ORACLE®

# Maximizing Performance

with

# GraalVM™

Thomas Wuerthinger (@thomaswue)
Senior Research Director
Oracle Labs
June 25, 2019

ORACLE®

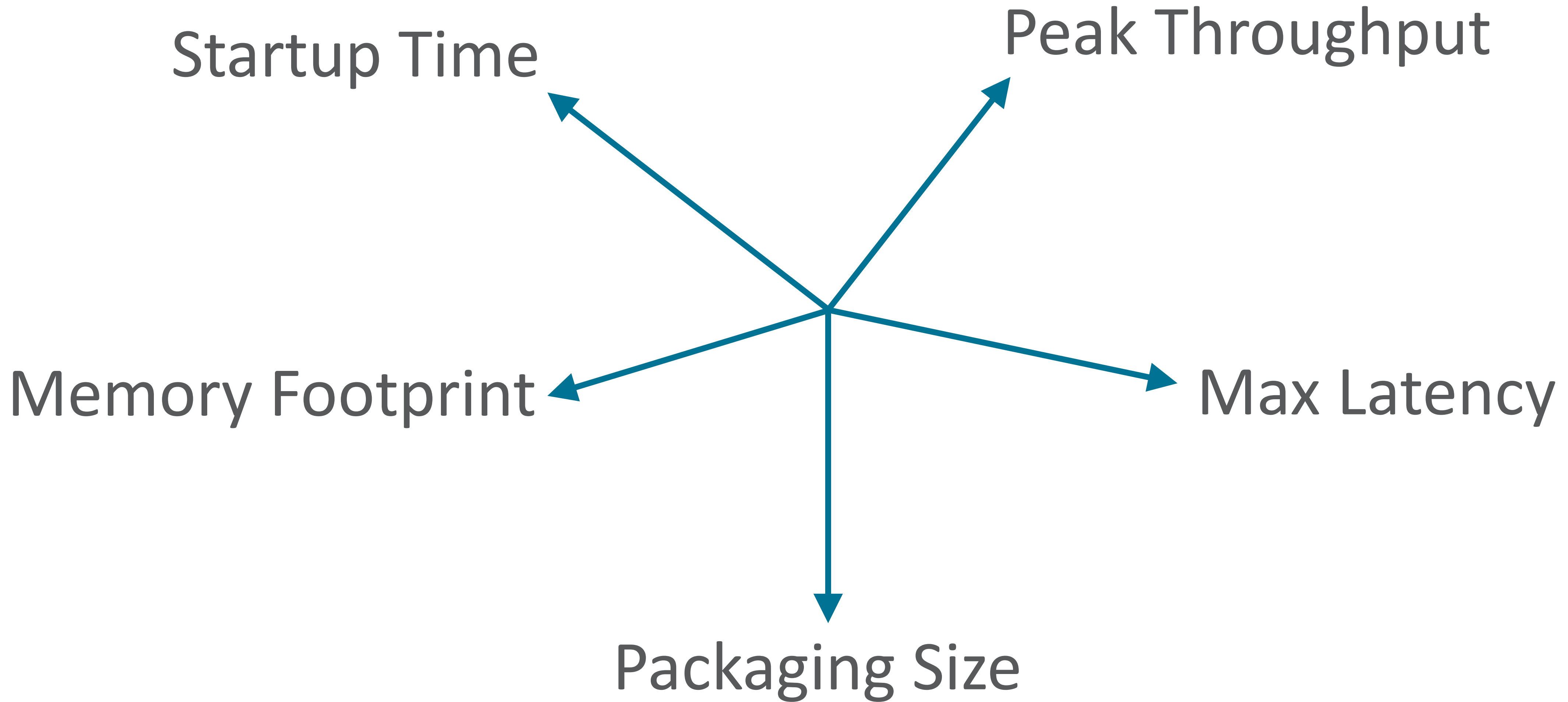
Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

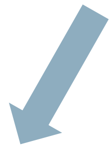
ORACLE

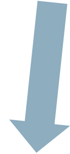
Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

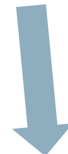



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |







OpenJDK

ORACLE
DATABASE

standalone

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

## Community Edition (CE)

GraalVM CE is available for free for development and production use. It is built from the GraalVM sources available on GitHub. We provide pre-built binaries for GraalVM CE for Linux on x86 64-bit systems.

DOWNLOAD FROM GITHUB

## Enterprise Edition (EE)

GraalVM EE provides additional performance, security, and scalability relevant for running critical applications in production. It is free for evaluation uses and available for download from the Oracle Technology Network. We provide binaries for GraalVM EE for Linux or Mac OS X on x86 64-bit systems.

