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

GHENT UNIVERSITY

VUB VRIJE UNIVERSITEIT BRUSSEL

intel

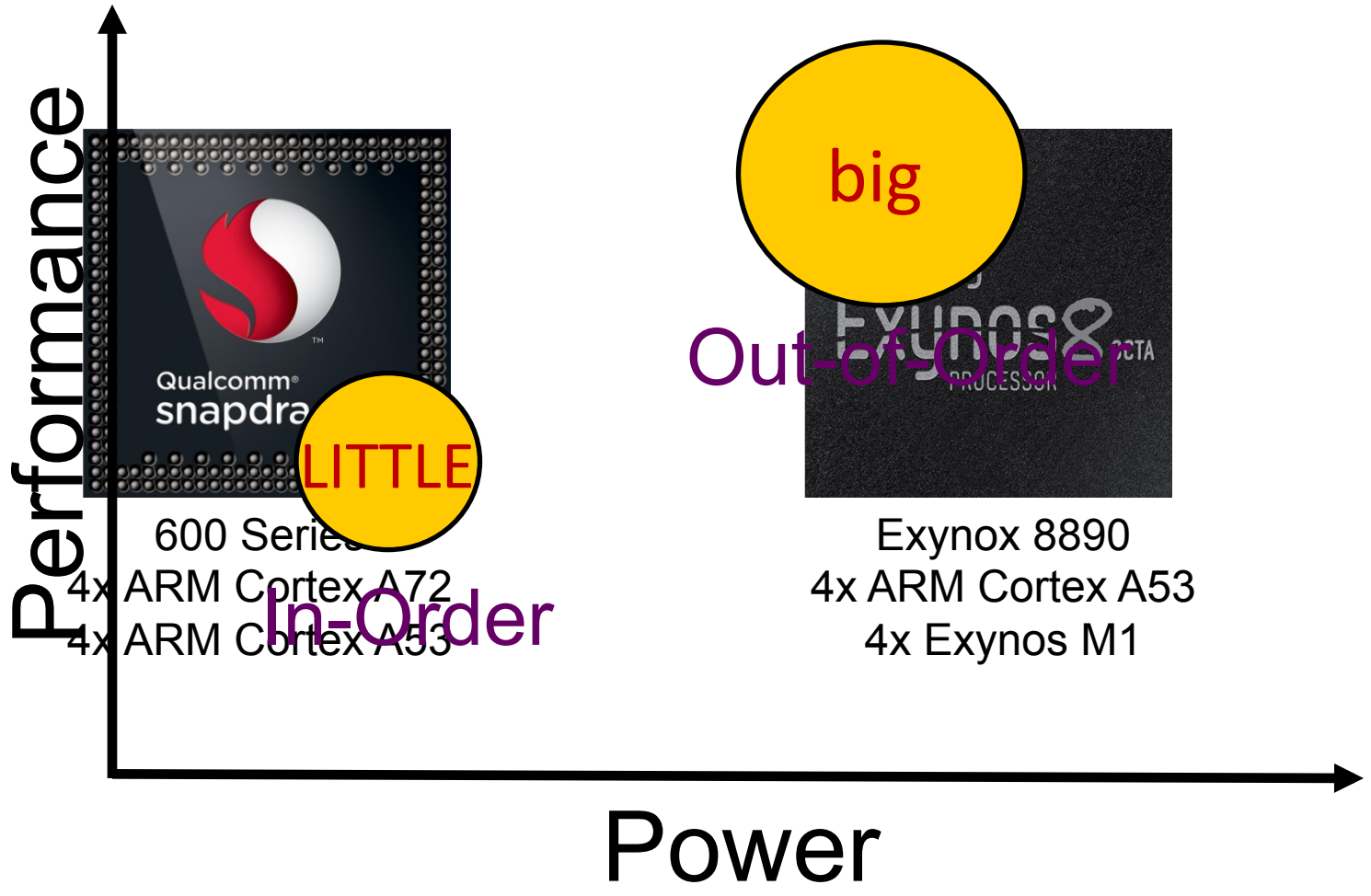
Popularity of Managed Languages

The 2015 Top Ten Programming Languages, spectrum.ieee.org.

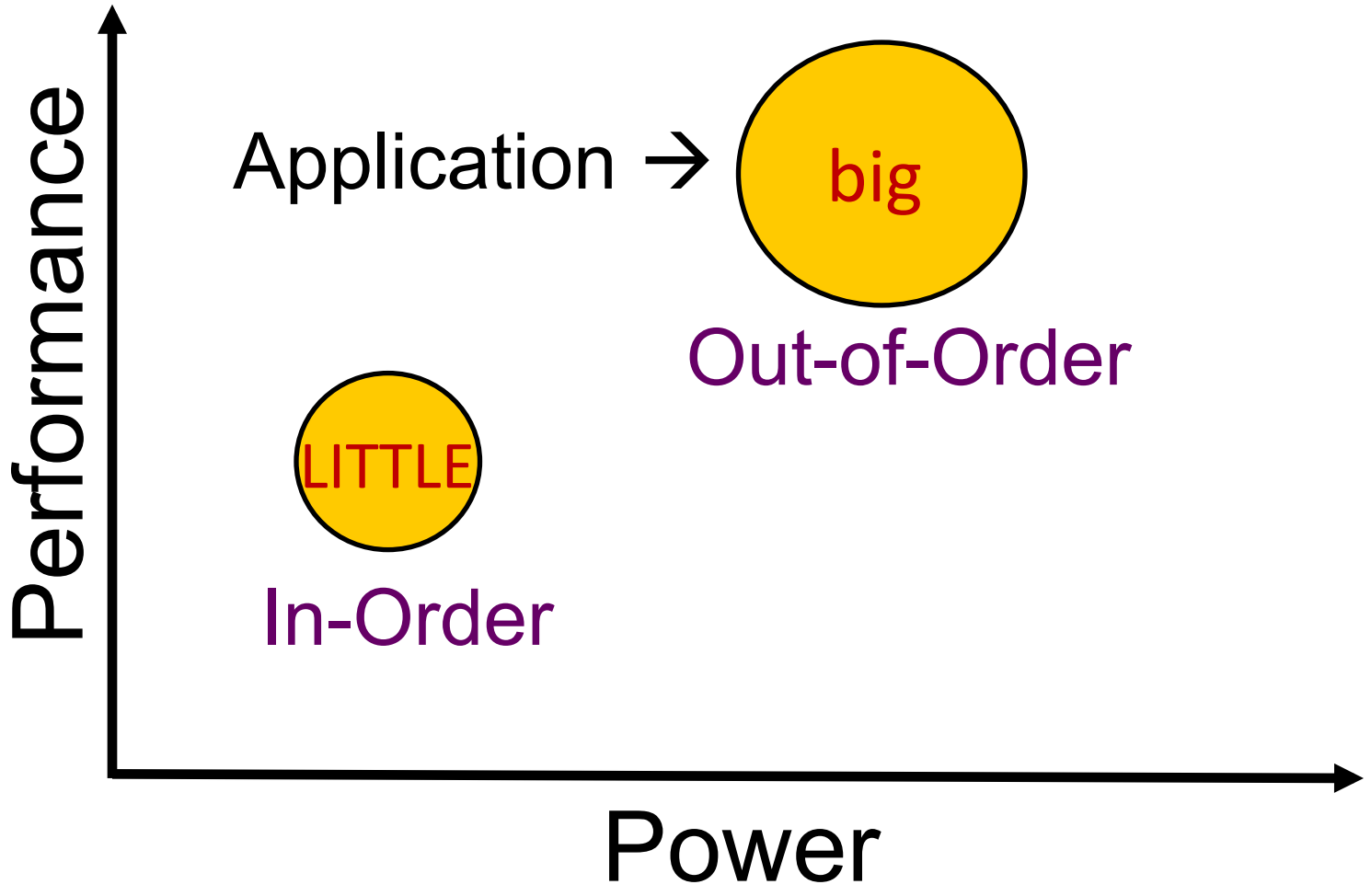
The Garbage Collection Advantage

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

Heterogeneous Multicores



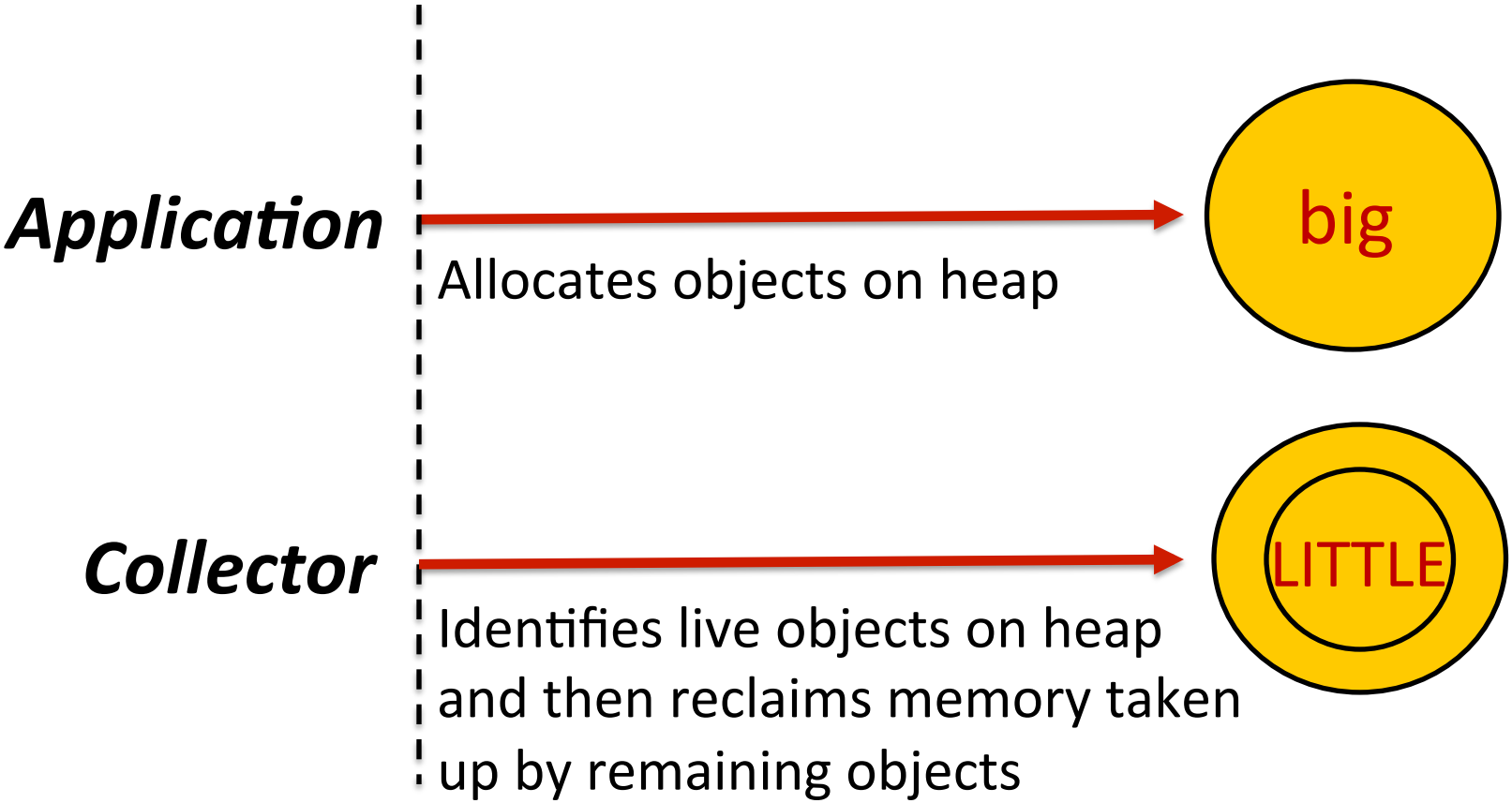
Managed Language Applications on Heterogeneous Multicores



Garbage Collector → big or LITTLE?

