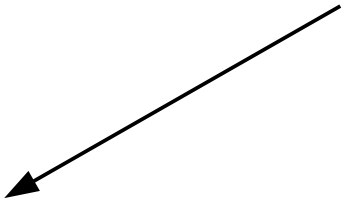

RED HAT
SUMMIT
LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.
June 11-14, 2013
Boston, MA
redhat.

The Bright Future of OpenJDK

Deepak Bhole
Supervisor [Java Group], Red Hat
June 13th 2013

Agenda

Persistent index will always show current point



What is OpenJDK
- Overview
- Components

Red Hat's focus
- Our goals
- Specific areas of focus
  - Support
  - Bridge the gap with proprietary
    - Deployment
    - Serviceability
  - Performance improvements
  - Architectural expansion
    - Zero
    - ARM32
    - AARCH64

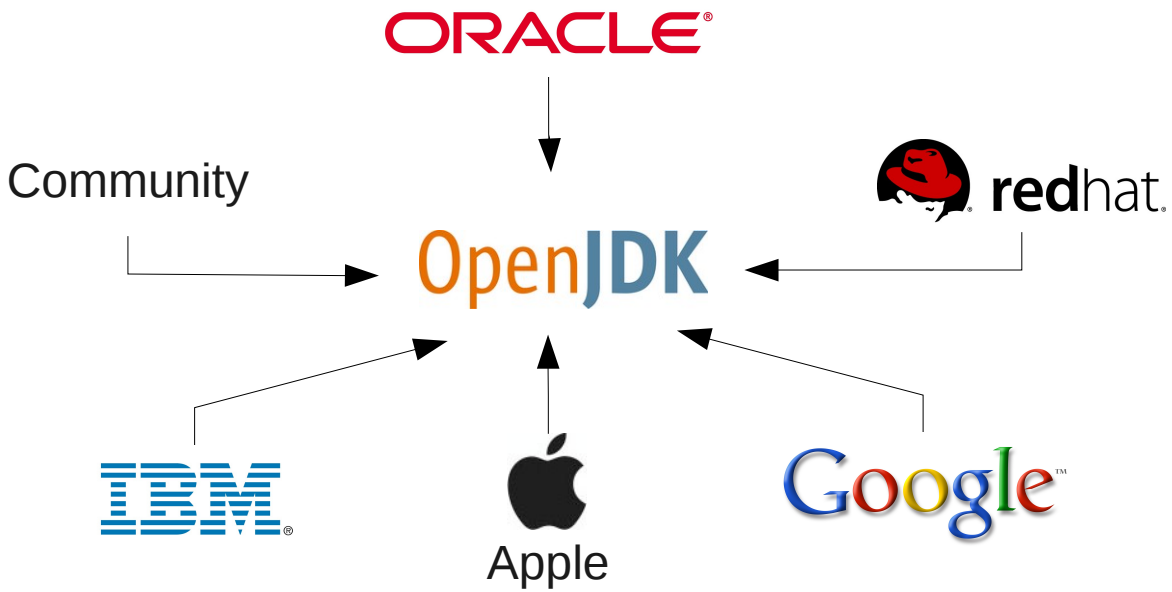
Upcoming features
- OpenJDK 8
  - Lambda
  - Bulk data operations
  - Java type annotations
  - Parameter name access
  - Parallel array sorting
  - Perm-gen removal
  - Date and Time API
  - Nashorn
- OpenJDK 9
  - Jigsaw
  - Other potential features

Questions?

OpenJDK

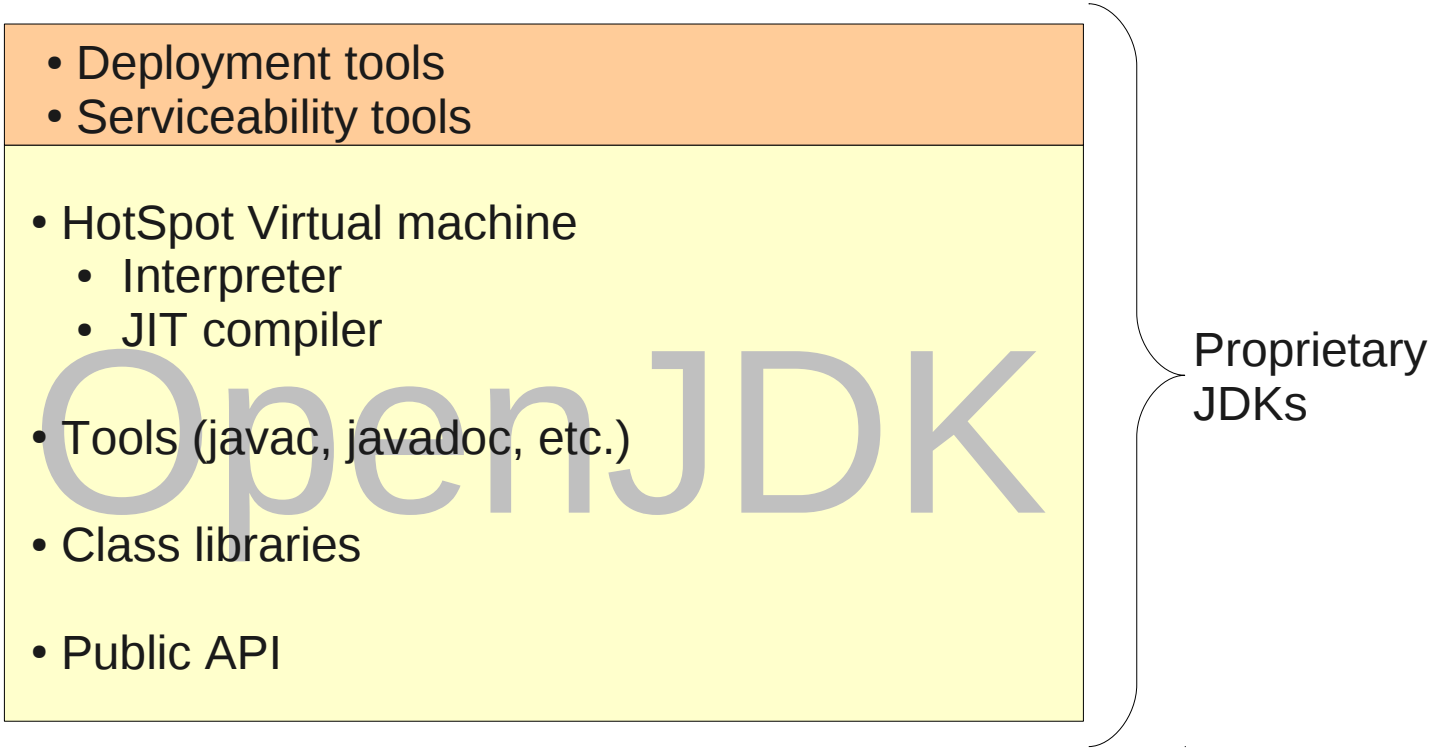
- The only 100% complete Open Source Java implementation
- Fedora and RHEL versions of OpenJDK 6 and OpenJDK 7 fully pass TCK6 and TCK7 respectively
- Basis of the proprietary Oracle JDK (95% shared code)
- Default Java implementation in Fedora, Red Hat Enterprise Linux, Ubuntu, SuSE and numerous other Linux distributions

