Java 9 Beyond Contention

Monica Beckwith
Code Karam LLC

About Me

- Masters in Electrical and Computer Engineering
- Java/JVM/GC performance engineer
  - Code Karam LLC
- I have worked with Oracle, Sun, AMD …
  (https://www.linkedin.com/in/monicabeckwith/)
  - JVM heuristics, generation of optimized JIT-ted code, GC performance
  - I used to work in the capacity of Garbage First GC performance lead @Oracle.

Agenda

- Setting the stage
  - An introduction to monitor locking
  - JVM improvements to locking
- Building a problem statement

Agenda

- Performance engineering approaches
  - Choosing the right approach!
    - Top-down or Bottom-up?
- Building our arsenal!

Agenda

- Observing the targeted improvements
  - Demo comparing JDK 9 with JDK 8
- Summarizing the observations

Setting The Stage

Locks

Uncontended Locks

A single thread ‘t’ is executing a synchronized method

Uncontended Locks

- Deflated locks
  - aka light weight locks
- Compare and Swap (CAS) stores pointer to a lock record in the object header.

Contended Locks

A different thread ‘u’ wants to enter the synchronized method that is already locked by thread ‘t’

Contended Locks

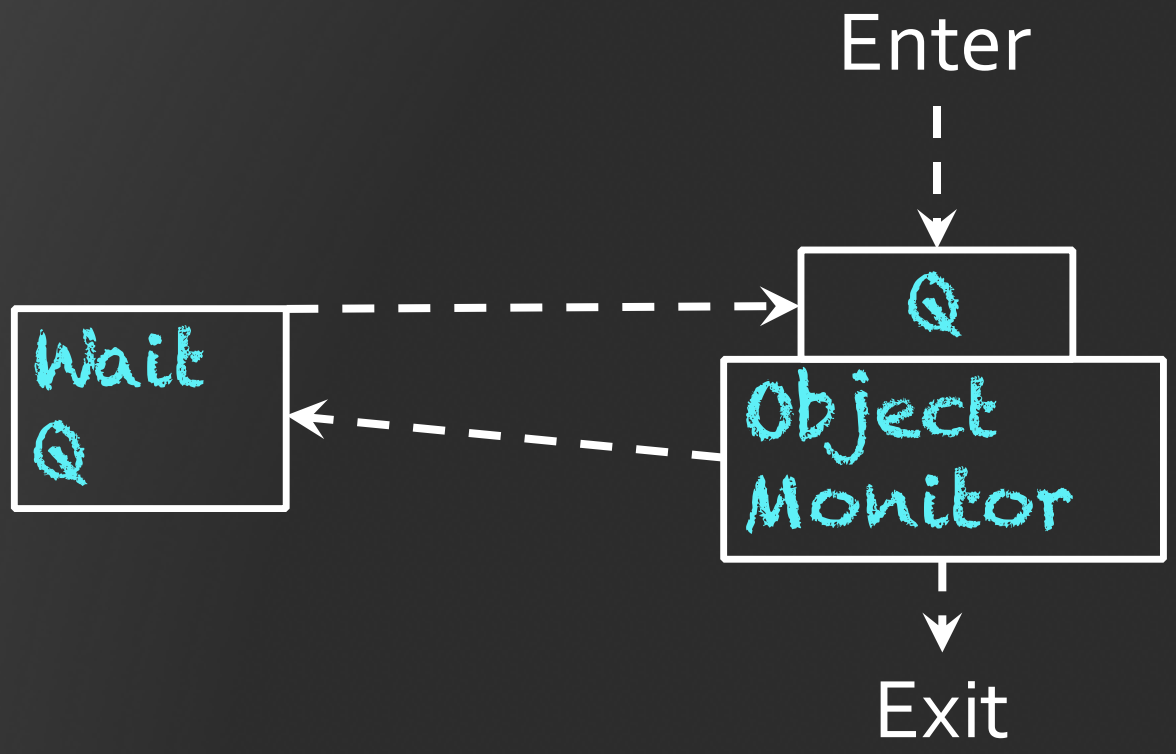
- Inflated locks
  - aka heavy weight locks
- Slower path
- Object monitors maintain their wait-sets