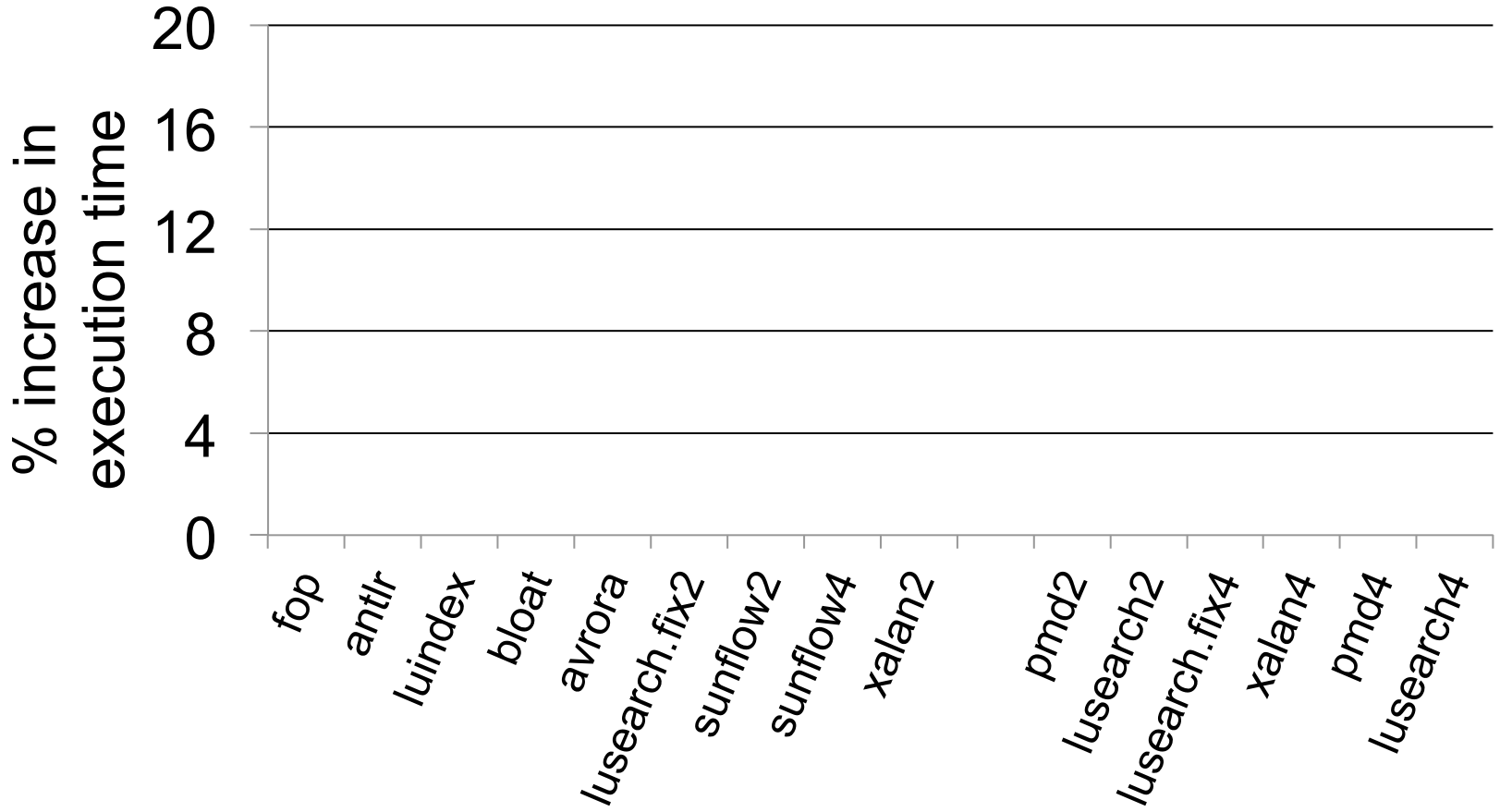
GC on big versus LITTLE

Application and collector running concurrently

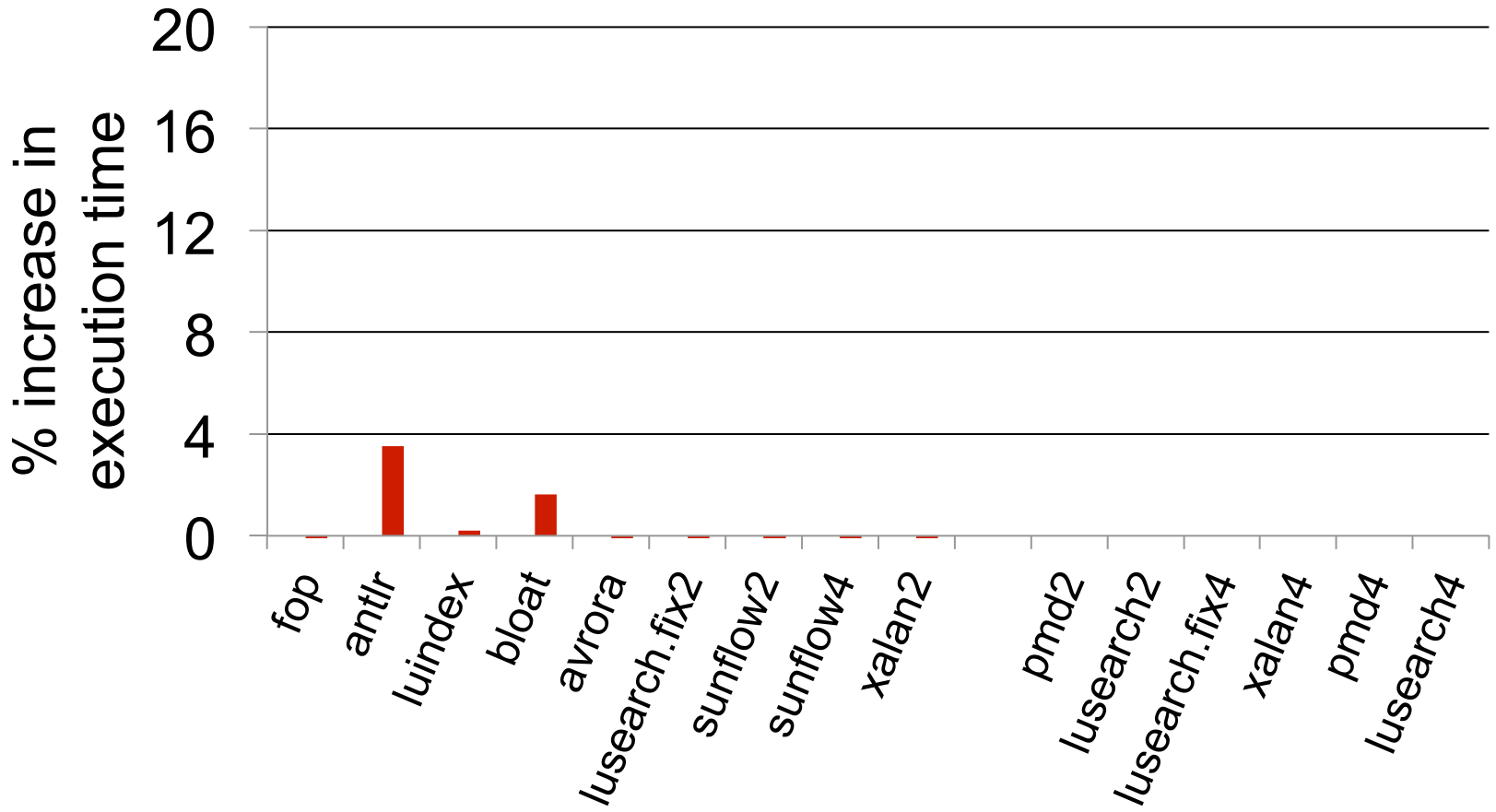


Run Collector on big versus LITTLE and measure the difference in execution time

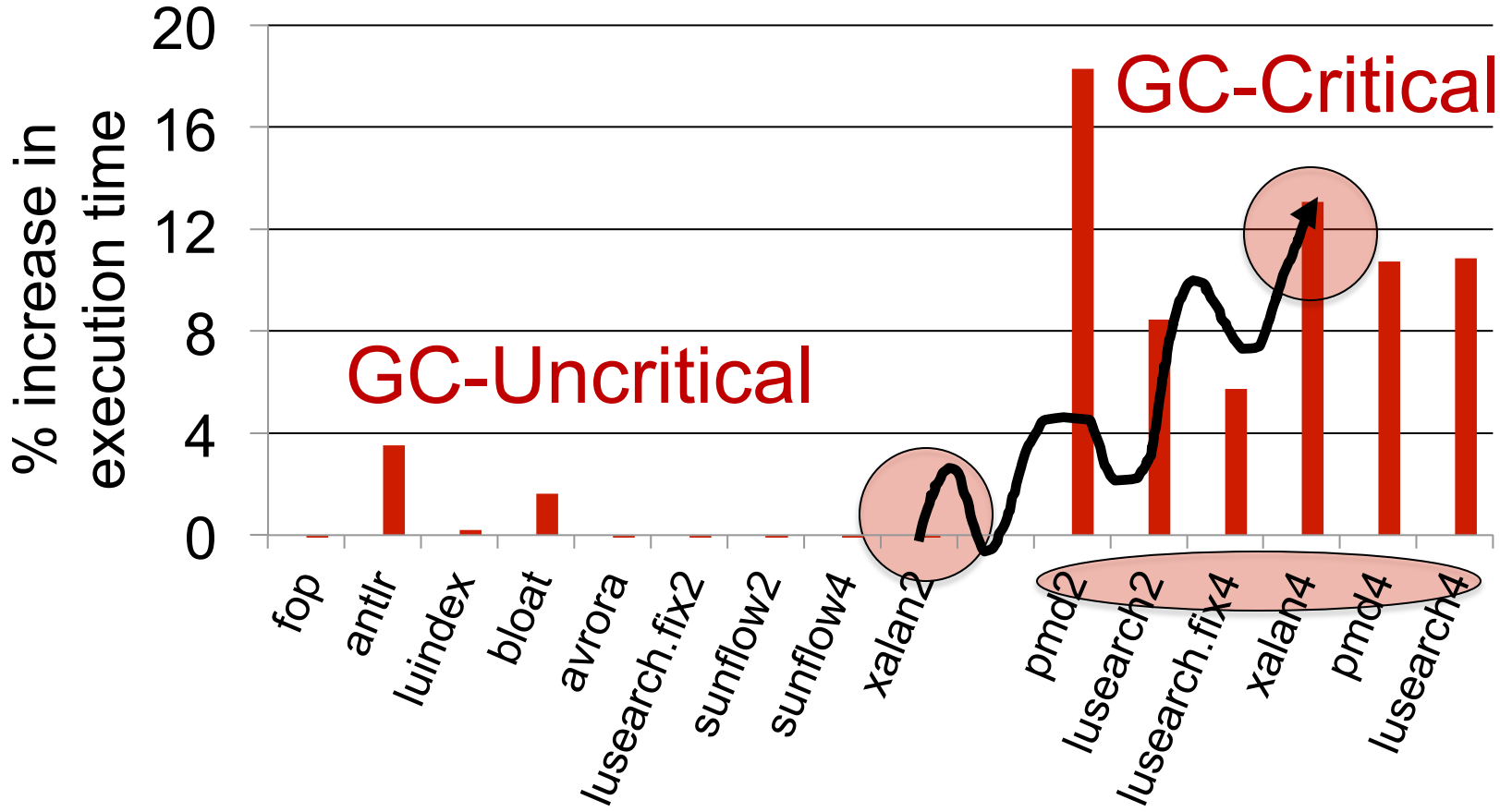
GC on big versus LITTLE



GC on big versus LITTLE



GC on big versus LITTLE

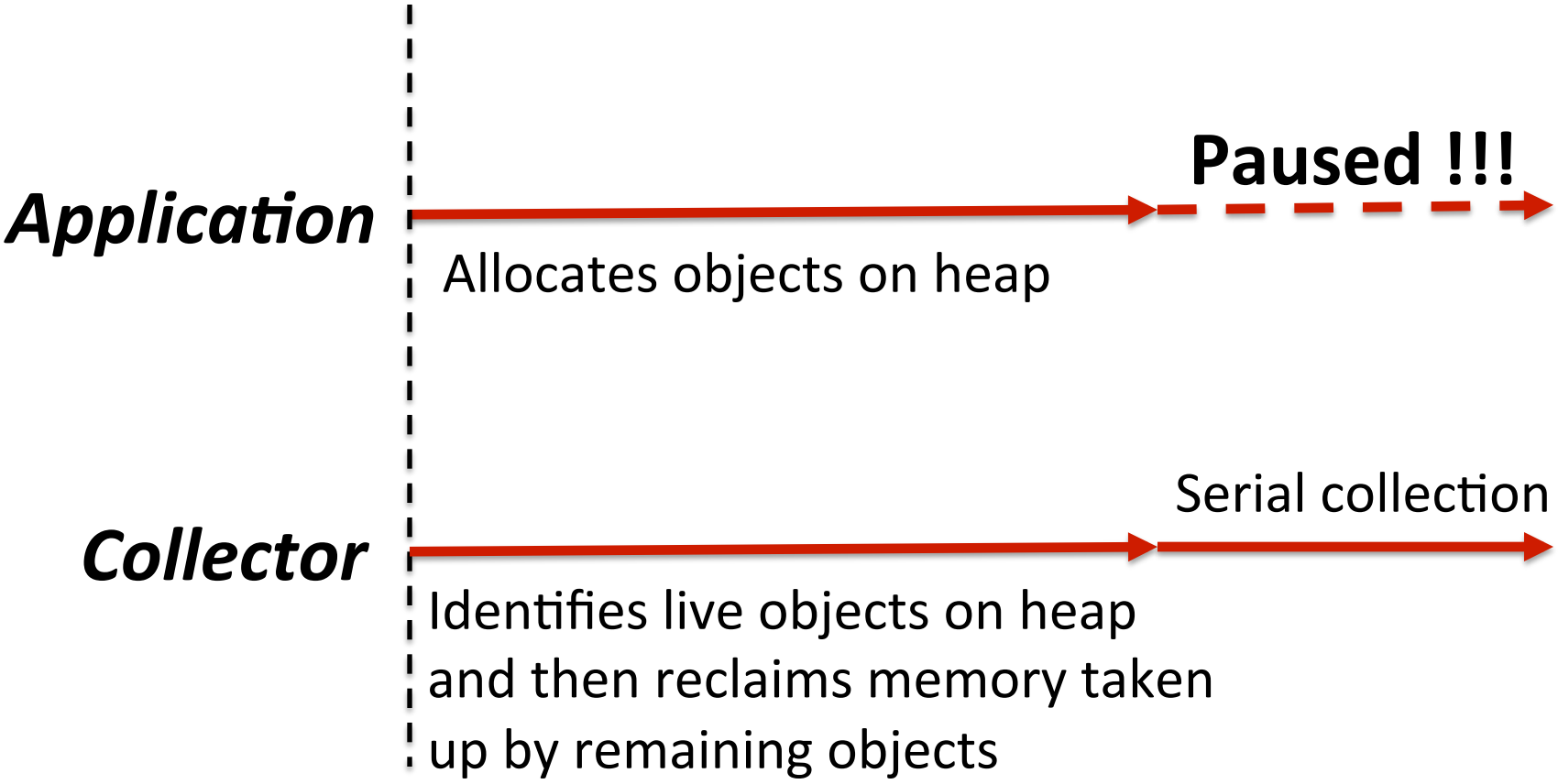


Some applications exhibit GC-Criticality

GC on LITTLE detrimental for GC-Critical

GC on big versus LITTLE

