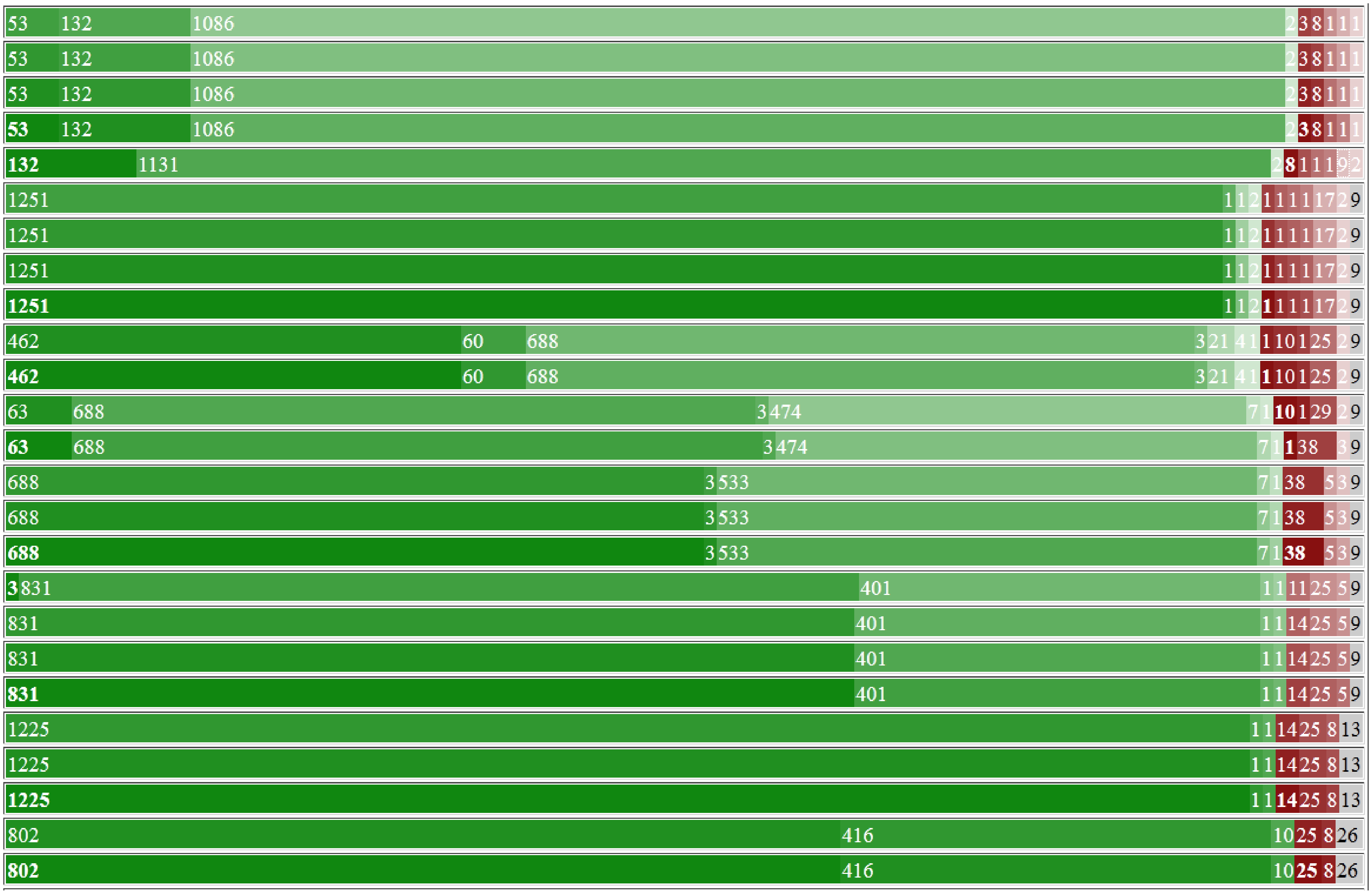
Test Case History

Test Name: **win_x86-32.SVT_Security.security.XmlDigSig.interop_SUN_IBM.Mode142.1**

- **Diff Outputs of Two Selected Rows**

Build	Job ID	Job Start Time	Duration	State
115770	14127580	2012-05-19 05:48:37	00:22:01	FAILED
115654	14105405	2012-05-18 04:54:22	00:05:01	PASSED
115607	14091756	2012-05-17 10:49:51	00:06:15	PASSED
115476	14059399	2012-05-16 04:55:11	00:06:06	PASSED
115402	14045306	2012-05-15 05:33:42	00:06:01	PASSED
115233	14016278	2012-05-12 06:35:56	00:06:58	PASSED
115024	13969890	2012-05-10 10:34:40	00:07:30	PASSED
114937	13948085	2012-05-09 05:46:47	00:04:44	PASSED
114631	13888362	2012-05-04 08:24:07	00:05:11	PASSED
114391	13852863	2012-05-01 06:47:17	00:07:18	PASSED
113771	13775243	2012-04-28 08:12:37	00:05:42	PASSED
113636	13754697	2012-04-26 08:38:49	00:05:59	PASSED
113569	13738844	2012-04-25 05:47:42	00:07:20	PASSED
113334	13702754	2012-04-19 05:16:44	00:07:48	PASSED
113269	13689681	2012-04-18 06:22:31	00:05:59	PASSED
113224	13678109	2012-04-17 05:47:11	00:05:37	PASSED
113189	13662572	2012-04-14 12:48:10	00:06:33	PASSED

Representing overall build test status



Summary

- Investment in test automation pays off
  - IBM delivered Java 7 within **11 weeks** of Oracle Java 7, compared with IBM Java 6 delivered **1 year** after Sun Java 6

- Automation is the **only economic way** to test complex software

- Automation is a more of an intellectual challenge – **and more fun!** – than manual testing

Summary

- Investment in test automation pays off
  - IBM delivered Java 7 within **11 weeks** of Oracle Java 7, compared with IBM Java 6 delivered **1 year** after Sun Java 6

- Automation is the **only economic way** to test complex software

- Automation is a more of an intellectual challenge – **and more fun!** – than manual testing

THANK YOU!

Questions?

References

- **Get Products and Technologies:**
  - IBM Java Runtimes and SDKs:
    - https://www.ibm.com/developerworks/java/jdk/
  - IBM Monitoring and Diagnostic Tools for Java:
    - https://www.ibm.com/developerworks/java/jdk/tools/
- **Learn:**
  - IBM Java InfoCenter:
    - http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp
- **Discuss:**
  - IBM Java Runtimes and SDKs Forum:
    - http://www.ibm.com/developerworks/forums/forum.jspa?forumID=367&start=0

Copyright and Trademarks