The Z Garbage Collector
Scalable Low-Latency GC in JDK 11

Erik Österlund
Garbage Collection Engineer
Java Platform Group, Oracle
November 16, 2018
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, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

1. What is ZGC?
2. A Peek Under the Hood
3. Performance
4. Using ZGC
5. Future Plans
What is ZGC?
New Garbage Collector in JDK 11

(Experimental feature, Linux/x86_64 only)
A Scalable Low-Latency Garbage Collector
Goals

- **TB**: Multi-terabyte heaps
- **10ms**: Max GC pause time
- **Easy to tune**
- **15%**: Max application throughput reduction
## ZGC at a Glance

<table>
<thead>
<tr>
<th>Concurrent Tracing</th>
<th>Region-based NUMA-aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compacting</td>
<td>Load barriers</td>
</tr>
<tr>
<td>Single generation</td>
<td>Colored pointers</td>
</tr>
</tbody>
</table>
ZGC pause times **do not** increase with the heap or live-set size
ZGC pause times do increase with the root-set size

(Number of Java Threads)
## Concurrent?

<table>
<thead>
<tr>
<th>Process</th>
<th>Serial</th>
<th>Parallel</th>
<th>CMS</th>
<th>G1</th>
<th>ZGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relocation/Compaction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relocation Set Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StringTable Cleaning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JNI WeakRef Cleaning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JNI GlobalRefs Scanning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class Unloading</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thread Stack Scanning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Concurrent?

<table>
<thead>
<tr>
<th></th>
<th>Serial</th>
<th>Parallel</th>
<th>CMS</th>
<th>G1</th>
<th>ZGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marking</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relocation/Compaction</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Processing</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relocation Set Selection</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StringTable Cleaning</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JNI WeakRef Cleaning</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JNI GlobalRefs Scanning</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class Unloading</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thread Stack Scanning</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Concurrent?

<table>
<thead>
<tr>
<th></th>
<th>Serial</th>
<th>Parallel</th>
<th>CMS</th>
<th>G1</th>
<th>ZGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marking</td>
<td></td>
<td></td>
<td>✔*</td>
<td>✔*</td>
<td></td>
</tr>
<tr>
<td>Relocation/Compaction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relocation Set Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StringTable Cleaning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JNI WeakRef Cleaning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JNI GlobalRefs Scanning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class Unloading</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thread Stack Scanning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) Old Gen Only
## Concurrent?

<table>
<thead>
<tr>
<th></th>
<th>Serial</th>
<th>Parallel</th>
<th>CMS</th>
<th>G1</th>
<th>ZGC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Marking</strong></td>
<td>-</td>
<td>-</td>
<td>✔*</td>
<td>✔*</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Relocation/Compaction</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Reference Processing</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Relocation Set Selection</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td><strong>StringTable Cleaning</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td><strong>JNI WeakRef Cleaning</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td><strong>JNI GlobalRefs Scanning</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔**</td>
</tr>
<tr>
<td><strong>Class Unloading</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔**</td>
</tr>
<tr>
<td><strong>Thread Stack Scanning</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

*) Old Gen Only
**) Post JDK 11
A Peek Under the Hood
ZGC Phases

Pause Mark Start

Pause Mark End

Pause Relocate Start

Concurrent Mark/Remap

Concurrent Prepare for Reloc.

Concurrent Relocate

GC Cycle
ZGC Phases

Pause Mark Start

Concurrent Mark/Remap

Pause Mark End

Concurrent Prepare for Reloc.

Pause Relocate Start

Concurrent Relocate

Scan thread stacks

GC Cycle
ZGC Phases

Pause Mark Start
Concurrent Mark/Remap

Pause Mark End
Concurrent Prepare for Reloc.

Pause Relocate Start
Concurrent Relocate

Walk object graph

GC Cycle
ZGC Phases

Pause Mark Start
- Concurrent Mark/Remap

Pause Mark End
- Concurrent Prepare for Reloc.

Pause Relocate Start
- Concurrent Relocate

Synchronization point

GC Cycle
ZGC Phases

- **Pause Mark Start**
  - Concurrent Mark/Remap

- **Pause Mark End**
  - Concurrent Prepare for Reloc.

- **Pause Relocate Start**
  - Concurrent Relocate

Reference processing
Weak root cleaning
Relocation set selection
ZGC Phases

Pause Mark Start
Concurrent Mark/Remap

Pause Mark End
Concurrent Prepare for Reloc.

Pause Relocate Start
Concurrent Relocate

GC Cycle
Scan thread stacks
ZGC Phases

Pause Mark Start → Concurrent Mark/Remap

Pause Mark End → Concurrent Prepare for Reloc.

Pause Relocate Start → Concurrent Relocate

GC Cycle

Compact heap
ZGC Phases

Pause Mark Start: 
- Concurrent Mark/Remap

Pause Mark End: 
- Concurrent Prepare for Reloc.

