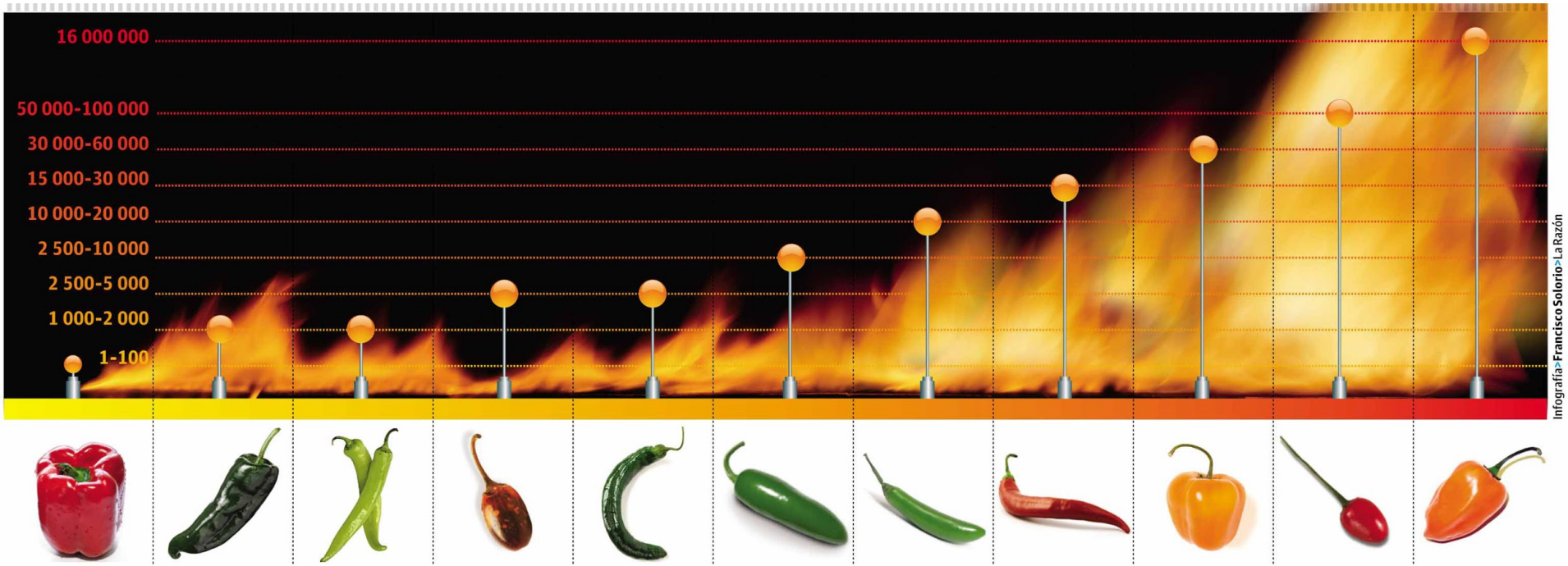
The Art of JVM Profiling

Andrei Pangin
Vadim Tsesko

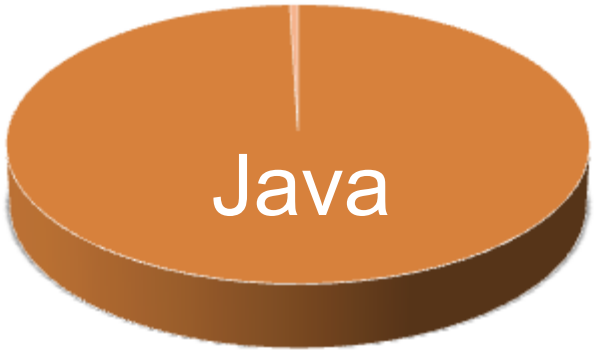
2017



http://recetasfamilia.com/escala-scoville/

Одноклассники

- 48 M DAU
- 8500 machines in 4 DC
- 1.2 Tb/s
- Up to 70 K QPS/server
- 99% < 100 ms



Profilers

0

What to profile?

- IO (disk, network)
- syscalls
- Synchronisation
- SQL queries
- …

How to profile?

How to profile?

Instrumenting

- Trace method transitions
- Measure/count

- **Slooow**

How to profile?

Instrumenting

- Trace method transitions
- Measure/count

- **Slooow**

Sampling

- Snapshot state
- Periodic

- **Suitable for PROD**

Thread Dump

1

How does it work?

Java

Thread.getAllStackTraces()

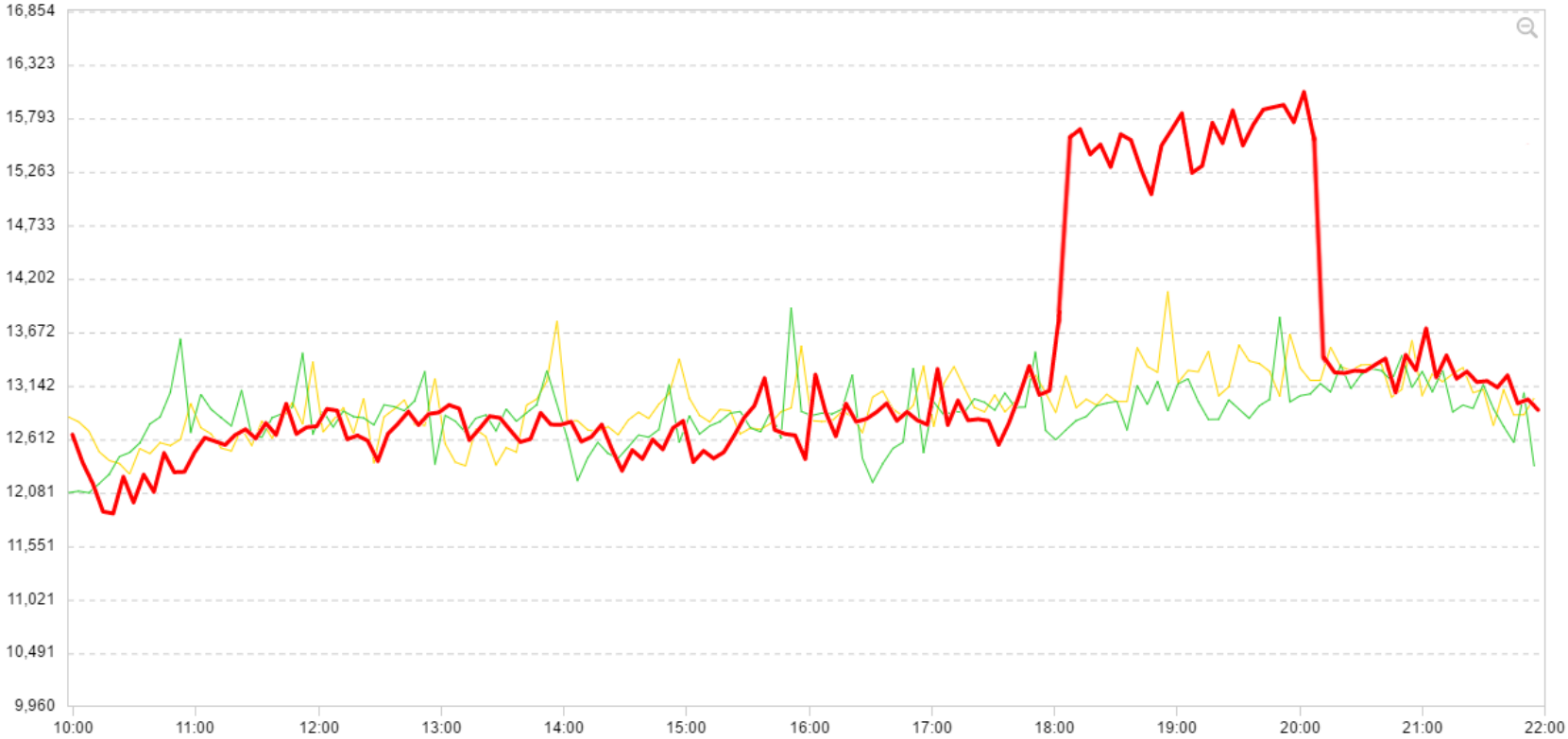
```
class StackTraceElement {
    String declaringClass;
    String methodName;
    String fileName;
    int    lineNumber;
}
```

Native (JVM TI)

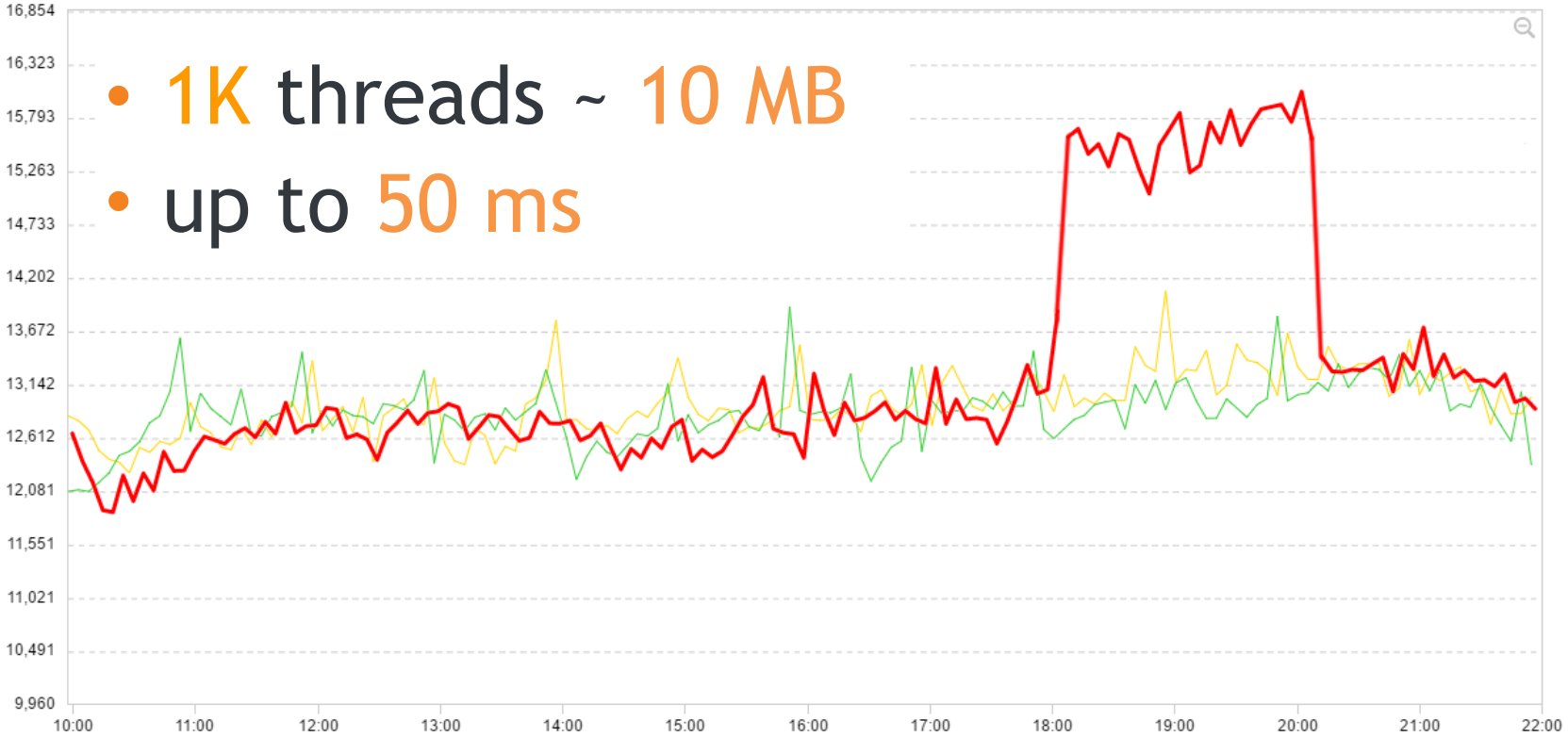
GetAllStackTraces()

```
struct {
    jmethodID method;
    jlocation location;
}
```

