Boosting the Priority of Garbage: Scheduling Collection on Heterogeneous Multicore Processors

Shoaib Akram, Jennifer B. Sartor, Kenzo Van Craeynest, Wim Heirman, Lieven Eeckhout
Ghent University, Belgium
Shoaib.Akram@UGent.be
Popularity of Managed Languages

The 2015 Top Ten Programming Languages, spectrum.ieee.org.
The Garbage Collection Advantage

Memory automatically reclaimed for reuse
Takes extra CPU cycles to provide the service
*Concurrent* collectors suited to multicores
Heterogeneous Multicores

- 600 Series
  - 4x ARM Cortex A72
  - 4x ARM Cortex A53

- Exynos 8890
  - 4x ARM Cortex A53
  - 4x Exynos M1

Performance vs. Power

In-Order

Out-of-Order
Managed Language Applications on Heterogeneous Multicores

Performance

Power

In-Order

Out-of-Order

Application →

big

big or LITTLE?

Garbage Collector →
GC on big versus LITTLE

Application and collector running concurrently

**Application**
- Allocates objects on heap

**Collector**
- Identifies live objects on heap and then reclaims memory taken up by remaining objects

Run Collector on big versus LITTLE and measure the difference in execution time
GC on big versus LITTLE

% increase in execution time

- fop
- antlr
- luindex
- bloat
- avrora
- lusearch.fix2
- sunflow2
- sunflow4
- xalan2
- pmd2
- lusearch2
- lusearch.fix4
- xalan4
- pmd4
- lusearch4
GC on big versus LITTLE

% increase in execution time

fop  antlr  luindex  bloat  avrora  lusearch.fix2  sunflow2  sunflow4  xalan2  pmd2  lusearch2  lusearch.fix4  xalan4  pmd4  lusearch4
Some applications exhibit GC-Criticality

GC on LITTLE detrimental for GC-Critical
GC on big versus LITTLE

What happens if GC runs on LITTLE for GC-Critical apps?

**Application**
- Allocates objects on heap
- Paused !!!

**Collector**
- Identifies live objects on heap and then reclaim memory taken up by remaining objects
- Serial collection

Application is paused if no free memory on heap because collector still running
Giving GC Fair Share of Big Core

- gc-fair
  - Equally share the big core among all threads
  - Based on Van Craeynest et al [PACT 2013]
- Baseline is gc-on-LITTLE
  - Pin the GC threads on LITTLE cores
- Observe the % reduction in execution time
Giving GC Fair Share of Big Core

- 3 LITTLE
- 2 LITTLE
- 1 LITTLE

% execution time reduction

GC-Uncritical

fop, antlr, luindex, bloat, avroria6, sunflow2, sunflow4, xalan2, avg-uncritical, pmd2, lusearch2, lusearch4, xalan4, pmd4, lusearch4, avg-critical, total average
Giving GC Fair Share of Big Core

% execution time reduction

-15 -10 -5 0 5 10 15 20 25

3 LITTLE
2 LITTLE
1 LITTLE

GC-Uncritical

fop antlr luindex bloat avrora6 lusearch.fix2 sunflow2 sunflow4 xalan2 avg-uncritical pmd2 lusearch2 lusearch.fix4 xalan4 pmd4 lusearch4 avg-critical total average
Giving GC Fair Share of Big Core

% execution time reduction

3 LITTLE
2 LITTLE
1 LITTLE

GC-Uncritical

gc-on-LITTLE for GC-Uncritical
Giving GC Fair Share of Big Core

% execution time reduction

-15 -10 -5 0 5 10 15 20 25

GC-Uncritical

GC-Critical

gc-on-LITTLE for GC-Uncritical

fop antlr luindex bloat avrora6 lusearch.fix2 sunflow2 sunflow4 xalan2 avg-uncritical pmd2 lusearch2 lusearch.fix4 xalan4 pmd4 lusearch4 avg-critical total average
Giving GC Fair Share of Big Core

