

Concurrent Multiprocess Benchmarking of Java Garbage Collection

Jacob Warrington

Advised by Dr. Tony Hosking

Purpose

Build on the results of previous research by testing multiple built-in Java garbage collectors with an identical benchmark, allowing us to directly compare the performance of multiple GC systems.

Background

- Dr. Hosking's publication:

 - “Portable, mostly-concurrent, mostly-copying garbage collection for multi-processors”

- GCOld Benchmark

- Expectations

Process

- Expand the Java GCOld benchmark to support multithreading
- Using tunable Java Virtual Machine parameters to set environment and test multiple GCs
- Data mining and sorting

Results Overview

○ Data quality

- Consistent across multiple benchmark runs
- Graphs plotted from averages of timing results

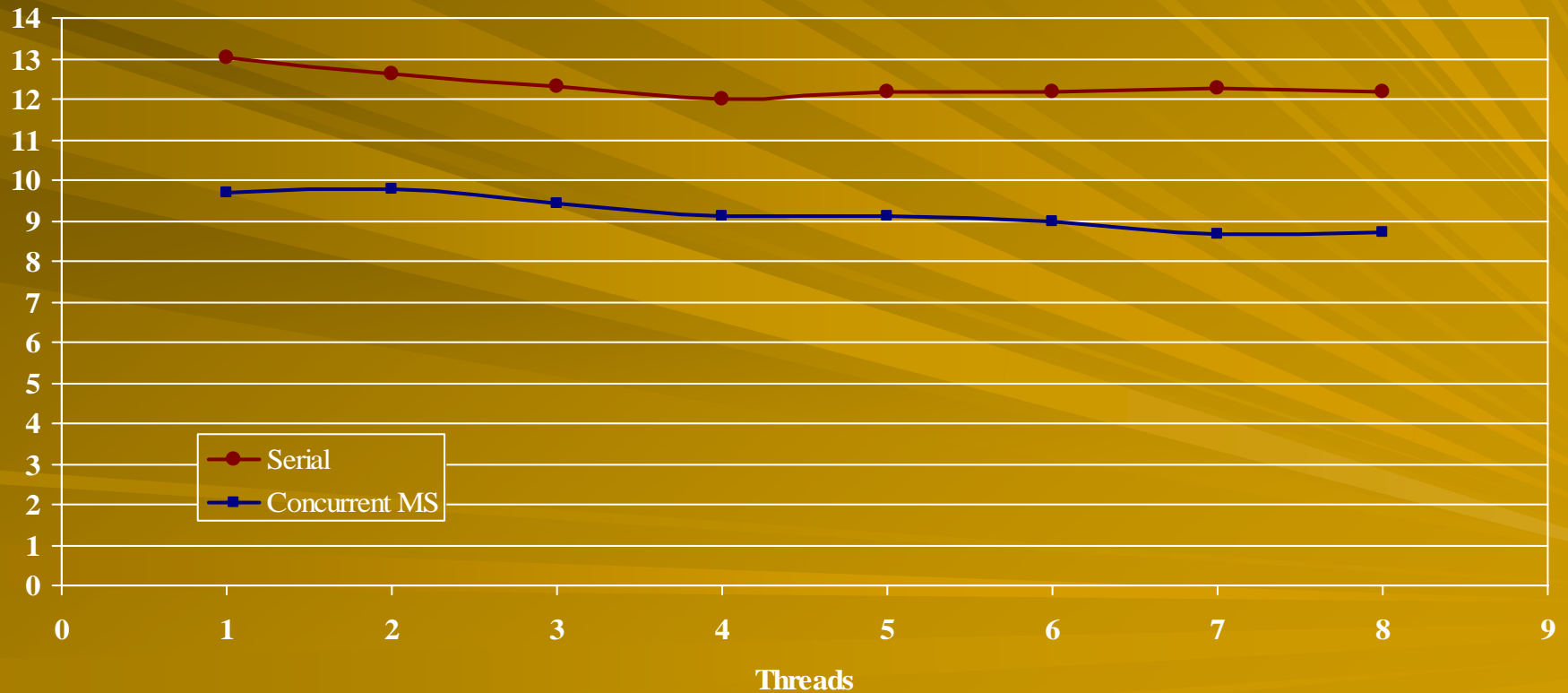
○ Multiprocessor scalability reflects expected trends

