Exploring Multi-Threaded Java Application Performance on Multicore Hardware[1]

Jennifer B. Sartor Lieven Eeckhout

Presented by Moritz Hoffmann

May 8, 2013

Objective

Analyze effects of multi-core and multi-socket systems on managed languages.

Motivation

- Many applications written in managed languages, for example Java running on the JVM.
- Runtime environment performs additional tasks.
  - Garbage collection
  - Compilation
  - Support functions
- Performance impact of tasks on multi-core/multi-socket platforms research topic.

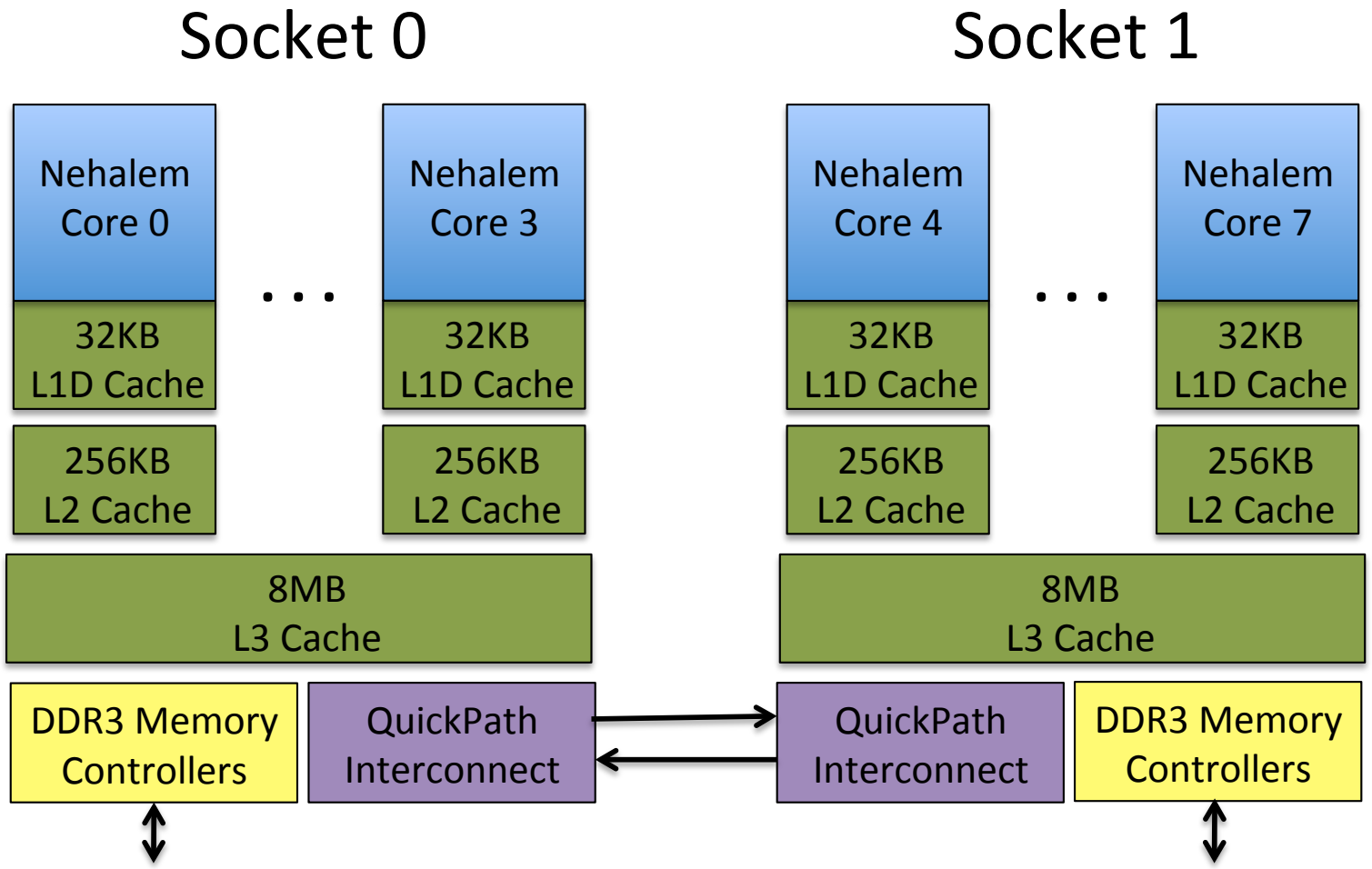
Test criteria

- Factors to vary:
  - Pinning of threads (application, utility) to cores
  - Pinning to sockets
  - Frequency of cores
  - Memory availability (heap size)

Test environment

- Intel Nehalem, 2012
- Two sockets
- Quad cores per socket
- No hyperthreading
- Clock frequency 1.5GHz, 2.0GHz and 3GHz
- Different heap sizes to trigger utility threads in different patterns.