Contended Locks

This is a heavy weight lock also known as inflated lock

Object Monitor maintains “WaitSet” for threads waiting for the contended lock

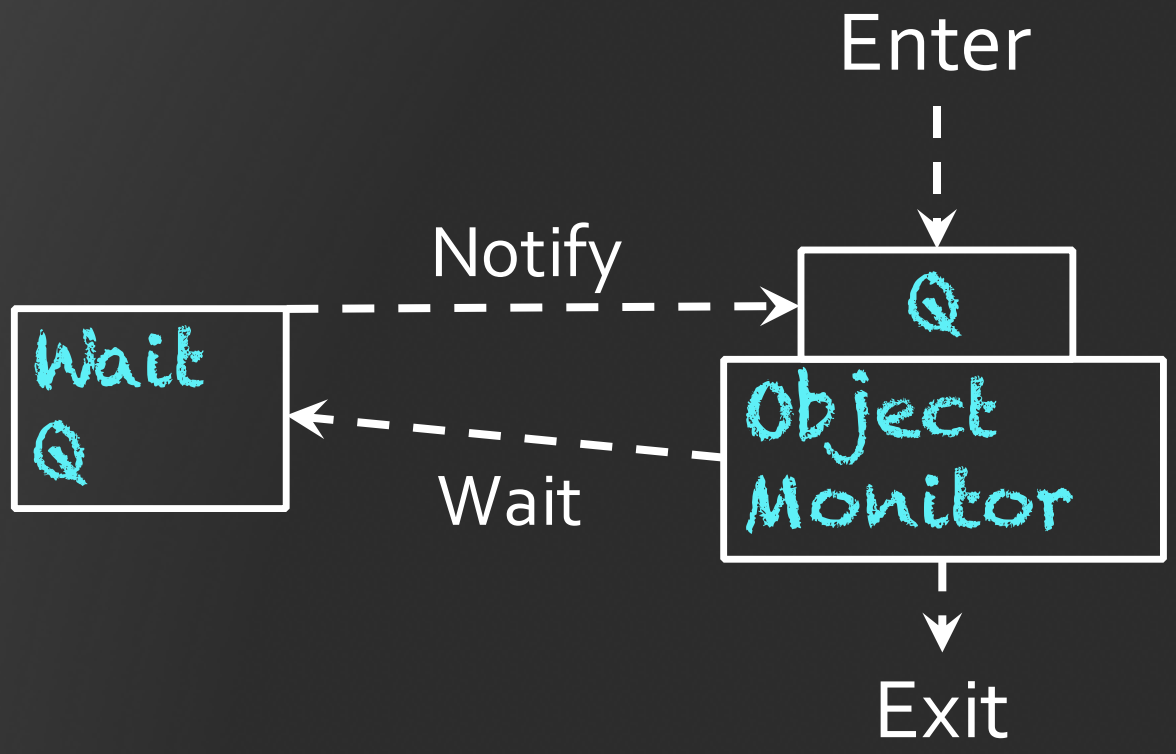
Java Monitors



Wait Set

- A set of threads
- At creation – empty set of threads
- Can be manipulated only by:
  - Object.wait, Object.notify, Object.notifyAll

Java Monitors



Locking Improvements in OpenJDK HotSpot VM

- Biased Locking
- Lock Elision
- Lock Coarsening
- Contended Locking - Quick Path

Escape Analysis

allocated object doesnt escape the compiled method

+

allocated object not passed as a parameter

=

remove allocation and lock (lock elision) and keep field values in registers

Escape Analysis

allocated object doesnt escape the compiled method

+

allocated object is passed as a parameter

=

perform lock elision and use optimized compare instructions

Building a Problem Statement

Contended Locking Improvements - Summary

"Improve the performance of contended Java object monitors"

Contended Locking Improvements - Summary

"Improve the performance of **contended** Java object monitors"

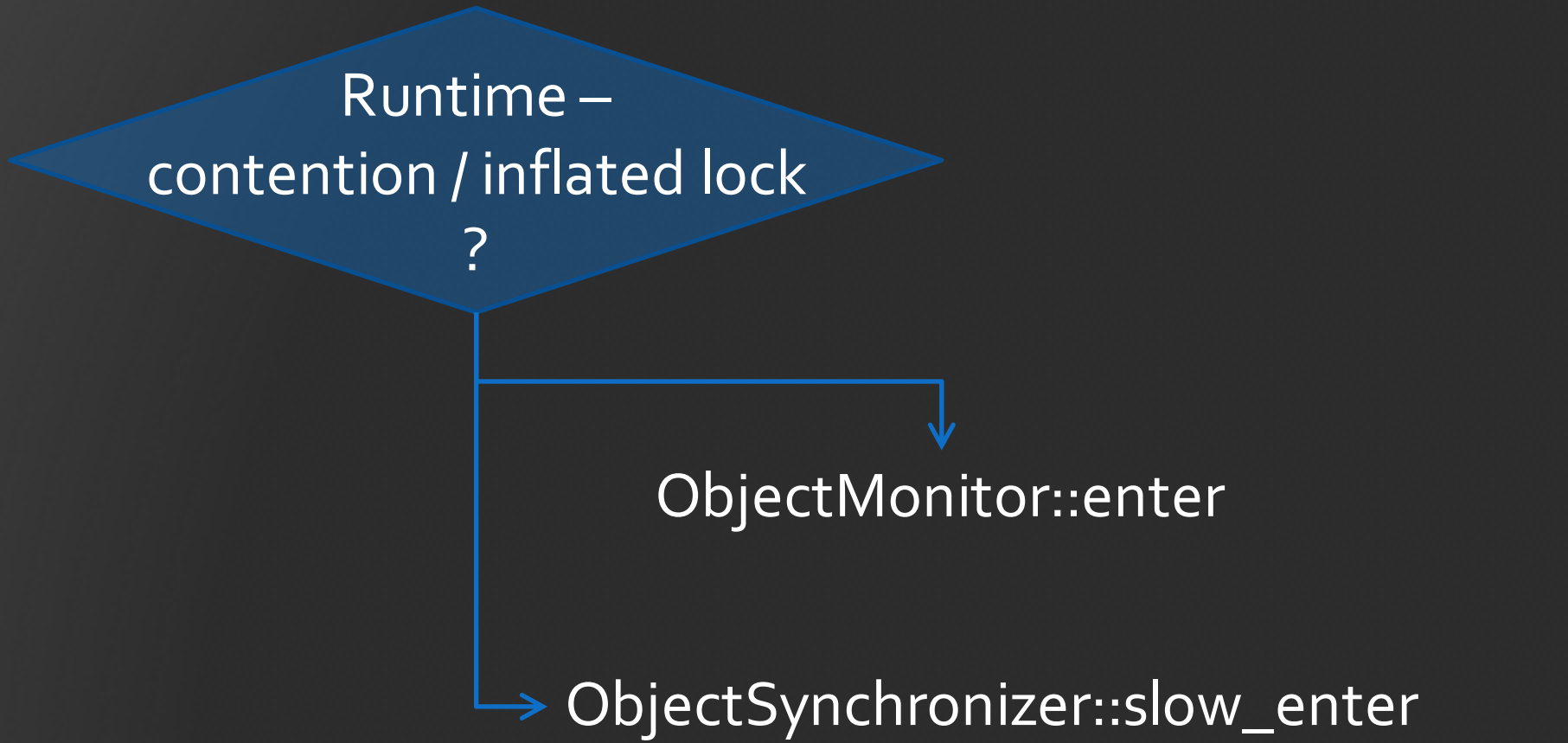
Speed Up Targets

- Java monitor enter operations
- PlatformEvent::unpark()
- Java monitor exit operations
- Java notify/notifyAll operations