○ Noted variations

- Slower performance overall
- Fastest GC not conditional, as in Modula-3 testing; Java Concurrent Mark-Sweep collector always fastest

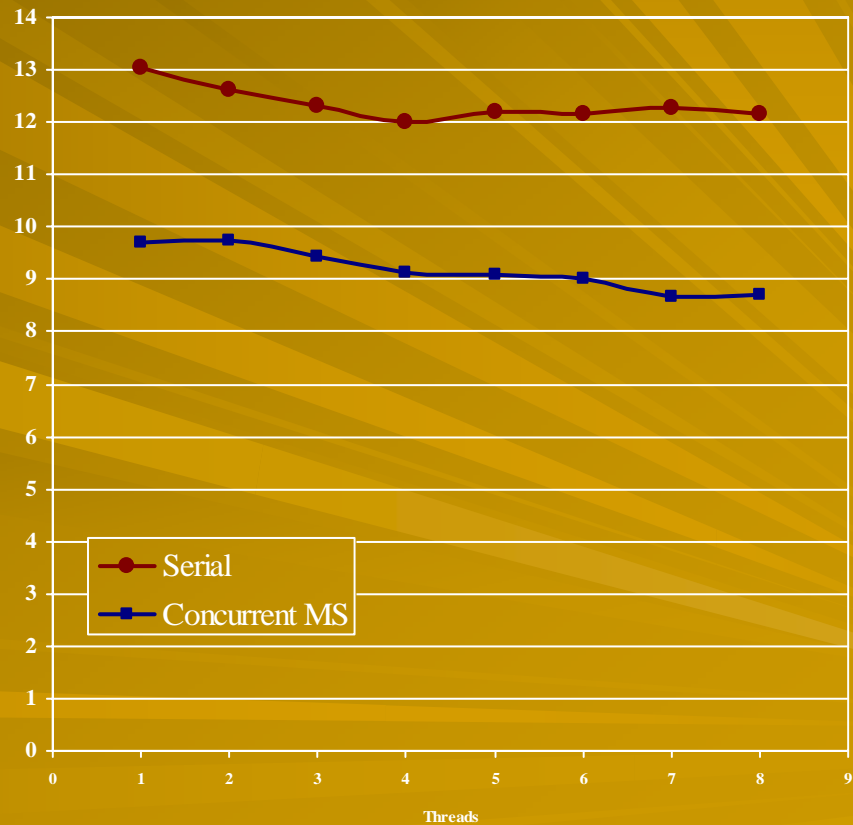
Experimental Results with Java Garbage Collectors

Elapsed Time Per Thread (seconds)
(Work=10, Heap=32MB*threads)



