elastic

Java Flight Recorder

The Secret Arrow in Your Quiver

Daniel Mitterdorfer
@dmitterd

Agenda

1. What is Java Flight Recorder?
2. How do I use Java Flight Recorder?
3. Demos & Examples
4. Summary

What is Java Flight Recorder?

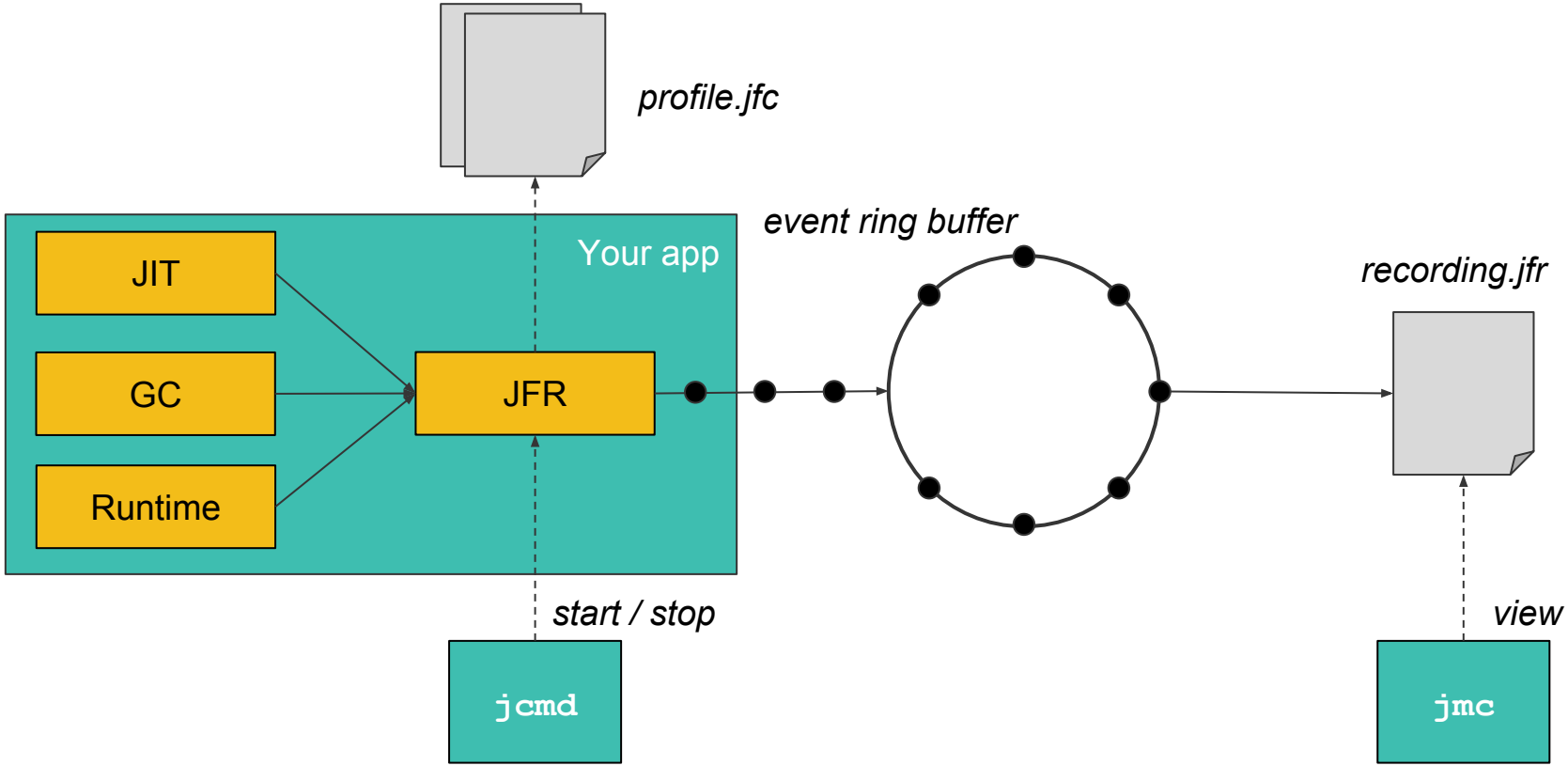
- Low-overhead sampling profiler
- Originally part of JRockit
- In **Oracle JDK** since Java 7u40


FLIGHT RECORDER
4GB FLASH/GPS
DO NOT OPEN


MISSION CONTROL CENTER
FCR-1 est. 1965
BETA= 16.1
ALT = 216.5
D/N: 0:17:40
GMT=32/18:29
LAT = 4.7
ALT = 196.7
LON= -52.8
SHK
ZKEP
DJS
ULD
USK
AQE
T27W
T27SW
SUN
SAA
-150
-120
-90
-60
-30
0
30
60
90
120
150

elastic

Java Flight Recorder from 10.000 feet



elastic

How do I use JFR?

Enabling Flight Recorder

Attach to a running application

- Start

```
jcmd $PID JFR.start
```

- Dump

```
jcmd $PID JFR.dump recording=1 filename=/tmp/r.jfr
```

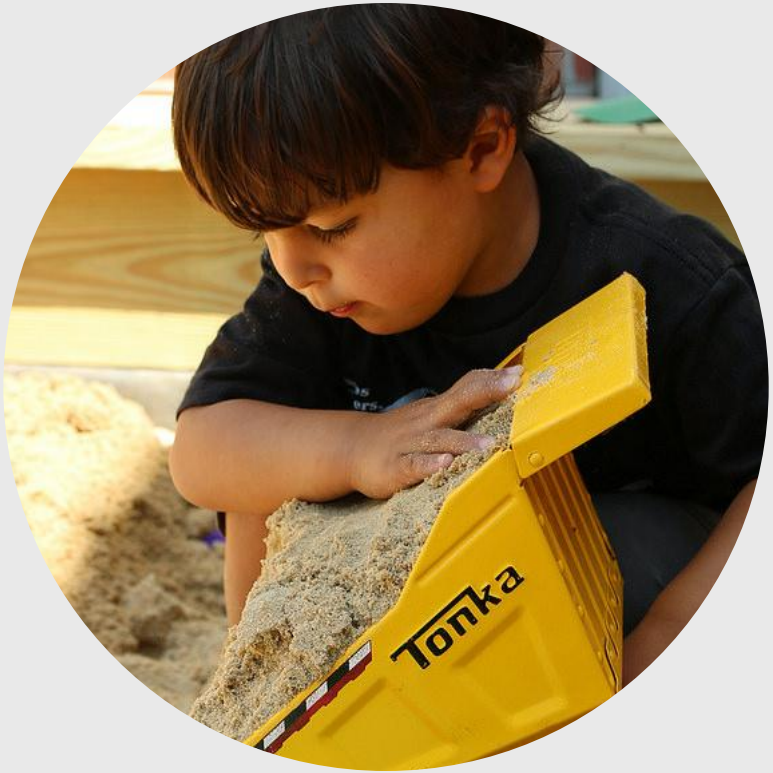
- Stop:

```
jcmd $PID JFR.stop recording=1
```

Enabling Flight Recorder

JVM flags (JDK 8)

```
-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder
```


Tonka

OR


Return To
Trust Company
of Fulton County
Gloversville, N. Y.

elastic

Avoiding Safepoint Bias

JVM flags (JDK 8)

```
-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder
-XX:+UnlockDiagnosticVMOptions
-XX:+DebugNonSafepoints
```

Safepoints?

Safepoints: Purpose

When the JVM needs to take control...