DOWNLOAD FROM OTN

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

GraalVM™

JIT

AOT

java MyMainClass

OpenJDK™

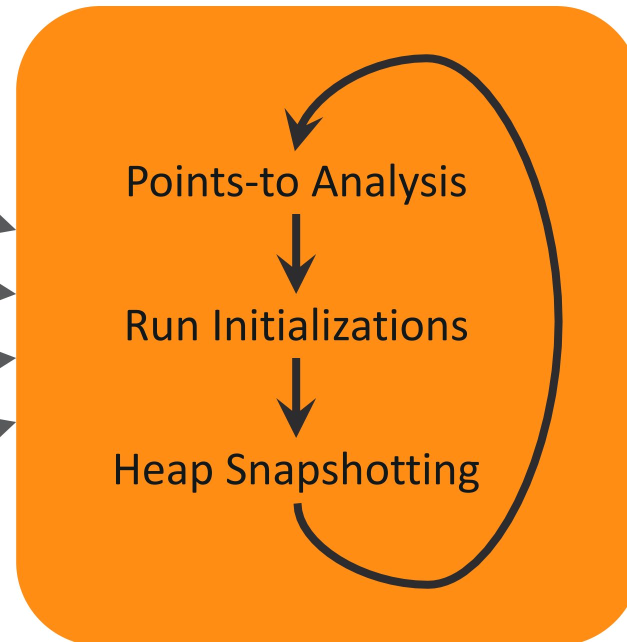
native-image MyMainClass
./mymainclass

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

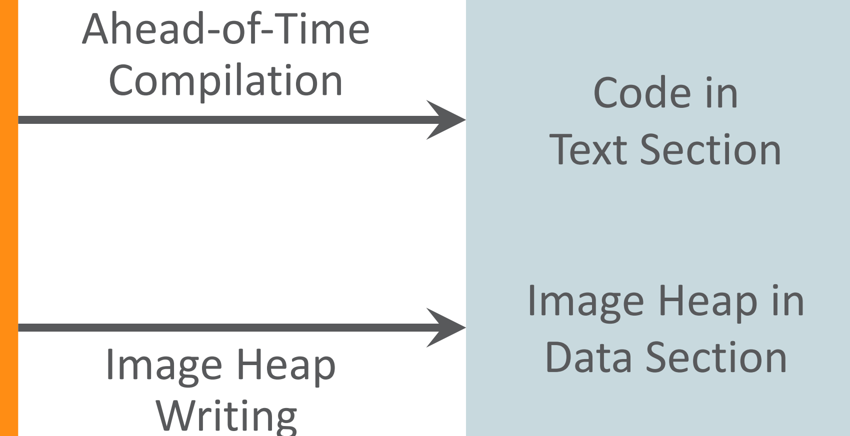
# GraalVM AOT for Native Images

Input:
All classes from application,
libraries, and VM




Iterative analysis until
fixed point is reached

Output:
Native executable




ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# AOT vs JIT: Startup Time

- JIT
  - Load JVM executable
  - Load classes from file system
  - Verify bytecodes
  - Start interpreting
  - Run static initializers
  - First tier compilation (C1)
  - Gather profiling feedback
  - Second tier compilation (GraalVM or C2)
  - Finally run with best machine code
- AOT
  - Load executable with prepared heap
  - Immediately start with best machine code

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# AOT vs JIT: Startup Time

| | GraalVM AOT | GraalVM JIT |
|---|---|---|
| Helidon | 35 ms | 988 ms |
| Micronaut | 37 ms | 2101 ms |
| Quarkus | 16 ms | 940 ms |

0 ms 500 ms 1000 ms 1500 ms 2000 ms 2500 ms

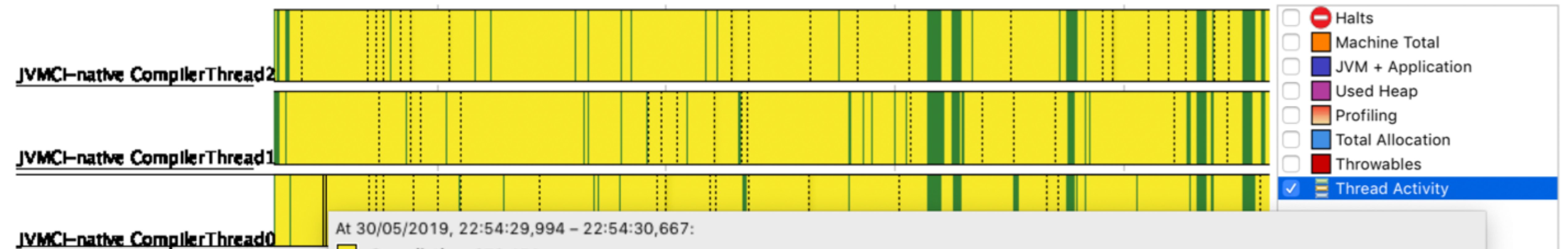
Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# Java Flight Recorder Compilation Information