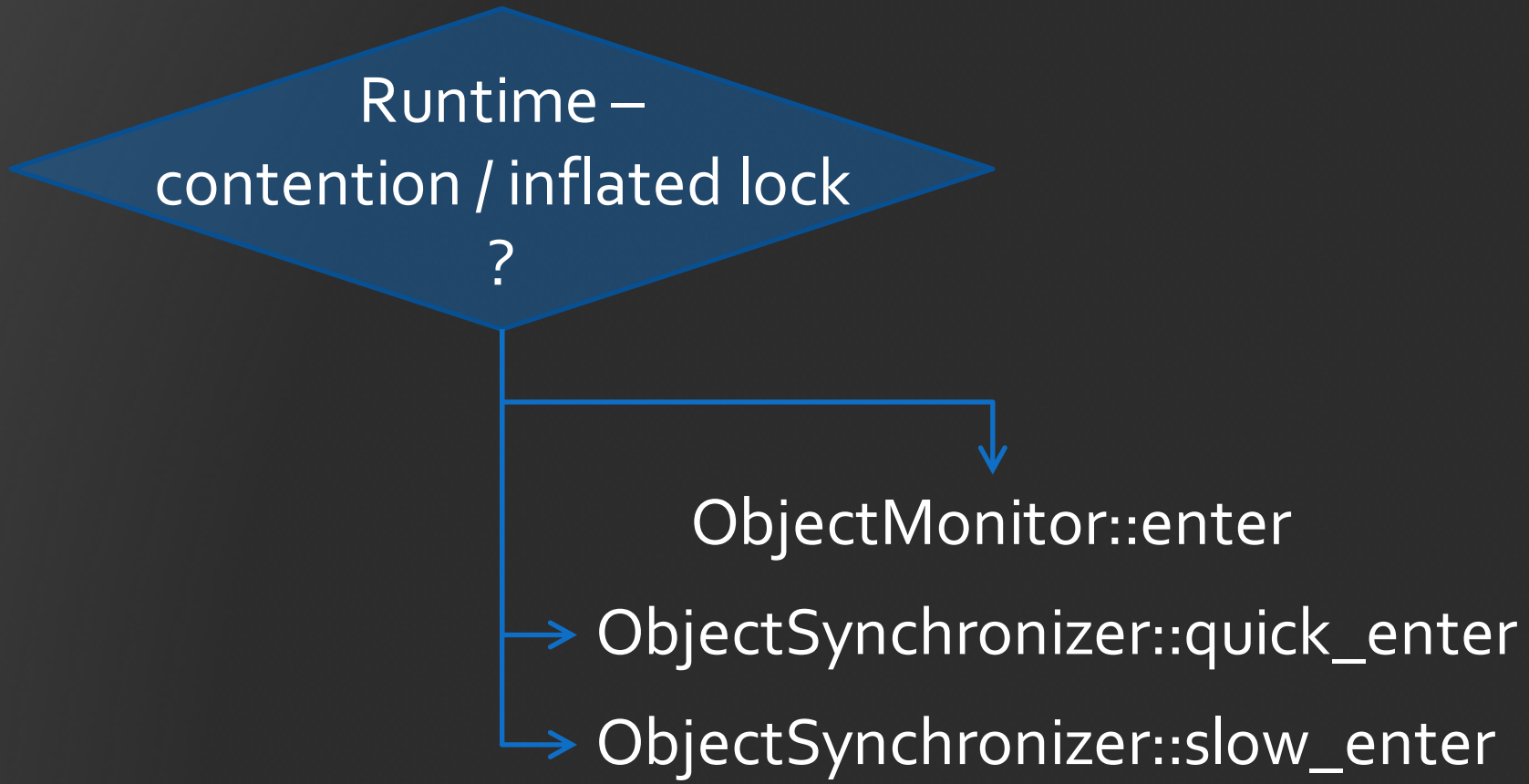
Speed Up Targets

- **Java monitor enter operations**
- …

Java Monitor Enter Operation

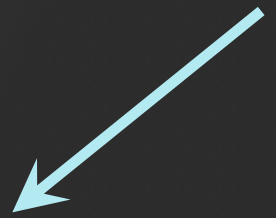


Java Monitor Enter Operation



Performance Engineering Approaches

Performance
Engineering Approaches



