Down the Rabbit Hole

An adventure in JVM wonderland

Me

- Charles Oliver Nutter
- Red Hat "Research and Prototyping Group"
- JRuby and JVM languages
- JVM hacking and spelunking
- @headius


DRINK

What are we going to do today?

- Look at some interesting Java features
- See how they’re compiled to bytecode
- Watch what the JVM does with them
- Examine the actual native code they become

Why?

Who Are You?

- Java developers?
- Performance engineers?
- Debuggers?
- All of the above?

Details Matter

- Cool features with hidden costs
  - Inner classes
  - Structural types in Scala
  - Serialization
- How code design impacts performance
- What JVM can and can’t do for you

Sufficiently Smart Compiler

“HighLevelLanguage H may be slower than the LowLevelLanguage L, but given a SufficientlySmartCompiler this would not be the case”

http://c2.com/cgi/wiki?SufficientlySmartCompiler

Sufficiently Smart Compiler

If you wait long enough*, the JVM will eventually optimize everything perfectly and even terrible code will perform well.

* for some definition of “long”

Part One: The Primer

Vocabulary

- Source
  - The .java text that represents a program
- Bytecode
  - The binary version of the program that all JVMs can load and execute

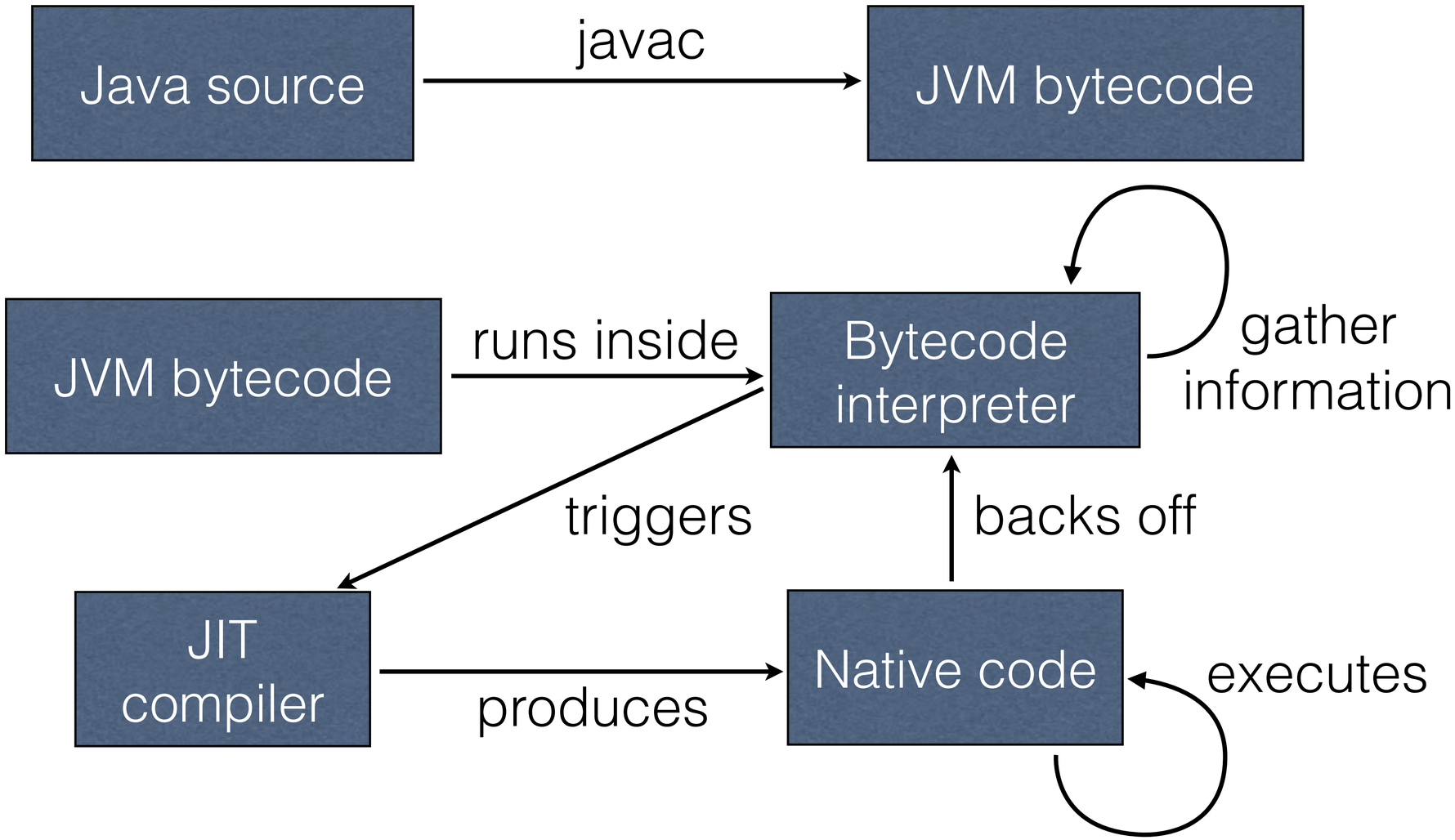
Vocabulary

- Native code
  - Machine code specific to the current platform (OS, CPU) that represents the program in a form the CPU can execute directly
- Heap
  - The JVM-controlled area of memory where Java objects live

Vocabulary

- JIT
  - “Just In Time” (compilation) that turns one program form into a lower program form, e.g. bytecode into native code **at runtime**
- AOT
  - Compilation that occurs **before runtime**

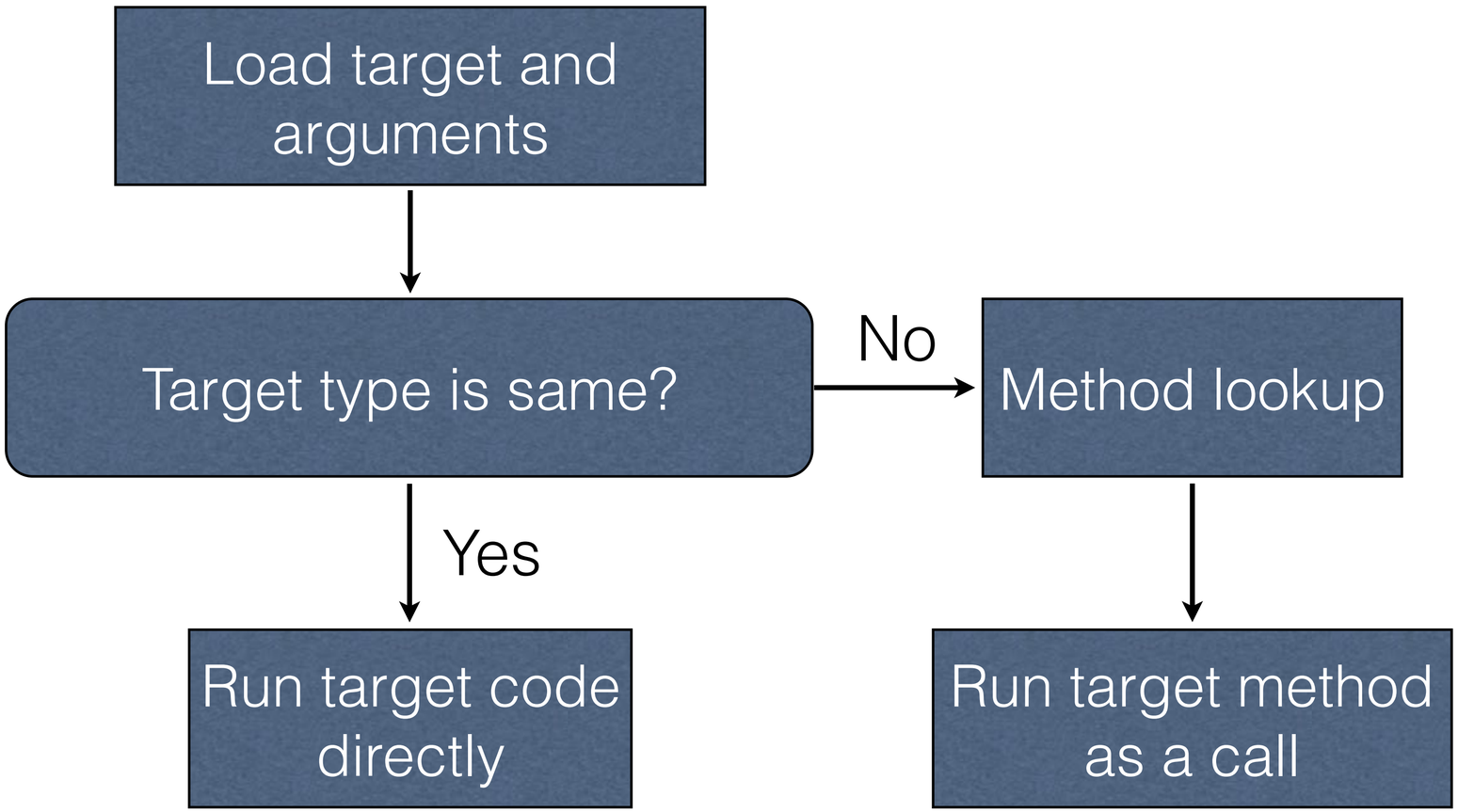
JVM 101


Java source
javac
JVM bytecode
JVM bytecode
runs inside
Bytecode interpreter
gather information
triggers
backs off
JIT compiler
produces
Native code
executes