Java Application

<No Selection> Aspect: <No Selection> Show concurrent: Contained Same threads Time Range: Set Clear

| Thread | Profiling Samples | Total I/O Time | Total Blocked Time | Class Loading Time | Total Allocati |
|---|---|---|---|---|---|
| Reference Handler | 10 | | | | |
| JVMCI-native CompilerThread2 | | | | | 34,4 M |
| JVMCI-native CompilerThread1 | | | | | 34,8 M |
| JVMCI-native CompilerThread0 | | | | | 23,6 M |
| JFR request timer | 2 | | | | 10,3 M |




At 30/05/2019, 22:54:29,994 – 22:54:30,667:

**Compilation**: 673,451 ms
Thread: JVMCI-native CompilerThread0
Java Method: Object org.scalastyle.scalariform.VisitorHelper$$anonfun$org$scalastyle$scalariform$VisitorHelper$$myVisit$1.apply(Object)
Compilation ID: 9.633
Compilation Level: 4
Succeeded: true
On Stack Replacement: false
Compiled Code Size: 35,2 KiB
Inlined Code Size: 11,3 KiB

At -∞ – ∞:

Thread Lifespan of JVMCI-native CompilerThread0: N/A

Stack Trace

Stack Trace
- Pattern$CharProperty java.util.regex.Pa
- Pattern$Node java.util.regex.Pattern.se
- Pattern$Node java.util.regex.Pattern.exp
- Pattern$Node java.util.regex.Pattern.group0() 3178

# AOT vs JIT: Memory Footprint

- JIT
  - Loaded JVM executable
  - Application data
  - Loaded bytecodes
  - Reflection meta-data
  - Code cache
  - Profiling data
  - JIT compiler data structures
- AOT
  - Loaded application executable
  - Application data

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# AOT vs JIT: Memory Footprint

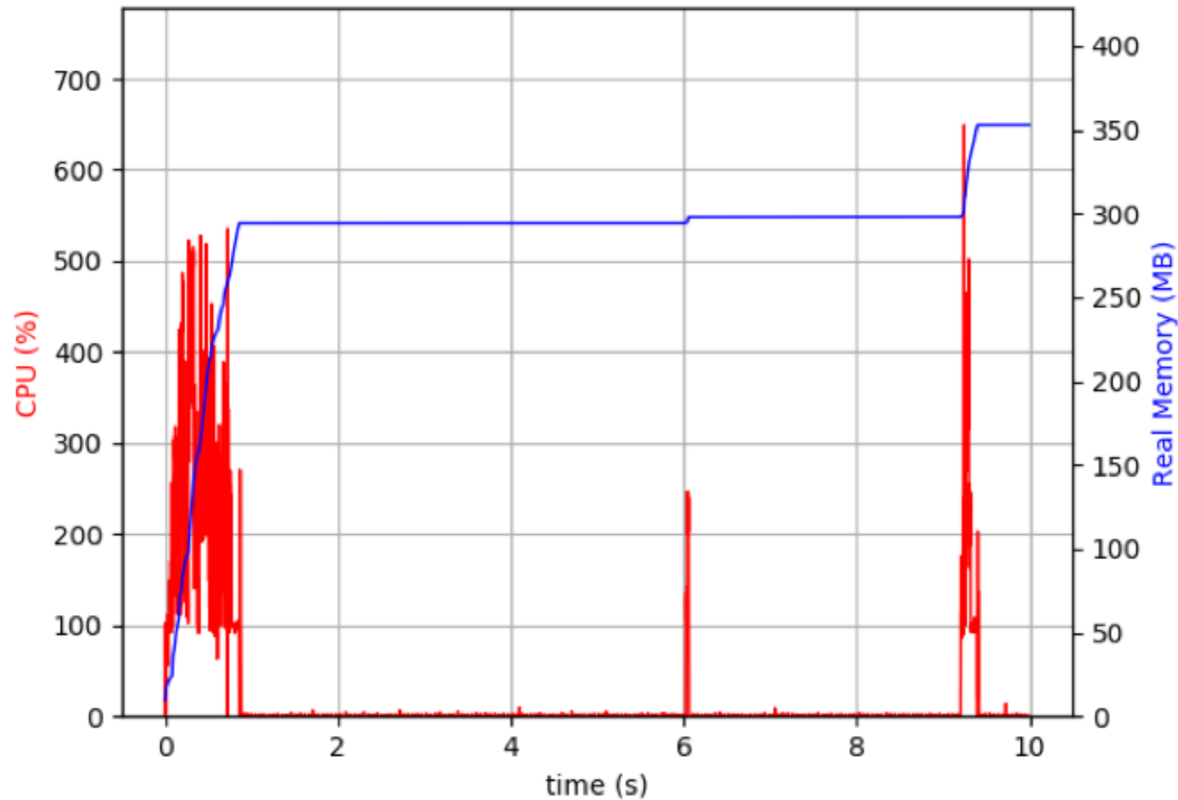
| | GraalVM AOT | GraalVM JIT |
|---|---|---|
| Helidon | 31 MB | 106 MB |
| Micronaut | 41 MB | 180 MB |
| Quarkus | 17 MB | 121 MB |