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

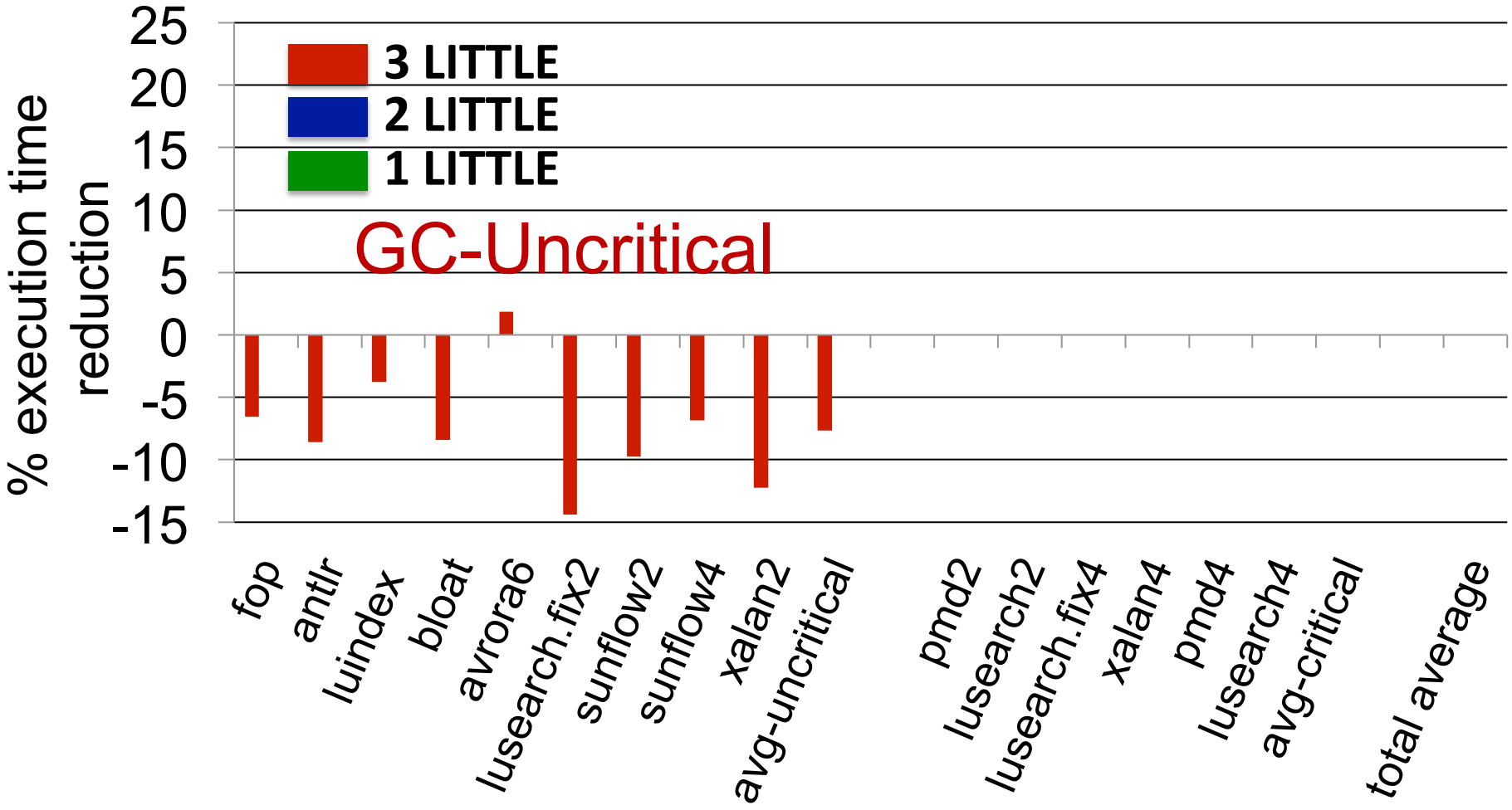


Application is paused if no free memory on heap because collector still running

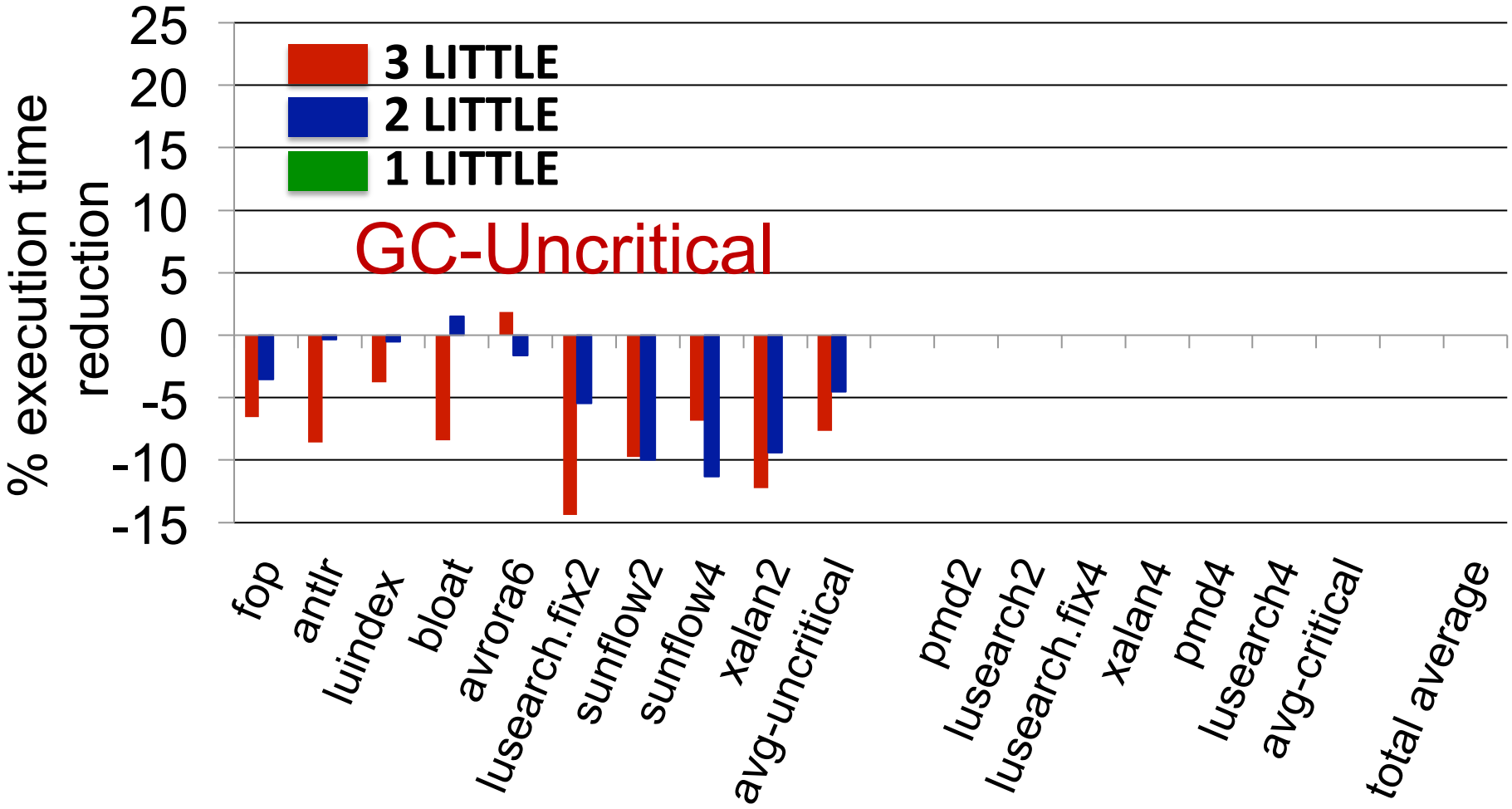
Giving GC Fair Share of Big Core

- 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

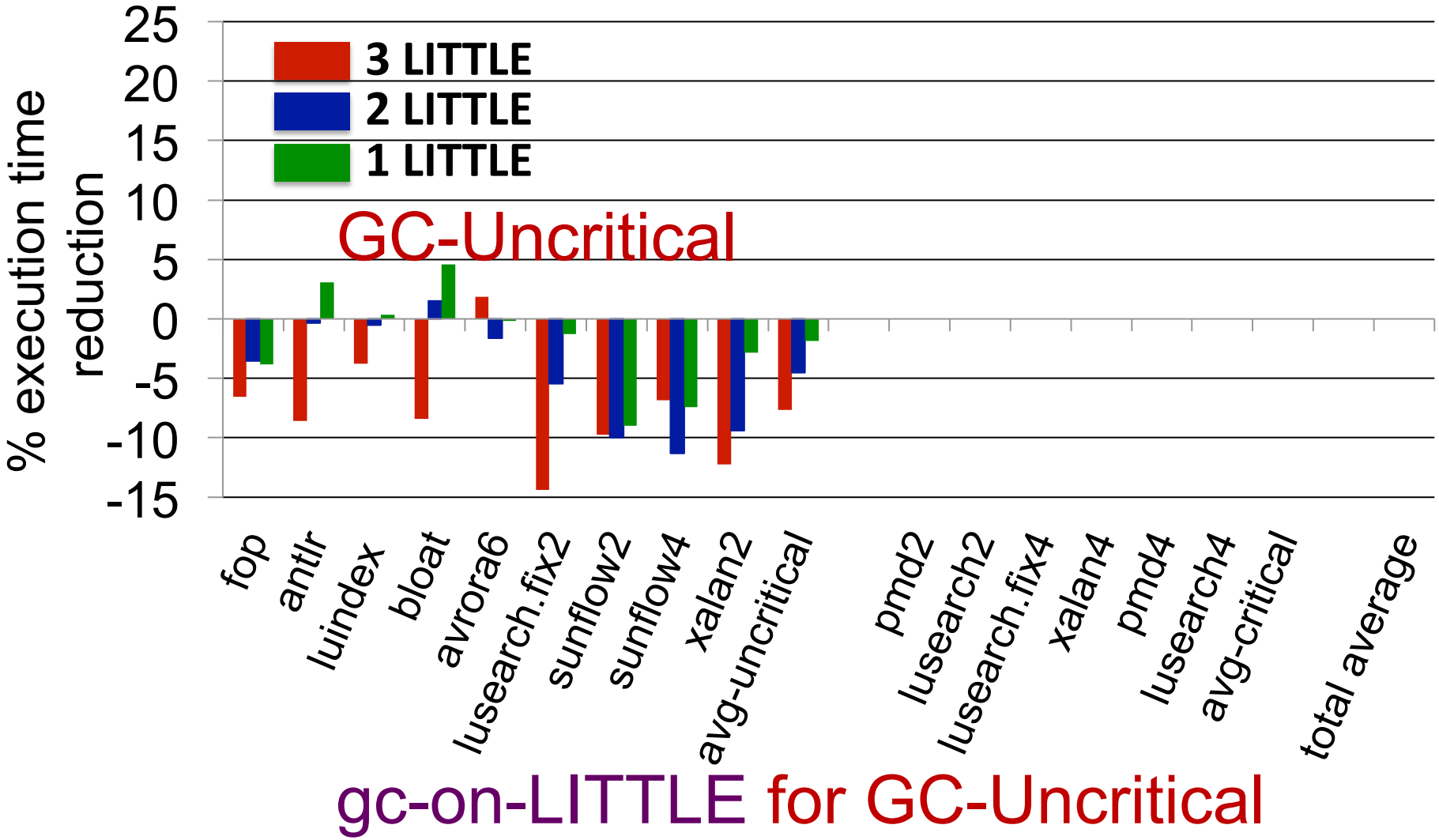
Giving GC Fair Share of Big Core



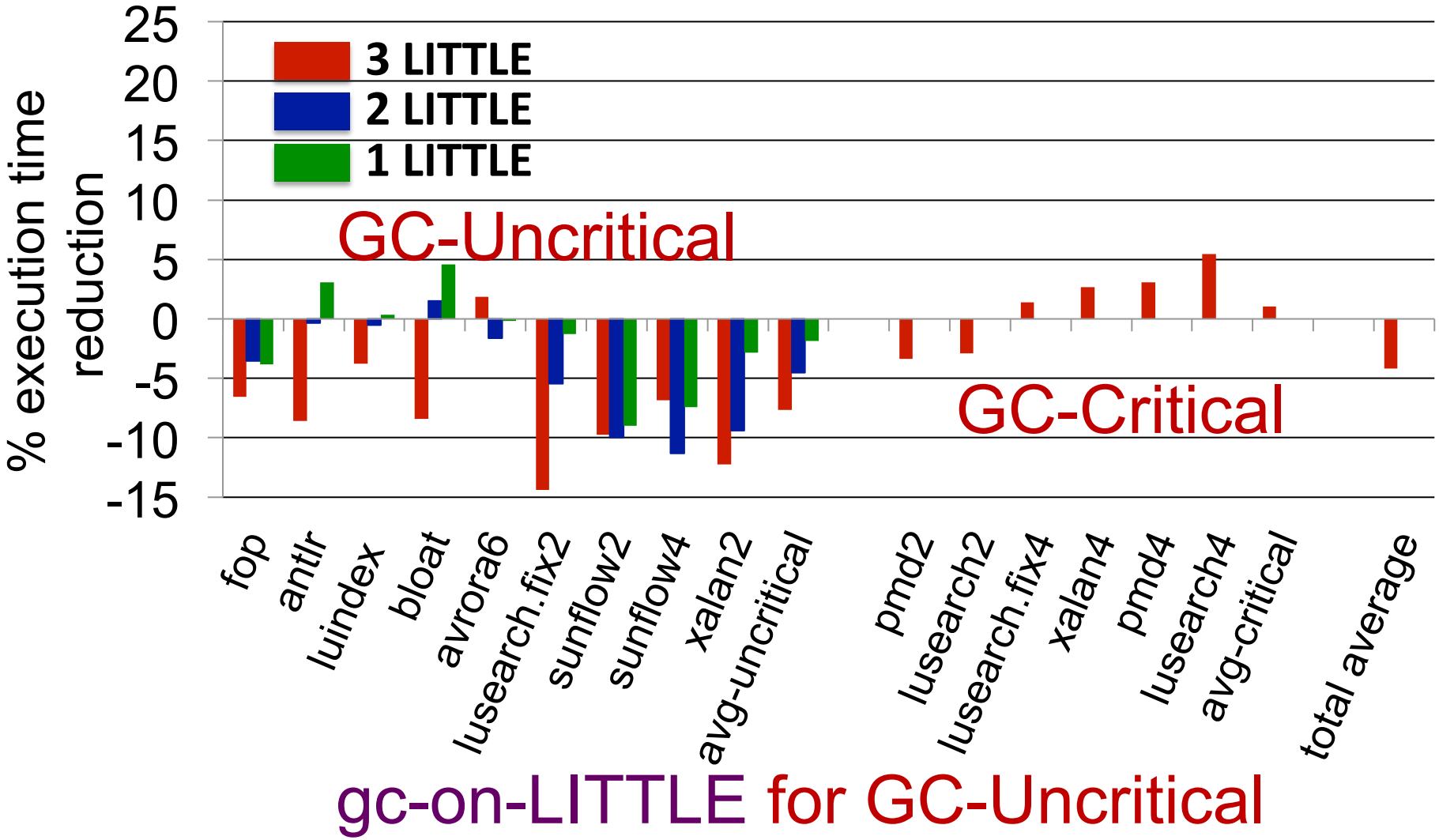
Giving GC Fair Share of Big Core



Giving GC Fair Share of Big Core

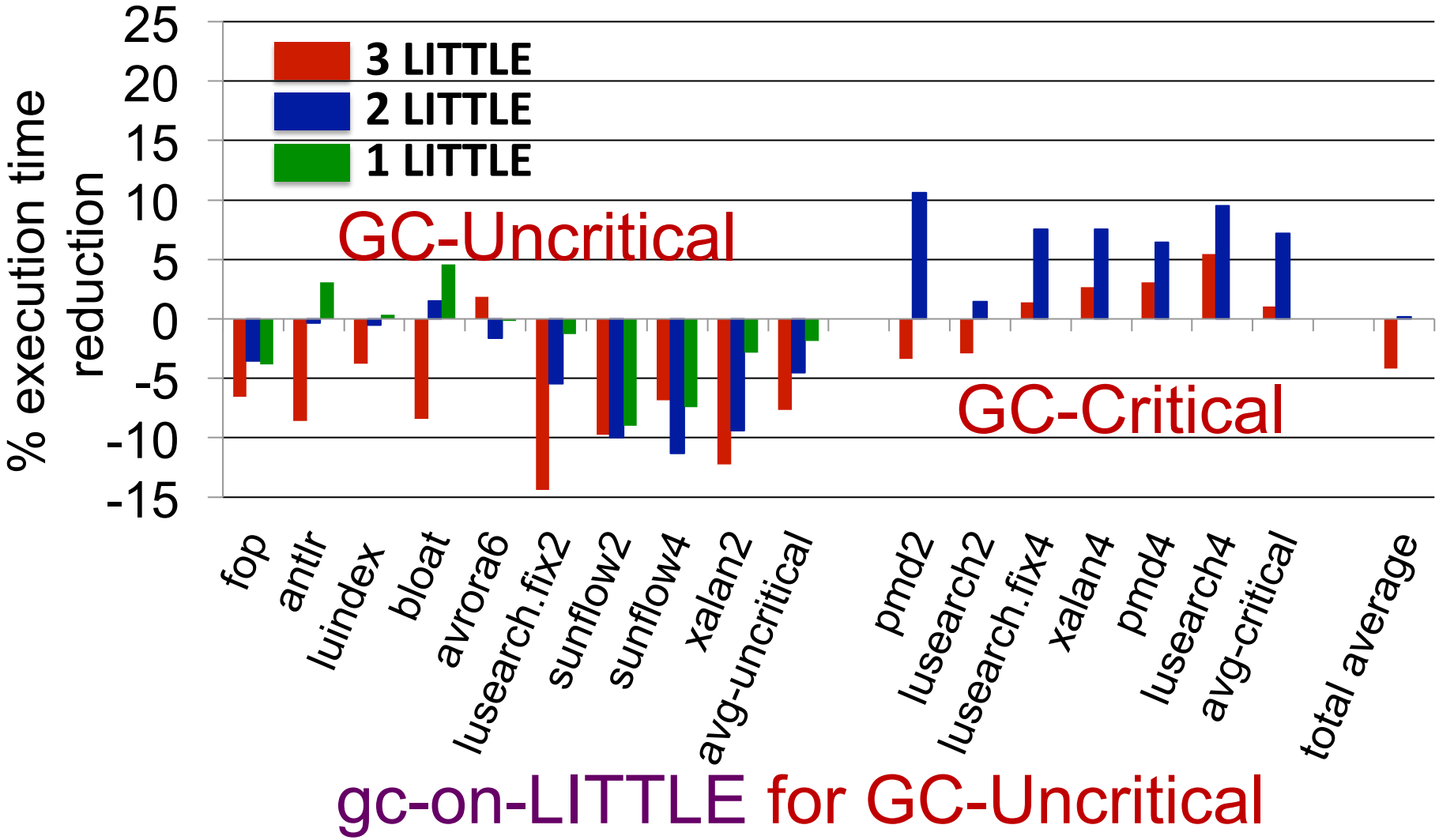