Overhead


16,854
16,323
15,793
15,263
14,733
14,202
13,672
13,142
12,612
12,081
11,551
11,021
10,491
9,960
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00

Overhead


• 1K threads ~ 10 MB
• up to 50 ms
16,854
16,323
15,793
15,263
14,733
14,202
13,672
13,142
12,612
12,081
11,551
11,021
10,491
9,960
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00

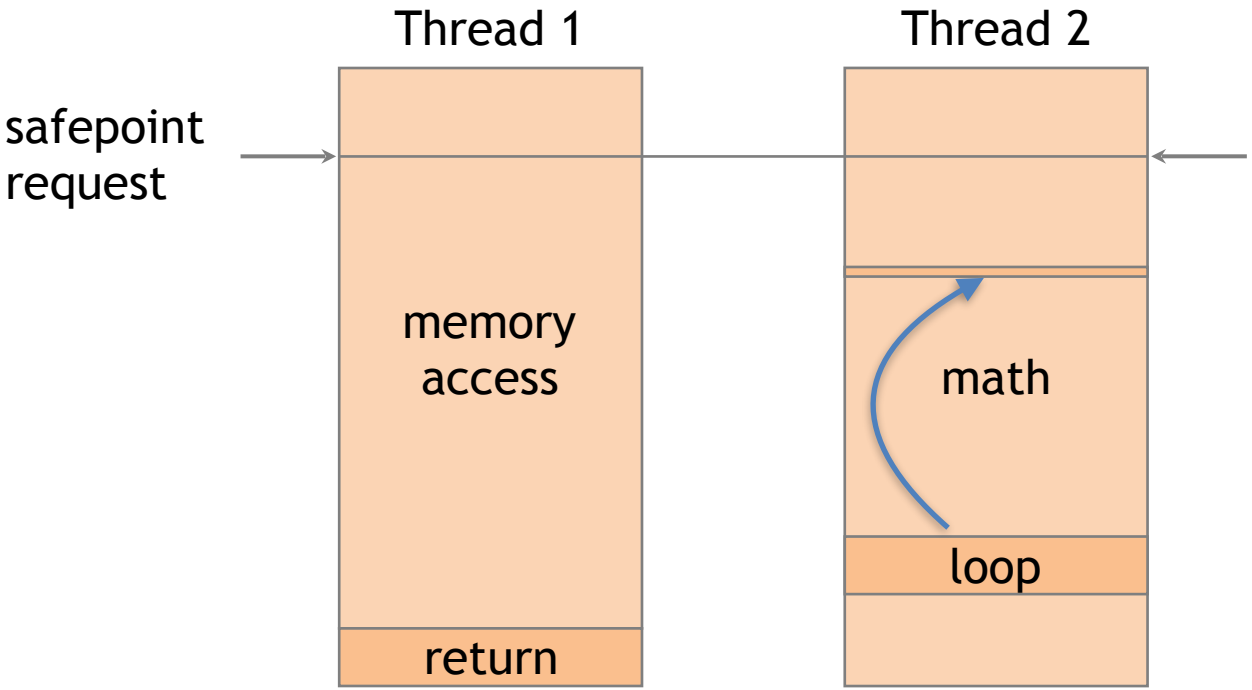
Advantages

- Simple
- All Java platforms and versions
- No JVM options needed

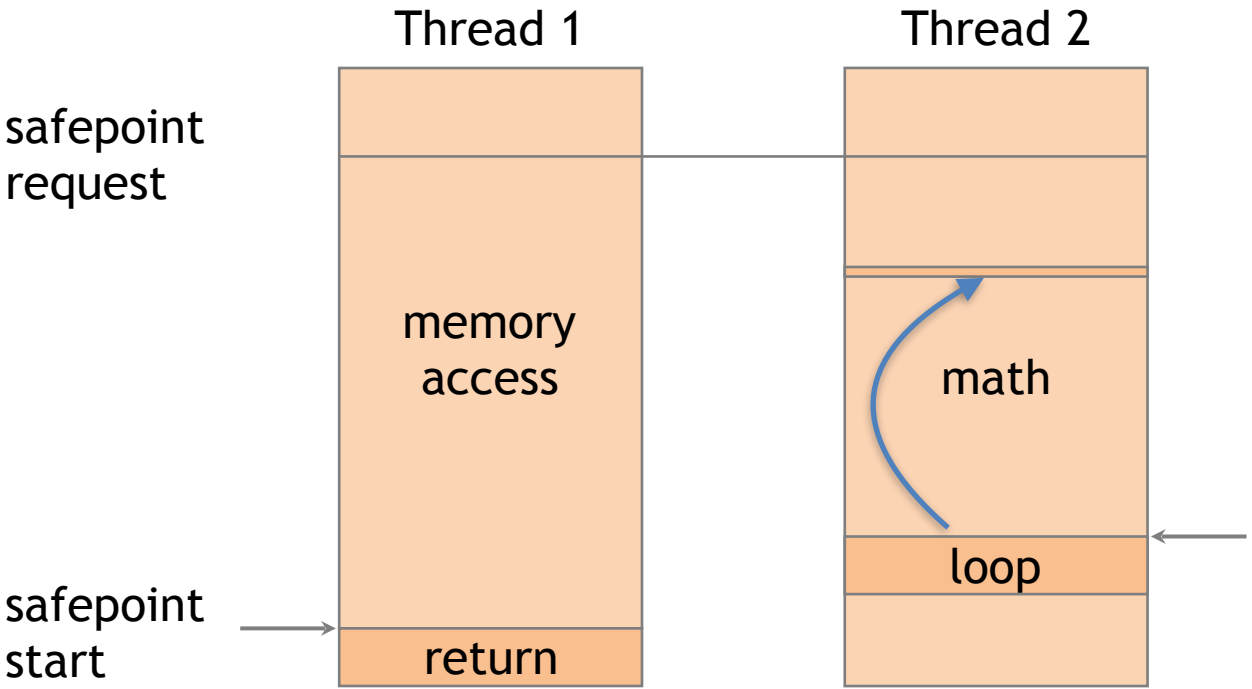
- VisualVM, Java Mission Control, YourKit, JProfiler, ...

DEMO

Safepoint


Thread 1
Thread 2
safepoint
request
memory
access
return
math
loop

Safepoint


Thread 1
Thread 2
safepoint
request
memory
access
math
loop
safepoint
start
return

Are we there yet?

for-loop

```
public Theme getThemeById(Long id) {
    for (int i = 0; i < themes.length; i++) {
        if (id.equals(themes[i].getId())) {
            return themes[i];
        }
    }
    return null;
}
```

for-loop

```
public Theme getThemeById(Long id) {
    for (int i = 0; i < themes.length; i++) {
        if (id.equals(themes[i].getId())) {
            return themes[i];
        }
    }
    return null;
}
```

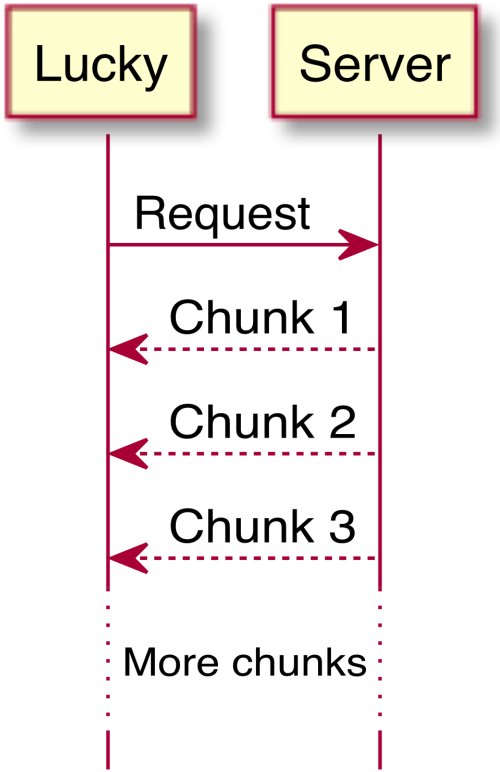
-XX:+UseCountedLoopSafepoints

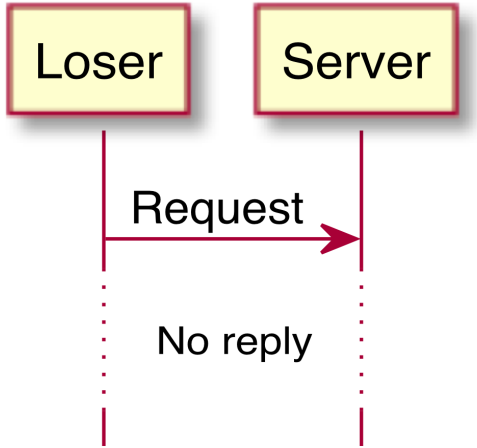
Safepoints make profiling

- Useless
- Unreliable
- It happens

https://jug.ru/2016/05/андрей-паньгин-всё-что-вы-хотели-зна/
http://psy-lob-saw.blogspot.ru/2016/02/why-most-sampling-java-profilers-are.html

DEMO


Lucky
Server
Request
Chunk 1
Chunk 2
Chunk 3
More chunks


Loser
Server
Request
No reply

Off CPU

- All threads
- Native → RUNNABLE
- How to interpret?