Pause Relocate Start: 
- Concurrent Relocate

GC Cycle
ZGC Phases

- Pause Mark Start
  - Concurrent Mark/Remap
  - < 10 ms

- Pause Mark End
  - Concurrent Prepare for Reloc.
  - < 10 ms

- Pause Relocate Start
  - Concurrent Relocate
  - < 10 ms

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.
Colored Pointers

- Core design concept in ZGC
- **Metadata** stored in unused bits in 64-bit pointers
  - No support for 32-bit platforms
  - No support for CompressedOops

![Diagram of Colored Pointers]

- 64-bit Object Pointer
- Unused (18 bits)
- Object Address (42 bits, 4TB address space)
- Colors
Load Barrier

• A small piece of code injected by the JIT in strategic places
  – When loading an object reference from the heap

• Checks if the loaded object reference has a **bad** color
  – If so, take **action** and **correct** it
Load Barrier

String n = person.name;       // Loading an object reference from heap
Load Barrier

String n = person.name; // Loading an object reference from heap

<load barrier needed here>
Load Barrier

String n = person.name;  // Loading an object reference from heap
<load barrier needed here>
String p = n;            // No barrier, not a load from heap
n.isEmpty();            // No barrier, not a load from heap
int age = person.age;   // No barrier, not an object reference
Load Barrier

```java
String n = person.name;  // Loading an object reference from heap
```

<load barrier needed here>
Load Barrier

```java
String n = person.name; // Loading an object reference from heap
if (n & bad_bit_mask) {
    slow_path(register_for(n), address_of(person.name));
}
```
Load Barrier

```asm
mov  0x10(%rax), %rbx  // String n = person.name;
test %rbx, (0x16)%r15  // Bad color?
jnz  slow_path        // Yes -> Enter slow path and
                      // mark/relocate/remap, adjust
                      // 0x10(%rax) and %rbx
```
Load Barrier

```assembly
mov 0x10(%rax), %rbx      // String n = person.name;
test %rbx, (0x16)%r15    // Bad color?
jnz slow_path            // Yes -> Enter slow path and
                          // mark/relocate/remap, adjust
                          // 0x10(%rax) and %rbx
```

~4% execution overhead on SPECjbb®2015
Performance
SPECjbb® 2015 – Score

Mode: Composite
Heap Size: 128G
OS: Oracle Linux 7.4
HW: Intel Xeon E5-2690 2.9GHz
2 sockets, 16 cores (32 hw-threads)

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.

(max-JOPS (Throughput)  critical-JOPS (Throughput with latency requirements))
SPECjbb® 2015 – Score

Mode: Composite
Heap Size: 128G
OS: Oracle Linux 7.4
HW: Intel Xeon E5-2690 2.9GHz
2 sockets, 16 cores (32 hw-threads)

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.
SPECjbb®2015 – Pause Times

Linear scale
(Lower is better)

Average 95th percentile 99th percentile 99.9th percentile Max

ZGC Parallel G1

GC Pause Times (ms)
SPECjbb®2015 – Pause Times

Logarithmic scale
(Lower is better)

GC Pause Times (ms)

ZGC  Parallel  G1

Average  95th percentile  99th percentile  99.9th percentile  Max

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.
## SPECjbb®2015 – Pause Times

<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>1.10</td>
<td>1.96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel</td>
<td>211</td>
<td>306</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td>156</td>
<td>544</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Logarithmic scale**
(Lower is better)
SPECjbb® 2015 – Pause Times

Logarithmic scale
(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>1.10</td>
<td>1.96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel</td>
<td>211</td>
<td>306</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td>156</td>
<td>544</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using ZGC
Enable

-XX:+UseZGC
Enable

-XX:+UnlockExperimentalVMOptions
-XX:+UseZGC
Tuning
Tuning
Set Max Heap Size

-Xmx<size>
Tuning

Maybe Set Number of Concurrent GC Threads

-Xmx<size>

-XX:ConcGCThreads=<number>
Tuning

That’s it?

-Xmx<size>
-XX:ConcGCThreads=<number>
Logging

-xlog:gc (basic)
-xlog:gc* (detailed)
Garbage Collection (Proactive) 13426M(10%)->2492M(2%)
Garbage Collection (Allocation Rate) 87676M(67%)->19578M(15%)
Garbage Collection (Allocation Rate) 55302M(42%)->17646M(13%)
Garbage Collection (Allocation Rate) 61794M(47%)->26794M(20%)
Garbage Collection (Allocation Rate) 60856M(46%)->31926M(24%)
Garbage Collection (Allocation Rate) 52744M(40%)->38950M(29%)
Garbage Collection (Allocation Rate) 42542M(32%)->32204M(25%)
Garbage Collection (Allocation Rate) 49974M(38%)->8534M(7%)
Garbage Collection (System.gc()) 8534M(7%)->282M(0%)
Garbage Collection (Allocation Rate) 95454M(73%)->25660M(20%)
Garbage Collection (Allocation Rate) 42478M(32%)->23812M(18%)
Garbage Collection (Allocation Rate) 56714M(43%)->29909M(22%)
Garbage Collection (Allocation Rate) 62802M(48%)->28648M(22%)
Garbage Collection (Allocation Rate) 59748M(46%)->23770M(18%)
Garbage Collection (Allocation Rate) 74946M(57%)->23284M(18%)
Garbage Collection (System.gc()) 44902M(34%)->422M(0%)
Garbage Collection (Allocation Rate) 94510M(72%)->20456M(16%)
Garbage Collection (Allocation Rate) 59694M(46%)->25834M(20%)
Garbage Collection (Allocation Rate) 63494M(48%)->29128M(22%)
Garbage Collection (Allocation Rate) 59034M(45%)->27948M(21%)
Garbage Collection (Allocation Rate) 66110M(50%)->25278M(19%)
Garbage Collection (Allocation Rate) 73410M(56%)->27968M(21%)
Garbage Collection (Allocation Rate) 76010M(53%)->32236M(25%)
Garbage Collection (Allocation Rate) 64444M(49%)->27612M(21%)
Garbage Collection (Allocation Rate) 64484M(49%)->29910M(23%)
Garbage Collection (Allocation Rate) 64128M(49%)->33184M(25%)
Garbage Collection (Allocation Rate) 59148M(45%)->27800M(21%)
Garbage Collection (Allocation Rate) 63104M(48%)->27976M(21%)
Garbage Collection (Allocation Rate) 64418M(49%)->34390M(26%)
Garbage Collection (Allocation Rate) 52284M(40%)->30654M(23%)
Garbage Collection (Allocation Rate) 58746M(45%)->32028M(24%)
Garbage Collection (Allocation Rate) 59468M(45%)->32804M(25%)
Garbage Collection (Allocation Rate) 53342M(41%)->18436M(14%)
Garbage Collection (Proactive) 13426M(10%)-&gt;2492M(2%)
Garbage Collection (Allocation Rate) 87676M(67%)-&gt;19578M(15%)
Garbage Collection (Allocation Rate) 55302M(42%)-&gt;17646M(13%)
Garbage Collection (Allocation Rate) 61794M(47%)-&gt;26794M(26%)
Garbage Collection (Allocation Rate) 60856M(46%)-&gt;31926M(24%)
Garbage Collection (Allocation Rate) 52744M(40%)-&gt;38050M(29%)
Garbage Collection (Allocation Rate) 42542M(32%)-&gt;32204M(25%)
Garbage Collection (Allocation Rate) 49974M(38%)-&gt;8534M(7%)
Garbage Collection (System.gc()) 8534M(7%)-&gt;282M(0%)