OpenJDK Contributors



All company logos are trademarks of their respective owners

Proprietary JDKs vs OpenJDK



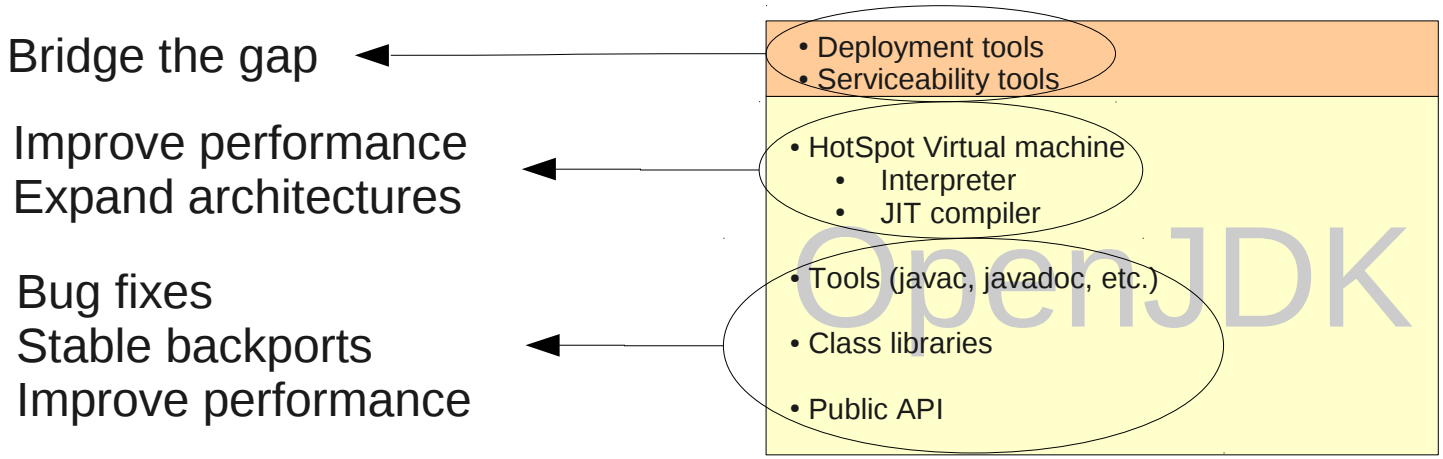
Red Hat and the Java platform

- Heavily invested in OpenJDK
- We want to make OpenJDK one of the top development platforms by means of:
  - Ensuring that customers receive full support for their deployments
  - Ensuring that Java is ready to handle tomorrow's computing needs
  - Ensuring that Java is ready to run on next generation hardware
  - Doing all of the above while balancing security, reliability, and feature set

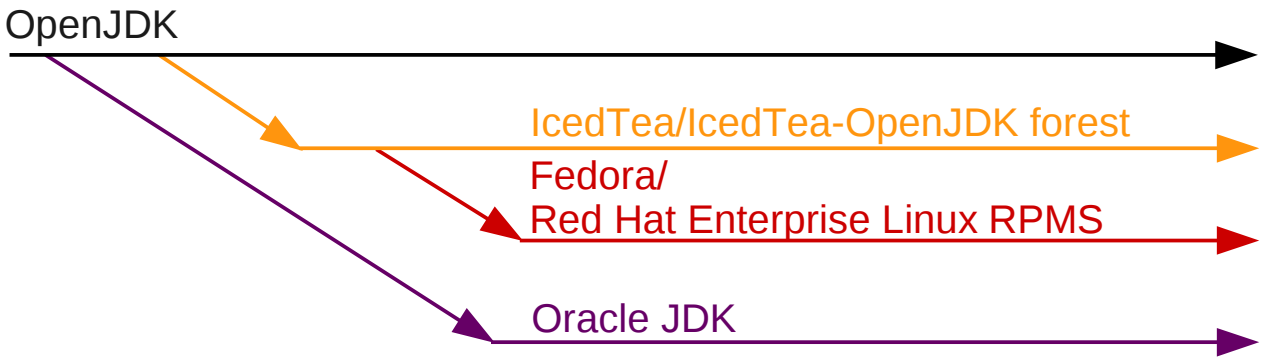
Red Hat's current focus

Support

+



Support (distribution)