1. Halt all application threads (“safepoint”)
2. Do housekeeping tasks:
   - Garbage Collection
   - Take Thread Dumps
   - Deoptimize code
   - ….
3. Let application threads continue

Safepoint Polling*

- JVM “arms” a page in memory
- Application threads poll and raise SEGV if page is armed

* specific to JIT-compiled code in OpenJDK: http://hg.openjdk.java.net/jdk8/jdk8/hotspot/file/tip/src/share/vm/runtime/safepoint.cpp

Safepoint Polling: Problems

Some operations defer safepoint polls

- Counted loops with int counters (considered “short”-running)
- Some I/O operations: e.g. writing mmapped buffers, `System.arrayCopy()`

JVMTI vs AsyncGetCallTrace

JVM Tool Interface (JVMTI)

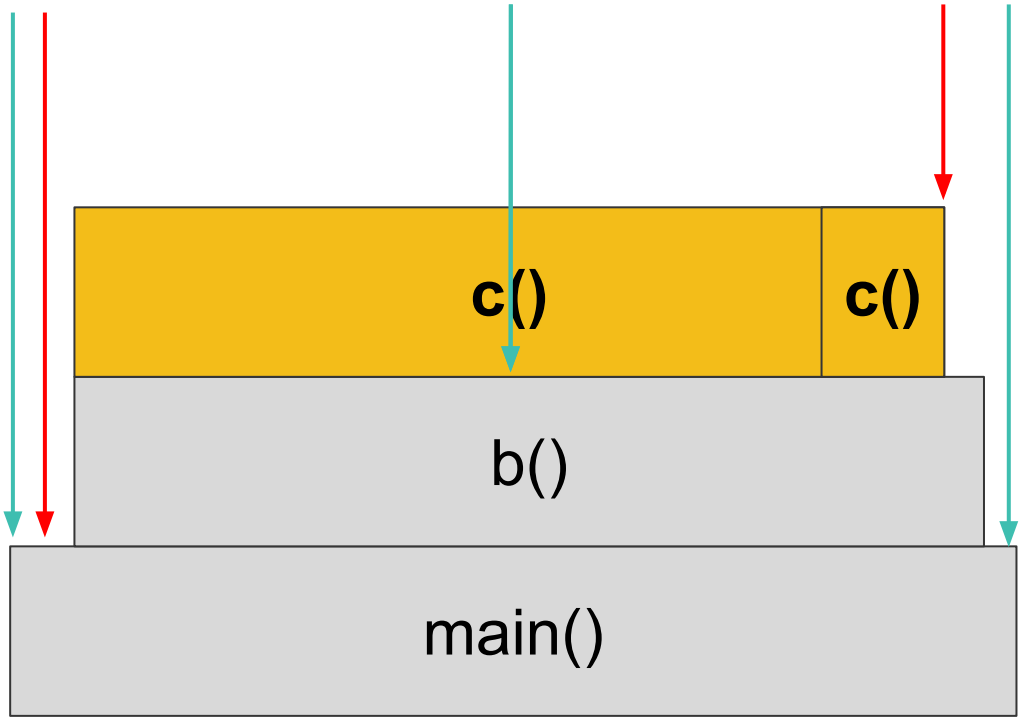
- “Official” interface for debuggers and profilers
- Captures stacks at safepoints
- Used by almost every profiler

JVMTI vs AsyncGetCallTrace

AsyncGetCallTrace

- Unofficial API
- Captures stacks at any time
- Used by Java Flight Recorder, Honest Profiler, Oracle Developer Studio

JVMTI: Safepoint Bias

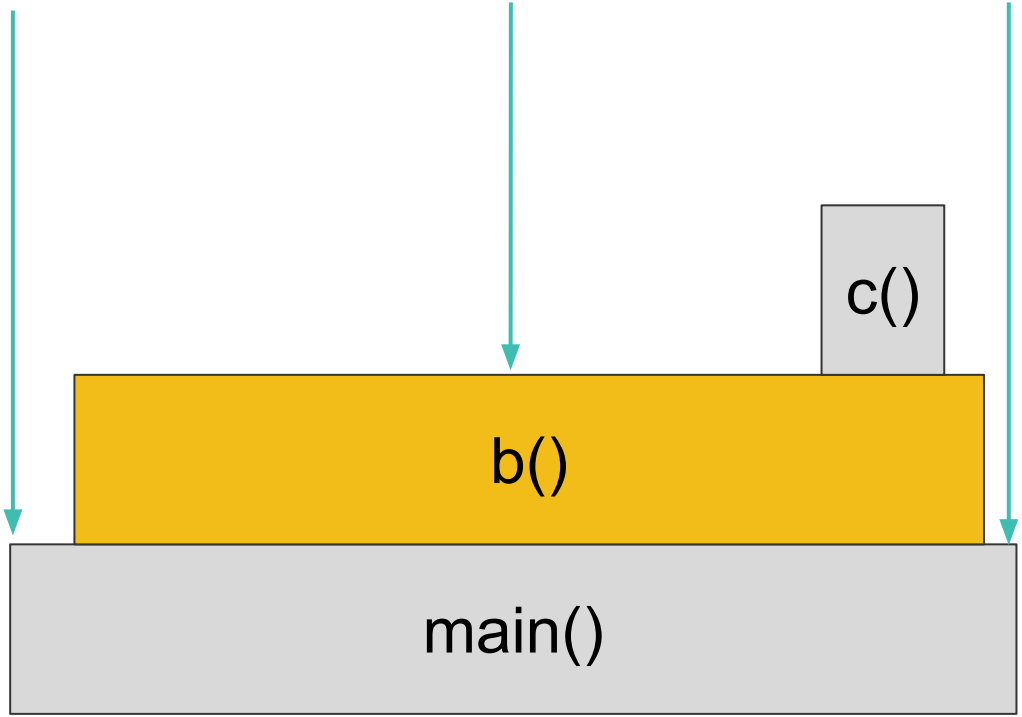


`JvmtiEnv::GetAllStackTraces()`

Safepoint Poll

elastic

AsyncGetCallTrace: No Safepoint Bias



AsyncGetCallTrace()

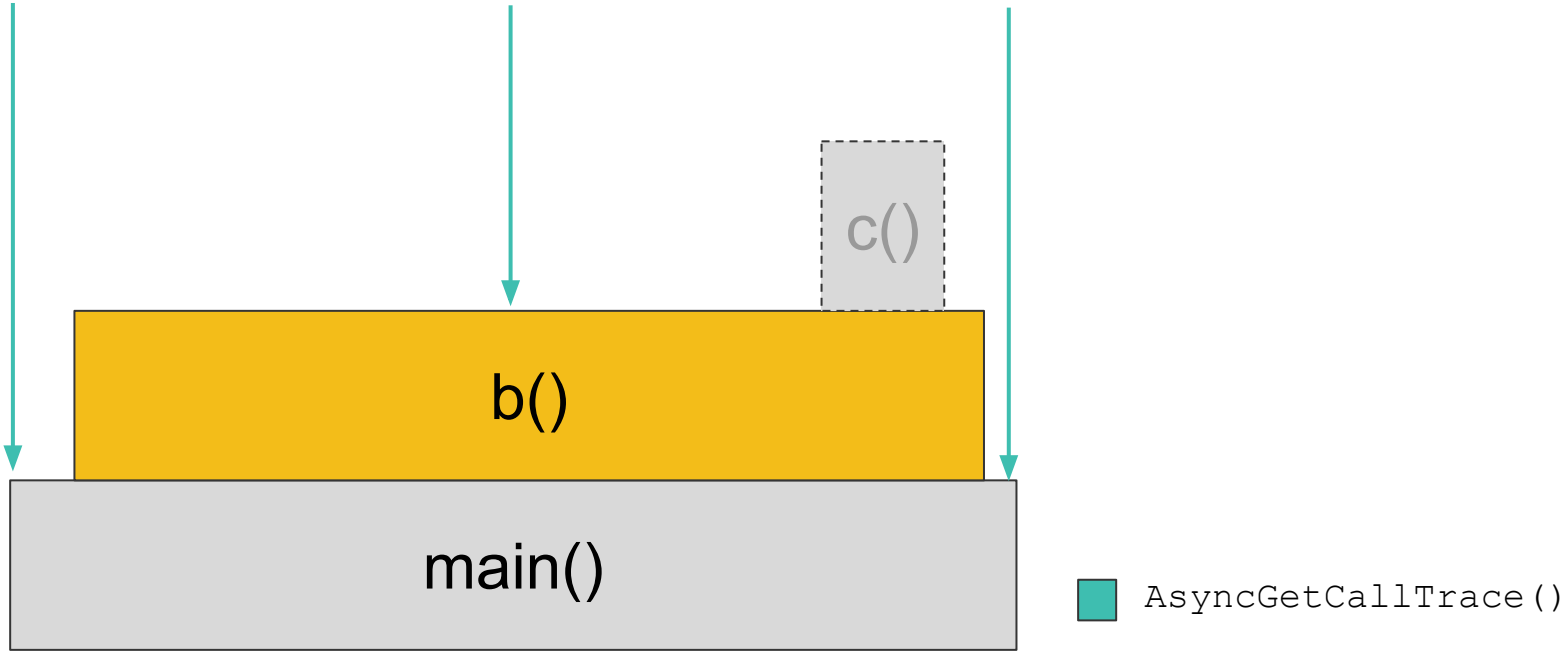
elastic

AsyncGetCallTrace is not a silver bullet

- Sampling profiler → sampling bias
- Sampling can fail:
  - GC active
  - Code is about to be deoptimized
  - Top stack frame is not a Java frame
  - ...