Can we do better?

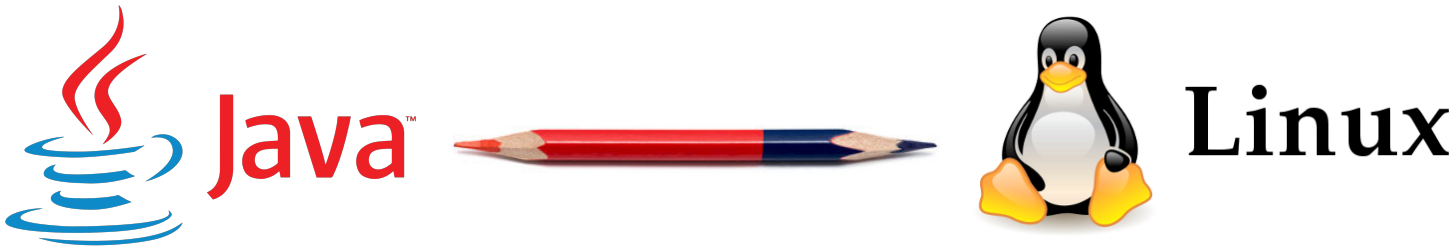
Problems

- Safepoints
- Off CPU
- Native

Can we do better?

Problems

- Safepoints
- Off CPU
- Native



AsyncGetCallTrace

2

How does it work?

```
AsyncGetCallTrace(ASGCT_CallTrace *trace,
                  jint depth,
                  void* ucontext)
```

from signal handler
itimer() + SIGPROF

How does it work?

```
AsyncGetCallTrace(ASGCT_CallTrace *trace,
                  jint depth,
                  void* ucontext)
```

from signal handler
itimer() + SIGPROF

- Oracle Developer Studio
- github.com/jvm-profiling-tools/honest-profiler
- github.com/apangin/async-profiler

DEMO

Advantages

- Not limited to safepoints

  -XX:+DebugNonSafepoints
- Active threads
- All Java: interpreted, compiled, inlined

Disadvantages

- ✖ Windows
- ✖ Native
- ✖ JVM (GC, compiler…)

DEMO

Problems

```
enum {
  ticks_no_Java_frame          =  0,
  ticks_no_class_load          = -1,
  ticks_GC_active              = -2,
  ticks_unknown_not_Java       = -3,
  ticks_not_walkable_not_Java  = -4,
  ticks_unknown_Java           = -5,
  ticks_not_walkable_Java      = -6,
  ticks_unknown_state          = -7,
  ticks_thread_exit            = -8,
  ticks_deopt                  = -9,
  ticks_safepoint              = -10
};
```

src/share/vm/prims/forte.cpp

Inconsistent frame

```
public int getX() {
    return x;
}
```

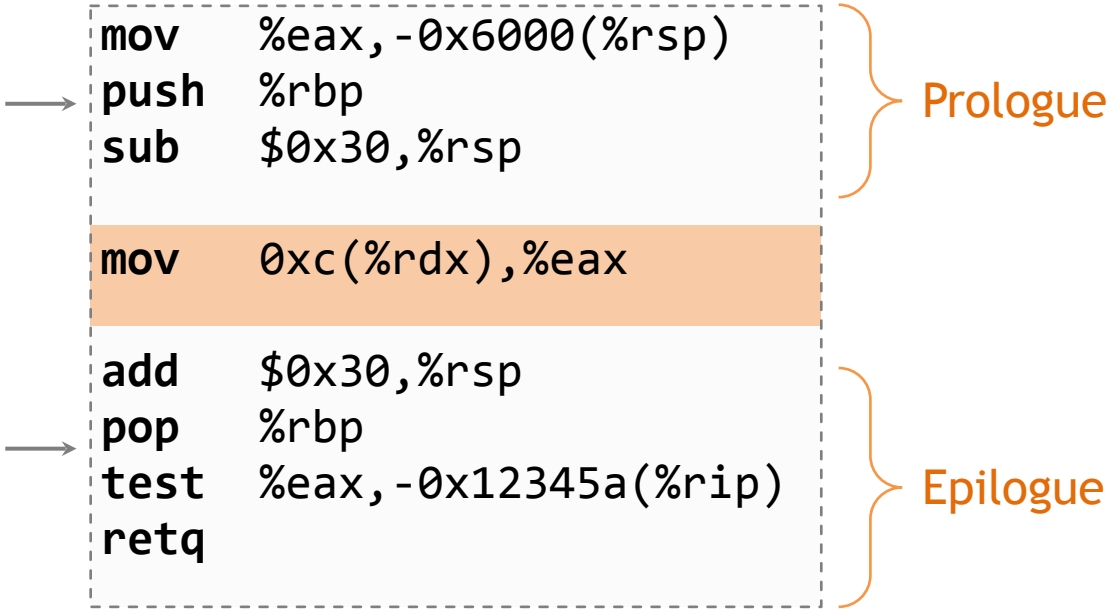
```
mov   %eax,-0x6000(%rsp)
push  %rbp
sub   $0x30,%rsp

mov   0xc(%rdx),%eax

add   $0x30,%rsp
pop   %rbp
test  %eax,-0x12345a(%rip)
retq
```

Inconsistent frame

```
public int getX() {
    return x;
}
```



Workaround

1. Fix SP, IP
2. Retry AsyncGetCallTrace()

unknown_Java < 0.05%

bugs.openjdk.java.net/browse/JDK-8178287

DEMO

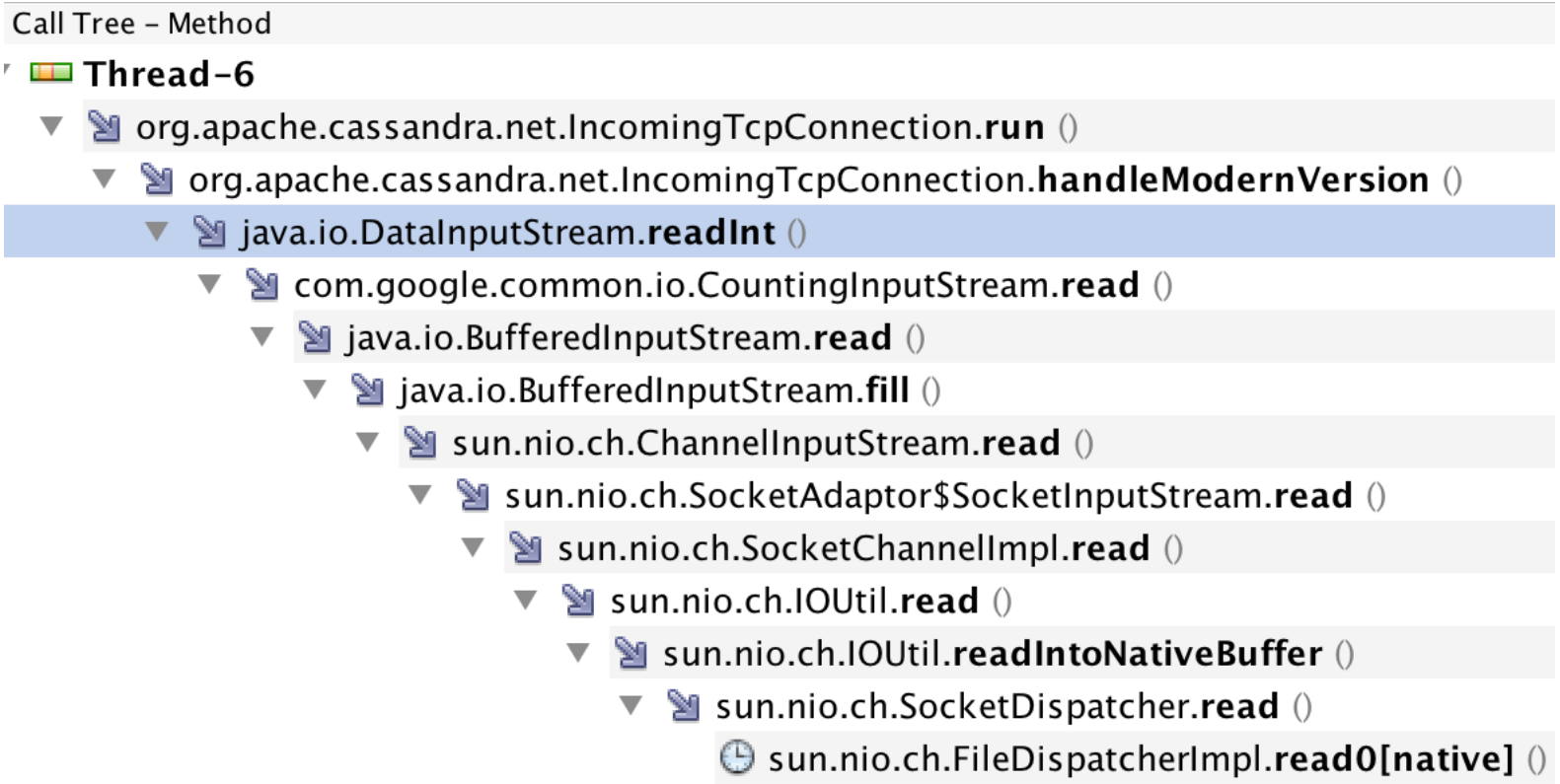
Visualisation

3

Flat

Hot Spots – Method	Self Time [%]
sun.nio.ch.FileDispatcherImpl.**read0[native]** ()	
java.net.SocketInputStream.**socketRead0[native]** ()	
one.nio.net.NativeSelector.**epollWait[native]** ()	
one.nio.net.NativeSocket.**accept0[native]** ()	
sun.nio.ch.ServerSocketChannelImpl.**accept0[native]** ()	
sun.management.ThreadImpl.**dumpThreads0[native]** ()	
sun.misc.Unsafe.**unpark[native]** ()	
sun.reflect.Reflection.**getCallerClass[native]** ()	
sun.nio.ch.NativeThread.**current[native]** ()	
sun.nio.ch.FileDispatcherImpl.**write0[native]** ()	
sun.misc.Unsafe.**park[native]** ()	
org.apache.cassandra.locator.DynamicEndpointSnitch.**getDatacenter** ()	
org.apache.cassandra.locator.AbstractNetworkTopologySnitch.**compareEndpoints** ()	
java.lang.Object.**hashCode[native]** ()	
java.lang.Object.**notifyAll[native]** ()	
java.util.concurrent.ConcurrentHashMap.**get** ()	
java.security.AccessController.**doPrivileged[native]** ()	
java.lang.String.**intern[native]** ()	