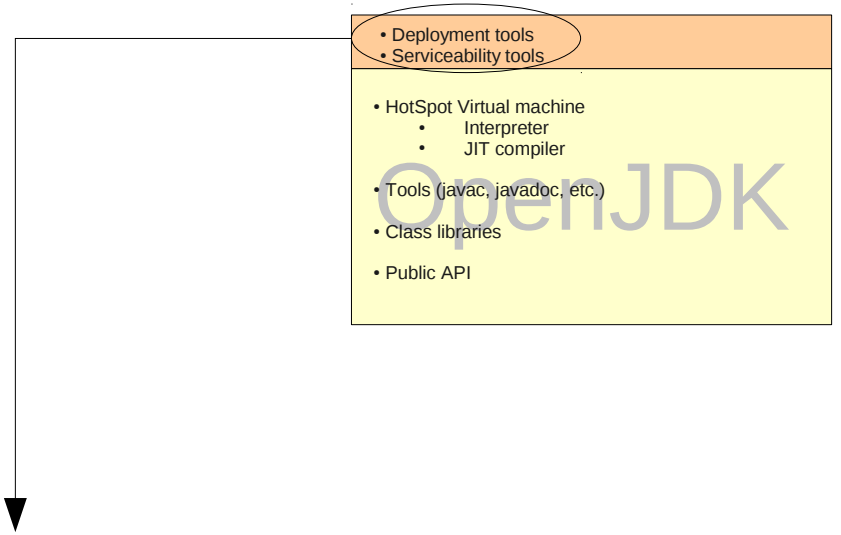


- Red Hat accounts for 80+% contribution to IcedTea/IcedTea-forest (on top of base OpenJDK)
- Red Hat accounts for 100% contribution to Fedora/RHEL packaging
- Early previews: OpenJDK 8 in Fedora 19

Support (OpenJDK 6)

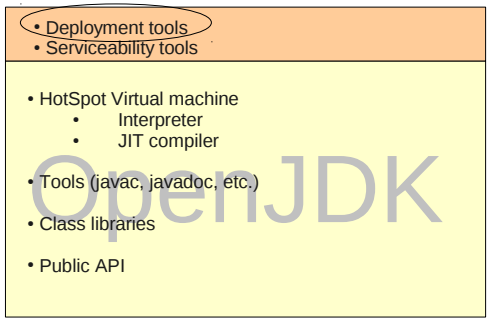
- Red Hat is now the global lead for OpenJDK 6
- Took over after Oracle EOLd it in February 2013
- OpenJDK 6 is in maintenance mode, all security fixes and major bug fixes are applied regularly

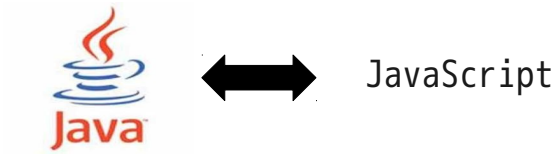
Bridging the gap



Deployment: IcedTea-Web
Serviceability: Thermostat

IcedTea-Web



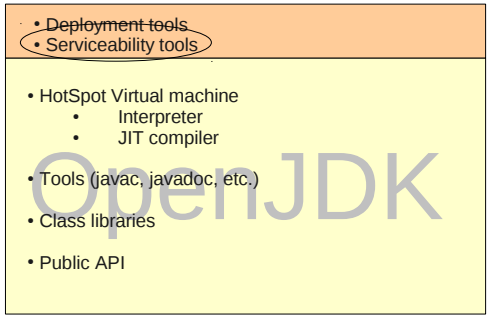


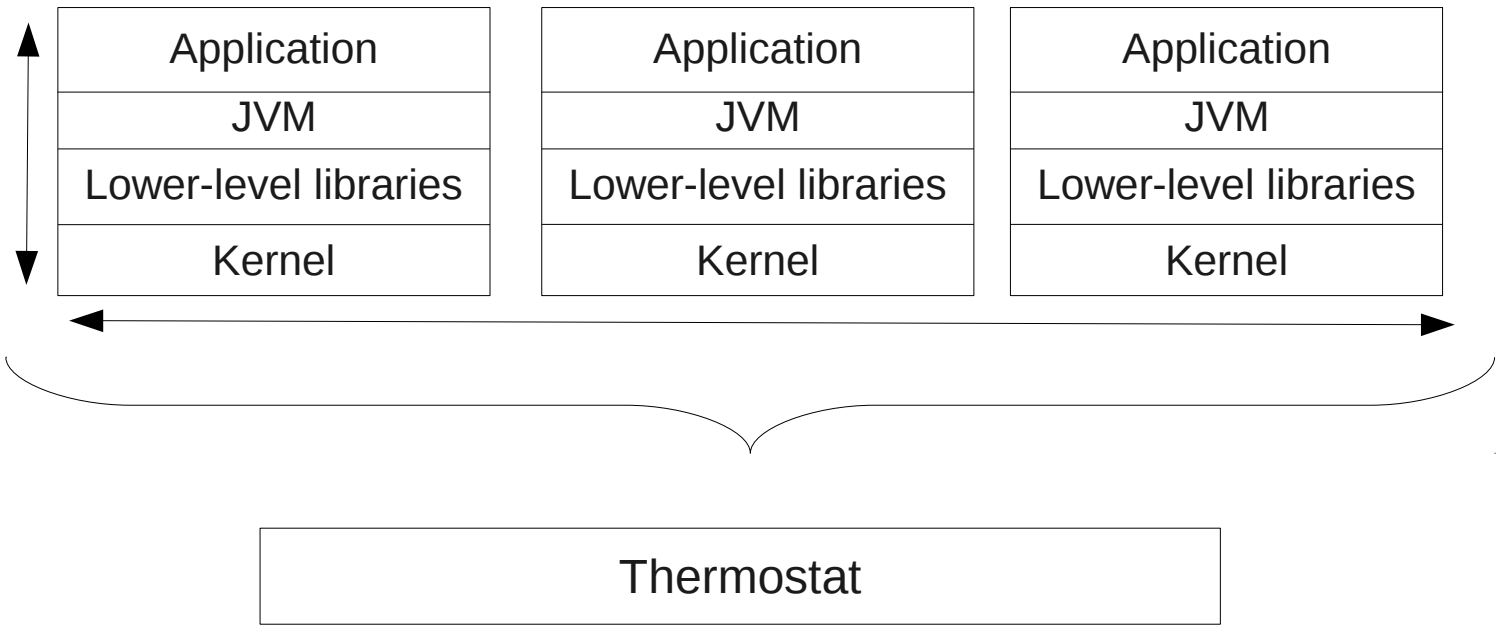
All company logos are trademarks of their respective owners

IcedTea-Web

- 1.3 and 1.4 released over the past 12 months
- Numerous bug fixes
- Improved Epiphany and Midori support
- Numerous new features including:
  - Splash screen, with GUI error reporting
  - Bi-directional cookie support
  - Removed GTK reliance
  - Compact download indicator
  - JVM selection option
  - Enhanced security settings dialog
  - And much more...

