ZGC: The Future of Low-Latency Garbage Collection Is Here

Per Liden
Consulting Member of Technical Staff, Oracle
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.
A Scalable Low-Latency Garbage Collector
Properties

Max GC pause time: 1ms
Multi-terabyte heaps: TB
What’s the Catch?

Expect some reduction in throughput
Easy to tune!
# GC Landscape
Oracle supported garbage collectors

<table>
<thead>
<tr>
<th>GC</th>
<th>Optimized For</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>Memory Footprint</td>
</tr>
<tr>
<td>Parallel</td>
<td>Throughput</td>
</tr>
<tr>
<td>G1</td>
<td>Throughput/Latency Balance</td>
</tr>
<tr>
<td>ZGC</td>
<td>Low Latency</td>
</tr>
</tbody>
</table>
ZGC at a Glance

Concurrent Tracing Compacting Single generation

Region-based NUMA-aware Load barriers Colored pointers
ZGC pauses are $O(1)$
Available on All Commonly Used Platforms

- **Linux**: x86 / Arm / PPC
- **Windows**: x86 / Arm
- **macOS**: x86 / Arm (64-bit)
Production Ready since JDK 15
Performance
SPECjbb2015 – Benchmark Score

(Higher is better)

ZGC

G1

max-jOPS (Throughput score)

critical-jOPS (Latency score)

128G Heap

40 Hyper-threads (Intel)

Copyright © 2022, Oracle and/or its affiliates

SPECjbb®2015 is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC’s requirements for general availability.
SPECjbb2015 – GC Pause Times

(Lower is better)

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>95th percentile</th>
<th>99th percentile</th>
<th>99.9th percentile</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZGC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

128G Heap
40 Hyper-threads (Intel)

SPECjbb®2015 is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC’s requirements for general availability.

Copyright © 2022, Oracle and/or its affiliates
SPECjbb2015 - GC Pause Times

1000 x Zoom

GC Pause Times (µs)

- Average
- 95th percentile
- 99th percentile
- 99.9th percentile
- Max

ZGC

G1

128G Heap
40 Hyper-threads (Intel)

Copyright © 2022, Oracle and/or its affiliates

SPECjbb®2015 is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC's requirements for general availability.
SPECjbb2015 – GC Pause Times

(Lower is better)

<table>
<thead>
<tr>
<th>GC Pause Times (µs)</th>
<th>SPECjbb2015 – GC Pause Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZGC</td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td></td>
</tr>
</tbody>
</table>

128G Heap
40 Hyper-threads (Intel)

Average
95th percentile
99th percentile
99.9th percentile
Max

Copyright © 2022, Oracle and/or its affiliates

SPECjbb®2015 is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC’s requirements for general availability.
BigRamTester – GC Pause Times
Lots of heap fragmentation

(Lower is better)

<table>
<thead>
<tr>
<th>GC Pause Times (ms)</th>
<th>ZGC</th>
<th>G1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95th percentile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99th percentile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.9th percentile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16G Heap
32 Hyper-threads (Intel)

Copyright © 2022, Oracle and/or its affiliates
BigRamTester - GC Pause Times

Lots of heap fragmentation

1000 x Zoom

GC Pause Times (µs)

- Average
- 95th percentile
- 99th percentile
- 99.9th percentile
- Max

16G Heap
32 Hyper-threads (Intel)

Lots of heap fragmentation
BigRamTester – GC Pause Times
Lots of heap fragmentation

(Lower is better)

16G Heap
32 Hyper-threads (Intel)
ZGC Improvements Over Time
SPECjbb2015 – Benchmark Score

(JDK 11) ZGC Improvements Over Time
SPECjbb2015 – Benchmark Score

(Higher is better)

max-jOPS (Throughput score)
critical-jOPS (Latency score)

128G Heap
40 Hyper-threads (Intel)

Copyright © 2022, Oracle and/or its affiliates

SPECjbb®2015 is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC’s requirements for general availability.
ZGC Improvements Over Time
SPECjbb2015 – GC Pause Times

(Lower is better)

128G Heap
40 Hyper-threads (Intel)

Copyright © 2022, Oracle and/or its affiliates

SPECjbb®2015 is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC’s requirements for general availability.
### ZGC Improvements Over Time (Large System)

