Hunting performance monsters

On the back of a Camel

Otavio R. Piske <orpiske@apache.org>

About me

Otavio R. Piske

- Principal Software Engineer @ Red Hat
- Committer + PMC @ Apache Camel
- Twitter: @otavio021

Agenda

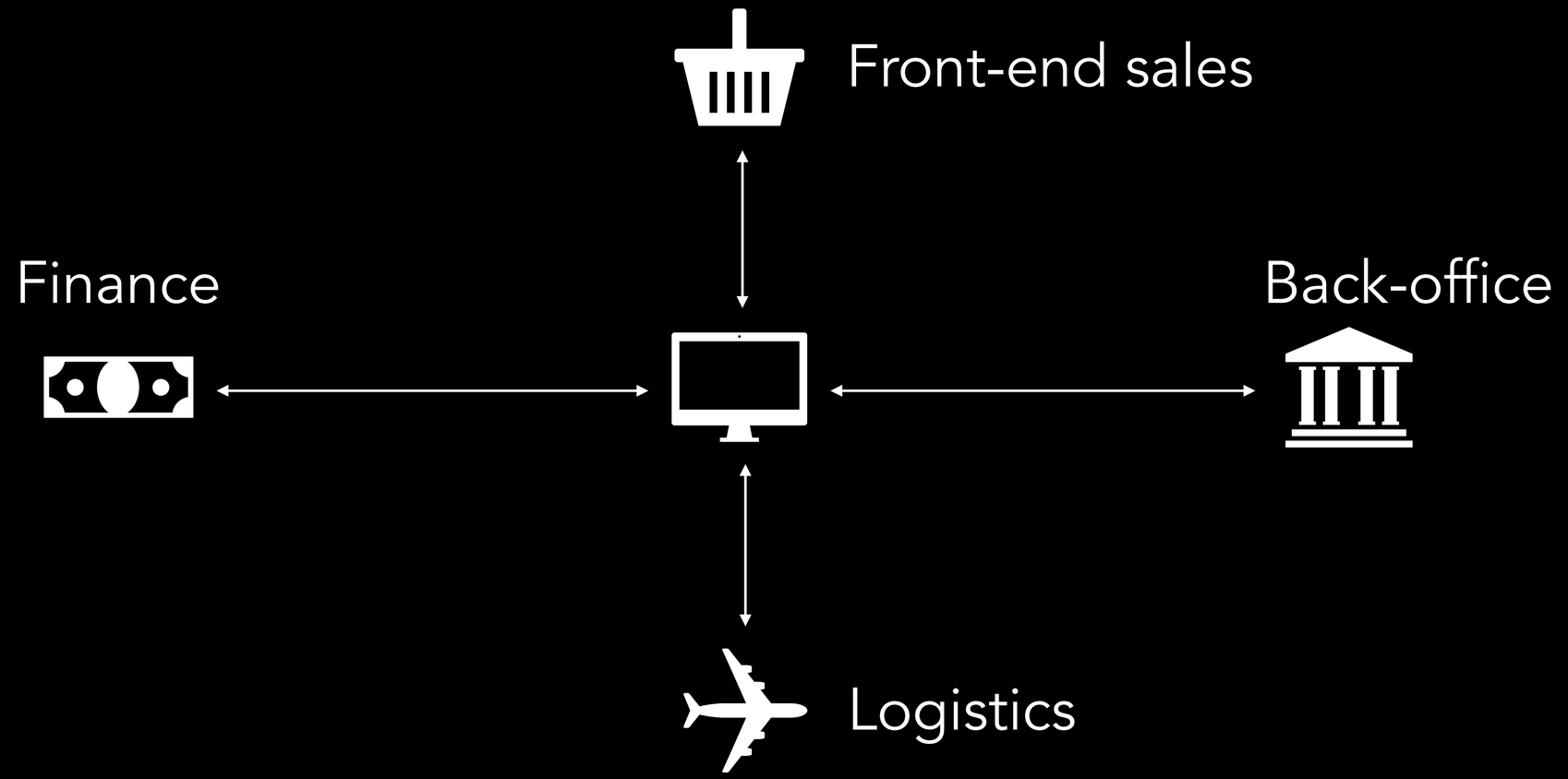
- Overview
- The weapons
- Monster hunting
- Collecting the rewards
- Q&A

Disclaimer

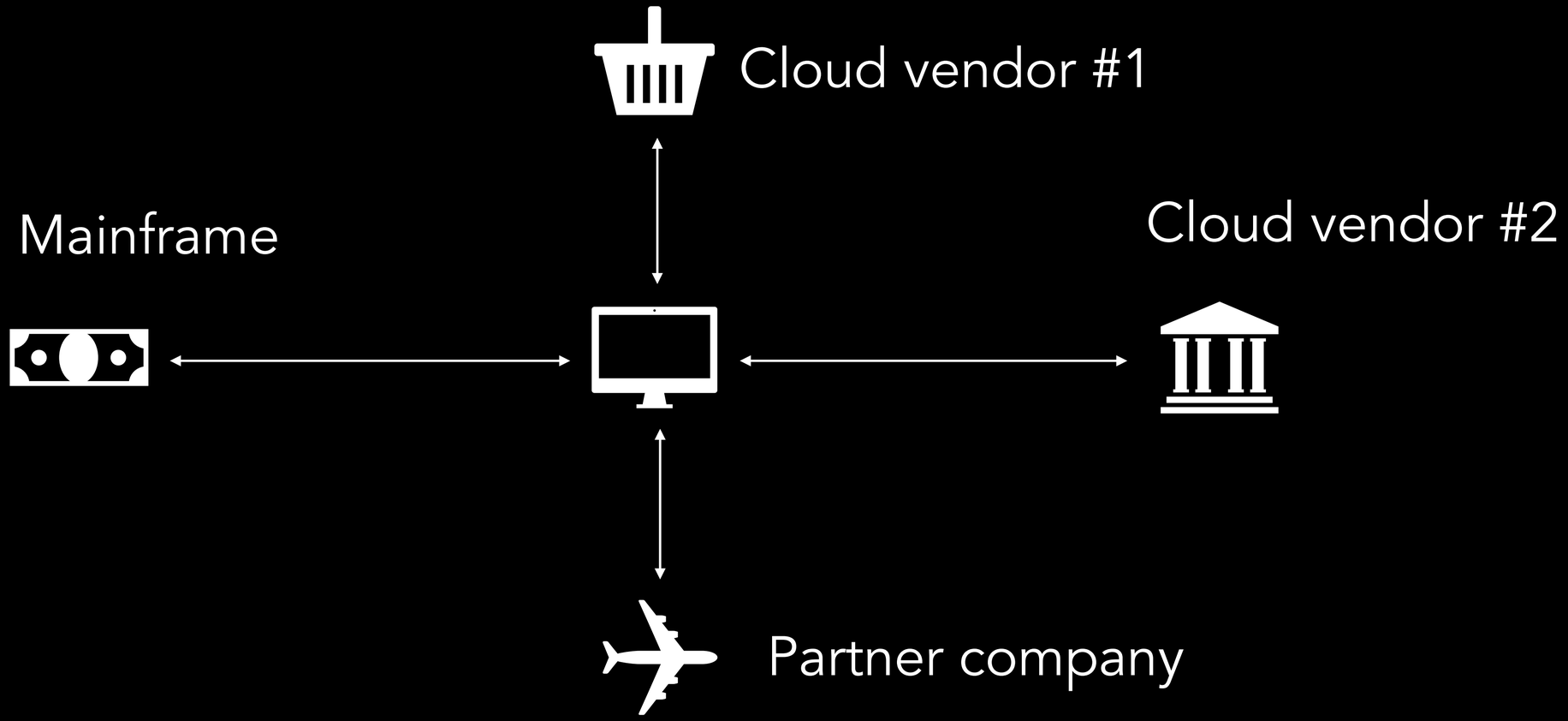
No monsters were harmed during the development of Apache Camel 4

What is Apache Camel?