Giving GC Fair Share of Big Core



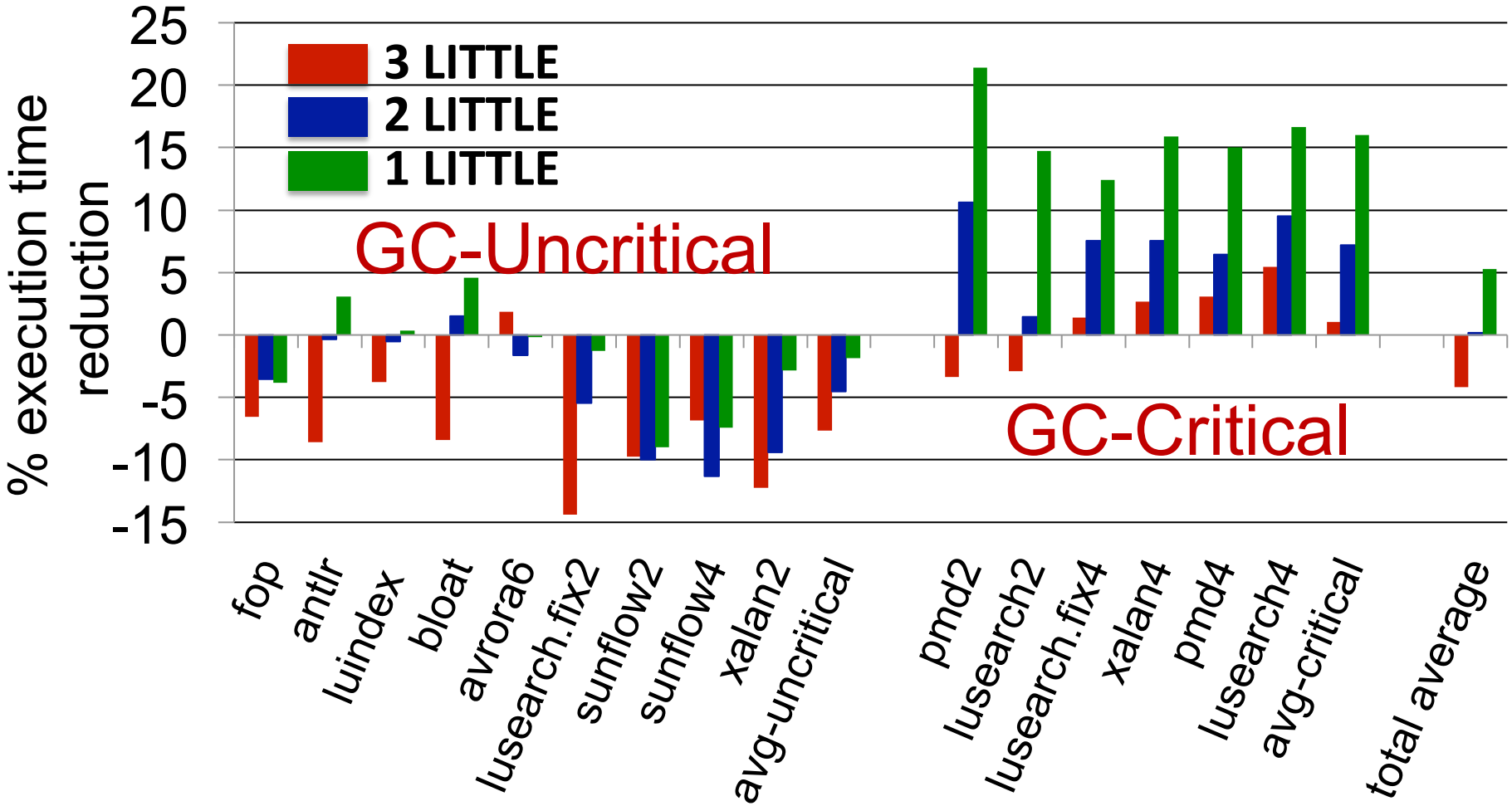
gc-on-LITTLE for GC-Uncritical

Giving GC Fair Share of Big Core



gc-on-LITTLE for GC-Uncritical

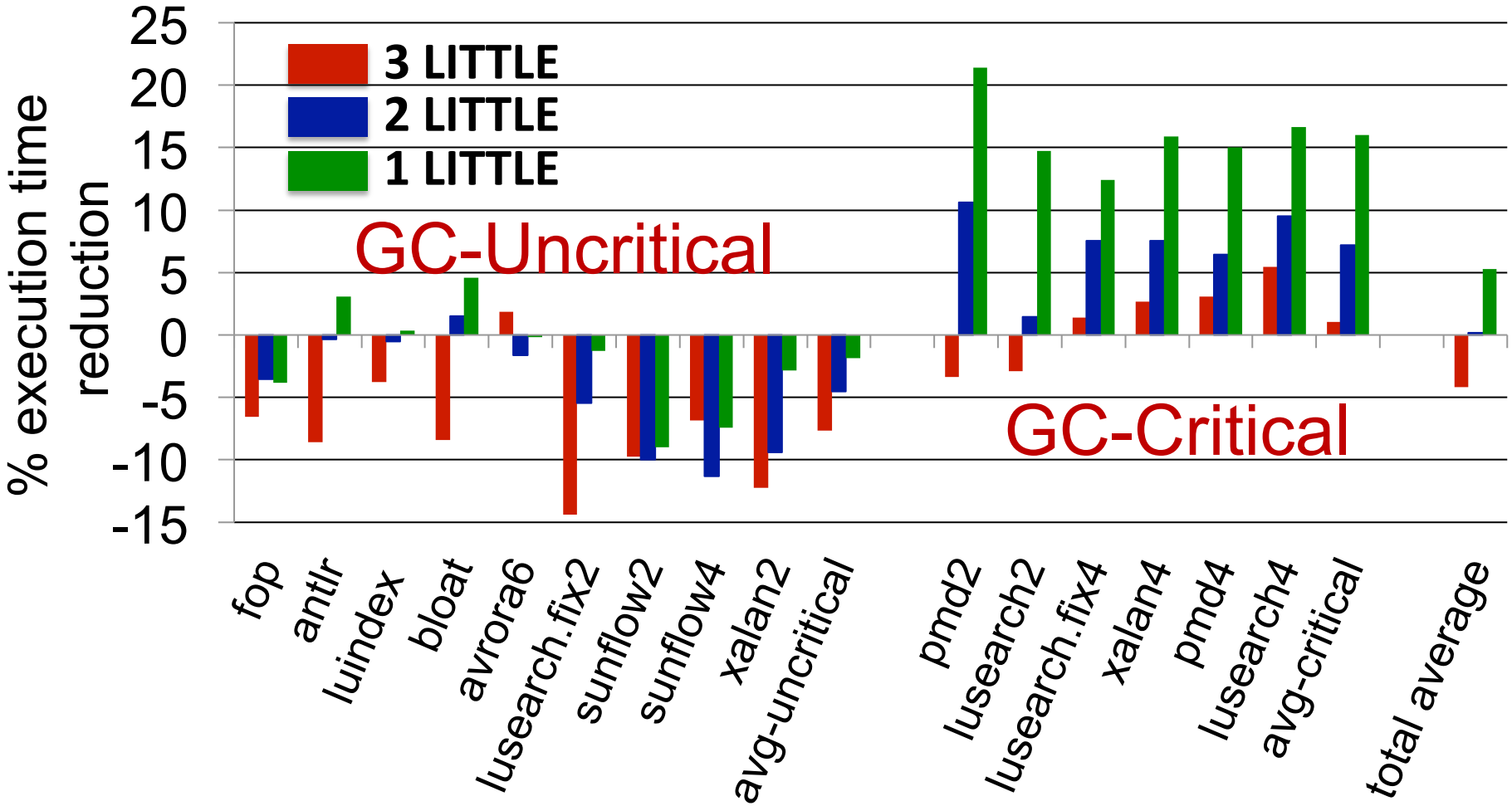
Giving GC Fair Share of Big Core



gc-on-LITTLE for GC-Uncritical

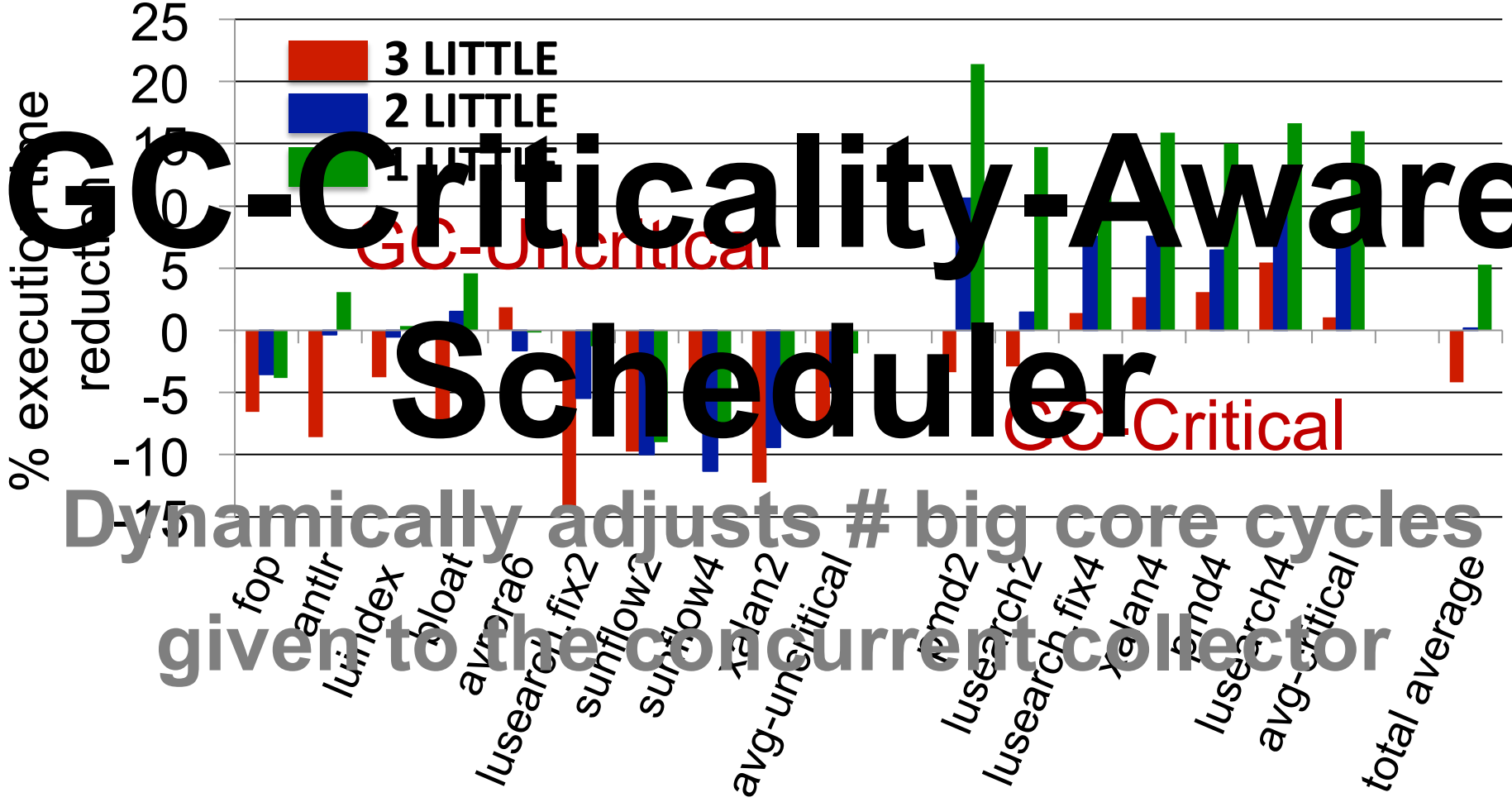
gc-fair for GC-Critical

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

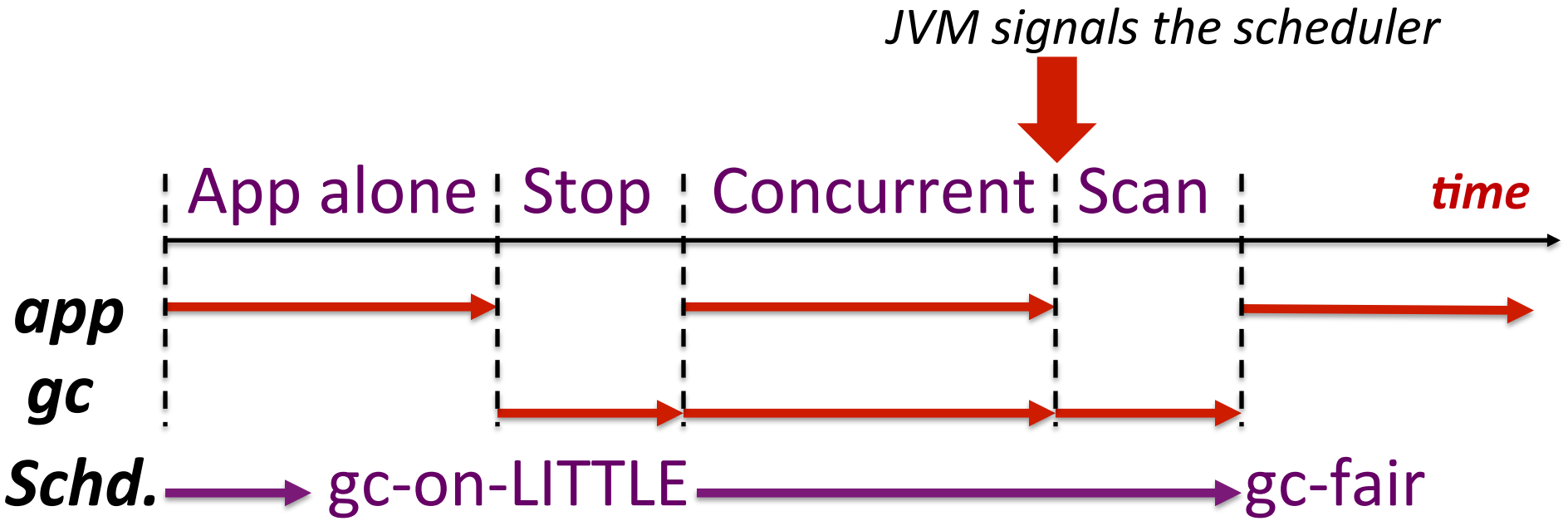
GC-Criticality-Aware Scheduler

Runtime Activity → How Scheduler Reacts?