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

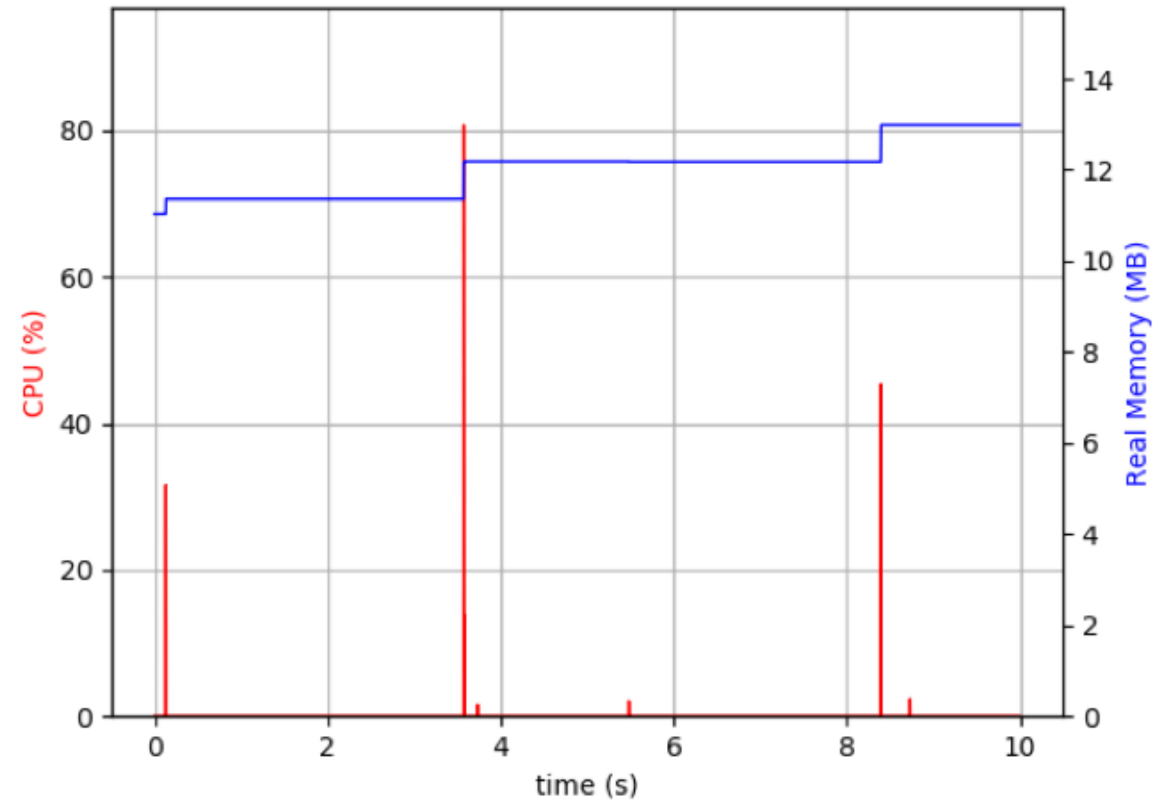
# Web Server Startup and Memory Footprint