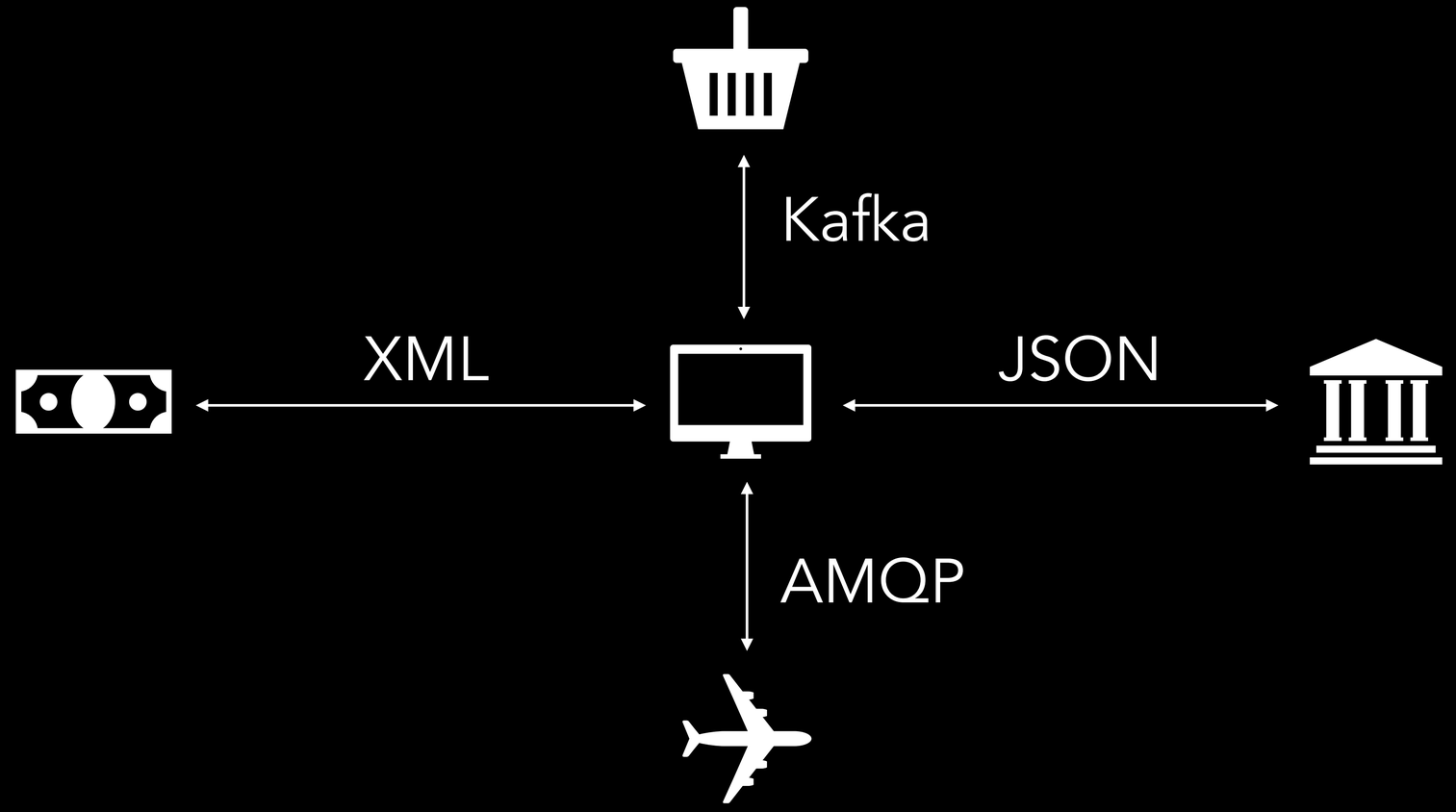
Business architecture



Enterprise architecture



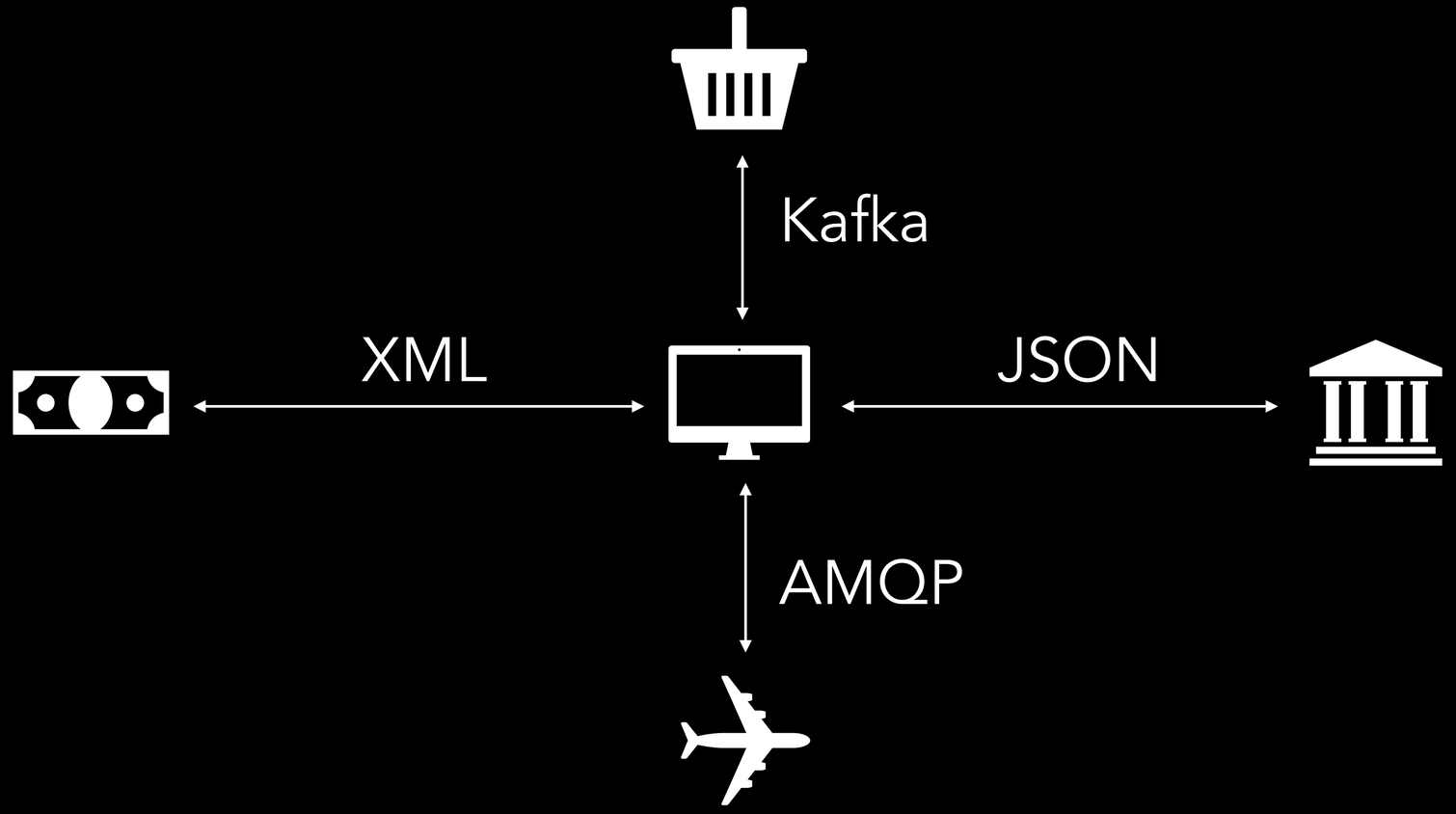
System architecture



Exchanging data is hard.

Apache Camel simplifies it.

Apache Camel



Apache Camel

290+ Components

AMQP

Azure

AWS (S3, SQS, SNS, etc)

JMS

Kafka

Languages

EIP Patterns

40+ Dataformats

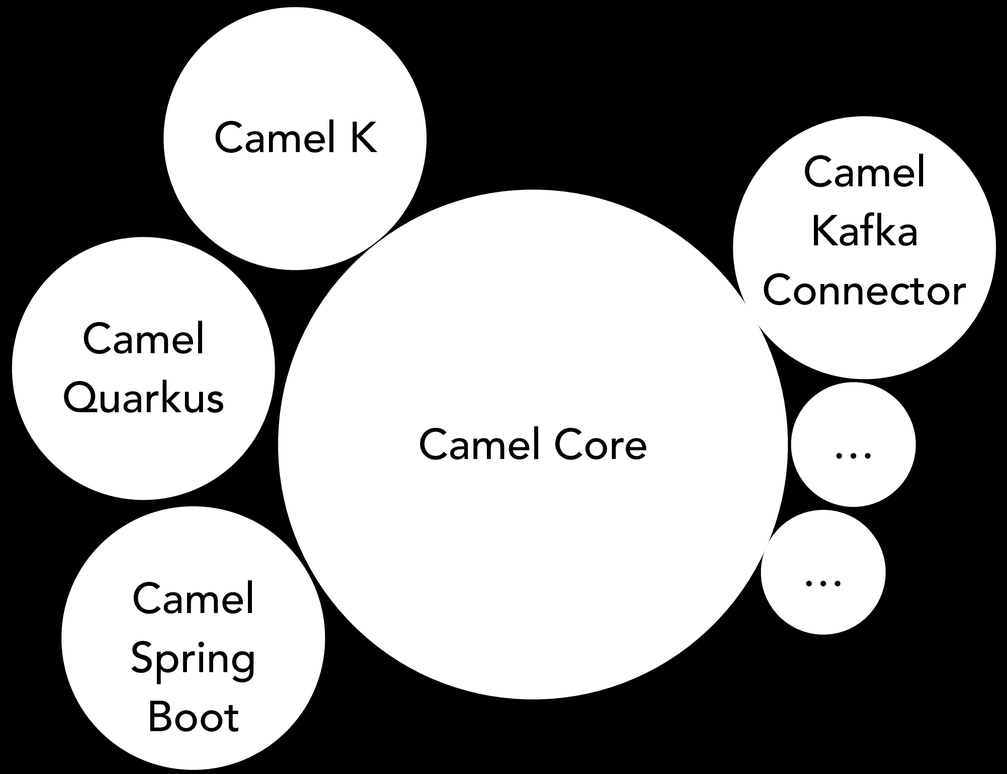
XML

JSON

HL7

Protobuf

Overview


Camel K
Camel Kafka Connector
Camel Quarkus
Camel Core
...
...
Camel Spring Boot

Overview

Simple Camel Route

```
public class SomeRoute extends RouteBuilder {
  @Override
  public void configure() {
      from("kafka:sales")
          .to(“jms:queue:logistics-processing");
  }
}
```

“Technology disciplines tend to be objective … Performance, on the other hand, is often subjective … it can be unclear whether there is an issue to begin with, and if so, when it has been fixed”

Brendan Gregg - Systems Performance

A computer that is wasting time with inefficiencies is still consuming resources.

Waste leads to more waste.


The weapons

Type Pollution Agent

- Java Agent
- Developed by Red Hat AS Performance Team
  - Open Source
- Identify code that may suffer from type-pollution problems

Async Profiler

- Open source low-overhead sampling profiler
- Works with many JVM distributions
- Can trace different types of events
  - CPU: cpu
  - Allocations: heap
  - Wall-clock: time

Perf

- Official Linux profiler
- Has several tools
  - stat: basic statistics about the runtime
  - c2c: cache statistics / false-sharing detection
- Used as a complement

JMH

- Microbenchmark
- Used interchangeably with other tools
  - Test hypothesis
- Different benchmark modes
  - Throughput
  - Average time
- Open source with lots of samples

Camel Load Generator

- Macro-benchmark tool
- Long running tests
- Personal project
- Designed to stress specific parts of Camel
- Used along with other tools

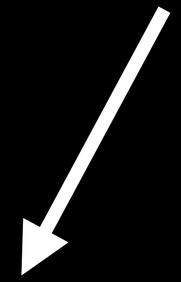
Monster Hunting


Camel Load Tester + Type Pollution Agent

Trail #1

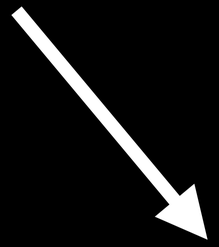
```
-javaagent:$AGENT_HOME/type-pollution-agent-0.1-SNAPSHOT.jar=org.apache.camel
```

Trail #1



```
-javaagent:$AGENT_HOME/type-pollution-agent-0.1-SNAPSHOT.jar=org.apache.camel
```

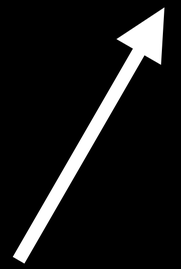
Trail #1



```
-javaagent:$AGENT_HOME/type-pollution-agent-0.1-SNAPSHOT.jar=org.apache.camel
```

Trail #1

```
-javaagent:$AGENT_HOME/type-pollution-agent-0.1-SNAPSHOT.jar=org.apache.camel
```



