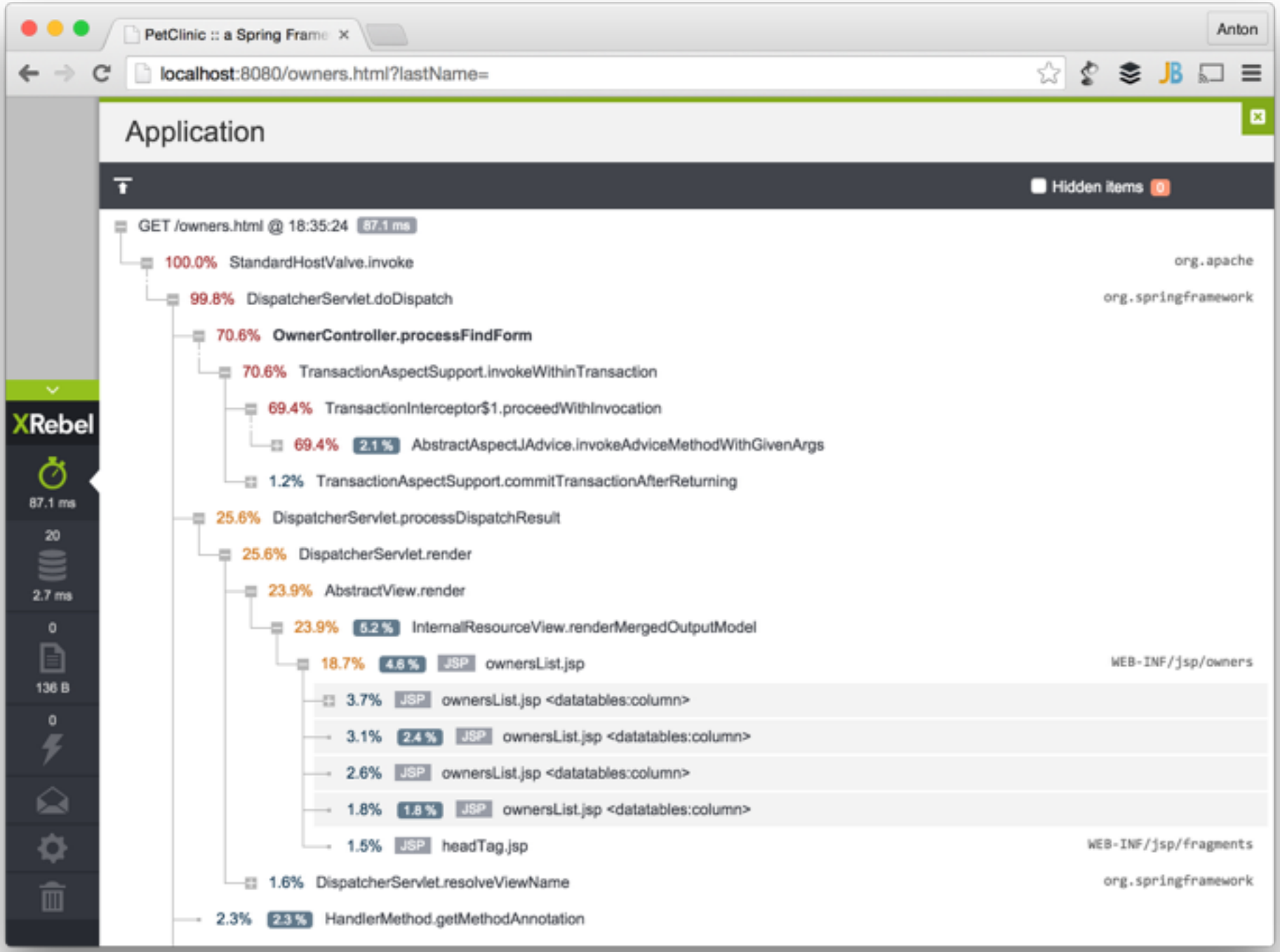
Is your profiler speaking the same language as you?

Simon Maple
@sjmaple

Simon Maple - @sjmaple

Agenda

- Performance Tools
- Performance by numbers
- Sampling vs Tracing
- XRebel
- JRebel


PetClinic :: a Spring Frame
Anton
localhost:8080/owners.html?lastName=
Application
Hidden items 0
GET /owners.html @ 18:35:24 87.1 ms
100.0% StandardHostValve.invoke org.apache
99.8% DispatcherServlet.doDispatch org.springframework
70.6% OwnerController.processFindForm
70.6% TransactionAspectSupport.invokeWithinTransaction
69.4% TransactionInterceptor$1.proceedWithInvocation
69.4% 2.1 % AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs
1.2% TransactionAspectSupport.commitTransactionAfterReturning
25.6% DispatcherServlet.processDispatchResult
25.6% DispatcherServlet.render
23.9% AbstractView.render
23.9% 5.2 % InternalResourceView.renderMergedOutputModel
18.7% 4.6 % JSP ownersList.jsp WEB-INF/jsp/owners
3.7% JSP ownersList.jsp <datatables:column>
3.1% 2.4 % JSP ownersList.jsp <datatables:column>
2.6% JSP ownersList.jsp <datatables:column>
1.8% 1.8 % JSP ownersList.jsp <datatables:column>
1.5% JSP headTag.jsp WEB-INF/jsp/fragments
1.6% DispatcherServlet.resolveViewName org.springframework
2.3% 2.3 % HandlerMethod.getMethodAnnotation
XRebel
87.1 ms
20
2.7 ms
0
136 B
0

Performance Tools

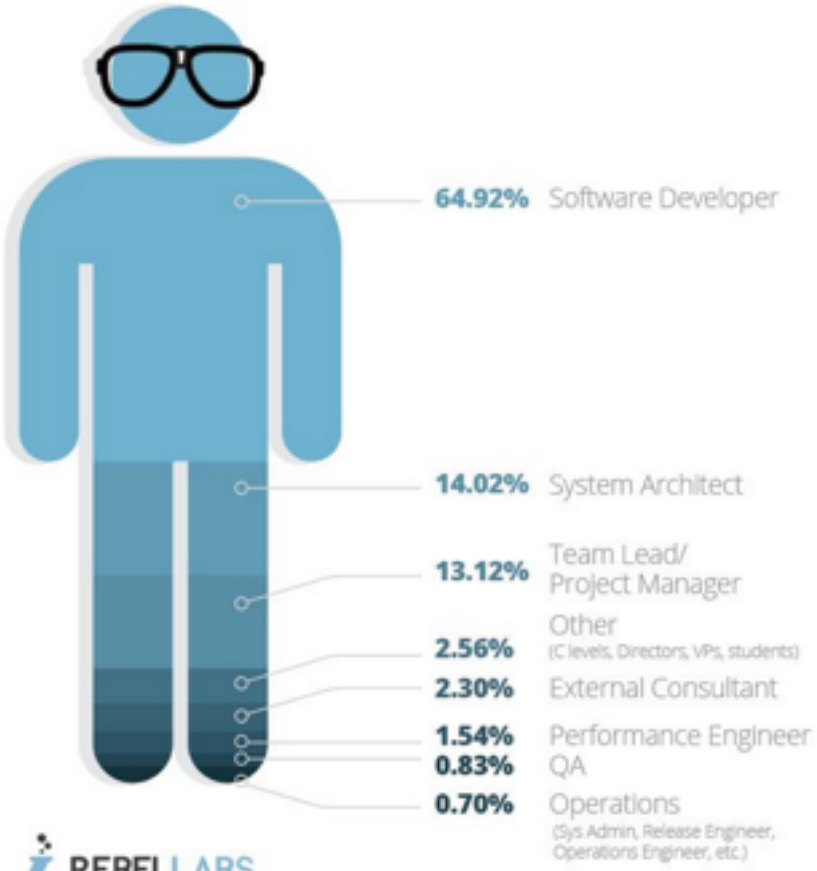
- Java Monitoring Tools
- Java Profilers
- Java Testing Tools

REBELLABS
BY ZEROTURNAROUND

Performance report

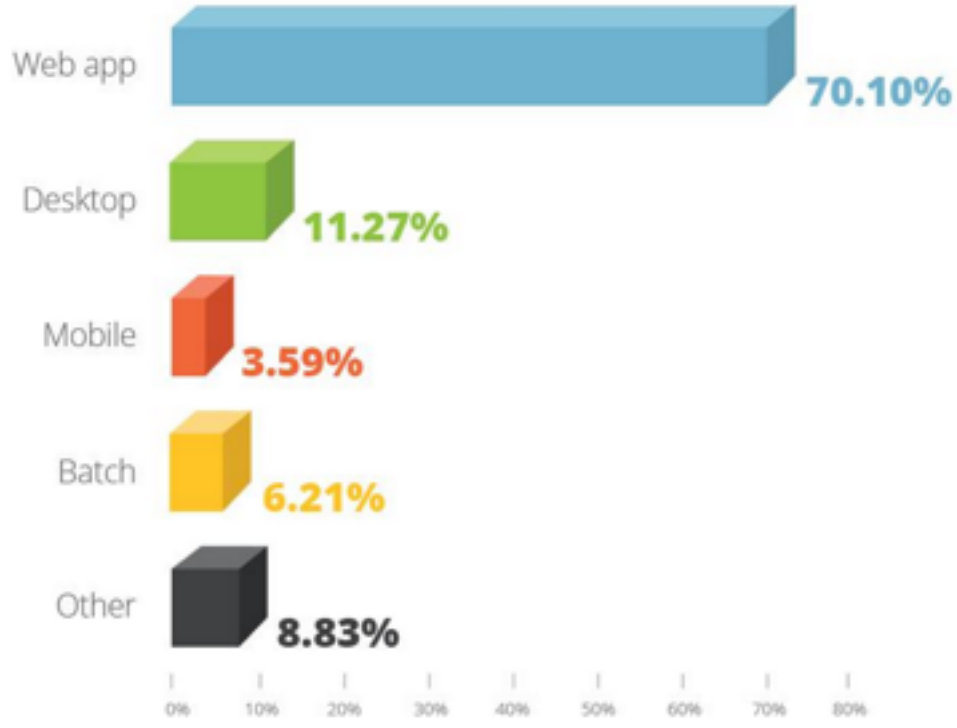

REBELLABS
by ZEROTURNAROUND
RebelLabs' Java Performance Survey 2015
Hey, you made it! We're really happy you can spend ~3 minutes (verified by devs) to tell us about your profiler and performance tools and technologies you use, enjoy and get results from.
For each completed survey, we're donating 50 cents to a charity that provide support dogs for children with disabilities! See? Now you HAVE TO complete it and share with friends! ;-)
Get started!
press ENTER

What is your job title/role/style/category?



