

# JVM Mechanics

When Does the JVM JIT & Deoptimize?



<https://github.com/dougqh/jvm-mechanics>

Douglas Q. Hawkins  
VM Engineer



# About Azul

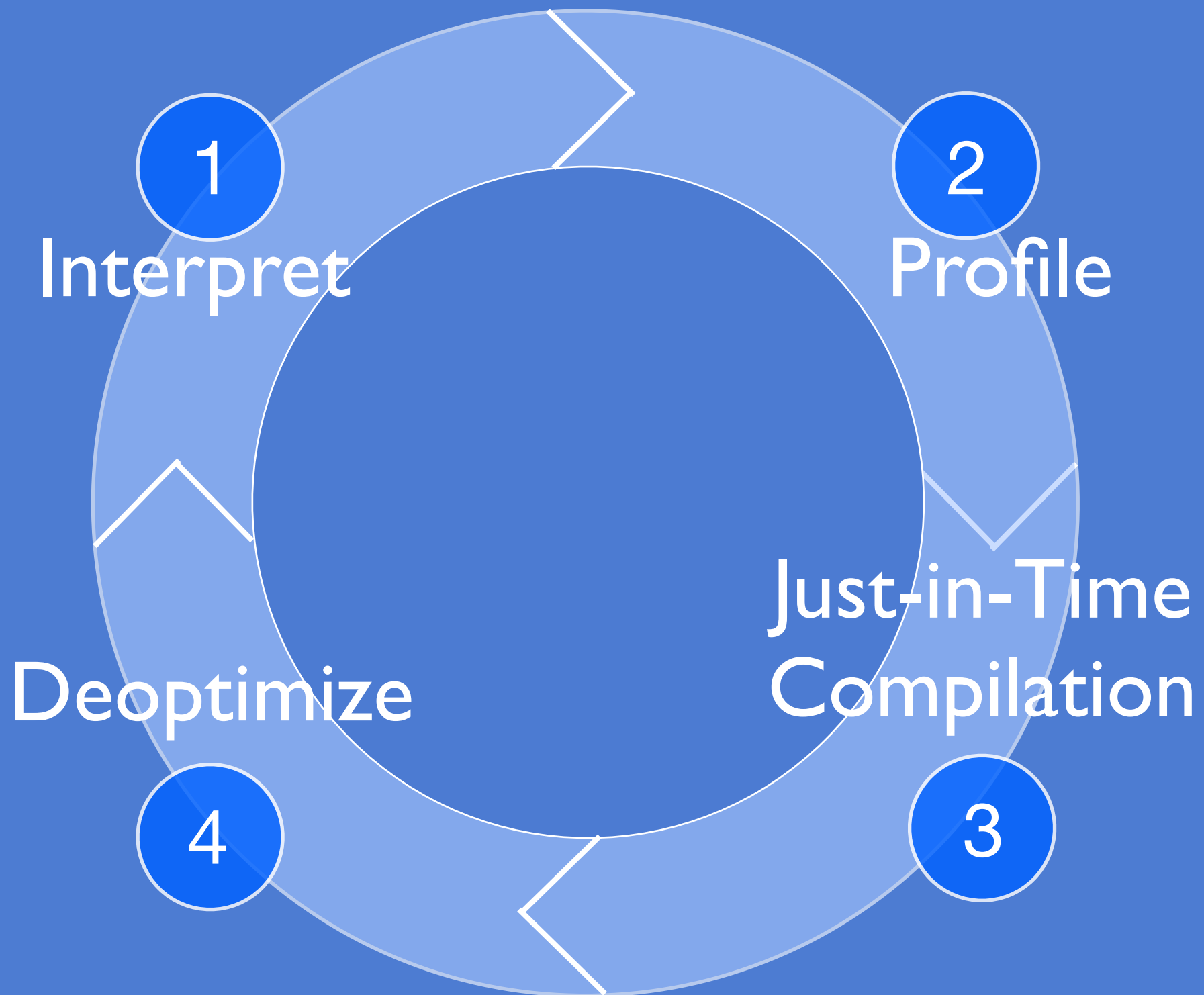
**Zulu**<sup>®</sup> Multi-Platform OpenJDK

Cloud Support including Docker and Azure  
Embedded Support

**Zing**<sup>®</sup> Highly Scalable VM

Continuously Concurrent Compacting Collector  
ReadyNow! for Low Latency Applications

# HotSpot Lifecycle





Why?

# A Simple Program

```
public class SimpleProgram {
    static final int CHUNK_SIZE = 1_000;

    public static void main(String[] args) {
        for ( int i = 0; i < 250; ++i ) {
            long startTime = System.nanoTime();

            for ( int j = 0; j < CHUNK_SIZE; ++j ) {
                new Object();
            }

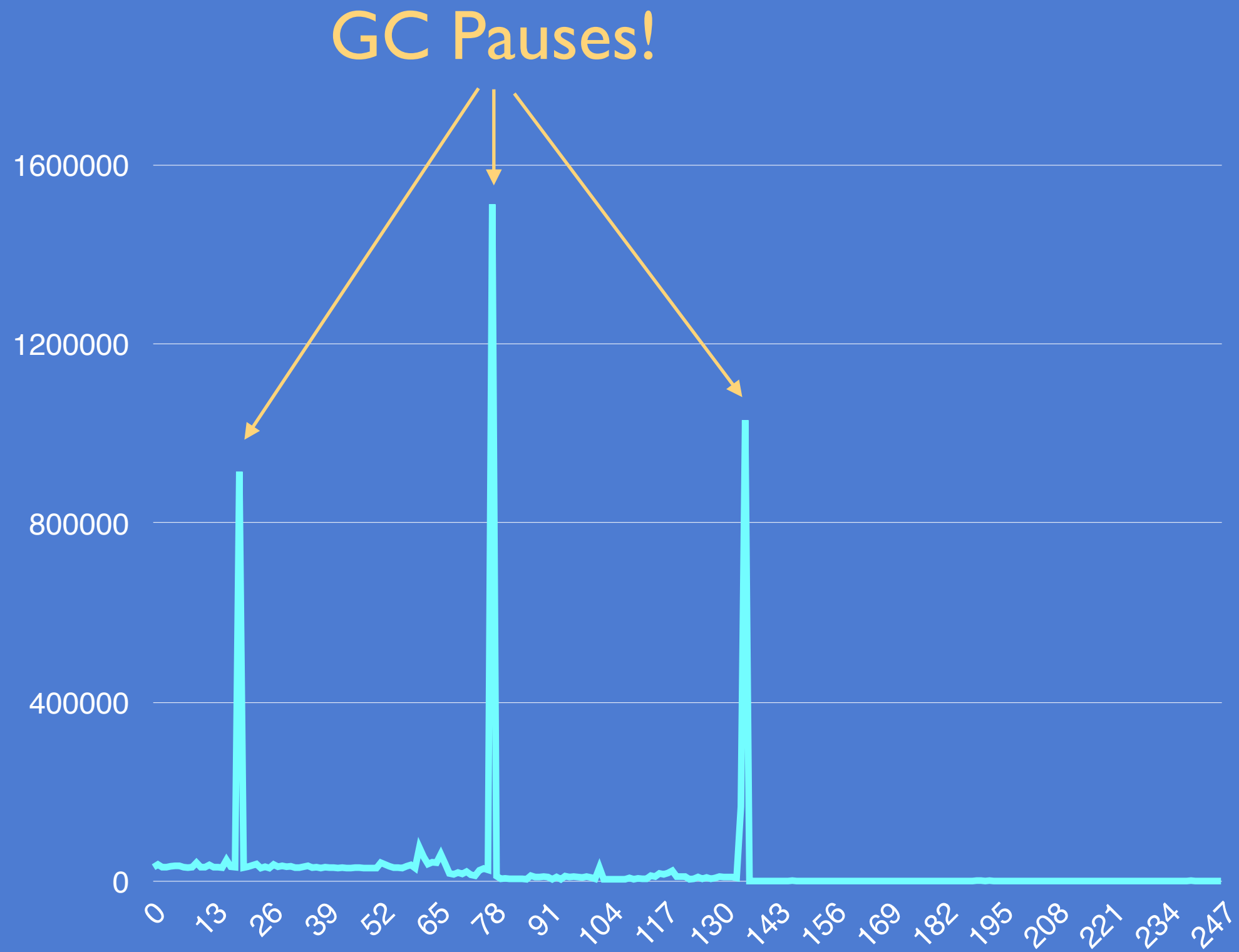
            long endTime = System.nanoTime();
            System.out.printf("%d\t%d%n", i, endTime - startTime);
        }
    }
}
```