Trail #1

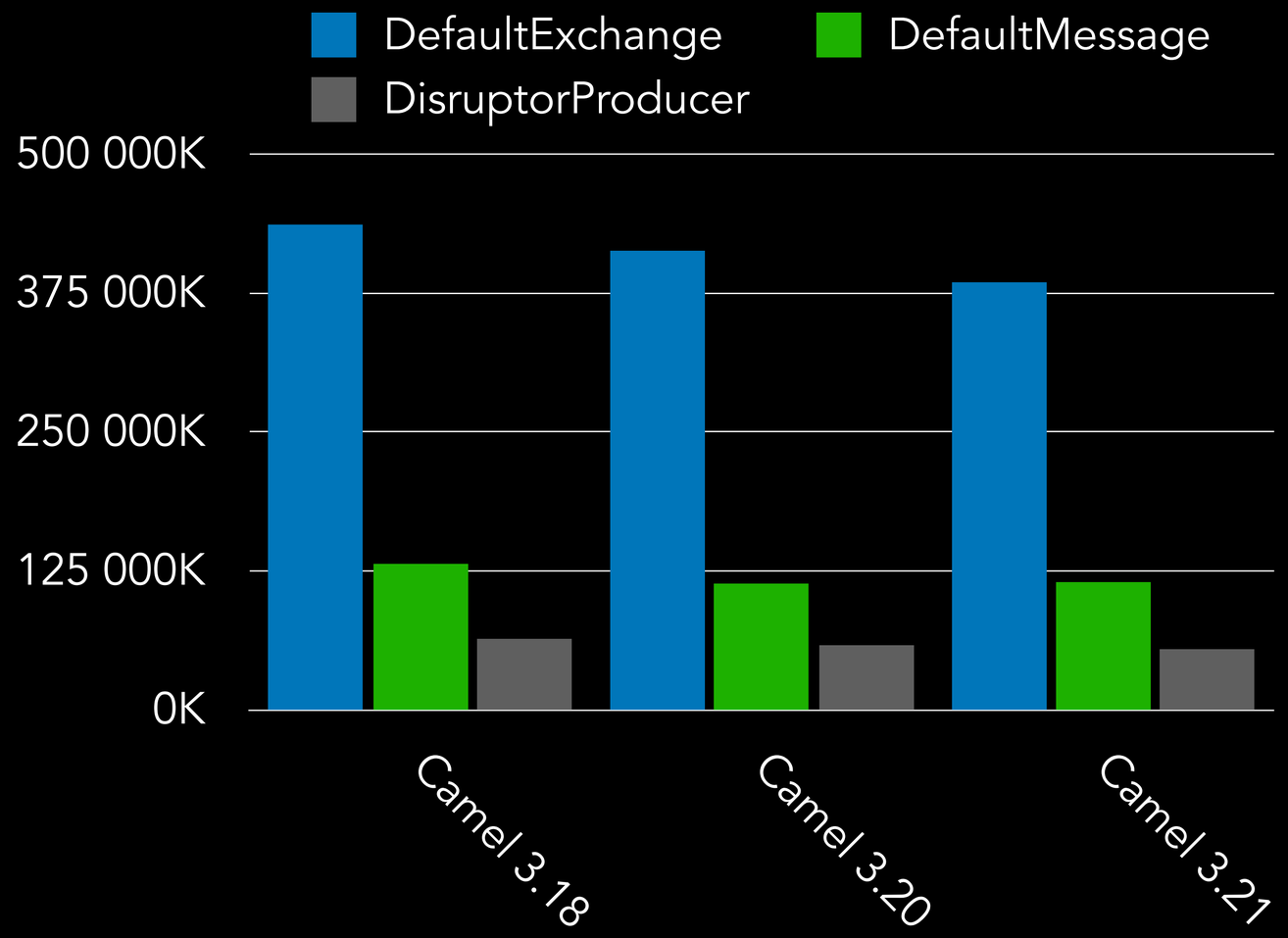
Type-pollution agent

```
-------------------------
Type Check Statistics:
-------------------------
Date:          2023/09/29 14:15:22
Last:          true
-------------------------
Type Pollution:
-------------------------
1:             org.apache.camel.support.DefaultExchange
Count:         436838067
Types:
               org.apache.camel.ExtendedExchange
               org.apache.camel.Exchange
Traces:
               org.apache.camel.support.AbstractExchange.adapt(AbstractExchange.java:607)
                              class: org.apache.camel.Exchange
                              count: 142594327
                              class: org.apache.camel.ExtendedExchange
                              count: 42925534
               org.apache.camel.support.ExchangeHelper.copyExchangeAndSetCamelContext(ExchangeHelper.java:826)
                              class: org.apache.camel.ExtendedExchange
                              count: 19744262
               org.apache.camel.component.disruptor.DisruptorConsumer.process(DisruptorConsumer.java:167)
                              class: org.apache.camel.ExtendedExchange
                              count: 19188788
```

Trail #1

Type-pollution agent

- 370+ million successful type checks for the DefaultExchange
- 114+ million for the DefaultMessage
- 50+ million for the DisruptorProducer

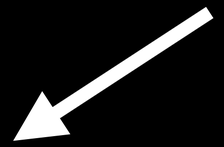



Camel Load Tester + Perf stat

Trail #2

```
perf stat -D 5000 java ${LOTS_OF_OPTIONS} -jar your-app.jar
```

Trail #2



```
perf stat -D 5000 java ${LOTS_OF_OPTIONS} -jar your-app.jar
```

Trail #2

```
perf stat -D 5000 java ${LOTS_OF_OPTIONS} -jar your-app.jar
```

Trail #2

```
perf stat -D 5000 java ${LOTS_OF_OPTIONS} -jar your-app.jar
```

Trail #2

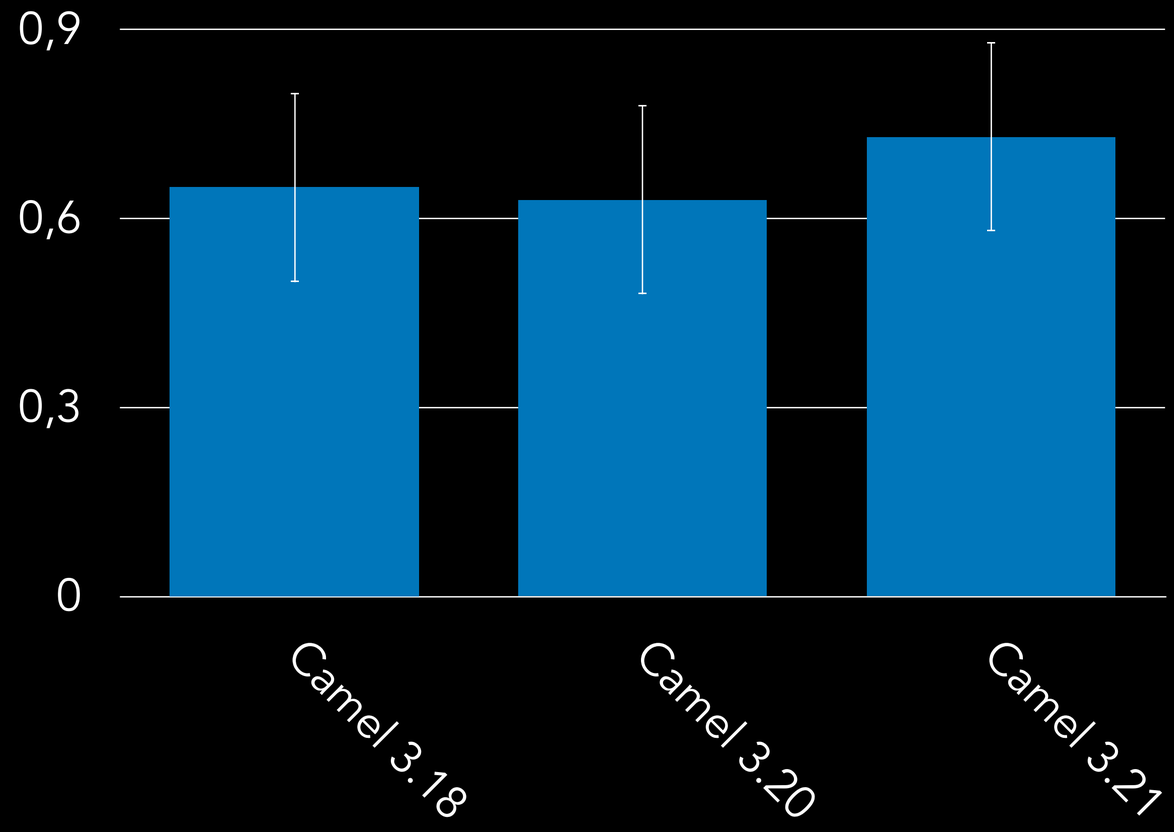
Perf stat

```
 Performance counter stats for process id '716275':

         193055.40 msec task-clock                       #    6.428 CPUs utilized
           1737641      context-switches                 #    9.001 K/sec
                97      cpu-migrations                   #    0.502 /sec
             18503      page-faults                      #   95.843 /sec
      457194305145      cycles                           #    2.368 GHz
      296078900393      instructions                     #    0.65  insn per cycle
       54848711045      branches                         #  284.109 M/sec
        1190686954      branch-misses                    #    2.17% of all branches

      30.032294196 seconds time elapsed
```

Trail #2

Perf stat

- Instructions per cycle (IPC) < 1



This is waste.