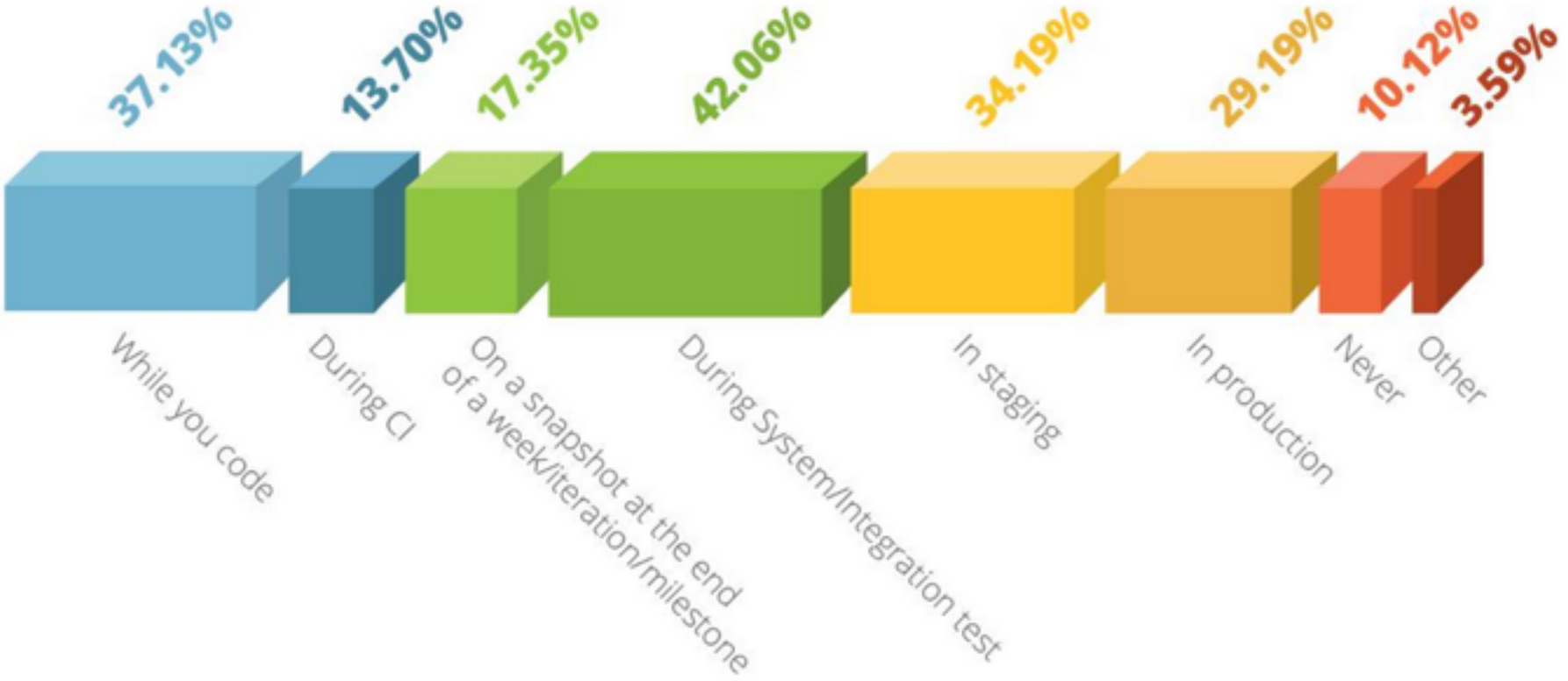
REBELLABS by ZEROTURNAROUND

Which best matches the application you work on?



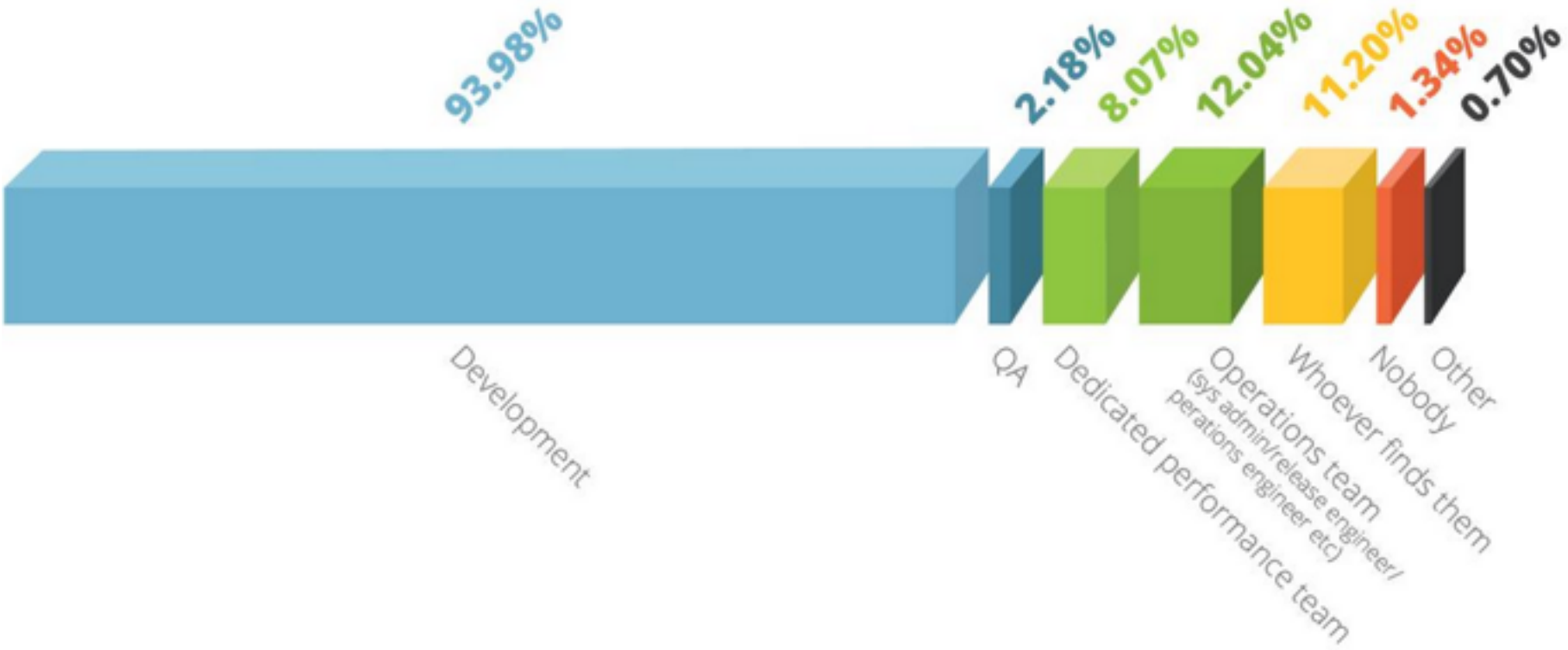
REBELLABS by ZEROTURNAROUND

At what stage do you perform profiling and performance tuning on your applications?



REBEL LABS by ZEROTURNAROUND

Who fixes the performance problems?