Thermostat





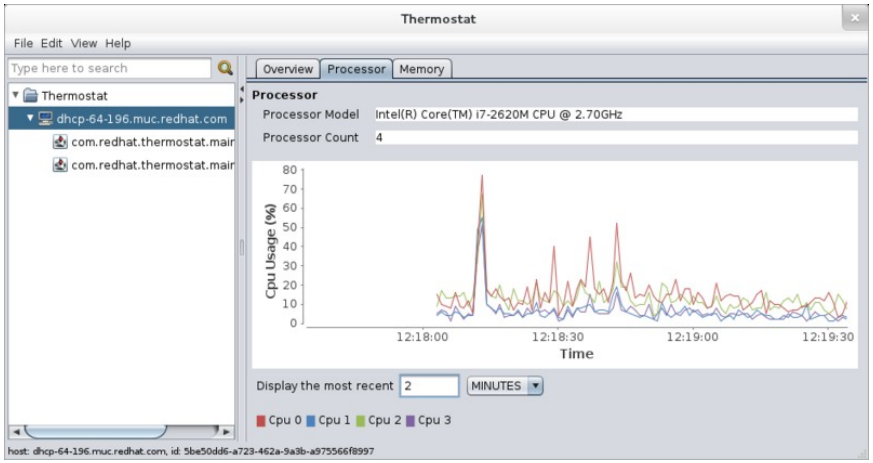
Thermostat

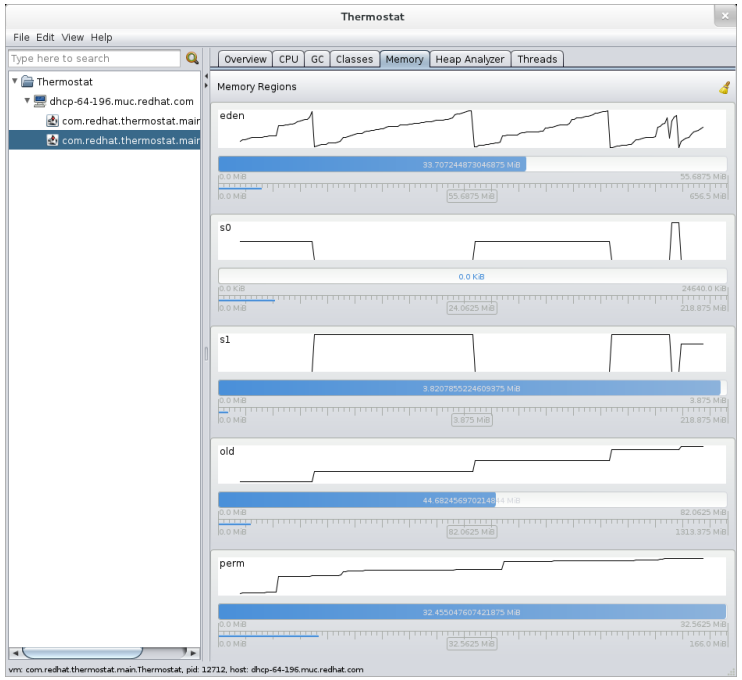
What is OpenJDK
- Overview
- Components

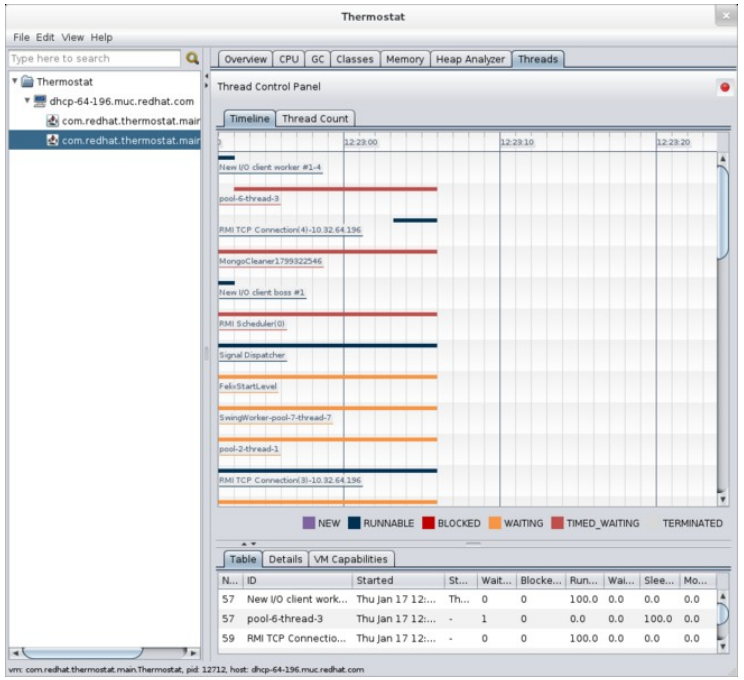
Red Hat's focus
- Our goals
- Specific areas of focus
- Support
- Bridge the gap with proprietary
- Deployment
- Serviceability
- Performance improvements
- Architectural expansion
- Zero
- ARM32
- AARCH64

Upcoming features
- OpenJDK 8
- Lambda
- Bulk data operations
- Java type annotations
- Parameter name access
- Parallel array sorting
- Perm-gen removal
- Date and Time API
- Nashorn
- OpenJDK 9
- Jigsaw
- Other potential features

Questions?







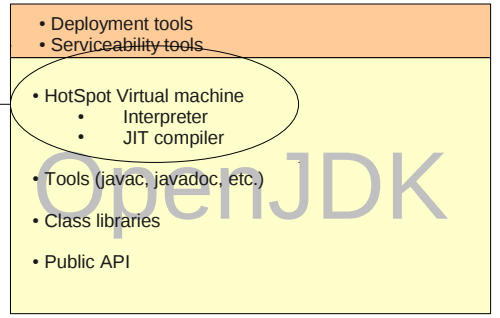
redhat.