Type check scalability issue

JDK-8180450

- Relates to a performance penalty for type checks
- Has been in the JVM for decades
- A fix is in progress (hopefully for Java 22)

Type check scalability issue

JDK-8180450

- instanceof
- Methods:
  - Class::isInstance
  - Class::cast
  - Class::isAssignableFrom
- checkcast
  - i.e.; generic type erasure

Given the following code

Type check scalability issue

JDK-8180450

```
public class SomeType implements T1, T2, T3 {
}
```

Consider the question: “is some variable a type Tx”

Type check scalability issue

JDK-8180450

```
if (someVar instanceof T2) {
    System.out.println("This is T2");
}

if (someVar instanceof T1) {
    System.out.println("This is T1");
}
```

Type check scalability issue

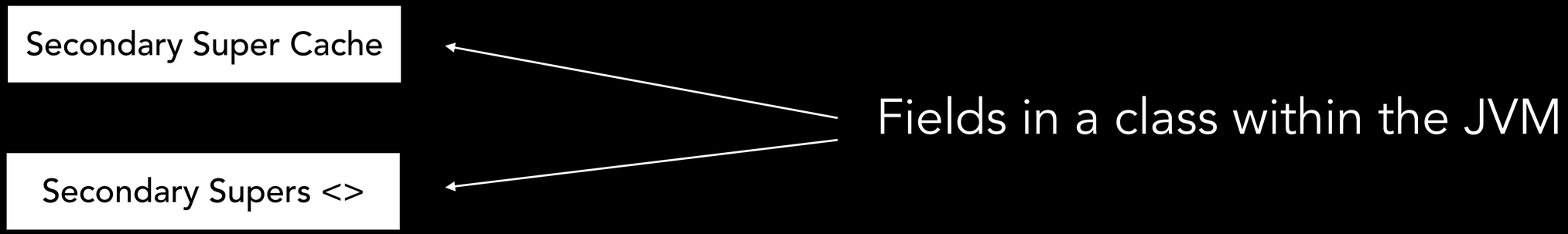
JDK-8180450

Secondary Super Cache

Secondary Supers <>

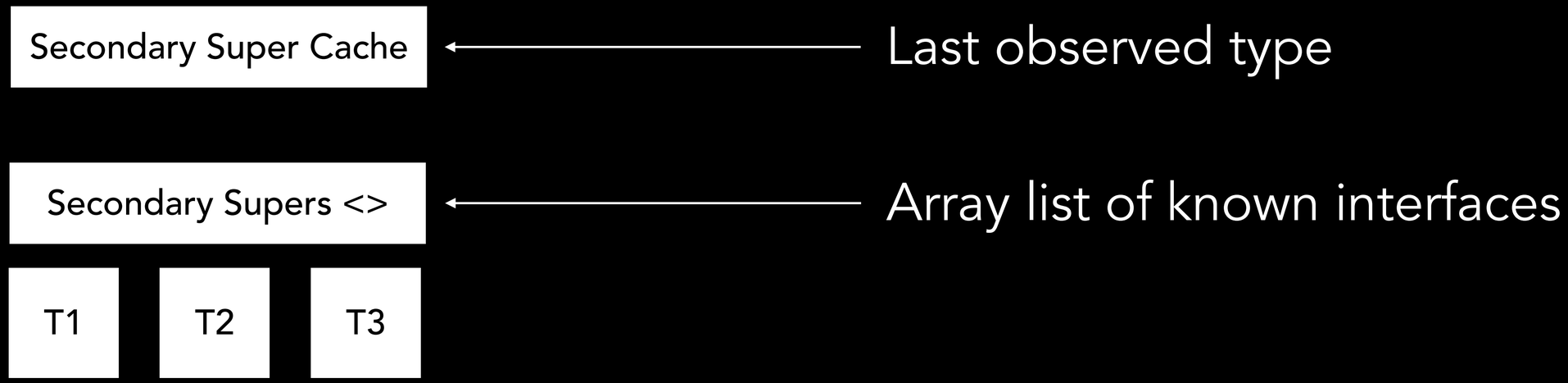
Type check scalability issue

JDK-8180450



Type check scalability issue

JDK-8180450



Type check scalability issue

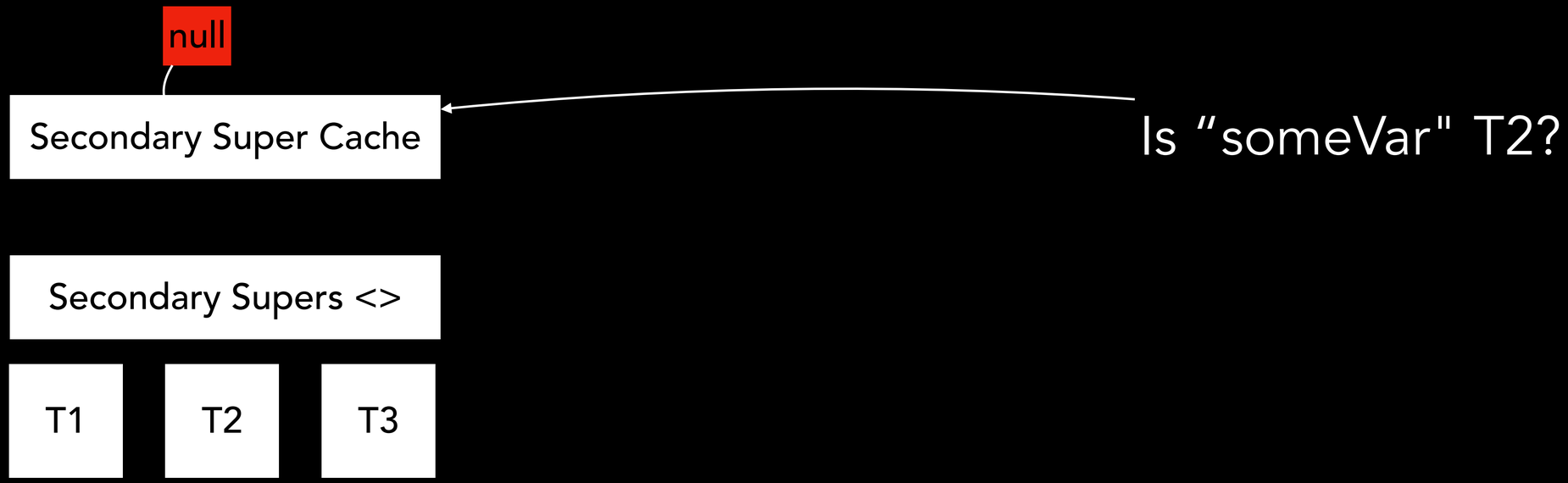
JDK-8180450

Is “someVar" T2?

```
if (someVar instanceof T2) {
    System.out.println("This is T2");
}

if (someVar instanceof T1) {
    System.out.println("This is T1”);
}
```