REBELLABS by ZEROTURNAROUND

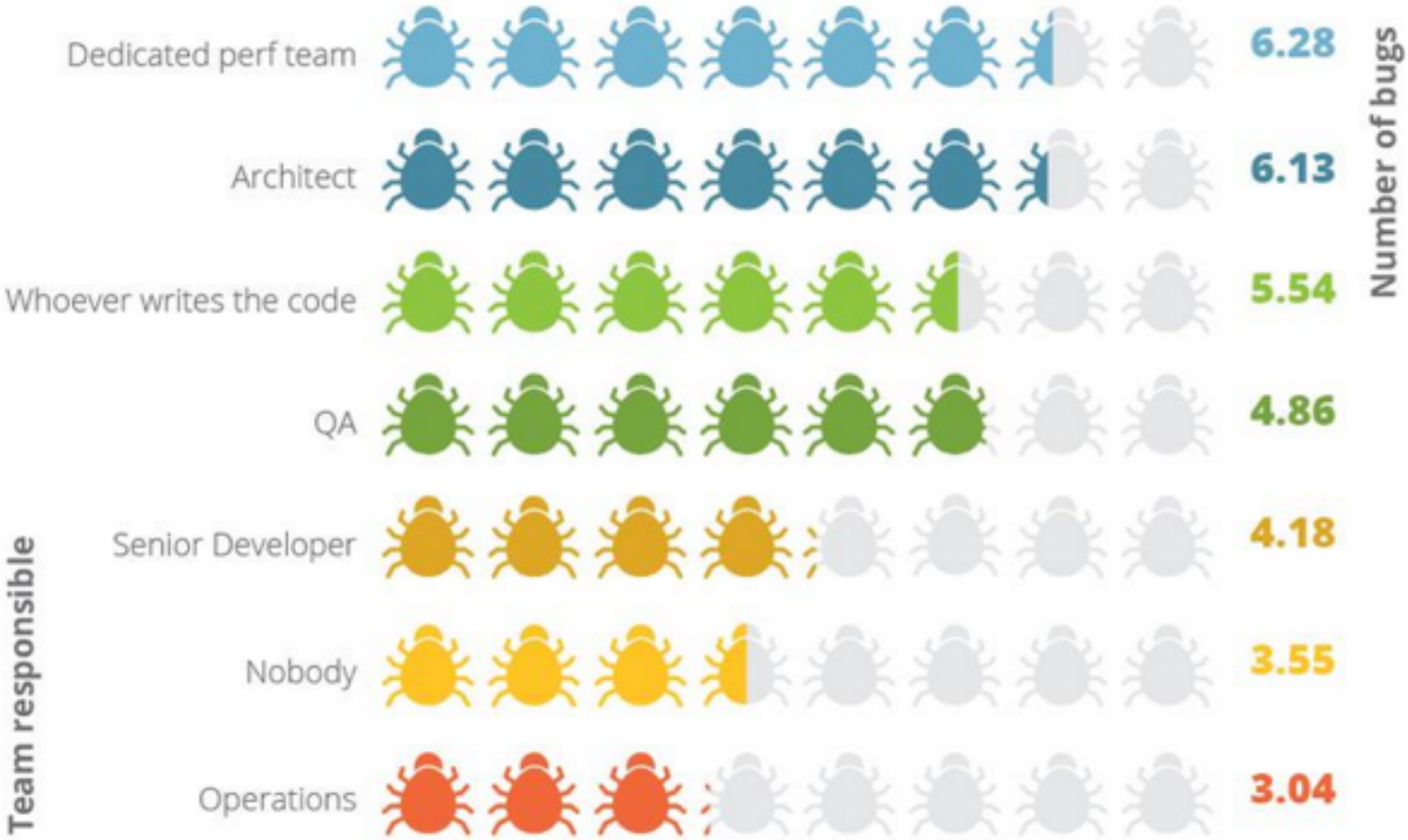
Are people who monitor and find the issues
the same people who fix them?

REBEL LABS by ZEROTURNAROUND

Do projects with specific perormance engineering teams
let fewer bugs to impact the user?

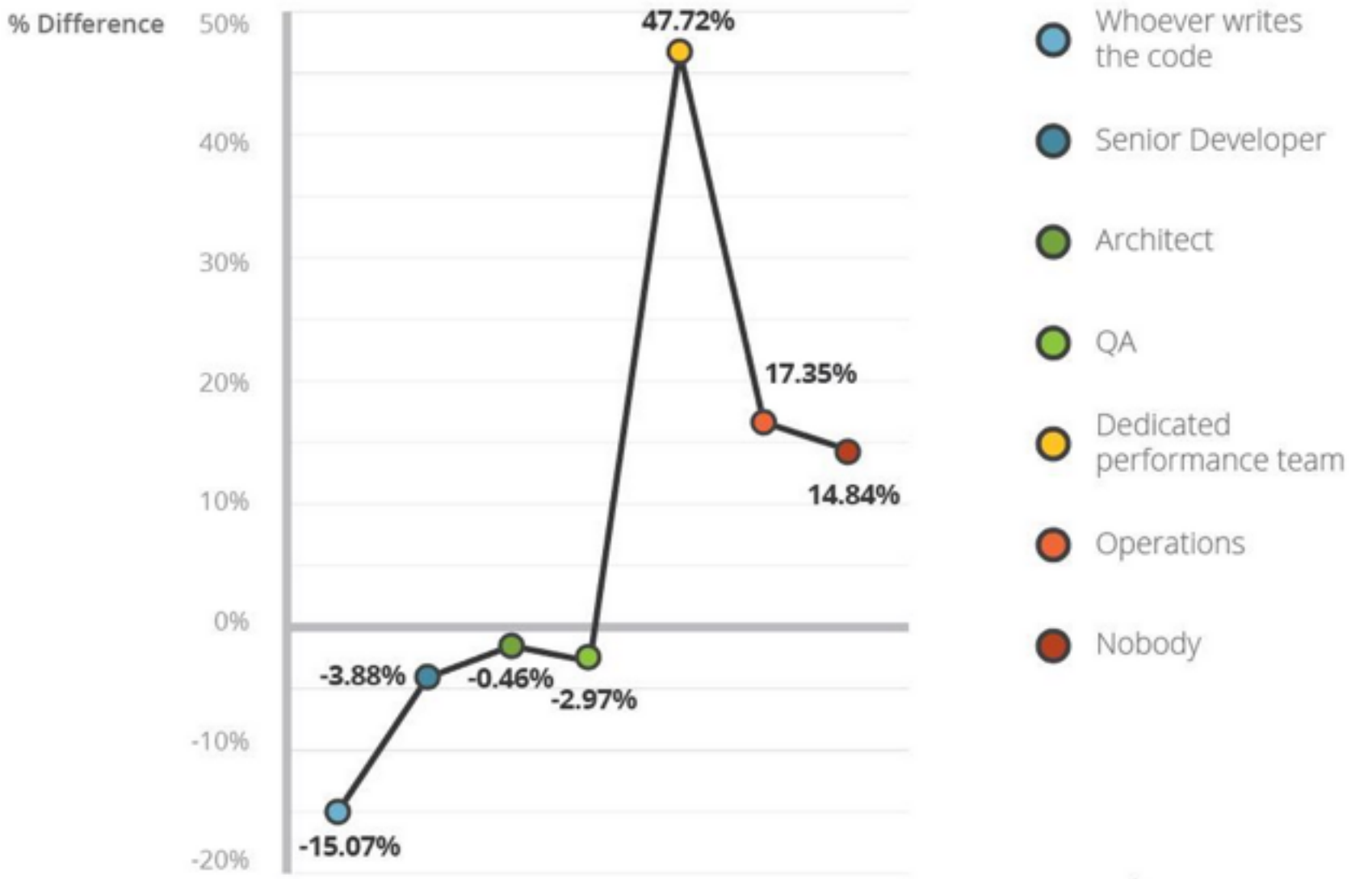
REBEL LABS by ZEROTURNAROUND

How many performance related bugs do you find
in a typical application release?



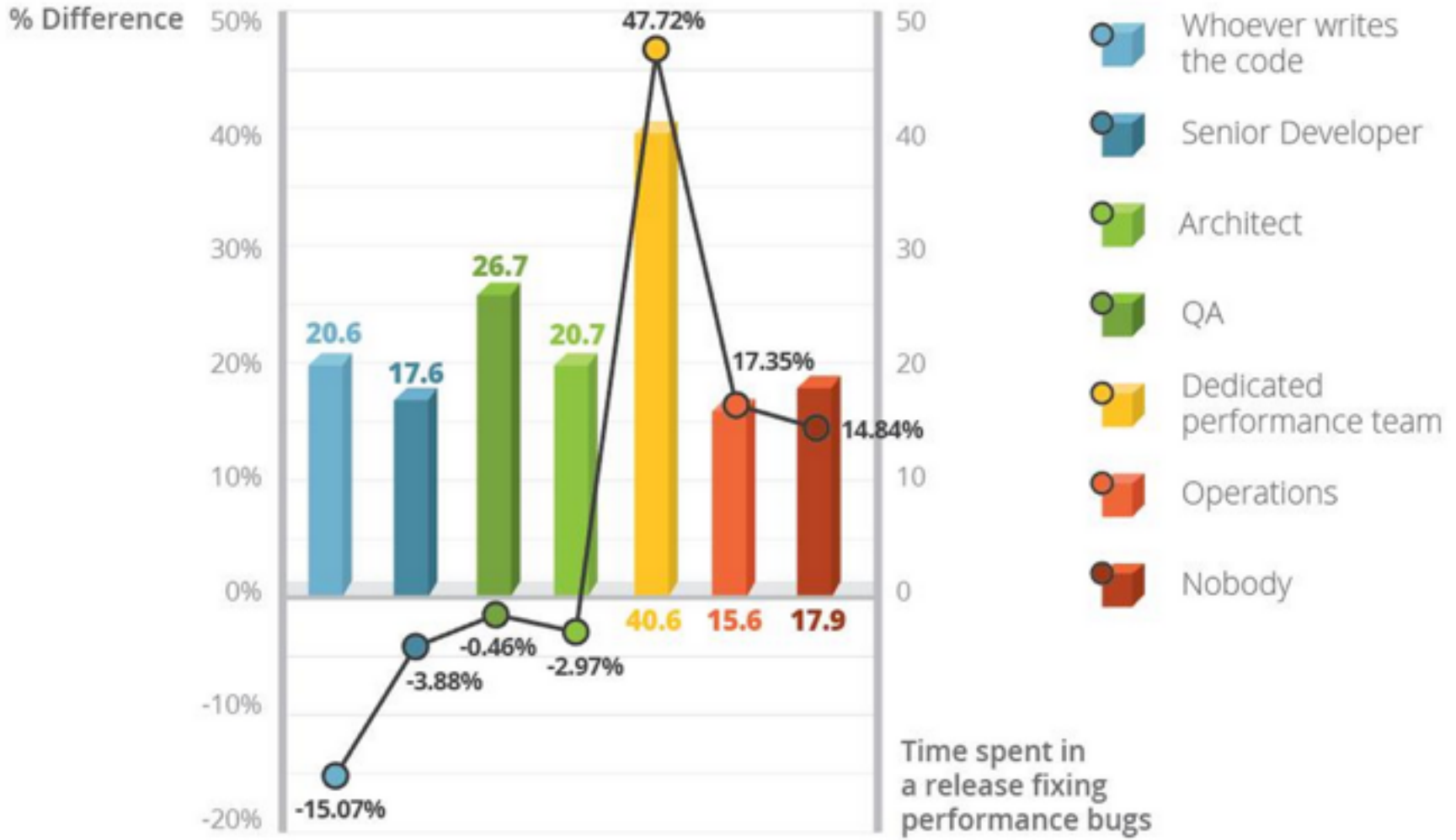
REBEL LABS by ZEROTURNAROUND

When you find an issue, how long does it take,
in days, to diagnose, fix and test on average?