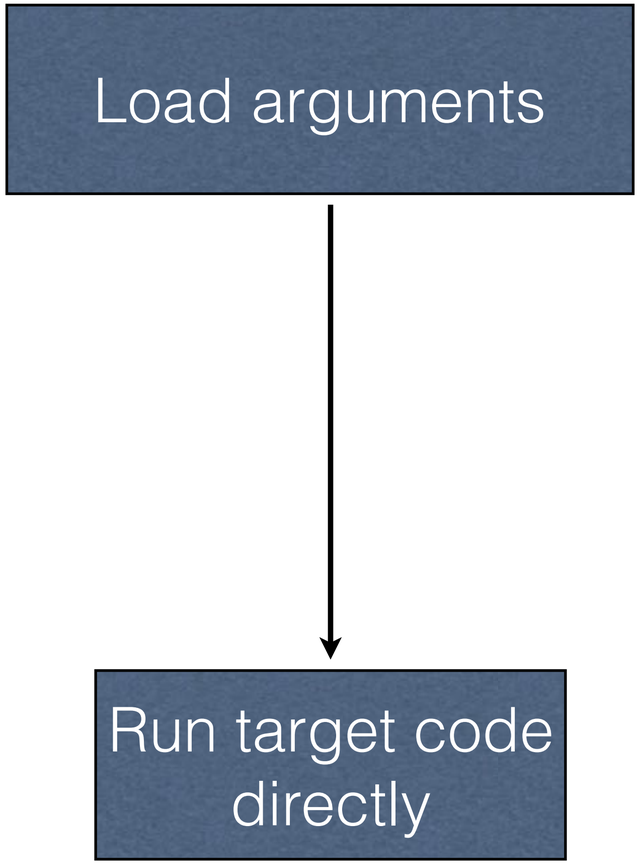
Vocabulary

- Inlining
  - Inserting the code of a called method into the caller, avoiding overhead of the call and optimizing the two together
- Optimization
  - Doing the least amount of work needed to accomplish some goal

Inlining Instance Method



Inlining Static or Special Method



Our Tools

- javac, obviously
- javap to dump .class data
- -XX:+PrintCompilation
- -XX:+PrintInlining
- -XX:+PrintAssembly

Part Two: Down We Go

Hello, world!

- We’ll start with something simple.

```java
package com.headius.talks.rabbithole;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Level 1: Bytecode

- javap
  - Java class file disassembler
  - Dump structure, data, metadata, and code

```
$ javap -cp dist/RabbitHole.jar \
        com.headius.talks.rabbithole.HelloWorld

Compiled from "HelloWorld.java"
public class com.headius.talks.rabbithole.HelloWorld {
  public com.headius.talks.rabbithole.HelloWorld();
  public static void main(java.lang.String[]);
}
```

```
$ javap -cp dist/RabbitHole.jar \
        -c \
        com.headius.talks.rabbithole.HelloWorld

Compiled from "HelloWorld.java"
public class com.headius.talks.rabbithole.HelloWorld {
...
  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2    // Field java/lang/
          System.out:Ljava/io/PrintStream;
       3: ldc           #3    // String Hello, world!
       5: invokevirtual #4    // Method java/io/
  PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

Our First Bytecodes

- getstatic/putstatic - static field access
- ldc - load constant value on stack
- invokevirtual - call a concrete instance method
- return - return from a void method

```
$ javap -cp dist/RabbitHole.jar \
        -c \
        com.headius.talks.rabbithole.HelloWorld

Compiled from "HelloWorld.java"
public class com.headius.talks.rabbithole.HelloWorld {
...
  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2    // Field java/lang/
          System.out:Ljava/io/PrintStream;
       3: ldc           #3    // String Hello, world!
       5: invokevirtual #4    // Method java/io/
  PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

Level 2: Compiler Logs

- -XX:+PrintCompilation
  - Display methods as they compile
- -XX:+PrintInlining
  - Display inlined methods as nested

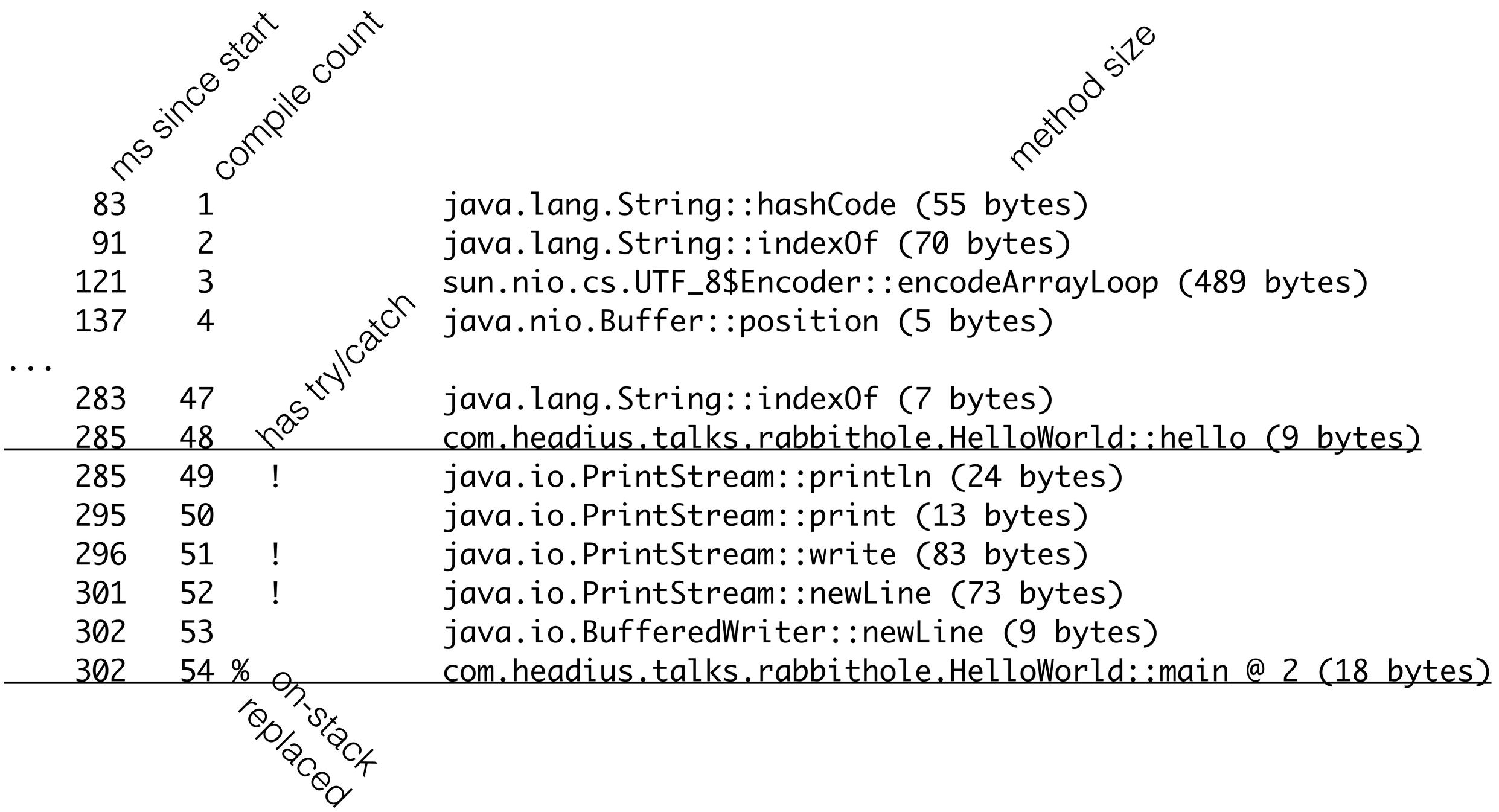
Hotspot JIT

- Code is interpreted first
- After some threshold, JIT fires
- Older Hotspot went straight to “client” or “server”
- Tiered compiler goes to “client plus profiling” and later “server”
  - We will disable tiered compilation