GC-Criticality-Aware Scheduler
gc-on-LITTLE to gc-fair

GC-Criticality-Aware Scheduler

gc-on-LITTLE to gc-fair



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

Scheduler	How many quanta scheduled on the BIG core?
gc-on-LITTLE	First GC thread = 0, Second GC thread = 0
gc-fair	First GC thread = 1, Second GC thread = 1

Boost the priority of garbage

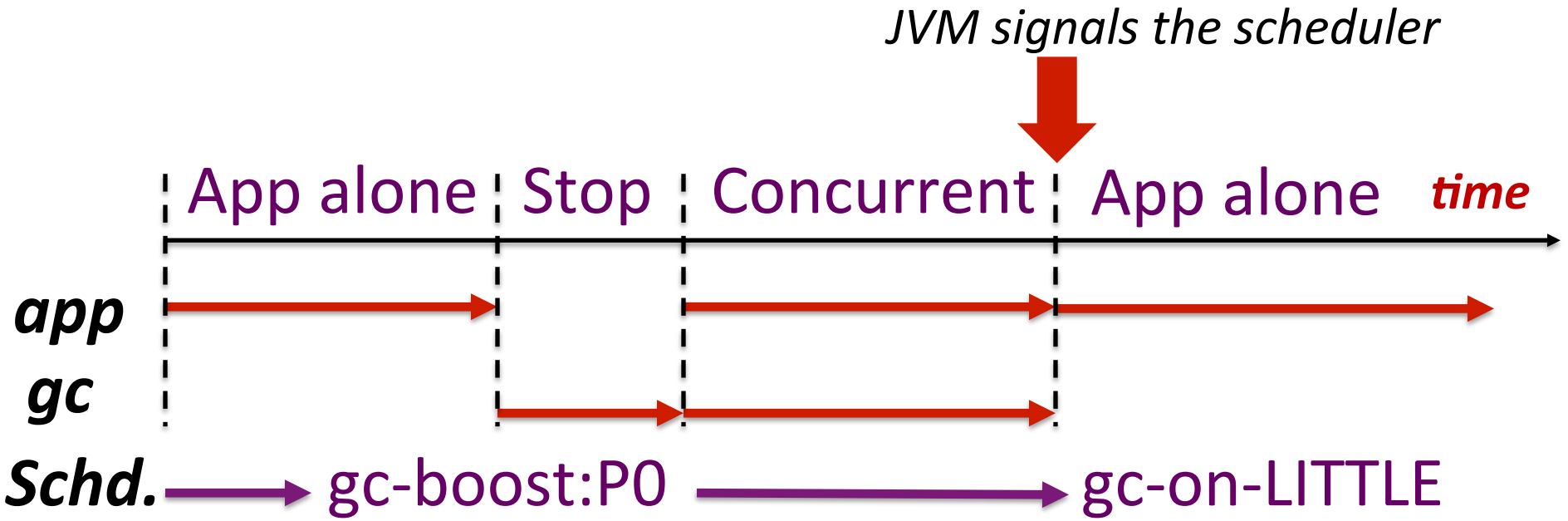
Give GC more consecutive quanta on big

Scheduler	State	How many quanta scheduled on the BIG core?
gc-boost	P0	First GC thread = 1, Second GC thread = 1
gc-boost	P1	First GC thread = 1, Second GC thread = 2
	…	

Degrade boost state when no longer critical

GC-Criticality-Aware Scheduler

gc-boost:P0 to gc-on-LITTLE



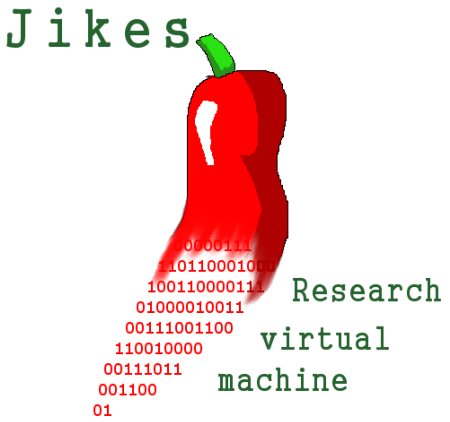
If no scan pause in state P0, go to gc-on-LITTLE
Can configure # zero stop scan intervals before returning to gc-on-LITTLE

GC-Criticality-Aware Scheduler
Summary

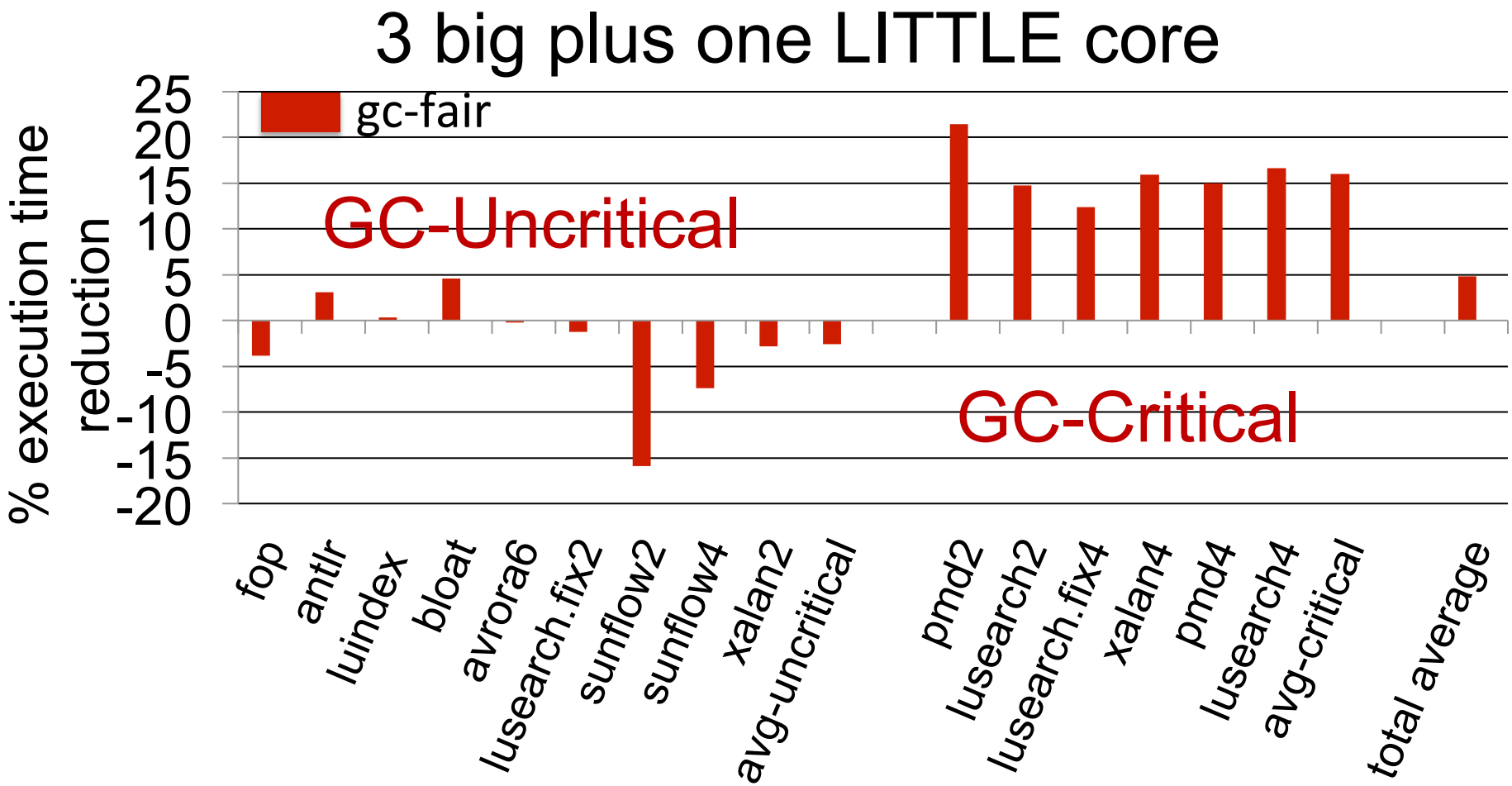
- JVM detects GC-Criticality during runtime
- Communicates criticality information down to the scheduler
- Scheduler dynamically adapts big core cycles given to GC

Experimental Setup

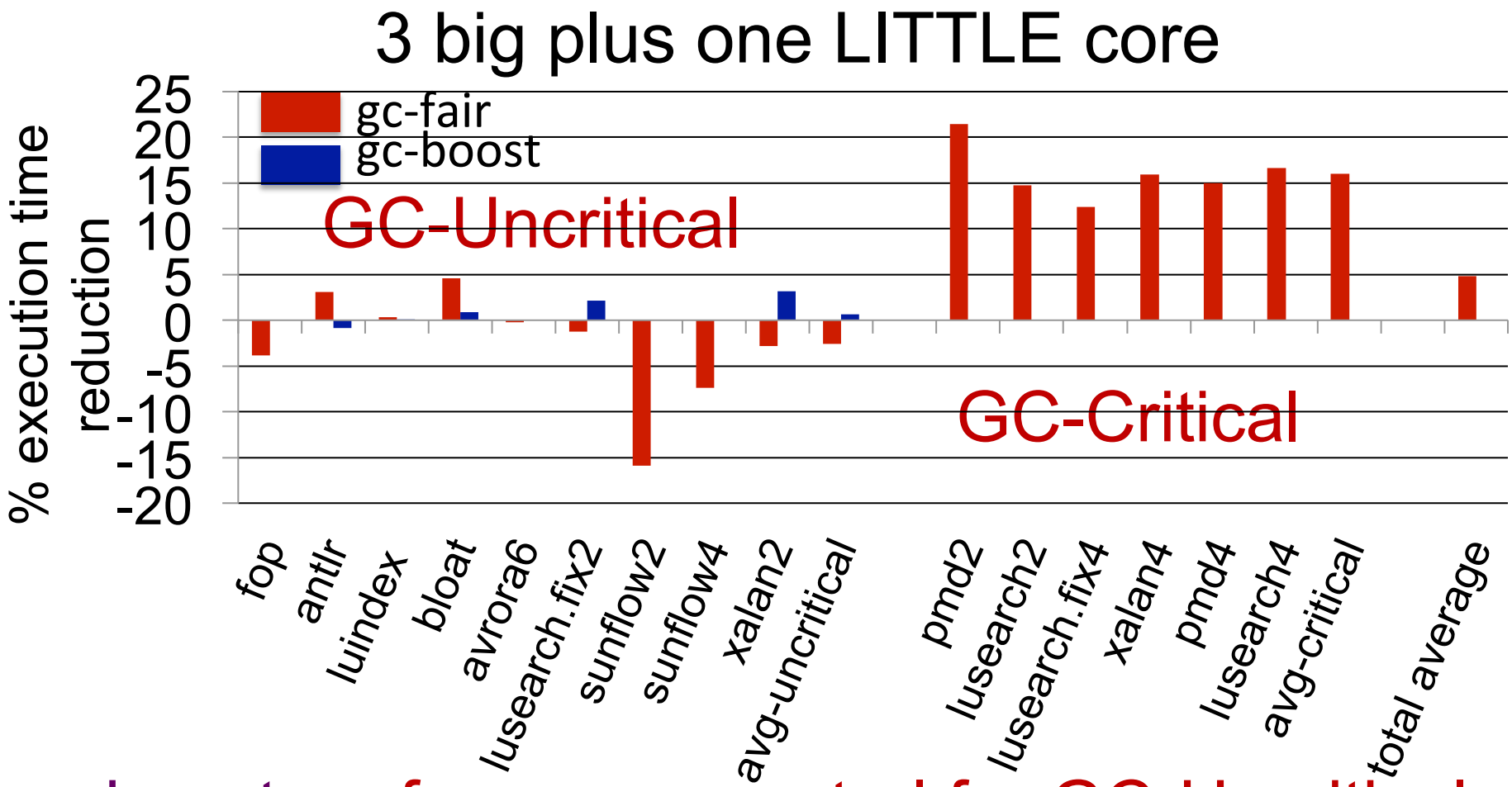
- Java Virtual Machine
  - Jikes Research Virtual Machine (Version 3.1.2)
  - Full-heap concurrent collector with two threads
  - Tackle non-determinism by warming up the JVM
  - Heap size 2x of minimum
- Benchmarks
  - Ten benchmarks from DaCapo
  - Vary the # threads – 1 to 4
- Heterogeneous Multicore Setup
  - Sniper multicore simulator (Version 4.0)
  - Different four core heterogeneous architectures
  - Varying # of big and LITTLE cores