REBEL LABS by ZERO TURNAROUND

When you find an issue, how long does it take,
in days, to diagnose, fix and test on average?



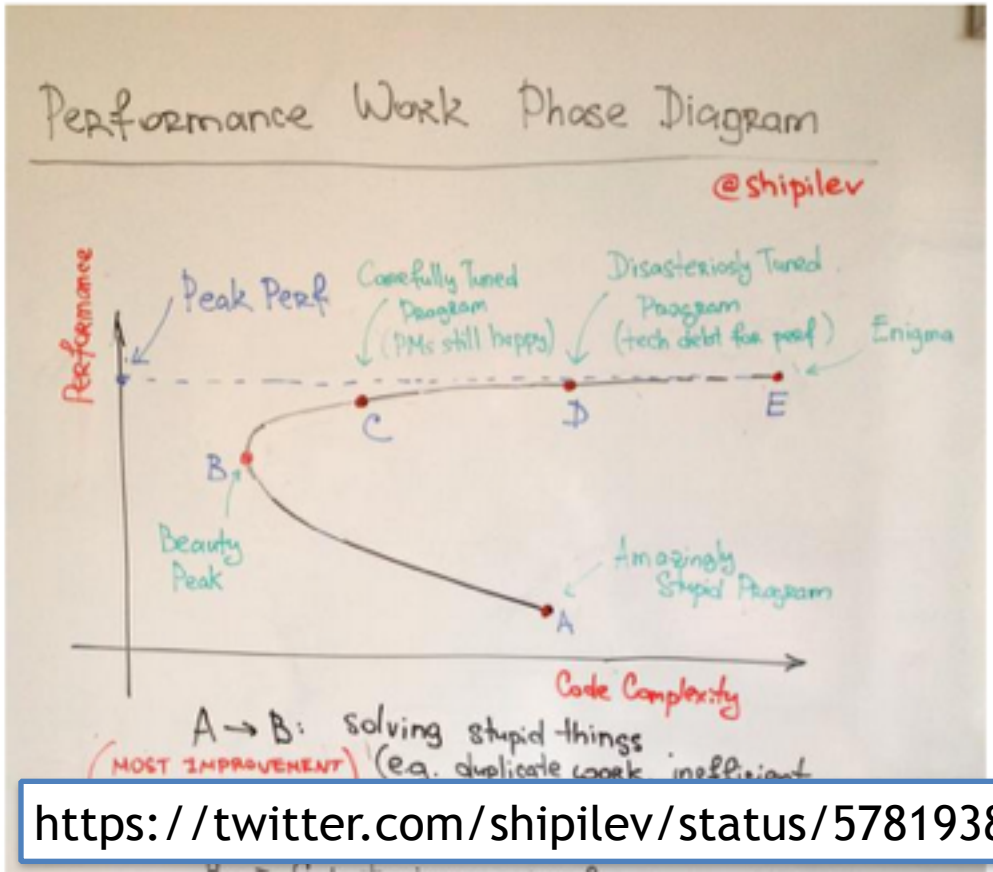
REBEL LABS by ZEROTURNAROUND

The Journey of Performance

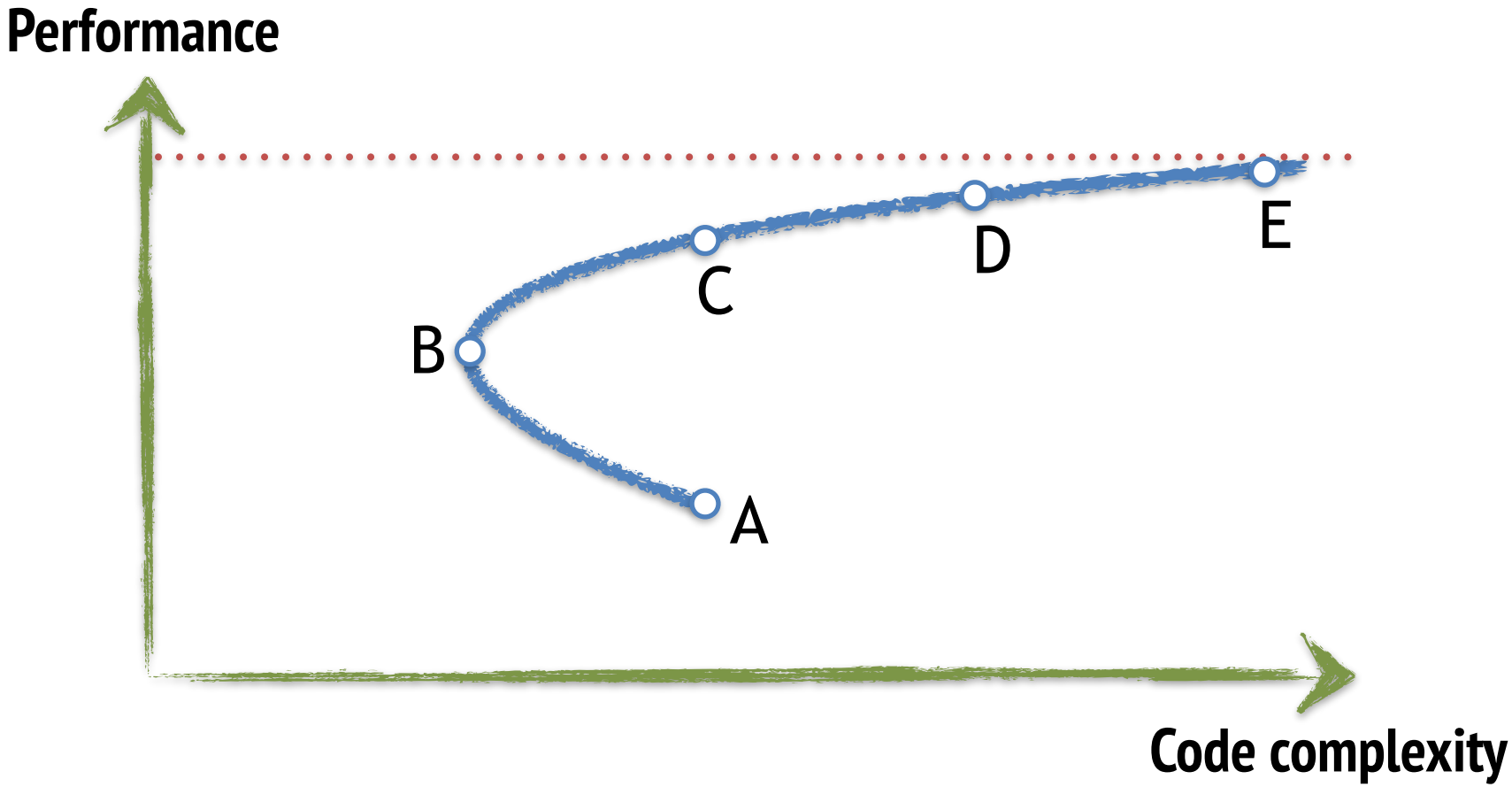
Aleksey Shipilëv
@shipilev

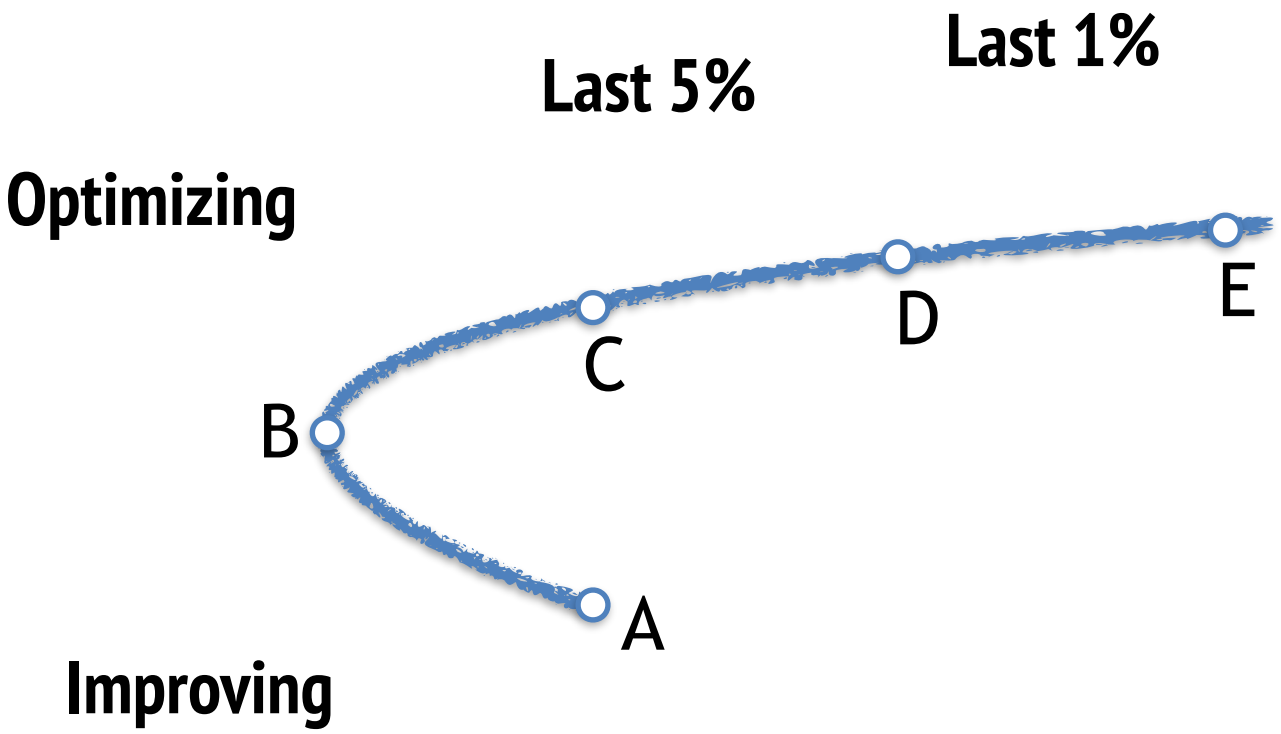
Following

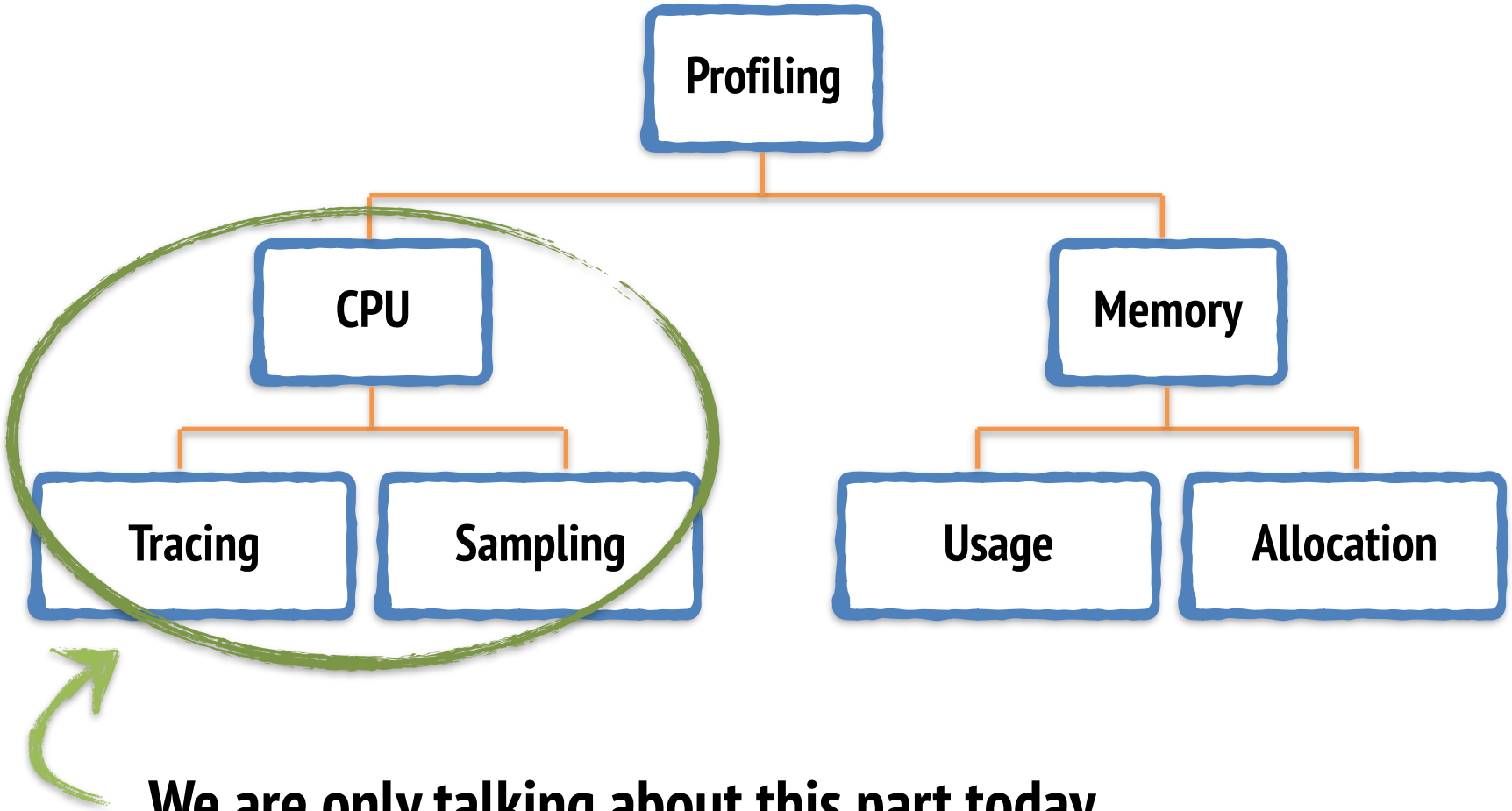