Sampling Bias

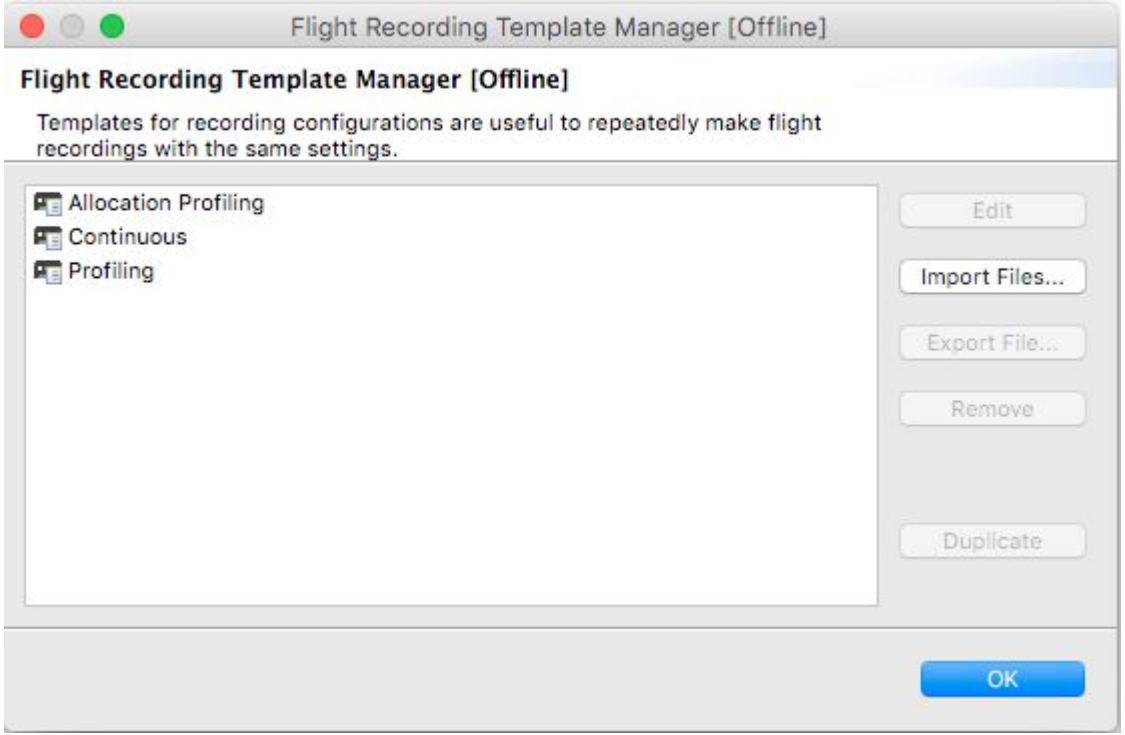
c() is “invisible”



elastic

Customized Recording Settings

Editing



elastic

Customized Recording Settings

Usage

- Stored as `*.jfc` in `$JAVA_HOME/jre/lib/jfr/`
- To use `memory.jfc`:

```
-XX:StartFlightRecording=defaultrecording=true,settings=memory
```

Starting a Profiling Recording (Fixed Time)

JVM flags (JDK 8)

```
-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder
-XX:+UnlockDiagnosticVMOptions
-XX:+DebugNonSafepoints
-XX:StartFlightRecording=delay=240s,duration=60s,
settings=profile,filename=rec.jfr
```

Starting a Continuous Recording

JVM flags (JDK 8)

```
-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder
-XX:+UnlockDiagnosticVMOptions
-XX:+DebugNonSafepoints
-XX:StartFlightRecording=defaultrecording=true
```

Low Overhead < 2%

Source: https://www.oracle.com/technetwork/java/javaseproducts/mission-control/java-mission-control-wp-2008279.pdf

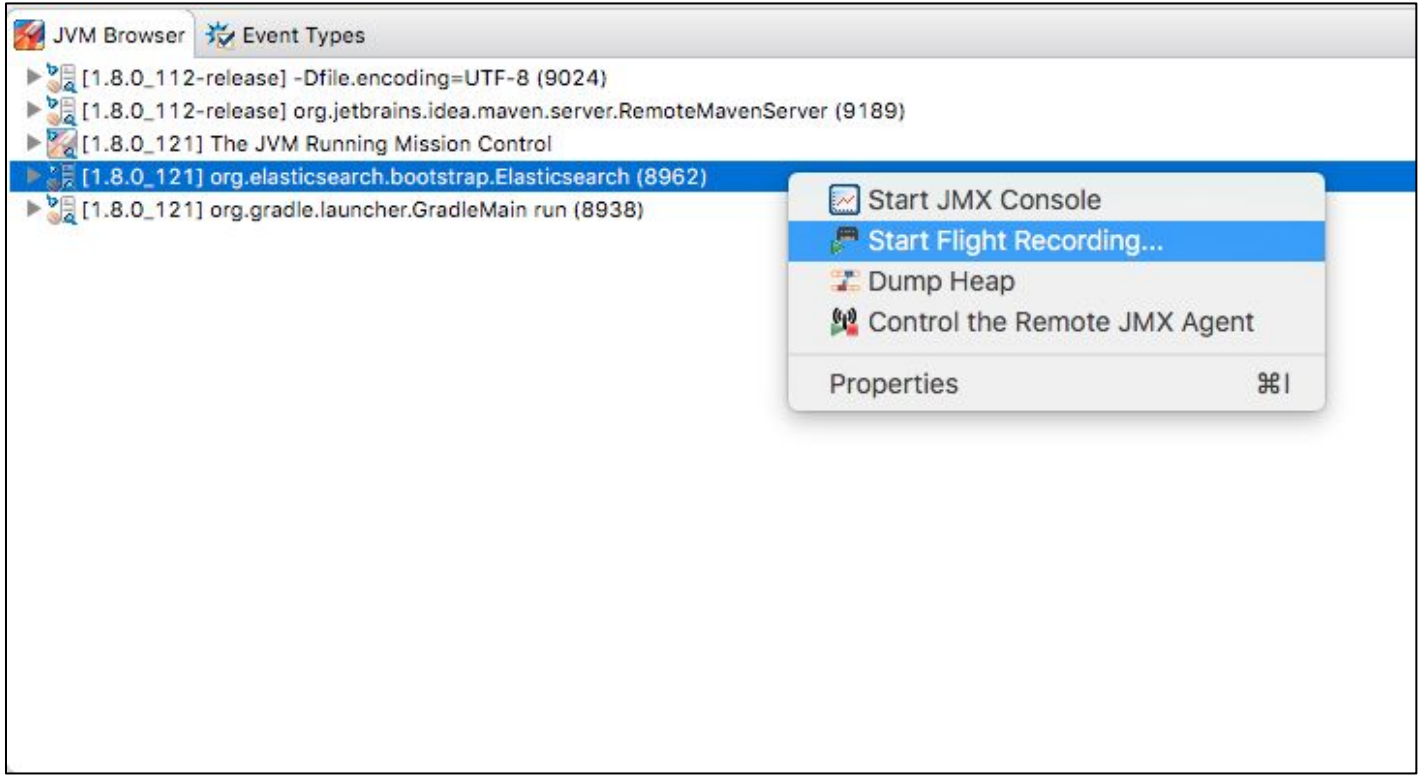
Dump a recording at application exit

JVM flags (JDK 8)

```
-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder
-XX:+UnlockDiagnosticVMOptions
-XX:+DebugNonSafepoints
-XX:StartFlightRecording=defaultrecording=true
-XX:FlightRecorderOptions=disk=true,maxage=0s,maxsize=0,
dumponexit=true,dumponexitpath=/rec.jfr
```