Garbage Collection (Allocation Rate) 95454M(73%)-&gt;25660M(20%)
Garbage Collection (Allocation Rate) 42478M(32%)-&gt;23812M(18%)
Garbage Collection (Allocation Rate) 56714M(43%)-&gt;29090M(22%)
Garbage Collection (Allocation Rate) 62802M(48%)-&gt;28648M(22%)
Garbage Collection (Allocation Rate) 59748M(46%)-&gt;23770M(18%)
Garbage Collection (Allocation Rate) 74946M(57%)-&gt;23284M(18%)
Garbage Collection (System.gc()) 44902M(34%)-&gt;422M(0%)
Garbage Collection (Allocation Rate) 94510M(72%)-&gt;20456M(16%)
Garbage Collection (Allocation Rate) 59694M(46%)-&gt;25834M(20%)
Garbage Collection (Allocation Rate) 63494M(48%)-&gt;29128M(22%)
Garbage Collection (Allocation Rate) 59034M(45%)-&gt;27094M(21%)
Garbage Collection (Allocation Rate) 66110M(50%)-&gt;25778M(19%)
Garbage Collection (System.gc()) 44902M(34%)-&gt;422M(0%)
Garbage Collection (System.gc()) 53342M(41%)-&gt;18436M(14%)

Garbage Collection (Allocation Rate) 95454M(73%)-&gt;25660M(20%)
Garbage Collection (Allocation Rate)
Pause Mark Start 0.949ms
Concurrent Mark 1151.425ms
Pause Mark End 0.882ms
Concurrent Process Non-Strong References 0.367ms
Concurrent Reset Relocation Set 18.090ms
Concurrent Destroy Detached Pages 0.002ms
Concurrent Select Relocation Set 12.295ms
Concurrent Prepare Relocation Set 70.922ms
Pause Relocate Start 1.419ms
Concurrent Relocate 645.941ms
Load: 15.77/10.68/9.93
MMU: 2ms/0.0%, 5ms/57.5%, 10ms/78.7%, 20ms/87.8%, 50ms/93.7%, 100ms/96.8%
Mark: 4 stripe(s), 2 proactive flush(es), 1 terminate flush(es), 0 completion(s), 0 continuation(s)
Relocation: Successful, 867M relocated
NMethods: 3209 registered, 1559 unregistered
Soft: 466881 encountered, 0 discovered, 0 enqueued
Weak: 5421 encountered, 3526 discovered, 1742 enqueued
Final: 55 encountered, 6 discovered, 0 enqueued
Phantom: 74 encountered, 59 discovered, 0 enqueued

Mark Start          Mark End        Relocate Start      Relocate End           High               Low
Capacity:   131072M (100%)     131072M (100%)     131072M (100%)     131072M (100%)     131072M (100%)     131072M (100%)
Reserve:       72M (0%)           72M (0%)           72M (0%)           72M (0%)           72M (0%)           72M (0%)
Free:     7654M (6%)         5898M (4%)        24018M (18%)      126296M (96%)      126306M (96%)        5866M (4%)
Used:   123346M (94%)      125102M (95%)      106982M (82%)        4704M (4%)       125134M (95%)        4694M (4%)
Live:     -                -               -                 -                -               -
Allocated:   -               -               -                 -                -               -
Garbage:     -               -               -                 -                -               -
Reclaimed:  -               -               -                 -                -               -
Garbage Collection (Allocation Rate) 123346M(94%)- >4704M(4%)
### Garbage Collection (Allocation Rate)