**SPECjbb2015 – GC Pause Times**

(Lower is better)

- **JDK 11**
- **JDK 15**
- **JDK 17**

**GC Pause Times (ms)**

**Average**

**95th percentile**

**99th percentile**

**99.9th percentile**

**Max**

---

**3T Heap**

224 Hyper-threads (Intel)

---

**ZGC Improvements**

- **Copyright © 2022, Oracle and/or its affiliates**

---

**SPECjbb®2015** is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC's requirements for general availability.
Using ZGC
Enable

-XX:+UseZGC
Set Max Heap Size

-Xmx<size>
Logging

-Xlog:gc  (basic)
-Xlog:gc*  (detailed)
Generational ZGC
The Generational Hypothesis

A very common pattern in Java applications is that most objects are short lived
Heap

Young Generation

Old Generation

New object allocated in the young generation
Object promoted to the old generation
Reduced Effort to Collect Garbage

Withstand higher allocation rates
Lower heap headroom
Lower CPU usage
Heap

Young Generation

Old Generation
ZGC Heap

Y Y Y O Y O O O
Y O O O O Y O
O Y O Y O Y O
Y O Y O O O
Y O Y O O O
Minor & Major Collections
Minor Collection

Young Generation

Heap

Old Generation

Remembered Set

Roots
Major Collection
Major Collection

Heap

Young Generation

Old Generation
Heap

Major Collection

Young Generation

Old Generation

Roots
Dynamic generation sizing

(No \(-Xmn\) needed)
Dynamic tenuring threshold

(No -XX:TenuringThreshold needed)
Dynamic number of threads

(No \texttt{-XX:ConcGCThreads} needed)
Just set the max heap size!

(\(-Xmx\))
Preliminary Benchmarks
Memory Needed to Maintain Low Latency
Using the SPECjbb2005 benchmark with 1.5G live-set

Memory Usage

GB

Non-generational ZGC

Generational ZGC

60%
Benchmark Score
Using the SPECjbb2005 benchmark with 1.5G live-set

![Benchmark Score Chart]

- **Non-generational (15G)**
- **Generational (6G)**

SPECjbb®2005 is a registered trademark of the Standard Performance Evaluation Corporation (spec.org). The actual results are not represented as compliant because the SUT may not meet SPEC’s requirements for general availability.
Memory Needed to Maintain Low Latency

Using the Extremem benchmark

Memory Usage

<table>
<thead>
<tr>
<th>GB</th>
<th>Non-generational ZGC</th>
<th>Generational ZGC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>15</td>
</tr>
</tbody>
</table>

70%
CPU Usage When Given the Same Amount of Memory
Using the ExtremeMem benchmark

- Non-generational ZGC
- Generational ZGC

CPU Usage

Seconds

- Non-generational ZGC: 140
- Generational ZGC: 60

CPU Usage: 58%
More Information
ZGC Wiki

The Z Garbage Collector, also known as ZGC, is a scalable low-latency garbage collector designed to meet the following goals:

- Sub-millisecond pause times
- Pause times do not increase with the heap, live-set or cost-set size
- Resilient heaps ranging from a 1MB to 16TB in size

ZGC was initially introduced as an experimental feature in JDK 11, and was declared Production Ready in JDK 16.

For more information, visit the OpenJDK Wiki page on ZGC at https://wiki.openjdk.java.net/display/zgc
ZGC | What's new in JDK 17
05 Oct 2021

JDK 17 was released on September 14. This is a Long-Term Support (LTS) release, meaning it will be supported and receive updates for many years. This is also the first LTS release where a production ready version of ZGC is...

ZGC | What's new in JDK 16
22 Mar 2021

JDK 16 is out, and as usual, each new release comes with a bunch of new features, enhancements and bug fixes. ZGC received 46 enhancements and 25 bug fixes. Here I'll cover a few of the more interesting enhancements. Sub-millisecond...

ZGC | Inside Java Podcast
18 Oct 2020

I had the pleasure of being invited to the Inside Java Podcast, where David Delabassee and I talked about ZGC. We covered some of the things that is new in JDK 15 as well as what's coming in JDK 16...
Thanks!

Per Liden
E-mail: per.liden@oracle.com
Twitter: @perliden
Blog: malloc.se
Download Java 18 Now
oracle.com/javadownload

#OracleDevLive