```
public class HelloWorld {
    public static void main(String[] args) {
        for (int i = 0; i < 100000; i++) {
            hello();
        }
    }

    private static void hello() {
        System.err.println("Hello, world!");
    }
}
```

```
$ java -Xbatch
       -XX:-TieredCompilation \
       -XX:+PrintCompilation \
       -cp dist/RabbitHole.jar \
       com.headius.talks.rabbithole.HelloWorld \
       2> /dev/null
```


ms since start
compile count
method size
has try/catch
on-stack replaced
83 1 java.lang.String::hashCode (55 bytes)
91 2 java.lang.String::indexOf (70 bytes)
121 3 sun.nio.cs.UTF_8$Encoder::encodeArrayLoop (489 bytes)
137 4 java.nio.Buffer::position (5 bytes)
...
283 47 java.lang.String::indexOf (7 bytes)
285 48 com.headius.talks.rabbithole.HelloWorld::hello (9 bytes)
285 49 ! java.io.PrintStream::println (24 bytes)
295 50 java.io.PrintStream::print (13 bytes)
296 51 ! java.io.PrintStream::write (83 bytes)
301 52 ! java.io.PrintStream::newLine (73 bytes)
302 53 java.io.BufferedWriter::newLine (9 bytes)
302 54 % com.headius.talks.rabbithole.HelloWorld::main @ 2 (18 bytes)


ms since start
compile count
method size
has try/catch
on-stack replaced
83 1 java.lang.String::hashCode (55 bytes)
91 2 java.lang.String::indexOf (70 bytes)
121 3 sun.nio.cs.UTF_8$Encoder::encodeArrayLoop (489 bytes)
137 4 java.nio.Buffer::position (5 bytes)
...
283 47 java.lang.String::indexOf (7 bytes)
285 48 com.headius.talks.rabbithole.HelloWorld::hello (9 bytes)
285 49 ! java.io.PrintStream::println (24 bytes)
295 50 java.io.PrintStream::print (13 bytes)
296 51 ! java.io.PrintStream::write (83 bytes)
301 52 ! java.io.PrintStream::newLine (73 bytes)
302 53 java.io.BufferedWriter::newLine (9 bytes)
302 54 % com.headius.talks.rabbithole.HelloWorld::main @ 2 (18 bytes)

```
$ java -Xbatch \
       -XX:-TieredCompilation \
       -XX:+PrintCompilation \
       -XX:+UnlockDiagnosticVMOptions \
       -XX:+PrintInlining \
       -cp dist/RabbitHole.jar \
       com.headius.talks.rabbithole.HelloWorld
       2> /dev/null
```

```
     82    1    b         java.lang.String::hashCode (55 bytes)
     94    2    b         java.lang.String::indexOf (70 bytes)
@ 66   java.lang.String::indexOfSupplementary (71 bytes)   too big
    132    3    b         sun.nio.cs.UTF_8$Encoder::encodeArrayLoop (489 bytes)
@ 1   java.nio.CharBuffer::array (35 bytes)   inline (hot)
@ 6   java.nio.CharBuffer::arrayOffset (35 bytes)   inline (hot)
...
    397   48    b         com.headius.talks.rabbithole.HelloWorld::hello (9 bytes)
 !m           @ 5   java.io.PrintStream::println (24 bytes)   inline (hot)
  @ 6   java.io.PrintStream::print (13 bytes)   inline (hot)
...
    446   54 %  b         com.headius.talks.rabbithole.HelloWorld::main @ 2 (18 bytes)
@ 8   com.headius.talks.rabbithole.HelloWorld::hello (9 bytes)
   already compiled into a big method
```

```
     82    1    b         java.lang.String::hashCode (55 bytes)
     94    2    b         java.lang.String::indexOf (70 bytes)
@ 66   java.lang.String::indexOfSupplementary (71 bytes)   too big
    132    3    b         sun.nio.cs.UTF_8$Encoder::encodeArrayLoop (489 bytes)
@ 1   java.nio.CharBuffer::array (35 bytes)   inline (hot)
@ 6   java.nio.CharBuffer::arrayOffset (35 bytes)   inline (hot)
...
    397   48    b         com.headius.talks.rabbithole.HelloWorld::hello (9 bytes)
 !m           @ 5   java.io.PrintStream::println (24 bytes)   inline (hot)
  @ 6   java.io.PrintStream::print (13 bytes)   inline (hot)
...
    446   54 %  b         com.headius.talks.rabbithole.HelloWorld::main @ 2 (18 bytes)
@ 8   com.headius.talks.rabbithole.HelloWorld::hello (9 bytes)
   already compiled into a big method
```

Level 3: Native Code

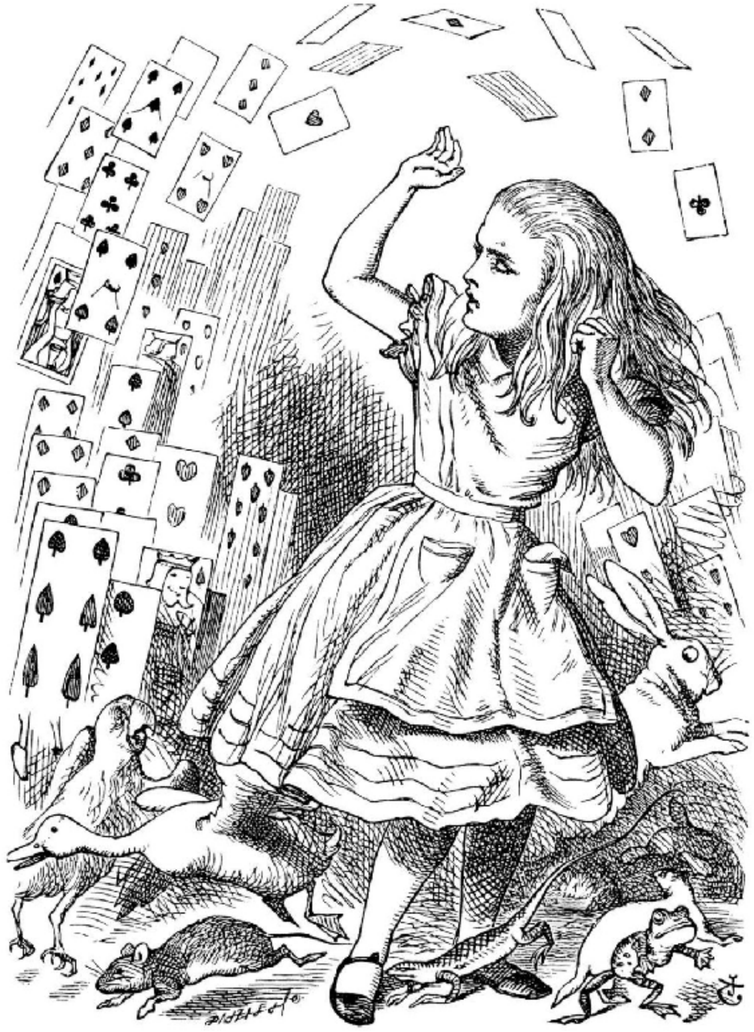
- -XX:+PrintAssembly
  - Dumps “human readable” JITed code
- Google for “hotspot printassembly”
- Aren’t you excited?!