System architecture



Benchmarks

- Avrora
- Lusearch, plus fixed version lusearch-fix
- Pmd
- Sunflow
- Xalan
- Pjbb2005

Benchmark results I

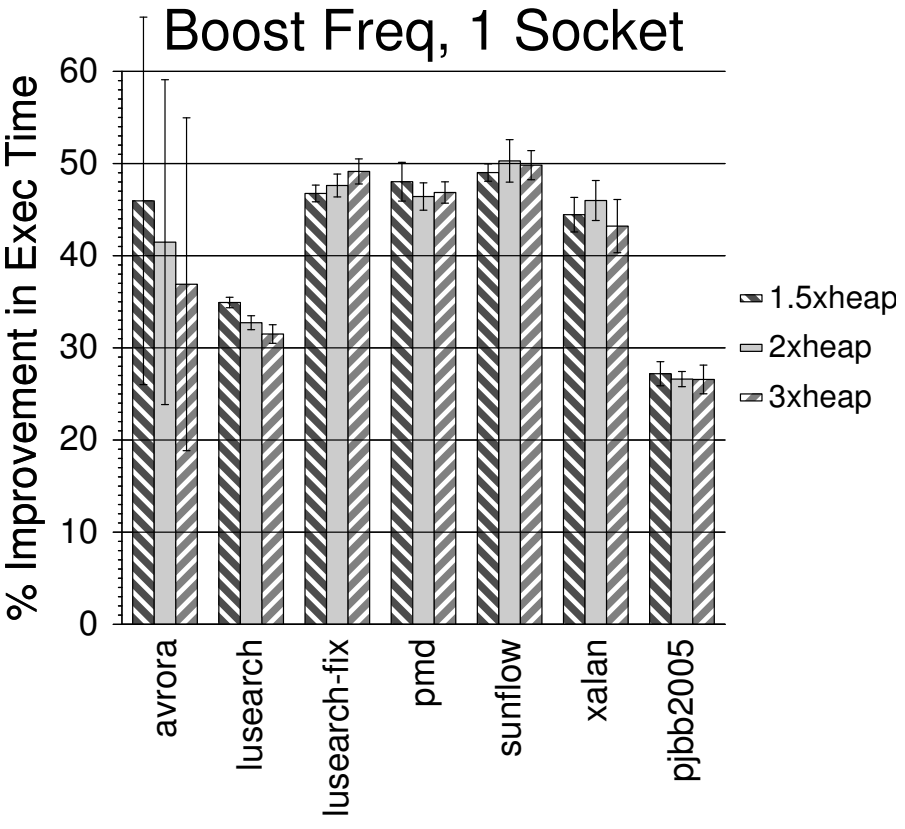


Figure 2. Percent execution time improvement when boosting frequency from lowest to highest on one socket.