Enabling Flight Recorder

From Mission Control

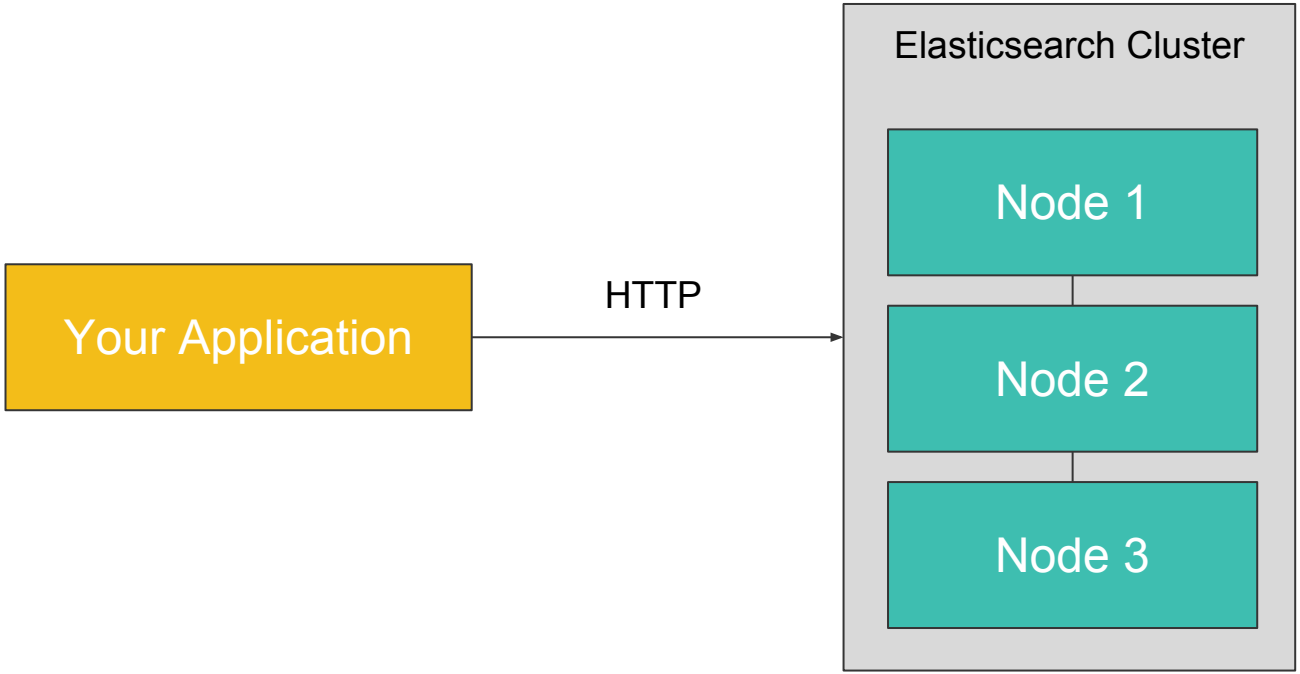


elastic

Enough talking: Demos & Examples

Example Application: Elasticsearch

Open Source Distributed Search Engine



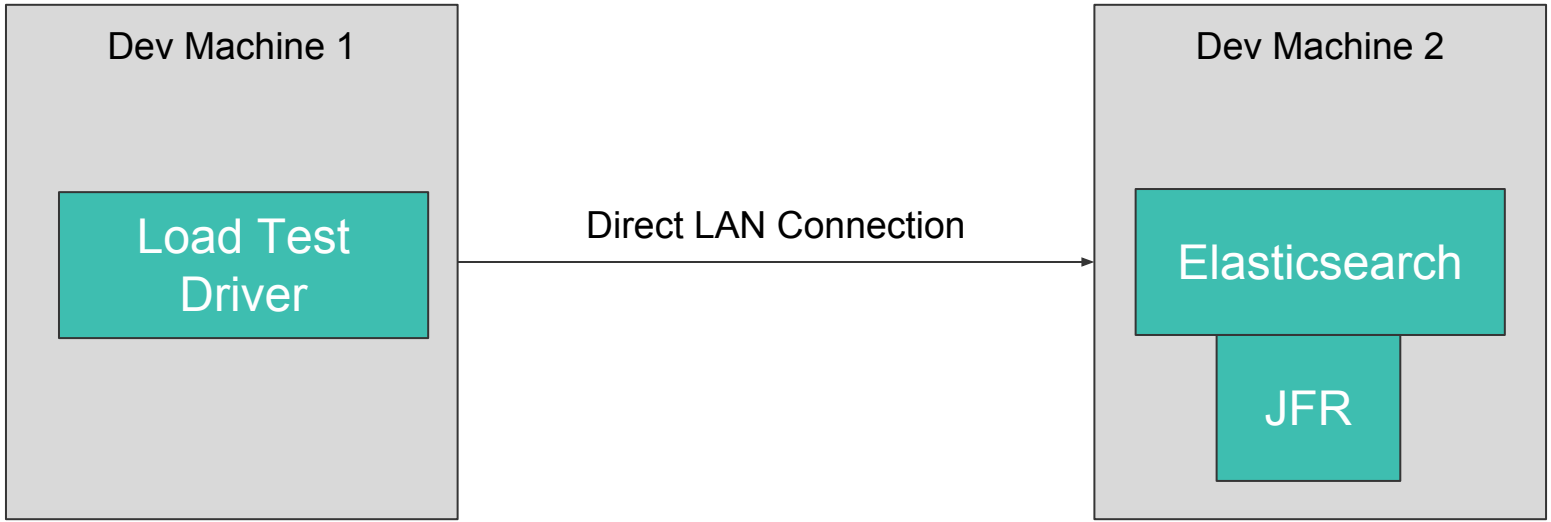
elastic

Example Application: Elasticsearch

Key Operations and Metrics

- **Adding Documents to the Index:** Throughput
- **Querying:** Latency

My Usual Setup



Demo: CPU Profiling

Demo: CPU Profiling

Background:elastic/elasticsearch#7309

What is a “sane” default compression level for HTTP traffic?

Demo: CPU Profiling

Parameters

Compression level: 0 (off) - 9 (max)

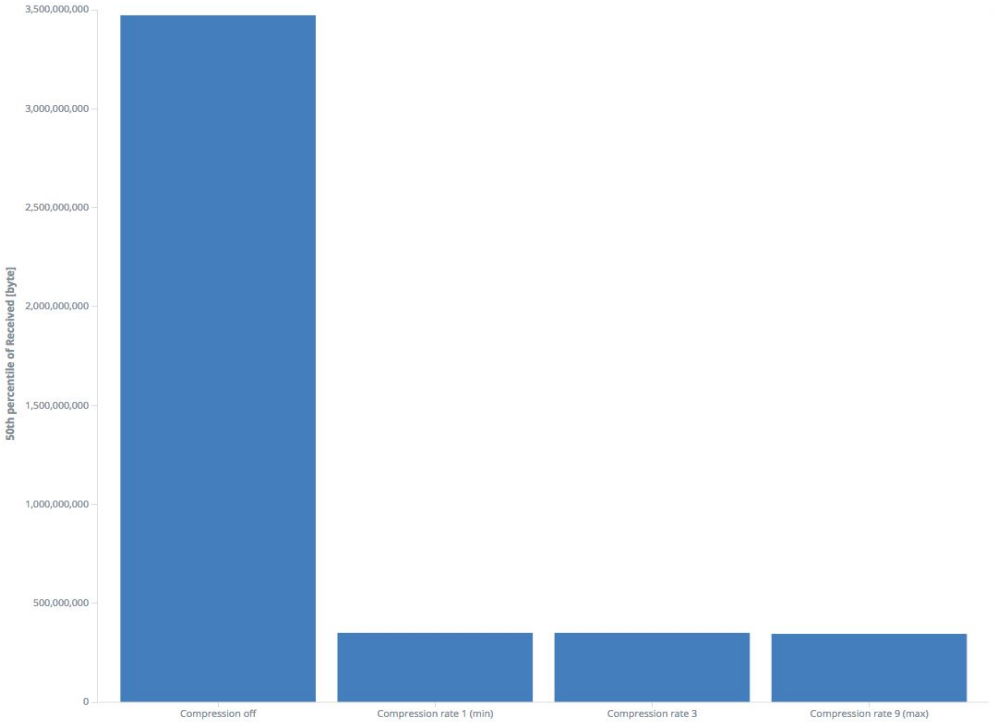
Demo: CPU Profiling

Influenced Variables

- Bytes transferred over network
- Throughput (Indexing)
- Latency (Queries)
- CPU Usage