Code Reference

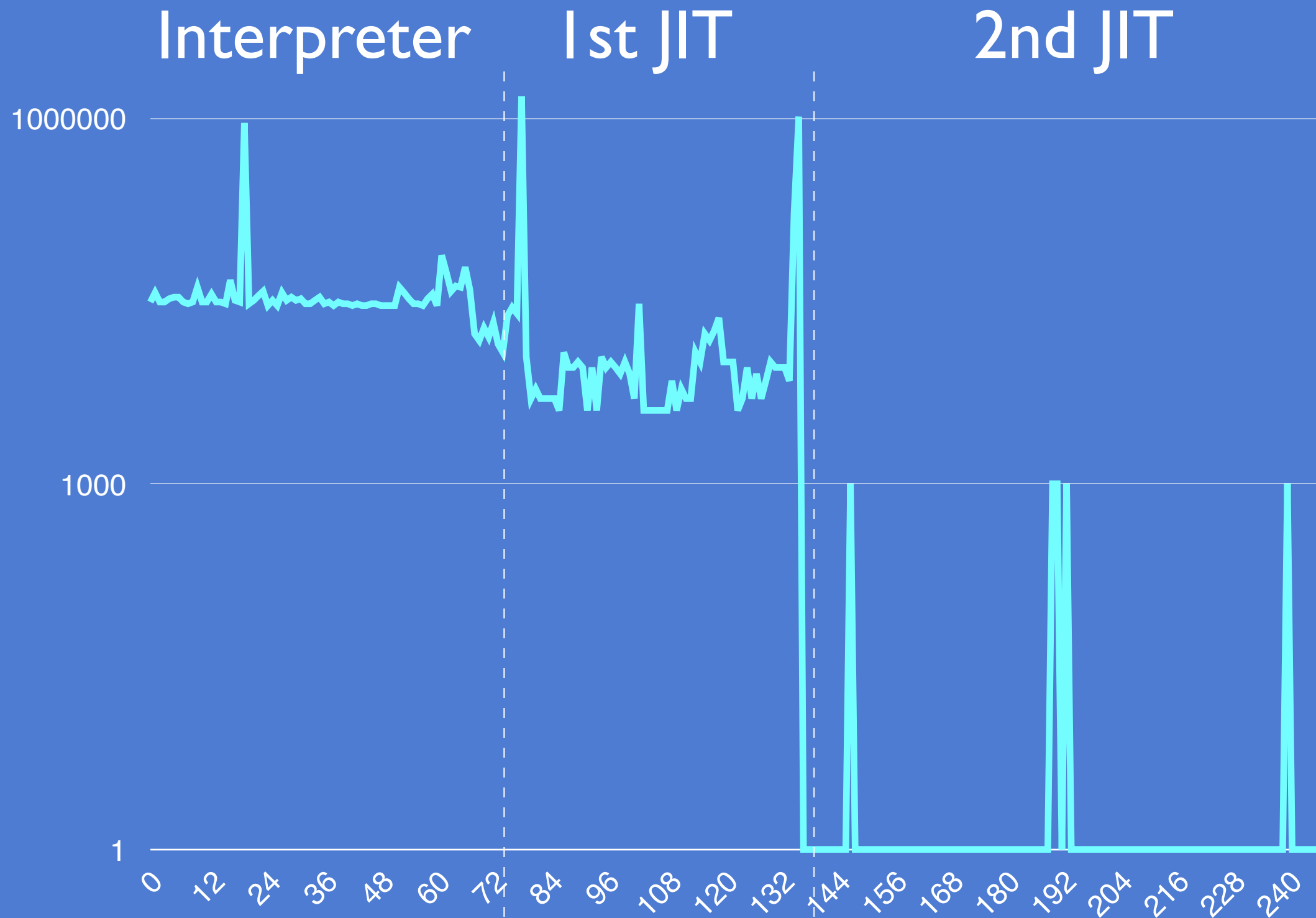


example01a.SimpleProgram

# Simple Program Performance



# Log Scale



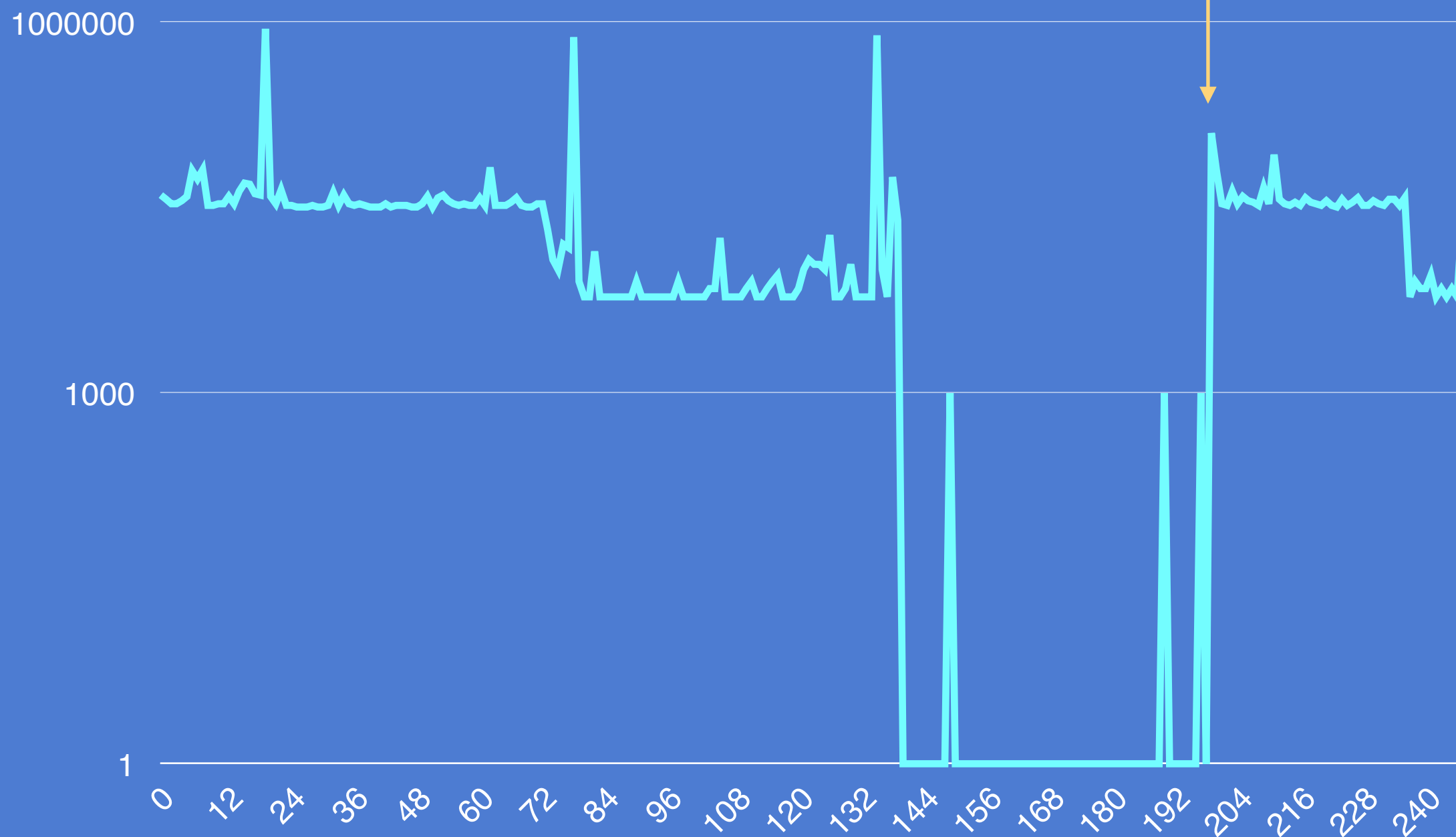
# Not So Simple Program

```
public class NotSoSimpleProgram {  
    static final int CHUNK_SIZE = 1_000;  
  
    public static void main(String[] args) {  
        Object trap = null;  
  
        for ( int i = 0; i < 250; ++i ) {  
            long startTime = System.nanoTime();  
  
            for ( int j = 0; j < CHUNK_SIZE; ++j ) {  
                new Object();  
  
                if ( trap != null ) {  
                    System.out.println("trap!");  
                    trap = null;  
                }  
            }  
  
            if ( i == 200 ) trap = new Object();  
  
            long endTime = System.nanoTime();  
            System.out.printf("%d\t%d\n", i, endTime - startTime);  
        }  
    }  
}
```



# Deoptimization

behavioral change,  
deoptimize!



# Interpreter

Slow!  
100000x

Dynamically Generated  
Threaded Interpreter  
Identify “Hot Spots”

# Invocation Counter

```
public class InvocationCounter {  
    public static void main(final String[] args)  
        throws InterruptedException  
    {  
        for ( int i = 0; i < 20_000; ++i ) {  
            hotMethod();  
        }  
  
        System.out.println("Waiting for compiler...");  
        Thread.sleep(5_000);  
    }  
  
    static void hotMethod() {}  
}
```

# Invocation Counter

-XX:+PrintCompilation

299	1 %	java.lang.String::indexOf @ 37 (70 bytes)
332	2	java.lang.String::indexOf (70 bytes)
364	3	example02.InvocationCounter::hotMethod (1 bytes)
365	4 %	example02.InvocationCounter::main @ 5 (33 bytes)

Waiting for compiler...

# Compilation Log

timestamp (since VM start)	compilation ID	method name	method size
5328	50	n java.io.FileOutputStream::writeBytes (native)	
5330	27	! java.nio.CharBuffer::wrap (20 bytes)	
5333	31	s java.io.BufferedOutputStream::flush (12 bytes)	
5477	56	% CompilationExample::main @ 37 (82 bytes)	
		on-stack replacement	
		synchronized method	
		exception handler	
		native	
			loop bytecode index

# synchronized is try/finally

```
synchronized ( foo.getBar() ) {  
    ...  
}
```



```
tmp = foo.getBar();  
monitor_enter(tmp);  
try {  
    ...  
} finally {  
    monitor_exit(tmp);  
}
```

# Duplicate Methods & Overloading

5208	4	<code>java.nio.Buffer::position</code> (5 bytes)
		...
5223	8	<code>java.nio.Buffer::position</code> (43 bytes)

# Invisible Overloads via Bridge Methods

```
public interface Supplier<T> {  
    public abstract T get();  
}
```

```
new Supplier<String>() {  
    public final String get() { return "foo"; }  
};
```

145	4	InvisibleOverload\$1::get (5 bytes)
145	5	InvisibleOverload\$1::get (3 bytes)



# Backedge Counter

```
public class BackedgeCounter {  
    public static void main(final String[] args)  
        throws InterruptedException  
    {  
        for ( int i = 0; i < 20_000; ++i ) {  
            hotMethod();  
        }  
  
        System.out.println("Waiting for compiler...");  
        Thread.sleep(5_000);  
  
        for ( int i = 0; i < 20_000; ++i ) {  
            hotMethod();  
        }  
  
        System.out.println("Waiting for compiler...");  
        Thread.sleep(5_000);  
    }  
  
    static void hotMethod() {}  
}
```

# Backedge Counter

-XX:+PrintCompilation

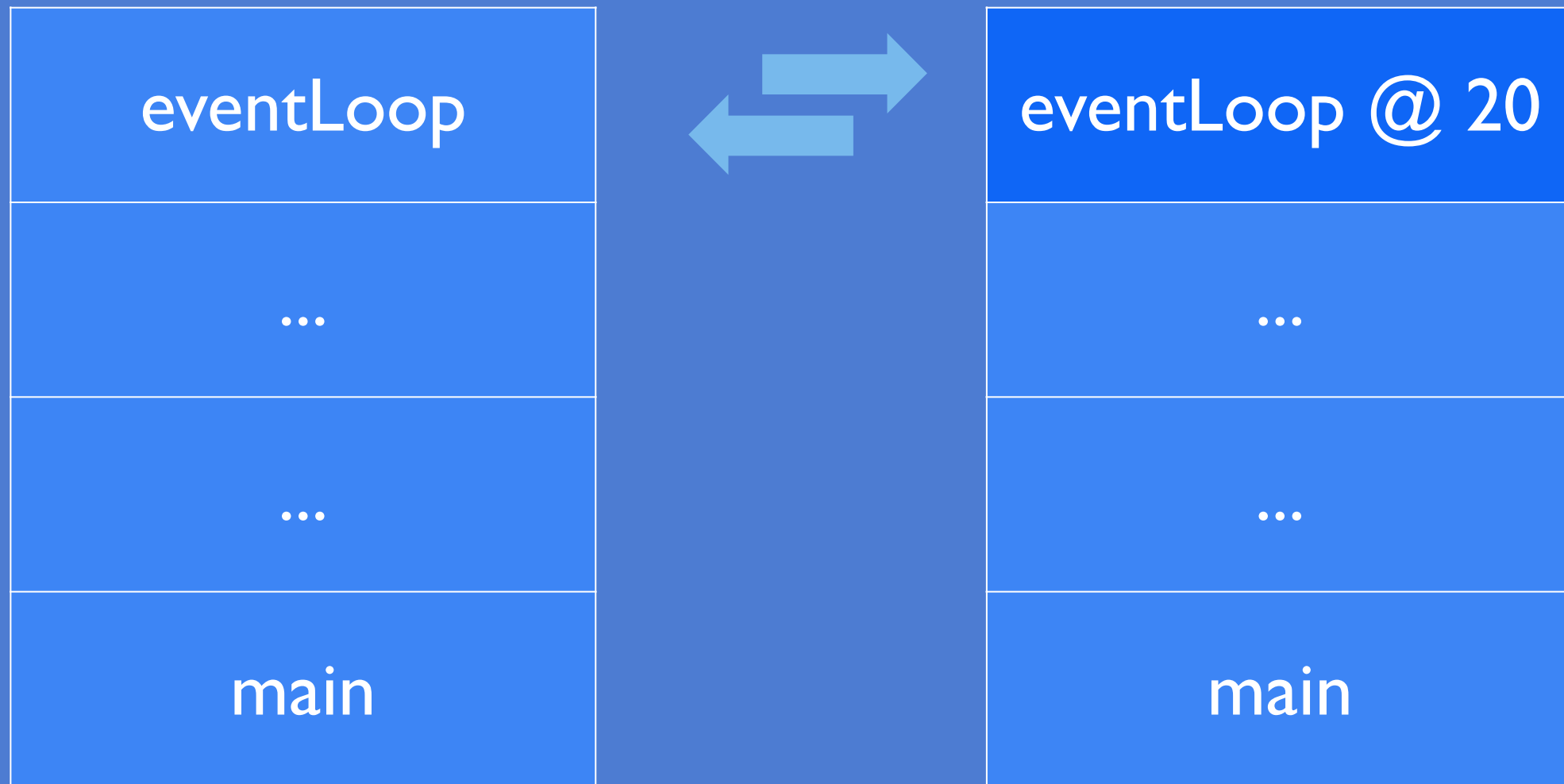
163	1	java.lang.String::charAt (29 bytes)
166	2	java.lang.String::hashCode (55 bytes)
171	3	java.lang.String::indexOf (70 bytes)
193	4	example03.BackedgeCounter::hotMethod (1 bytes)
194	5 %	example03.BackedgeCounter::main @ 5 (65 bytes)

Waiting for compiler...

5196	6 %	example03.BackedgeCounter::main @ 37 (65 bytes)
------	-----	---

Waiting for compiler...

# On-Stack Replacement



# Both Counters

```
public class BothCounters {  
    public static void main(final String[] args)  
        throws InterruptedException  
    {  
        for ( int i = 0; i < 2; ++i ) {  
            outerMethod();  
        }  
  
        System.out.println("Waiting for compiler...");  
        Thread.sleep(5000);  
    }  
  
    static void outerMethod() {  
        for ( int i = 0; i < 10_000; ++i ) {  
            innerMethod();  
        }  
    }  
  
    static void innerMethod() {}  
}
```

# Both Counters

-XX:+PrintCompilation

```
115    1 %    java.lang.String::indexOf @ 37 (70 bytes)
123    2      example04.BothCounters::innerMethod (1 bytes)
124    3      example04.BothCounters::outerMethod (19 bytes)
125    4 %    example04.BothCounters::outerMethod @ 5 (19 bytes)
```

Waiting for compiler...

# HotSpot:

A Tale of Two Compilers

C1 client

C2 server

# C1 Client VM

## The *Fast Acting* Compiler

Compiles with Count > 1,000  
(2,000 in Tiered)

Produces Compilations Quickly  
Generated Code Runs *Relatively* Slowly

# C2 Server VM

The *Smart* compiler

2x  
Speed!