Benchmark results II

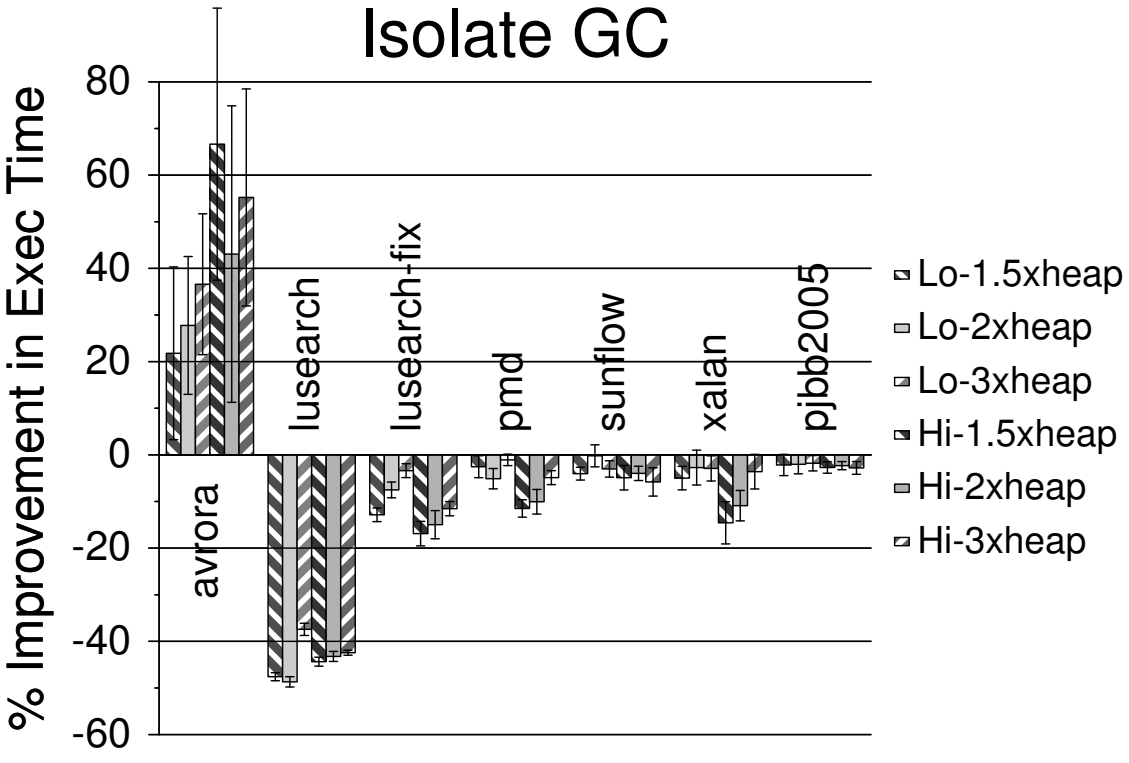


Figure 3. Percent execution time improvement of isolating four collector threads to a second socket.

Benchmark results III

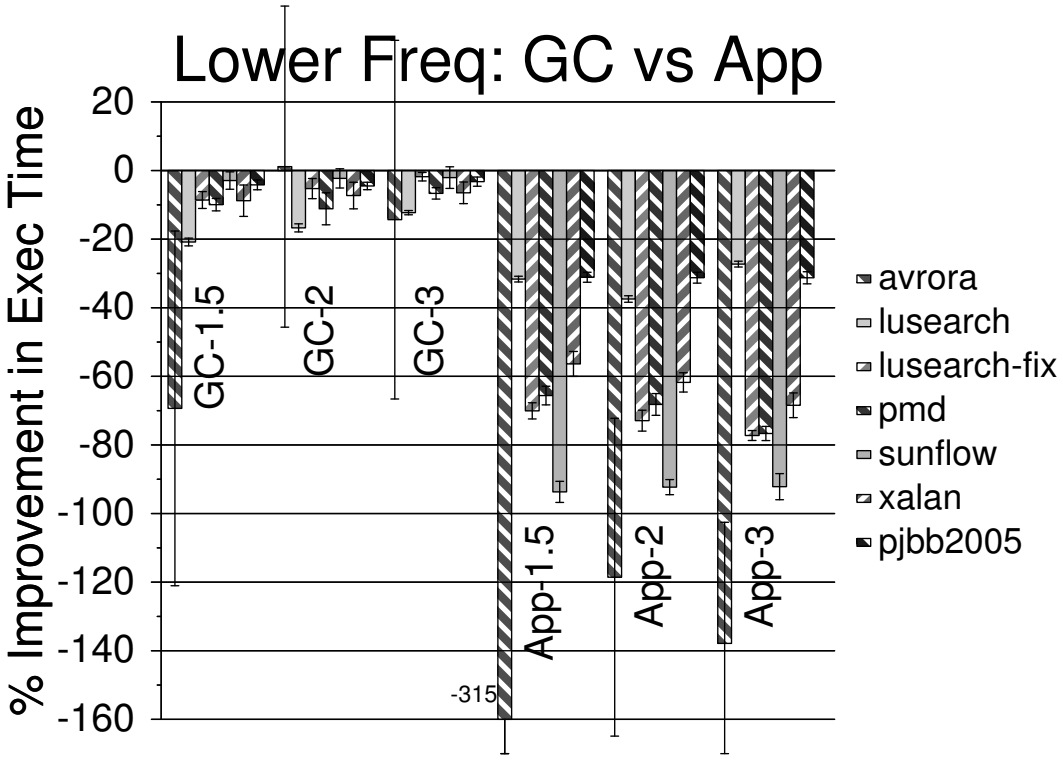


Figure 12. Percent execution time improvement when lowering frequency from highest to lowest, either four collector threads, or application (plus the rest) socket.