Demo: CPU Profiling

Bytes Transferred

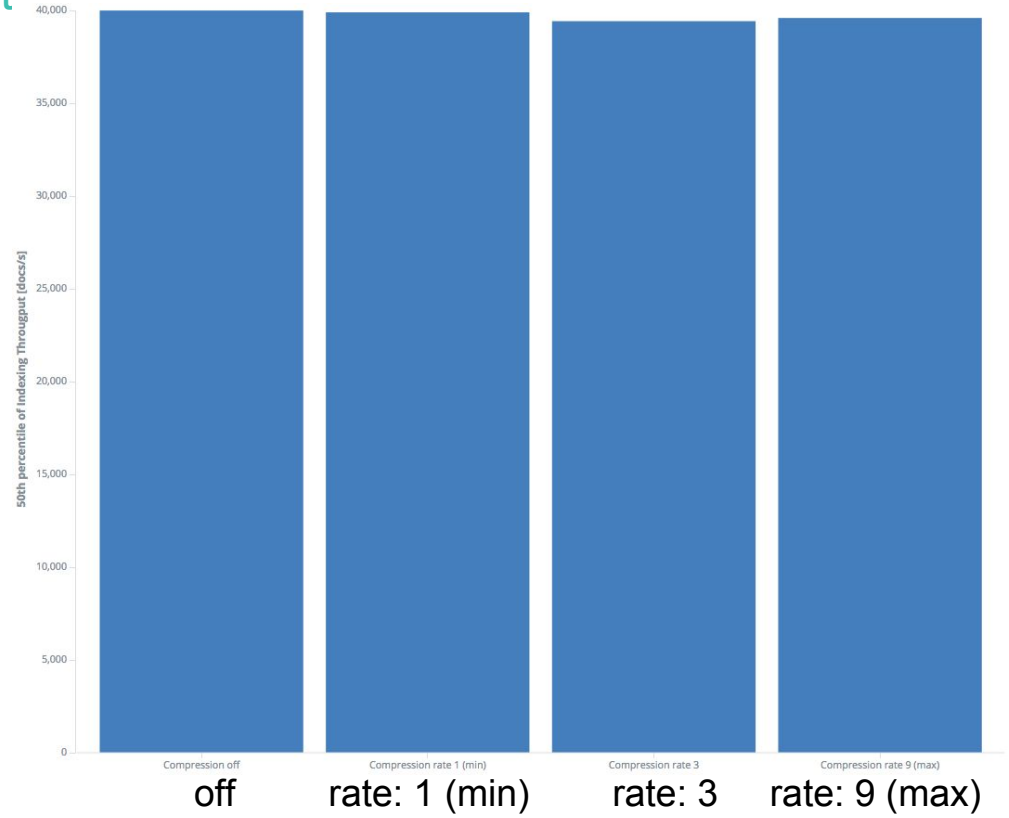


off rate: 1 (min) rate: 3 rate: 9 (max)