Compiles with Count > 10,000  
(15,000 in Tiered)

Produces Compilations Slowly  
Generated Code Runs Fast

Profile Guided  
Speculative



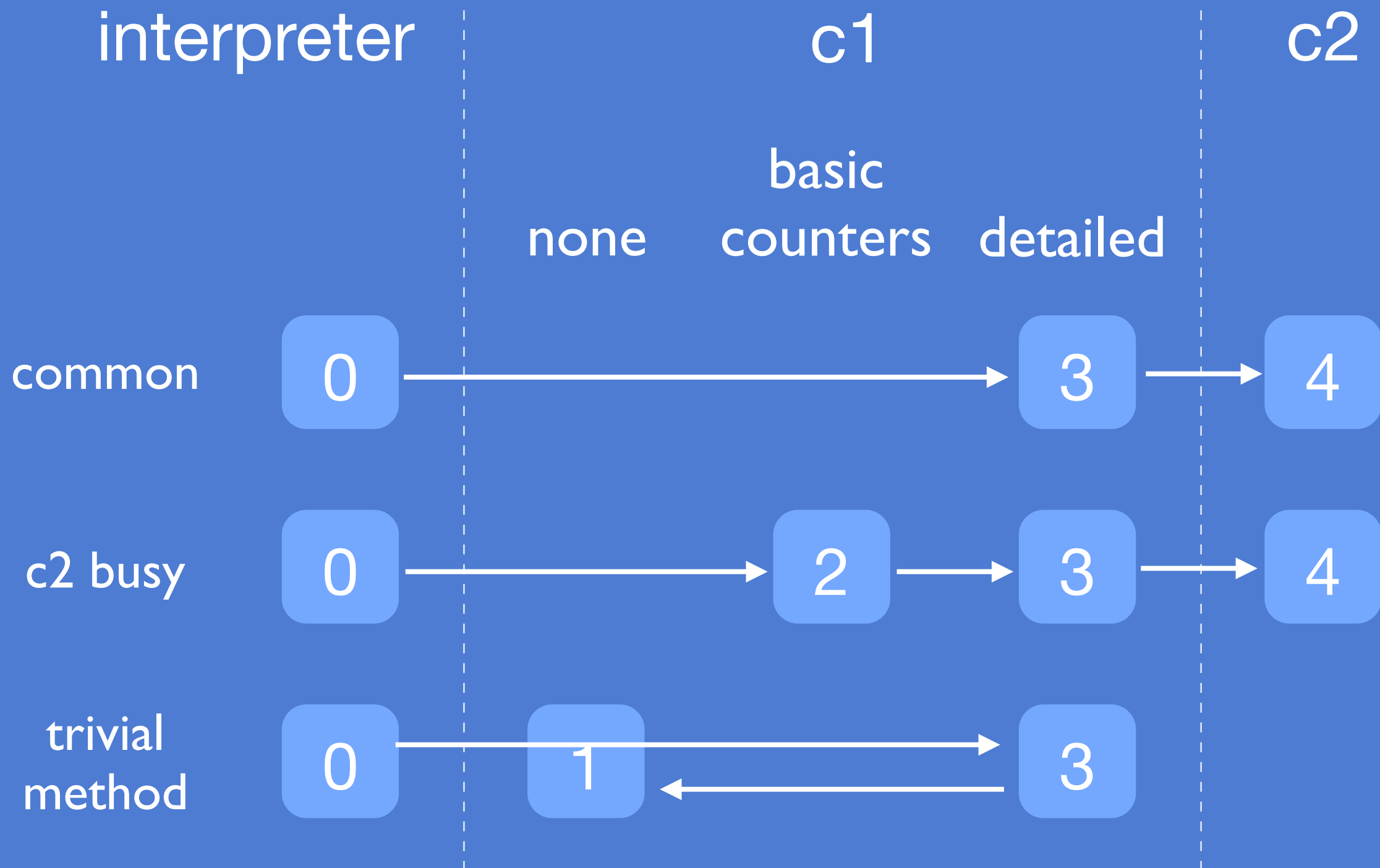
# Tiered Compilation

Available in Java 7 - default in Java 8

Best of Both Worlds - C1 & C2



# Tiered Compilation



# Tiered Compilation

```
public final class TieredCompilation {
    public static final void main(final String[] args)
        throws InterruptedException
    {
        for ( int i = 0; i < 3_000; ++i ) {
            method();
        }

        System.out.println("Waiting for the compiler...");
        Thread.sleep(5_000);

        for ( int i = 0; i < 20_000; ++i ) {
            method();
        }

        System.out.println("Waiting for the compiler...");
        Thread.sleep(5_000);
    }

    private static final void method() {
        // Do something while doing nothing.
        System.out.print('\0');
    }
}
```

# Tiered Compilation

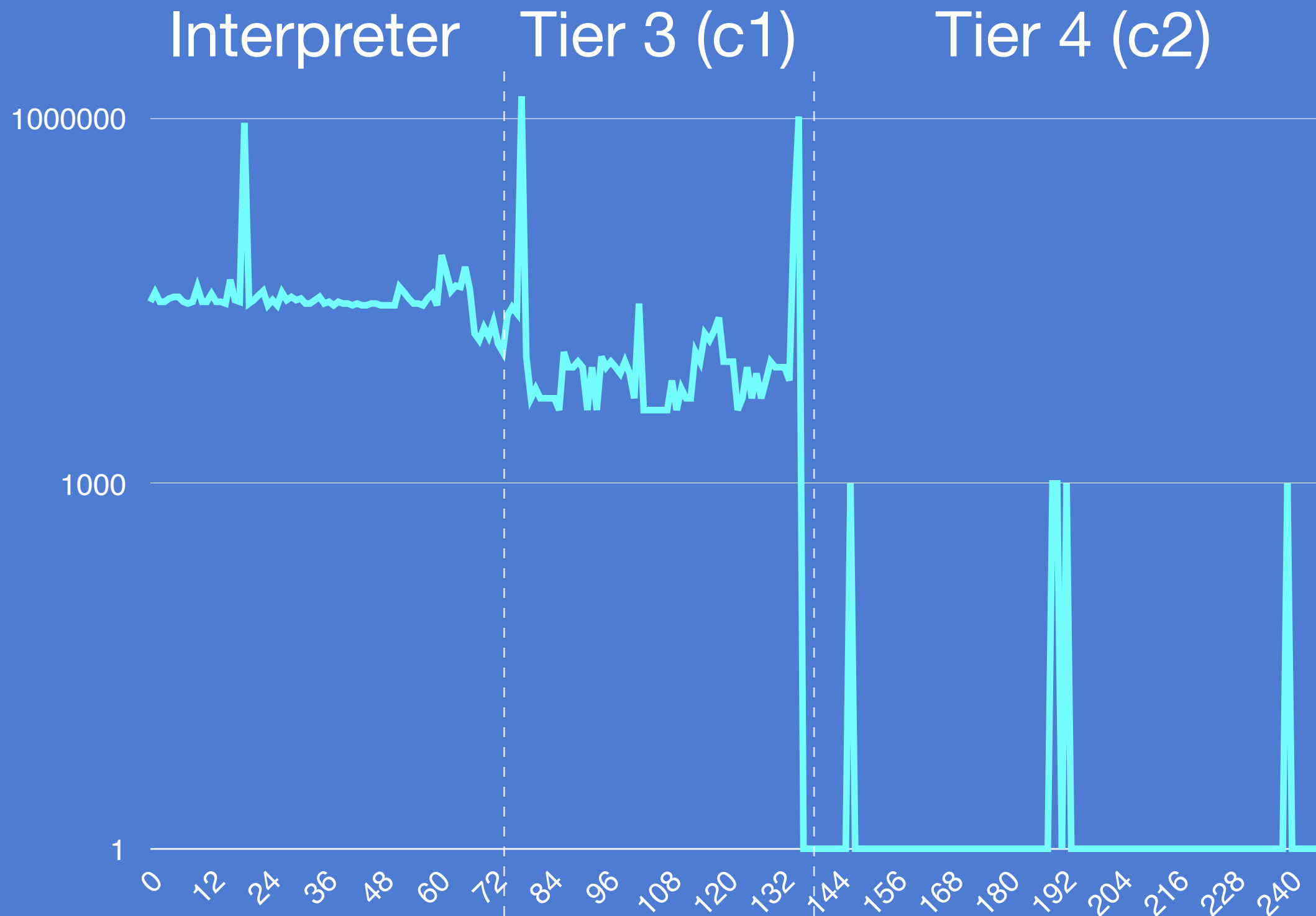
-XX:+TieredCompilation

-XX:+PrintCompilation

tier  
↓

183	69	3	sun...SingleByte\$Encoder::encodeArrayLoop (236 bytes)
...			
5237	101	4	sun...SingleByte\$Encoder::encodeArrayLoop (236 bytes)
5255	69	3	sun...SingleByte\$Encoder::encodeArrayLoop (236 bytes) made not entrant
...			
131	3	3	java.lang.Object::<init> (1 bytes)
...			
140	15	1	java.lang.Object::<init> (1 bytes)
141	3	3	java.lang.Object::<init> (1 bytes) made not entrant
...			
126	6 n 0		java.lang.System::arraycopy (native) (static)

# Tiered Compilation



# Optimizations

# Intrinsics

Special code built-into the VM for a particular method

Built-in to the VM - not written in Java (usually)

Often use special hardware capabilities:

MMX, AVX2, etc

## For Example...

`System.arraycopy`

`Math.sin / cos / tan`

# Common Sub-Expression Elimination

```
int a = b * c + g;  
int d = b * c * e;
```



```
int tmp = b * c;  
int a = tmp + g;  
int d = tmp * e;
```



# Loop Unswitching

```
bool isDebugEnabled = LOGGER.isDebugEnabled();
for ( User user: users ) {
    ...do something...
    if ( isDebugEnabled ) {
        LOGGER.debug(user.getName());
    }
}
```




```
bool isDebugEnabled = LOGGER.isDebugEnabled();
if ( isDebugEnabled ) {
    for ( User user: users ) {
        ...do something...
        LOGGER.debug(user.getName());
    }
} else {
    for ( User user: users ) {
        ...do something...
    }
}
```

# Dead Code Elimination

```
public int ArrayList::indexOf(Object o) {  
    if (o == null) {  
        for (int i = 0; i < size; i++)  
            if (elementData[i]==null)  
                return i;  
    } else {  
        for (int i = 0; i < size; i++)  
            if (o.equals(elementData[i]))  
                return i;  
    }  
    return -1;  
}
```

# Lock Coarsening

```
StringBuffer buffer = ...  
buffer.append("Hello");  
buffer.append(name);  
buffer.append("\n");
```



```
StringBuffer buffer = ...  
lock(buffer); buffer.append("Hello"); unlock(buffer);  
lock(buffer); buffer.append(name); unlock(buffer);  
lock(buffer); buffer.append("\n"); unlock(buffer);
```



```
StringBuffer buffer = ...  
lock(buffer);  
buffer.append("Hello");  
buffer.append(name);  
buffer.append("\n");  
unlock(buffer);
```

IPO: Inter-procedural Optimization

# Inlining

“The mother of all optimizations.”

# Intrinsic Inlining

```
public class Intrinsics {  
    public static void main(String[] args)  
        throws InterruptedException  
    {  
        int[] data = randomInts(100_000);  
  
        int min = Integer.MAX_VALUE;  
        for ( int x: data ) {  
            min = Math.min(min, x);  
        }  
  
        Thread.sleep(5_000);  
  
        System.out.println(min);  
    }  
  
    static final int[] randomInts(int size) {  
        ...  
    }  
}
```

# Intrinsic Inlining

-XX:+PrintCompilation

-XX:+UnlockDiagnosticVMOptions

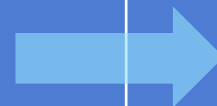
-XX:+PrintInlining

```
376      8 %   example06.Intrinsics::randomInts @ 13 (30 bytes)
@ 16      java.util.concurrent.ThreadLocalRandom::nextInt (8 bytes)   inline (hot)
@ 1      java.util.concurrent.ThreadLocalRandom::nextSeed (32 bytes)   inline (hot)
@ 3      java.lang.Thread::currentThread (0 bytes)   (intrinsic)
@ 18     sun.misc.Unsafe::getLong (0 bytes)   (intrinsic)
@ 27     sun.misc.Unsafe::putLong (0 bytes)   (intrinsic)
@ 4      java.util.concurrent.ThreadLocalRandom::mix32 (26 bytes)   inline (hot)
379      9      java.lang.Math::min (11 bytes)
379     10 %   example06.Intrinsics::main @ 22 (58 bytes)
@ 30     java.lang.Math::min (11 bytes)   (intrinsic)
```

# Direct Call Inlining

## static, private, constructor calls

```
public class Inlining {  
    public static void main(String[] args)  
        throws InterruptedException  
    {  
        System.setOut(new NullPrintStream());  
        for ( int i = 0; i < 20_000; ++i ) {  
            hotMethod();  
        }  
        Thread.sleep(5_000);  
    }  
  
    public static void hotMethod() {  
        System.out.println(square(7));  
        System.out.println(square(9));  
    }  
  
    static int square(int x) {  
        return x * x;  
    }  
}
```



```
public static void hotMethod() {  
    System.out.println(7 * 7);  
    System.out.println(9 * 9);  
}
```

# Direct Call Inlining

-XX:+PrintCompilation

-XX:+UnlockDiagnosticVMOptions

-XX:+PrintInlining

```
226  53  example07.DirectInlining::hotMethod (23 bytes)
      @ 5  example07.DirectInlining::square (4 bytes)  inline (hot)
!m    @ 8  java.io.PrintStream::println (24 bytes)  already compiled into a big method
      @ 16 example07.DirectInlining::square (4 bytes)  inline (hot)
!m    @ 19 java.io.PrintStream::println (24 bytes)  already compiled into a big method
228  54 %  example07.DirectInlining::main @ 15 (35 bytes)
      @ 15 example07.DirectInlining::hotMethod (23 bytes)  inline (hot)
      @ 5  example07.DirectInlining::square (4 bytes)  inline (hot)
!m    @ 8  java.io.PrintStream::println (24 bytes)  already compiled into a big method
      @ 16 example07.DirectInlining::square (4 bytes)  inline (hot)
!m    @ 19 java.io.PrintStream::println (24 bytes)  already compiled into a big method
```



# Printing Assembly for a Method

Download hsdis-amd64 dynamic library

Copy to jre/lib directory

Run HotSpot with...