% execution time reduction

-15 -10 -5 0 5 10 15 20 25

GC-Uncritical

GC-Critical

fop antlr luindex blot avrorafix2 sunflow2 sunflow4 xalan2 avg-uncritical pmd2 lusearch2 lusearchfix4 xalan4 pmd4 lusearch4 avg-critical total average

gc-on-LITTLE for GC-Uncritical
Giving GC Fair Share of Big Core

% execution time reduction

-15 -10 -5 0 5 10 15 20 25

GC-Uncritical

GC-Critical

3 LITTLE
2 LITTLE
1 LITTLE

gc-on-LITTLE for GC-Uncritical
gc-fair for GC-Critical

fop antlr luindex blobt avrora6 lusearch.fix2 sunflow2 sunflow4 xalan2 avg-uncritical pmd2 lusearch fix4 xalan4 pmd4 lusearch4 avg-critical total average
Giving GC Fair Share of Big Core

GC-Criticality depends on architecture, application, and runtime environment
Our Contribution

GC-Criticality-Aware Scheduler
Dynamically adjusts # big core cycles given to the concurrent collector

GC-Criticality depends on architecture, application, and runtime environment

GC-Criticality
GC-Uncritical

% execution time reduction

3 LITTLE
2 LITTLE
1 LITTLE

.top, antlr, itunes, xalan, avg-uncritical, lusearch, md, avg, total average
GC-Criticality-Aware Scheduler
Runtime Activity → How Scheduler Reacts?

App alone

app

gc

Schd.
gc-on-LITTLE
GC-Criticality-Aware Scheduler

gc-on-LITTLE to gc-fair

App alone

app

gc

Schd. gc-on-LITTLE
Stop pause to do book-keeping ignored
Scan stop pause: JVM signals scheduler
gc-fair gives equal priority to GC and app
GC-Criticality-Aware Scheduler
Boost States

Stop scan pauses observed even with gc-fair

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>How many quanta scheduled on the BIG core?</th>
</tr>
</thead>
<tbody>
<tr>
<td>gc-on-LITTLE</td>
<td>First GC thread = 0, Second GC thread = 0</td>
</tr>
<tr>
<td>gc-fair</td>
<td>First GC thread = 1, Second GC thread = 1</td>
</tr>
</tbody>
</table>

Boost the priority of garbage
Give GC more consecutive quanta on big

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>State</th>
<th>How many quanta scheduled on the BIG core?</th>
</tr>
</thead>
<tbody>
<tr>
<td>gc-boost</td>
<td>P0</td>
<td>First GC thread = 1, Second GC thread = 1</td>
</tr>
<tr>
<td>gc-boost</td>
<td>P1</td>
<td>First GC thread = 1, Second GC thread = 2</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Degradate boost state when no longer critical
GC-Criticality-Aware Scheduler

gc-boost:P0 to gc-on-LITTLE

JVM signals the scheduler

If no scan pause in state P0, go to gc-on-LITTLE
Can configure # zero stop scan intervals before returning to gc-on-LITTLE
GC-Criticality-Aware Scheduler

Summary

• JVM detects GC-Criticality during runtime
• Communicates criticality information down to the scheduler
• Scheduler dynamically adapts big core cycles given to GC
Experimental Setup

- **Java Virtual Machine**
  - Jikes Research Virtual Machine (Version 3.1.2)
  - Full-heap concurrent collector with two threads
  - Tackle non-determinism by warming up the JVM
  - Heap size 2x of minimum

- **Benchmarks**
  - Ten benchmarks from DaCapo
  - Vary the # threads – 1 to 4

- **Heterogeneous Multicore Setup**
  - Sniper multicore simulator (Version 4.0)
  - Different four core heterogeneous architectures
  - Varying # of big and LITTLE cores
Performance of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core

% execution time reduction

gc-fair

GC-Uncritical

GC-Critical

fop, antlr, luindex, bloat, avror6, sunflow2, sunflow4, xalan2, avg-uncritical, pmd2, lusearch2, xalan4, pmd4, lusearch4, avg-critical, total average
Performance of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core

gc-fair

gc-boost

GC-Uncritical

GC-Critical

gc-boost performance neutral for GC-Uncritical
Performance of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core