Thermostat

CLI interface for scriptability

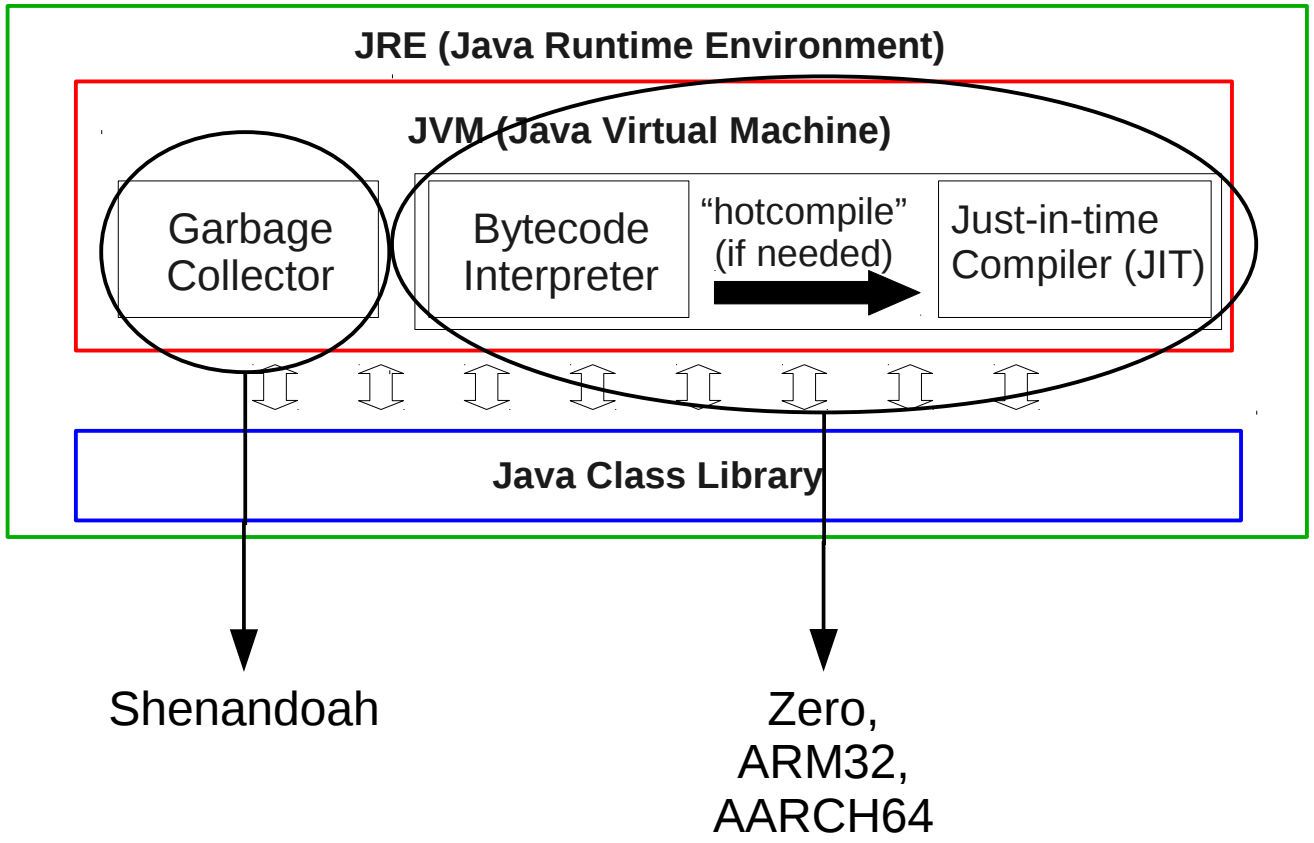
```
Thermostat > object-info --objectId 0x6e5f9dc08 --heapId a952ab0d-2330-46da-a954-
f740361c1cae-2821-1370620094485

Dump file created Fri Jun 07 11:48:13 EDT 2013
Resolving 124623 objects...
Chasing references, expect 24 dots.......................
Eliminating duplicate references.......................
Object ID:      0x6e5f9dc08
Type:           java.net.URL
Size:           104 bytes
Heap allocated: true
References:
                [??] -> java.lang.Class@0x6e5dea4f8
                [field handler] -> sun.net.www.protocol.jar.Handler@0x6e5f62858
                [field path] -> java.lang.String@0x6e5f9dc48
                [field authority] -> java.lang.String@0x6e5c118b8
                [field path] -> java.lang.String@0x6e5f9dc48
                [field authority] -> java.lang.String@0x6e5c118b8
                [field protocol] -> java.lang.String@0x6e5c50360
Referrers:
                sun.misc.URLClassPath$JarLoader@0x6e5f9dbd8 -> [field base]

Thermostat > _
```