Starting up and serving 2 requests in the first 10s

- **JIT 800ms / 350Mb**




- **AOT 8ms / 13Mb**




ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# Which is fastest?

```
int negate1(int a) {
    return -a;
}

int negate2(int a) {
    int b = a + 0;
    return -b * 1;
}

int negate3(int a) {
    Object[] array = new Object[] {Integer.valueOf(a)};
    return -(Integer)array[0];
}
```

```
static Object[] cachedArray = new Object[1];
int negate4(int a) {
    cachedArray[0] = Integer.valueOf(a);
    return -(Integer)cachedArray[0];
}
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# Performance is hard to measure

↓ apparat (ms)

metric.value:AVG

6000 5000 4000 3000 2000 1000 0

12/02/2019 20:33:20 | 24/02/2019 10:20:00 | 08/03/2019 00:06:40 | 19/03/2019 13:53:20 | 31/03/2019 04:40:00 | 11/04/2019 18:26:40 | 23/04/2019 08:13:20 | 04/05/2019 22:00:00 | 16/05/2019 11:46:40 | 28/05/2019 01:33:20 | 08/06/2019 15:20:00 | 20/06/2019 05:06:40

commit.committer-ts

commit.committer-ts 1561363624:
▲server,graal-enterprise: ▲ 5506.774193548387

↓ apparat (ms)

metric.value:AVG

15000 10000 5000 0

17/05/2017 07:46:40 | 14/07/2017 04:40:00 | 10/09/2017 01:33:20 | 06/11/2017 21:26:40 | 03/01/2018 18:20:00 | 02/03/2018 15:13:20 | 29/04/2018 13:06:40 | 26/06/2018 10:00:00 | 23/08/2018 06:53:20 | 20/10/2018 03:46:40 | 16/12/2018 23:40:00 | 12/02/2019 20:33:20 | 11/04/2019 18:26:40 | 08/06/2019 15:20:00