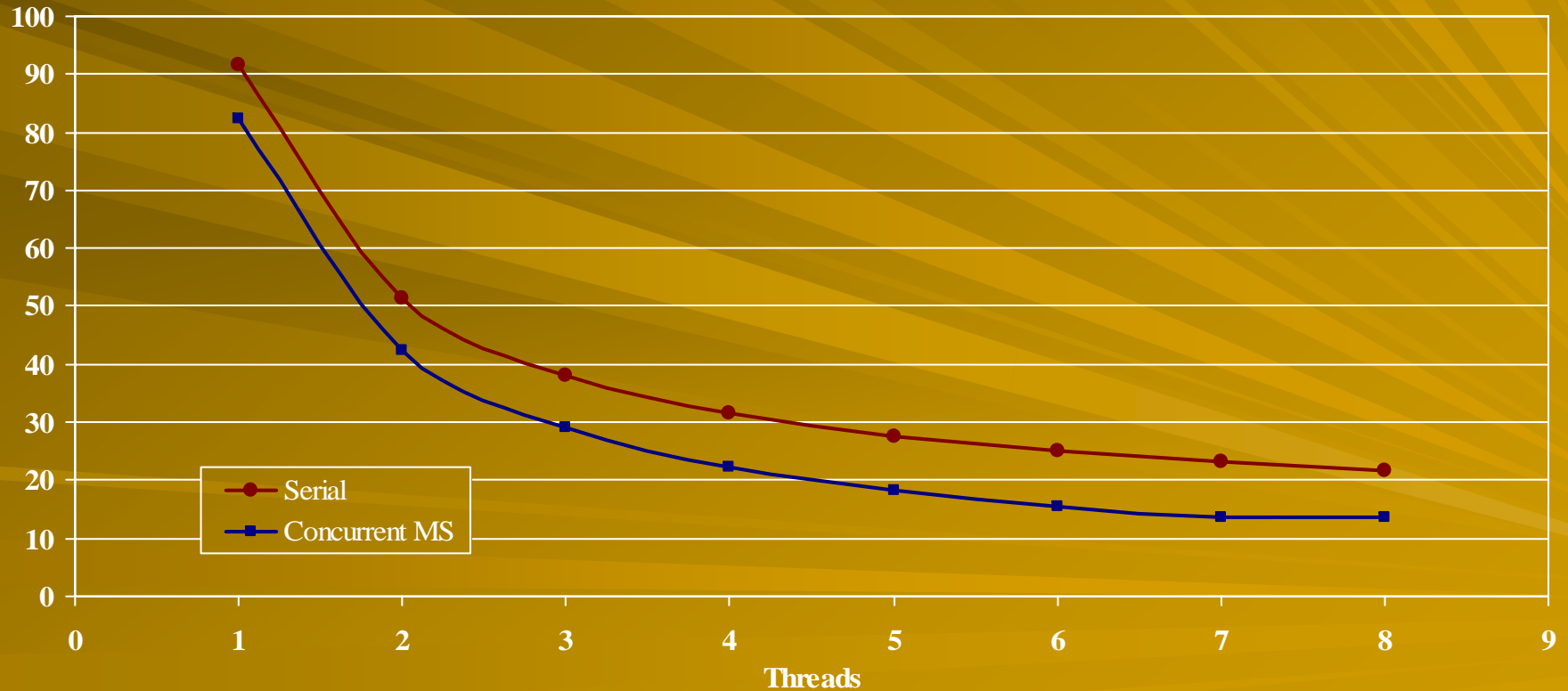
Comparing Results to Modula-3 Benchmarking

Elapsed Time Per Thread (seconds)
(Work=10, Heap=32MB*threads)