<table>
<thead>
<tr>
<th>Action</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause Mark Start</td>
<td>0.949ms</td>
</tr>
<tr>
<td>Concurrent Mark</td>
<td>1151.425ms</td>
</tr>
<tr>
<td>Pause Mark End</td>
<td>0.882ms</td>
</tr>
<tr>
<td>Concurrent Process Non-Strong References</td>
<td>0.367ms</td>
</tr>
<tr>
<td>Concurrent Reset Relocation Set</td>
<td>18.090ms</td>
</tr>
<tr>
<td>Concurrent Destroy Detached Pages</td>
<td>0.002ms</td>
</tr>
<tr>
<td>Concurrent Select Relocation Set</td>
<td>12.295ms</td>
</tr>
<tr>
<td>Concurrent Prepare Relocation Set</td>
<td>70.922ms</td>
</tr>
<tr>
<td>Pause Relocate Start</td>
<td>1.419ms</td>
</tr>
<tr>
<td>Concurrent Relocate</td>
<td>645.941ms</td>
</tr>
</tbody>
</table>

**Load:** 15.77/10.68/9.93

**MMU:** 2ms/0.0%, 5ms/57.5%, 10ms/78.7%, 20ms/91.6%

**Mark:** 4 stripe(s), 2 proactive flush(es)

**Relocation:** Successful, 867M relocated

**NMethods:** 3209 registered, 1559 unregistered

**Soft:** 466881 encountered, 0 discovered, 0 enqueued

**Weak:** 5421 encountered, 3526 discovered, 1742 enqueued

**Final:** 55 encountered, 6 discovered, 0 enqueued

**Phantom:** 74 encountered, 59 discovered, 0 enqueued

---

**Capacity:** 131072M (100%)

**Reserve:** 72M (0%)

**Free:** 7654M (6%)

**Used:** 123346M (94%)

**Live:** 123182M (95%)

**Allocated:** 121623M (93%)

**Garbage:** 103353M (79%)

**Reclaimed:** 121496M (93%)

**Used:** 4704M (4%)

**Reclaimed:** 4694M (4%)

**Garbage Collection (Allocation Rate)**

- 123346M(94%) -> 4704M(4%)
### Garbage Collection (Allocation Rate)

<table>
<thead>
<tr>
<th></th>
<th>Mark Start</th>
<th>Mark End</th>
<th>Relocate Start</th>
<th>Relocate End</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacity:</strong></td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
</tr>
<tr>
<td><strong>Reserve:</strong></td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
</tr>
<tr>
<td><strong>Free:</strong></td>
<td>7654M (6%)</td>
<td>5898M (4%)</td>
<td>24018M (18%)</td>
<td>126296M (96%)</td>
</tr>
<tr>
<td><strong>Used:</strong></td>
<td>123346M (94%)</td>
<td>125102M (95%)</td>
<td>106982M (82%)</td>
<td>4704M (4%)</td>
</tr>
<tr>
<td><strong>Live:</strong></td>
<td>-</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
</tr>
<tr>
<td><strong>Allocated:</strong></td>
<td>-</td>
<td>1796M (1%)</td>
<td>1962M (1%)</td>
<td>3766M (3%)</td>
</tr>
<tr>
<td><strong>Garbage:</strong></td>
<td>-</td>
<td>121623M (93%)</td>
<td>103353M (79%)</td>
<td>127M (0%)</td>
</tr>
<tr>
<td><strong>Reclaimed:</strong></td>
<td>-</td>
<td>-</td>
<td>18270M (14%)</td>
<td>121496M (93%)</td>
</tr>
</tbody>
</table>

**Mark Start**          **Mark End**          **Relocate Start**       **Relocate End**

- Capacity: 131072M (100%) 131072M (100%) 131072M (100%) 131072M (100%)
- Reserve: 72M (0%) 72M (0%) 72M (0%) 72M (0%)
- Free: 7654M (6%) 5898M (4%) 24018M (18%) 126296M (96%)
- Used: 123346M (94%) 125102M (95%) 106982M (82%) 4704M (4%)
- Live: - 1722M (1%) 1722M (1%) 1722M (1%)
- Allocated: - 1796M (1%) 1962M (1%) 3766M (3%)
- Garbage: - 121623M (93%) 103353M (79%) 127M (0%)
- Reclaimed: - - 18270M (14%) 121496M (93%)

MD: 3209, NMethods: 3209
Soft: 466881 encountered, 0 discovered, 0 enqueued
Weak: 5421 encountered, 3526 discovered, 1742 enqueued
Final: 55 encountered, 6 discovered, 0 enqueued
Phantom: 74 encountered, 59 discovered, 0 enqueued

Load: 15.77/10.68/9.93
MMU: 2ms/0.0%, 5ms/57.5%, 10ms/78.7%, 20ms/87.8%, 50ms/93.7%, 100ms/96.8%
Mark: 4 stripe(s), 2 proactive flush(es), 1 terminate flush(es), 0 completion(s), 0 continuation(s)
Relocation: Successful, 867M relocated
Garbage: 121623M (93%) > 4704M (4%)