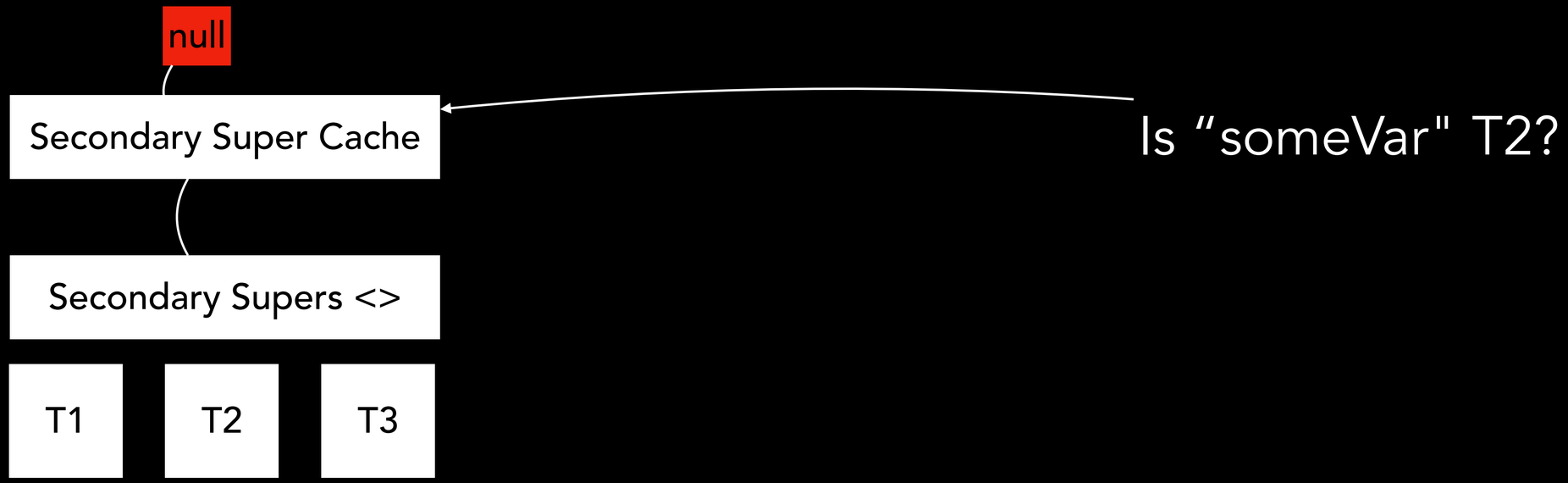
Type check scalability issue

JDK-8180450



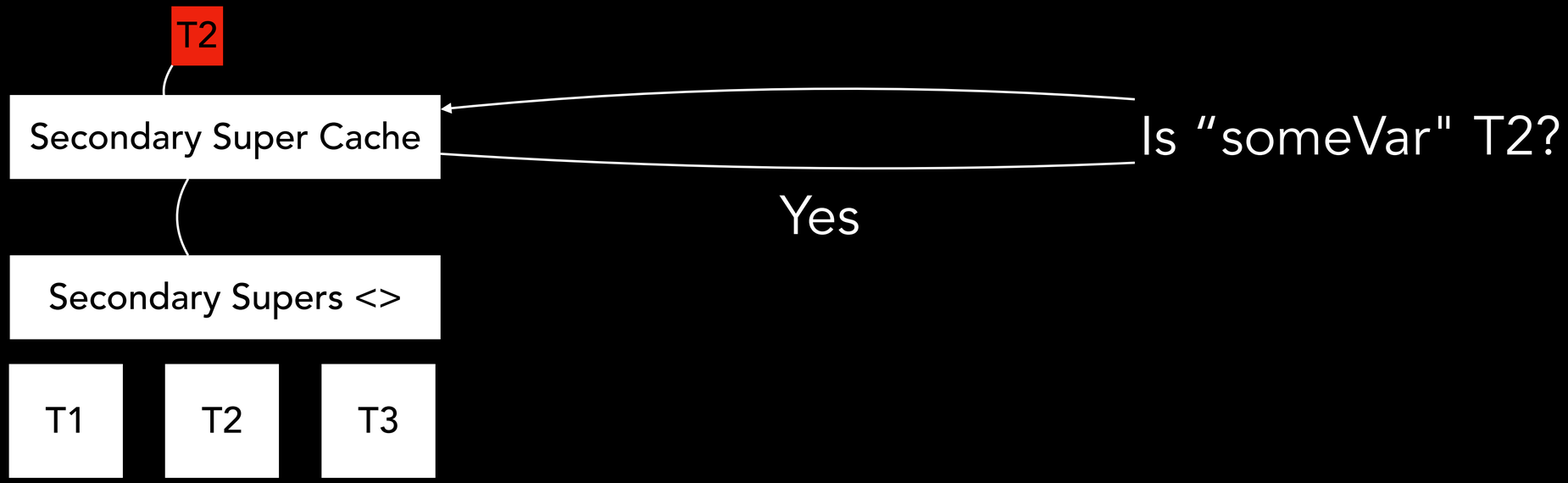
Type check scalability issue

JDK-8180450



Type check scalability issue

JDK-8180450



Type check scalability issue

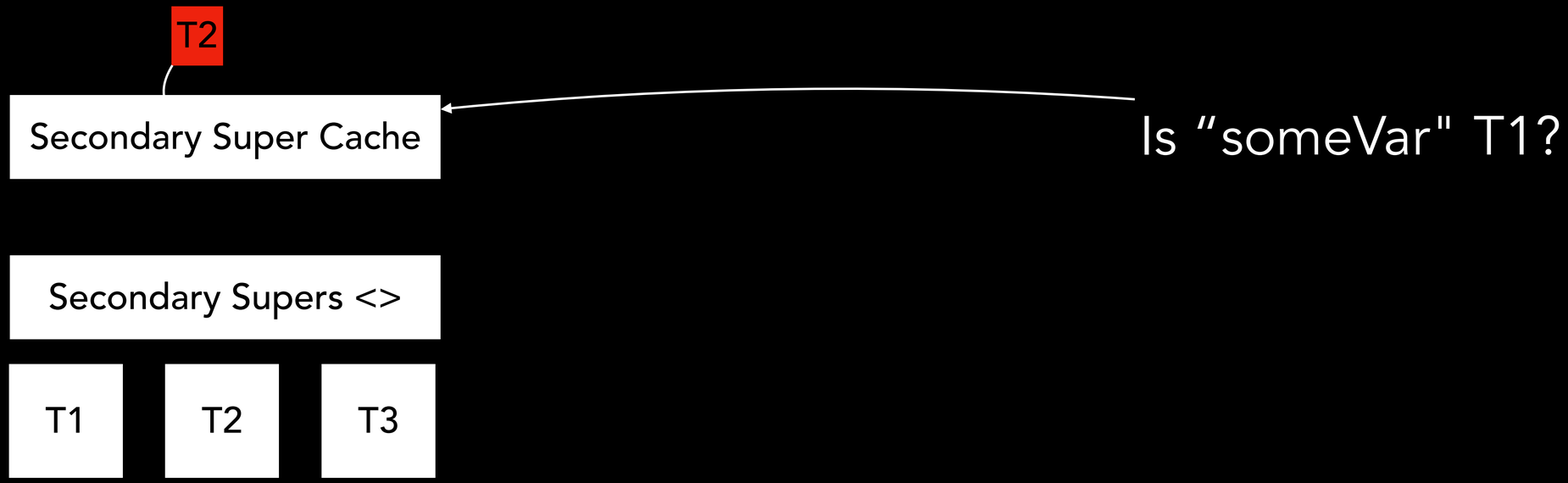
JDK-8180450

```
if (someVar instanceof T2) {
    System.out.println("This is T2");
}

if (someVar instanceof T1) {
    System.out.println("This is T1");
}
```

Is “someVar" T1?

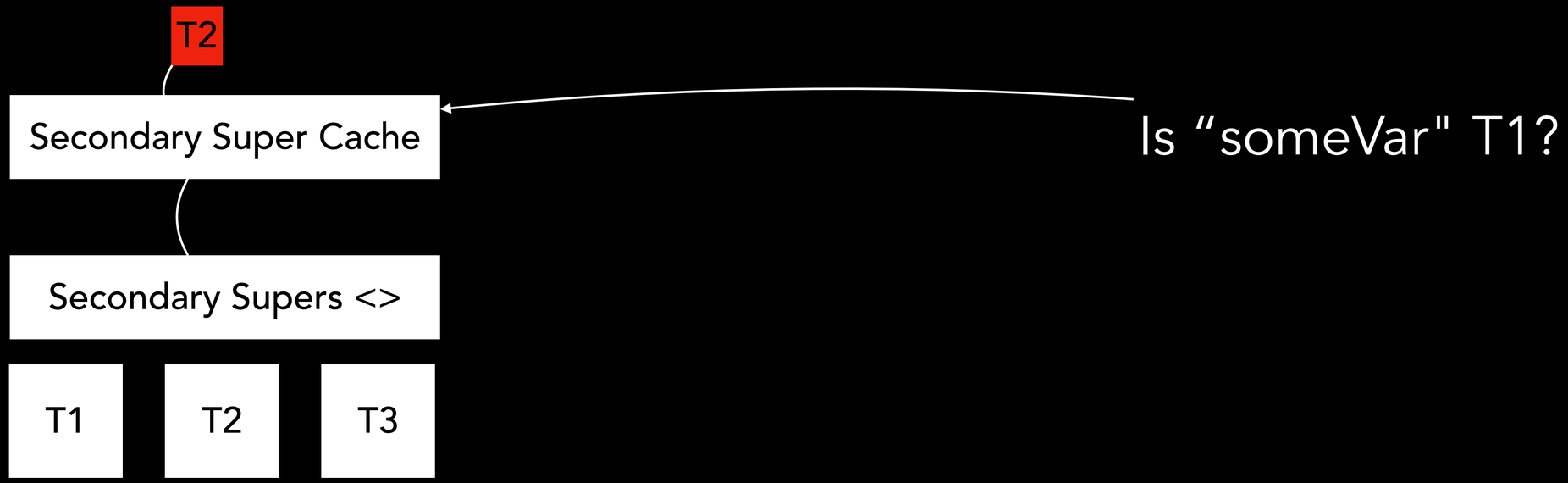
Type check scalability issue

JDK-8180450



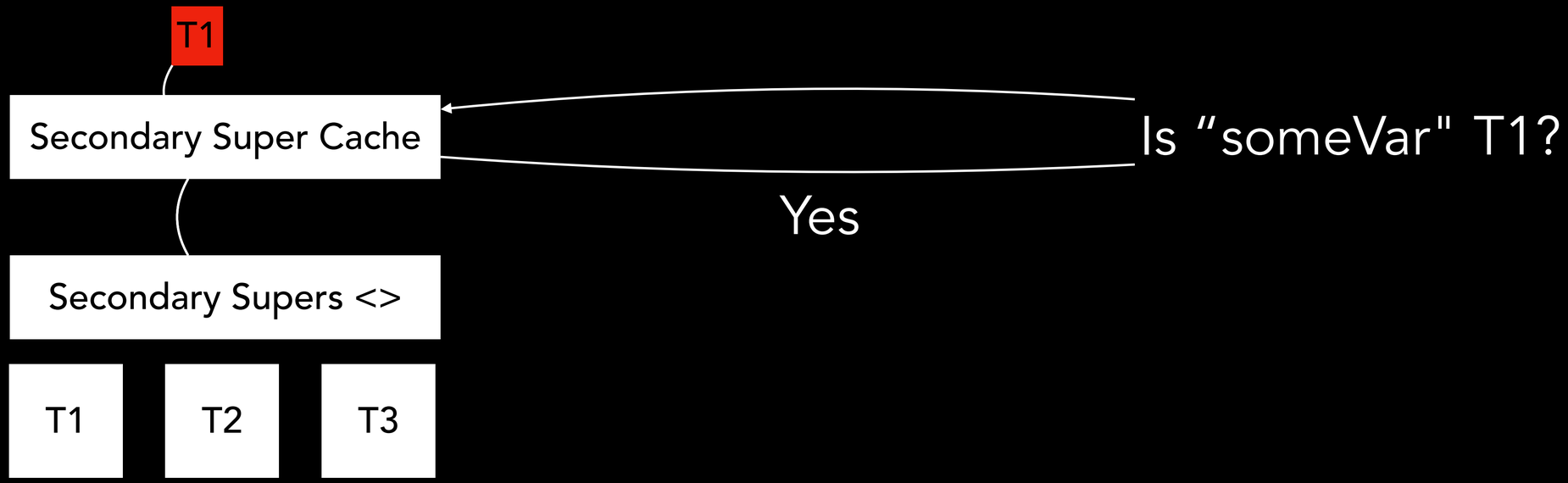
Type check scalability issue

JDK-8180450



Type check scalability issue

JDK-8180450



Why this is bad?

Type check scalability issue

JDK-8180450

- Ping pong of the cache state
- May result in an invalidation of the cache line
- Maintaining cache coherency can be costly
- Multiple threads can make it worse

Learn more

JDK-8180450

- Talks:
  - Cracking the scalability wall (Java Zone 2023)
  - Cracking the scalability wall (Devoxx UK)
- Blog posts
  - Seeing through the hardware counters (Netflix blog)

How did it look like?

Type check scalability issue

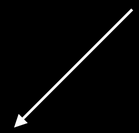
JDK-8180450

```
@Override
public <T extends CamelContext> T adapt(Class<T> type) {
    return type.cast(this);
}
```

Type check scalability issue

JDK-8180450

Access to restricted APIs



```
@Override
public <T extends CamelContext> T adapt(Class<T> type) {
    return type.cast(this);
}
```

It was pattern …

CAMEL-15105

Create an uniform interface for plugins

- Create an uniform interface
- Simplify access to plugins
- Simplify access to restricted operations
  - CamelContext
  - Exchange

Defeating the type check scalability monster

JDK-8180450

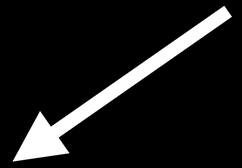
```
public static PeriodTaskResolver getPeriodTaskResolver(CamelContext camelContext) {
    return getPeriodTaskResolver(camelContext.getCamelContextExtension());
}
```


Camel Load Tester + Async Profiler

Trail #3

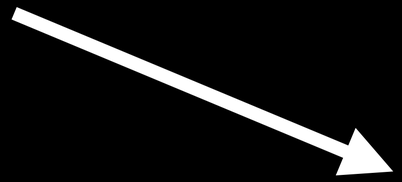
```
-agentpath:/path/to/libasyncProfiler.so=start,ann,threads,event=cpu,file=report.html
```

Trail #3



```
-agentpath:/path/to/libasyncProfiler.so=start,ann,threads,event=cpu,file=report.html
```

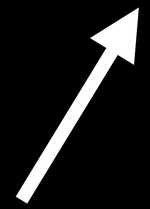
Trail #3



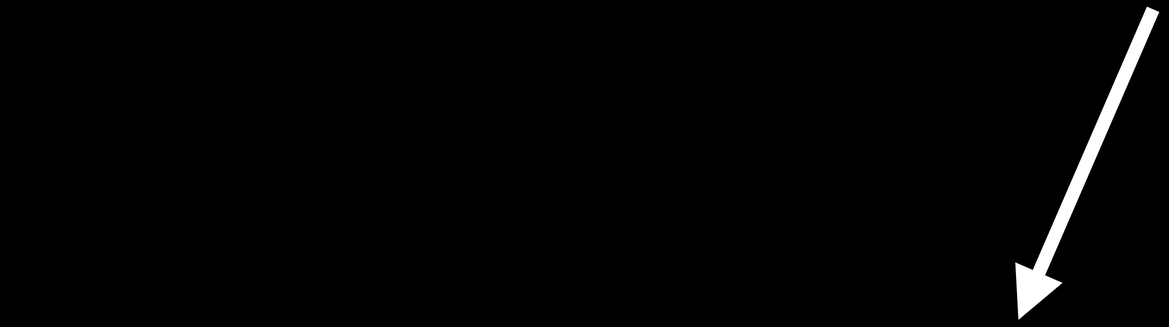
```
-agentpath:/path/to/libasyncProfiler.so=start,ann,threads,event=cpu,file=report.html
```