Improve performance, Expand architectures

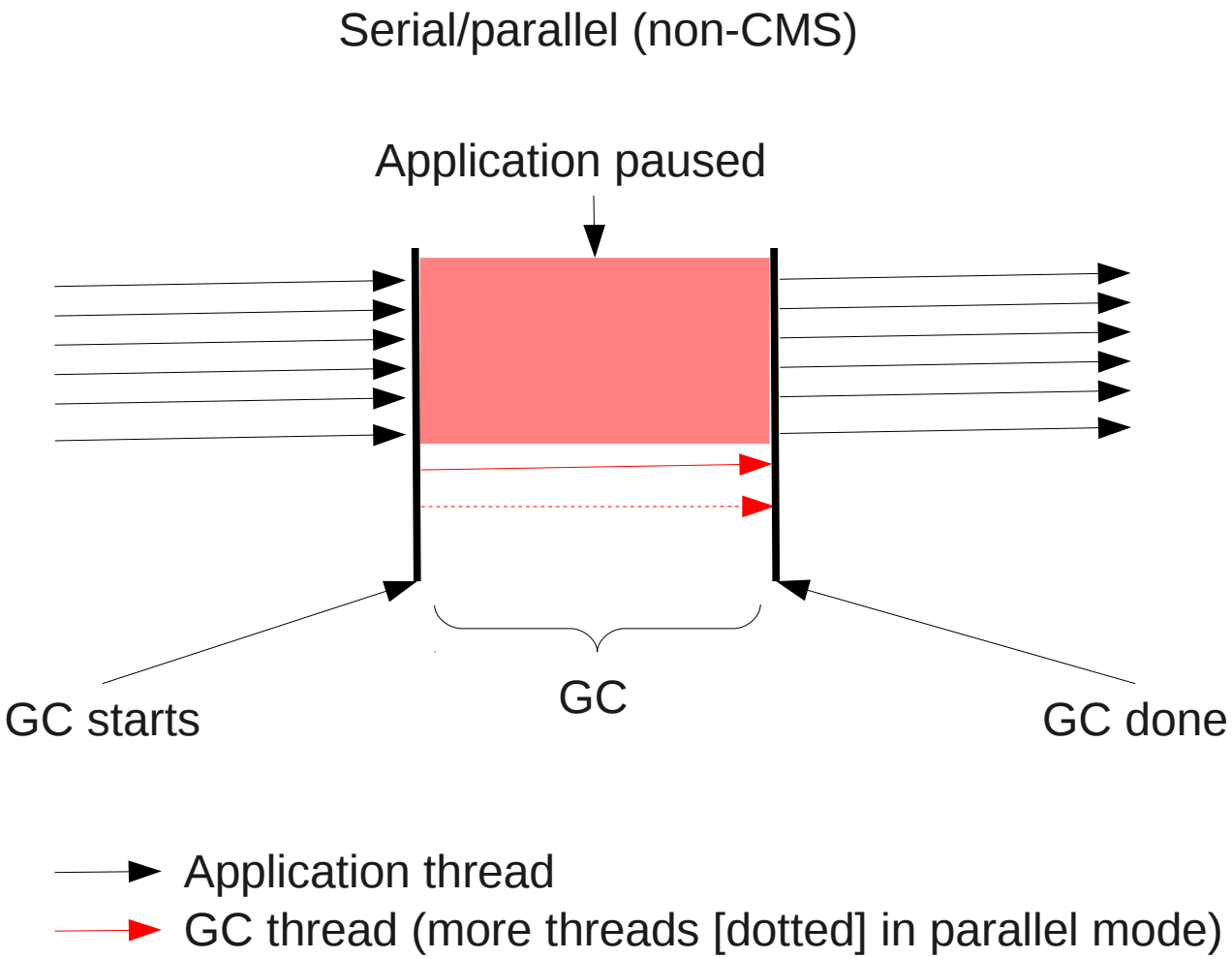


Performance: New GC: Shenandoah
Expand platforms: Zero, ARM32, AARCH64

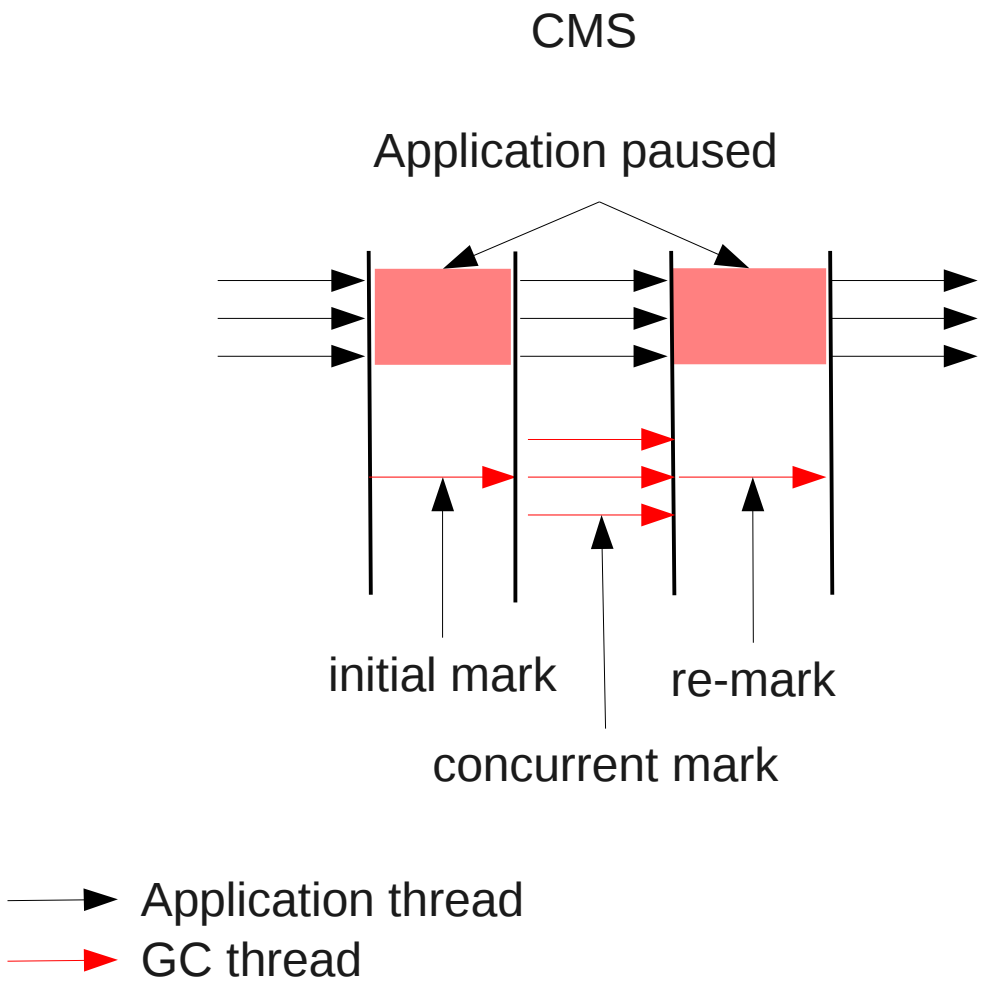
Improve performance, Expand architectures