Benchmark results IV

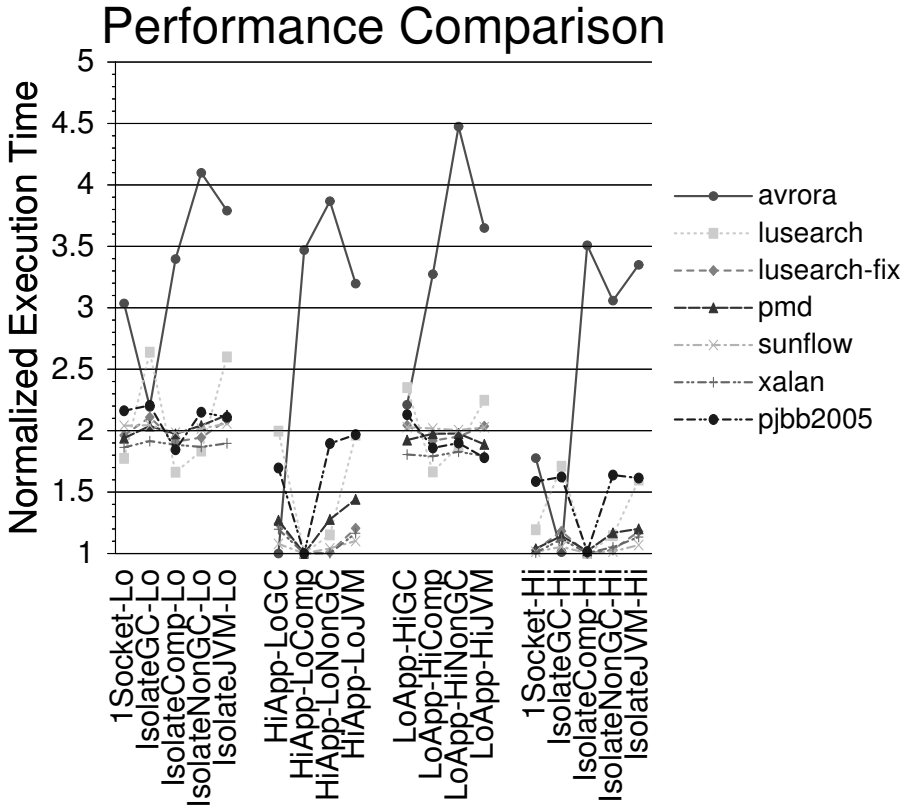


Figure 19. Execution time comparison normalized to minimum with different thread mappings and scaling frequencies, at twice the minimum heap size.

Benchmark results V

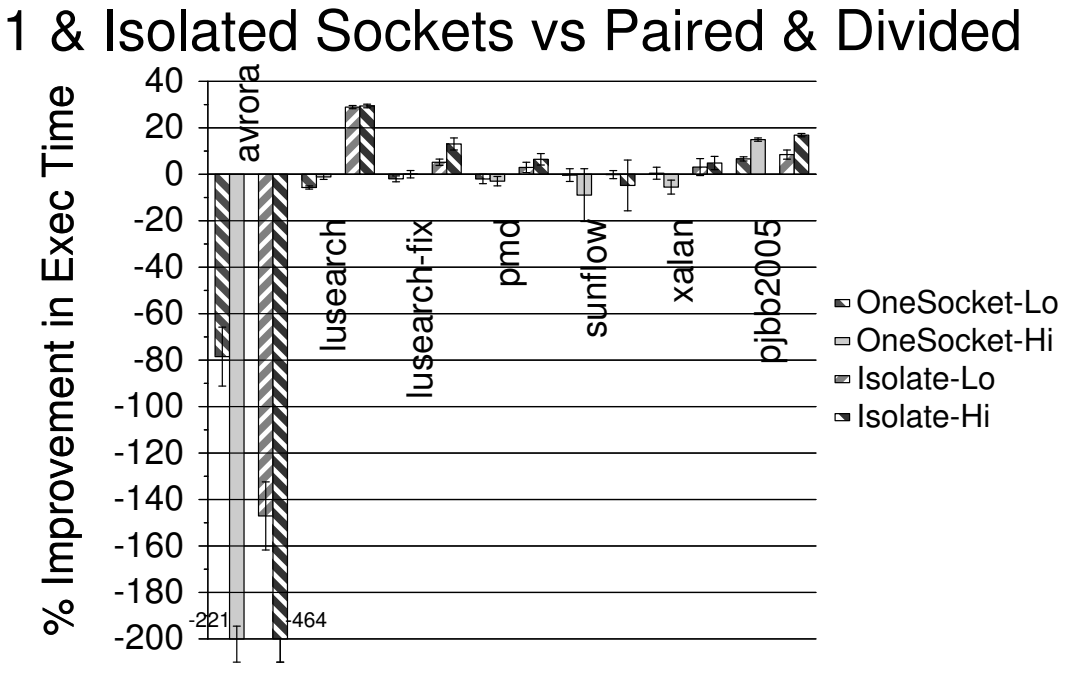


Figure 20. Percent execution time improvement going from all threads on one socket and from isolating four collector threads on a second socket to pairing application and collector threads and placing half each socket, at twice the minimum heap size.