Top-down approach

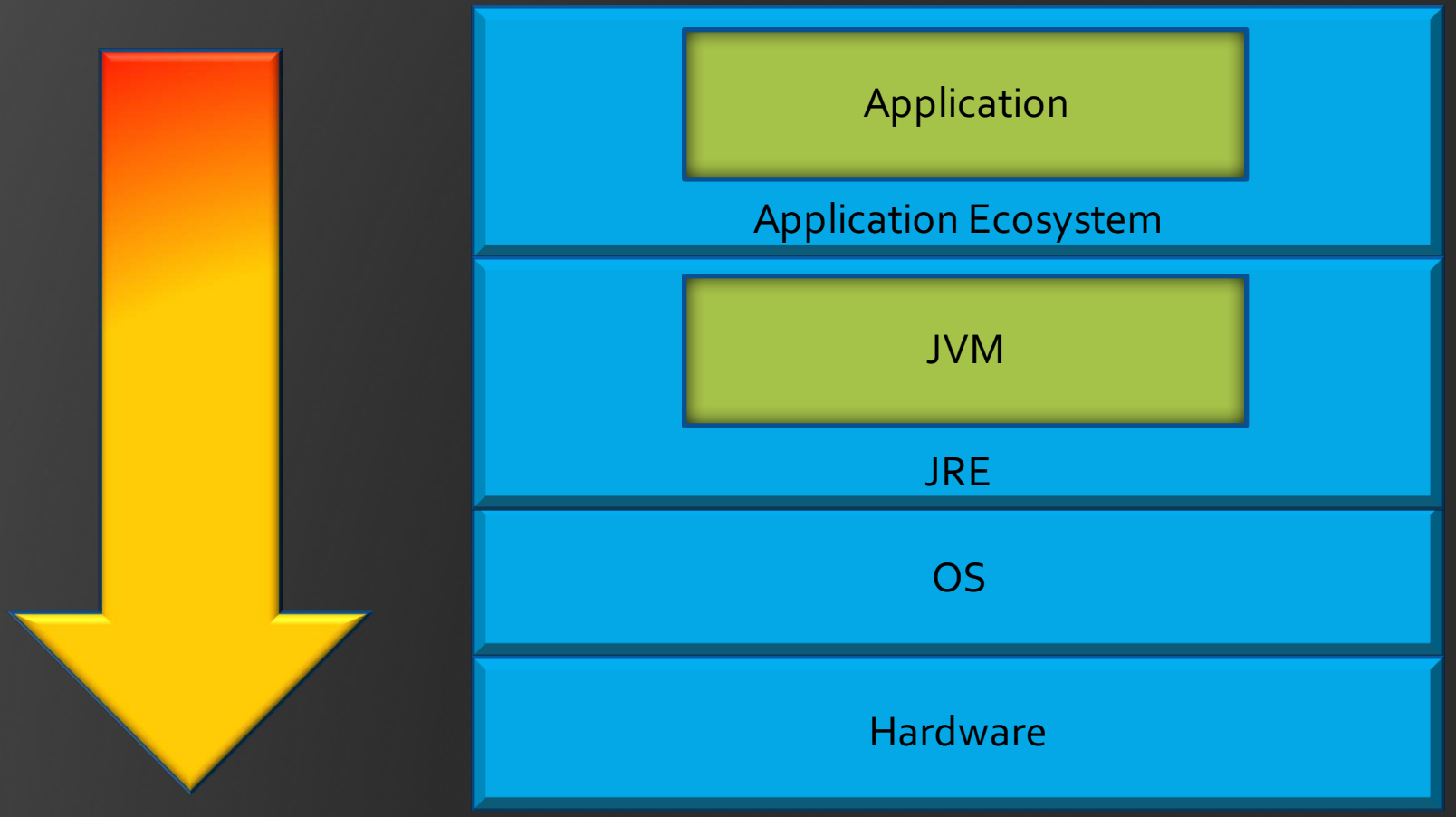
Top-down approach

Top-Down Approach

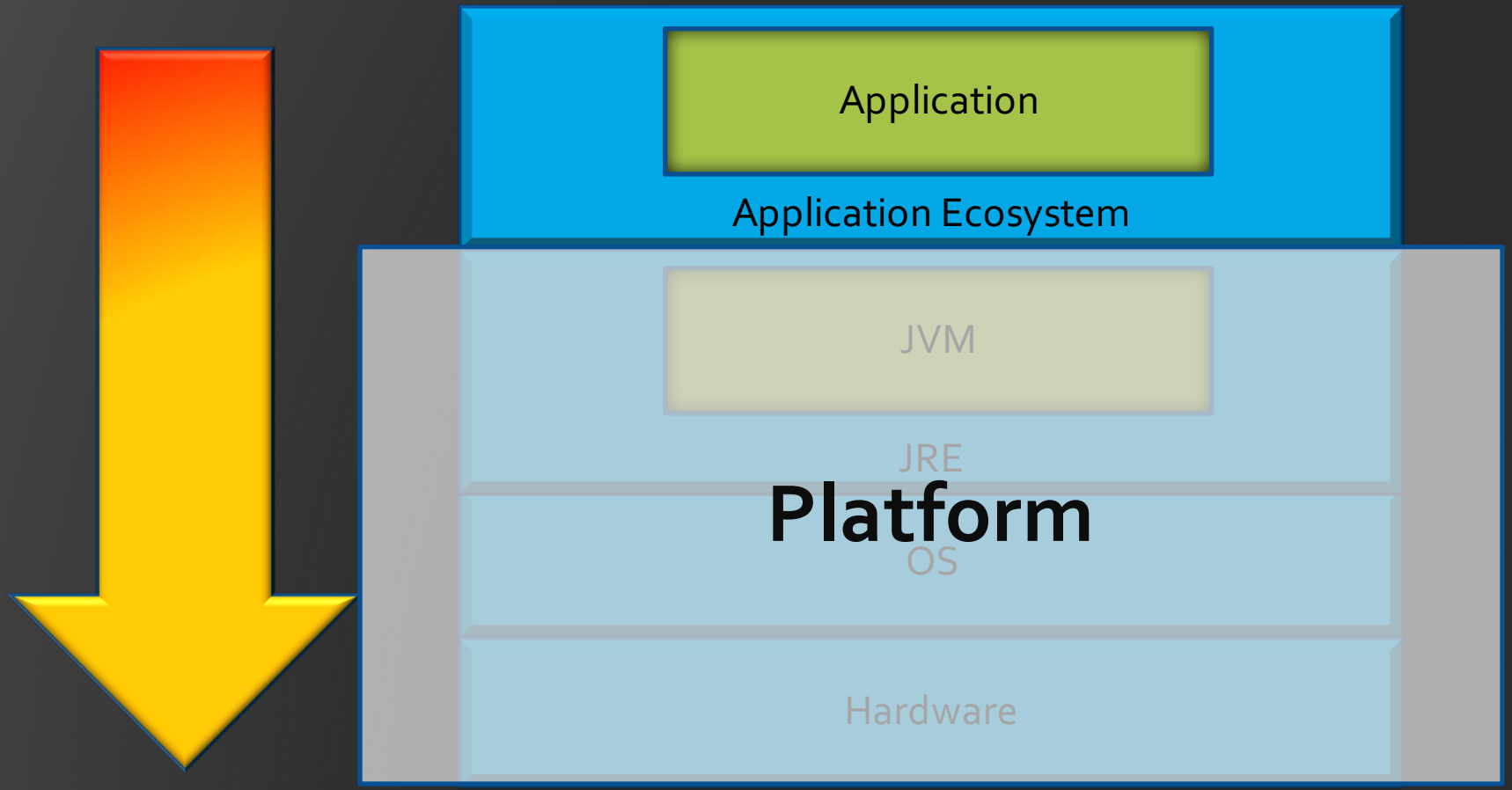
I HAVE the power!!

... to modify the code

Top-Down Approach



Top-Down Approach



When Do You Apply the Top-Down Approach?