commit.committer-ts

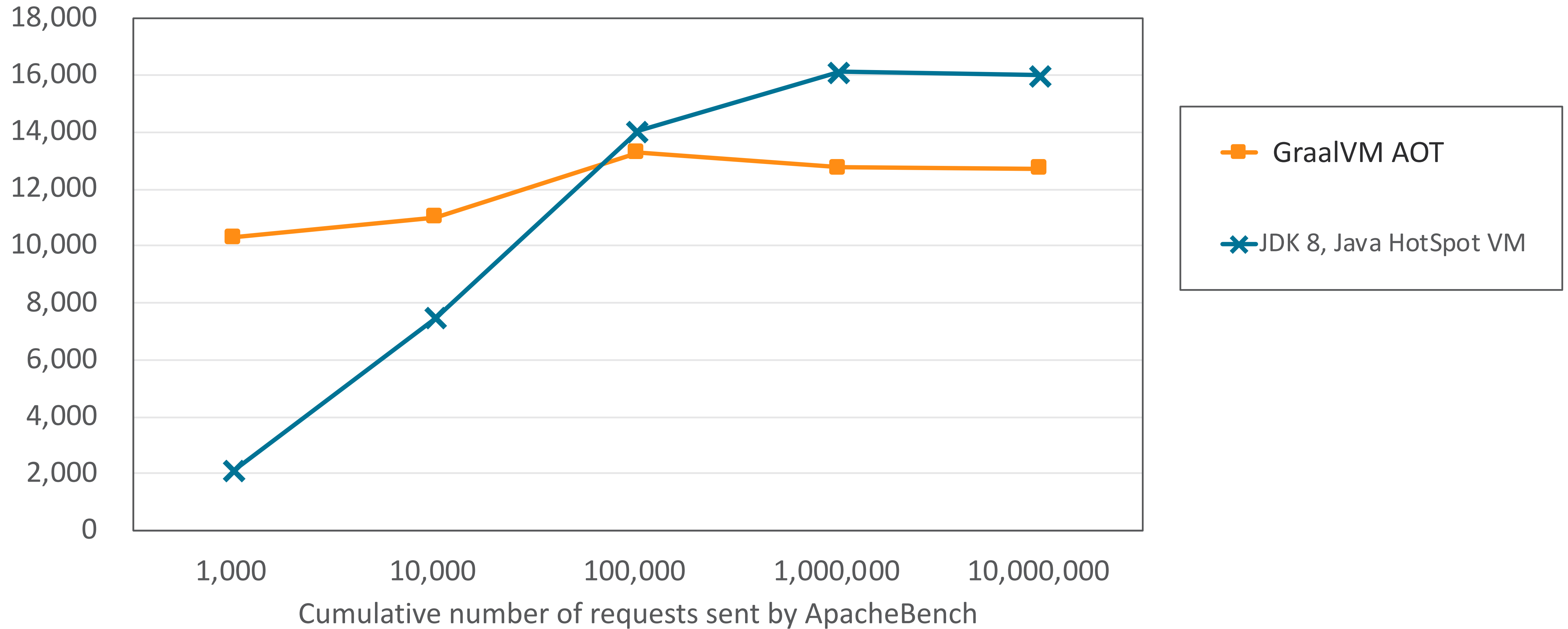
commit.committer-ts 1561363624:
▲server,graal-enterprise: ▲ 5506.774193548387

click to pin at 24/06/2019 10:07:04, and show more info
server,graal-enterprise ▲ Value: 5506.774193548387 ms ± 182.12 ms (type: n/a) Δ: -42.645 ms Δ%: -0.77%

ORACLE

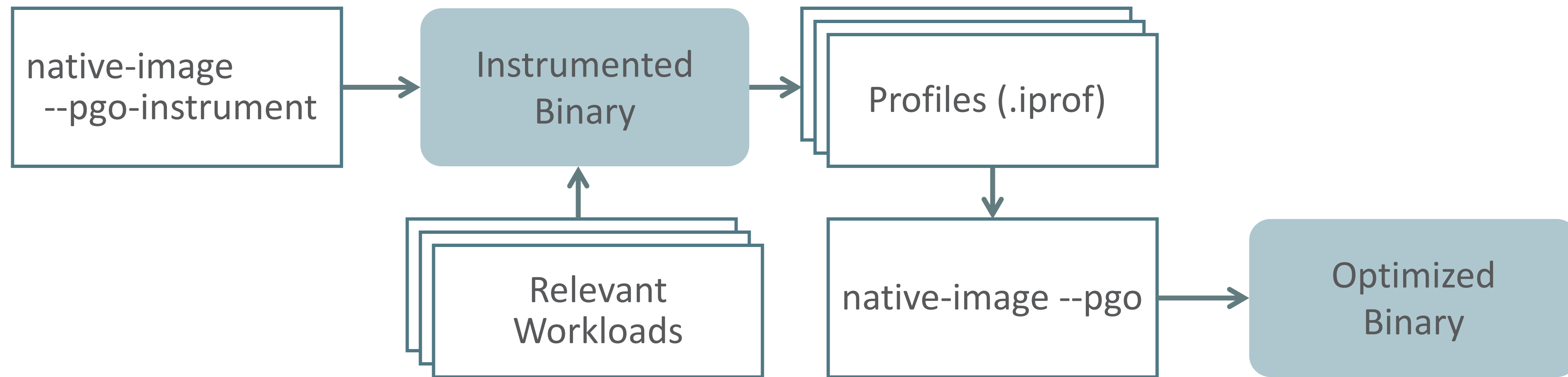
Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# AOT vs JIT: Throughput




ORACLE

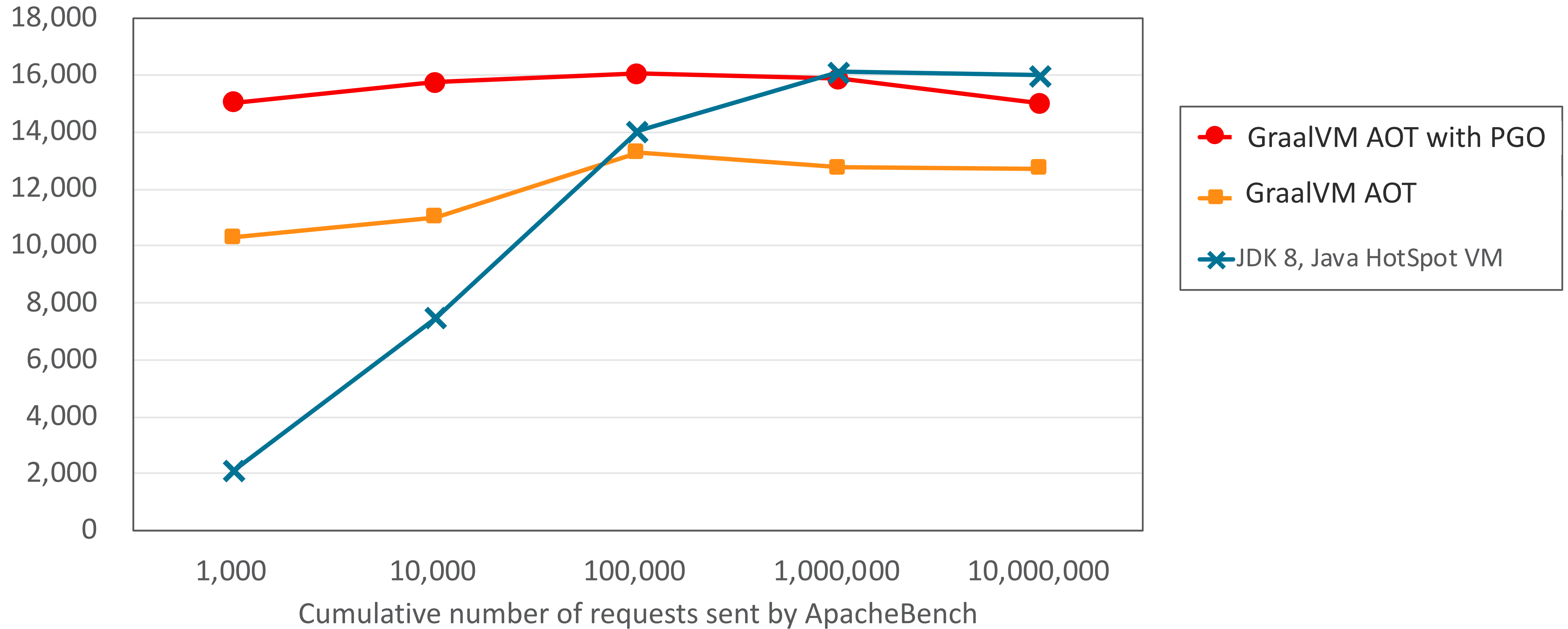
# Profile-Guided Optimizations (PGO)




ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# AOT vs JIT: Throughput




ORACLE

# AOT vs JIT: Peak Performance

- JIT
  - Profiling at startup enabled better optimizations
  - Can make optimistic assumptions about the profile and deoptimize
- AOT
  - Needs to handle all cases in machine code
  - Profile-guided optimizations help
  - Predictable performance

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

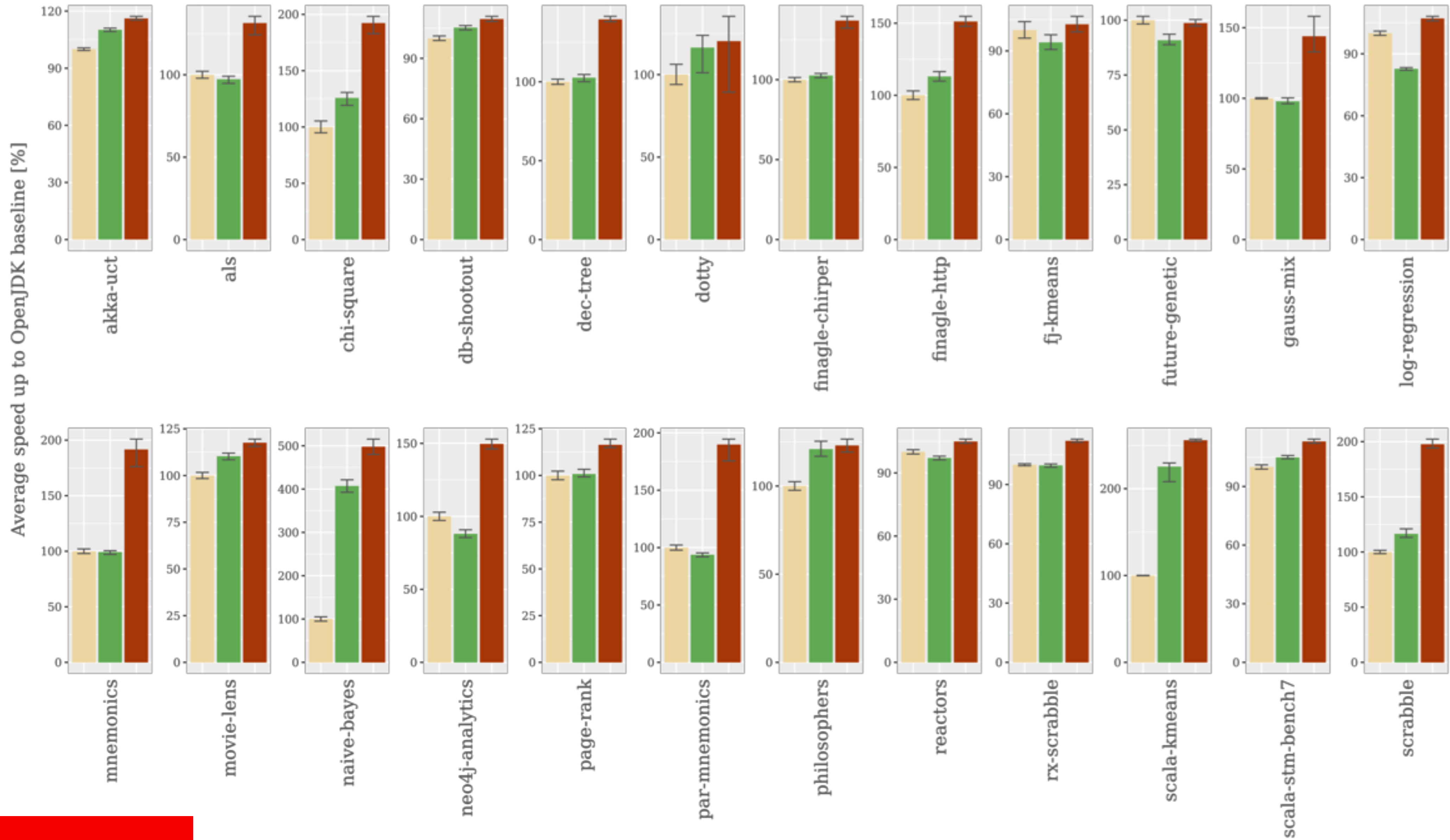
# More Benchmarks…

- Optimizing a compiler for too few benchmarks results in typical overfitting problems