elastic

Demo: CPU Profiling

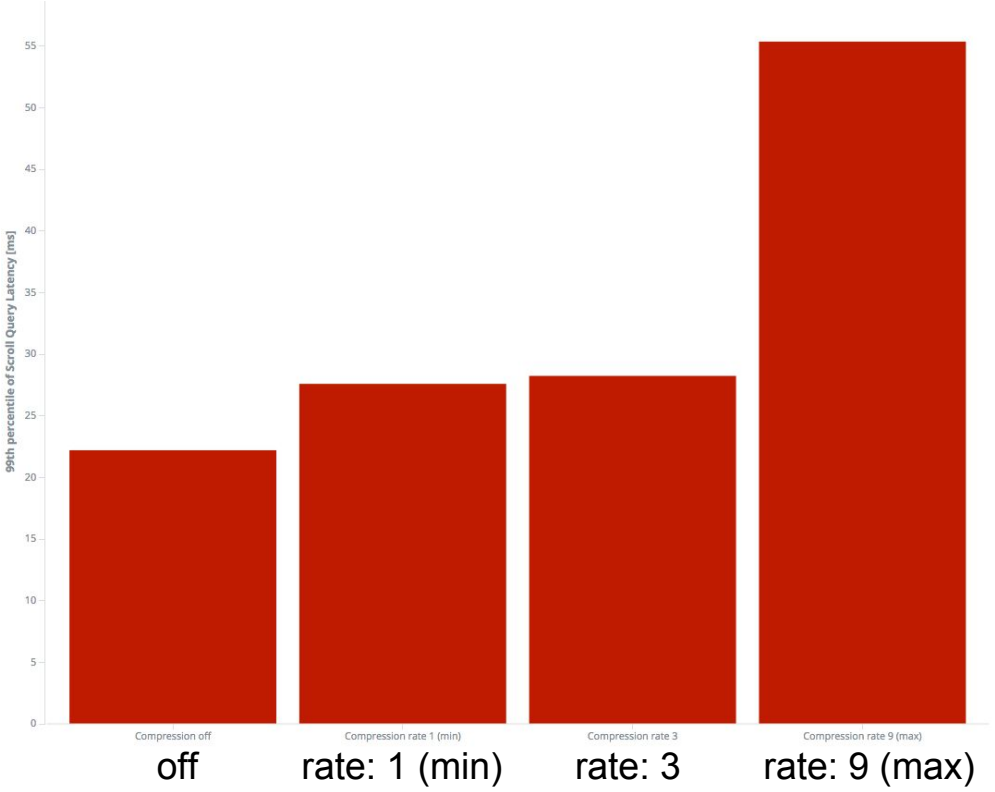
Indexing Throughput



elastic

Demo: CPU Profiling

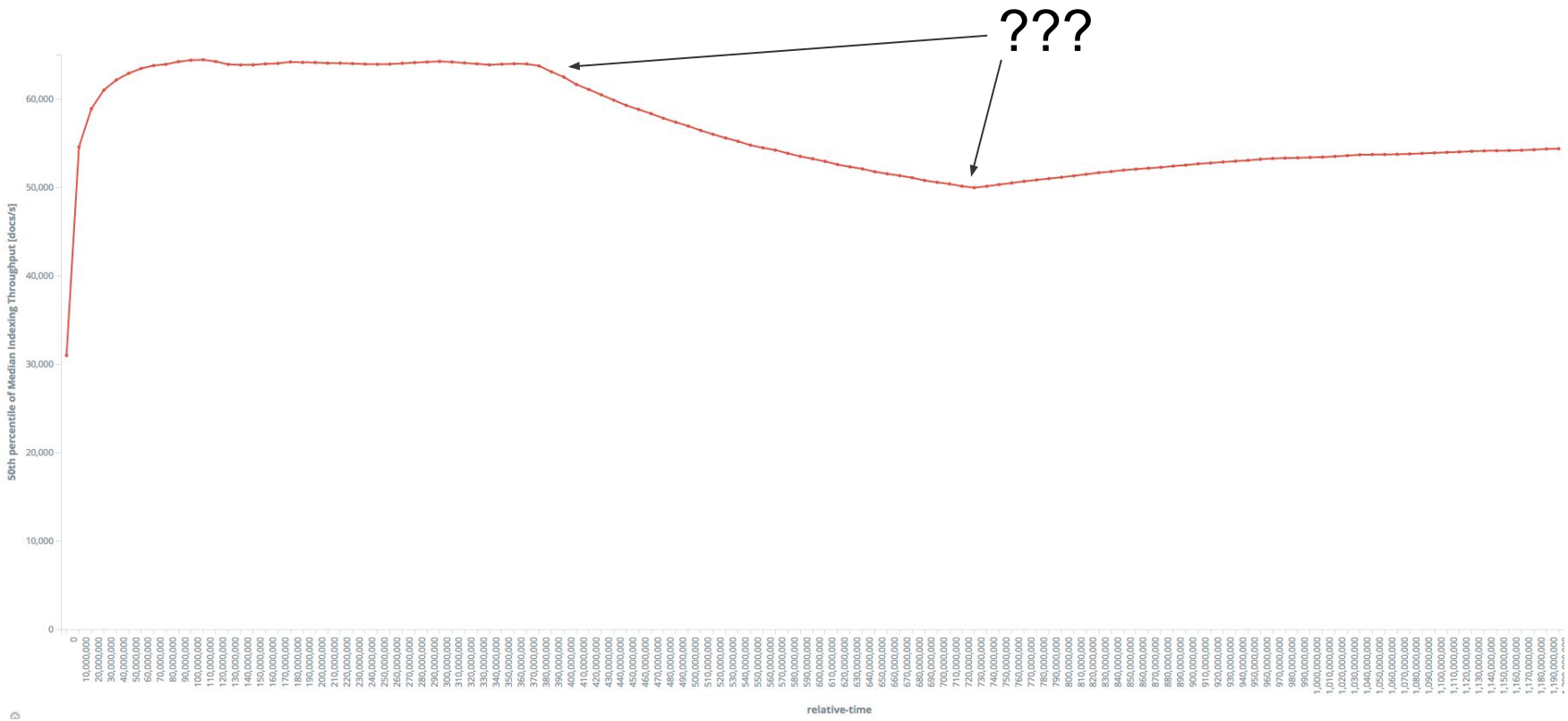
Query Latency



elastic

Demo: Slow I/O

Demo: Slow I/O



elastic

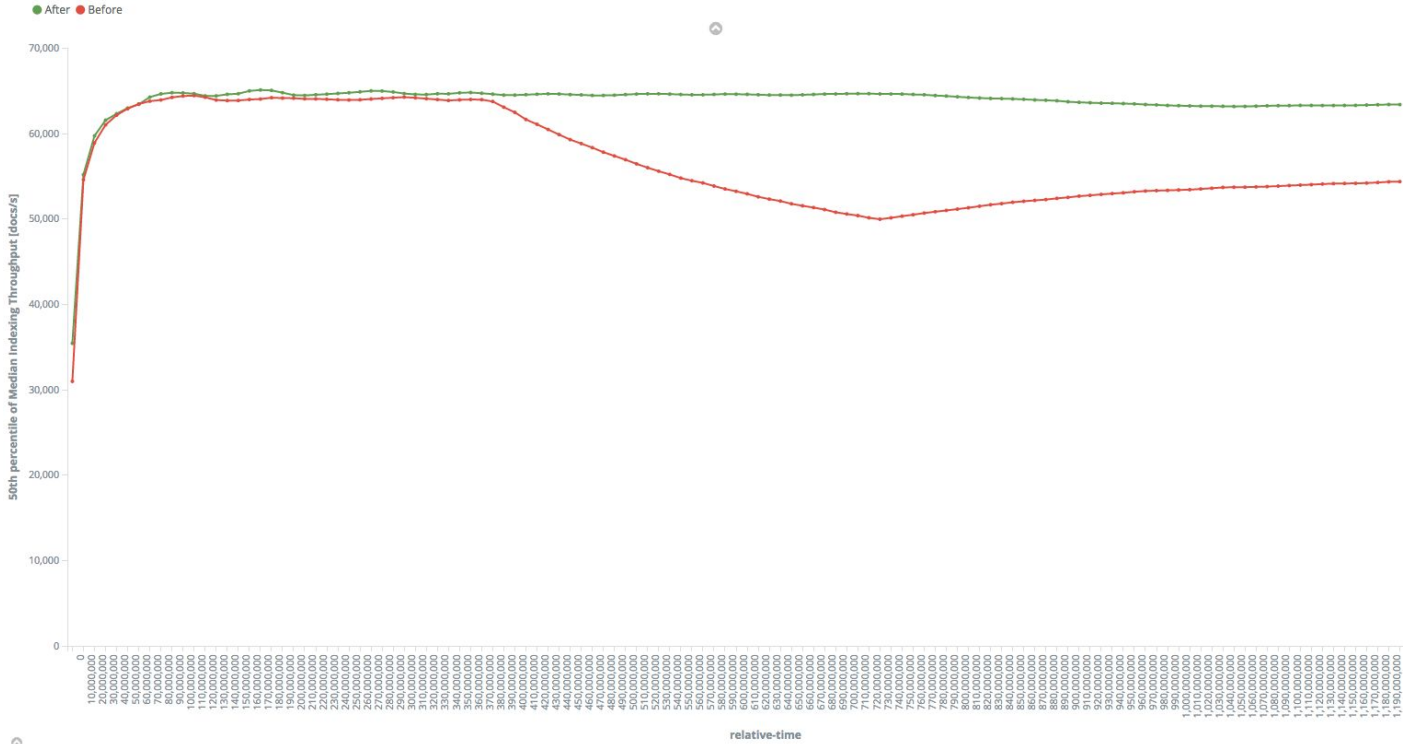
Demo: Slow I/O

Culprit: TRIM

- **Baseline:** 620MB/s → 560MB/s (10% drop)
- **This system:** 670MB/s → 60MB/s (90% drop)