Performance of GC-Criticality-Aware Scheduler

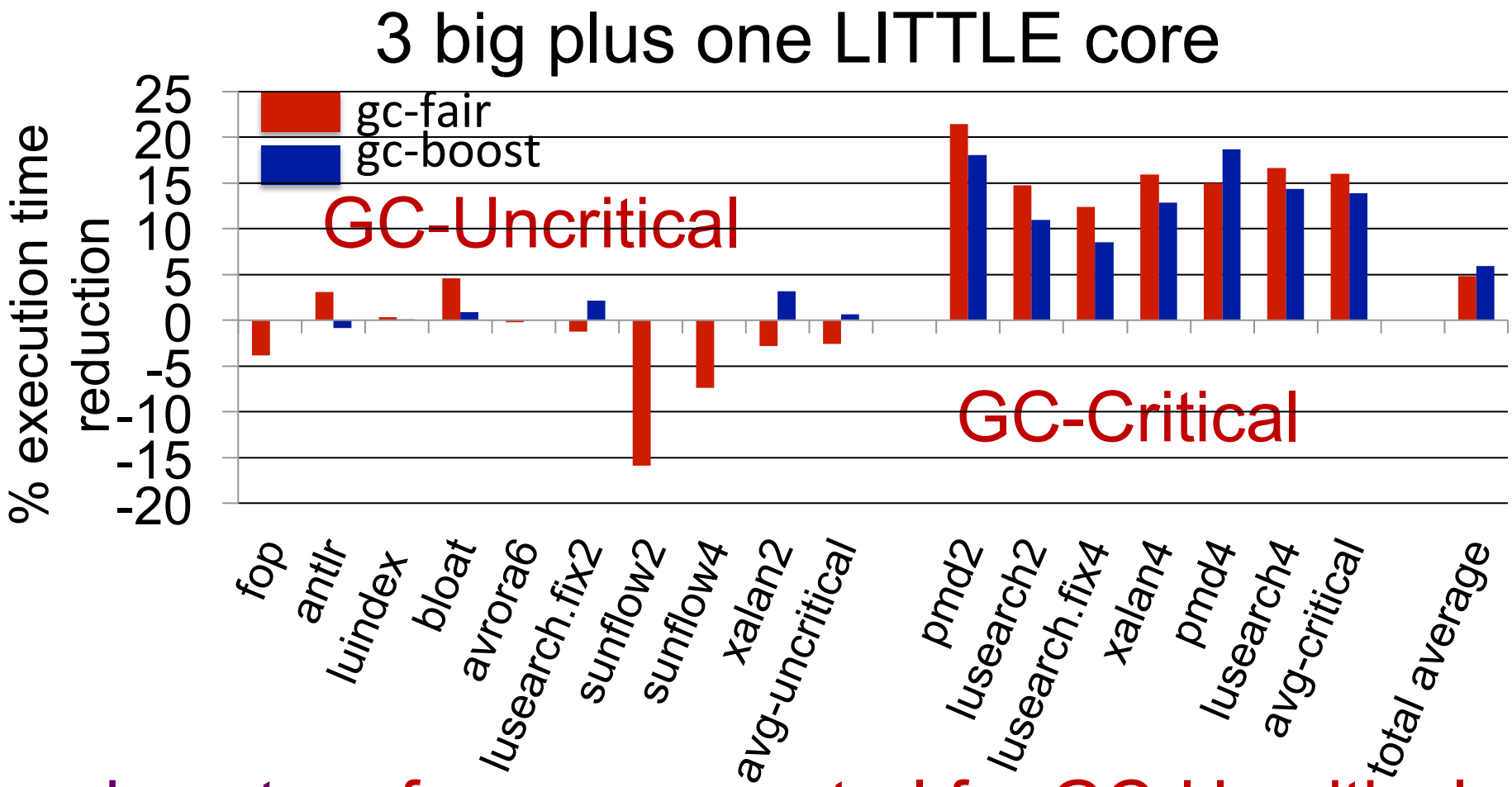


Performance of GC-Criticality-Aware Scheduler



gc-boost performance neutral for GC-Uncritical

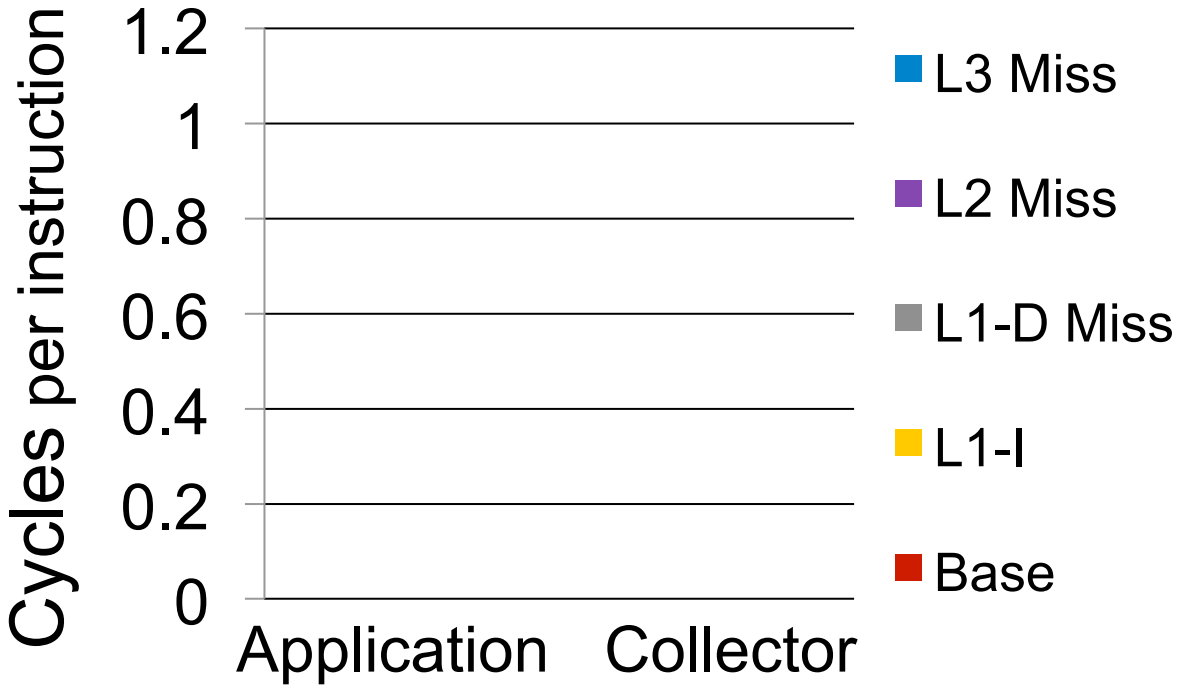
Performance of GC-Criticality-Aware Scheduler



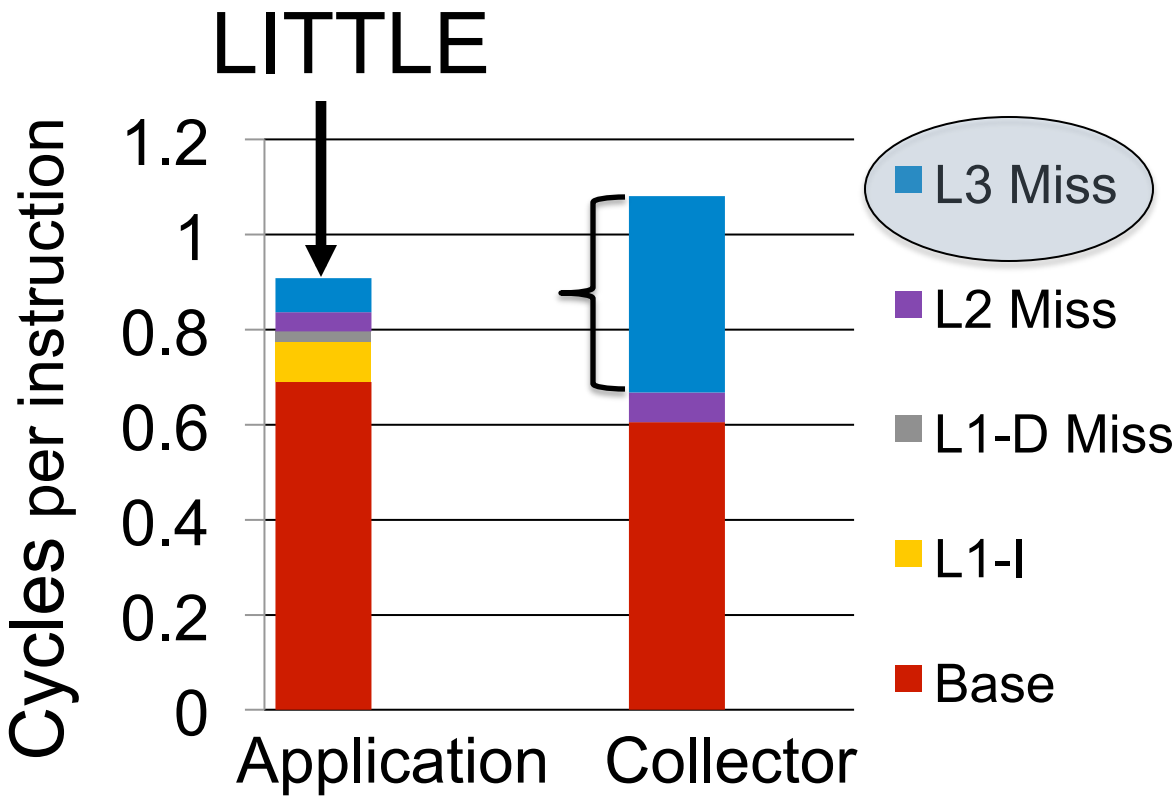
gc-boost performance neutral for GC-Uncritical

Improves perf. of GC-Critical by 14% on avg.