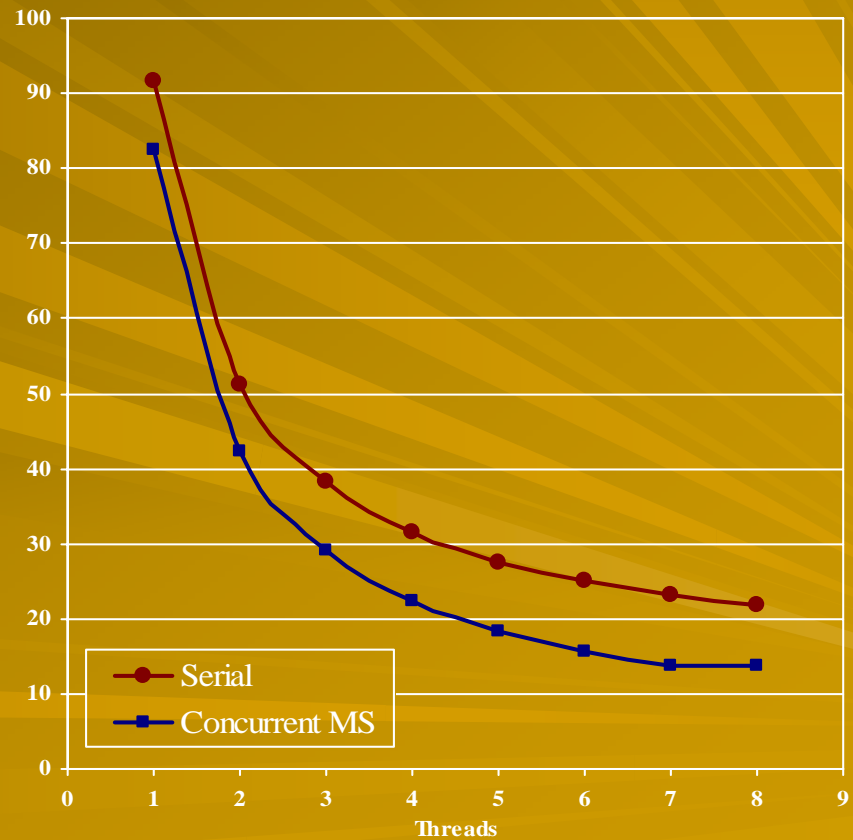
Experimental Results with Java Garbage Collectors

Elapsed Time Per Thread (seconds)
(Work=1000, Heap=32MB*threads)



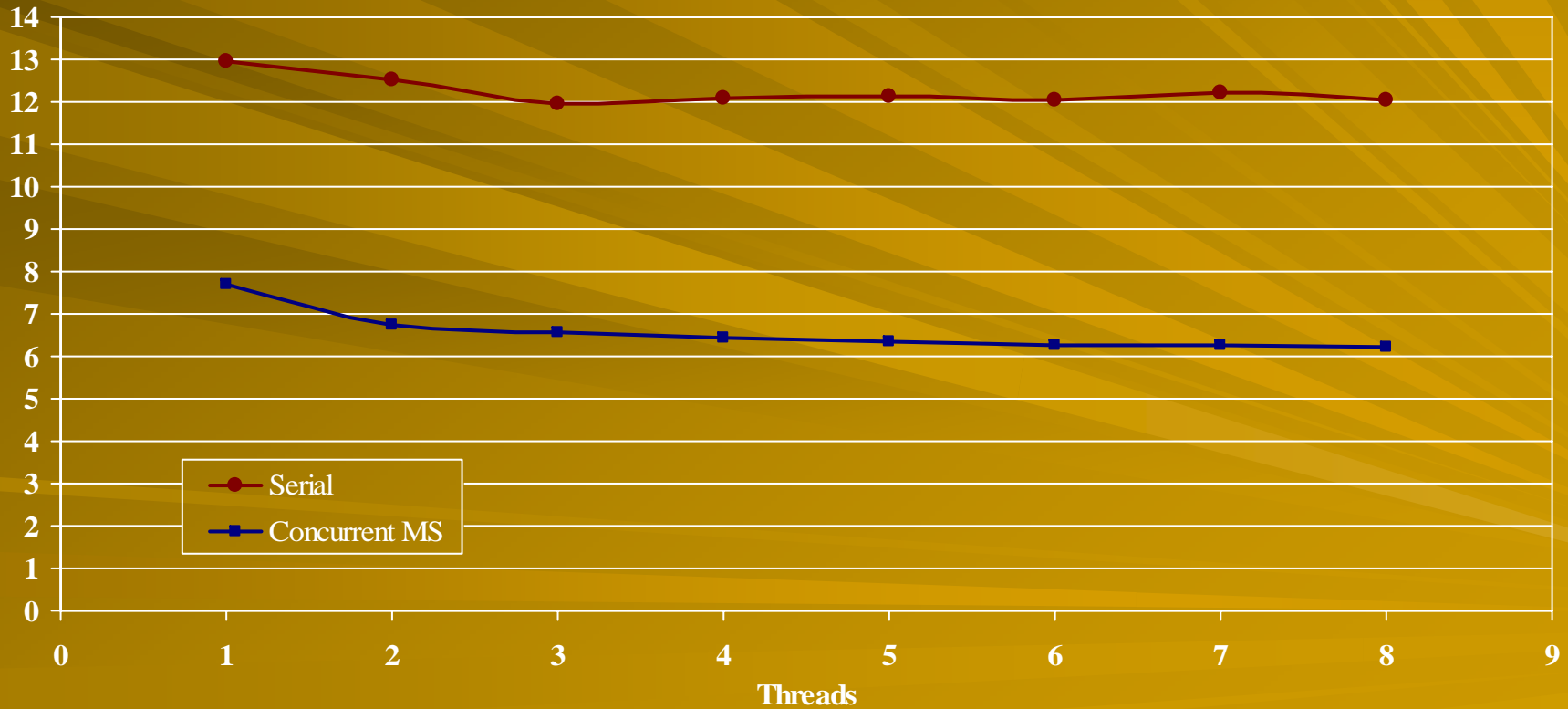
Comparing Results to Modula-3 Benchmarking