```
$ java -Xbatch \
       -XX:-TieredCompilation \
       -XX:+UnlockDiagnosticVMOptions \
       -XX:+PrintAssembly \
       -cp dist/RabbitHole.jar \
       com.headius.talks.rabbithole.HelloWorld
       2> /dev/null
       | less
```

```
Decoding compiled method 0x0000000110526110:
Code:
[Entry Point]
[Verified Entry Point]
[Constants]
  # {method}
 {0x00000001100a6420} 'hello' '()V' in 'com/headius/talks/rabbithole/HelloWorld'
  #           [sp+0x70]  (sp of caller)
  0x0000000110526300: mov    %eax,-0x14000(%rsp)
  0x0000000110526307: push   %rbp
  0x0000000110526308: sub    $0x60,%rsp         ;*synchronization entry
      ; - com.headius.talks.rabbithole.HelloWorld::hello@-1 (line 13)

  0x000000011052630c: movabs $0x7aaa80c78,%r10  ;   {oop(a 'java/lang/Class' = 'java/lang/System')}
  0x0000000110526316: mov    0x70(%r10),%r11d   ;*getstatic err
      ; - com.headius.talks.rabbithole.HelloWorld::hello@0 (line 13)

  0x000000011052631a: mov    %r11d,0x10(%rsp)
  0x000000011052631f: test   %r11d,%r11d
  0x0000000110526322: je     0x000000011052664e  ;*invokevirtual println
      ; - com.headius.talks.rabbithole.HelloWorld::hello@5 (line 13)

...THOUSANDS OF LINES OF OUTPUT OMITTED
```

Too much!

- Server produces ~2700 bytes of ASM
- Client produces ~594 bytes of ASM
- Most of server output is due to inlining
- More profiling, more code, more perf
  - ...and slower startup



```
public class Tiny1 {
    public static void main(String[] args) {
        for (int i = 0; i < 100000; i++) {
            tiny();
        }
    }

    public static int tiny() {
        return 1 + 1;
    }
}
```

iconst_2: load integer 2 on stack
ireturn: return int

```
public static int tiny();
  Code:
     0: iconst_2
     1: ireturn
```

```
110    3    b         com.headius.talks.rabbithole.Tiny1::tiny (2 bytes)
111    4 %  b         com.headius.talks.rabbithole.Tiny1::main @ 2 (19 bytes)
          @ 8   com.headius.talks.rabbithole.Tiny1::tiny
         (2 bytes)   inline (hot)
```

```
{0x000000010994c3c0} 'tiny' '()I' in 'com/headius/talks/rabbithole/Tiny1'
 #           [sp+0x40]  (sp of caller)
 0x0000000109e566a0: mov    %eax,-0x14000(%rsp)
 0x0000000109e566a7: push   %rbp
 0x0000000109e566a8: sub    $0x30,%rsp          ;*iconst_2
     ; -    com.headius.talks.rabbithole.Tiny1::tiny@0 (line 11)

 0x0000000109e566ac: mov    $0x2,%eax
 0x0000000109e566b1: add    $0x30,%rsp
 0x0000000109e566b5: pop    %rbp
 0x0000000109e566b6: test   %eax,-0x9a05bc(%rip)        # 0x00000001094b6100
     ;   {poll_return}
 0x0000000109e566bc: retq
```

```
{0x000000010994c3c0} 'tiny' '()I' in 'com/headius/talks/rabbithole/Tiny1'
 #           [sp+0x40]  (sp of caller)
 0x0000000109e566a0: mov    %eax,-0x14000(%rsp)
 0x0000000109e566a7: push   %rbp
 0x0000000109e566a8: sub    $0x30,%rsp          ;*iconst_2
     ; -    com.headius.talks.rabbithole.Tiny1::tiny@0 (line 11)

 0x0000000109e566ac: mov    $0x2,%eax
 0x0000000109e566b1: add    $0x30,%rsp
 0x0000000109e566b5: pop    %rbp
 0x0000000109e566b6: test   %eax,-0x9a05bc(%rip)        # 0x00000001094b6100
     ;   {poll_return}
 0x0000000109e566bc: retq
```

```
{0x000000010e67d300} 'main' '([Ljava/lang/String;)V' in 'com/headius/talks/rabbithole/Tiny1'
 0x000000010eb879a0: mov    %eax,-0x14000(%rsp)
 0x000000010eb879a7: push   %rbp
 0x000000010eb879a8: sub    $0x40,%rsp         ;*iconst_0
     ; - com.headius.talks.rabbithole.Tiny1::main@0 (line 5)

 0x000000010eb879ac: mov    $0x0,%esi
 0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
     ; - com.headius.talks.rabbithole.Tiny1::main@2 (line 5)

 0x000000010eb879b6: xchg   %ax,%ax
 0x000000010eb879b8: inc    %esi ; OopMap{off=26}
     ;*goto
     ; - com.headius.talks.rabbithole.Tiny1::main@15 (line 5)

 0x000000010eb879ba: test   %eax,-0x9a08c0(%rip)        # 0x000000010e1e7100
     ;*goto
     ; - com.headius.talks.rabbithole.Tiny1::main@15 (line 5)
     ;   {poll}
 0x000000010eb879c0: cmp    $0x186a0,%esi
 0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
     ; - com.headius.talks.rabbithole.Tiny1::main@5 (line 5)

 0x000000010eb879c8: add    $0x40,%rsp
 0x000000010eb879cc: pop    %rbp
 0x000000010eb879cd: test   %eax,-0x9a08d3(%rip)        # 0x000000010e1e7100
     ;   {poll_return}
 0x000000010eb879d3: retq         ;*return
     ; - com.headius.talks.rabbithole.Tiny1::main@18 (line 8)
```

```
0x000000010eb879a0: mov    %eax,-0x14000(%rsp)
0x000000010eb879a7: push   %rbp
0x000000010eb879a8: sub    $0x40,%rsp          ;*iconst_0
0x000000010eb879ac: mov    $0x0,%esi
0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
0x000000010eb879b6: xchg   %ax,%ax
0x000000010eb879b8: inc    %esi ; OopMap{off=26}
0x000000010eb879ba: test   %eax,-0x9a08c0(%rip)        # 0x000000010e1e7100
0x000000010eb879c0: cmp    $0x186a0,%esi
0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
0x000000010eb879c8: add    $0x40,%rsp
0x000000010eb879cc: pop    %rbp
0x000000010eb879cd: test   %eax,-0x9a08d3(%rip)        # 0x000000010e1e7100
0x000000010eb879d3: retq       ;*return
```

```
  0x000000010eb879a0: mov    %eax,-0x14000(%rsp)
  0x000000010eb879a7: push   %rbp
  0x000000010eb879a8: sub    $0x40,%rsp          ;*iconst_0
  0x000000010eb879ac: mov    $0x0,%esi
  0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
  0x000000010eb879b6: xchg   %ax,%ax
  0x000000010eb879b8: inc    %esi ; OopMap{off=26}
  0x000000010eb879ba: test   %eax,-0x9a08c0(%rip)        # 0x000000010e1e7100
  0x000000010eb879c0: cmp    $0x186a0,%esi
  0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
  0x000000010eb879c8: add    $0x40,%rsp
  0x000000010eb879cc: pop    %rbp
  0x000000010eb879cd: test   %eax,-0x9a08d3(%rip)        # 0x000000010e1e7100
  0x000000010eb879d3: retq                       ;*return
```

```
0x000000010eb879ac: mov    $0x0,%esi
0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
0x000000010eb879b6: xchg   %ax,%ax
0x000000010eb879b8: inc    %esi ; OopMap{off=26}
0x000000010eb879ba: test   %eax,-0x9a08c0(%rip)        # 0x000000010e1e7100
0x000000010eb879c0: cmp    $0x186a0,%esi
0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
0x000000010eb879cd: test   %eax,-0x9a08d3(%rip)        # 0x000000010e1e7100
0x000000010eb879d3: retq       ;*return
```

```
0x000000010eb879ac: mov    $0x0,%esi
0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
0x000000010eb879b6: xchg   %ax,%ax
0x000000010eb879b8: inc    %esi ; OopMap{off=26}
0x000000010eb879ba: test   %eax,-0x9a08c0(%rip)        # 0x000000010e1e7100
0x000000010eb879c0: cmp    $0x186a0,%esi
0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
0x000000010eb879cd: test   %eax,-0x9a08d3(%rip)        # 0x000000010e1e7100
0x000000010eb879d3: retq       ;*return
```

```
0x000000010eb879ac: mov    $0x0,%esi
0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
0x000000010eb879b6: xchg   %ax,%ax
0x000000010eb879b8: inc    %esi ; OopMap{off=26}
0x000000010eb879c0: cmp    $0x186a0,%esi
0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
0x000000010eb879d3: retq       ;*return
```

```
  0x000000010eb879ac: mov    $0x0,%esi
  0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
  0x000000010eb879b6: xchg   %ax,%ax
  0x000000010eb879b8: inc    %esi ; OopMap{off=26}
  0x000000010eb879c0: cmp    $0x186a0,%esi
  0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
  0x000000010eb879d3: retq                       ;*return
```

```
0x000000010eb879ac: mov    $0x0,%esi
0x000000010eb879b1: jmpq   0x000000010eb879c0  ;*iload_1
0x000000010eb879b8: inc    %esi ; OopMap{off=26}
0x000000010eb879c0: cmp    $0x186a0,%esi
0x000000010eb879c6: jl     0x000000010eb879b8  ;*if_icmpge
0x000000010eb879d3: retq       ;*return
```

```
1: mov     $0,%esi
2: jmpq    4:
3: inc     %esi
4: cmp     $1000000,%esi
5: jl      3:
6: retq
```