I relax by drawing stuff on my large whiteboard. Here's "Perf Work Phase Diagram" for you.



https://twitter.com/shipilev/status/578193813946134529


Performance
A
B
C
D
E
Code complexity


Last 1%
Last 5%
Optimizing
E
D
C
B
A
Improving


Profiling
CPU
Memory
Tracing
Sampling
Usage
Allocation
We are only talking about this part today

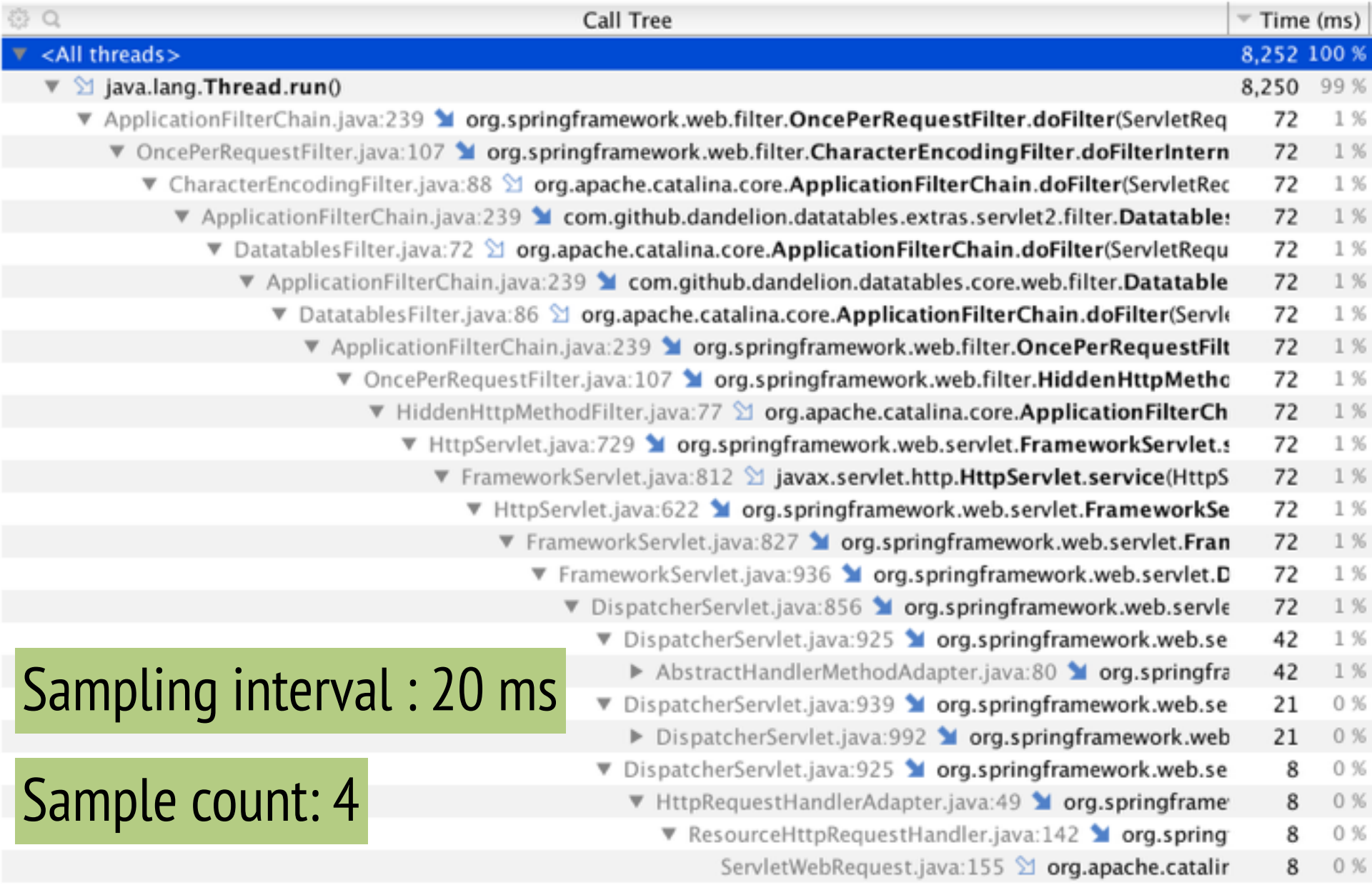
What is a sample?