When you are trying to improve your application

Bottom Up Approach

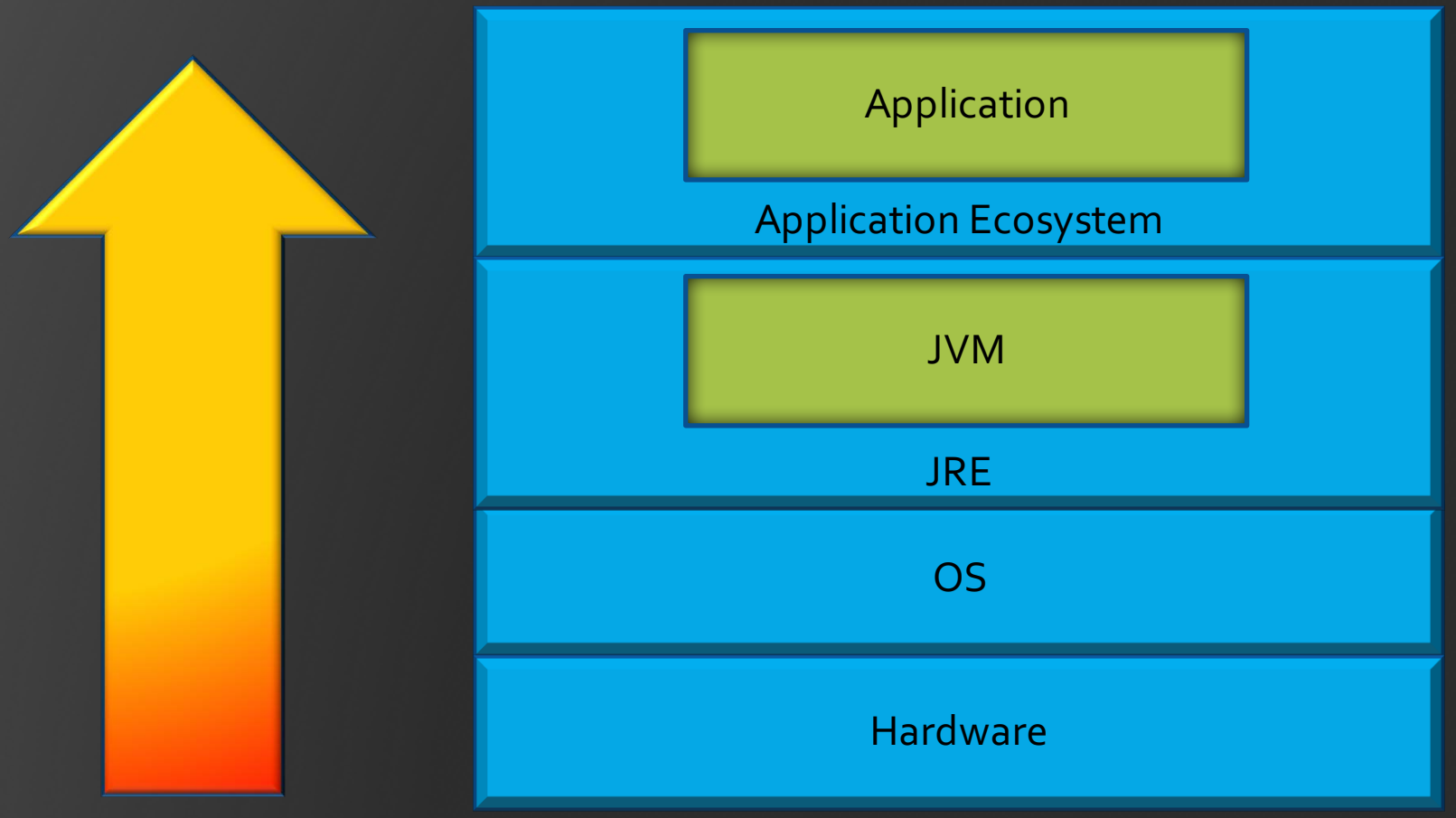
I NEED the power!!