Trail #3

```
-agentpath:/path/to/libasyncProfiler.so=start,ann,threads,event=cpu,file=report.html
```



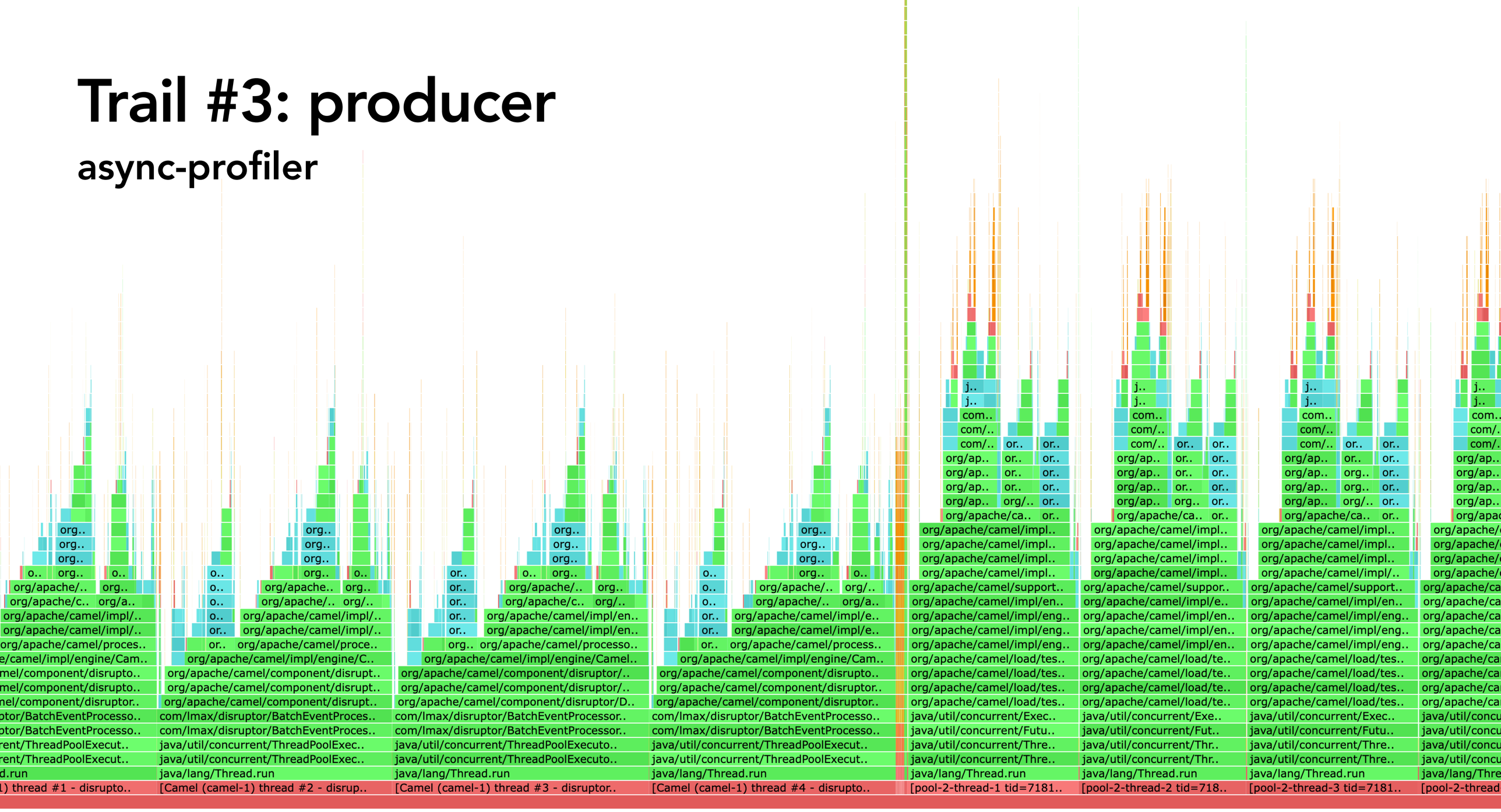
Trail #3



```
-agentpath:/path/to/libasyncProfiler.so=start,ann,threads,event=cpu,file=report.html
```