Demo: Slow I/O

Throughput Comparison

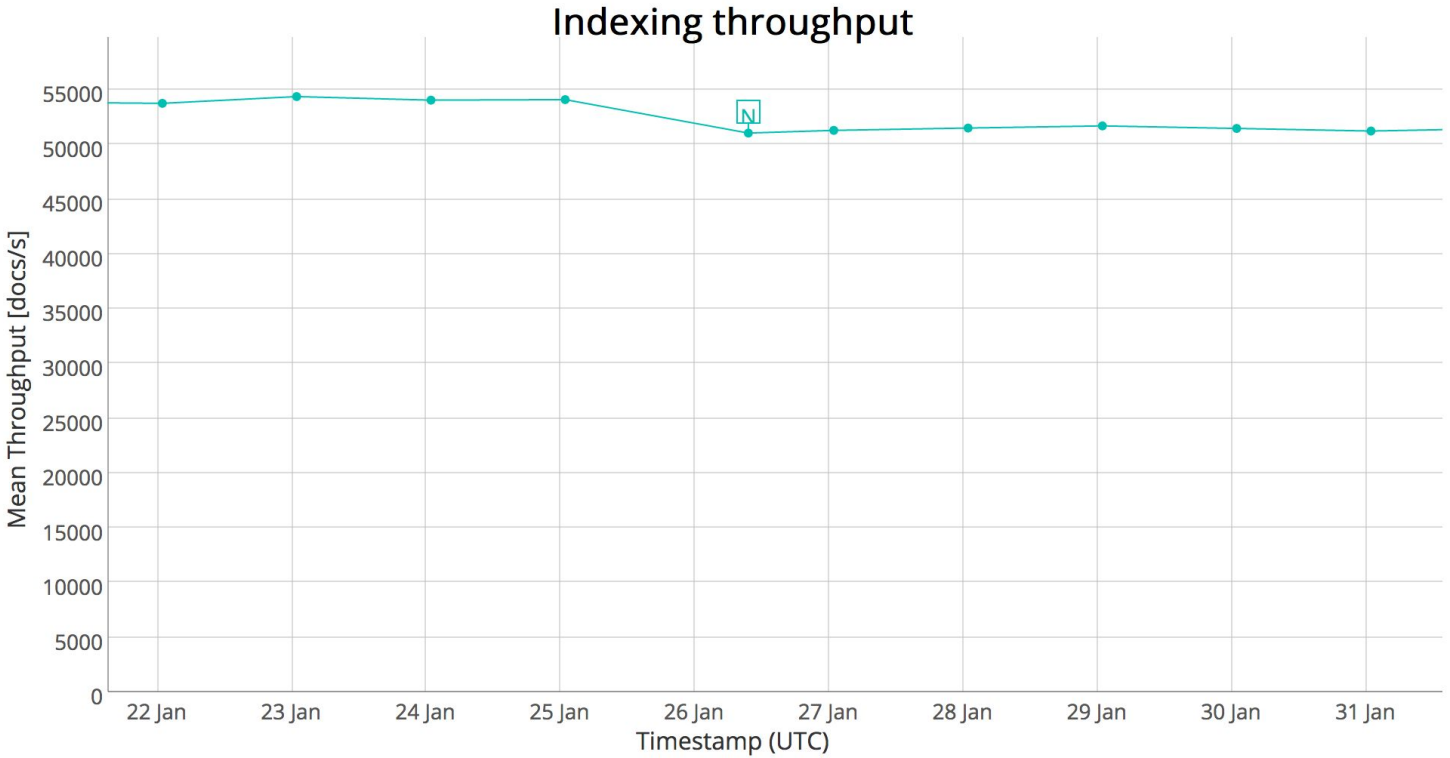


elastic

Demo: Allocation Profiling

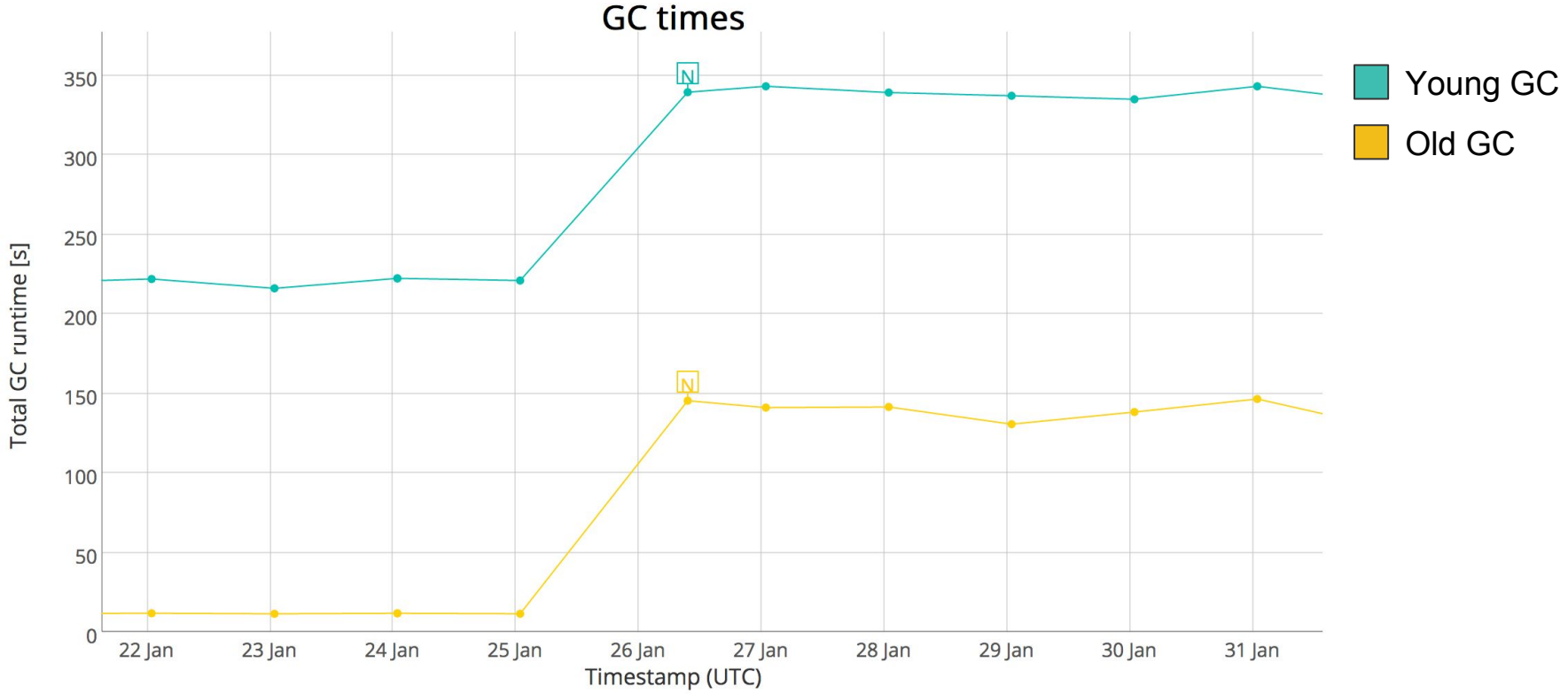
Demo: Allocation Profiling

Background:elastic/elasticsearch#23185



Source: https://elasticsearch-benchmarks.elastic.co/nyc_taxis/

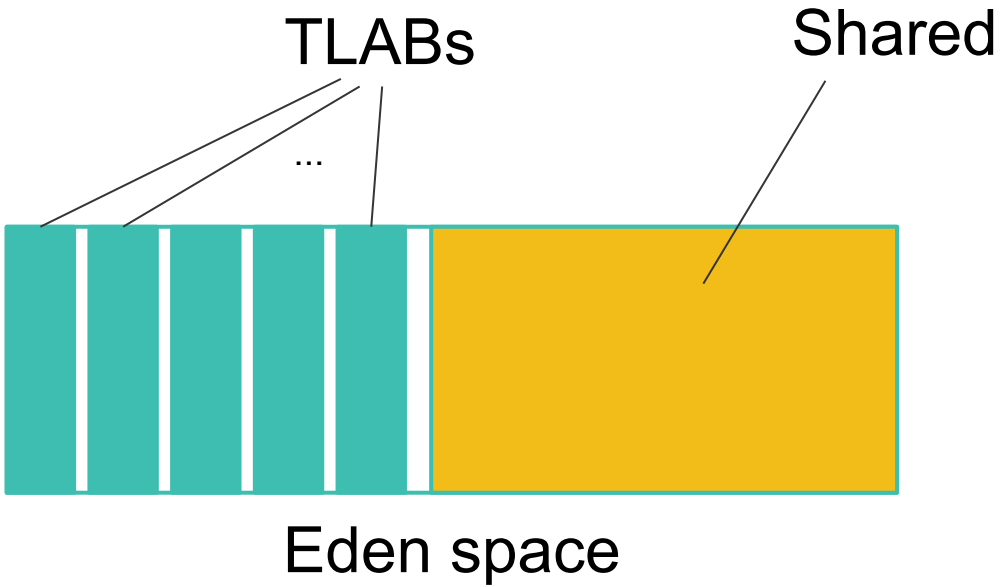
Demo: Allocation Profiling



Source: https://elasticsearch-benchmarks.elastic.co/nyc_taxis/

Demo: Allocation Profiling

Thread Local Allocation Buffers (TLABs) in the JVM

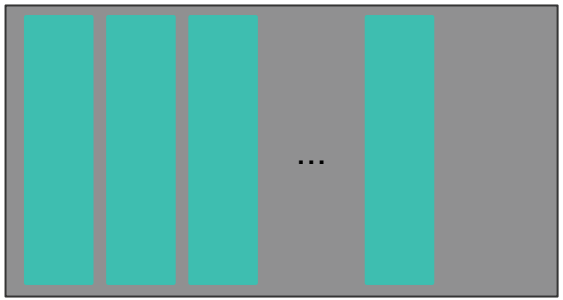


elastic

Demo: Allocation Profiling

Buffers in Netty

Many Buffers - One per read



One HTTP Request

elastic

Demo: Allocation Profiling

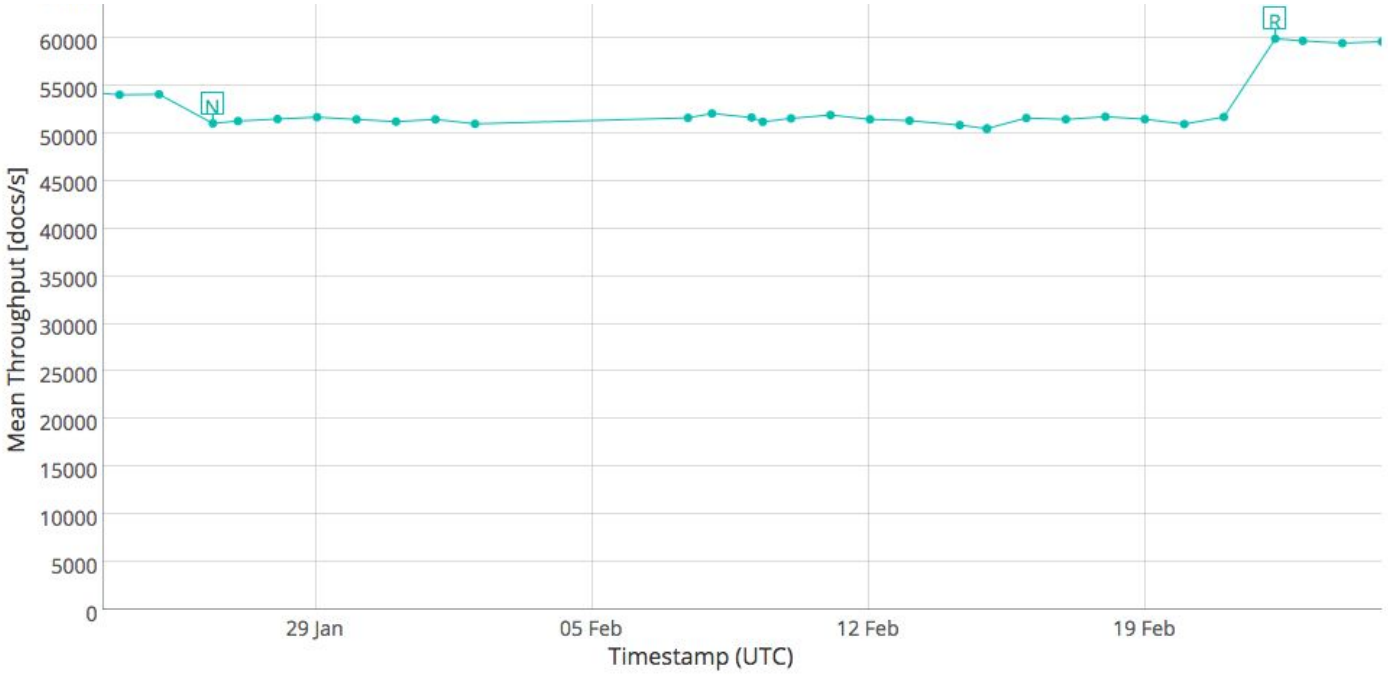
Problem: MTU

- **MTU Loopback:** 65536 bytes
- **MTU Ethernet:** 1500 bytes

→ Lower internal buffer sizes

Demo: Allocation Profiling

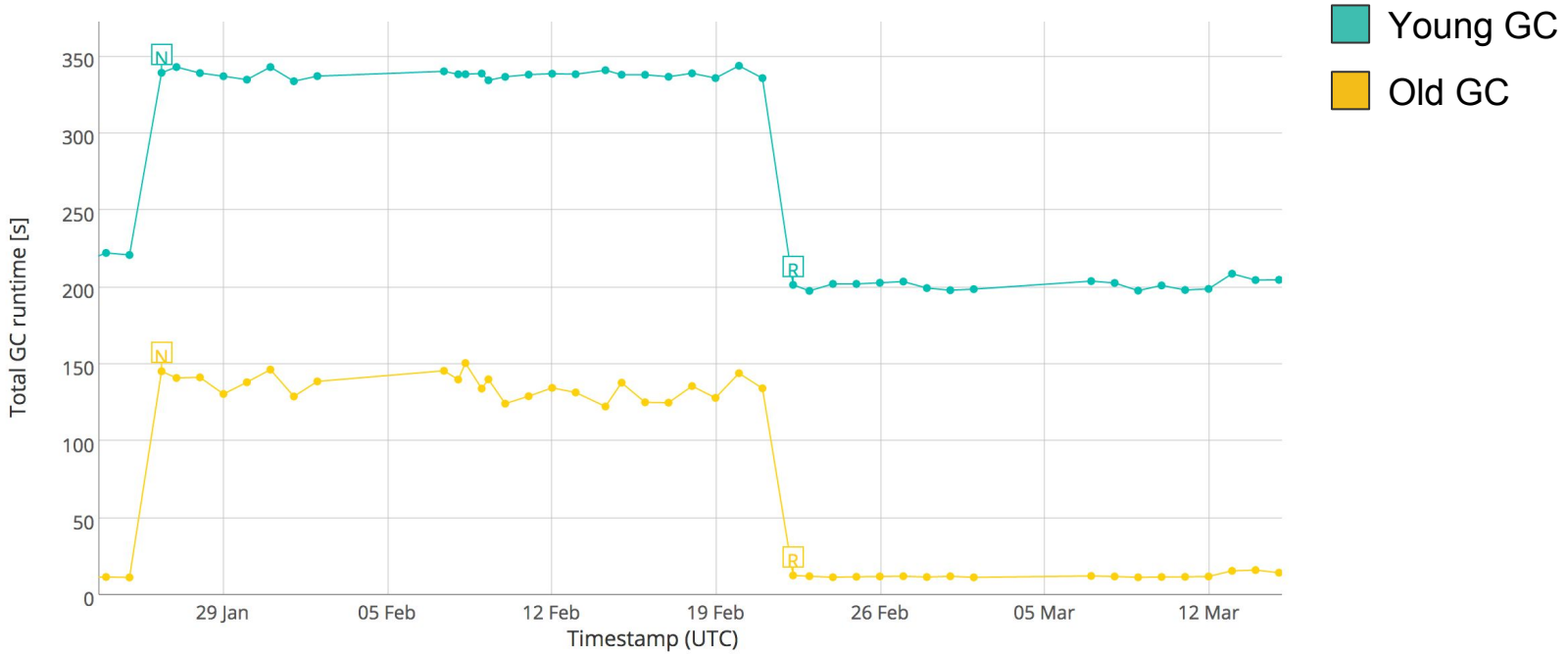
Throughput Comparison



Source: https://elasticsearch-benchmarks.elastic.co/nyc_taxis/

Demo: Allocation Profiling

GC Times Comparison



Source: https://elasticsearch-benchmarks.elastic.co/nyc_taxis/

elastic

Summary & Outlook

JFR: A versatile tool

- Commercial extension of Oracle JDK
- Low overhead, customizable profiles
- Profiling and Continuous Mode