JVM has ability to produce thread dump:

Press **Ctrl+Break** on Windows

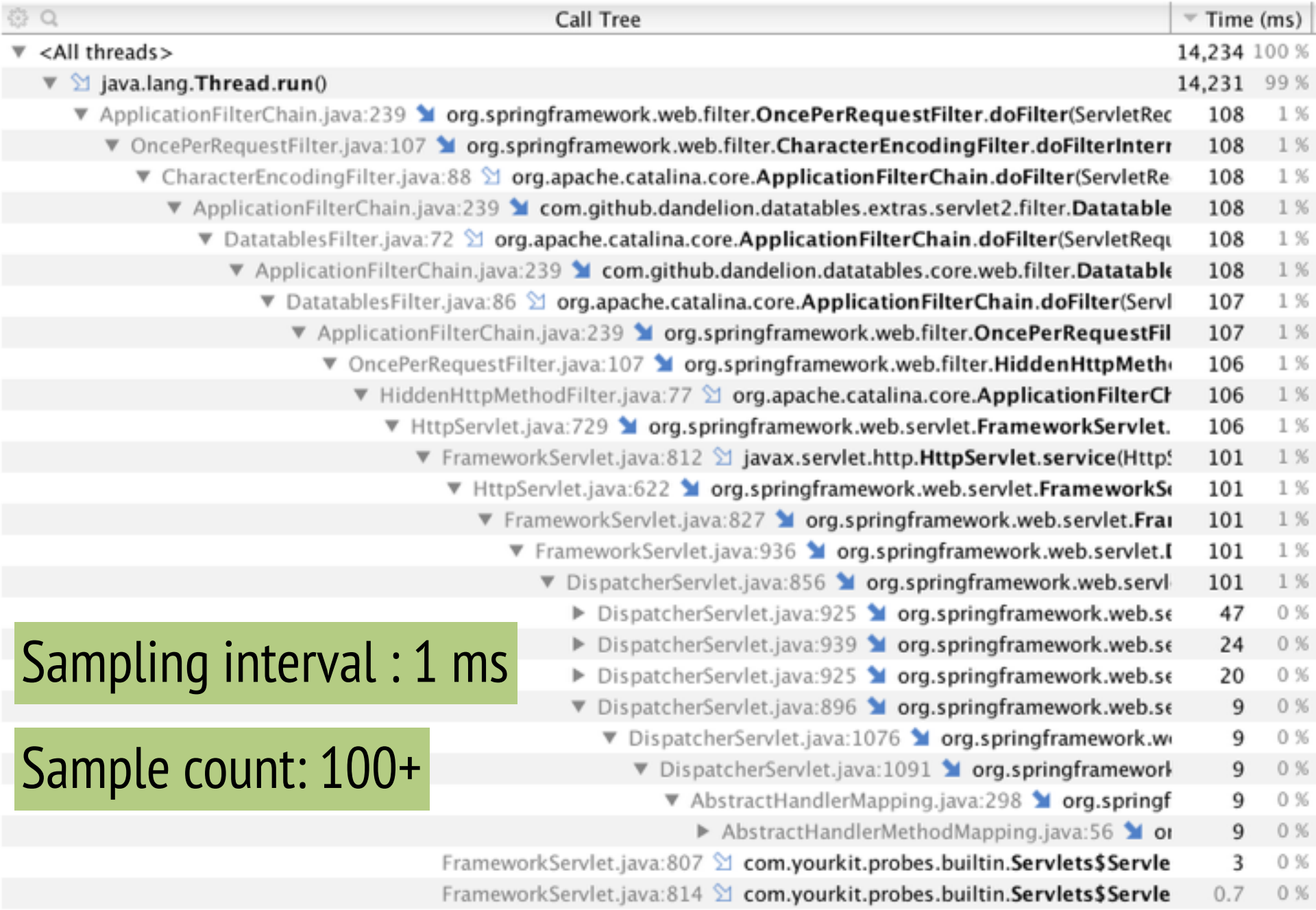
kill -3 PID on *nix

```
"http-nio-8080-exec-1" #40 daemon prio=5 os_prio=31 tid=0x00007fd7057ed800 nid=0x7313 runnable
   java.lang.Thread.State: RUNNABLE
      at o.s.s.p.w.OwnerController.processFindForm(OwnerController.java:89)
      at s.r.NativeMethodAccessorImpl.invoke0(Native Method)
      at s.r.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
      at s.r.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
      at j.l.r.Method.invoke(Method.java:497)
```