Trail #3: producer

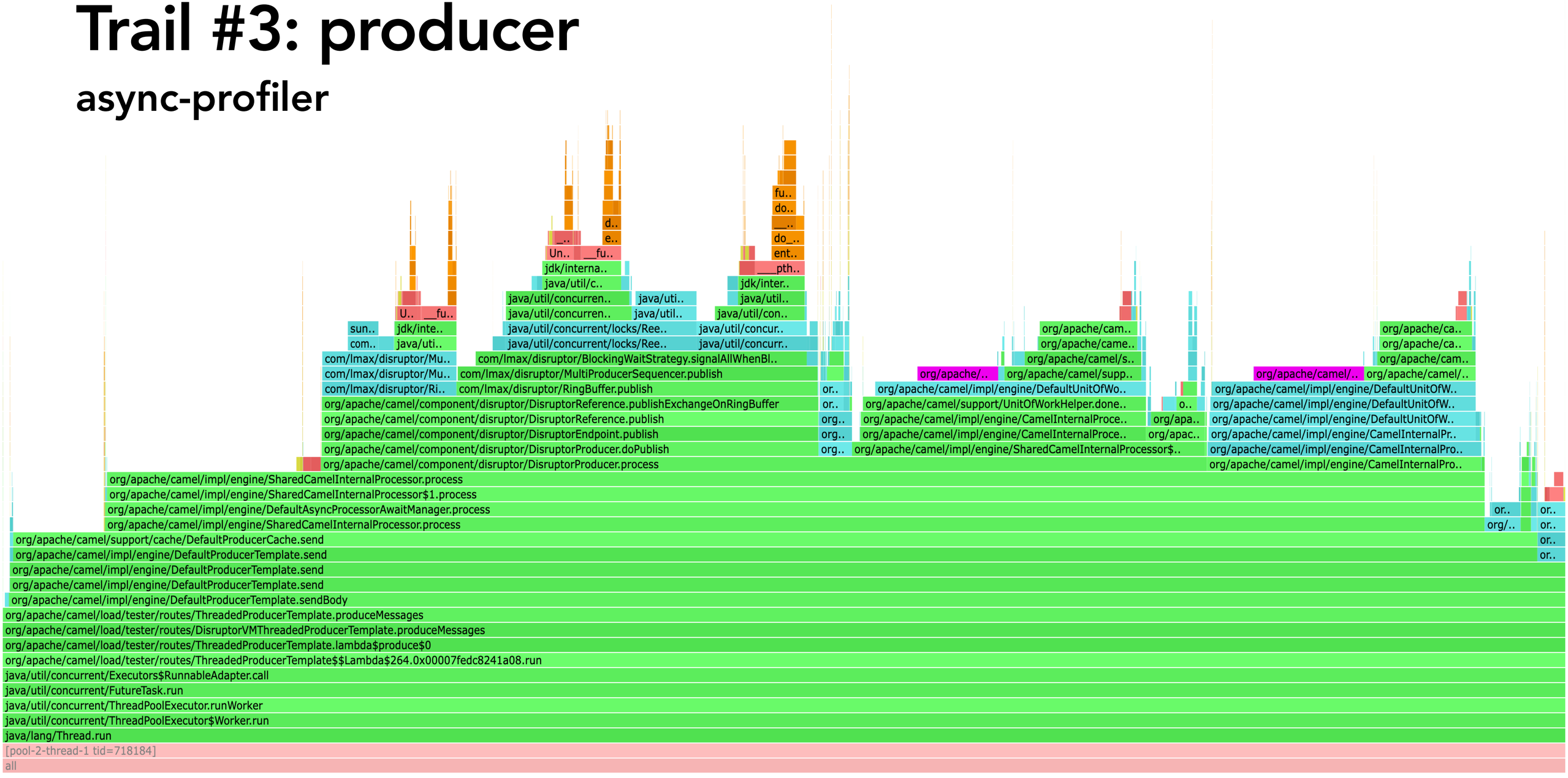
async-profiler



The plot thickens

Trail #3: producer

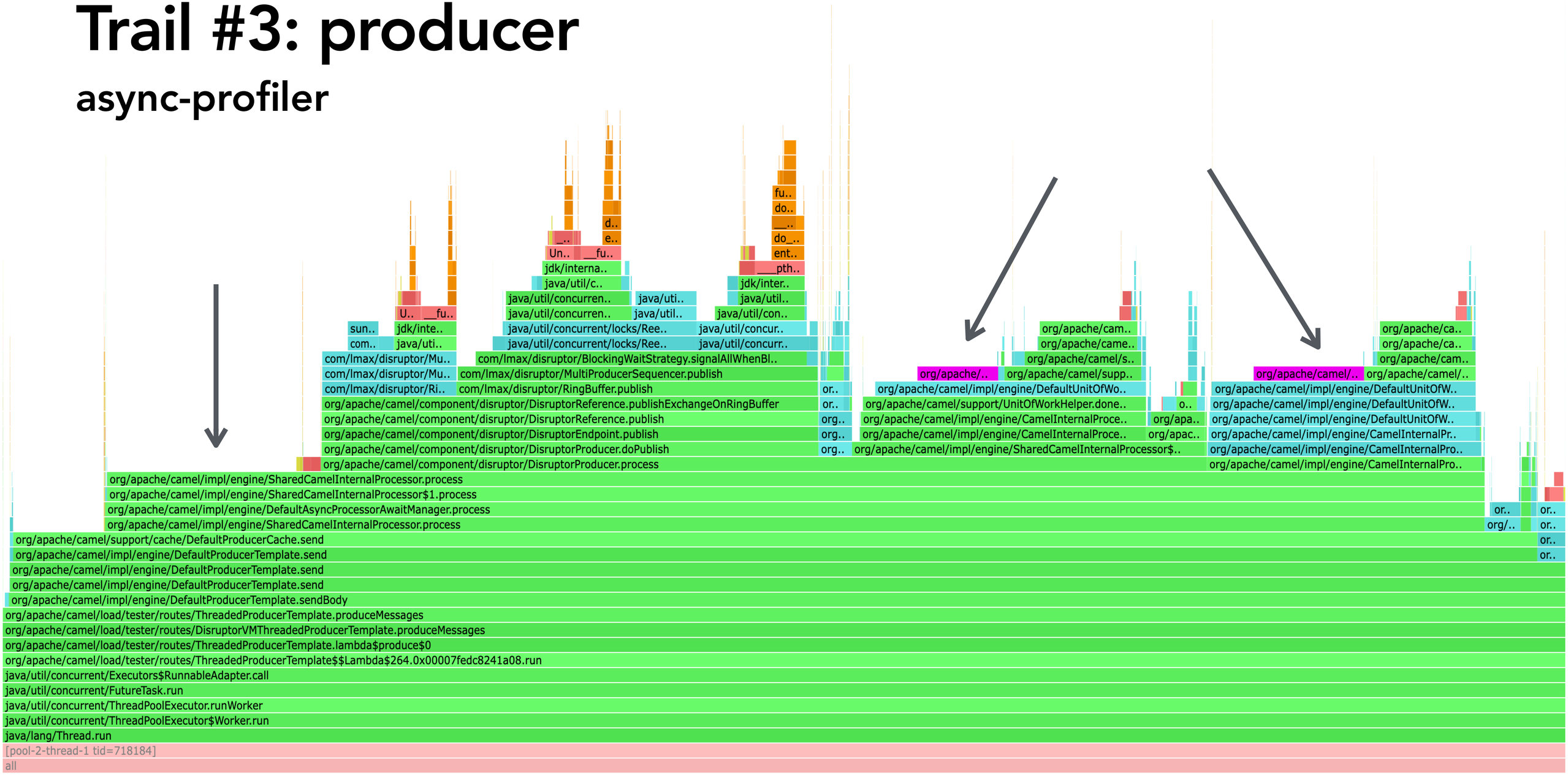
async-profiler



Matched: 12.23%

Trail #3: producer

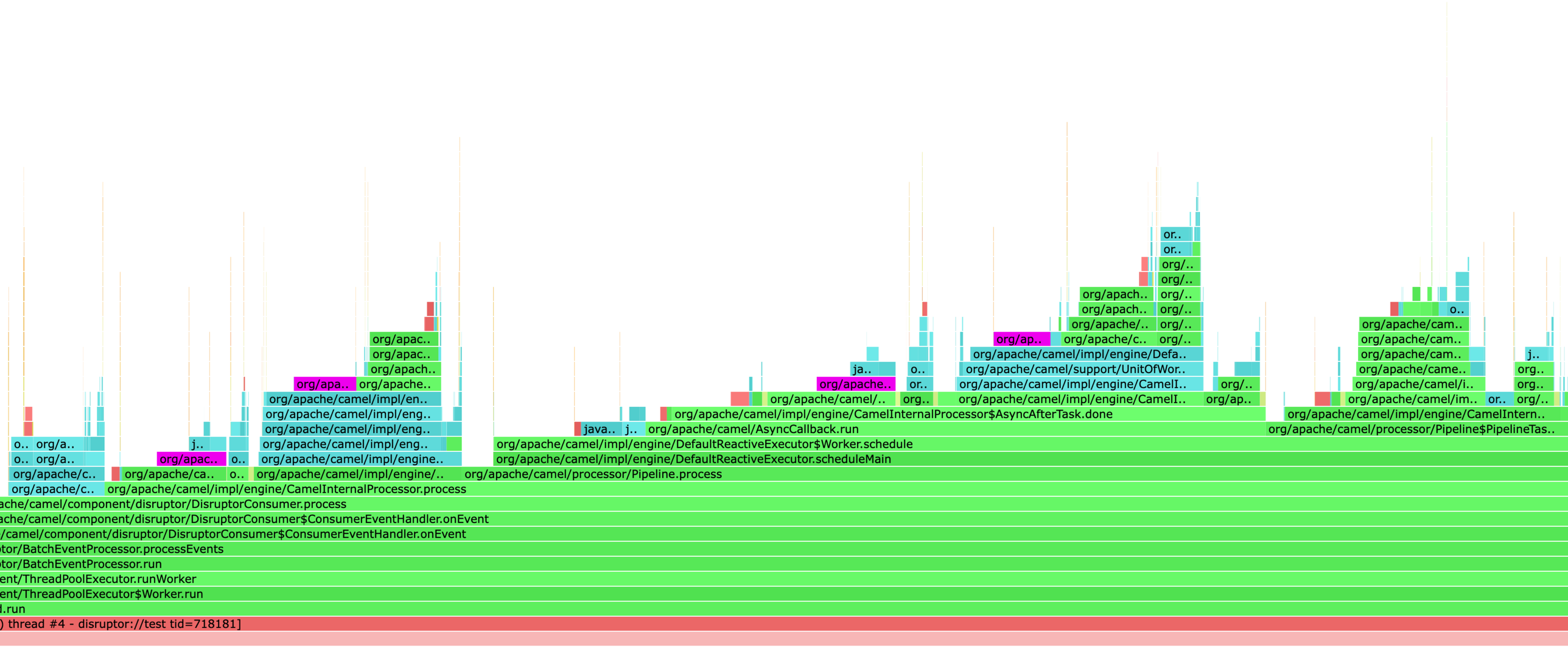
async-profiler



Matched: 12.23%

Trail #3: consumer

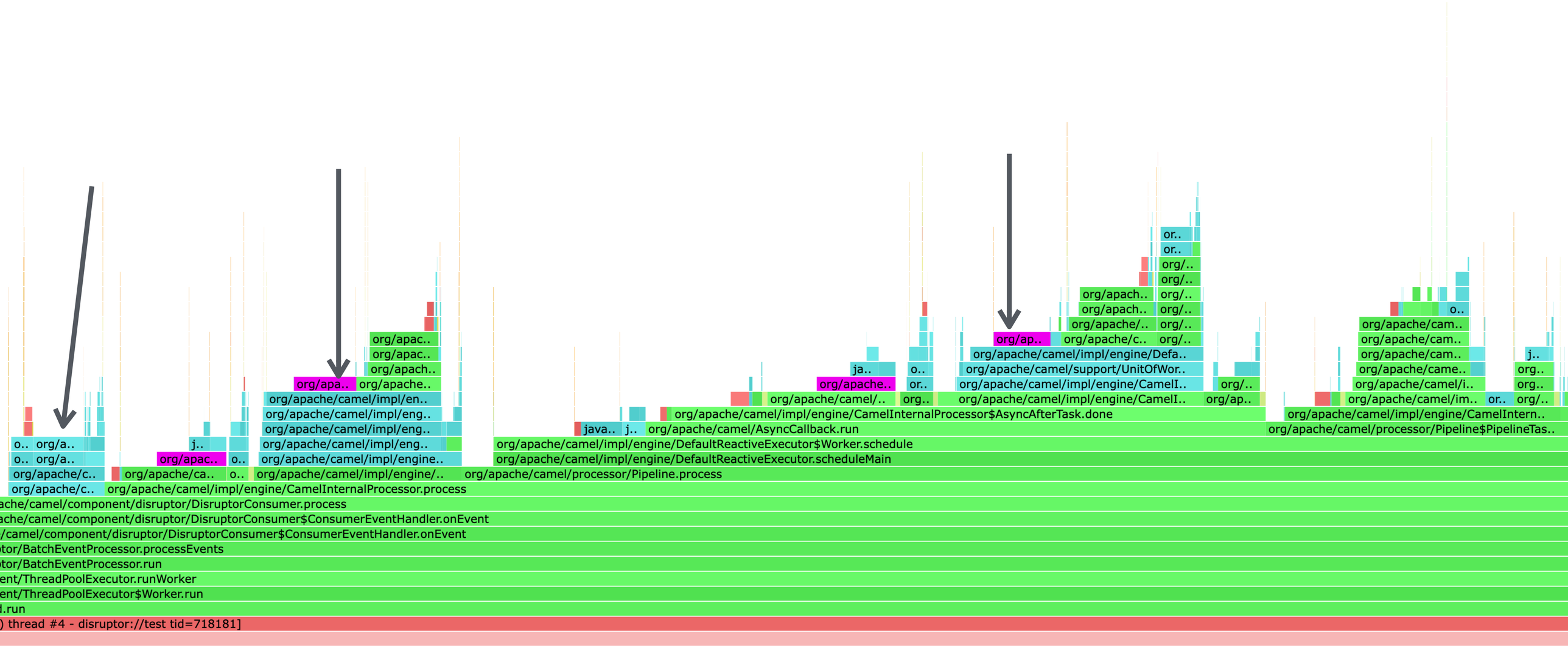
async-profiler



Mat

Trail #3: consumer

async-profiler



Trail #3

async-profiler

- Methods doing unnecessary work
- Large methods hiding valuable information
  - Likely also preventing inlining
- Inefficient operations
  - Replaced with intrinsic operations