Current Garbage Collectors



Current Garbage Collectors



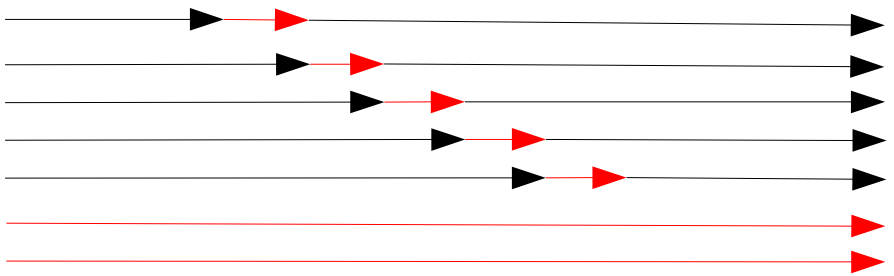
Current Garbage Collectors

G1

- Replacement for CMS
- Parallelizes operations like CMS
- Divides memory into regions and sacrifices some memory (book-keeping) to track region liveness
- Tries to free regions with mostly dead objects
- Performs compaction regularly, unlike CMS
- Like CMS, subject to stop-the-world pauses
- Targets heaps 6+GB, with 500ms pause time*

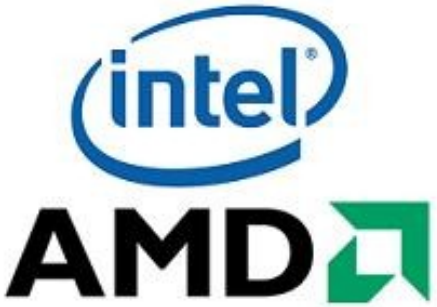
* http://docs.oracle.com/javase/7/docs/technotes/guides/vm/G1.html

Shenandoah



- Currently being developed solely by Red Hat
- Region based
- Parallel, concurrent marking
- Parallel, concurrent evacuation
- Target is 200 GB heaps with a pause time of < 10ms

Multi-arch support via Zero



Java interpreter written in C++, for any platform where GCC is available

All company logos are trademarks of their respective owners

ARM32 JIT

- Based on an older Thumb2 JIT from ARM
- Updated to work with OpenJDK and multiple ARM CPUs by Red Hat

AARCH64

- AARCH64 is still in its infancy
- No hardware available yet
- Very different from current gen ARM (unlike x86_64 vs x86)

http://www.arm.com/images/AARCH64.jpg

AARCH64

- Need a solution in place by the time hardware rolls out
- Red Hat is leading the OpenJDK AARCH64 port
- Template interpreter is reasonably complete*
- C1 core functionality done*
- C2 core operations now working*
- C2 optimization is the next major target

** Since hardware is not yet available, completion is measured against a simulator written based on the official AARCH64 specification*

Upcoming features in Java

Feature ↓ … Java version →	Java 8	Java 9
JSR-335: Lambda	●	●
JSR-335: Bulk Data operations	●	●
JSR-308: Java Type Annotations	●	●
JEP-118: Access to parameter names at runtime	●	●
JEP-103: Parallel Array Sorting	●	●
JEP-122: Perm-gen removal	●	●
JSR-310: Date and Time API	●	●
JEP-174: Nashorn	●	●
JSR-294: Jigsaw		●
Large heap support (Shenandoah?)		●

Java 9 features are subject to change as they are under heavy development

Highlights – Java 8

JSR-335: Project Lambda

- Closures for Java
- Will help get rid of many anonymous classes
- Simple interface that requires no changes to existing declarations
- Non-final variables can be captured
- No more confusion around what "**this**" refers to
- Will also introduce default methods (which will allow interface extension without breaking compatibility)
- Syntax:

```
Runnable r = () -> System.out.println("Hello World");
```

  - Above will initialize an instance of runnable such that its run() method prints "Hello World"

Highlights – Java 8

JSR-335: Bulk data operations

- Will leverage lambda
- Provide new collection methods to parallelize operations
- e.g.

```
Object o = findFirstNotNull(l);
public Object findFirstNotNull(List l) {
      for (Object o: l) {
            if (o != null) {
                  return o;
            }
      }
}
```

```
Object o = findFirstNotNull(l);
public Object findFirstNotNull(List l) {
      l.stream().
            filter(o -> o != null).
            findFirst();
}
```

```
Object o = l.stream().filter(o -> o != null).findFirst();
```

Highlights – Java 8

JSR-308: Type Annotations

- Allows usage of annotations in more places, including generic types. For example:
  - `List<@NotNull String> l = new ArrayList<String>();`
- Backward compatible (by writing annotations in comments)

Highlights – Java 8

JEP-118: Access to parameter names at runtime

- Currently parameter information is available via a fragmented API
- Current API is not reliable
- Goal is to provide a reliable API that provides detailed parameter information