---

---
<table>
<thead>
<tr>
<th></th>
<th>Mark Start</th>
<th>Mark End</th>
<th>Relocate Start</th>
<th>Relocate End</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacity</strong></td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
</tr>
<tr>
<td><strong>Reserve</strong></td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
</tr>
<tr>
<td><strong>Free</strong></td>
<td>7654M (6%)</td>
<td>5898M (4%)</td>
<td>24018M (18%)</td>
<td>126296M (96%)</td>
</tr>
<tr>
<td><strong>Used</strong></td>
<td>123346M (94%)</td>
<td>125102M (95%)</td>
<td>106982M (82%)</td>
<td>4704M (4%)</td>
</tr>
<tr>
<td><strong>Live</strong></td>
<td>-</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
</tr>
<tr>
<td><strong>Allocated</strong></td>
<td>-</td>
<td>1796M (1%)</td>
<td>1962M (1%)</td>
<td>3766M (3%)</td>
</tr>
<tr>
<td><strong>Garbage</strong></td>
<td>-</td>
<td>121623M (93%)</td>
<td>103353M (79%)</td>
<td>127M (0%)</td>
</tr>
<tr>
<td><strong>Reclaimed</strong></td>
<td>-</td>
<td>-</td>
<td>18270M (14%)</td>
<td>121496M (93%)</td>
</tr>
</tbody>
</table>

**Mark Start**          **Mark End**            **Relocate Start**      **Relocate End**       **High**            **Low**

<table>
<thead>
<tr>
<th></th>
<th>131072M (100%)</th>
<th>131072M (100%)</th>
<th>131072M (100%)</th>
<th>131072M (100%)</th>
<th>131072M (100%)</th>
<th>131072M (100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reserve</strong></td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
</tr>
<tr>
<td><strong>Free</strong></td>
<td>7654M (6%)</td>
<td>5898M (4%)</td>
<td>24018M (18%)</td>
<td>126296M (96%)</td>
<td>126306M (96%)</td>
<td>5866M (4%)</td>
</tr>
<tr>
<td><strong>Used</strong></td>
<td>123346M (94%)</td>
<td>125102M (95%)</td>
<td>106982M (82%)</td>
<td>4704M (4%)</td>
<td>125134M (95%)</td>
<td>4694M (4%)</td>
</tr>
<tr>
<td><strong>Live</strong></td>
<td>-</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Allocated</strong></td>
<td>-</td>
<td>1796M (1%)</td>
<td>1962M (1%)</td>
<td>3766M (3%)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Garbage</strong></td>
<td>-</td>
<td>121623M (93%)</td>
<td>103353M (79%)</td>
<td>127M (0%)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Reclaimed</strong></td>
<td>-</td>
<td>-</td>
<td>18270M (14%)</td>
<td>121496M (93%)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Garbage Collection (Allocation Rate)**

123346M (94%) -> 4704M (4%)
Garbage Collection (Allocation Rate)
Pause Mark Start 0.949ms
Concurrent Mark 1151.425ms
Pause Mark End 0.882ms
Concurrent Process Non-Strong References 0.367ms
Concurrent Reset Relocation Set 18.090ms
Concurrent Destroy Detached Pages 0.002ms
Concurrent Select Relocation Set 12.295ms
Concurrent Prepare Relocation Set 70.922ms
Pause Relocate Start 1.419ms
Concurrent Relocate 645.941ms
Load: 15.77/10.68/9.93
MMU: 2ms/0.0%, 5ms/57.5%, 10ms/78.7%, 20ms/87.8%, 50ms/93.7%, 100ms/96.8%
Mark: 4 stripe(s), 2 proactive flush(es), 1 terminate flush(es), 0 completion(s), 0 continuation(s)
Relocation: Successful, 867M relocated
NMethods: 3209 registered, 1559 unregistered
Soft: 466881 encountered, 0 discovered, 0 enqueued
Weak: 5421 encountered, 3526 discovered, 1742 enqueued
Final: 55 encountered, 6 discovered, 0 enqueued
Phantom: 74 encountered, 59 discovered, 0 enqueued

<table>
<thead>
<tr>
<th></th>
<th>Mark Start</th>
<th>Mark End</th>
<th>Relocate Start</th>
<th>Relocate End</th>
<th>High</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity:</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
<td>131072M (100%)</td>
</tr>
<tr>
<td>Reserve:</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
<td>72M (0%)</td>
</tr>
<tr>
<td>Free:</td>
<td>7654M (6%)</td>
<td>5898M (4%)</td>
<td>24018M (18%)</td>
<td>126296M (96%)</td>
<td>126306M (96%)</td>
<td>5866M (4%)</td>
</tr>
<tr>
<td>Used:</td>
<td>123346M (94%)</td>
<td>125102M (95%)</td>
<td>106982M (82%)</td>
<td>4704M (4%)</td>
<td>125134M (95%)</td>
<td>4694M (4%)</td>
</tr>
<tr>
<td>Live:</td>
<td>-</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
<td>1722M (1%)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Allocated:</td>
<td>-</td>
<td>1796M (1%)</td>
<td>1962M (1%)</td>
<td>3766M (3%)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Garbage:</td>
<td>-</td>
<td>121623M (93%)</td>
<td>103353M (79%)</td>
<td>127M (0%)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reclaimed:</td>
<td>-</td>
<td>-</td>
<td>18270M (14%)</td>
<td>121496M (93%)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Garbage Collection (Allocation Rate) 123346M(94%)->4704M(4%)
## GC Statistics

--- Garbage Collection Statistics ---

<table>
<thead>
<tr>
<th>Collector: Garbage Collection Cycle</th>
<th>Last 10s Avg / Max</th>
<th>Last 10m Avg / Max</th>
<th>Last 10h Avg / Max</th>
<th>Total Avg / Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1906.804 / 1906.804</td>
<td>2184.368</td>
<td>2764.747</td>
<td>6555.377</td>
<td>6565.377</td>
</tr>
<tr>
<td>0.000 / 0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000 / 0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000 / 0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