```java
public class Tiny1 {
    public static void main(String[] args) {
        for (int i = 0; i < 100000; i++) {
            tiny();
        }
    }

    public static int tiny() {
        return 1 + 1;
    }
}
```

```
1: mov    $0,%esi
2: jmpq   4:
3: inc    %esi
4: cmp    $1000000,%esi
5: jl     3:
6: retq
```

1: retq

It’s not that hard
once you know what to look at.

Part 3: Wonderland

Java Features

- final fields
- synchronization
- string switch
- lambda

#1: Final Fields

- Final fields can’t be modified
- The compiler pipeline can take advantage
  - ...but it doesn’t always

```
public class Fields {
    private static final String MY_STRING =
            "This is a static string";
    private static final String MY_PROPERTY =
            System.getProperty("java.home");

    public static void main(String[] args) {
        System.out.println(MY_STRING);
        System.out.println(MY_PROPERTY);
    }
}
```

```java
private static final String MY_STRING =
                    "This is a static string";
private static final String MY_PROPERTY =
                     System.getProperty("java.home");
```

```
public static void main(java.lang.String[]);
  Code:
     0: getstatic     #7    // Field java/lang/System.out:Ljava/io/PrintStream;
     3: ldc           #9    // String This is a static string
     5: invokevirtual #10   // Method java/io/PrintStream.println:(Ljava/lang/String;)V
     8: getstatic     #7    // Field java/lang/System.out:Ljava/io/PrintStream;
    11: getstatic     #11   // Field MY_PROPERTY:Ljava/lang/String;
    14: invokevirtual #10   // Method java/io/PrintStream.println:(Ljava/lang/String;)V
```

```
private static int addHashes() {
    return MY_STRING.hashCode() + MY_PROPERTY.hashCode();
}
```

```
movabs $0x7aab6c4f8,%r10  ;   {oop("This is a static string")}
mov    %eax,0x10(%r10)    ;*iload_1
           ; - String::hashCode@53 (line 1467)
           ; - Fields::addHashes@2 (line 36)

movabs $0x7aaa97a98,%rcx
           ;   {oop(".../jdk1.8.0.jdk/Contents/Home/jre")}
mov    0x10(%rcx),%r10d   ;*getfield hash
           ; - String::hashCode@1 (line 1458)
           ; - Fields::addHashes@8 (line 36)
```

```
private final String myString = "This is an instance string";
private final String myProperty = System.getProperty("java.home");

public int addHashes2() {
    return myString.hashCode() + myProperty.hashCode();
}
```

```
private int addHashes2();
  Code:
     0: ldc           #2   // String This is an instance string
     2: invokevirtual #18  // Method java/lang/String.hashCode:()I
     5: aload_0
     6: getfield      #6   // Field myProperty:Ljava/lang/String;
     9: invokevirtual #18  // Method java/lang/String.hashCode:()I
    12: iadd
    13: ireturn
```

```
movabs $0x7aab6d318,%rcx  ;   {oop("This is an instance string")}
mov    0x10(%rcx),%r10d   ;*getfield hash
          ; - String::hashCode@1 (line 1458)
          ; - Fields::addHashes2@2 (line 40)
```

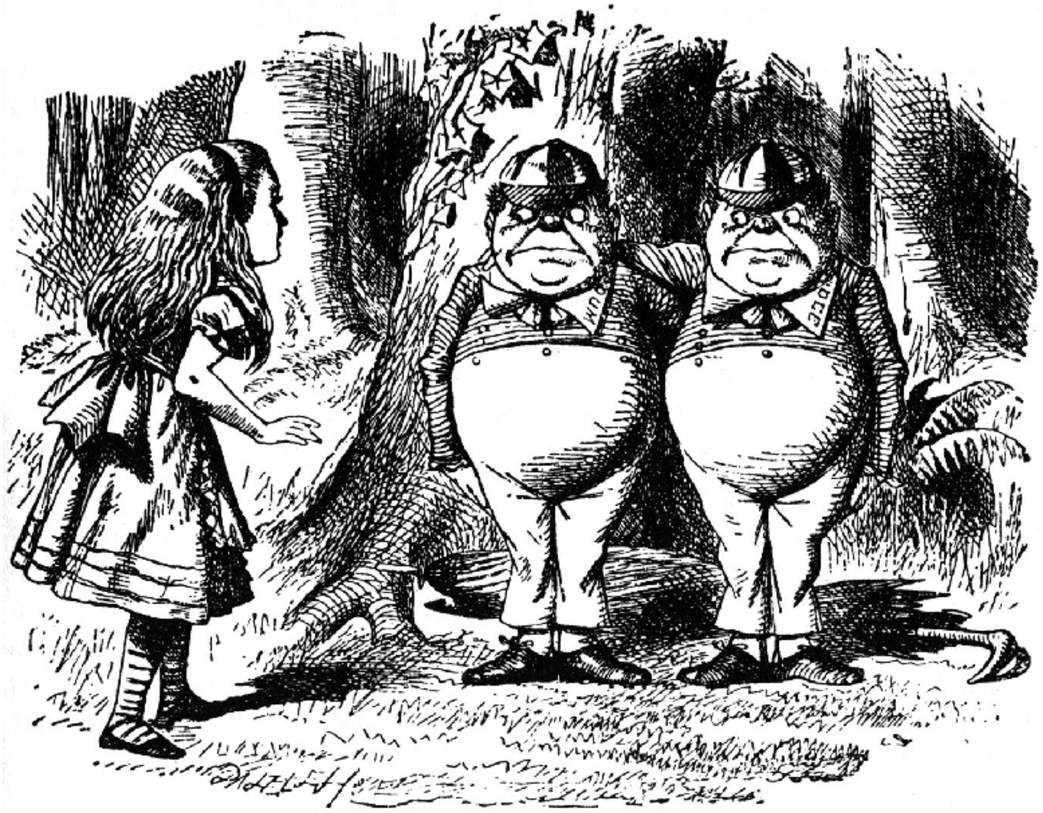
```
mov    0x10(%rsi),%ecx    ;*getfield myProperty
                          ; - Fields::addHashes2@6 (line 40)

mov    0x10(%r12,%rcx,8),%eax  ;*getfield hash
                               ; - String::hashCode@1 (line 1458)
                               ; - Fields::addHashes2@9 (line 40)
```

ACHIEVEMENT UNLOCKED:

Find something Hotspot could do better

#2: Concurrency Stuff

- What does “synchronized” do?
- What does “volatile” do?



```
public class Concurrency {
    public static void main(String[] args) {
        System.out.println(getTime());
        System.out.println(getTimeSynchronized());
    }

    public static long getTime() {
        return System.currentTimeMillis();
    }

    public static synchronized long getTimeSynchronized() {
        return System.currentTimeMillis();
    }
}
```

```
public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: invokestatic  #3  // Method getTime:()J
       6: invokevirtual #4  // Method java/io/PrintStream.println:(J)V
       9: getstatic     #2  // Field java/lang/System.out:Ljava/io/PrintStream;
      12: invokestatic  #5  // Method getTimeSynchronized:()J
      15: invokevirtual #4  // Method java/io/PrintStream.println:(J)V
```

```
public static long getTime();
  Code:
     0: invokestatic  #7  // Method java/lang/System.currentTimeMillis:()J
     3: lreturn

public static synchronized long getTimeSynchronized();
  Code:
     0: invokestatic  #7  // Method java/lang/System.currentTimeMillis:()J
     3: lreturn
```

```
'getTime' '()J' in 'com/headius/talks/rabbithole/Concurrency'

  movabs $0x1015dbd3e,%r10
  callq  *%r10                ;*invokestatic currentTimeMillis
                              ; - Concurrency::getTime@0 (line 22)
  retq
```

????