... to stress the platform

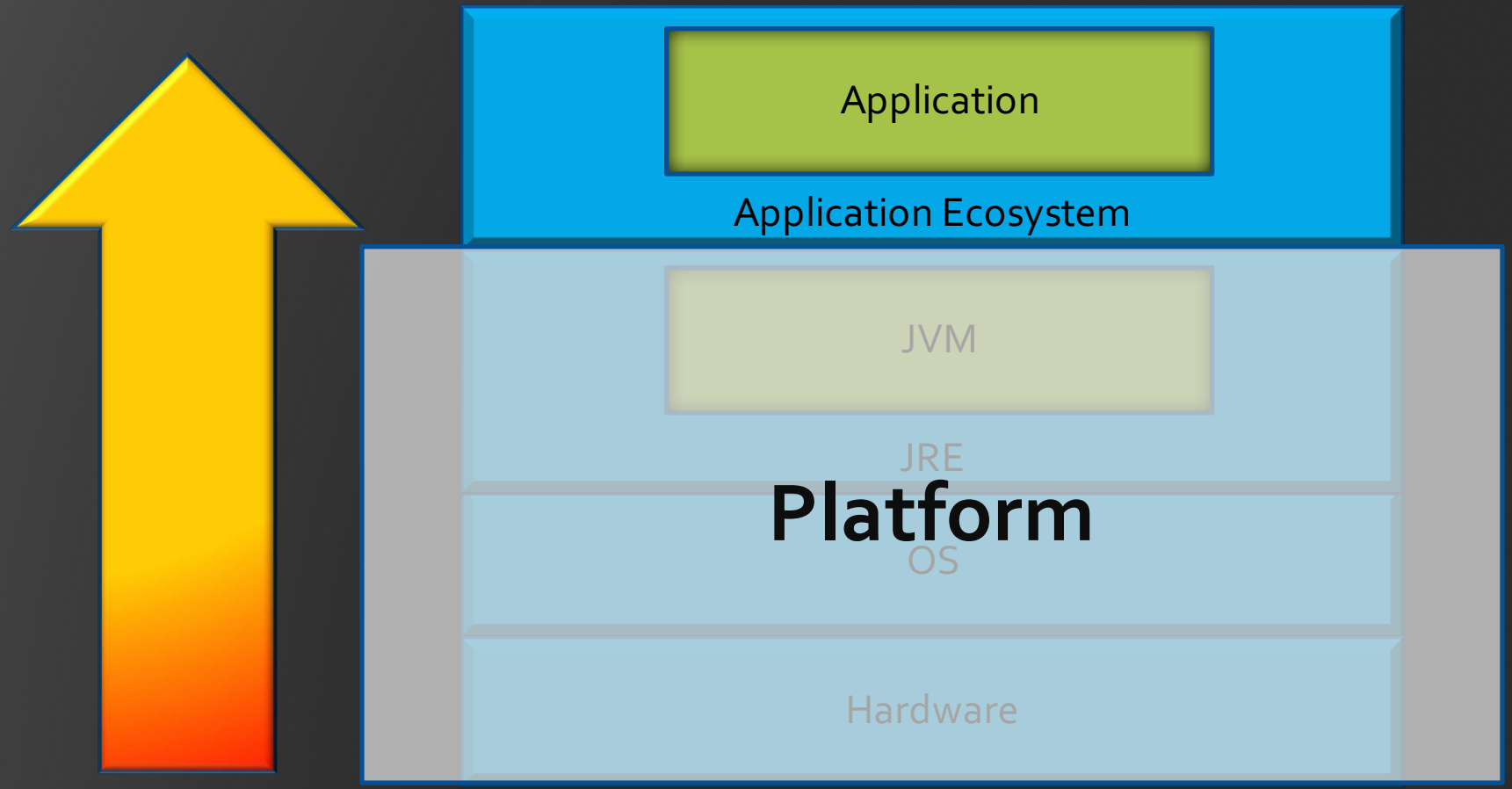
Top Down Approach - Process

- Step 1
- Step 2
- Step 3
- Step 4

Bottom-Up Approach



Bottom-Up Approach



When Do You Apply the Bottom-Up Approach?

When you are trying to improve your platform

Performance Engineering Process

- Step 1
- Step 2
- Step 3
- Step 4

Choosing The Right Approach ...

What Are We Trying To Do?

Improve the JVM

Hence We Choose…

The Bottom-Up Approach

Building Our Arsenal

Where Do We Start?

- Know what you are stressing
- Get/ write the appropriate workload/ application
- Get/write the appropriate tools

Know What You're Stressing...

Remember Our Speed Up Targets?

- Java monitor enter operations
- PlatformEvent::unpark()
- Java monitor exit operations
- Java notify/notifyAll operations

Get/Write The Appropriate Workload/Application ...

Benchmarking

LockLoops-JSR166-Doug-Sept2009 (was LockLoops):

The benchmark compares multiple locking techniques

For our purpose, we just need to test the contended locks.

Contended Lock Benchmarking

```
private static class BuiltinLockLoop extends LockLoop {
        final int loop(int n) {
            int sum = 0;
            while (n-- > 0) {
                synchronized (this) {
                    v = LoopHelpers.compute1(v);
                }
                sum += LoopHelpers.compute2(v);
            }
            return sum;
        }
    }
```

Contended Lock Benchmarking

```
private static class BuiltinLockLoop extends LockLoop {
        final int loop(int n) {
            int sum = 0;
            while (n-- > 0) {
                    synchronized (this) {
                        v = LoopHelpers.compute1(v);
                    }
                    sum += LoopHelpers.compute2(v);
            }
            return sum;
        }
    }
```

Where to Next?

Ensure that you are in fact measuring contended object monitor performance!

How Do We Do That?

Bypass biased locking: Use -XX:-UseBiasedLocking

Bypass stack based locking: Use -XX:+UseHeavyMonitors

Get/Write The Appropriate Tools ...

Oracle Developer Studio Performance Tool

Profiling with 'collect'

- -j on: default for when target is Java
- -p on: default clock-profiling rate of ~100 samples/second
- -H on: heap tracing
- -t <duration>: time over which to record data
- -h <ctr_def>...: specify HW counter profiling

Observing The Targeted Improvements

Speed Up Targets

- Java monitor enter operations
- PlatformEvent::unpark()
- Java monitor exit operations
- Java notify/notifyAll operations

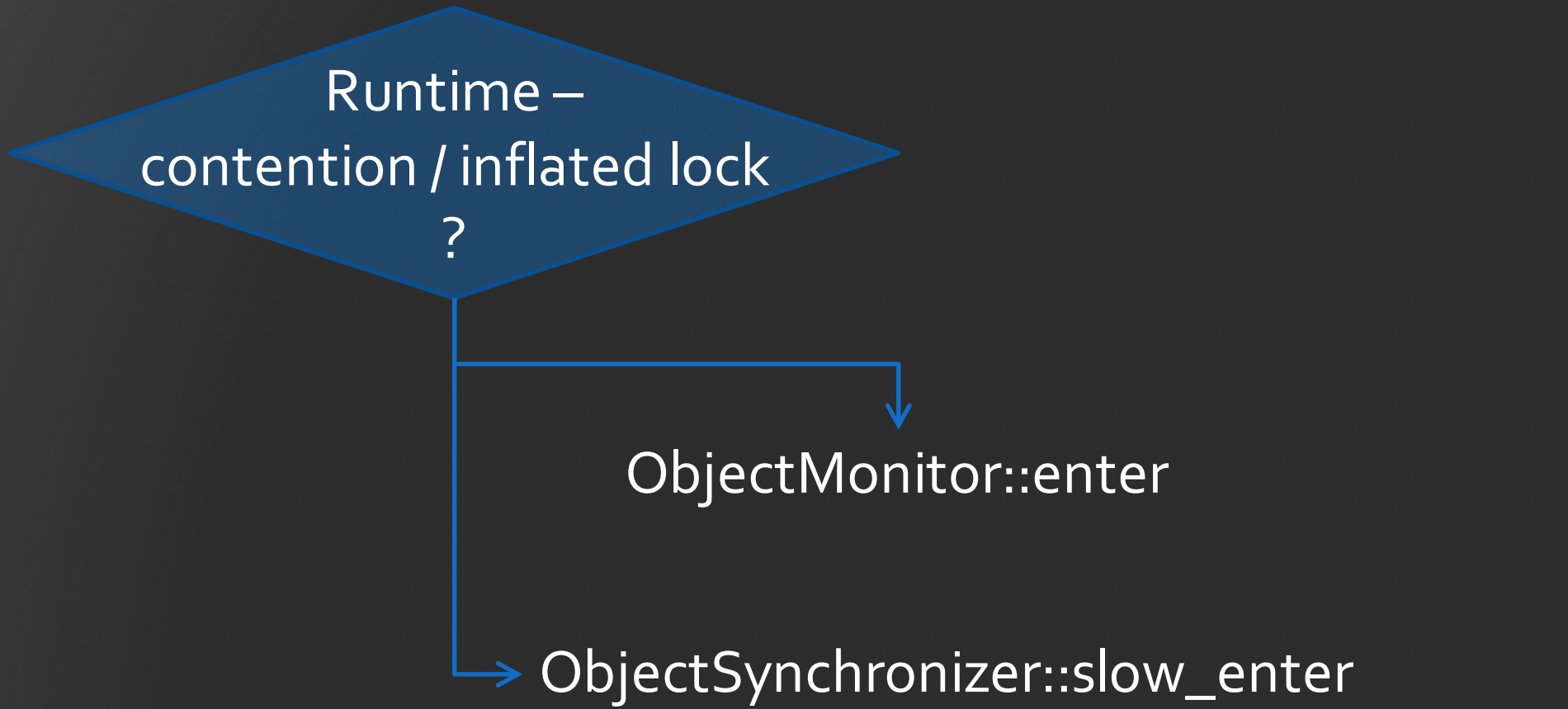
Demo - Comparing Contended Locking in JDK 9 to JDK 8