### Contention: Mark Segment Reset Contention
- 27 / 275 9 / 1137 14 / 9460

### Critical: GC Locker Stall
- 1458 / 1672 1679 / 7914 1679 / 7914

### Memory: Page Cache Miss
- 8 / 81 771 / 3777

### Critical: Allocation Stall
- 0 / 0 0 / 0 0 / 0

### Memory: Page Cache Flush
- 0 / 0 0 / 0 0 / 0

### Critical: GC Locker Stall
- 0 / 0 0 / 0 0 / 0

### Memory: Out Of Memory
- 0 / 0 0 / 0 0 / 0

### Collector: Garbage Collection Cycle

### Memory: Heap Used After Mark
- 125102 / 125102 79656 / 125102 79566 / 125102

### Critical: GC Locker Stall
- 0 / 0 0 / 0 0 / 0

### Memory: Page Cache Hit L1
- 710 / 1037 771 / 3777

### Memory: Page Cache Hit L2
- 12 / 77 5 / 517

### Memory: Page Cache Miss
- 8 / 81 29 / 1932

### Memory: Page Cache Hit L1
- 710 / 1037 771 / 3777

### Memory: Page Cache Hit L2
- 12 / 77 5 / 517

### Memory: Page Cache Miss
- 8 / 81 29 / 1932

### Average / Maximum

---
### Garbage Collection Statistics

<table>
<thead>
<tr>
<th>Collector: Garbage Collection Cycle</th>
<th>Last 10s Avg / Max</th>
<th>Last 10m Avg / Max</th>
<th>Last 10h Avg / Max</th>
<th>Total Avg / Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector: Garbage Collection Cycle</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 2</td>
<td>0 / 2</td>
</tr>
<tr>
<td>Critical: Allocation Stall</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
</tr>
<tr>
<td>Critical: GC Locker Stall</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>Memory: Allocation Rate</td>
<td>1458 / 1672</td>
<td>1089 / 1904</td>
<td>1679 / 7914</td>
<td>1679 / 7914</td>
</tr>
<tr>
<td>Memory: Heap Used After Mark</td>
<td>125102 / 125102</td>
<td>97411 / 125102</td>
<td>70566 / 125102</td>
<td>70566 / 125102</td>
</tr>
<tr>
<td>Memory: Heap Used After Relocation</td>
<td>4704 / 4704</td>
<td>4019 / 4814</td>
<td>17042 / 34092</td>
<td>17042 / 34092</td>
</tr>
<tr>
<td>Memory: Heap Used Before Mark</td>
<td>123346 / 123346</td>
<td>95981 / 123346</td>
<td>61669 / 123346</td>
<td>61669 / 123346</td>
</tr>
<tr>
<td>Memory: Heap Used Before Relocation</td>
<td>106982 / 106982</td>
<td>85095 / 106982</td>
<td>60136 / 106982</td>
<td>60136 / 106982</td>
</tr>
<tr>
<td>Memory: Out Of Memory</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>Memory: Page Cache Flush</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>Memory: Page Cache Hit L1</td>
<td>710 / 1037</td>
<td>499 / 1037</td>
<td>771 / 3777</td>
<td>771 / 3777</td>
</tr>
<tr>
<td>Memory: Page Cache Hit L2</td>
<td>12 / 77</td>
<td>3 / 179</td>
<td>5 / 517</td>
<td>5 / 517</td>
</tr>
<tr>
<td>Memory: Page Cache Miss</td>
<td>8 / 81</td>
<td>19 / 651</td>
<td>29 / 1932</td>
<td>29 / 1932</td>
</tr>
<tr>
<td>Memory: Undo Object Allocation Failed</td>
<td>1 / 12</td>
<td>0 / 18</td>
<td>0 / 64</td>
<td>0 / 64</td>
</tr>
<tr>
<td>Memory: Undo Object Allocation Succeeded</td>
<td>26 / 263</td>
<td>9 / 1137</td>
<td>13 / 9460</td>
<td>13 / 9460</td>
</tr>
<tr>
<td>Phase: Concurrent Destroy Detached Pages</td>
<td>0.002 / 0.002</td>
<td>0.001 / 0.002</td>
<td>0.041 / 2.230</td>
<td>0.041 / 2.230</td>
</tr>
<tr>
<td>Phase: Concurrent Mark</td>
<td>1151.425 / 1151.425</td>
<td>1163.417 / 1452.750</td>
<td>1584.987 / 3028.289</td>
<td>1584.987 / 3028.289</td>
</tr>
<tr>
<td>Phase: Concurrent Mark Continue</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
</tr>
<tr>
<td>Phase: Concurrent Prepare Relocation Set</td>
<td>70.922 / 70.922</td>
<td>67.627 / 74.532</td>
<td>81.298 / 347.926</td>
<td>81.298 / 347.926</td>
</tr>
<tr>
<td>Phase: Concurrent Process Non-Strong References</td>
<td>0.367 / 0.367</td>
<td>0.391 / 0.463</td>
<td>0.394 / 0.774</td>
<td>0.394 / 0.774</td>
</tr>
<tr>
<td>Phase: Concurrent Relocate</td>
<td>645.941 / 645.941</td>
<td>682.349 / 645.941</td>
<td>1051.985 / 2198.100</td>
<td>1051.985 / 2198.100</td>
</tr>
<tr>
<td>Phase: Pause Mark End</td>
<td>0.882 / 0.882</td>
<td>0.997 / 1.095</td>
<td>0.941 / 1.199</td>
<td>0.941 / 1.199</td>
</tr>
<tr>
<td>Phase: Pause Mark Start</td>
<td>0.949 / 0.949</td>
<td>0.875 / 0.983</td>
<td>0.832 / 1.013</td>
<td>0.832 / 1.013</td>
</tr>
<tr>
<td>Phase: Pause Relocate Start</td>
<td>1.419 / 1.419</td>
<td>1.532 / 2.091</td>
<td>1.474 / 2.127</td>
<td>1.474 / 2.127</td>
</tr>
</tbody>
</table>