```
movabs $0x7aab6bee8,%r10  ;   {oop(a 'java/lang/Class' = '.../Concurrency')}
mov    (%r10),%rax
mov    %rax,%r10
and    $0x7,%r10
cmp    $0x5,%r10
jne    0x000000010ef0665f
mov    $0xdf3803fe,%r11d  ;   {metadata('java/lang/Class')}
mov    0xa8(%r12,%r11,8),%r10
mov    %r10,%r11
or     %r15,%r11
mov    %r11,%r8
xor    %rax,%r8
$0xffffffffffffff87,%r8
jne    0x000000010ef068e4
mov    %r14d,(%rsp)       ;*synchronization entry
                          ; - Concurrency::getTimeSynchronized@-1 (line 26)
                          ; - Concurrency::main@58 (line 16)

movabs $0x10de5ad3e,%r10
callq  *%r10              ;*invokestatic currentTimeMillis
                          ; - Concurrency::getTimeSynchronized@0 (line 26)
                          ; - Concurrency::main@58 (line 16)
```

```
movabs $0x7aab6bee8,%r10  ;   {oop(a 'java/lang/Class' = '.../Concurrency')}
mov    (%r10),%rax
mov    %rax,%r10
and    $0x7,%r10
cmp    $0x5,%r10
jne    0x000000010ef0665f
mov    $0xdf3803fe,%r11d  ;   {metadata('java/lang/Class')}
mov    0xa8(%r12,%r11,8),%r10
mov    %r10,%r11
or     %r15,%r11
mov    %r11,%r8
xor    %rax,%r8
$0xffffffffffffff87,%r8
jne    0x000000010ef068e4
mov    %r14d,(%rsp)       ;*synchronization entry
                          ; - Concurrency::getTimeSynchronized@-1 (line 26)
                          ; - Concurrency::main@58 (line 16)
```

```
movabs $0x10de5ad3e,%r10
callq  *%r10              ;*invokestatic currentTimeMillis
                          ; - Concurrency::getTimeSynchronized@0 (line 26)
                          ; - Concurrency::main@58 (line 16)
```

```
movabs $0x7aab6bee8,%r10  ;   {oop(a 'java/lang/Class' = '.../Concurrency')}
mov    (%r10),%rax
mov    %rax,%r10
and    $0x7,%r10
cmp    $0x5,%r10
jne    0x000000010ef0665f
mov    $0xdf3803fe,%r11d  ;   {metadata('java/lang/Class')}
mov    0xa8(%r12,%r11,8),%r10
mov    %r10,%r11
or     %r15,%r11
mov    %r11,%r8
xor    %rax,%r8
$0xffffffffffffff87,%r8
jne    0x000000010ef068e4
mov    %r14d,(%rsp)       ;*synchronization entry
                          ; - Concurrency::getTimeSynchronized@-1 (line 26)
                          ; - Concurrency::main@58 (line 16)
```

```
movabs $0x10de5ad3e,%r10
callq  *%r10              ;*invokestatic currentTimeMillis
                          ; - Concurrency::getTimeSynchronized@0 (line 26)
                          ; - Concurrency::main@58 (line 16)
```

Vocabulary

- Lock coarsening
  - Expanding the use of multiple fine-grained locks into a single coarse-grained lock
- Lock eliding
  - Eliminating locking when it will not affect the behavior of the program

```
movabs $0x7aab6bee8,%r10  ;   {oop(a 'java/lang/Class' = '.../Concurrency')}
mov    (%r10),%rax
mov    %rax,%r10
and    $0x7,%r10
cmp    $0x5,%r10
jne    0x000000010ef0665f
mov    $0xdf3803fe,%r11d  ;   {metadata('java/lang/Class')}
mov    0xa8(%r12,%r11,8),%r10
mov    %r10,%r11
or     %r15,%r11
mov    %r11,%r8
xor    %rax,%r8
$0xffffffffffffff87,%r8
jne    0x000000010ef068e4
mov    %r14d,(%rsp)       ;*synchronization entry
                          ; - Concurrency::getTimeSynchronized@-1 (line 26)
                          ; - Concurrency::main@58 (line 16)

movabs $0x10de5ad3e,%r10
callq  *%r10              ;*invokestatic currentTimeMillis
                          ; - Concurrency::getTimeSynchronized@0 (line 26)
                          ; - Concurrency::main@58 (line 16)
```

```
0x000000010ef0665f: movabs $0x7aab6bee8,%r11
        ;   {oop(a 'java/lang/Class' = '../Concurrency')}
0x000000010ef06669: lea    0x10(%rsp),%rbx
0x000000010ef0666e: mov    (%r11),%rax
0x000000010ef06671: test   $0x2,%eax
0x000000010ef06676: jne    0x000000010ef0669f
0x000000010ef0667c: or     $0x1,%eax
0x000000010ef0667f: mov    %rax,(%rbx)
0x000000010ef06682: lock cmpxchg %rbx,(%r11)
0x000000010ef06687: je     0x000000010ef066bc
```

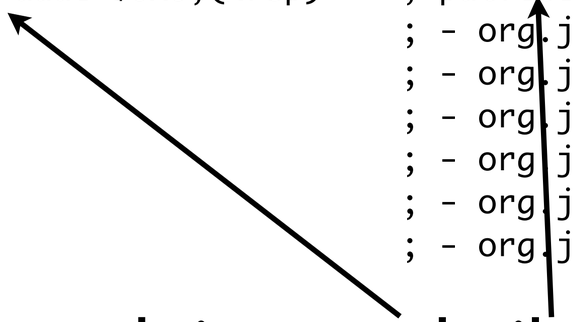
Volatile

- Forces commit of memory
- Forces code ordering
- Prevents some optimizations
- Similar impact to locking
  - ...but it can’t ever be removed

LOCK

Code from a RubyBasicObject's default constructor.

```
11345d823: mov    0x70(%r8),%r9d    ;*getstatic NULL_OBJECT_ARRAY
                                    ; - org.jruby.RubyBasicObject::<init>@5 (line 76)
                                    ; - org.jruby.RubyObject::<init>@2 (line 118)
                                    ; - org.jruby.RubyNumeric::<init>@2 (line 111)
                                    ; - org.jruby.RubyInteger::<init>@2 (line 95)
                                    ; - org.jruby.RubyFixnum::<init>@5 (line 112)
                                    ; - org.jruby.RubyFixnum::newFixnum@25 (line 173)
11345d827: mov    %r9d,0x14(%rax)
11345d82b: lock addl $0x0,(%rsp)    ;*putfield varTable
                                    ; - org.jruby.RubyBasicObject::<init>@8 (line 76)
                                    ; - org.jruby.RubyObject::<init>@2 (line 118)
                                    ; - org.jruby.RubyNumeric::<init>@2 (line 111)
                                    ; - org.jruby.RubyInteger::<init>@2 (line 95)
                                    ; - org.jruby.RubyFixnum::<init>@5 (line 112)
                                    ; - org.jruby.RubyFixnum::newFixnum@25 (line 173)
```



Why are we doing a volatile write in the constructor?

LOCK

```
public class RubyBasicObject ... {
    private static final boolean DEBUG = false;
    private static final Object[] NULL_OBJECT_ARRAY = new Object[0];

    // The class of this object
    protected transient RubyClass metaClass;

    // zeroed by jvm
    protected int flags;

    // variable table, lazily allocated as needed (if needed)
    private volatile Object[] varTable = NULL_OBJECT_ARRAY;
```

Maybe it's not such a good idea to pre-init a volatile?

LOCK

```
~/projects/jruby ➔ git log 2f935de1e40bfd8b29b3a74eaed699e519571046 -1 | cat
commit 2f935de1e40bfd8b29b3a74eaed699e519571046
Author: Charles Oliver Nutter <headius@headius.com>
Date:   Tue Jun 14 02:59:41 2011 -0500

    Do not eagerly initialize volatile varTable field in RubyBasicObject; speeds
object creation significantly.
```

ACHIEVEMENT UN"LOCK"ED:

Fix a Java performance bug by reading assembly code

#3: String Switch

- Use literal strings as "case" targets
- Added in Java 7
  - ...and there was much rejoicing
- But how does it really work?

A Normal Switch

- Variable switch parameter
- Constant case values
- Branch based on a table (fast) for narrow range of cases
- Branch based on a lookup (less fast) for broad range of cases

```
public class StringSwitch {
    public static void main(String[] args) {
        String count = "unknown";
        switch (args.length) {
            case 0: count = "zero"; break;
            case 1: count = "one"; break;
            case 2: count = "two"; break;
        }
...
```

```
public static void main(java.lang.String[]);
    Code:
       0: ldc           #2  // String unknown
       2: astore_1
       3: aload_0
       4: arraylength
       5: tableswitch   { // 0 to 2
                     0: 32
                     1: 38
                     2: 44
                default: 47
          }
      32: ldc           #3  // String zero
      34: astore_1
      35: goto          47
      38: ldc           #4  // String one
      40: astore_1
      41: goto          47
      44: ldc           #5  // String two
      46: astore_1
```

```
switch (args.length) {
     case 2000000: count = "two million"; break;
     case 1000000: count = "one million"; break;
     case 3000000: count = "three million"; break;
}
```

```
49: lookupswitch  { // 3
              1000000: 90
              2000000: 84
              3000000: 96
            default: 99
        }
```

Comparison

- tableswitch is O(1)
  - Indexed lookup of target
- lookupswitch is O(log n)
  - Binary search for target

Get to the point already!

The Point

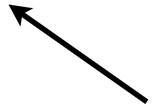
- What kind of switch do we use for String?
  - Table doesn’t work for hashcodes
  - Lookup might collide
- Answer: both, plus .equals()

```
static String chooseGreeting(String language) {
    switch (language) {
        case "Java": return "I love to hate you!";
        case "Scala": return "I love you, I think!";
        case "Clojure": return "(love I you)";
        case "Groovy": return "I love ?: you";
        case "Ruby": return "I.love? you # => true";
        default: return "Who are you?";
    }
}
```

```
static java.lang.String chooseGreeting(java.lang.String);
   Code:
      0: aload_0
      1: astore_1
      2: iconst_m1
      3: istore_2
      4: aload_1
      5: invokevirtual #16 // Method java/lang/String.hashCode:()I
      8: lookupswitch  { // 5
          -1764029756: 88
              2301506: 60
              2558458: 116
             79698214: 74
           2141368366: 102
              default: 127
         }
```

Hidden int variable...