-XX:+UnlockDiagnosticVMOptions  
-XX:CompileCommand=print,{package/Class::method}

# Direct Call Inlining

-XX:+UnlockDiagnosticVMOptions

-XX:CompileCommand=print,example07/DirectInlining::hotMethod

```
0x00000000106c57843: mov    $0x31,%edx
0x00000000106c57848: data32 xchg %ax,%ax
0x00000000106c5784b: callq  0x00000000106c10b60
; OopMap{rbp=Oop off=48}
; *invokevirtual println
; - DirectInlining::hotMethod@7 (line 17)
; {optimized virtual_call}
```

```
0x00000000106c5785d: mov    $0x51,%edx
0x00000000106c57862: nop
0x00000000106c57863: callq  0x00000000106c10b60
; OopMap{off=72}
; *invokevirtual println
; - DirectInlining::hotMethod@17 (line 18)
; {optimized virtual_call}
```

# Escape Analysis

```
long sum = 0;  
for ( Long x: list ) {  
    sum += iter.next();  
}
```



```
long sum = 0;  
for ( Iterator<Long> iter = list.iterator();  
      iter.hasNext(); )  
{  
    sum += iter.next();  
}
```



# Escape Analysis

```
long sum = 0;

// ArrayList$Itr.<init>
int iter$size = list.size();
int iter$cursor = 0;
int iter$lastRet = -1;

// ArrayList$Itr.hasNext
for ( ; iter$cursor != iter$size; ) {
    // ArrayList$Itr.next
    int i = iter$cursor;
    Object[] elementData = list.elementData;
    iter$cursor = i + 1;
    Long x = (Long)elementData[iter$lastRet = i];

    sum += x;
}
```

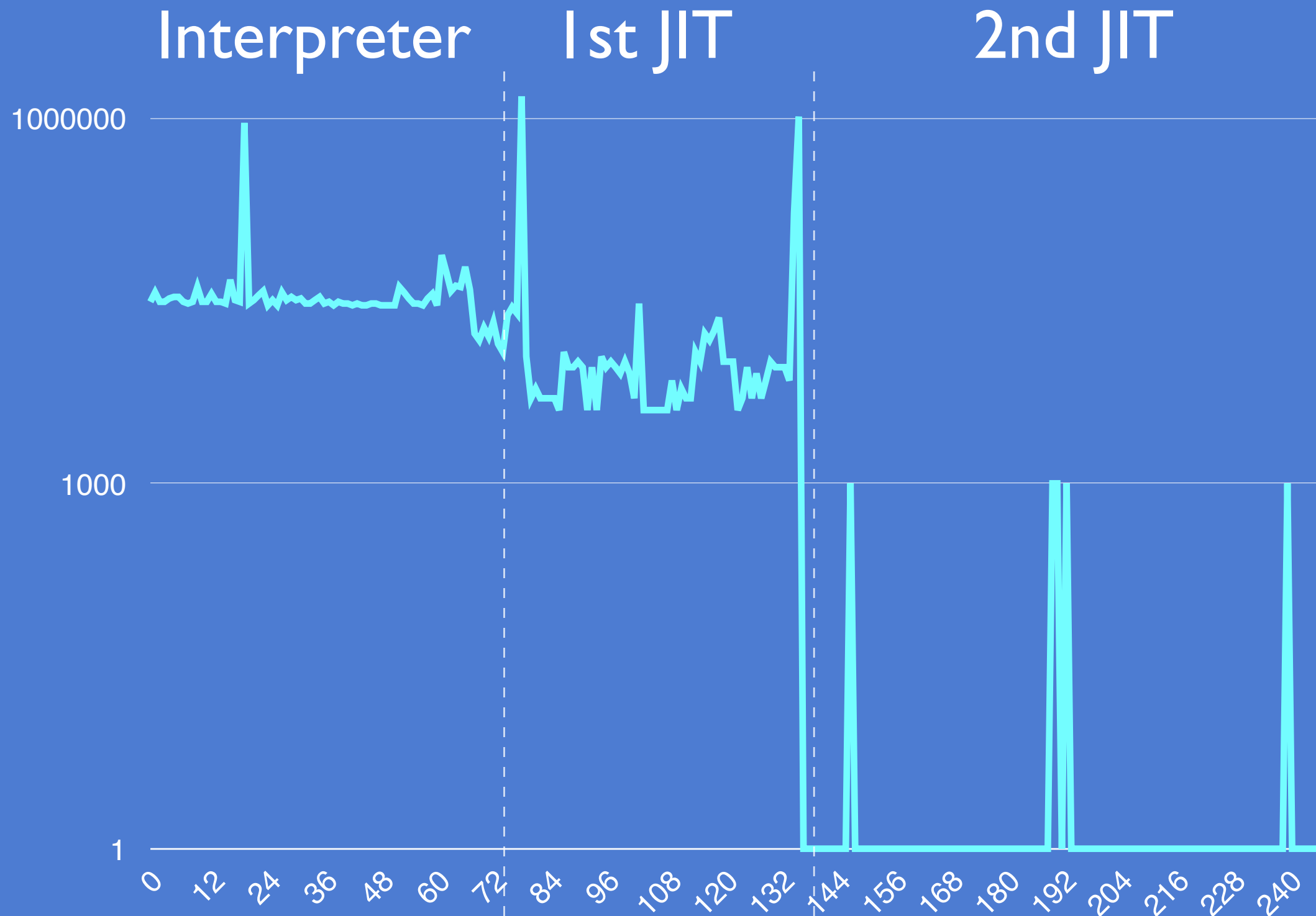
# Simple Program Revisited

```
public class SimpleProgram {  
    static final int CHUNK_SIZE = 1_000;  
  
    public static void main(String[] args) {  
        for ( int i = 0; i < 250; ++i ) {  
            long startTime = System.nanoTime();  
  
            for ( int j = 0; j < CHUNK_SIZE; ++j ) {  
                //new Object  
                obj = calloc(sizeof(Object));  
                obj.<init>(); // call ctor - empty  
            }  
  
            long endTime = System.nanoTime();  
            System.out.printf(  
                "%d\t%d\n", i, endTime - startTime);  
        }  
    }  
}
```

Annotations for the inner loop:

- 3: empty loop / eliminate loop (points to the loop header)
- 2: escape analysis / dead store (points to the `obj = calloc(...)` line)
- 1: inlined (points to the `obj.<init>()` line)

# Simple Program Revisited



That's Boring!

Speculative  
Optimizations

# Implicit Null Check

```
public class NullCheck {
    public static void main(String[] args)
        throws InterruptedException
    {
        for ( int i = 0; i < 20_000; ++i ) {
            hotMethod("hello");
        }

        Thread.sleep(5_000);

        for ( int i = 0; i < 10; ++i ) {
            System.out.printf("tempting fate %d%n", i);
            try {
                hotMethod(null);
            } catch ( NullPointerException e ) {
                // ignore
            }
        }
    }

    static final void hashIt(final Object value) {
        value.hashCode();
    }
}
```



# Implicit Null Check

-XX:+UnlockDiagnosticVMOptions

-XX:CompileCommand=print,example08a/NullCheck::hotMethod

```
0x0000000010795f9c0: mov    %eax,-0x14000(%rsp)
0x0000000010795f9c7: push  %rbp
0x0000000010795f9c8: sub    $0x50,%rsp
    ;*synchronization entry
example08a.NullCheck::doSomething@-1 (line 26)
0x0000000010795f9cc: mov    0x8(%rsi),%r10d
    ; implicit exception: dispatches to 0x0000000010795fe1d
0x0000000010795f9d0: movabs $0x7eaa80d10,%r11
    ; {oop(a 'java/lang/Class' = 'java/lang/System')}
0x0000000010795f9da: mov    0x74(%r11),%r11d
    ;*getstatic out
```

# Implicit Null Check

```
if ( value == null ) {  
    throw new NullPointerException();  
}
```

value.toString();

Possible, but  
improbable NPE

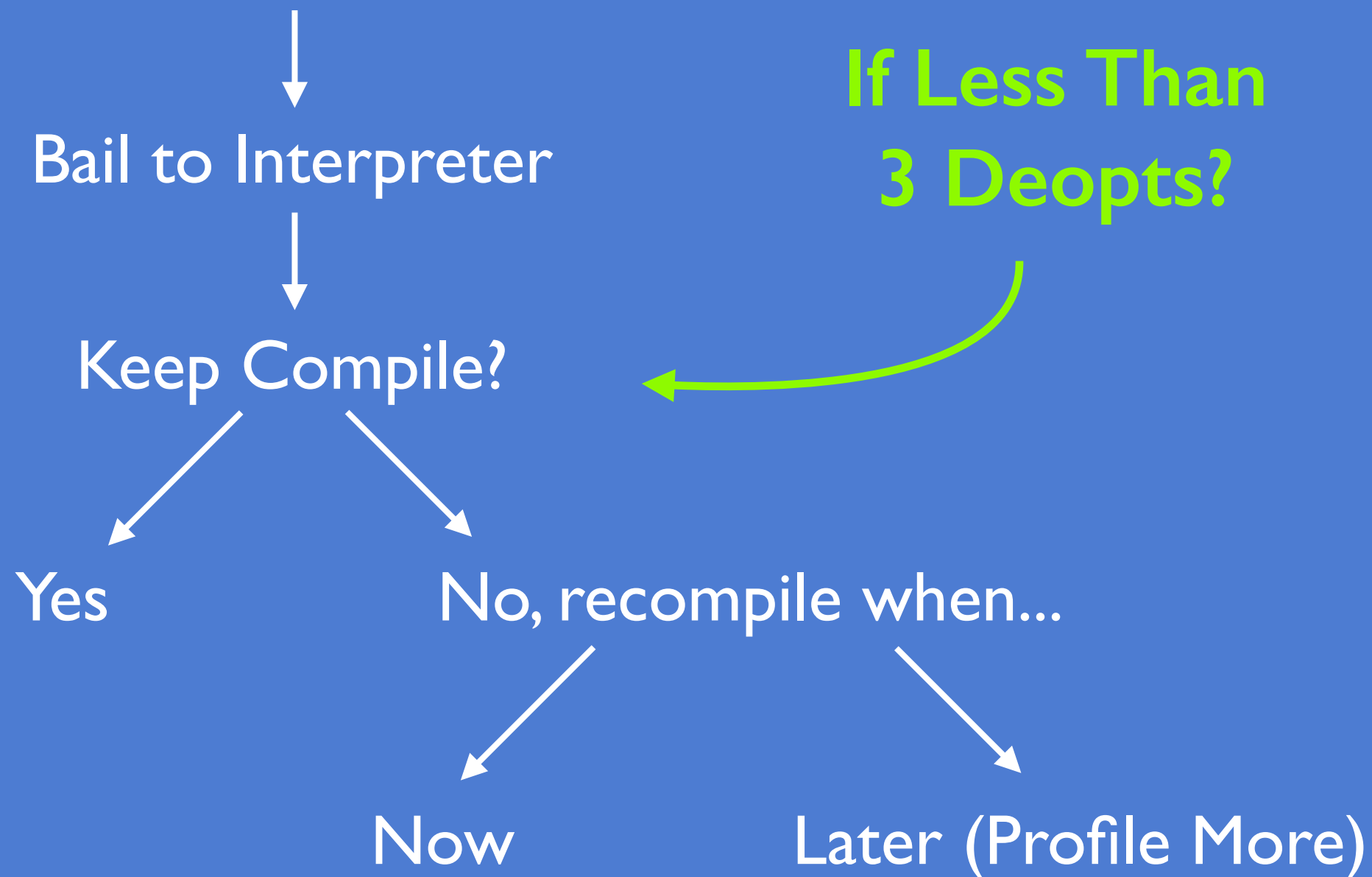


# Null Check Deoptimization

-XX:+PrintCompilation

124	1	java.lang.String::hashCode (55 bytes)	
138	2	example08a.NullCheck::hotMethod (6 bytes)	
138	3 % !	example08a.NullCheck::main @ 5 (69 bytes)	
tempting	fate 0		
tempting	fate 1		
tempting	fate 2		
5147	2	example08a.NullCheck::hotMethod (6 bytes)	made not entrant
tempting	fate 3		
tempting	fate 4		
tempting	fate 5		
tempting	fate 6		
tempting	fate 7		
tempting	fate 8		
tempting	fate 9		

# Not Thrown Away the First Time



# Hot Exception Optimization

```
int caughtCount = 0;
Set<NullPointerException> nullPointerExceptions =
    new HashSet<>();

for ( Object object : objects ) {
    try {
        object.toString();
    } catch ( NullPointerException e ) {
        nullPointerExceptions.add( e );
        caughtCount += 1;
    }
}
```

Null Proportion: 0.100000	Caught: 10057	Unique: 2015
Null Proportion: 0.500000	Caught: 50096	Unique: 7191
Null Proportion: 0.900000	Caught: 89929	Unique: 11030

# Hot Exceptions

```
int caughtCount = 0;
HashSet<NullPointerException> nullPointerExceptions =
    new HashSet<>();


for ( Object object : objects ) {
    try {
        object.toString();
    } catch ( NullPointerException e ) {
        boolean added = nullPointerExceptions.add(e);
        if ( !added ) e.printStackTrace();
        caughtCount += 1;
    }
}
```

java.lang.NullPointerException

No StackTrace???