- Therefore we started together with academic collaborators https://renaissance.dev
- All benchmark data can be interesting; careful with conclusions though.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# Rennaissance.dev




JVM Implementation OpenJDK 8 GraalVM 19 CE GraalVM 19 Enterprise

ORACLE

# AOT vs JIT: Max Latency

- JIT
  - Many low latency GC options available
    - G1
    - CMS
    - ZGC
    - Shenandoah
- AOT
  - Only regular stop© collector
  - Assumes small heap configuration
  - Can quickly restart; could use load balancer instead of GC

- Peak vs max latency trade-offs:
  - Loop safepoints
  - Parallel stop-the-world GC

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# AOT vs JIT: Packaging Size

- JIT
  - Use jlink for smaller package
  - Lightweight docker image (e.g., alpine linux)
- AOT
  - Everything in single binary
  - Can run on bare metal docker
  - Substantially smaller constant overhead

- Peak vs packaging trade-offs:
  - Inlining
  - Code duplication

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# GraalVM JIT

Peak Throughput

Max Latency

No Configuration

# GraalVM AOT

Startup Time

Memory Footprint

Packaging Size

## Can AOT get better?

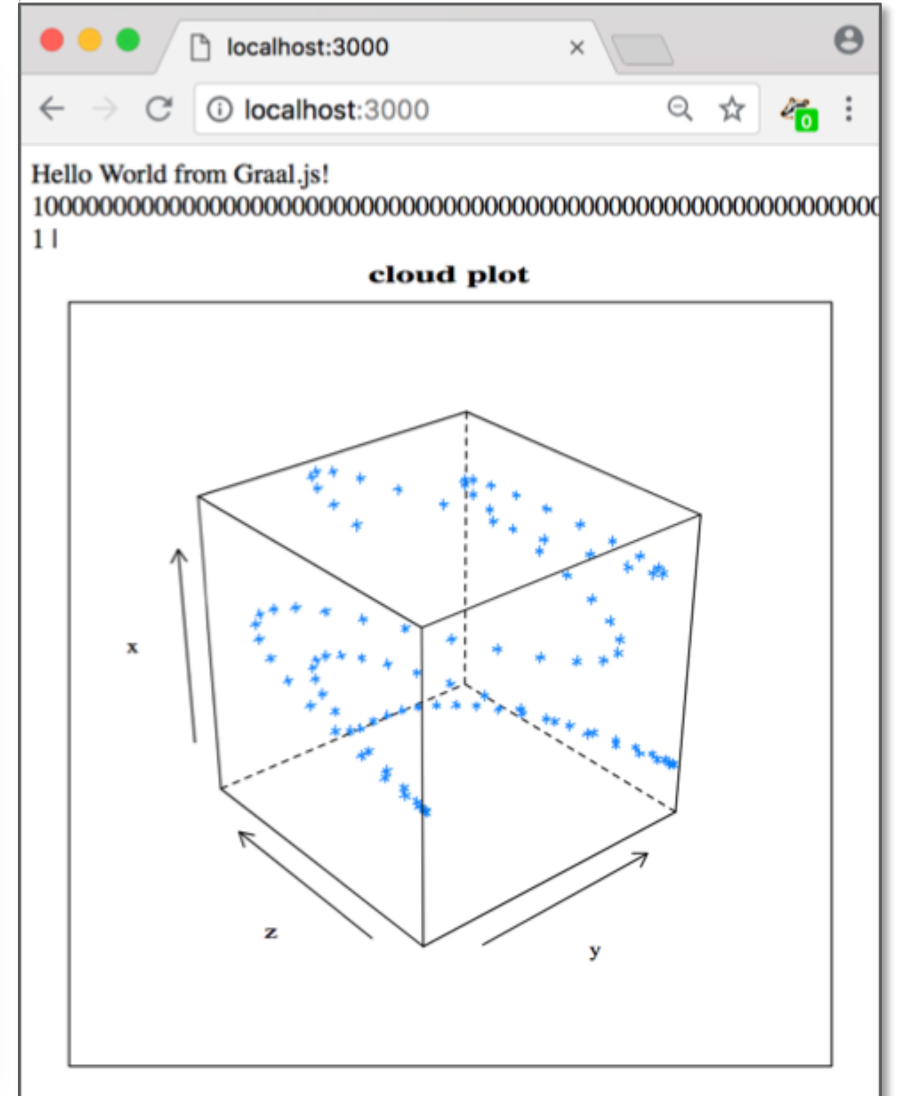
- Collecting profiles up-front
- Low-latency GC option
- Tracing agent for configuration

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# GraalVM can do much more…

```
const express = require('express');
const app = express();
app.listen(3000);
app.get('/', function(req, res) {
  var text = 'Hello World!';
  const BigInteger = Java.type(
    'java.math.BigInteger');
  text += BigInteger.valueOf(2)
    .pow(100).toString(16);
  text += Polyglot.eval(
    'R', 'runif(100)')[0];
  res.send(text);
})
```



# Multiplicative Value-Add of GraalVM Ecosystem

**Add your own language or emedding or language-agnostic tools!**

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# GraalVM Community

- https://www.graalvm.org
- Open source on GitHub at https://github.com/oracle/graal

oracle / graal

Unwatch 376 | Unstar 8,448 | Fork 528

Code | Issues 265 | Pull requests 31 | Insights | Settings

GraalVM: Run Programs Faster Anywhere 🚀 https://www.graalvm.org

Edit

polyglot vm java javascript python r ruby c Manage topics

34,809 commits | 7 branches | 126 releases | 1 environment | 124 contributors | View license

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

# Q/A

@graalvm

@thomaswue

ORACLE

# Integrated Cloud

## Applications & Platform Services

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

ORACLE®