There’s too many monsters out there

Monster slaying strategy

Too many monsters and they move fast

- Fast moving target
  - Watch constantly
- Automate
- View the trends

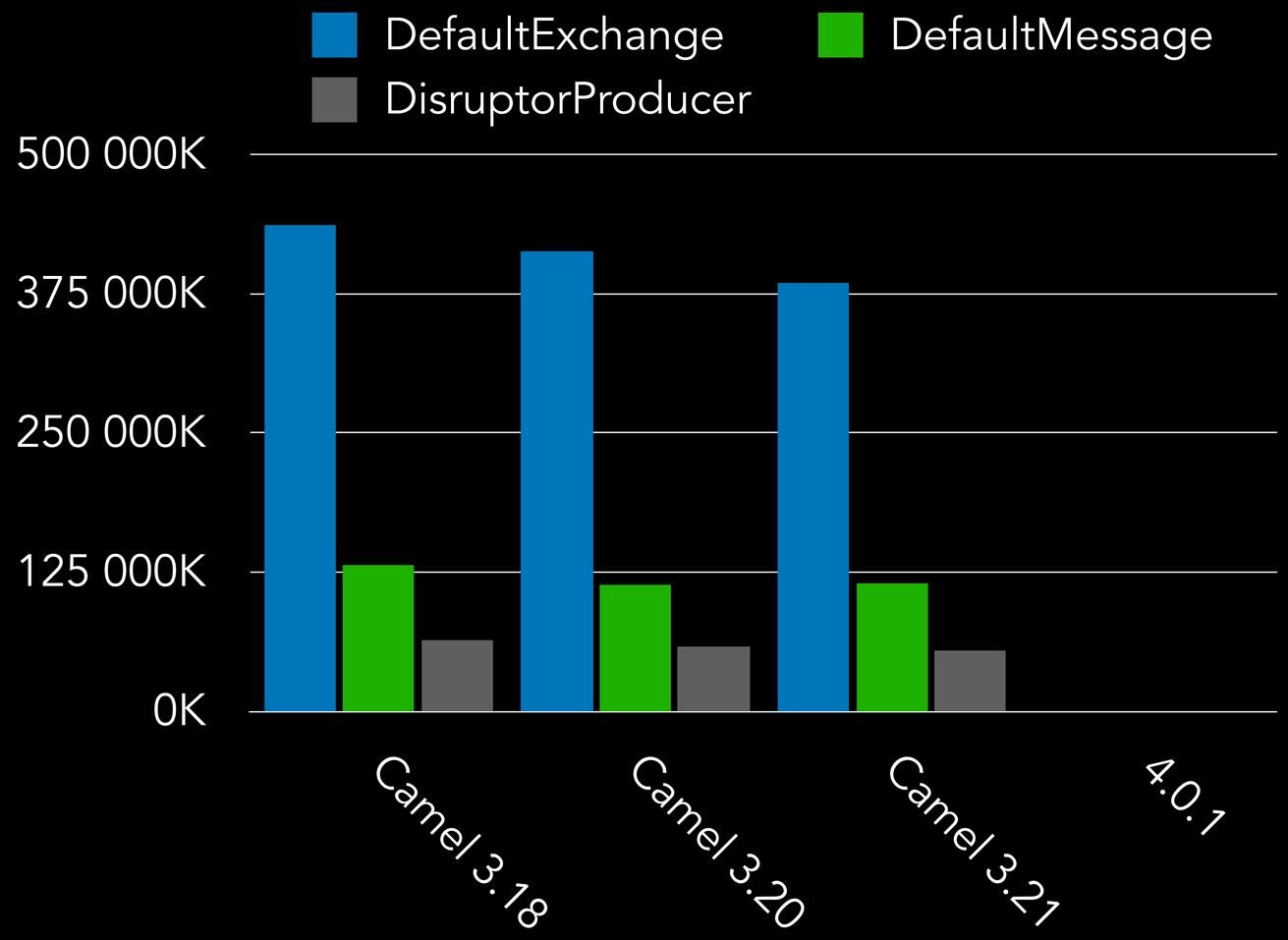
Monster slaying challenges

Benchmarking is difficult

- Variations across the stack
- Impacts from system architecture (i.e.: NUMA, core-to-core latency, etc)
- Type pollution issue:
  - (Approximately) 1 in 8 chance to happen

Collecting the rewards

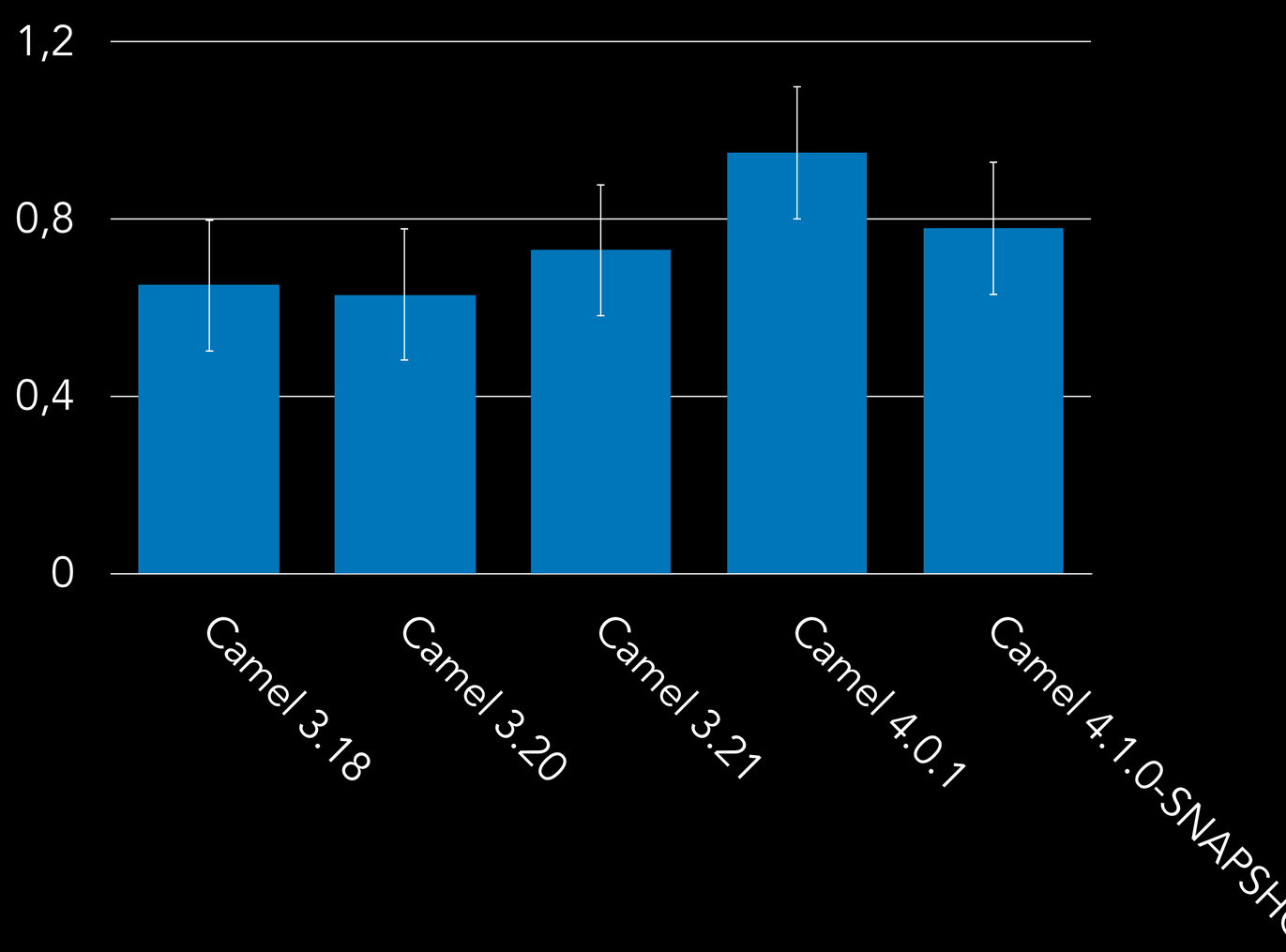
- Zero type checks for
  - Critical classes
  - In the hot path*



Collecting the rewards

Improved system usage

- Camel 4.0.1
  - Improved IPC
- Camel 4.1.0 / 4.2.0:
  - Small regression
  - Under investigation



Collecting rewards

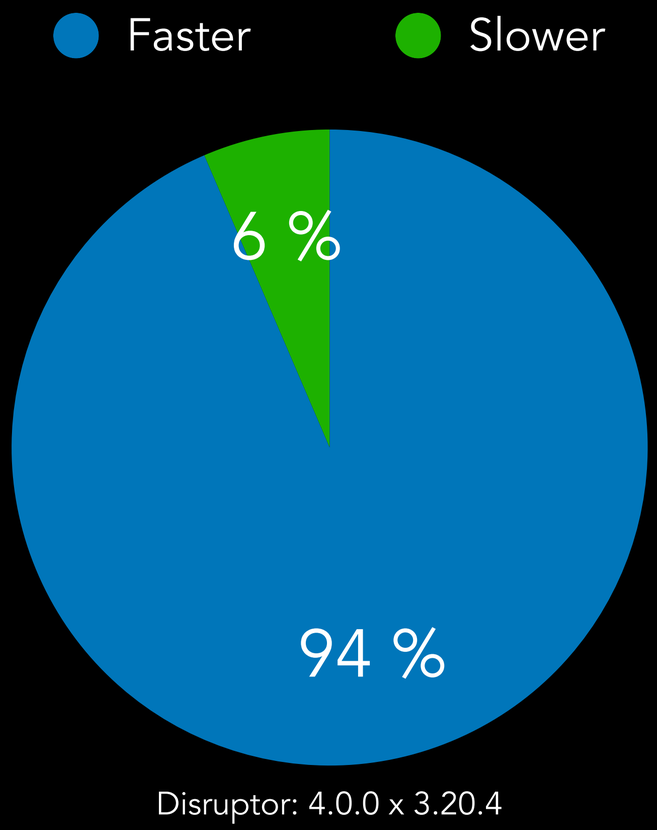
JMH

- Camel 4.0.1
  - Consistently better than 3.x
- Camel 4.1.0
  - On par with 4.0.x with a few regressions
  - Need to reduce the uncertainty in a few scenarios

Collecting rewards

Camel Load Tester

- Throughput is better
- Components
  - Disruptor
  - Seda

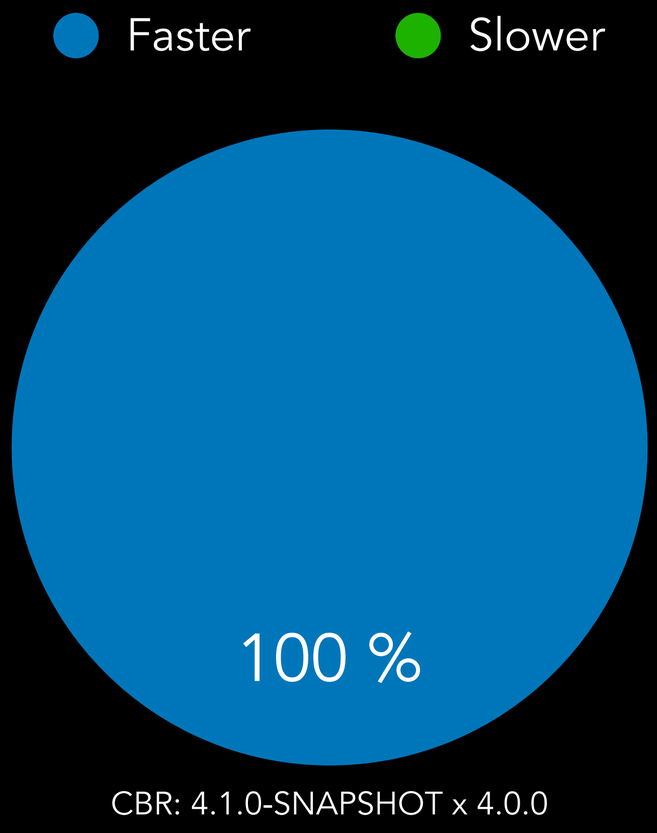


Disruptor: 4.0.0 x 3.20.4

Collecting rewards

Camel Load Tester

- Patterns (4.1.0)
  - Content-based-router
  - Filter
  - Aggregator



CBR: 4.1.0-SNAPSHOT x 4.0.0

Closing comments

- Some tools can uncover hidden bottlenecks
- Many times the fixes are simple
  - If they are not part of a pattern
- Benchmarking is hard
  - JMH simplifies a lot

Find me online

Otavio R. Piske