Benchmark results VI

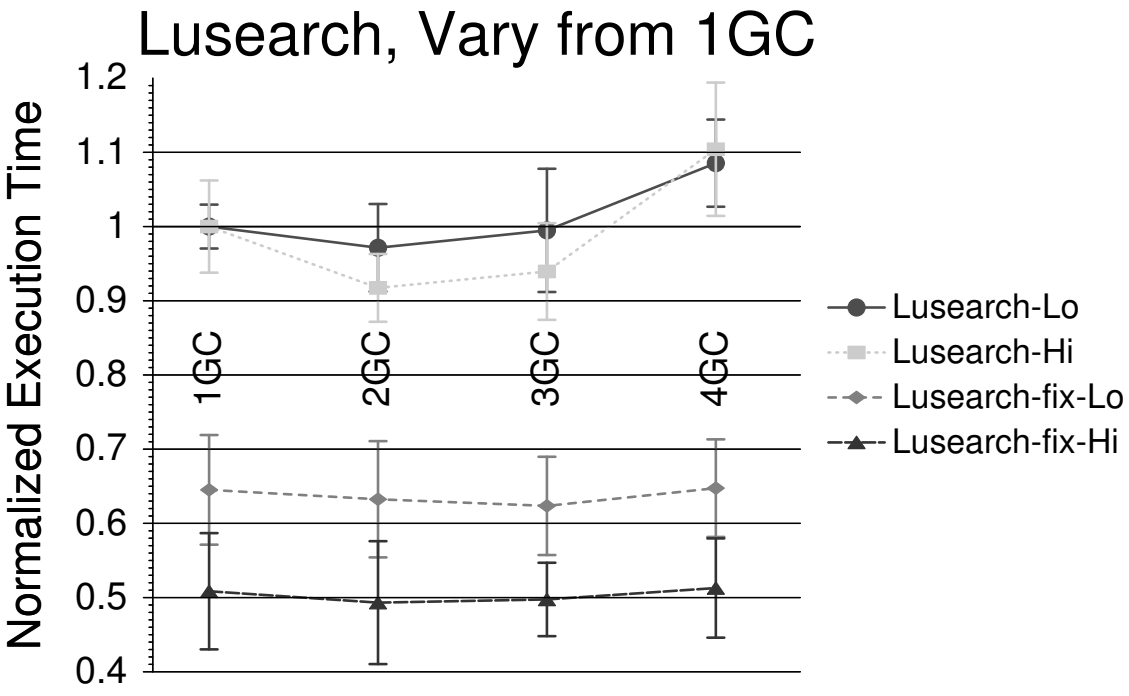


Figure 24. Execution time comparison for lusearch and lusearch-fix varying the number of collector threads normalized to lusearch with one collector thread, on one socket at twice the minimum heap size.

Results

- Factors have an impact on performance.
- Dependent on benchmark.
- Other factors have similar performance effects on all benchmarks.
- Factors have a different impact during startup and steady-state operation.

Related work

Understanding JVM behavior

- Analyze importance of utility threads versus application threads.
- Impact of hardware characteristics, such as memory availability, cache size.
- Typically 20%, ranges from 10% to 55%[2]

DVFS (Dynamic Voltage and Frequency Scaling)

- Save power while keeping performance.

Conclusion

- Core frequency has strong influence on performance.
- Garbage collection has a strong influence depending on workload. Separation to different sockets/cores might degrade performance.
- Future work should focus on closer operating system and runtime environment integration.

Impact

- No citations.
- Research for performance analysis has to be done closer to hardware.

For Further Reading I

1. Exploring multi-threaded Java application performance on multicore hardware; Sartor, Jennfer B and Eeckhout, Lieven; Proceedings of the ACM international conference on Object oriented programming systems languages and applications; page 281–296, 2012; ACM
2. T. Cao, S. M. Blackburn, T. Gao, and K. S. McKinley. The yin and yang of power and performance for asymmetric hardware and managed software. In The 39th International Symposium on Computer Architecture (ISCA), pages 225–236, June 2012.