Sampling interval : 20 ms

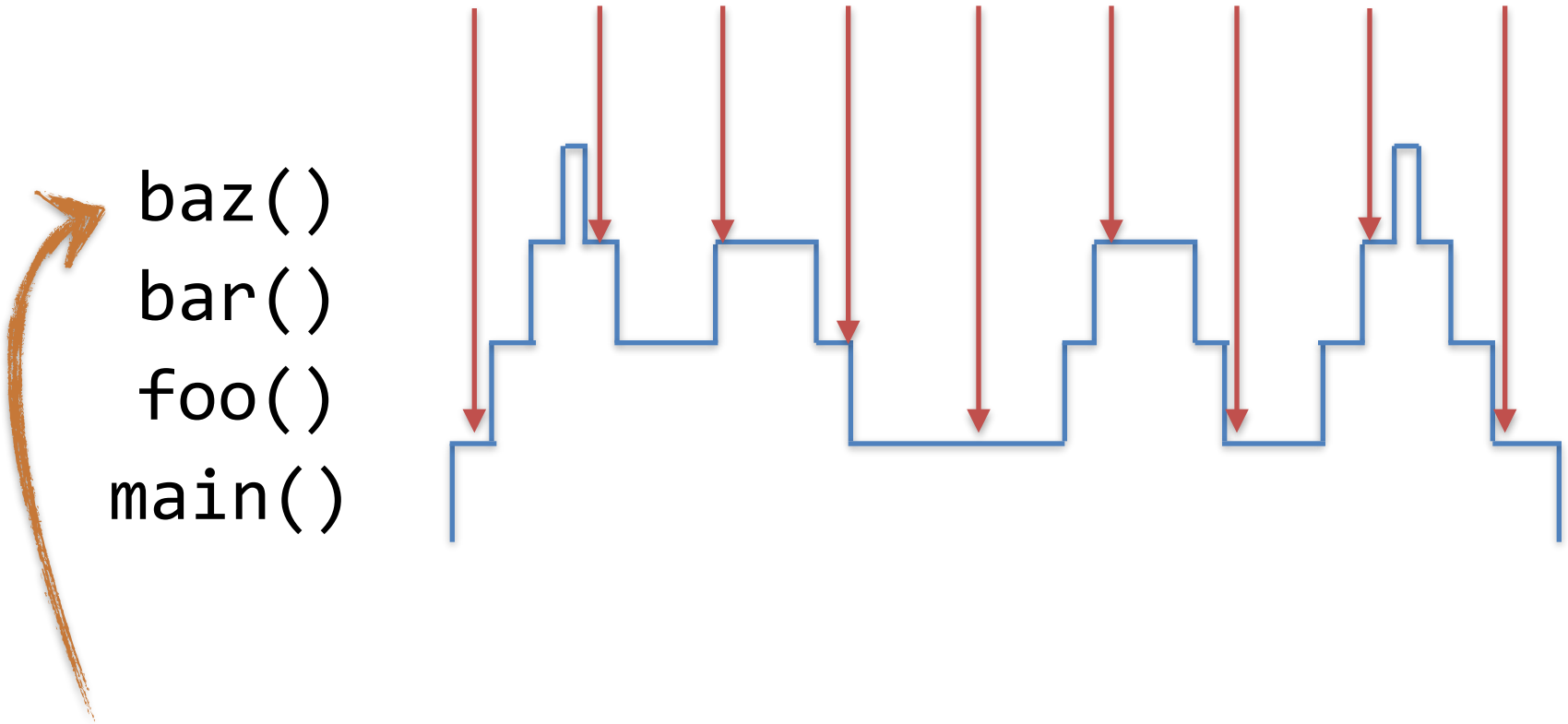
Sample count: 4



Sampling interval : 1 ms

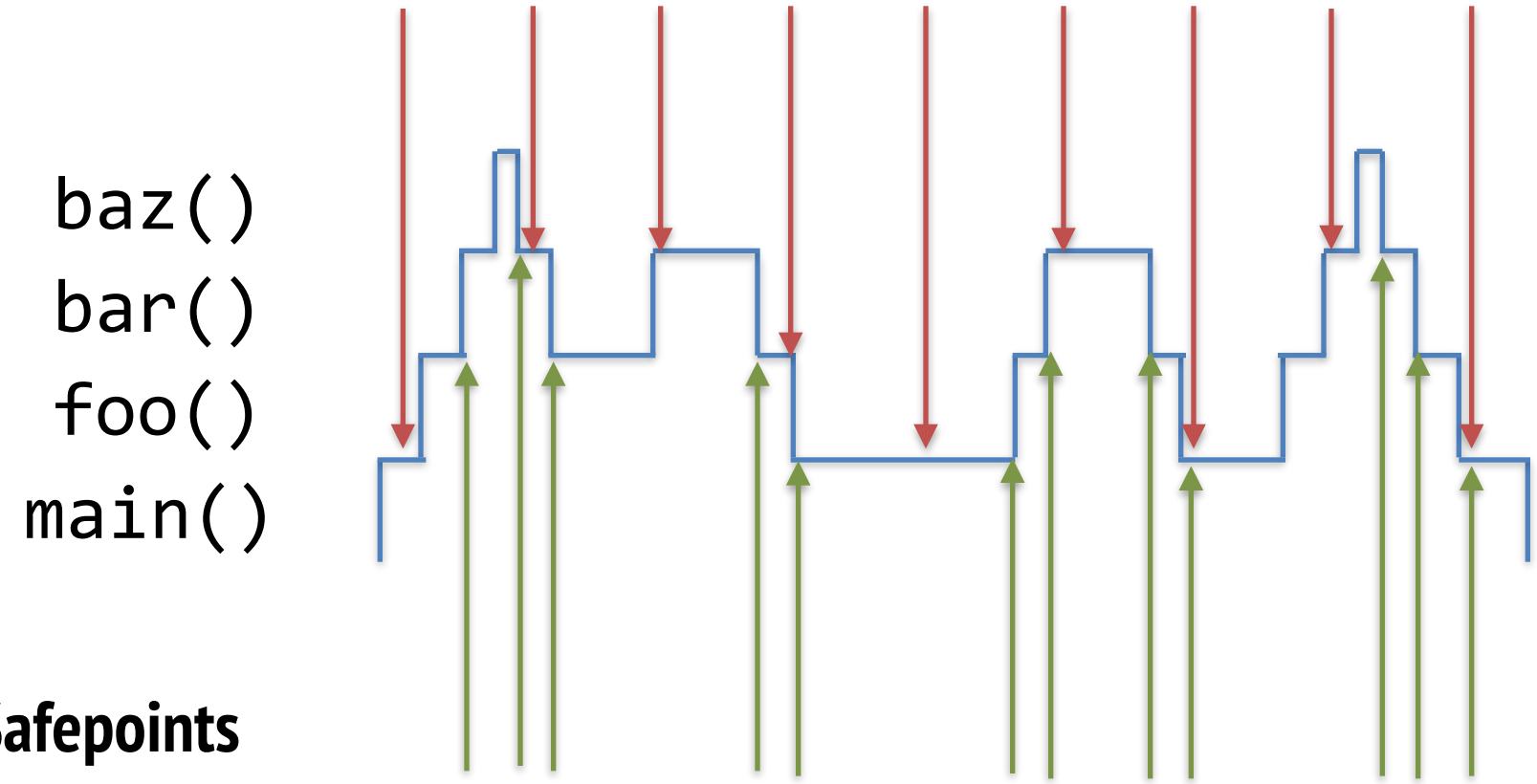
Sample count: 100+

Sampling

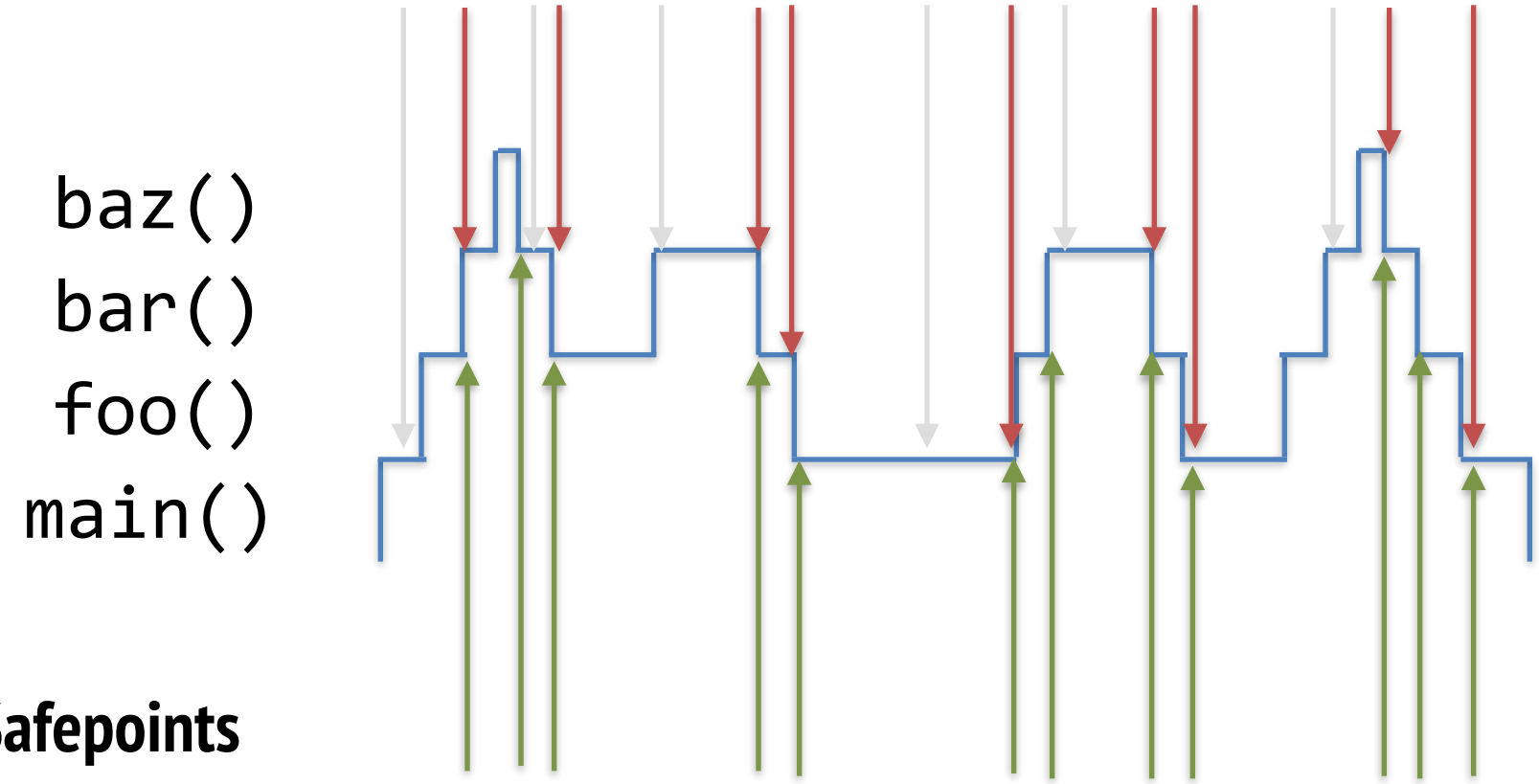


never captured :(

Safepoint bias



Safepoint bias



Taming safepoint bias

Java Mission Control

Proprietary protocol

Java 7u40

Honest Profiler

JVMTI

AsyncGetCallTrace **NB! not documented**

https://github.com/RichardWarburton/honest-profiler

Tracing

(Instrumentation)

```
public void businessMethod() {
   long start = System.currentTimeMillis();
   work();
   Profiler.log(System.currentTimeMillis()-start);
}
```

Tracing

```
public void businessMethod() {
  long start = System.nanoTime();
  work();
  Profiler.log(System.nanoTime()-start);
}
```

Tracing

```
public void businessMethod() {
  Profiler.start(“businessMethod");
  try {
    work();
  } finally {
    Profiler.log("businessMethod");
  }
}
```

System.nanoTime

System.currentTimeMillis

Parallel thread updating easily accessible memory location

- sleep-wakeup-update
- yield-wakeup-update
- busy-loop-update

```
class Profiler {

  Loop loop;

  public static void start(String method) {
    long now = loop.getTime();
    …
  }
```

Busy Loop

```
public class Loop implements Runnable {
    private volatile long time;
    public void run() {
        while (running) {
            time = System.nanoTime();
            sleep();
        }
    }

    public final long getTime() {
        return time;
    }
```

Nano seconds put into perspective

Reading memory is not free.

It takes cycles = nanoseconds

Each (software) layer is not free.

JVM, JNI, OS, HW

Nanotrusting the NanoTime

http://shipilev.net/blog/2014/nanotrusting-nanotime/

Nano seconds put into perspective

granularity_nanotime:	26.300 +- 0.205 ns	Linux
latency_nanotime:	25.542 +- 0.024 ns	
granularity_nanotime:	29.322 +- 1.293 ns	Solaris
latency_nanotime:	29.910 +- 1.626 ns	
granularity_nanotime:	371,419 +- 1,541 ns	Windows
latency_nanotime:	14,415 +- 0,389 ns	

Nanotrusting the NanoTime

http://shipilev.net/blog/2014/nanotrusting-nanotime/

Sleep timer: time-slicing & scheduling

- Minimum sleep times:
  - Win8: 1.7 ms (1.0 ms using JNI + socket poll)
  - Linux: 0.1 ms
  - OS X: 0.05 ms
  - VirtualBox + Linux: don't ask :)
  - Still uses 25-50% of CPU core
- Yield?
  - Windows scheduler skips yielded threads in case of CPU starvation

Busy Loop

```
public class Loop implements Runnable {
    private volatile long time;
    public void run() {
        while (running) {
            time = System.nanoTime();
            sleep();
        }
    }
    private void sleep() {
      if (!MiscUtil.isWindows()) {
         Thread.yield();
      }
    }
```

Busy Loop

```
public class Loop implements Runnable {
    private volatile long time;
    public void run() {
        while (running) {
            time = System.nanoTime();
            sleep();
        }
    }
    private void sleep() {
      if (!MiscUtil.isWindows()) {
         Thread.yield();
      }
    }
```

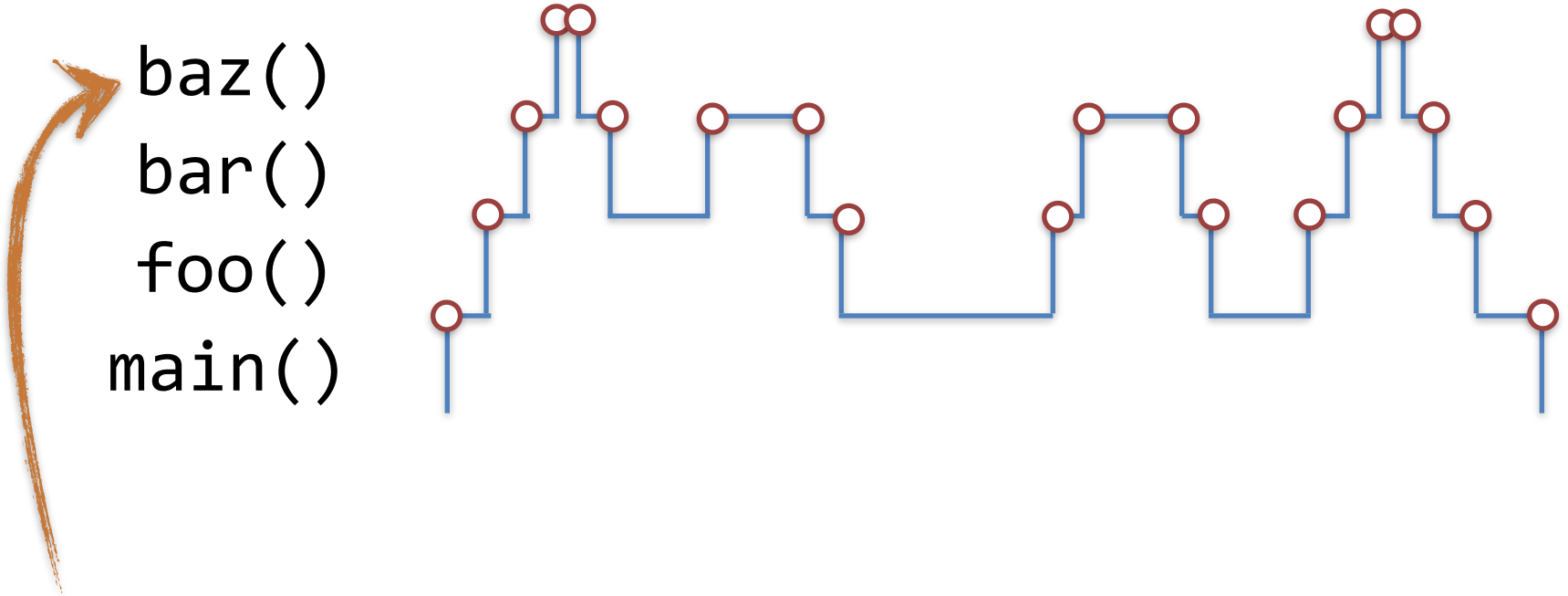
Busy Loop

```
public class Loop implements Runnable {
    private volatile long time;
    public void run() {
        while (running) {
            time = System.nanoTime();
            sleep();
        }
    }
    private void sleep() {
      if (!MiscUtil.isWindows()) {
         Thread.yield();
      }
    }
```