```
74: aload_1
75: ldc           #14 // String Scala
77: invokevirtual #17 // Method String.equals:(Ljava/lang/Object;)Z
80: ifeq          127
83: iconst_1
84: istore_2
```



Same hidden int variable now = 1

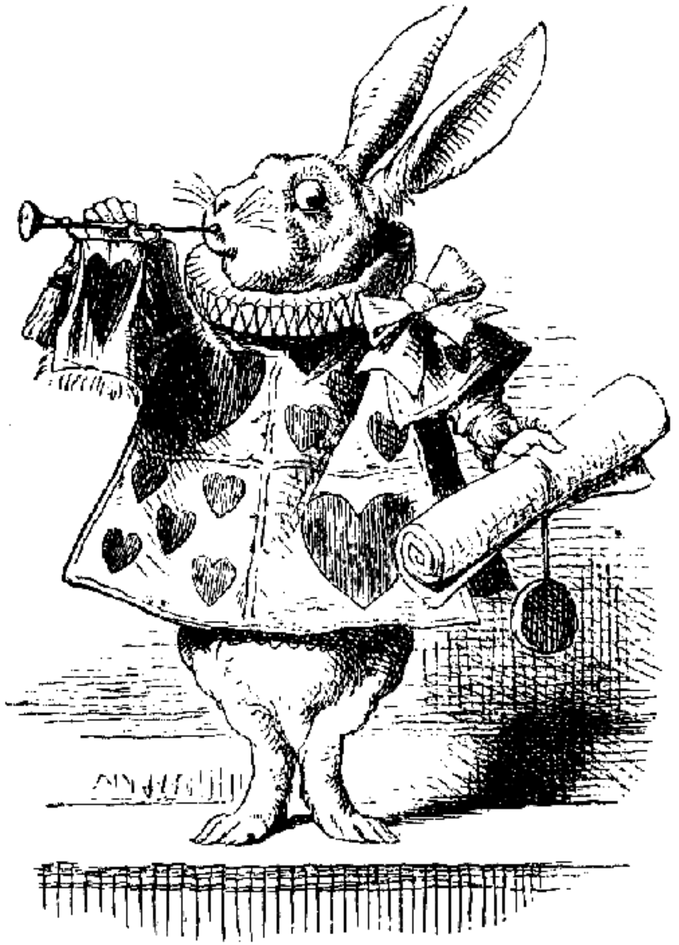
```
127: iload_2
128: tableswitch   { // 0 to 4
                0: 164
                1: 167
                2: 170
                3: 173
                4: 176
          default: 179
     }
164: ldc           #20  // String I love to hate you!
166: areturn
167: ldc           #21  // String I love you, I think!
169: areturn
170: ldc           #22  // String (love I you)
172: areturn
173: ldc           #23  // String I love ?: you
175: areturn
176: ldc           #24  // String I.love? you # => true
178: areturn
179: ldc           #25  // String Who are you?
181: areturn
```

```
static String chooseGreeting2(String language) {
    int hash = language.hashCode();
    int target = -1;
    switch (hash) {
        case 2301506: if (language.equals("Java")) target = 0; break;
        case 79698214: if (language.equals("Scala"))target = 1; break;
        case -1764029756: if (language.equals("Clojure"))target = 2; break;
        case 2141368366: if (language.equals("Groovy"))target = 3; break;
        case 2558458: if (language.equals("Ruby"))target = 3; break;
    }
    switch (target) {
        case 0: return "I love to hate you!";
        case 1: return "I love you, I think!";
        case 2: return "(love I you)";
        case 3: return "I love ?: you";
        case 4: return "I.love? you # => true";
        default: return "Who are you?";
    }
}
```

It’s just a hash table!

#4: Lambda Expressions

- New for Java 8
  - ...and there was much rejoicing
- Key goals
  - Lighter-weight than inner classes
  - No class-per-lambda
  - Optimizable by JVM



```
public class LambdaStuff {
    public static void main(String[] args) {
        List<String> list = Arrays.asList(
                "Clojure",
                "Java",
                "Ruby",
                "Groovy",
                "Scala"
        );

        for (int i = 0; i < 100000; i++) {
            doSort(list);
            getRest(list);
            getAllCaps(list);
            getInitials(list);
            getInitialsManually(list);
        }
```

```
public static void doSort(List<String> input) {
    Collections.sort(input,
                     (a,b)->Integer.compare(a.length(), b.length()));
}
```

```
public static void doSort(java.util.List<java.lang.String>);
  Code:
     0: aload_0
     1: invokedynamic #36,  0
// InvokeDynamic #4:compare:()Ljava/util/Comparator;
     6: invokestatic  #37
// Method java/util/Collections.sort ...
     9: return
```

```
public static void doSort(java.util.List<java.lang.String>);
  Code:
     0: aload_0
     1: invokedynamic #36,  0
// InvokeDynamic #4:compare:()Ljava/util/Comparator;
     6: invokestatic  #37
// Method java/util/Collections.sort ...
     9: return
```

Invokedynamic is used to create the initial lambda object and then cache it forever.

Compare to anonymous inner classes, where an instance is created every time.

```
$ javap -cp dist/RabbitHole.jar \
        -verbose \
        -c \
        com.headius.talks.rabbithole.LambdaStuff
```

```
  BootstrapMethods:
...
    4: #142 invokestatic java/lang/invoke/LambdaMetafactory.metafactory...
                                  ...bunch of types here
      Method arguments:
        #167 (Ljava/lang/Object;Ljava/lang/Object;)I
        #168 invokestatic LambdaStuff.lambda$2:(Ljava/lang/String;Ljava/lang/String;)I
        #169 (Ljava/lang/String;Ljava/lang/String;)I
```

LambdaMetaFactory generates an implementation of our interface (Comparator) using Method Handles (from JSR292)

```
  BootstrapMethods:
...
    4: #142 invokestatic java/lang/invoke/LambdaMetafactory.metafactory...
                                    ...bunch of types here
      Method arguments:
        #167 (Ljava/lang/Object;Ljava/lang/Object;)I
        #168 invokestatic LambdaStuff.lambda$2:(Ljava/lang/String;Ljava/lang/String;)I
        #169 (Ljava/lang/String;Ljava/lang/String;)I
```

LambdaMetaFactory generates an implementation of our interface (Comparator) using Method Handles (from JSR292)