...
### Garbage Collection Statistics

<table>
<thead>
<tr>
<th></th>
<th>Last 10s</th>
<th>Last 10m</th>
<th>Last 10h</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg / Max</td>
<td>Avg / Max</td>
<td>Avg / Max</td>
<td>Avg / Max</td>
</tr>
<tr>
<td>Contention: Mark Segment Reset Contention</td>
<td>3 / 24</td>
<td>0 / 34</td>
<td>0 / 73</td>
<td>0 / 73 ops/s</td>
</tr>
<tr>
<td>Contention: Mark SeqNum Reset Contention</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0 ops/s</td>
</tr>
<tr>
<td>Contention: Relocation Contention</td>
<td>27 / 275</td>
<td>9 / 1137</td>
<td>14 / 9460</td>
<td>14 / 9460 ops/s</td>
</tr>
<tr>
<td>Critical: Allocation Stall</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000 ms</td>
</tr>
<tr>
<td>Critical: GC Locker Stall</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000 ops/s</td>
</tr>
<tr>
<td>Memory: Allocation Rate</td>
<td>1458 / 1672</td>
<td>1089 / 1904</td>
<td>1679 / 7914</td>
<td>1679 / 7914 MB/s</td>
</tr>
<tr>
<td>Memory: Heap Used After Mark</td>
<td>125102 / 125102</td>
<td>97411 / 125102</td>
<td>70566 / 125102</td>
<td>70566 / 125102 MB</td>
</tr>
<tr>
<td>Memory: Heap Used After Relocation</td>
<td>4704 / 4704</td>
<td>4019 / 4814</td>
<td>17042 / 34092</td>
<td>17042 / 34092 MB</td>
</tr>
<tr>
<td>Memory: Heap Used Before Mark</td>
<td>123346 / 123346</td>
<td>95981 / 123346</td>
<td>61669 / 106982</td>
<td>61669 / 106982 MB</td>
</tr>
<tr>
<td>Memory: Heap Used Before Relocation</td>
<td>106982 / 106982</td>
<td>85095 / 106982</td>
<td>60136 / 106982</td>
<td>60136 / 106982 MB</td>
</tr>
<tr>
<td>Memory: Out Of Memory</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0 ops/s</td>
</tr>
<tr>
<td>Memory: Page Cache Flush</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 0 MB/s</td>
</tr>
<tr>
<td>Memory: Page Cache Hit L1</td>
<td>710 / 1037</td>
<td>499 / 1037</td>
<td>771 / 3777</td>
<td>771 / 3777 ops/s</td>
</tr>
</tbody>
</table>

### Phase: Concurrent Destroy Detached Pages

<table>
<thead>
<tr>
<th>Phase: Concurrent Destroy Detached Pages</th>
<th>Last 10s</th>
<th>Last 10m</th>
<th>Last 10h</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase: Concurrent Mark Continue</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Prepare Relocation Set</td>
<td>70.922 / 70.922</td>
<td>67.627 / 74.532</td>
<td>81.298 / 347.926</td>
<td>81.298 / 347.926 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Process Non-Strong References</td>
<td>0.367 / 0.367</td>
<td>0.391 / 0.463</td>
<td>0.394 / 0.774</td>
<td>0.394 / 0.774 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Relocate</td>
<td>645.941 / 645.941</td>
<td>682.340 / 645.941</td>
<td>1051.985 / 2198.100</td>
<td>1051.985 / 2198.100 ms</td>
</tr>
<tr>
<td>Phase: Pause Mark End</td>
<td>0.882 / 0.882</td>
<td>0.997 / 1.095</td>
<td>0.941 / 1.199</td>
<td>0.941 / 1.199 ms</td>
</tr>
<tr>
<td>Phase: Pause Mark Start</td>
<td>0.949 / 0.949</td>
<td>0.875 / 0.983</td>
<td>0.832 / 1.013</td>
<td>0.832 / 1.013 ms</td>
</tr>
<tr>
<td>Phase: Pause Relocate Start</td>
<td>1.419 / 1.419</td>
<td>1.532 / 2.091</td>
<td>1.474 / 2.127</td>
<td>1.474 / 2.127 ms</td>
</tr>
</tbody>
</table>
### Garbage Collection Statistics