Tree


Call Tree - Method
Thread-6
org.apache.cassandra.net.IncomingTcpConnection.run ()
org.apache.cassandra.net.IncomingTcpConnection.handleModernVersion ()
java.io.DataInputStream.readInt ()
com.google.common.io.CountingInputStream.read ()
java.io.BufferedInputStream.read ()
java.io.BufferedInputStream.fill ()
sun.nio.ch.ChannelInputStream.read ()
sun.nio.ch.SocketAdaptor$SocketInputStream.read ()
sun.nio.ch.SocketChannelImpl.read ()
sun.nio.ch.IOUtil.read ()
sun.nio.ch.IOUtil.readIntoNativeBuffer ()
sun.nio.ch.SocketDispatcher.read ()
sun.nio.ch.FileDispatcherImpl.read0[native] ()

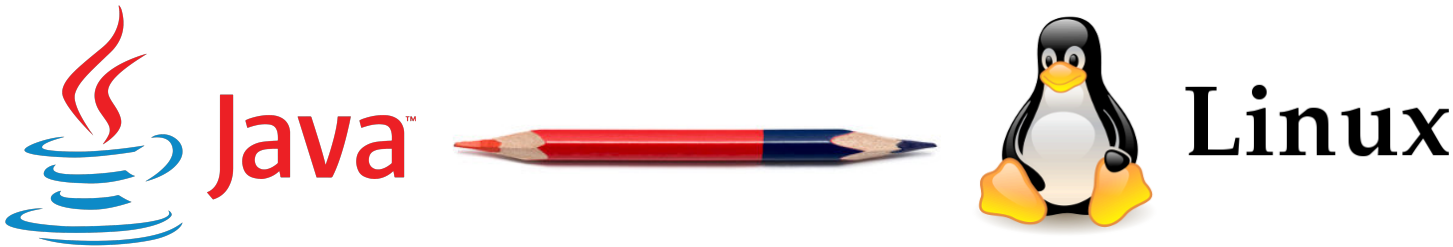
DEMO

- brendangregg.com

Can we do better?

Problems

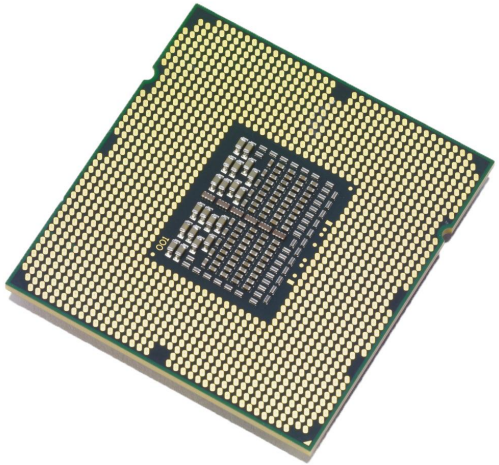
- Safepoints
- Off CPU
- Native



Perf Events

4

PMU



PMU

HW
interrupts

- HW Events
  - Cycles, instructions
  - Cache misses, branch misses

PMU

HW
interrupts

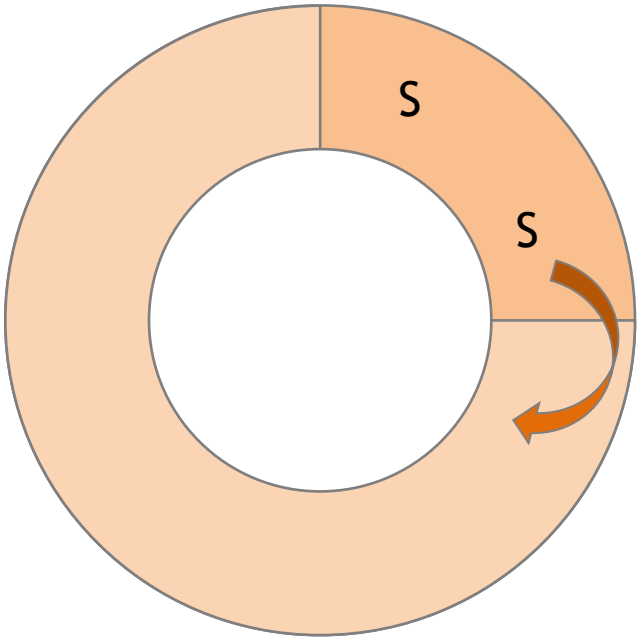
- HW Events
  - Cycles, instructions
  - Cache misses, branch misses
- SW events
  - CPU clock
  - Page faults
  - Context switches

perf_event_open()

- Linux syscall
  - fd → counter
  - mmap page → samples

perf_event_open()

- Linux syscall
  - fd → counter
  - mmap page → samples
- Samples
  - pid, tid
  - CPU registers
  - Call chain (user + kernel)



perf

```
$ perf record -F 999 java ...
$ perf report
```

perf.wiki.kernel.org/index.php/Tutorial

perf

```
$ perf record -F 999 java ...
$ perf report
```

```
  4.70%    java    [kernel.kallsyms]     [k] clear_page_c
  2.10%    java    libpthread-2.17.so    [.] pthread_cond_wait
  1.97%    java    libjvm.so             [.] Unsafe_Park
  1.40%    java    libjvm.so             [.] Parker::park
  1.31%    java    [kernel.kallsyms]     [k] try_to_wake_up
  1.31%    java    perf-18762.map        [.] 0x00007f8510e9e757
  1.21%    java    perf-18762.map        [.] 0x00007f8510e9e89e
  1.17%    java    perf-18762.map        [.] 0x00007f8510e9cc17
```

perf.wiki.kernel.org/index.php/Tutorial

perf

```
$ perf record -F 999 java ...
$ perf report
```

```
  4.70%    java    [kernel.kallsyms]     [k] clear_page_c
  2.10%    java    libpthread-2.17.so    [.] pthread_cond_wait
  1.97%    java    libjvm.so             [.] Unsafe_Park
  1.40%    java    libjvm.so             [.] Parker::park
  1.31%    java    [kernel.kallsyms]     [k] try_to_wake_up
  1.31%    java    perf-18762.map        [.] 0x00007f8510e9e757
  1.21%    java    perf-18762.map        [.] 0x00007f8510e9e89e
  1.17%    java    perf-18762.map        [.] 0x00007f8510e9cc17
```

perf.wiki.kernel.org/index.php/Tutorial

Java symbols

- No symbols for JITted code
- /tmp/perf-<pid>.map

```
7fe0e9117220   80  java.lang.Object::<init>
7fe0e91175e0  140  java.lang.String::hashCode
7fe0e9117900   20  java.lang.Math::min
7fe0e9117ae0   60  java.lang.String::length
7fe0e9117d20  180  java.lang.String::indexOf
```

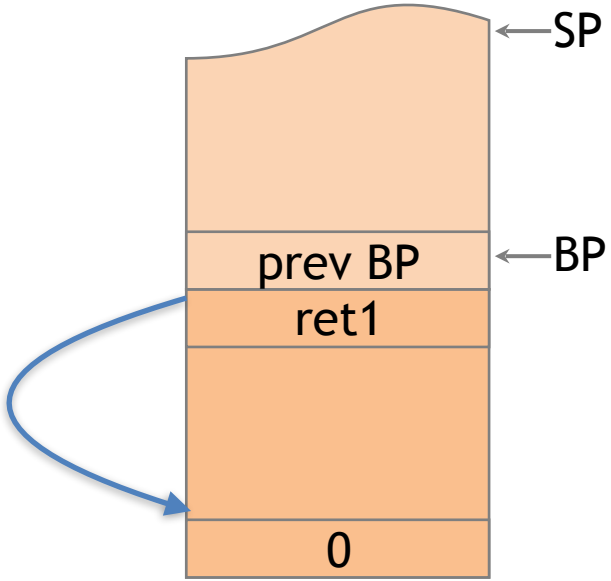
JVM TI

```
CompiledMethodLoad()        // Compiled Java
DynamicCodeGenerated()      // VM Runtime
```

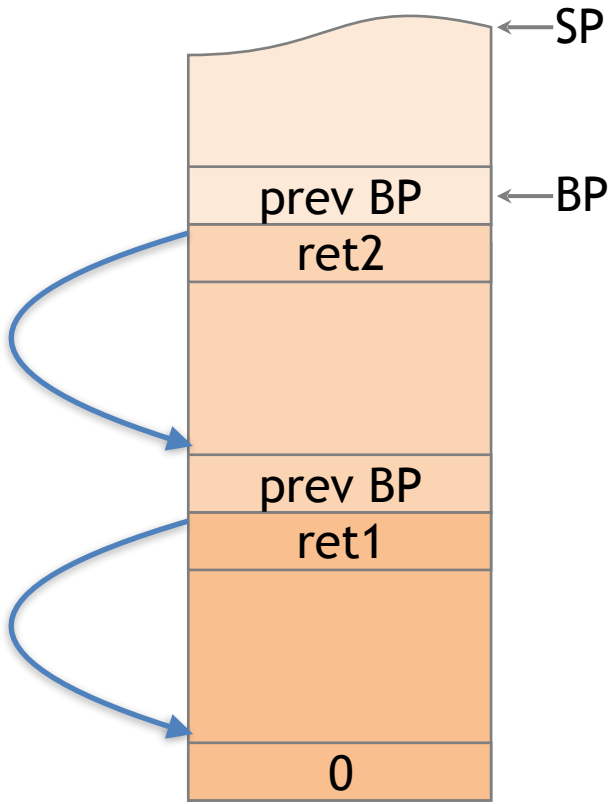
```
$ java -agentpath:/usr/lib/libperfmap.so …
```