# Unreached Deoptimization

```
public class Unreached {  
    public static volatile Object thing = null;  
  
    public static void main(final String[] args)  
        throws InterruptedException  
    {  
        for ( int i = 0; i < 20_000; ++i ) {  
            hotMethod();  
        }  
  
        Thread.sleep(5_000);  
        thing = new Object(); ← phase change  
  
        for ( int i = 0; i < 20_000; ++i ) {  
            hotMethod();  
        }  
        Thread.sleep(5_000);  
    }  
  
    static final void hotMethod() {  
        if ( thing == null )  
            System.out.print("");  
        else  
            System.out.print("");  
    }  
}
```



```
static final void hotMethod() {  
    if ( thing == null )  
        System.out.print("");  
    else  
        uncommon_trap(unreached);  
}
```

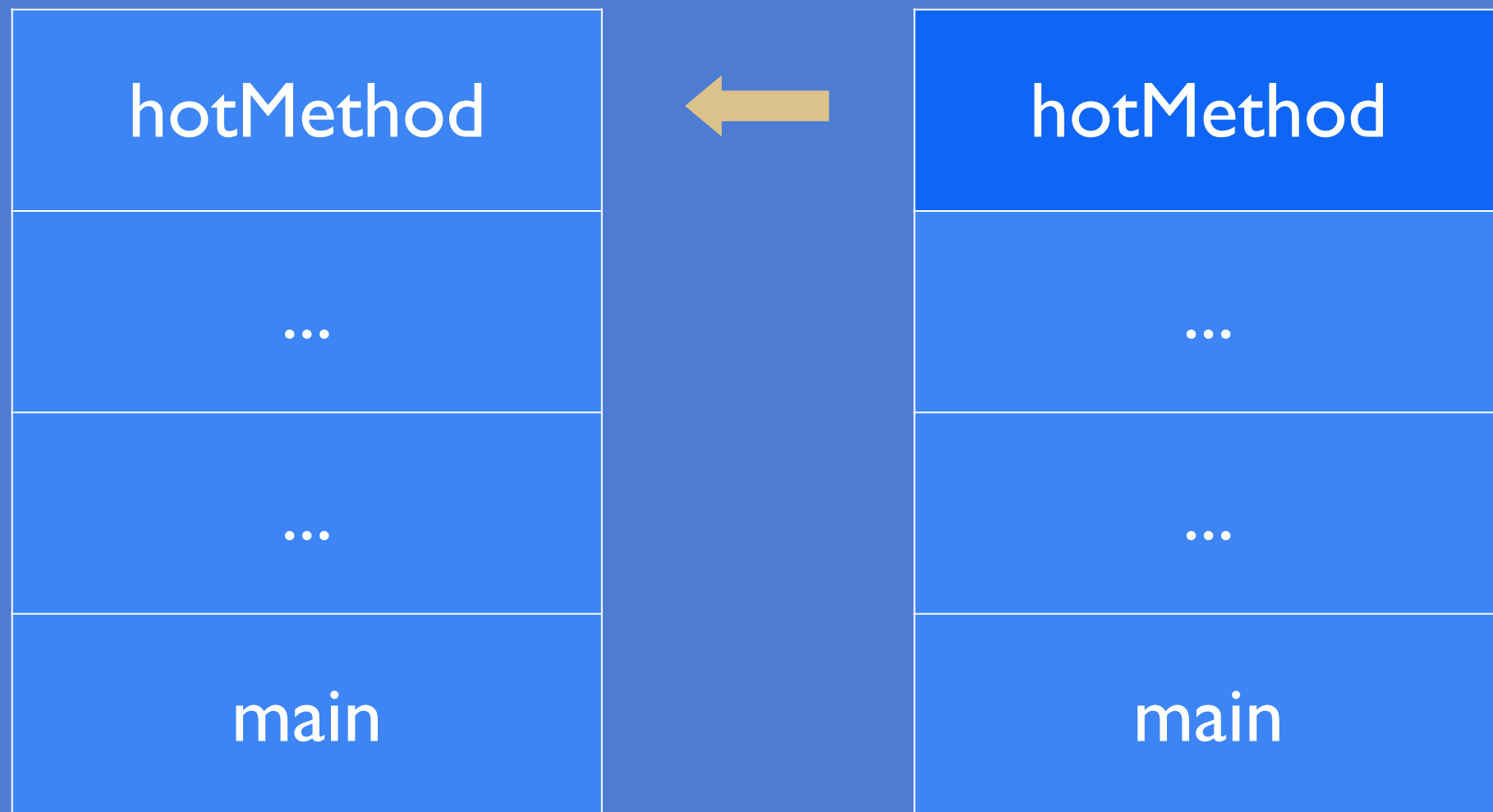
# Unreached Deoptimization

-XX:+PrintCompilation

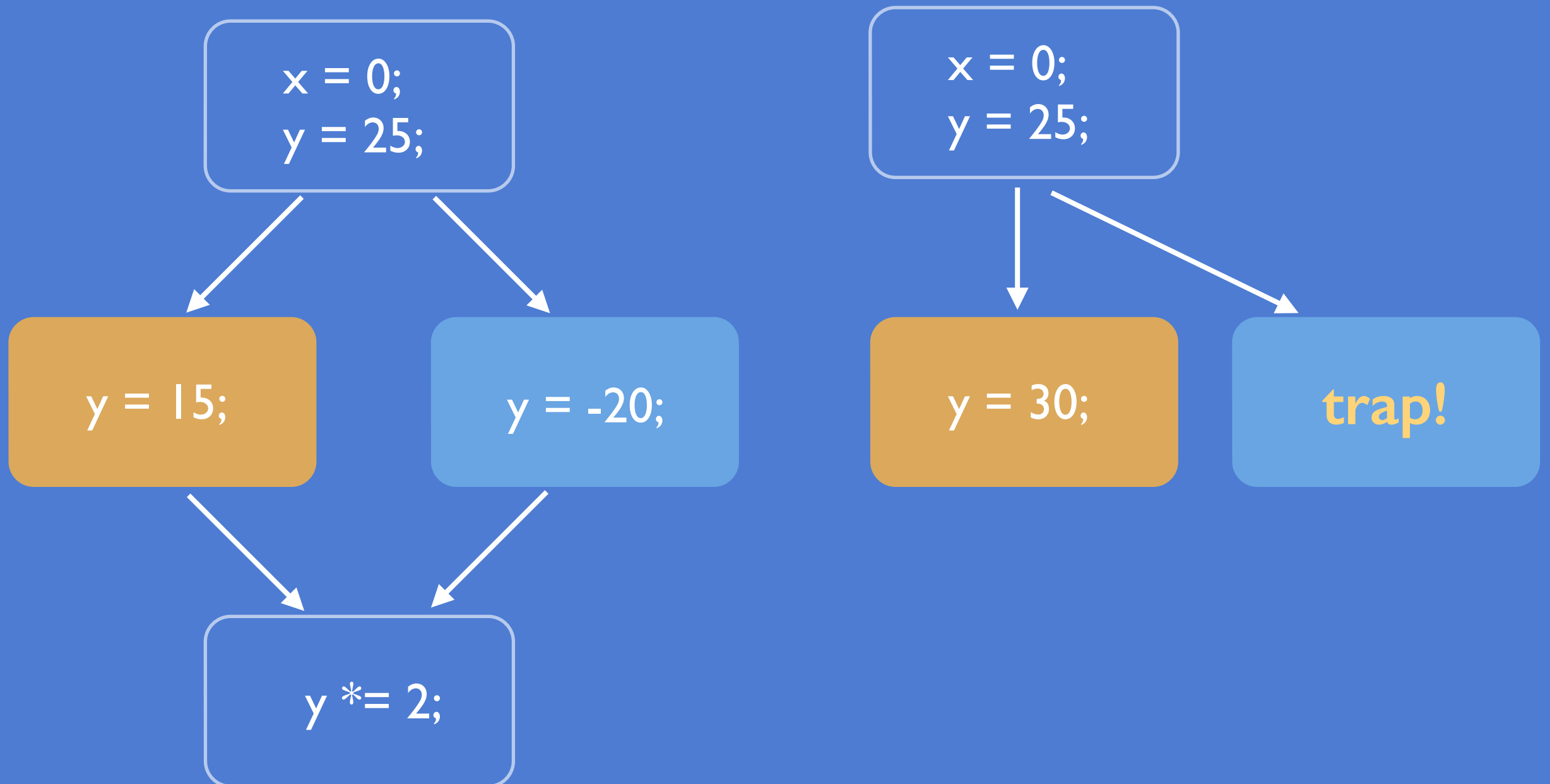
217	1		java.lang.String::hashCode (55 bytes)	
235	2		java.lang.String::indexOf (70 bytes)	
238	3		java.io.BufferedWriter::ensureOpen (18 bytes)	
244	4		java.lang.String::length (6 bytes)	
244	5		java.lang.String::indexOf (7 bytes)	
245	6		java.nio.Buffer::position (5 bytes)	
245	7		example09.Unreached::hotMethod (26 bytes)	
...				
265	14		java.io.OutputStreamWriter::flushBuffer (8 bytes)	
265	15	!	sun.nio.cs.StreamEncoder::flushBuffer (42 bytes)	
267	16		sun.nio.cs.StreamEncoder::isOpen (5 bytes)	
267	17		sun.nio.cs.StreamEncoder::implFlushBuffer (15 bytes)	
267	18	%	example09.Unreached::main @ 5 (59 bytes)	
5255	7		example09.Unreached::hotMethod (26 bytes)	made not entrant
5257	19		example09.Unreached::hotMethod (26 bytes)	
5257	20	%	example09.Unreached::main @ 39 (59 bytes)	