% execution time reduction

-20 -15 -10 -5 0 5 10 15 20 25

gc-fair
gc-boost

GC-Uncritical

GC-Critical

fop antlr luindex bloat avrora6 lusearch.fix2 sunflow2 sunflow4 xalan2 avg-uncritical pmd2 lusearch4.x4 xalan4 pmd4 lusearch4 avg-critical total average

gc-boost performance neutral for GC-Uncritical
Improves perf. of GC-Critical by 14% on avg.
Understanding the Performance Advantage of Big Core

![Graph showing cycles per instruction for different cache misses and instruction types.]

- L3 Miss
- L2 Miss
- L1-D Miss
- L1-I
- Base
Understanding the Performance Advantage of Big Core

Collector performs a heap traversal chasing pointers
Understanding the Performance Advantage of Big Core

Collector performs a heap traversal chasing pointers

Instruction-level parallelism 😊
Memory-level parallelism 😞
Performance of GC-Criticality-Aware Scheduler

Lowering frequency of LITTLE core

% execution time reduction

Similar freq.

GC-Uncritical  GC-Critical

fop antlr lusearch luindex bloat avroa6 sunflow2 sunflow4 xalan2 avg-uncritical pmd2 lusearch2 xalan4 pmd4 lusearch4 avg-critical total average
Performance of GC-Criticality-Aware Scheduler

Lowering frequency of LITTLE core

- Similar freq.
- 1 GHz slower

GC-Uncritical  GC-Critical

Lowering frequency increases GC-Criticality
Performance of GC-Criticality-Aware Scheduler

Lowering frequency of LITTLE core

-20 -15 -10 -5 0 5 10 15 20 25
% execution time reduction

Similar freq. 1 GHz slower

GC-Uncritical GC-Critical

Lowering frequency increases GC-Criticality
Improves perf. of GC-Critical by 20% on avg.
Performance of GC-Criticality-Aware Scheduler

Different # LITTLE cores

Allocation rate lowers with more LITTLE cores
gc-boost is beneficial for different # LITTLE
Energy Efficiency of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core

Negligible change in EDP for GC-Uncritical
20% avg. reduction in EDP for GC-Critical
More in the Paper

• Sensitivity studies
  – Varying number of total cores
  – Scheduling quantum and # zero scan intervals
  – Heap size

• GC-Criticality using OpenJDK’s collector
Conclusions

• Concurrent garbage collection benefits from out-of-order execution
• Java applications that allocate rapidly exhibit GC-Criticality
• GC-Criticality-Aware scheduler adjusts big core cycles given to GC on a heterogeneous multicore
  – Uses information provided by the JVM
  – Improves both performance and energy efficiency
Boosting the Priority of Garbage: Scheduling Collection on Heterogeneous Multicore Processors

Thank You!

Shoaib.Akram@UGent.be
http://users.elis.ugent.be/~sakram
GC Criticality with OpenJDK’s CMS

% increase in execution time

- fop
- antlr
- luindex
- bloat
- avrora
- sunflow2
- sunflow4
- xalan2
- pmd2
- lusearch.fix4
- xalan4
- pmd4
- lusearch4
Triggering Concurrent GC Every 32 MB of Allocation

% reduction in energy delay product

- pmd2
- lusearch2
- xalan4
- pmd4
- lusearch.fix4
- lusearch4
- avg-critical
GC-Criticality-Aware Scheduler

gc-boost:P0 to gc-boost:P1

JVM signals the scheduler

App alone  Stop  Concurrent  Scan  

app gc gc-boost:P0 gc-boost:P1

Schd. gc-boost:P1 gives GC two quanta on big
GC-Criticality-Aware Scheduler

gc-boost:P1 to gc-boost:P0

JVM signals the scheduler

Degrade boost state if no stop scan pause
Energy Efficiency of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core

% reduction in energy-delay product

25
20
15
10
5
0
-5
-10
Energy Efficiency of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core

Negligible change in EDP for GC-Uncritical