github.com/jrudolph/perf-map-agent

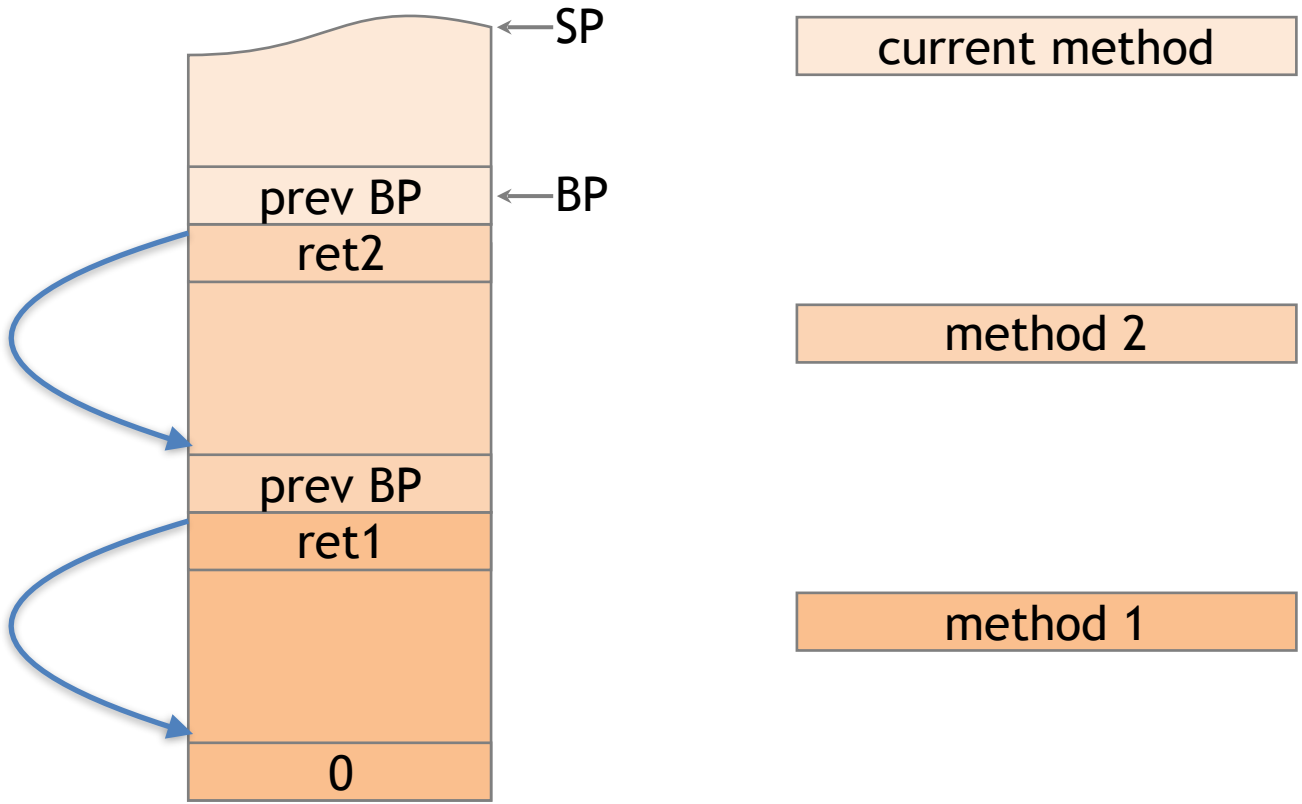
Native stack



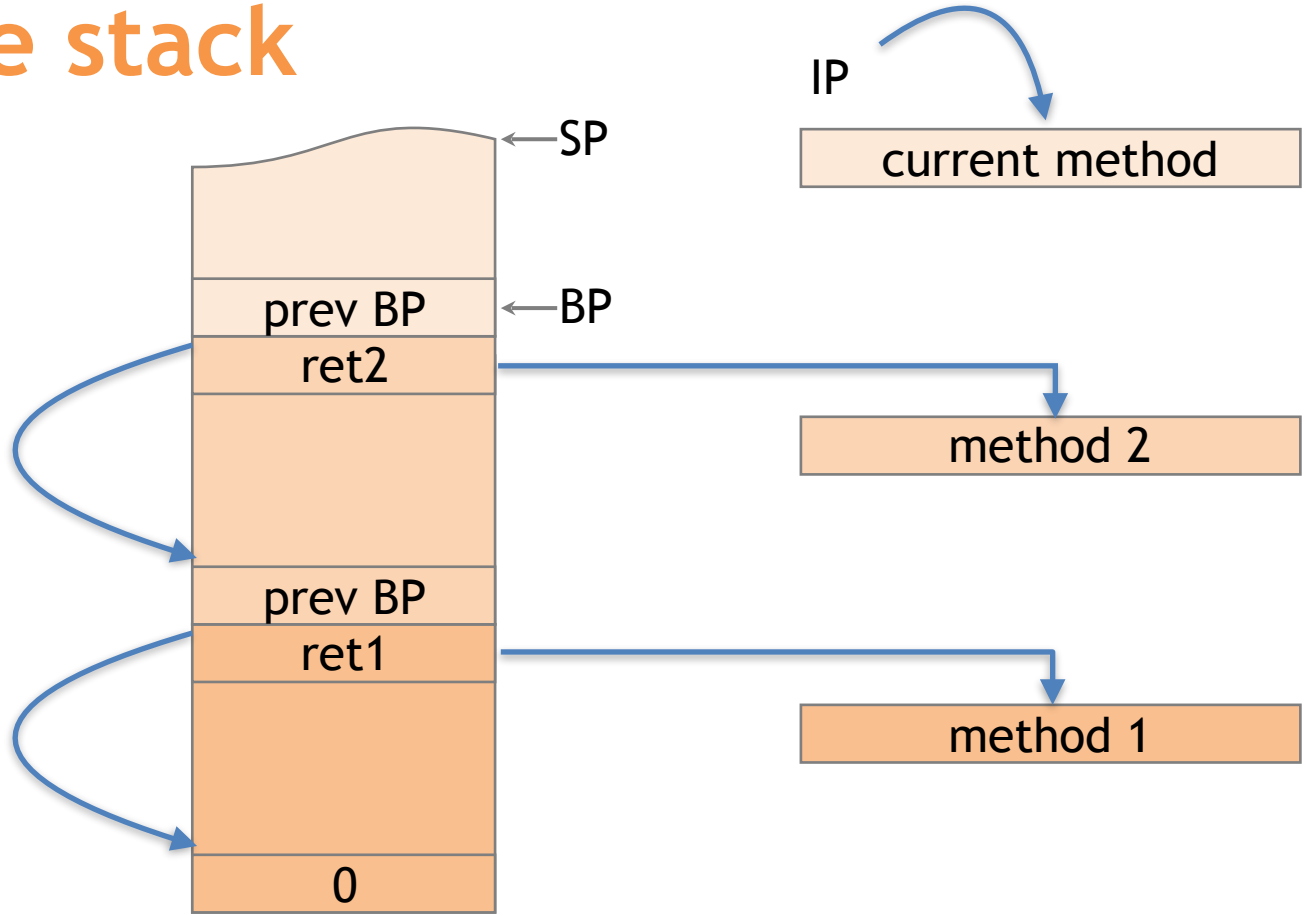
Native stack



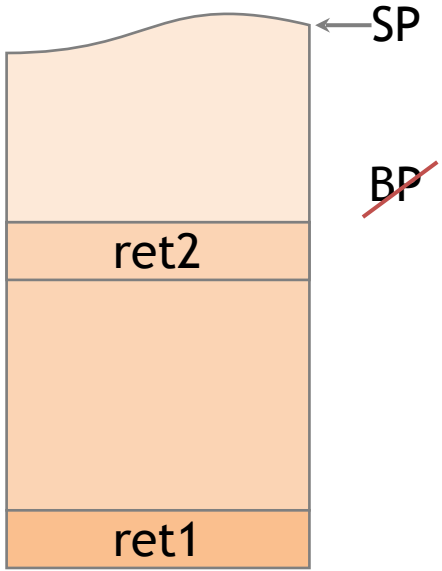
Native stack



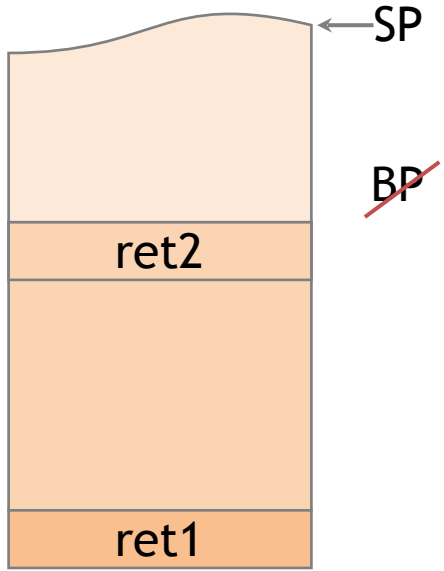
Native stack



Java stack



Java stack



-XX:+PreserveFramePointer

DEMO

```
$ perf record -F $HZ -o $RAW -g -p $PID -- sleep $SEC
```

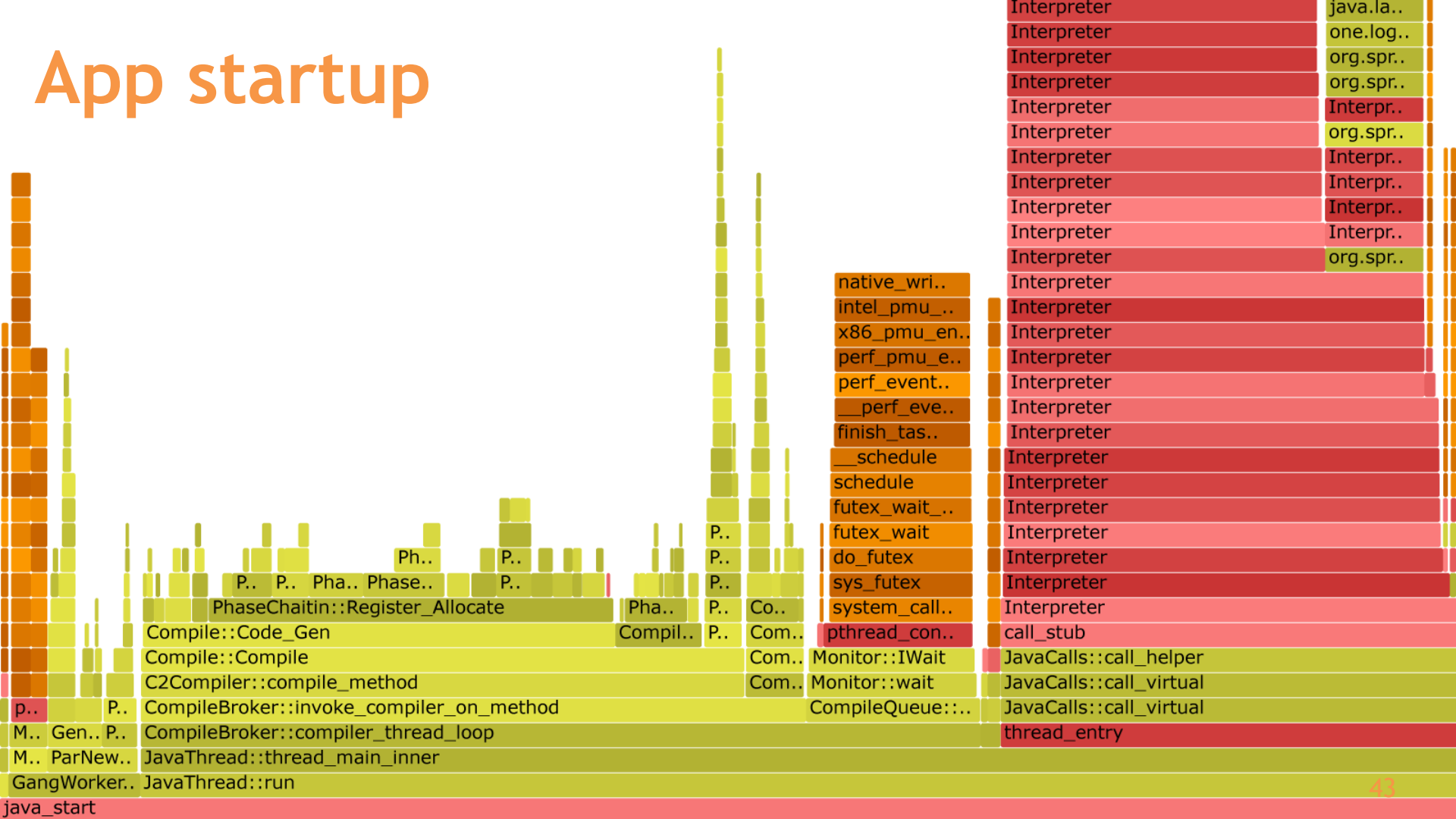
```
$ perf script -i $RAW > $PERF
```

```
$ FlameGraph/stackcollapse-perf.pl $PERF > $STACKS
```

```
$ FlameGraph/flamegraph.pl $STACKS > $SVG
```

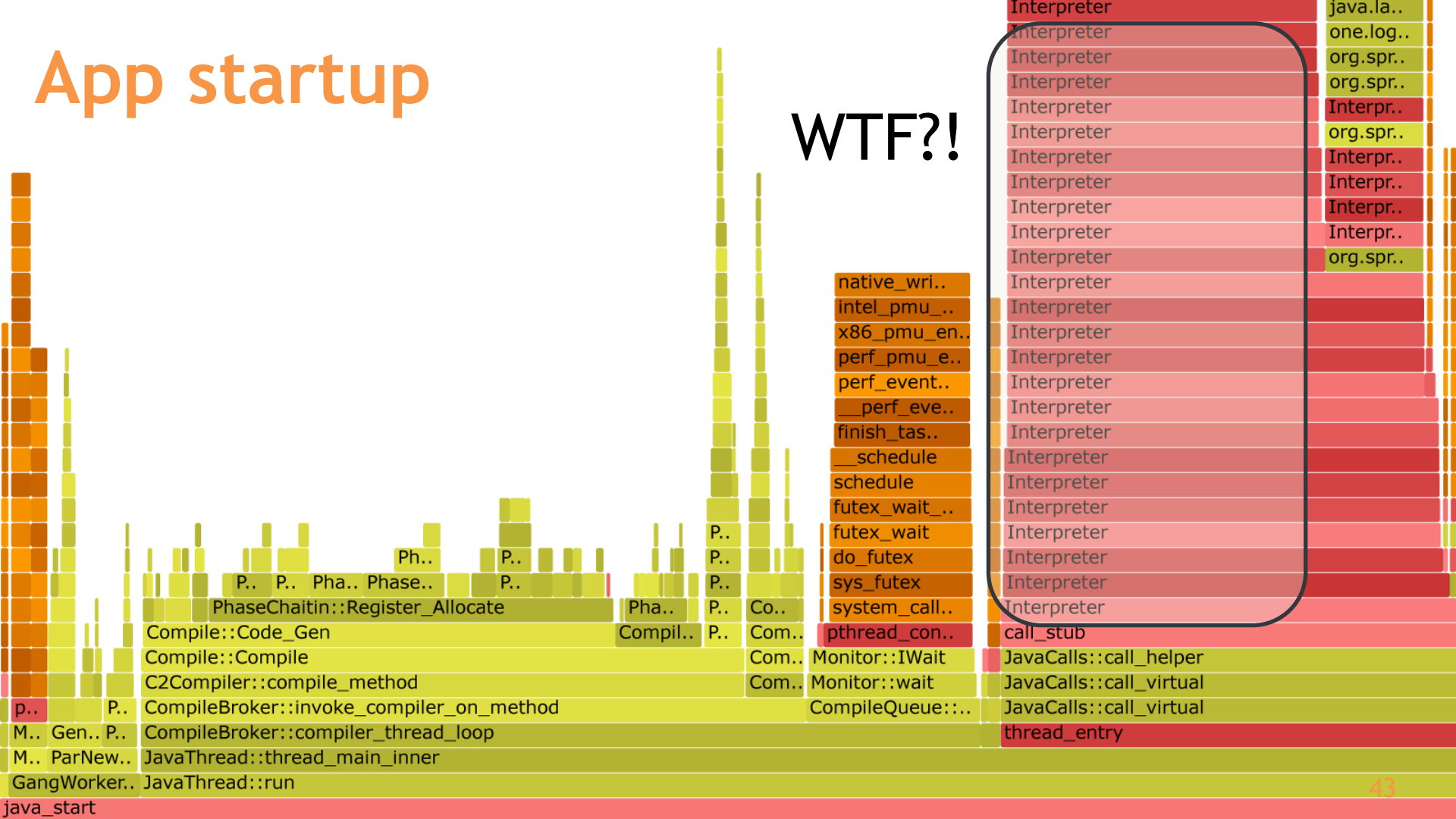
github.com/brendangregg/FlameGraph

App startup

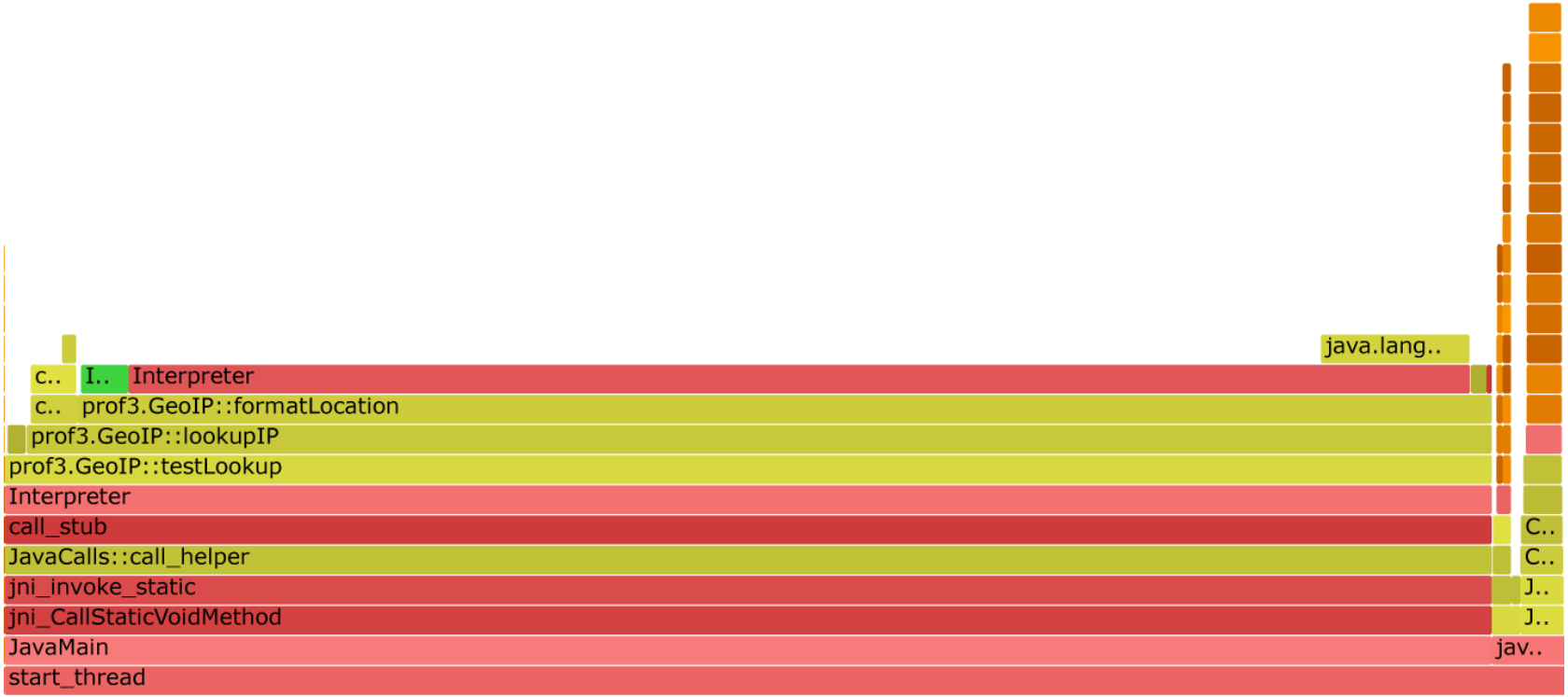

Interpreter
java.la..
one.log..
org.spr..
org.spr..
Interpr..
org.spr..
Interpr..
Interpr..
Interpr..
Interpr..
org.spr..
native_wri..
intel_pmu_..
x86_pmu_en..
perf_pmu_e..
perf_event..
__perf_eve..
finish_tas..
__schedule
schedule
futex_wait_..
futex_wait
do_futex
sys_futex
system_call..
pthread_con..
Monitor::IWait
Monitor::wait
CompileQueue::..
P..
Ph..
Pha..
Phase..
PhaseChaitin::Register_Allocate
Co..
Com..
Compile::Code_Gen
Compil..
Compile::Compile
C2Compiler::compile_method
p..
M..
Gen..
ParNew..
CompileBroker::invoke_compiler_on_method
CompileBroker::compiler_thread_loop
JavaThread::thread_main_inner
GangWorker..
JavaThread::run
java_start
call_stub
JavaCalls::call_helper
JavaCalls::call_virtual
JavaCalls::call_virtual
thread_entry