Understanding the Performance Advantage of Big Core

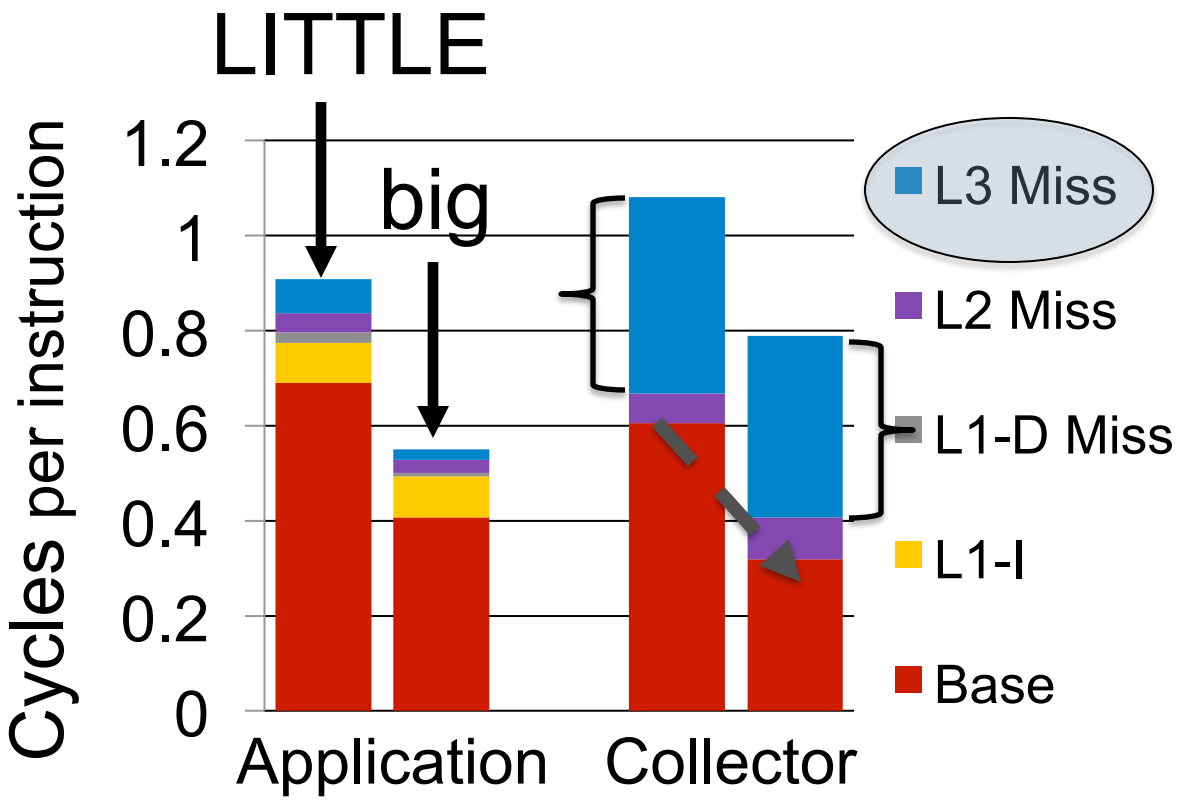


Understanding the Performance Advantage of Big Core



Collector performs a heap traversal chasing pointers

Understanding the Performance Advantage of Big Core

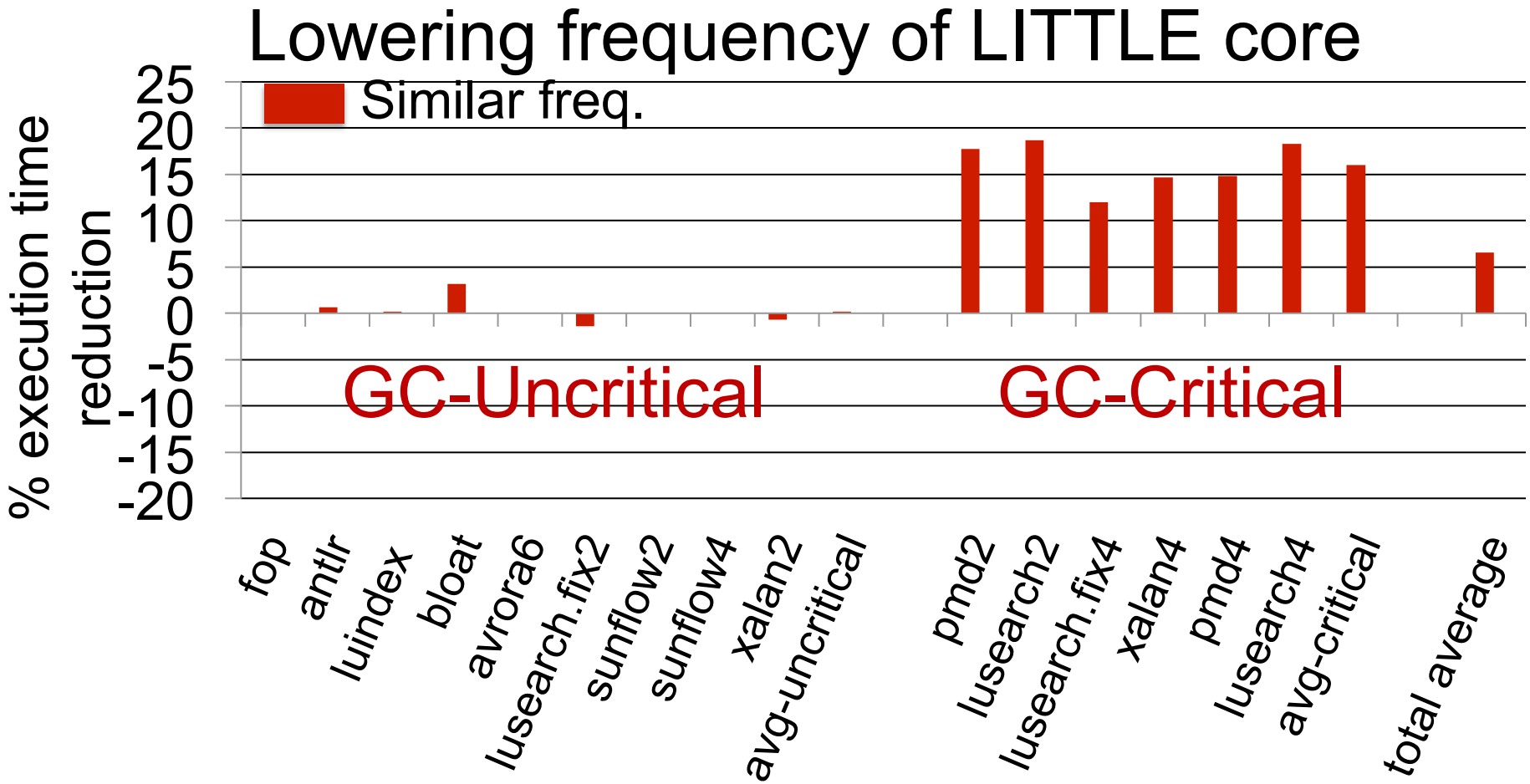


Collector performs a heap traversal chasing pointers

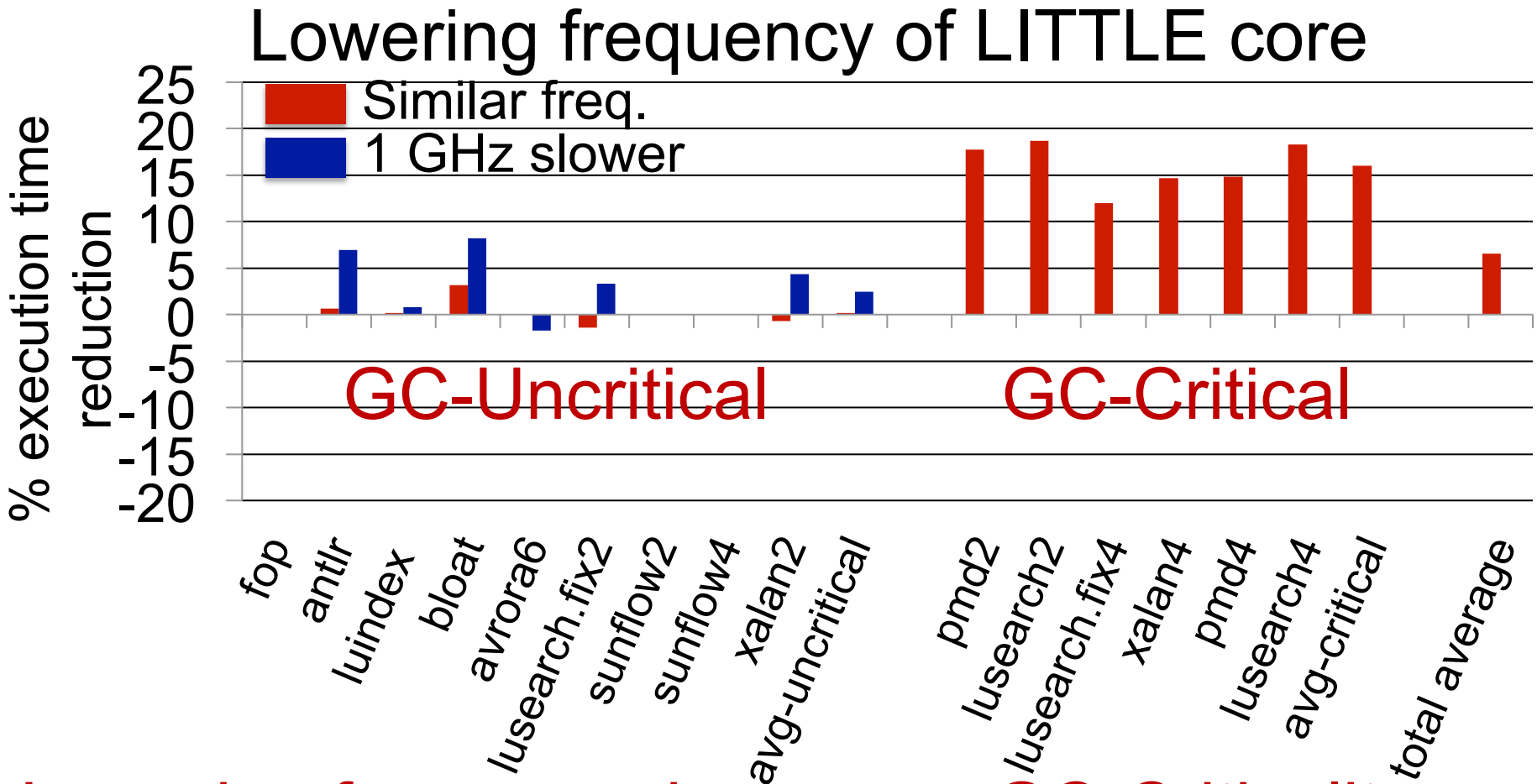
Instruction-level parallelism ☺

Memory-level parallelism ☹

Performance of GC-Criticality-Aware Scheduler

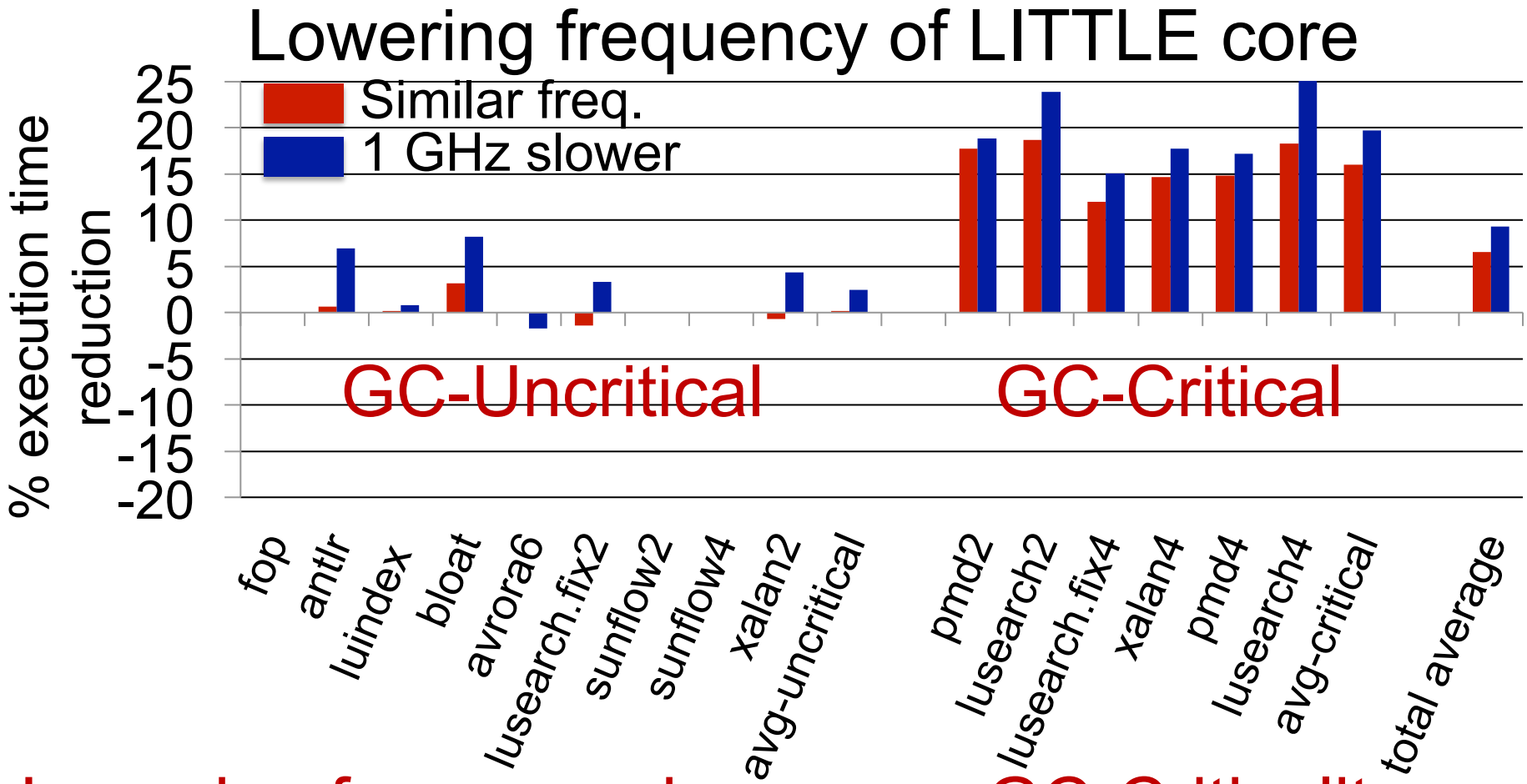


Performance of GC-Criticality-Aware Scheduler



Lowering frequency increases GC-Criticality

Performance of GC-Criticality-Aware Scheduler

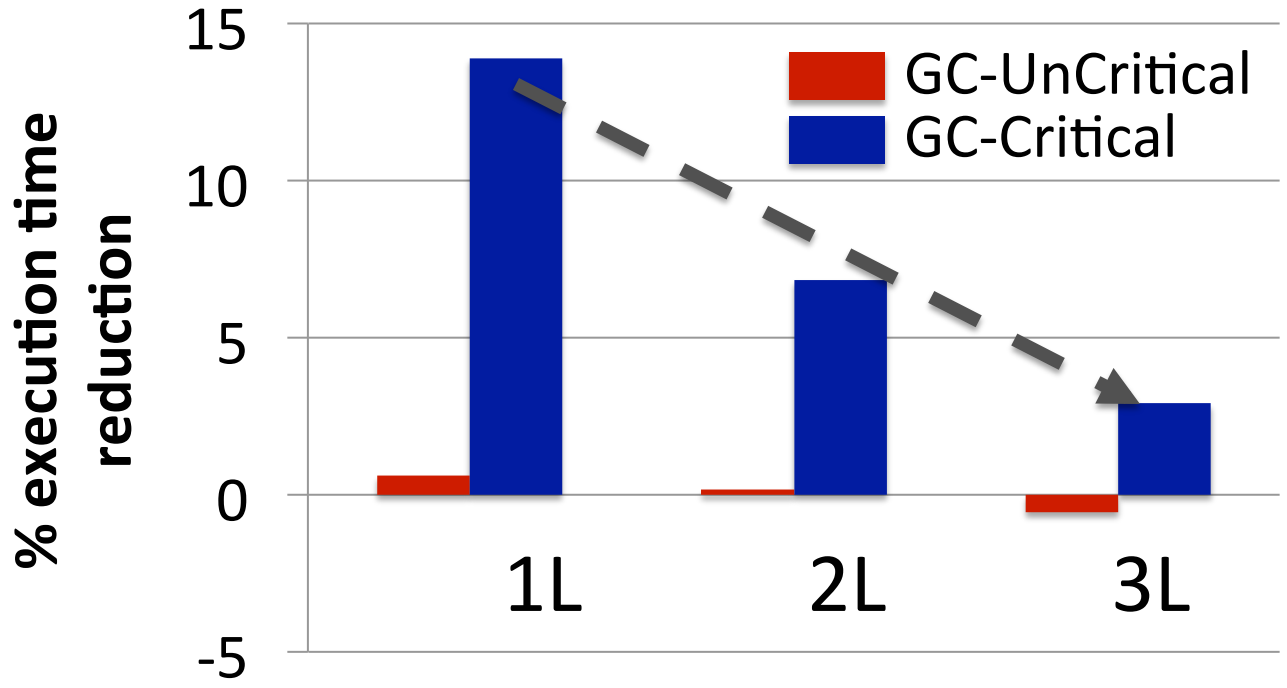


Lowering frequency increases GC-Criticality

Improves perf. of GC-Critical by 20% on avg.