# Bail to Interpreter

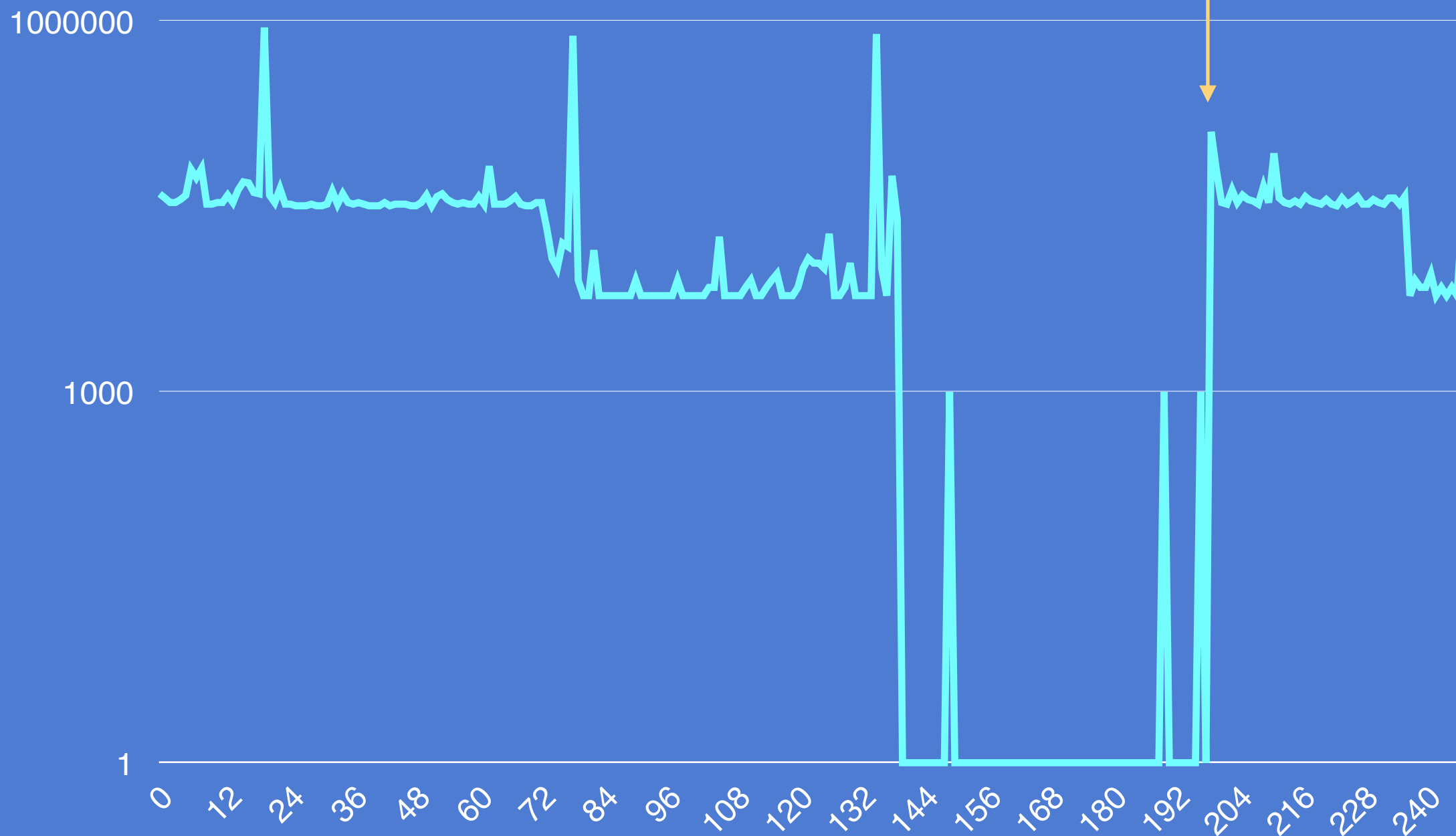


# Why Speculate?



# Not So Simple Program Revisited

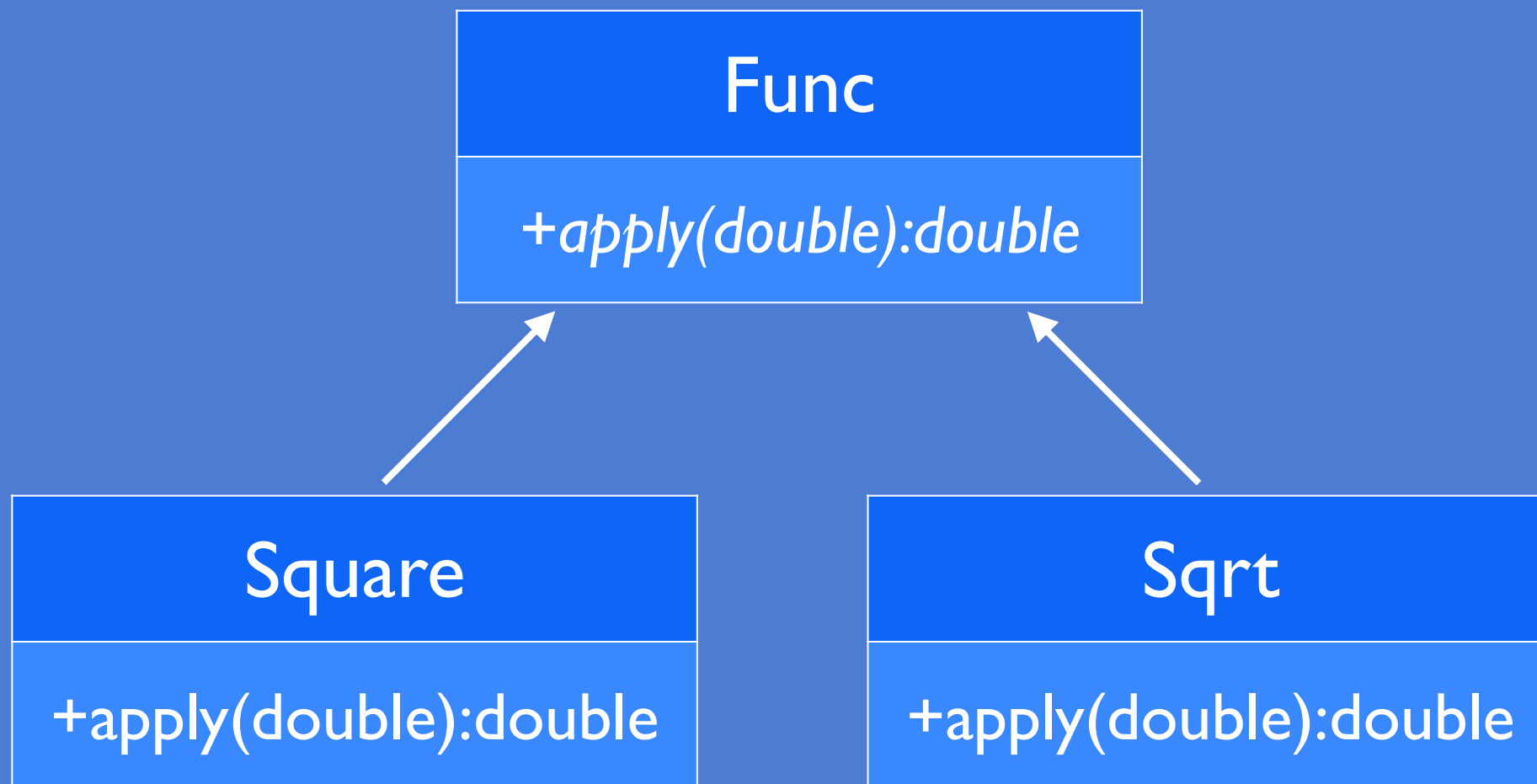
unreached deopt,  
bail to interpreter!



# Virtual Call Inlining

Class Hierarchy Analysis (CHA)

Type Profile



# Monomorphic

```
public class Monomorphic {  
    public static void main(String[] args)  
        throws InterruptedException  
    {  
        Func func = new Square();  
  
        for ( int i = 0; i < 20_000; ++i ) {  
            apply(func, i);  
        }  
  
        Thread.sleep(5_000);  
    }  
  
    static double apply(Func func, int x) {  
        return func.apply(x);  
    }  
}
```

# Monomorphic

-XX:+PrintCompilation

-XX:+UnlockDiagnosticVMOptions

-XX:+PrintInlining

```
217      1      java.lang.String::hashCode (55 bytes)
234      2      example10a.Monomorphic::apply (7 bytes)
234      3      example10.support.Square::apply (4 bytes)
    @ 3      example10.support.Square::apply (4 bytes)    inline (hot)
234      4 %    example10a.Monomorphic::main @ 13 (30 bytes)
    @ 15     example10a.Monomorphic::apply (7 bytes)    inline (hot)
    @ 3      example10.support.Square::apply (4 bytes)    inline (hot)
```

# Beyond Monomorphic

```
Func func = ...  
double result = func.apply(20);
```



```
Func func = ...  
//no type guard!  
double result = 20 * 20;
```



## More Types?

# Potential for Deopt Storm

```
public class ChaStorm {  
    public static void main(String[] args)  
        throws InterruptedException  
    {  
        Func func = new Square();  
  
        for ( int i = 0; i < 10_000; ++i ) {  
            apply1(func, i);  
            ...  
            apply8(func, i);  
        }  
  
        System.out.println("Waiting for compiler...");  
        Thread.sleep(5_000);  
  
        System.out.println("Deoptimize...");  
        System.out.println(Sqrt.class);  
  
        Thread.sleep(5_000);  
    }  
}
```

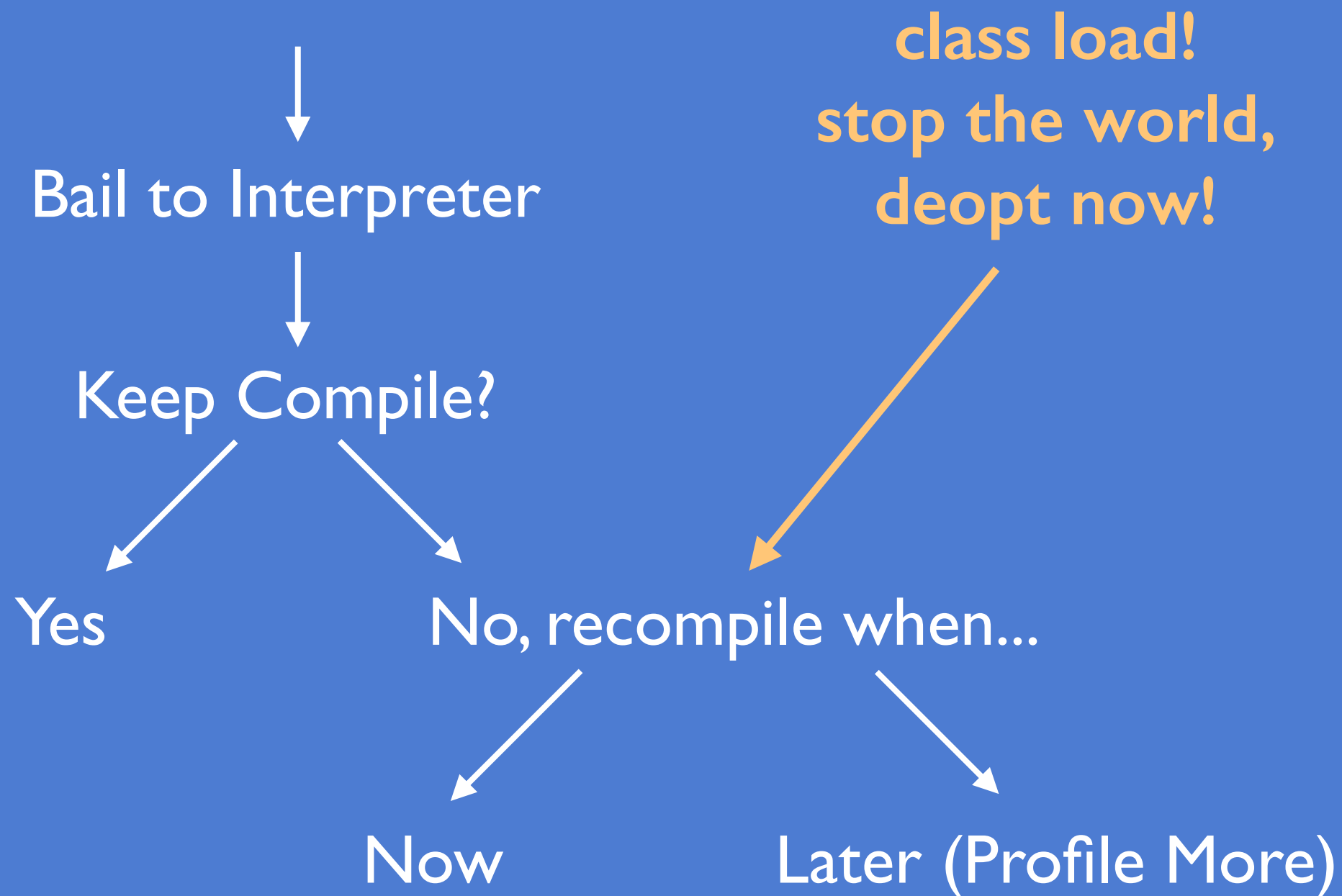


# Potential for Deopt Storm

-XX:+PrintCompilation

```
152      1      java.lang.String::hashCode (55 bytes)
166      2      example10.support.Square::apply (4 bytes)
173      3      example10b.ChaStorm::apply1 (7 bytes)
173      4      example10b.ChaStorm::apply2 (7 bytes)
Waiting for compiler...
174      5      example10b.ChaStorm::apply3 (7 bytes)
174      6      example10b.ChaStorm::apply4 (7 bytes)
174      7      example10b.ChaStorm::apply5 (7 bytes)
174      8      example10b.ChaStorm::apply6 (7 bytes)
174      9      example10b.ChaStorm::apply7 (7 bytes)
174     10      example10b.ChaStorm::apply8 (7 bytes)
Deoptimize...
5176      9      example10b.ChaStorm::apply7 (7 bytes)      made not entrant
5176      8      example10b.ChaStorm::apply6 (7 bytes)      made not entrant
5176      7      example10b.ChaStorm::apply5 (7 bytes)      made not entrant
5176      5      example10b.ChaStorm::apply3 (7 bytes)      made not entrant
5176      6      example10b.ChaStorm::apply4 (7 bytes)      made not entrant
5176      4      example10b.ChaStorm::apply2 (7 bytes)      made not entrant
5176      3      example10b.ChaStorm::apply1 (7 bytes)      made not entrant
5176     10      example10b.ChaStorm::apply8 (7 bytes)      made not entrant
class example10.support.Sqrt
```

# Another Way to Deopt



# Another Way to Deopt


-XX:+PrintCompilation  
-XX+PrintSafepointStatistics  
-XX:PrintSafepointStatisticsCount=1

Total time for which application threads were stopped: 0.0001010 seconds


5096	10	example10b.ChaStorm::apply8 (7 bytes)	made not entrant
5096	8	example10b.ChaStorm::apply6 (7 bytes)	made not entrant
5096	7	example10b.ChaStorm::apply5 (7 bytes)	made not entrant
5096	5	example10b.ChaStorm::apply3 (7 bytes)	made not entrant
5096	6	example10b.ChaStorm::apply4 (7 bytes)	made not entrant
5096	4	example10b.ChaStorm::apply2 (7 bytes)	made not entrant
5096	9	example10b.ChaStorm::apply7 (7 bytes)	made not entrant
5096	3	example10b.ChaStorm::apply1 (7 bytes)	made not entrant
vmop		[threads: total initially_running wait_to_block]	...
5.096: Deoptimize		[ 7 0 0 ]	...