App startup

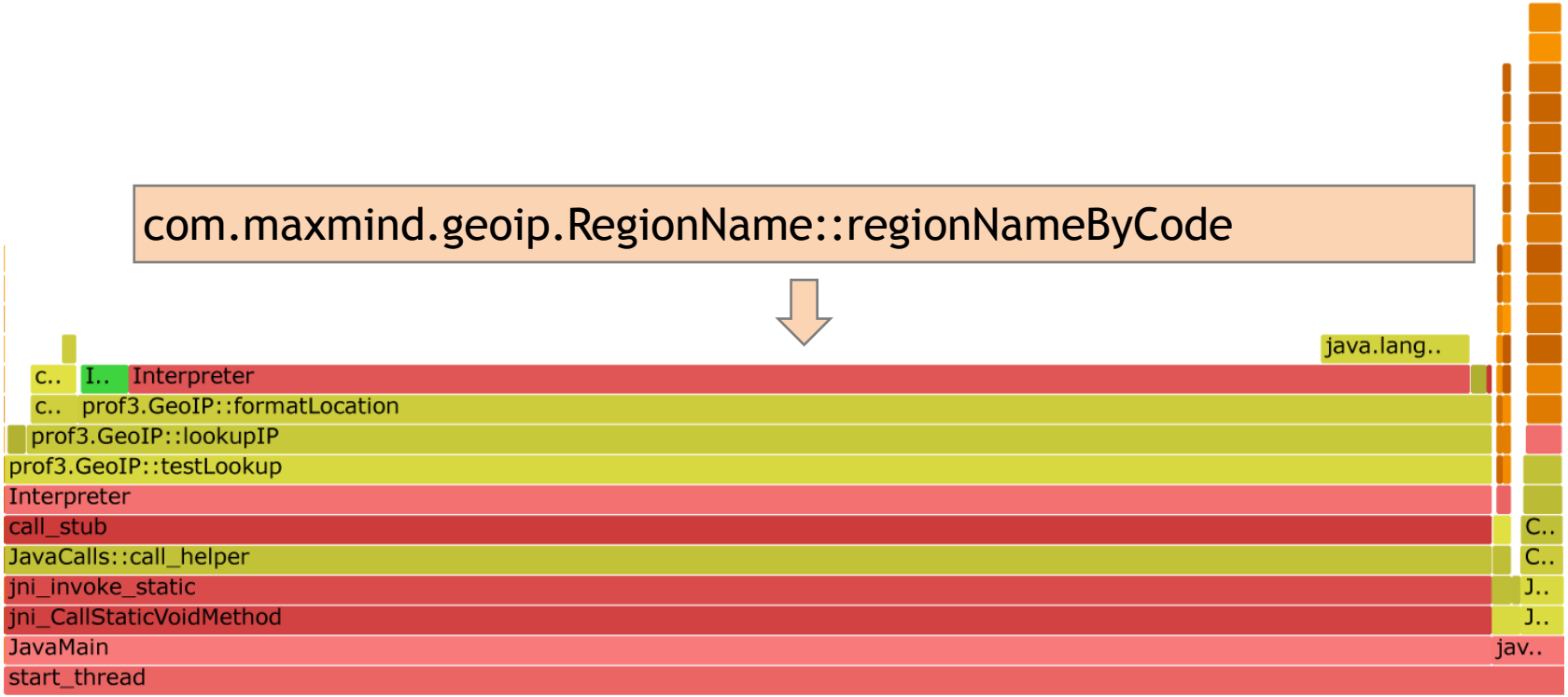
WTF?!



Hot interpreter



Hot interpreter



Poor GeoIP library

```
if (country_code.equals("RU")) {
    switch (region_code) {
        case 1:
            name = "Adygeya";
            break;
        case 2:
            name = "Aginsky Buryatsky AO";
            break;
        case 3:
            name = "Gorno-Altaysk";
            break;
        ...
```

Poor GeoIP library

```
if (country_code.equals("RU")) {
    switch (region_code) {
        case 1:
            name = "Adygeya";
            break;
        case 2:
            name = "Aginsky Buryatsky AO";
            break;
        case 3:
            name = "Gorno-Altaysk";
            break;
        ...
```

-XX:-DontCompileHugeMethods

Disadvantages

Disadvantages

- No interpreted Java

Disadvantages

- No interpreted Java
- -XX:+PreserveFramePointer

Disadvantages

- No interpreted Java
- -XX:+PreserveFramePointer
- Java ≥ 8u60

Disadvantages

- No interpreted Java
- -XX:+PreserveFramePointer
- Java ≥ 8u60
- JIT recompile

Disadvantages

- No interpreted Java
- -XX:+PreserveFramePointer
- Java ≥ 8u60
- JIT recompile
- /proc/sys/kernel/perf_event_paranoid

Disadvantages

- No interpreted Java
- -XX:+PreserveFramePointer
- Java ≥ 8u60
- JIT recompile
- /proc/sys/kernel/perf_event_paranoid
- Limited stack depth

Disadvantages

- No interpreted Java
- -XX:+PreserveFramePointer
- Java ≥ 8u60
- JIT recompile
- /proc/sys/kernel/perf_event_paranoid
- Limited stack depth
- Unstable (many threads)

Disadvantages

- No interpreted Java
- -XX:+PreserveFramePointer
- Java ≥ 8u60
- JIT recompile
- /proc/sys/kernel/perf_event_paranoid
- Limited stack depth
- Unstable (many threads)
- Big data :)

Full-stack Profiler

5

Ideal profiler

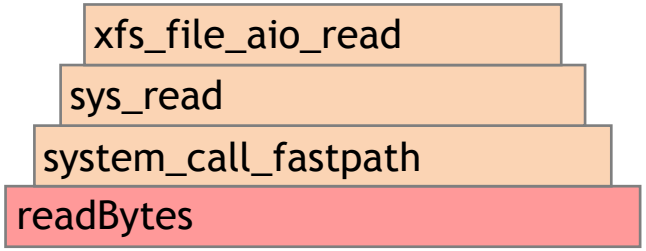
perf_event_open()

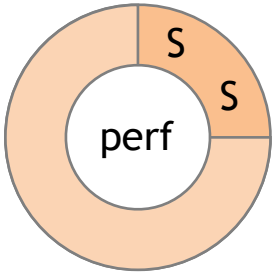
- Kernel + native stacks
- HW counters

- Full Java stack
- Fast and simple

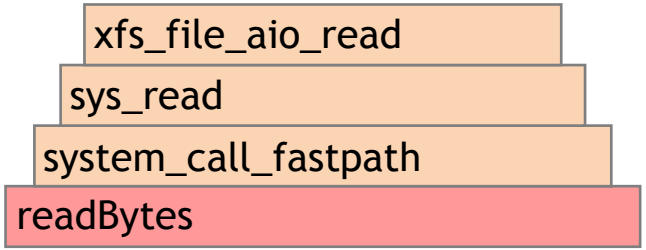
AsyncGetCallTrace()

Put together

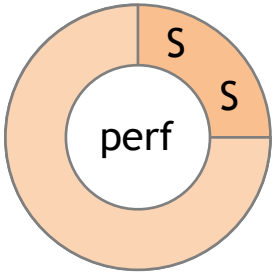


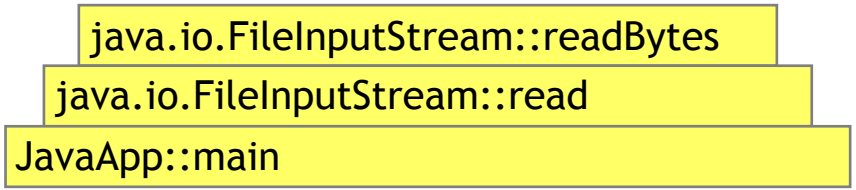


Put together



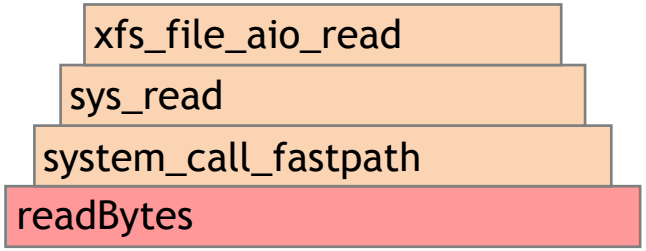
SIGIO



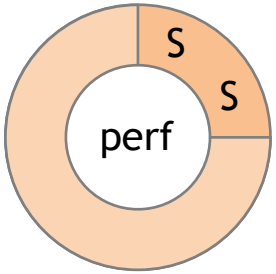


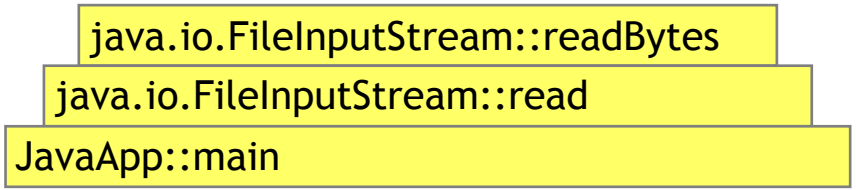
AsyncGetCallTrace

Put together



SIGIO





AsyncGetCallTrace

fcntl(): signal owner = this thread

Issues

- Stack merge point
- Online aggregation
  - Native symbols
- Event-per-thread
  - ulimit -n
  - /proc/sys/kernel/perf_event_mlock_kb
- Concurrency

Case: file reading

```
byte[] buf = new byte[bufSize];

try (FileInputStream in = new FileInputStream(fileName)) {
    int bytesRead;
    while ((bytesRead = in.read(buf)) > 0) {
        ...
    }
}
```