<table>
<thead>
<tr>
<th>Collector: Garbage Collection Cycle</th>
<th>Last 10s Avg / Max</th>
<th>Last 10m Avg / Max</th>
<th>Last 10h Avg / Max</th>
<th>Total Avg / Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contention: Mark Segment Reset Contention</td>
<td>3 / 24</td>
<td>0 / 34</td>
<td>0 / 73</td>
<td>0 / 73 ops/s</td>
</tr>
<tr>
<td>Contention: Mark SeqNum Reset Contention</td>
<td>0 / 0</td>
<td>0 / 0</td>
<td>0 / 2</td>
<td>0 / 2 ops/s</td>
</tr>
<tr>
<td>Contention: Relocation Contention</td>
<td>27 / 275</td>
<td>9 / 1137</td>
<td>14 / 9460</td>
<td>14 / 9460 ops/s</td>
</tr>
<tr>
<td>Critical: Allocation Stall</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000 ms</td>
</tr>
<tr>
<td>Memory: Page Cache Miss</td>
<td>8 / 81</td>
<td>19 / 651</td>
<td>29 / 1932</td>
<td>29 / 1932 ops/s</td>
</tr>
<tr>
<td>Memory: Undo Object Allocation Failed</td>
<td>1 / 12</td>
<td>0 / 18</td>
<td>0 / 64</td>
<td>0 / 64 ops/s</td>
</tr>
<tr>
<td>Memory: Undo Object Allocation Succeeded</td>
<td>26 / 263</td>
<td>9 / 1137</td>
<td>13 / 9460</td>
<td>13 / 9460 ops/s</td>
</tr>
<tr>
<td>Phase: Concurrent Destroy Detached Pages</td>
<td>0.002 / 0.002</td>
<td>0.001 / 0.002</td>
<td>0.041 / 2.230</td>
<td>0.041 / 2.230 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Mark</td>
<td>1151.425 / 1151.425</td>
<td>1163.417 / 1452.750</td>
<td>1584.987 / 3028.289</td>
<td>1584.987 / 3028.289 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Mark Continue</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000</td>
<td>0.000 / 0.000 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Prepare Relocation Set</td>
<td>70.922 / 70.922</td>
<td>67.627 / 74.532</td>
<td>81.298 / 374.926</td>
<td>81.298 / 374.926 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Process Non-Strong References</td>
<td>0.367 / 0.367</td>
<td>0.391 / 0.463</td>
<td>0.394 / 0.774</td>
<td>0.394 / 0.774 ms</td>
</tr>
<tr>
<td>Phase: Concurrent Relocate</td>
<td>645.941 / 645.941</td>
<td>682.340 / 645.941</td>
<td>1051.985 / 2198.100</td>
<td>1051.985 / 2198.100 ms</td>
</tr>
<tr>
<td>Phase: Pause Mark End</td>
<td>0.882 / 0.882</td>
<td>0.997 / 1.095</td>
<td>0.941 / 1.199</td>
<td>0.941 / 1.199 ms</td>
</tr>
<tr>
<td>Phase: Pause Mark Start</td>
<td>0.949 / 0.949</td>
<td>0.875 / 0.983</td>
<td>0.832 / 1.013</td>
<td>0.832 / 1.013 ms</td>
</tr>
<tr>
<td>Phase: Pause Relocate Start</td>
<td>1.419 / 1.419</td>
<td>1.532 / 2.091</td>
<td>1.474 / 2.127</td>
<td>1.474 / 2.127 ms</td>
</tr>
</tbody>
</table>

Average / Maximum
Future Plans
Future Plans

• Short-term
  – Concurrent class unloading
  – Turn ZGC into a product feature
Future Plans

• Short-term
  – Concurrent class unloading
  – Turn ZGC into a product feature

Available today in the ZGC development repository!
Future Plans

• Short-term
  – Concurrent class unloading
  – Turn ZGC into a product feature
Future Plans

• Short-term
  – Concurrent class unloading
  – Turn ZGC into a product feature
Future Plans

• **Short-term**
  – Concurrent class unloading
  – Turn ZGC into a product feature

• **Long-term**
  – Generational
  – Sub-millisecond max pause times
  – Additional platform support
  – Graal JIT support
Future Plans

• **Short-term**
  – Concurrent class unloading
  – Turn ZGC into a product feature

• **Long-term**
  – Generational
  – Sub-millisecond max pause times
  – Additional platform support
  – Graal JIT support

*Generational*
- Withstand higher allocation rates
- Lower heap overhead
- Lower CPU usage
Future Plans

• **Short-term**
  – Concurrent class unloading
  – Turn ZGC into a product feature

• **Long-term**
  – Generational
  – Sub-millisecond max pause times
  – Additional platform support
  – Graal JIT support

**Sub-millisecond max pause times**
• Within reach
• Reduce root set size
• Time-to-Safepoint, etc
Future Plans

• Short-term
  – Concurrent class unloading
  – Turn ZGC into a product feature

• Long-term
  – Generational
  – Sub-millisecond max pause times
  – Additional platform support
  – Graal JIT support

Additional platform support
• macOS?
• Windows?
• Sparc?
• Aarch64?
Future Plans

• **Short-term**
  – Concurrent class unloading
  – Turn ZGC into a product feature

• **Long-term**
  – Generational
  – Sub-millisecond max pause times
  – Additional platform support
  – Graal JIT support
Get Involved!
ZGC Project
Follow, Participate, Give Feedback

zgc-dev@openjdk.java.net

http://wiki.openjdk.java.net/display/zgc/Main
ZGC Project
Source Code

OpenJDK

http://hg.openjdk.java.net/jdk/jdk

http://hg.openjdk.java.net/zgc/zgc
Thanks!
Questions?