- Analyze various performance issues: CPU, Memory, Locking, ...

JFR and JDK 9

- JDK 9 will bring a new version of Mission Control
- Flight recording also produced on crashes
- Adds supported APIs
  - Create your own events
  - Parse flight recordings
- More events

Source: https://www.jug.ch/events/slides/170215_SwissJUG2017_zurich_sneakpeek.pdf

References and Further Reading

- AsyncGetCallTrace: http://psy-lob-saw.blogspot.de/2016/06/the-pros-and-cons-of-agct.html
- Safepoints and Safepoint Bias
  - http://psy-lob-saw.blogspot.de/2015/12/safepoints.html
  - http://psy-lob-saw.blogspot.de/2016/02/why-most-sampling-java-profilers-are.html
- Flight Recorder Docs: https://docs.oracle.com/javacomponents/jmc-5-5/jfr-runtime-guide/
- Generally recommended: http://hirt.se/

Image Credits

- So klein und schon ein flight recorder by Uwe Hauck (used with written permission of the author)
- Nasa Social February 2017 by Nasa Johnson (license: CC BY-NC-ND 2.0)
- Tyler Greene Dodges Eric Young by Paul Martinez (license: CC BY-NC-ND 2.0)
- IMG_2857 by John M (license: CC BY-NC-ND 2.0)
- Greed by las - initially (license: CC BY-NC-ND 2.0)
- IMG_9861 by 7263255 (license: CC BY-SA 2.0)
- 2014 Bahrain GP - FP3 & Qualifying by CaterhamF1 (license: CC BY-NC-ND 2.0)