Summarizing the Observations

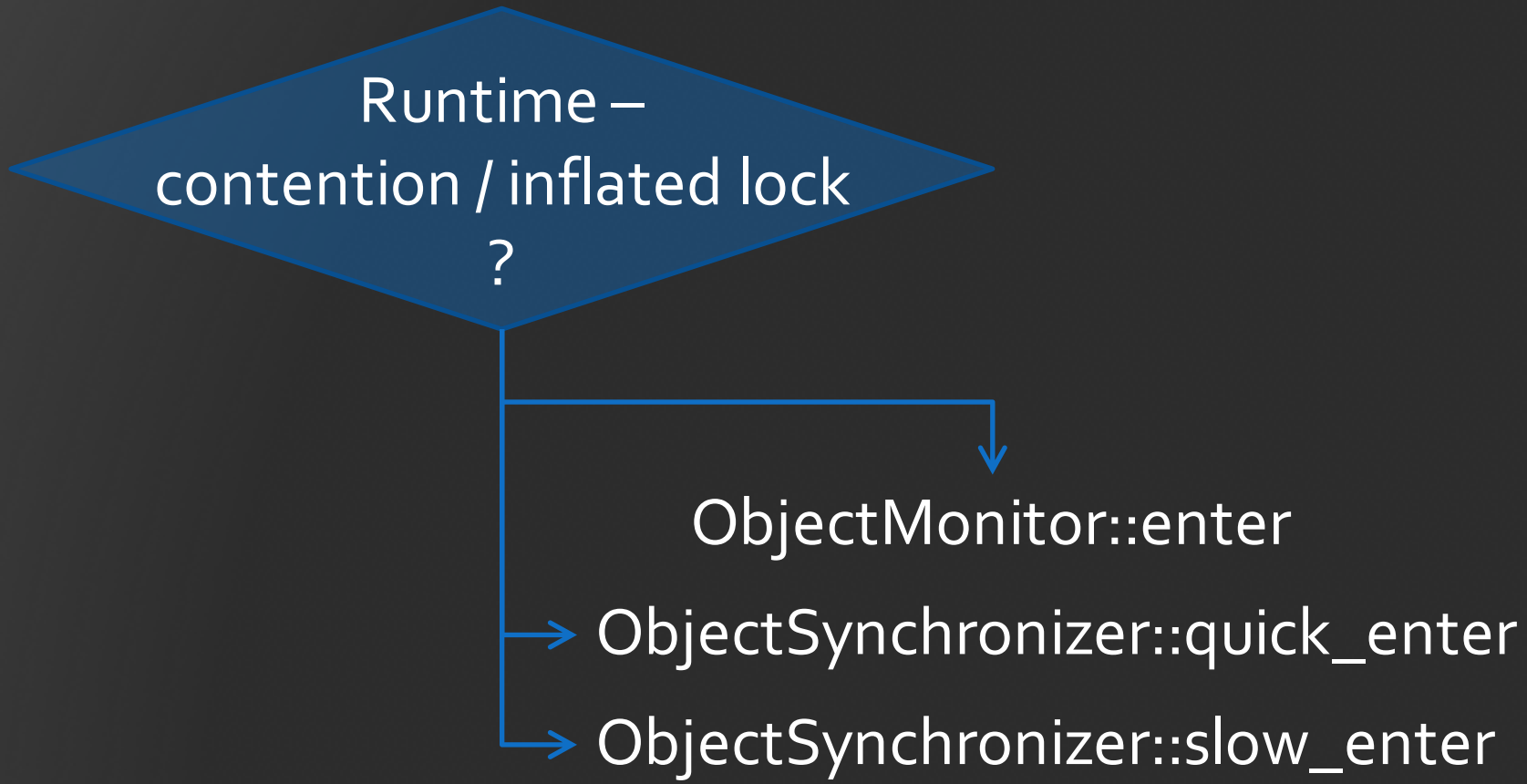
Speed Up Targets

- **Java monitor enter operations**
- …

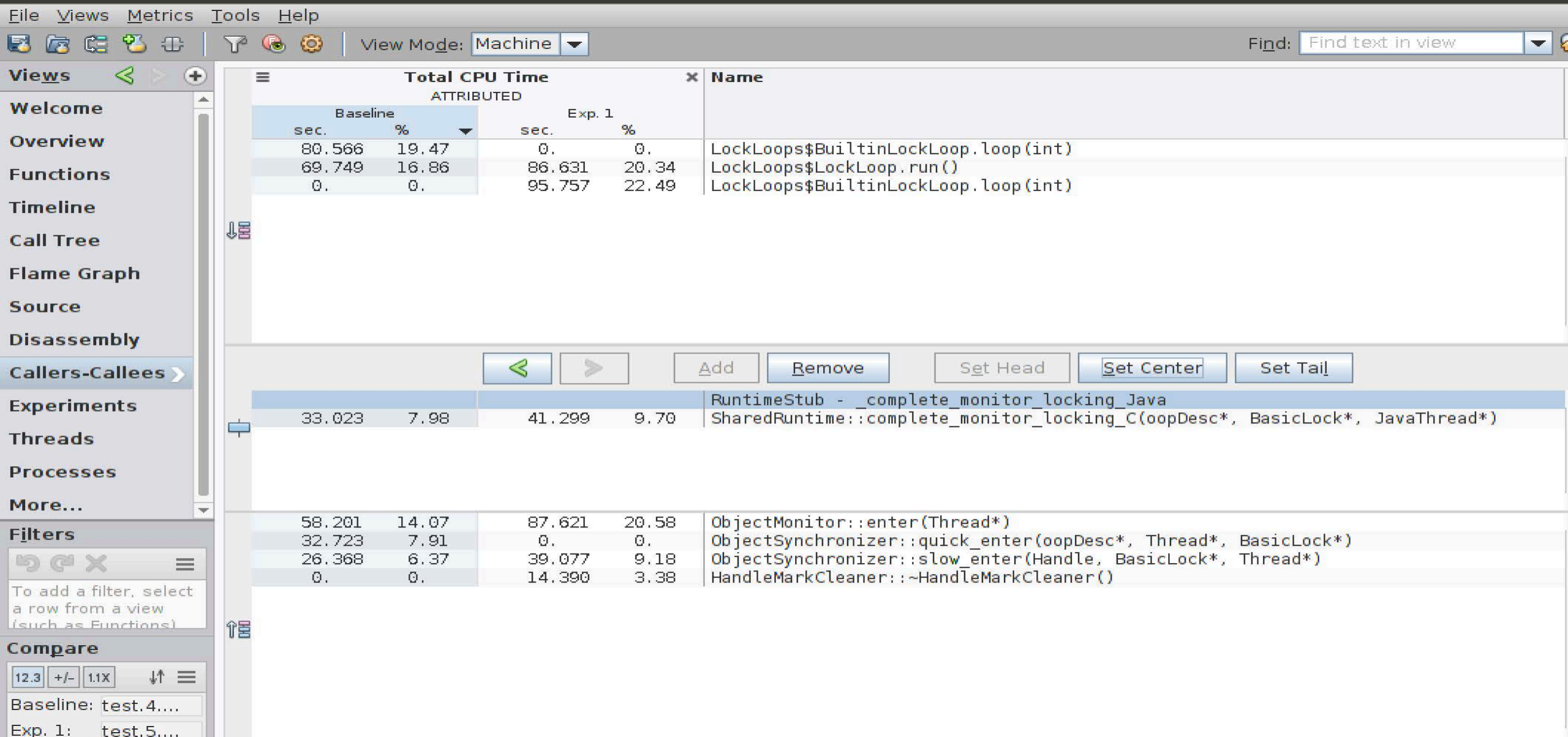
Java Monitor Enter Operation



Java Monitor Enter Operation



Java Monitor Enter Operation



Monitor Enter

JDK 9 Exclusive Time	JDK8 Exclusive Time	Method Name
26.038	50.415	ObjectMonitor::enter(Thread*)

JDK 9 Inclusive Time	JDK8 Inclusive Time	Method Name
58.201	87.621	ObjectMonitor::enter(Thread*)

Quick Enter

JDK 9 Exclusive Time	JDK8 Exclusive Time	Method Name
32.723	0	ObjectSynchronizer::quick_enter

JDK 9 Inclusive Time	JDK8 Inclusive Time	Method Name
32.723	0	ObjectSynchronizer::quick_enter

Hs-9/src/share/vm/runtime/synchronizer.cpp
Hs-9/src/share/vm/runtime/sharedRuntime.cpp

Slow Enter

JDK 9 Exclusive Time	JDK8 Exclusive Time	Method Name
10.317	20.604	ObjectSynchronizer::slow_enter

JDK 9 Inclusive Time	JDK8 Inclusive Time	Method Name
26.368	39.087	ObjectSynchronizer::slow_enter

Speed Up Targets

- **PlatformEvent::unpark()**
- …

Unpark

