FOSDEM 2020
G1: To Infinity and Beyond

Stefan Johansson
Principal Member of Technical Staff
HotSpot GC-team, Oracle
Twitter: @kstefanj
Safe Harbor

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.

Statements in this presentation relating to Oracle’s future plans, expectations, beliefs, intentions and prospects are “forward-looking statements” and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle’s Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading “Risk Factors.” These filings are available on the SEC’s website or on Oracle’s website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.
Agenda

• GC in OpenJDK
• Short intro to G1
• Progress since JDK 8
• The future
GC in OpenJDK
Introduction
GC in OpenJDK

• Fast allocation
• Efficient reclamation
• Concepts and tradeoffs
  • Throughput
  • Latency
  • Footprint
## Current Collectors

**GC in OpenJDK**

<table>
<thead>
<tr>
<th>Collector name</th>
<th>Status</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>Supported</td>
<td>Low memory footprint</td>
</tr>
<tr>
<td>Parallel</td>
<td>Supported</td>
<td>Throughput oriented</td>
</tr>
<tr>
<td>G1</td>
<td>Supported/Default</td>
<td>Balanced performance</td>
</tr>
<tr>
<td>ZGC</td>
<td>Experimental</td>
<td>Low latency</td>
</tr>
<tr>
<td>Shenandoah</td>
<td>Experimental</td>
<td>Low latency</td>
</tr>
<tr>
<td>CMS</td>
<td>Removed in JDK 14</td>
<td>Unmaintained</td>
</tr>
</tbody>
</table>
Short intro to G1
Basic idea
Short intro to G1

- Balance between throughput and latency
- Region based
- Concurrent marking
- Main tuning knob: pause-time goal
Current status

Short intro to G1

- Supported since JDK 7u4
- Default since JDK 9
- Stable and mature
- Continuously being improved
Progress since JDK 8
Overview
Progress since JDK 8

• 700 enhancements since JDK 8
• Significant improvements
• Across all areas
• Cut away old tradeoffs
Throughput
Progress since JDK 8

- Improved NUMA awareness
- More efficient concurrent work
- Parallel Full collection
Throughput
Progress since JDK 8

- SPECjbb® 2015 results
- 16GB heap
- Not only GC improvements
- More time spent running Java
Latency
Progress since JDK 8

- Improved parallelism
- More efficient reference processing
- Better pause time predictions
- Abortable mixed collections
Latency
Progress since JDK 8

- SPECjbb® 2015 results
- 16GB heap
- Mostly GC improvements
- Trade throughput for latency
Footprint
Progress since JDK 8

• Rebuild remembered sets concurrently
• Improved sizing ergonomics
• Return memory to the operating system
Footprint
Progress since JDK 8

- BigRamTester
- 16GB heap
The Future
Investigations
The Future

• Humongous object handling
• Improved write barrier
• Footprint reductions
Massive improvements since JDK 8
Exciting features in the pipe