# Bimorphic

```
Func func = ...  
double result = func.apply(20);
```



```
Func func = ...  
//no type guard!  
double result = 20 * 20;
```



```
Func func = ...  
double result;  
if ( func.getClass().equals(Square.class) ) {  
    result = 20 * 20;  
} else {  
    uncommon_trap(class_check);  
}
```



add another type

give up!



# Class Devirtualization

```
public class ClassDevirtualization {  
    public static void main(String[] args) throws InterruptedException {  
        System.out.println("Using Square...");  
        Func func = new Square();  
        for ( int i = 0; i < 20_000; ++i ) {  
            apply1(func, i);  
            apply2(func, i);  
        }  
        Thread.sleep(5_000);
```

```
        System.out.printf("Loading %s to Deopt Now!%n", Sqrt.class);
```

**stop the world,  
deopt now!**

```
        System.out.println("Keep using Square in apply1...");  
        func = new Square();  
        for ( int i = 0; i < 20_000; ++i ) apply1(func, i);  
        Thread.sleep(5_000);
```

```
        System.out.println("Use AlsoSquare in apply1...");  
        func = new AlsoSquare();  
        for ( int i = 0; i < 20_000; ++i ) apply1(func, i);  
        Thread.sleep(5_000);
```

**class check  
deopt!**

```
        System.out.println("Use AnotherSquare in apply1...");  
        func = new AnotherSquare();  
        for ( int i = 0; i < 20_000; ++i ) apply1(func, i);  
        Thread.sleep(5_000);
```

**bimorphic  
deopt!**

```
        ...after 3 types no more deopts...
```

```
    }  
}
```

# Class Devirtualization

-XX:+PrintCompilation

```
89      1      java.lang.String::hashCode (55 bytes)
Using Square...
104     2      example10.support.Square::apply (4 bytes)
105     3      example10c.ClassDevirtualization::apply1 (7 bytes)
105     4      example10c.ClassDevirtualization::apply2 (7 bytes)
106     5 %    example10c.ClassDevirtualization::main @ 21 (240 bytes)
5116    5 %    example10c.ClassDevirtualization::main @ -2 (240 bytes)   made not entrant
5116    4      example10c.ClassDevirtualization::apply2 (7 bytes)   made not entrant
5116    3      example10c.ClassDevirtualization::apply1 (7 bytes)   made not entrant
Loading class example10.support.Sqrt to Deoptimize Now!
Keep using Square in apply1...
5128    6      example10c.ClassDevirtualization::apply1 (7 bytes)
5128    7 %    example10c.ClassDevirtualization::main @ 88 (240 bytes)
Use AlsoSquare in apply1...
10131   6      example10c.ClassDevirtualization::apply1 (7 bytes)   made not entrant
10131   8      example10c.ClassDevirtualization::apply1 (7 bytes)
10132   9 %    example10c.ClassDevirtualization::main @ 131 (240 bytes)
Use AnotherSquare in apply1...
15134   8      example10c.ClassDevirtualization::apply1 (7 bytes)   made not entrant
15134  10      example10c.ClassDevirtualization::apply1 (7 bytes)
15135  11 %    example10c.ClassDevirtualization::main @ 174 (240 bytes)
Use YetAnotherSquare in apply1...
20139  12 %    example10c.ClassDevirtualization::main @ 217 (240 bytes)
```

# Worse Yet...

```
class Square extends Func {  
    double final exec(double x) {  
        return x * x;  
    }  
}
```

```
class AlsoSquare extends Square {}
```

```
class AnotherSquare extends Square {}
```

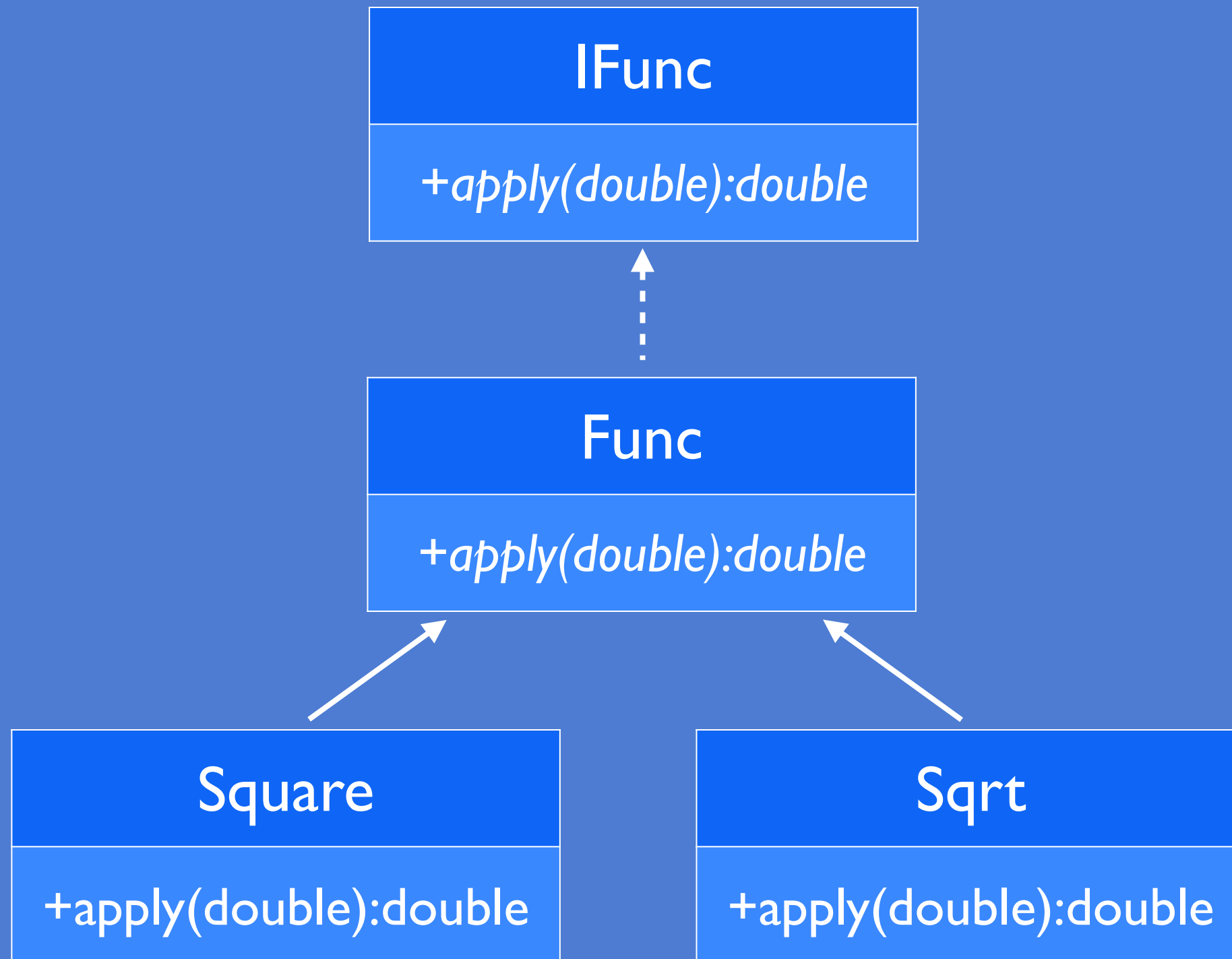
```
class YetAnotherSquare extends Square {}
```

# Unintentional Megamorphism

```
new HashMap<String, Integer>() {{  
    put("foo", 20);  
    put("bar", 30);  
}};
```



# No CHA for Interfaces



# No CHA for Interfaces

-XX:+PrintCompilation

```
68      1      java.lang.String::hashCode (55 bytes)
Using Square...
79      2      example10.support.Square::apply (4 bytes)
80      3      example10d.InterfaceDevirtualization::apply1 (9 bytes)
80      4      example10d.InterfaceDevirtualization::apply2 (9 bytes)
81      5 %    example10d.InterfaceDevirtualization::main @ 21 (240 bytes)
Loading class example10.support.Sqrt - no CHA for interfaces!
Keep using Square in apply1...
5090     6 %    example10d.InterfaceDevirtualization::main @ 88 (240 bytes)
Use AlsoSquare in apply1...
10094     5 %    example10d.InterfaceDevirtualization::main @ -2 (240 bytes)    made not entrant
10094     6 %    example10d.InterfaceDevirtualization::main @ -2 (240 bytes)    made not entrant
10094     3      example10d.InterfaceDevirtualization::apply1 (9 bytes)    made not entrant
10095     7      example10d.InterfaceDevirtualization::apply1 (9 bytes)
10095     8 %    example10d.InterfaceDevirtualization::main @ 131 (240 bytes)
Use AnotherSquare in apply1...
15101     7      example10d.InterfaceDevirtualization::apply1 (9 bytes)    made not entrant
15102     9      example10d.InterfaceDevirtualization::apply1 (9 bytes)
15102    10 %    example10d.InterfaceDevirtualization::main @ 174 (240 bytes)
```

# Inlining Numbers to Remember...

``java -XX:+PrintFlagsFinal``

MaxTrivialSize	6
MaxInlineSize	35
FreqInlineSize	325
MaxInlineLevel	9
MaxRecursiveInlineLevel	1
MinInliningThreshold	250
Tier I MaxInlineSize	8
Tier I FreqInlineSize	35

# “Fun” with Unloaded

```
public class UnloadedForever {  
    public static void main(String[] args) {  
        for ( int i = 0; i < 100_000; ++i ) {  
            try {  
                factory();  
            } catch ( Throwable t ) {  
                // ignore  
            }  
        }  
    }  
  
    static DoesNotExist factory() {  
        return new DoesNotExist();  
    }  
}
```

# Unloaded Forever

-XX:+PrintCompilation

72	1	java.lang.String::hashCode (55 bytes)	
156	2	java.lang.Object::<init> (1 bytes)	
183	3	s java.lang.Throwable::fillInStackTrace (29 bytes)	
183	4	n java.lang.Throwable::fillInStackTrace (native)	
183	5	java.lang.LinkageError::<init> (6 bytes)	
184	6	java.lang.Error::<init> (6 bytes)	
184	7	java.lang.Throwable::<init> (34 bytes)	
185	8	example11a.UnloadedForever::factory (8 bytes)	
186	9	java.lang.NoClassDefFoundError::<init> (6 bytes)	
186	8	example11a.UnloadedForever::factory (8 bytes)	made not entrant
223	10	% ! example11a.UnloadedForever::main @ 5 (23 bytes)	
273	11	example11a.UnloadedForever::factory (8 bytes)	
274	11	example11a.UnloadedForever::factory (8 bytes)	made not entrant
358	12	example11a.UnloadedForever::factory (8 bytes)	
358	12	example11a.UnloadedForever::factory (8 bytes)	made not entrant
441	13	example11a.UnloadedForever::factory (8 bytes)	
442	13	example11a.UnloadedForever::factory (8 bytes)	made not entrant
528	14	example11a.UnloadedForever::factory (8 bytes)	
978	8	example11a.UnloadedForever::factory (8 bytes)	made zombie

# “Fun” with Uninitialized

```
public class UninitializedForever {  
    static class Uninitialized {  
        static {  
            if ( true ) throw new RuntimeException();  
        }  
    }  
  
    public static void main(String[] args) {  
        for ( int i = 0; i < 100_000; ++i ) {  
            try {  
                new Uninitialized();  
            } catch ( Throwable t ) {  
                // ignore  
            }  
        }  
    }  
}
```

# Uninitialized Forever

-XX:+PrintCompilation

74	1	java.lang.String::hashCode (55 bytes)	
162	2	java.lang.Object::<init> (1 bytes)	
188	3	s java.lang.Throwable::fillInStackTrace (29 bytes)	
189	4	n java.lang.Throwable::fillInStackTrace (native)	
189	5	java.lang.LinkageError::<init> (6 bytes)	
190	6	java.lang.Error::<init> (6 bytes)	
190	7	java.lang.Throwable::<init> (34 bytes)	
191	8	java.lang.NoClassDefFoundError::<init> (6 bytes)	
233	9	% ! example11b.UninitializedForever::main @ 5 (25 bytes)	
241	9	% ! example11b.UninitializedForever::main @ -2 (25 bytes)	made not entrant
252	10	% ! example11b.UninitializedForever::main @ 5 (25 bytes)	
263	10	% ! example11b.UninitializedForever::main @ -2 (25 bytes)	made not entrant
272	11	% ! example11b.UninitializedForever::main @ 5 (25 bytes)	
281	11	% ! example11b.UninitializedForever::main @ -2 (25 bytes)	made not entrant
290	12	% ! example11b.UninitializedForever::main @ 5 (25 bytes)	
299	12	% ! example11b.UninitializedForever::main @ -2 (25 bytes)	made not entrant
308	13	% ! example11b.UninitializedForever::main @ 5 (25 bytes)	
318	13	% ! example11b.UninitializedForever::main @ -2 (25 bytes)	made not entrant
328	14	% ! example11b.UninitializedForever::main @ 5 (25 bytes)	
337	14	% ! example11b.UninitializedForever::main @ -2 (25 bytes)	made not entrant

# Reasons for Deoptimizing...

null_check	unexpected null or zero divisor
null_assert	unexpected non-null or non-zero
range_check	unexpected array index
class_check	unexpected object class
array_check	unexpected array class
intrinsic	unexpected operand to intrinsic
bimorphic	unexpected object class in bimorphic inlining
unloaded	unloaded class or constant pool entry
uninitialized	bad class state (uninitialized)
unreached	code is not reached, compiler
unhandled	arbitrary compiler limitation
constraint	arbitrary runtime constraint violated
div0_check	a null_check due to division by zero
age	nmethod too old; tier threshold reached
predicate	compiler generated predicate failed
loop_limit_check	compiler generated loop limits check failed



# Actions When Deoptimizing...

none	just interpret, do not invalidate nmethod
maybe_recompile	recompile the nmethod; need not invalidate
<b>make_not_entrant</b>	invalidate the nmethod, recompile (probably)
<b>reinterpret</b>	invalidate the nmethod, reset IC, maybe recompile
make_not_compilable	invalidate the nmethod and do not compile

uncommon trap!

class load!  
stop the world,  
deopt now!