Busy Loop

```
public class Loop implements Runnable {
    private volatile long time;
    public void run() {
        while (running) {
            time = System.nanoTime();
            sleep();
        }
    }
    private void sleep() {
      if (!MiscUtil.isWindows()) {
         Thread.yield();
      }
    }
```

Tracing



relatively higher overhead for fast methods :(

What other performance considerations should we have?

Various application layers
3rd-party components
Database access
Web Services
RMI

Caches
File system access
Exceptions
HTTP session
View rendering
Serialization
GC

Questions you should be asking

……..

……..

……..

……..

Questions you should be asking

Where was most of the time spent?

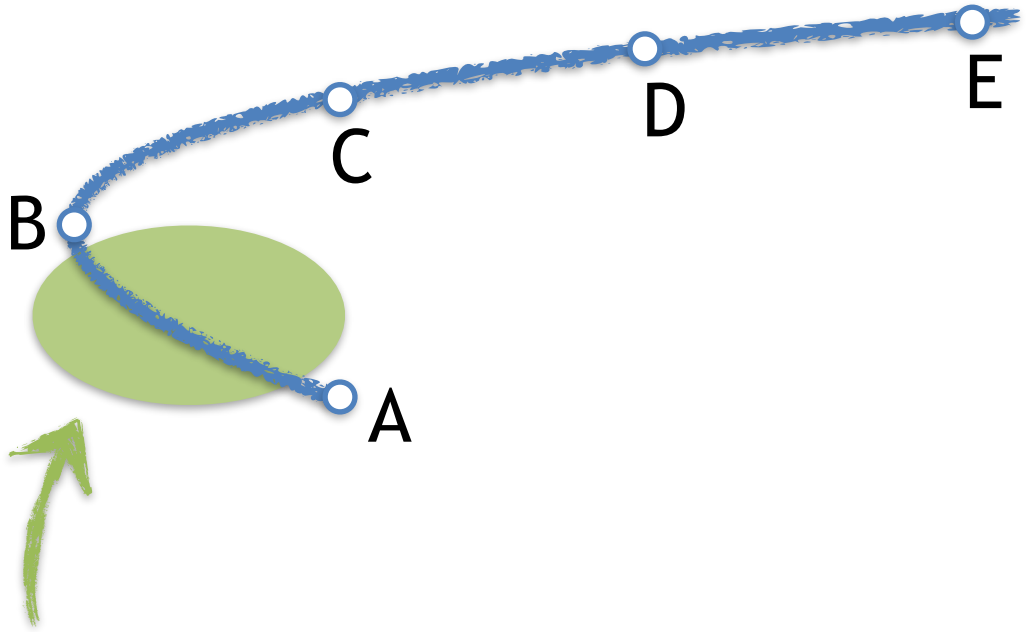
How many SQL queries were executed?

How much time did the external calls take?

Is caching working properly?

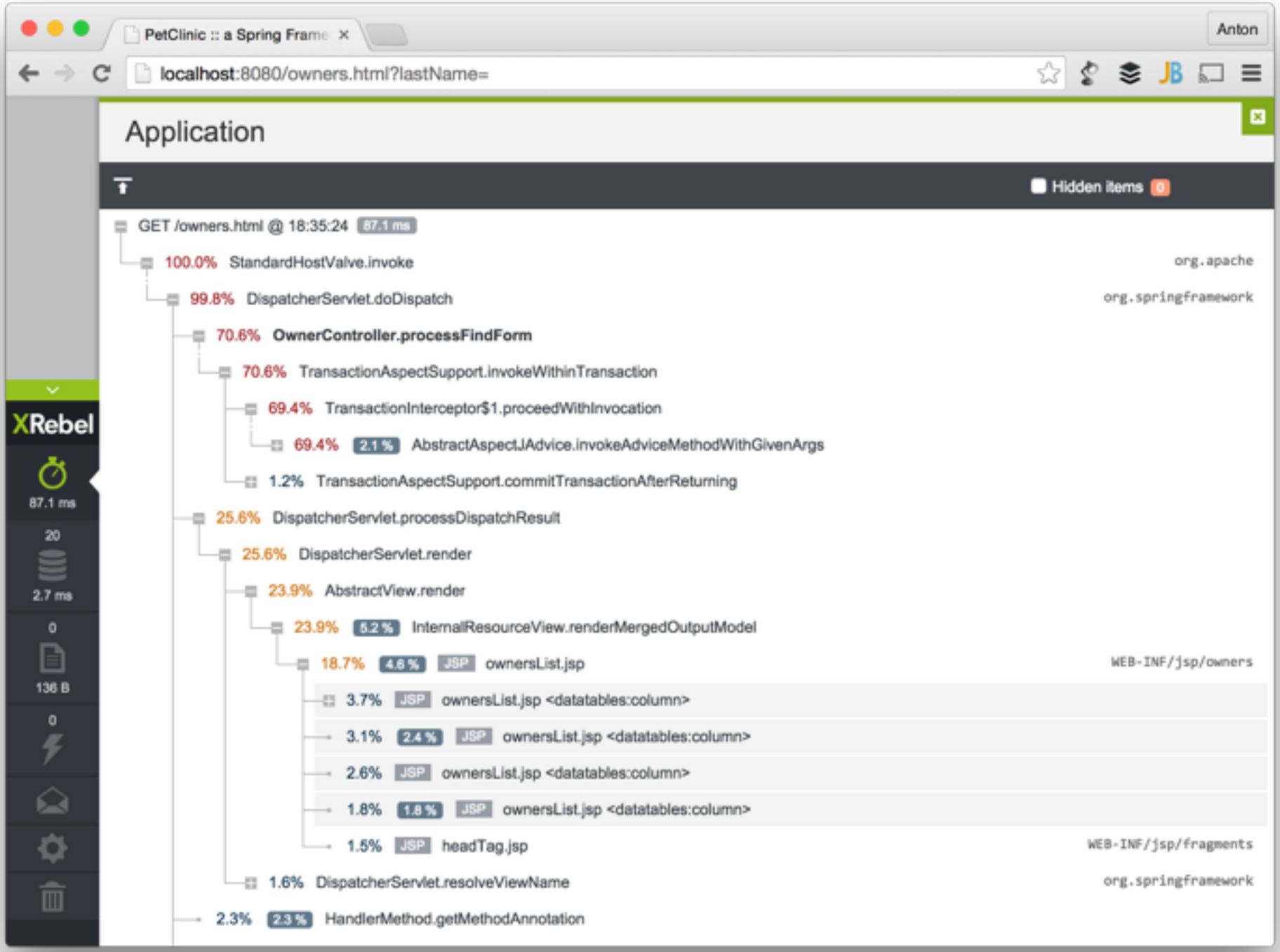
What's my HTTP session state?

Is there any excessive work done by the app?



Most improvement gained here!

<u>This is where we should focus first!</u>


PetClinic :: a Spring Frame
Anton
localhost:8080/owners.html?lastName=
Application
Hidden items 0
GET /owners.html @ 18:35:24 87.1 ms
100.0% StandardHostValve.invoke org.apache
99.8% DispatcherServlet.doDispatch org.springframework
70.6% OwnerController.processFindForm
70.6% TransactionAspectSupport.invokeWithinTransaction
69.4% TransactionInterceptor$1.proceedWithInvocation
69.4% 2.1 % AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs
1.2% TransactionAspectSupport.commitTransactionAfterReturning
25.6% DispatcherServlet.processDispatchResult
25.6% DispatcherServlet.render
23.9% AbstractView.render
23.9% 5.2 % InternalResourceView.renderMergedOutputModel
18.7% 4.6 % JSP ownersList.jsp WEB-INF/jsp/owners
3.7% JSP ownersList.jsp <datatables:column>
3.1% 2.4 % JSP ownersList.jsp <datatables:column>
2.6% JSP ownersList.jsp <datatables:column>
1.8% 1.8 % JSP ownersList.jsp <datatables:column>
1.5% JSP headTag.jsp WEB-INF/jsp/fragments
1.6% DispatcherServlet.resolveViewName org.springframework
2.3% 2.3 % HandlerMethod.getMethodAnnotation
XRebel
87.1 ms
20
2.7 ms
0
136 B
0

Download trials and get FREE shirts :)

JRebel: http://0t.ee/docklandsjr

XRebel: http://0t.ee/docklandsxr