JDK 9 Exclusive Time	JDK8 Exclusive Time	Method Name
0.060	0.150	os::PlatformEvent::unpark()

JDK 9 Inclusive Time	JDK8 Inclusive Time	Method Name
2.942	3.973	os::PlatformEvent::unpark()

Speed Up Targets

- **Java monitor exit operations**
- …

Monitor Exit

JDK 9 Exclusive Time	JDK8 Exclusive Time	Method Name
0.470	0.340	ObjectMonitor::exit(bool, Thread*)
0.040	0.210	ObjectMonitor::ExitEpilog(Thread*,...)

JDK 9 Inclusive Time	JDK8 Inclusive Time	Method Name
2.752	0.400	ObjectMonitor::exit(bool, Thread*)
0.721	4.263	ObjectMonitor::ExitEpilog(Thread*,...)

Further Reading

http://openjdk.java.net/jeps/143
https://docs.oracle.com/javase/specs/jls/se7/html/jls-17.html
https://wiki.openjdk.java.net/display/HotSpot/Synchronization
https://blogs.oracle.com/dave/entry/biased_locking_in_hotspot
JITWatch: https://github.com/AdoptOpenJDK/jitwatch/wiki/Instructions

Java Performance Book by Charlie Hunt and Binu John.

http://mail.openjdk.java.net/mailman/listinfo
- hotspot-dev@openjdk.java.net
- hotspot-gc-dev@openjdk.java.net