Performance of GC-Criticality-Aware Scheduler

Different # LITTLE cores

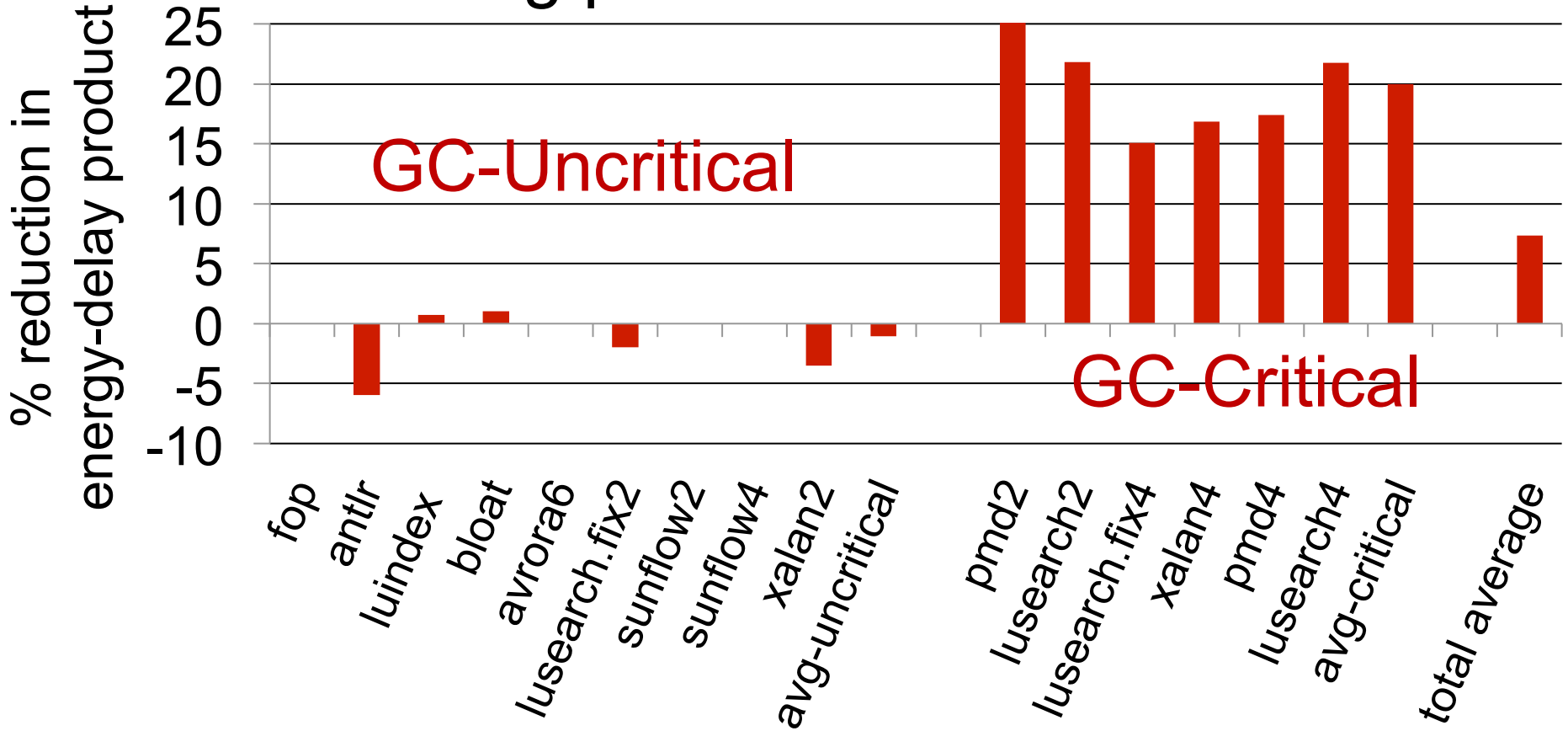


Allocation rate lowers with more LITTLE cores

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

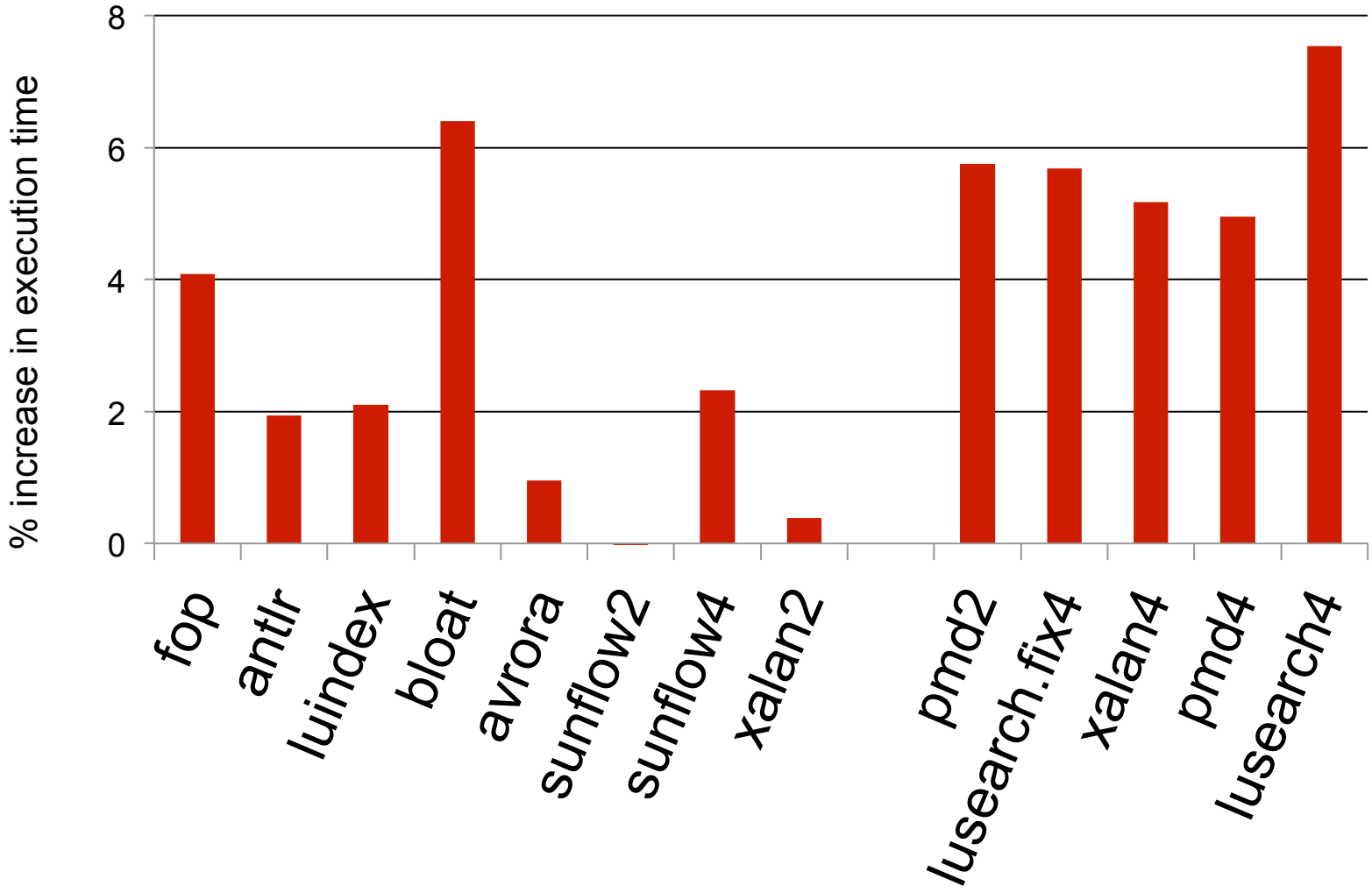
GHENT UNIVERSITY

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

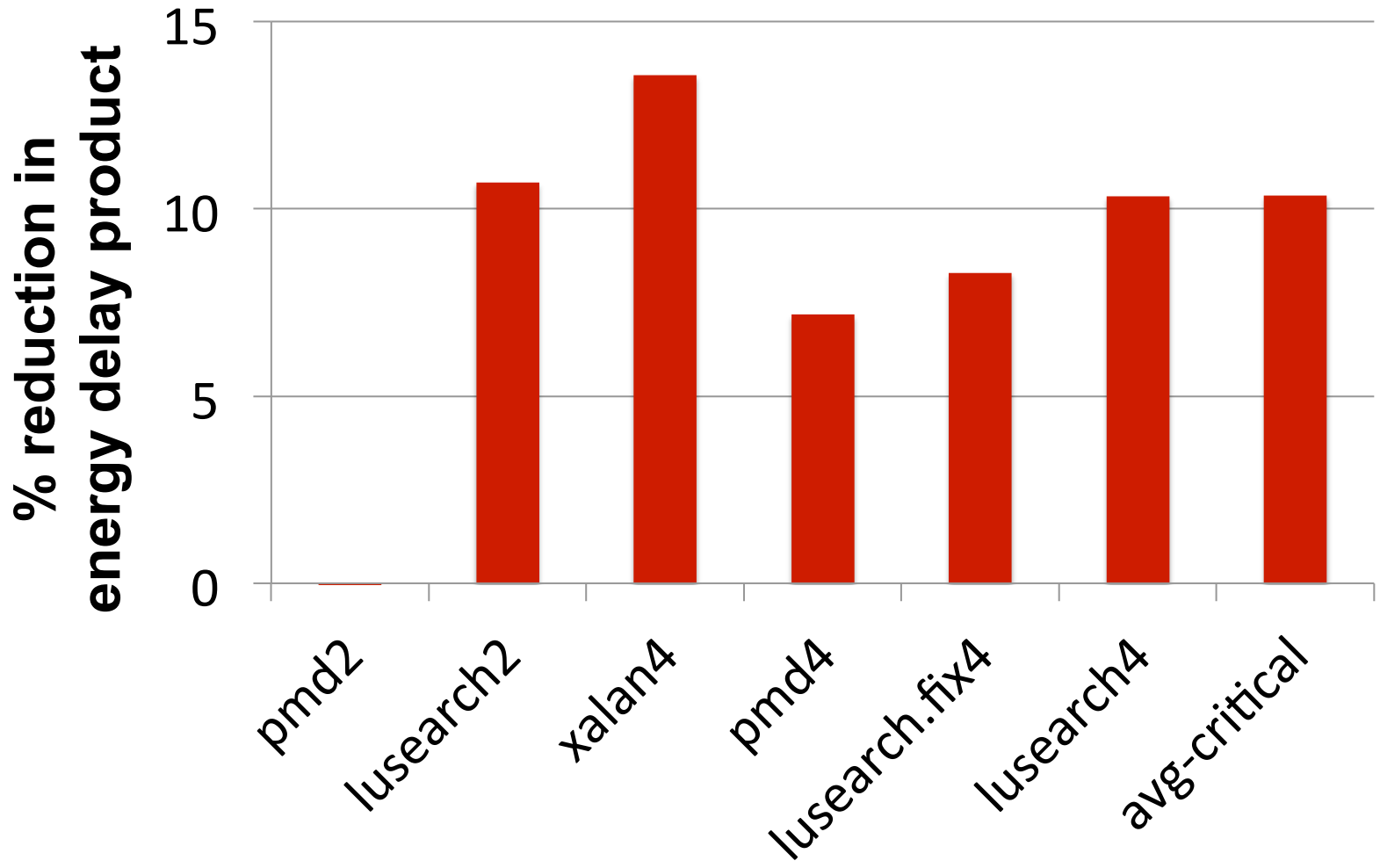
Thank You !

Shoaib.Akram@UGent.be
http://users.elis.ugent.be/~sakram

GC Criticality with OpenJDK's CMS

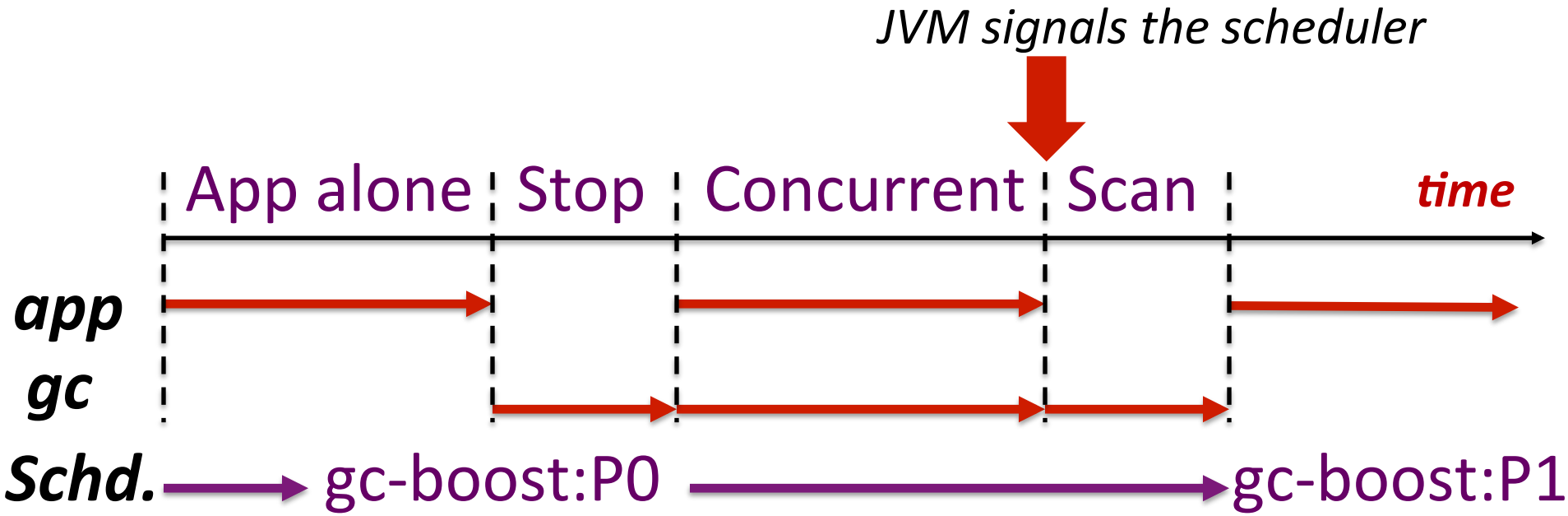


Triggering Concurrent GC Every 32 MB of Allocation



GC-Criticality-Aware Scheduler

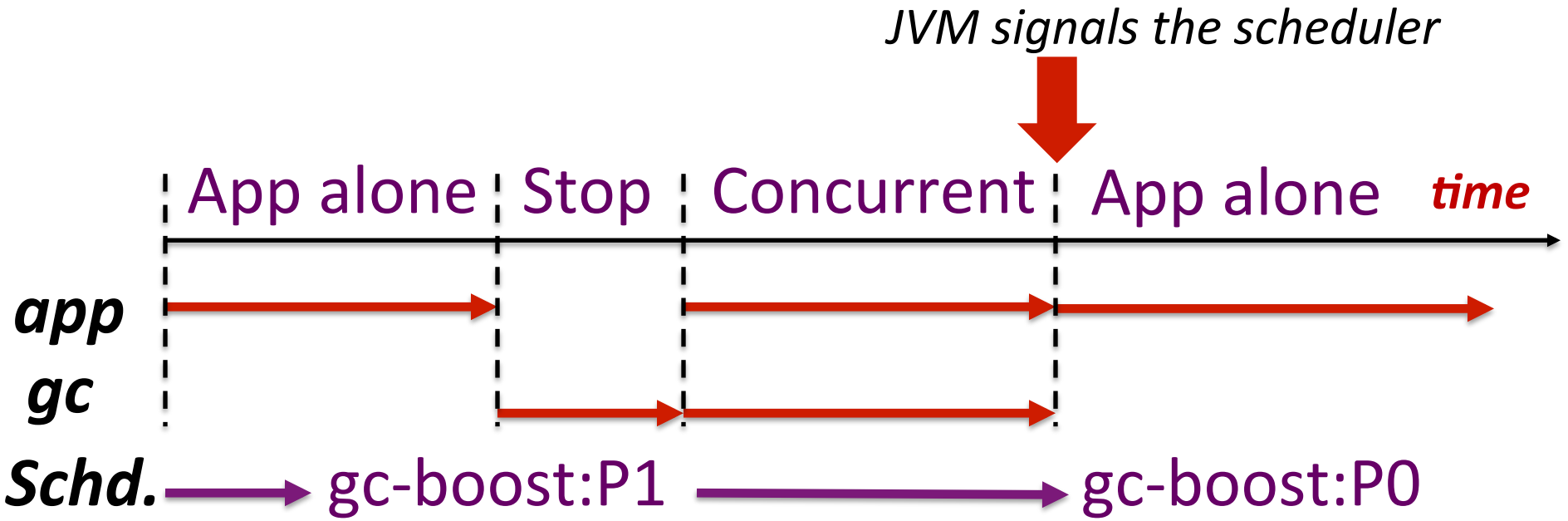
gc-boost:P0 to gc-boost:P1



gc-boost:P1 gives GC two quanta on big

GC-Criticality-Aware Scheduler

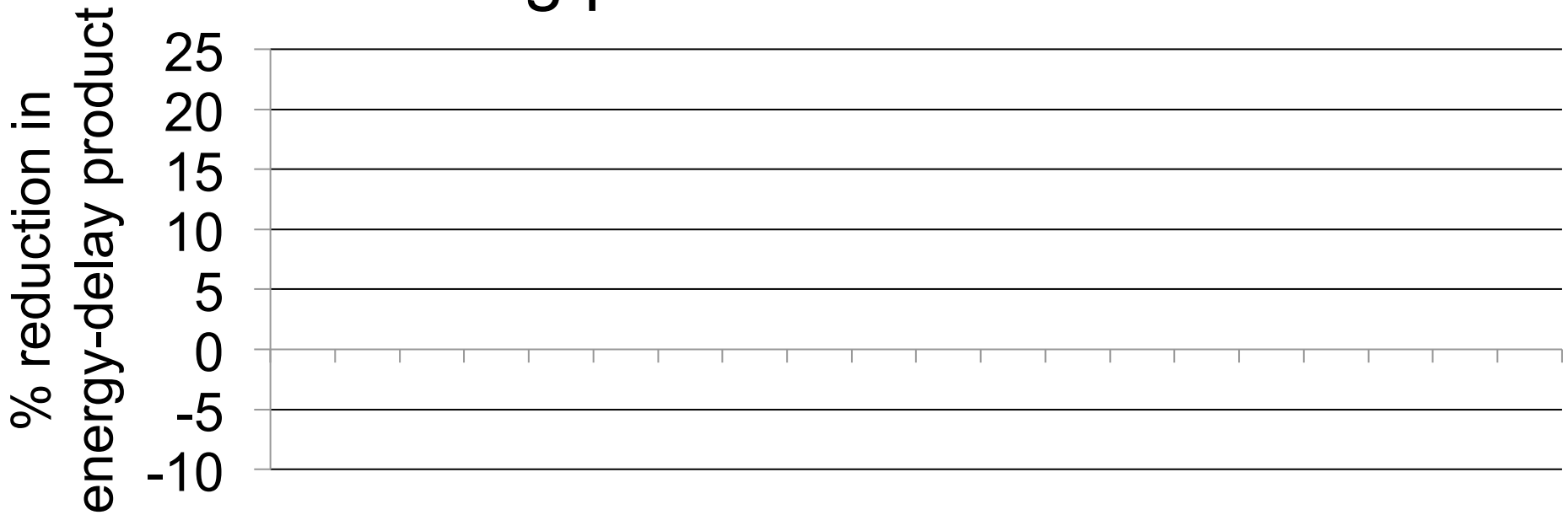
gc-boost:P1 to gc-boost:P0



Degrade boost state if no stop scan pause

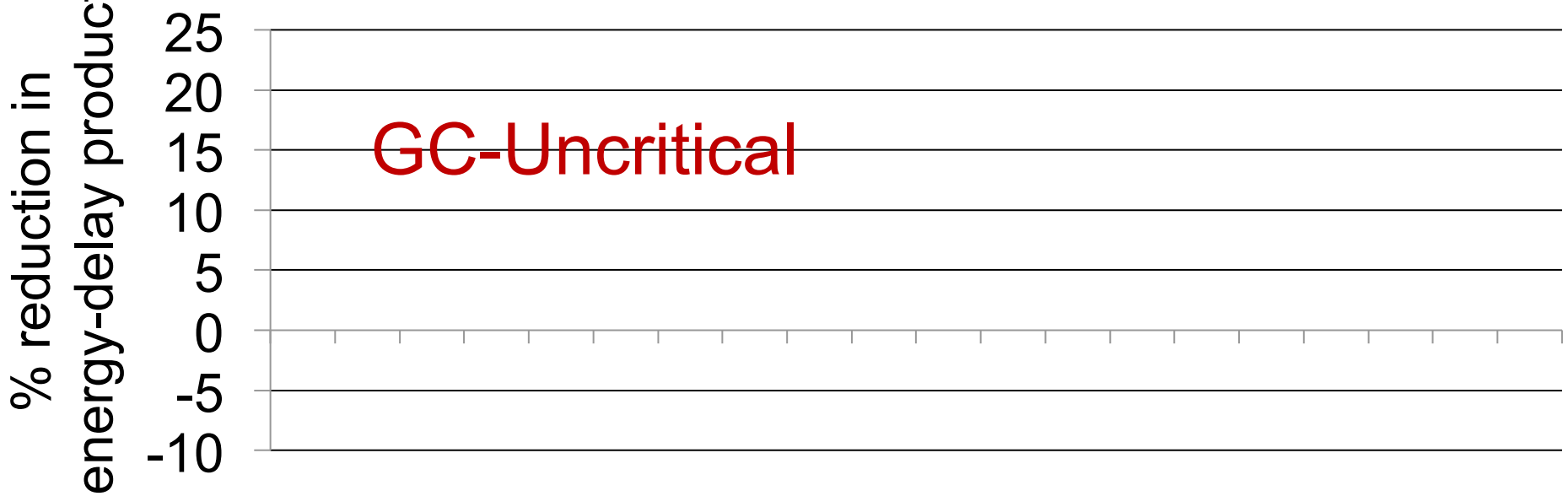
Energy Efficiency of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core



Energy Efficiency of GC-Criticality-Aware Scheduler

3 big plus one LITTLE core



Negligible change in EDP for GC-Uncritical