Case: file reading

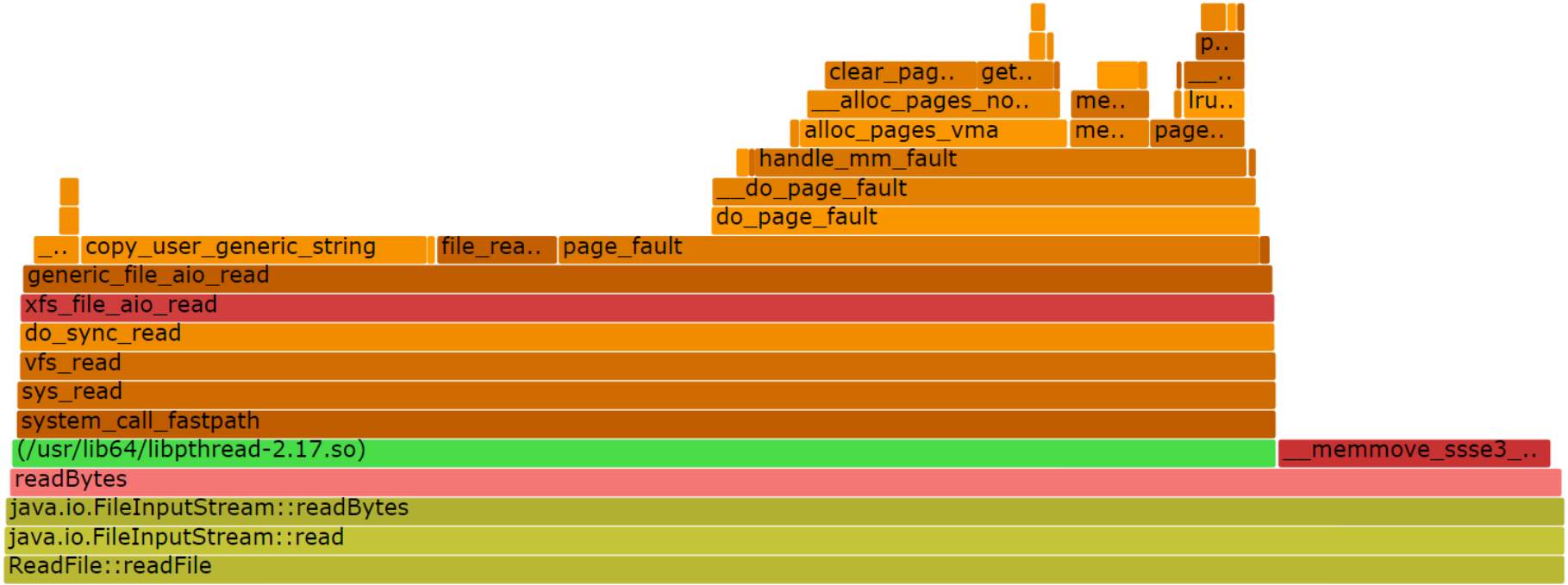
```
byte[] buf = new byte[bufSize];

try (FileInputStream in = new FileInputStream(fileName)) {
    int bytesRead;
    while ((bytesRead = in.read(buf)) > 0) {
        ...
    }
}
```

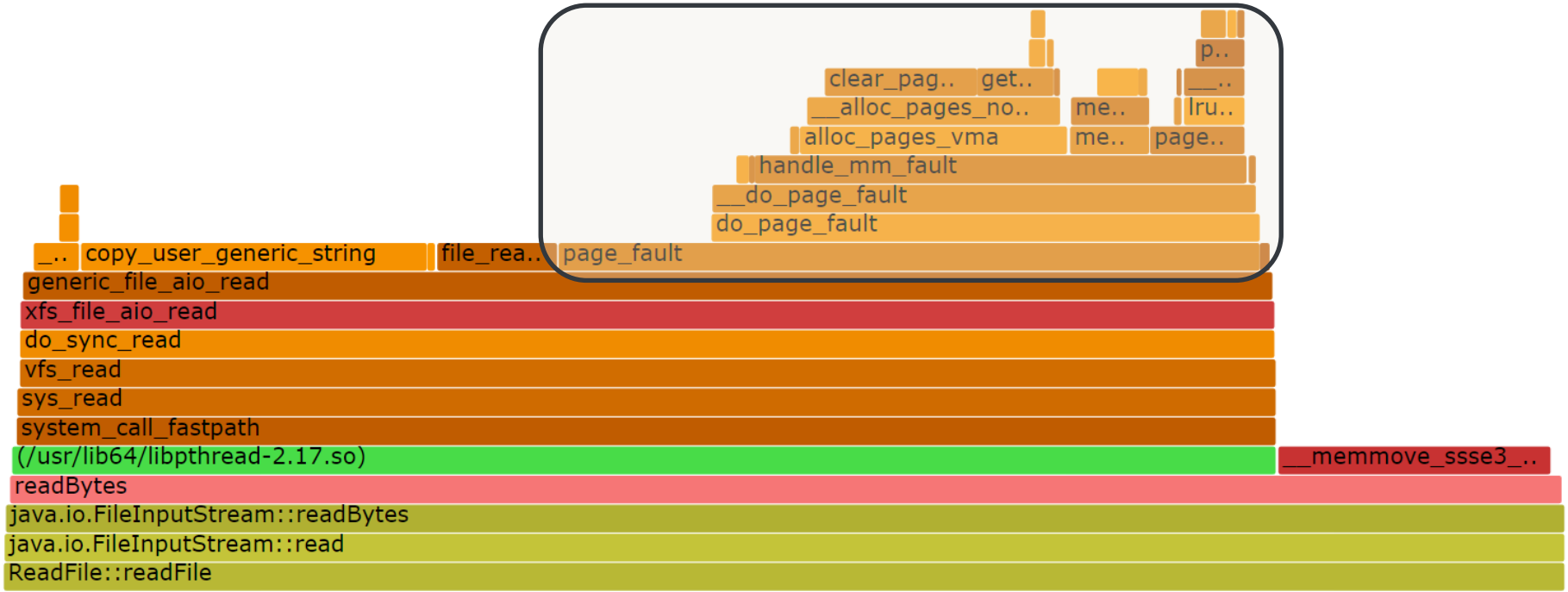
Buffer size?

- 8 K
- 64 K
- 250 K
- 1 M
- 4 M

Full-stack profile

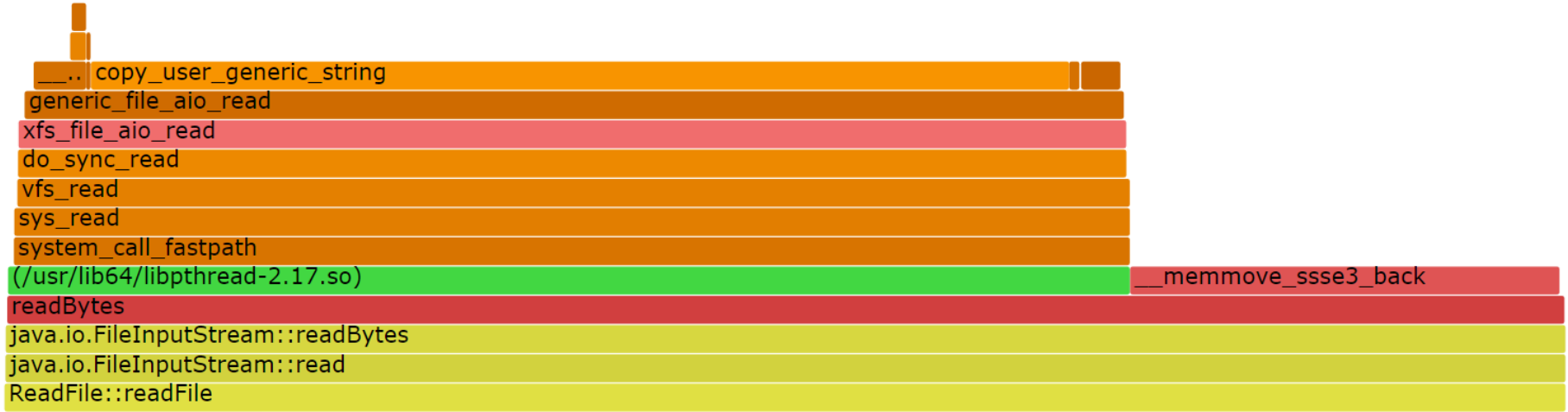


Full-stack profile

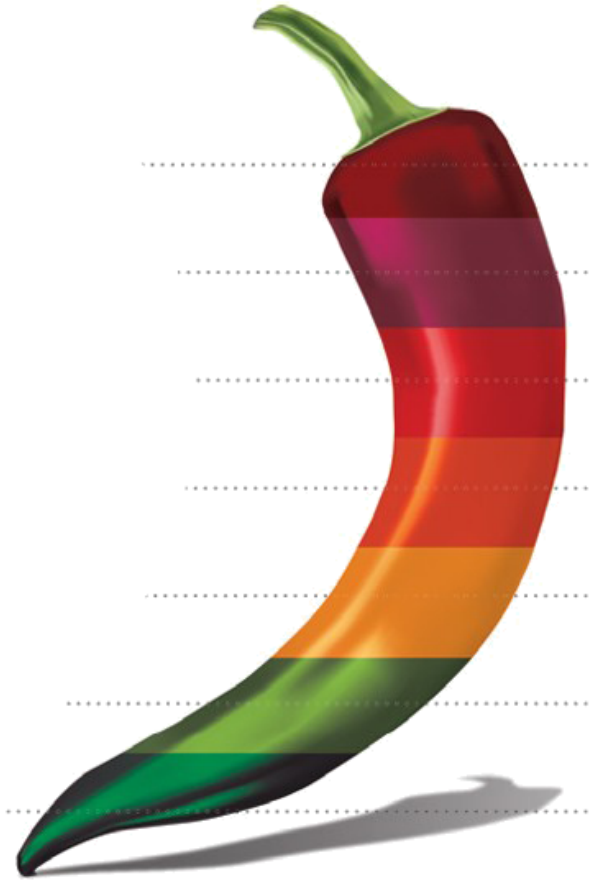


Full-stack profile

Read buffer: 260K => 250K



	AsyncGCT	Perf	Full-stack Profiler
Java stack	Yes	No interpreted	Yes
Native stack	No	Yes	Yes
Kernel stack	No	Yes	Yes
JDK version	6+	8u60+	6+
Idle overhead	0	2-5%	0
Online aggregation	Yes	No	Yes
Stable	Yes	No	Yes



Future improvements

Full-stack profiler

Perf

AsyncGetCallTrace

Thread dump

CPU utilisation?

Performance problem?

Try it

- github.com/apangin/async-profiler
- Contributions are welcome!

Contacts

Andrei Pangin
@AndreiPangin

Vadim Tsesko
@incubos

https://v.ok.ru/vacancies.html


одноклассники