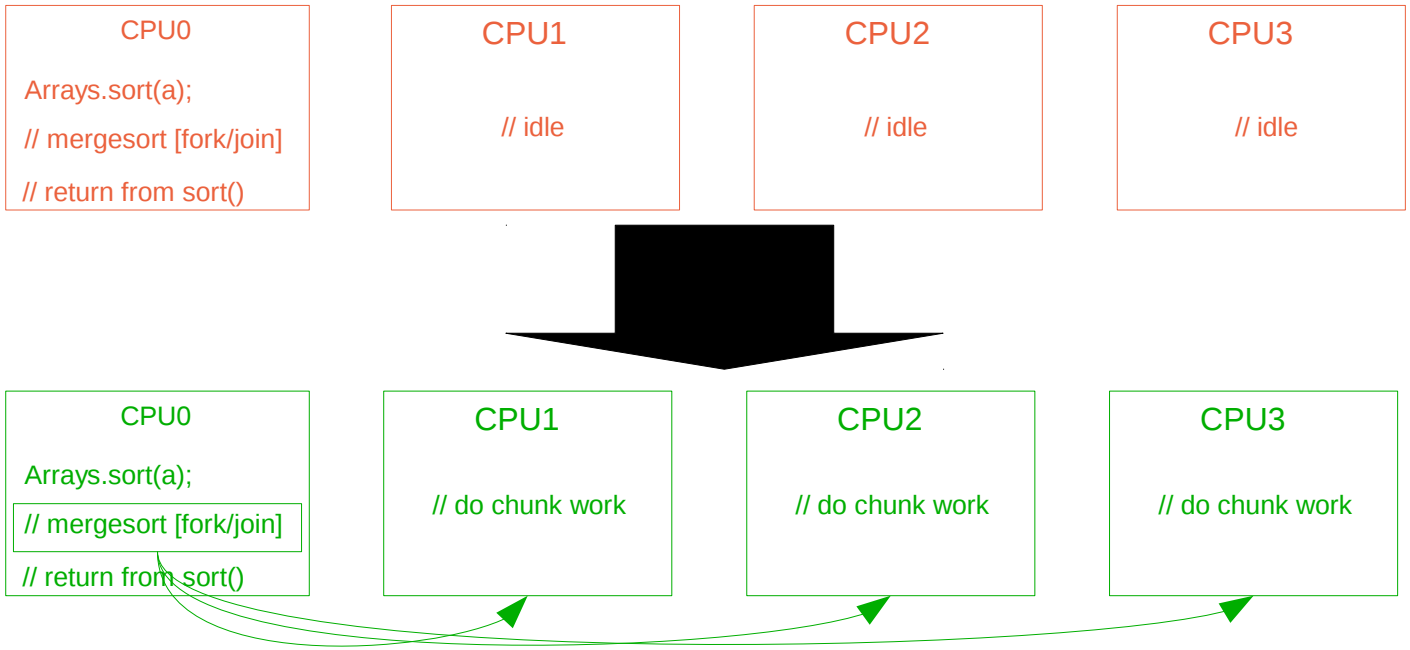
```
public void printParameterInfo(Method m) {
      for (int i=0; i < m.getParameterTypes().length(); i++) {
            // No way to get name
            System.outprintln(m.getParameterTypes()[i]); //type
            for (int j=0; j < m.getParameterAnnotations[i].length(); j++) {
                  System.outprintln(m.getParameterAnnotations[i][j]);
            } // annotations
      }
}
```

```
public void printParameterTypesInfo(Method m) {
      for (Parameter p: m.getParameters()) {
            System.out.println(p.getName()); // name
            System.out.println(p.getType()); // type
            for (Annotation a: p.getAnnotations) {
                  System.outprintln(a);
            } // annotations
      }
}
```

Highlights – Java 8

JEP-103: Parallel Array Sorting

- Current sort method is sequential, doing work in the caller thread
- Currently unable to leverage multi-core CPUs efficiently
- Java 8 will use the new Fork/Join framework to parallelize Array sorting and leverage multiple cores/cpus



Highlights – Java 8

JEP-122: Perm-gen Removal

- Perm-gen stores class metadata
- Once set up during VM initialization, it cannot be expanded
- An issue for VMs that run for a long enough time
- Interned strings moved to heap
- Class metadata and others moved to “metaspace”

Highlights – Java 8

JSR-310: Date/Time API

- A comprehensive API to handle Date/Time related functionality
- Flexible representations of various combinations of Date/Time:
  - Date and Time
  - Date only
  - Time only
- Instants
- Durations
- Clearer representation of Timezone's, UTC offsets, etc.
- And much more!

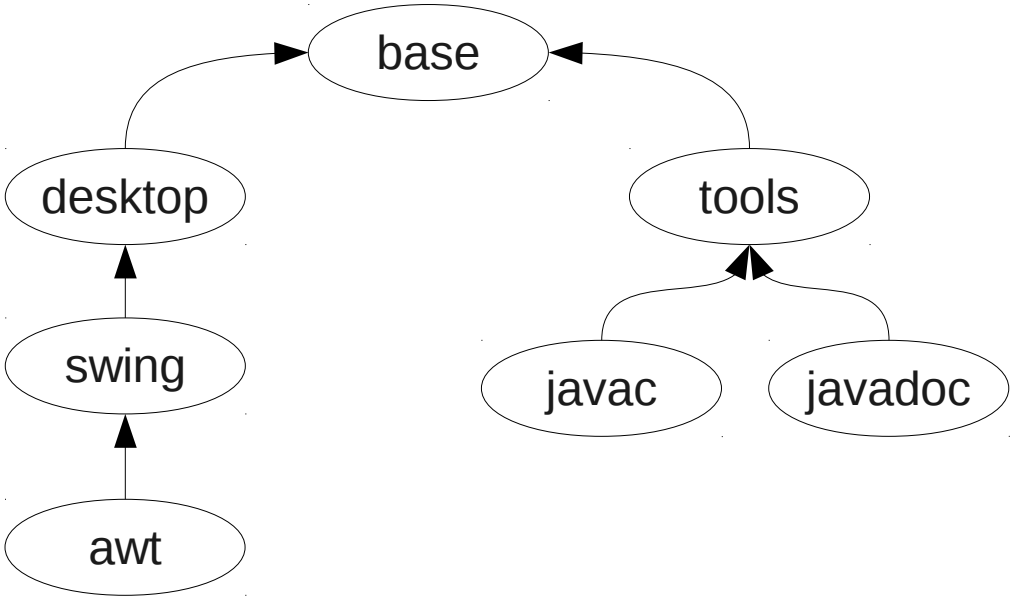
Highlights – Java 8

JEP-174: Nashorn

- JavaScript (JS) engine for the JVM that leverages JSR-292
- Will allow JS access through Java code
- Significantly improved performance over Rhino
- New tool (`jjs`) to allow command-list scripting/script launching
- Supports JSR-223 (javax.script/Scripting Engine for Java)

Highlights – Java 9

JSR-294: Jigsaw



Example to illustrate how Jigsaw can potentially break down components

Highlights – Java 9

- Other potential features:
  - Large heap support
  - Multi-tenancy support
  - Tail call optimizations
  - re-ification

Questions?

OpenJDK: http://openjdk.java.net/
IcedTea: http://icedtea.classpath.org/
IcedTea-Web: http://icedtea.classpath.org/wiki/IcedTea-Web
Thermostat: http://icedtea.classpath.org/wiki/Thermostat

IcedTea Bugs: http://icedtea.classpath.org/bugzilla/
RHEL/Fedora bugs: http://bugzilla.redhat.com

Mailing List: distro-pkg-dev@openjdk.java.net

IRC: irc.oftc.net , #openjdk

Deepak Bhole
dbhole@redhat.com
dbhole on IRC