Bail to Interpreter

Keep Compile?

Yes, recompile?

No, recompile when...

No

Yes

Now

Later (Profile More)

---

none

maybe  
recompile

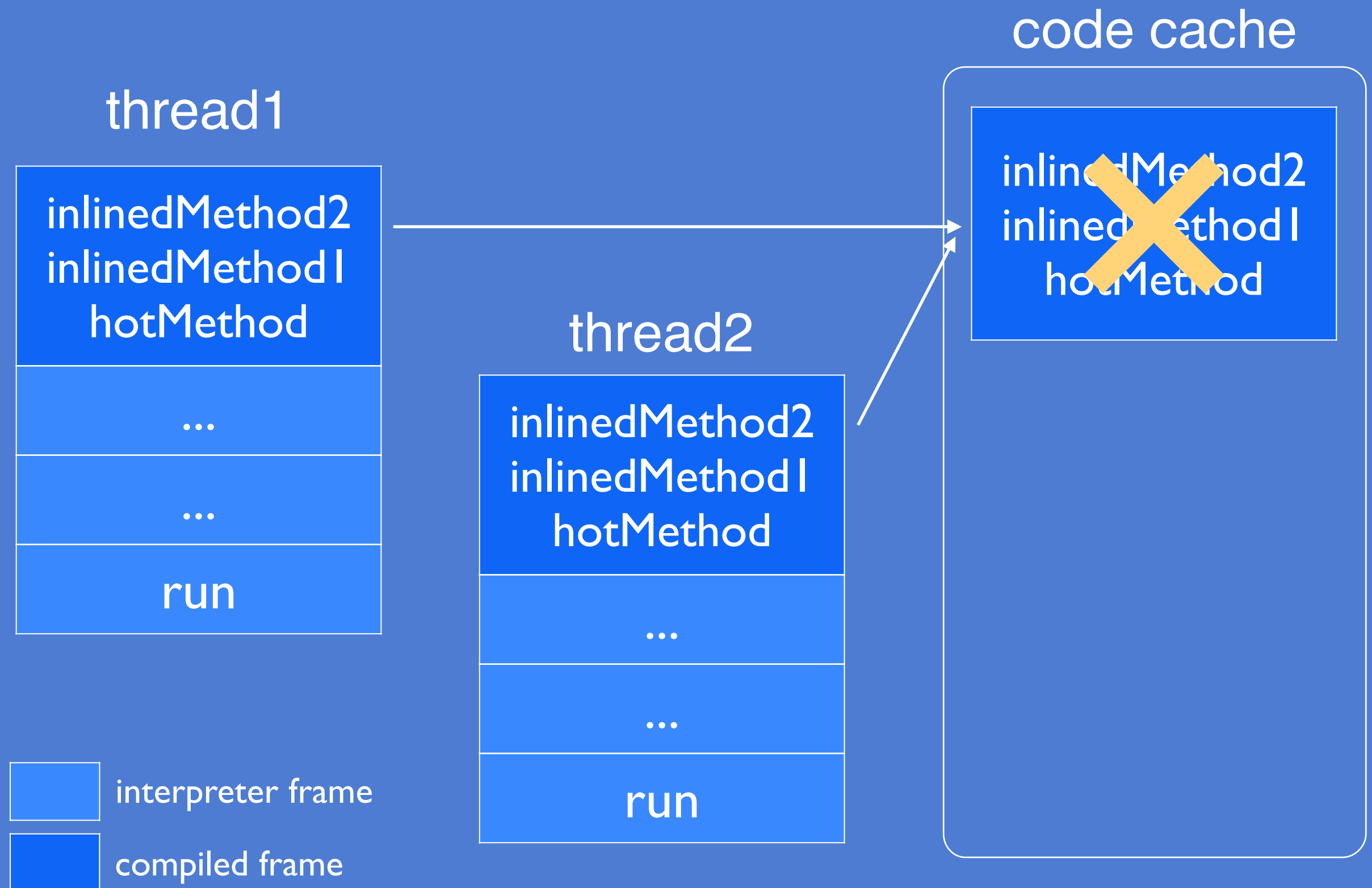
make  
not entrant

reinterpret

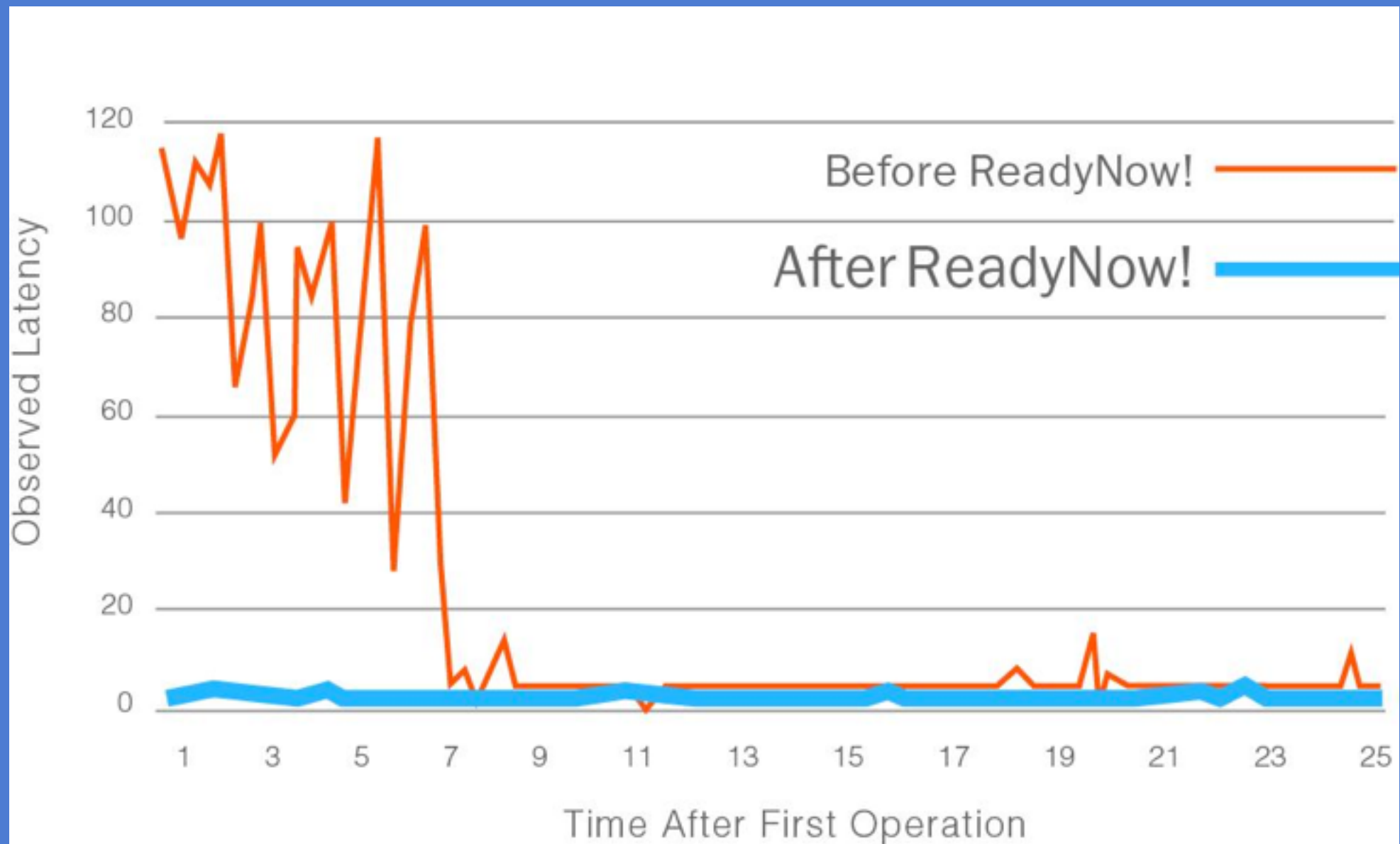
# Reasons & Actions

null_check	make_not_entrant
null_assert	make_not_entrant
range_check	make_not_entrant
class_check	maybe_recompile
array_check	maybe_recompile
intrinsic	maybe_recompile, make_not_entrant
bimorphic	maybe_recompile
unloaded	reinterpret
uninitialized	reinterpret
unreached	reinterpret
unhandled	none
constraint	CHA - deopt now!
div0_check	make_not_entrant
age	maybe_recompile
predicate	none?
loop_limit_check	none?

# Does This Matter?



# ReadyNow!



# Recommending Reading



**Dr Heinz M Kabutz**

<http://www.javaspecialists.eu>

The DeveloperWorks logo, which consists of the word "developerWorks" in a white serif font on an orange rectangular background. A registered trademark symbol (®) is at the end of the word.

developerWorks®

**Brian Goetz**

[http://www.ibm.com/developerworks/views/java/libraryview.jsp?  
contentarea\\_by=Java+technology&search\\_by=brian+goetz](http://www.ibm.com/developerworks/views/java/libraryview.jsp?contentarea_by=Java+technology&search_by=brian+goetz)

The OpenJDK Wiki logo. It features the word "Open" in orange and "JDK" in teal, followed by "Wiki" in a smaller, grey font, all on a white rectangular background.

OpenJDK Wiki

<https://wiki.openjdk.java.net/display/HotSpot>

# VM Developer Blogs

*PSYCHOSOMATIC, LOBOTOMY, SAW*

Nitsan Wakart

<http://psy-lob-saw.blogspot.com/>

ORACLE®

Aleksey Shipilëv

<http://shipilev.net/>

Igor Veresov

<https://twitter.com/maddocig>

# JITWatch

JITWatch - HotSpot Compilation Inspector

Sandbox Open Log Start Stop Config Chart Stats Histo TopList Code Cache TriView Suggest

☒ Hide interfaces ☒ Hide uncompiled classes ☒ Hide non JIT-compiled class members

▼ Packages

- ▼ com
  - ▼ com.chrisnewland
    - ▼ com.chrisnewland.jitwatch
      - ▼ com.chrisnewland.jitwatch.demo
        - MakeHotSpotLog
- ▼ java
- ▼ sun

▼ Methods

- private boolean test(int,int)
- private void testCallChain(long)
- private void testCallChain3()
- public long testCallChainReturn(long)
- private void testLeaf(long)

▼ Fields

Type	Name	Value
Queued	backedge_count	14,563
Queued	bytes	57
Queued	comment	backedge_count
Queued	compile_id	71
Queued	compile_kind	osr

00:00:02.426 Queued : private int com.chrisnewland.jitwatch.demo.MakeHotSpotLog.timesHundred(int)

00:00:02.427 Compiled (C2) : private int com.chrisnewland.jitwatch.demo.MakeHotSpotLog.timesHundred(int)

00:00:02.427 Queued : public int java.lang.String.indexOf(int,int)

00:00:02.429 Compiled (C2) : public int java.lang.String.indexOf(int,int)

00:00:02.430 Compiled (C2) : private void com.chrisnewland.jitwatch.demo.MakeHotSpotLog.testLoopUnrolling(long)

Finished reading log file.

Heap: 53/97M Errors (0)

Mounted class version: 52.0 (Java 8) public BigInteger java.math.BigInteger(String,int) compiled with C1

<https://github.com/AdoptOpenJDK/jitwatch/>



# Questions?

Douglas Q. Hawkins  
VM Engineer