```
private static int lambda$2(java.lang.String, java.lang.String);
  Code:
     0: aload_0
     1: invokevirtual #53  // Method java/lang/String.length:()I
     4: aload_1
     5: invokevirtual #53  // Method java/lang/String.length:()I
     8: invokestatic  #54  // Method java/lang/Integer.compare:(II)I
    11: ireturn
```

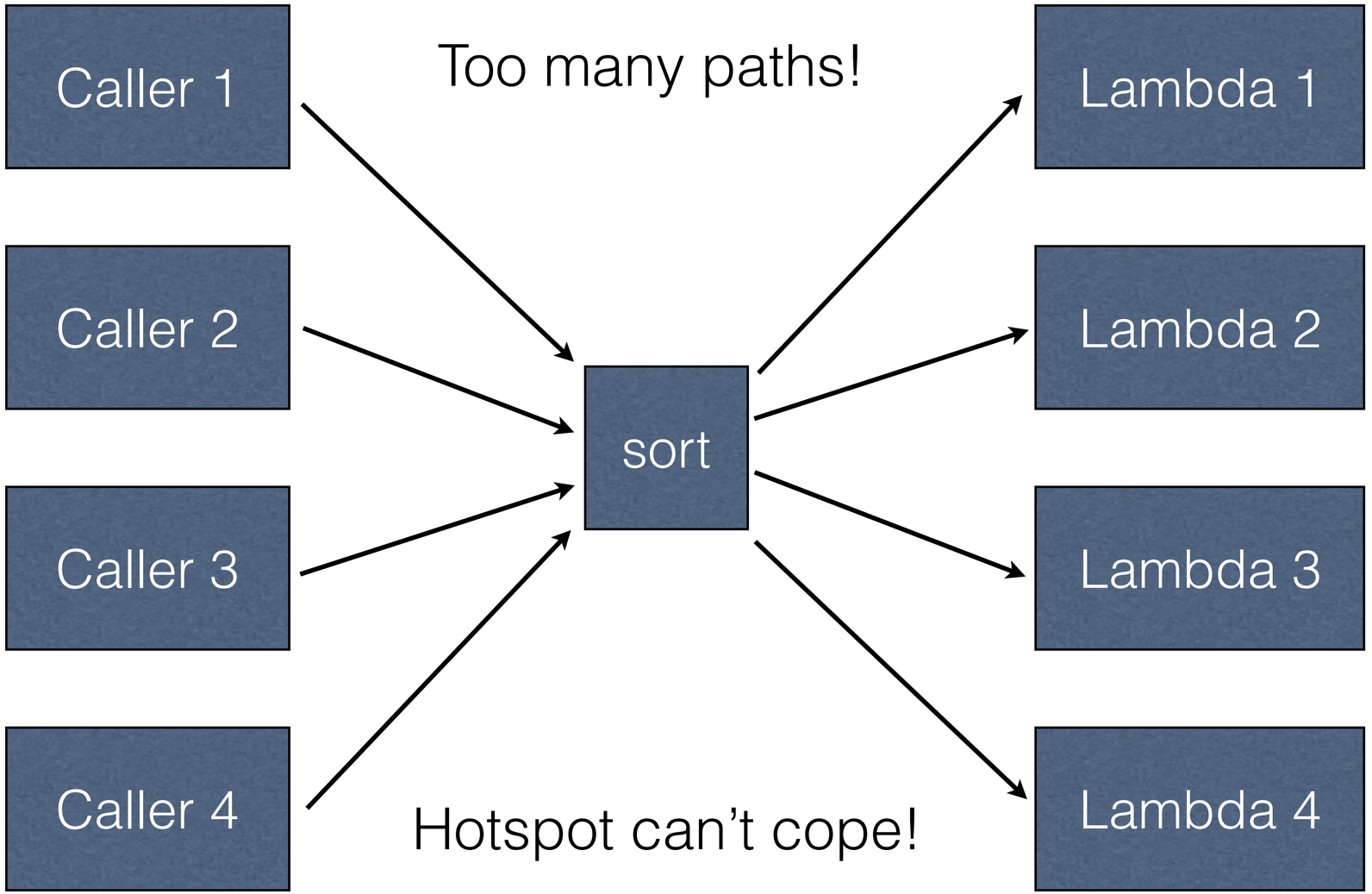
Lambda body is just a static method;
all state is passed to it.

Because the wrapper is generated and the body is just a static method, we have no extra classes and potentially **no** allocation.

Will It Blend?

The Problem

- In order to inline code, we need:
  - A consistent target method
  - A distinct path through the code
- Collections.sort's lambda callback
  - Will see many different methods
  - Will be called via many different paths


Caller 1
Caller 2
Caller 3
Caller 4
Too many paths!
sort
Hotspot can’t cope!
Lambda 1
Lambda 2
Lambda 3
Lambda 4

```
public static String getInitials(List<String> input) {
    return input.stream()
            .map(x->x.substring(0,1))
            .collect(Collectors.joining());
}

public static String getInitialsManually(List<String> input) {
    StringBuilder builder = new StringBuilder();
    UnaryOperator<String> initial = (String x)->x.substring(0,1);
    for (String s : input) {
        builder.append(initial.apply(s));
    }
    return builder.toString();
}
```

```
public static void time(Object name, int iterations, Runnable body) {
    long start = System.currentTimeMillis();
    for (int i = 0; i < iterations; i++) {
        body.run();
    }
    System.out.println(name.toString()
            + ": "
            + (System.currentTimeMillis() - start));
}
```

```
Function<List<String>, String> getInitials =
                                LambdaStuff::getInitials;
Function<List<String>, String> getInitialsManually =
                                LambdaStuff::getInitialsManually;

for (int i = 0; i < 10; i++) {
    time("getInitials", 1000000,
         ()->getInitials.apply(list));
    time("getInitialsManually", 1000000,
         ()->getInitialsManually.apply(list));
}
```

Drum roll, please...

```
public static String getInitials(List<String> input) {
    return input.stream()
                .map(x->x.substring(0,1))
                .collect(Collectors.joining());
}
```

```
mov     %r10d,0x24(%r9)     ;*putfield nextStage
                            ; - java.util.stream.AbstractPipeline::<init>@28 (line 200)
                            ; - java.util.stream.ReferencePipeline::<init>@3 (line 94)
                            ; - java.util.stream.ReferencePipeline$StatelessOp::<init>@3 (line 627)
                            ; - java.util.stream.ReferencePipeline$3::<init>@16 (line 188)
                            ; - java.util.stream.ReferencePipeline::map@22 (line 187)
                            ; - com.headius.talks.rabbithole.LambdaStuff::getInitials@11 (line 57)
```

Methods like map() and collect() inline...

```
public static String getInitials(List<String> input) {
    return input.stream()
                .map(x->x.substring(0,1))
                .collect(Collectors.joining());
}
```

```
callq  0x0000000105973f20  ; OopMap{rbp=Oop [0]=NarrowOop off=2776}
                           ;*invokeinterface apply
                           ; - java.util.stream.ReferencePipeline::collect@118 (line 512)
                           ;   {runtime_call}
```

But they can’t inline all those lambdas.

```
public static String getInitialsManually(List<String> input) {
    StringBuilder builder = new StringBuilder();
    UnaryOperator<String> initial = (String x)->x.substring(0,1);
    for (String s : input) {
        builder.append(initial.apply(s));
    }
    return builder.toString();
}
```

```
mov    0x60(%r15),%rcx
mov    %rcx,%r10
add    $0x18,%r10
cmp    0x70(%r15),%r10
jae    0x0000000104548d78
mov    %r10,0x60(%r15)
prefetchnta 0xc0(%r10)
mov    $0xdf3802e6,%r10d  ;   {metadata('java/lang/String')}
mov    0xa8(%r12,%r10,8),%r10
mov    %r10,(%rcx)
movl   $0xdf3802e6,0x8(%rcx)  ;   {metadata('java/lang/String')}
mov    %r12d,0xc(%rcx)
mov    %r12,0x10(%rcx)    ;*new  ; - String::substring@65 (line 1961)
                                 ; - LambdaStuff::lambda$6@3 (line 75)
                                 ; - LambdaStuff$$Lambda$9::apply@4
                                 ; - LambdaStuff::getInitialsManually@45 (line 77)
```

```java
public static String getInitialsManually(List<String> input) {
    StringBuilder builder = new StringBuilder();
    UnaryOperator<String> initial = (String x)->x.substring(0,1);
    for (String s : input) {
        builder.append(initial.apply(s));
    }
    return builder.toString();
}
```

```
mov    0x60(%r15),%rcx
mov    %rcx,%r10
add    $0x18,%r10
cmp    0x70(%r15),%r10
jae    0x0000000104548d78
mov    %r10,0x60(%r15)
prefetchnta 0xc0(%r10)
mov    $0xdf3802e6,%r10d  ;   {metadata('java/lang/String')}
mov    0xa8(%r12,%r10,8),%r10
mov    %r10,(%rcx)
movl   $0xdf3802e6,0x8(%rcx)  ;   {metadata('java/lang/String')}
mov    %r12d,0xc(%rcx)
mov    %r12,0x10(%rcx)    ;*new  ; - String::substring@65 (line 1961)
                                 ; - LambdaStuff::lambda$6@3 (line 75)
                                 ; - LambdaStuff$$Lambda$9::apply@4
                                 ; - LambdaStuff::getInitialsManually@45 (line 77)
```

What Have We Learned?

The JVM is an amazing platform

But it is not perfect.

Every feature has a cost.

You’ll be a better developer if you remember that.

And if, like Alice...

…you aren’t afraid to go down the rabbit hole.

Thank You!

- Charles Oliver Nutter
- @headius
- headius@headius.com
- http://blog.headius.com