Elapsed Time Per Thread (seconds)
(Work=1000, Heap=32MB*threads)



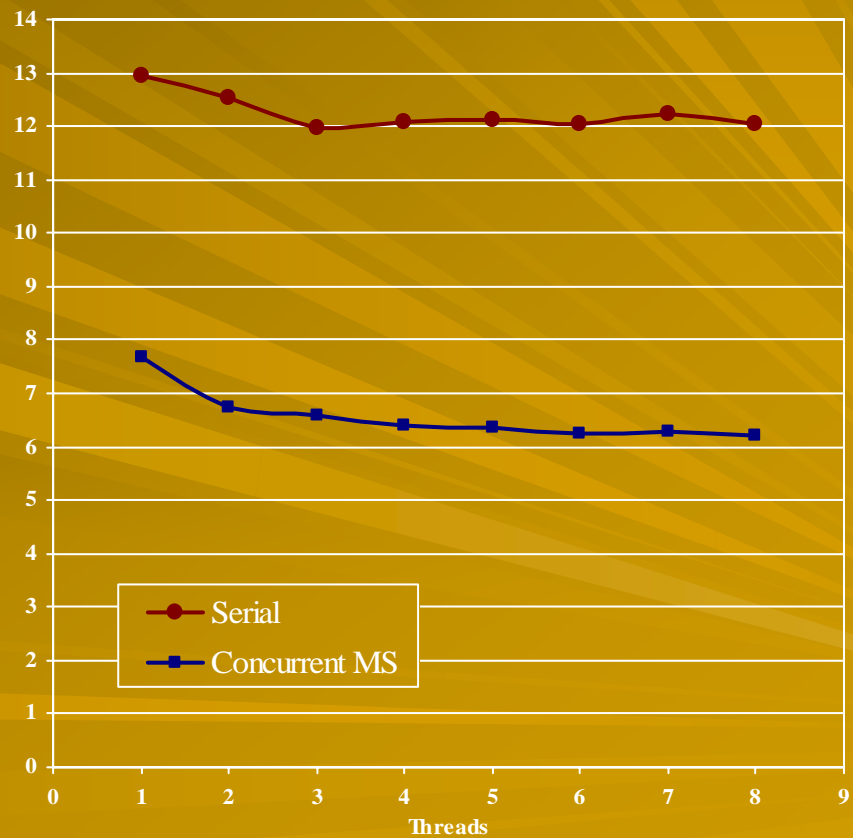
Experimental Results with Java Garbage Collectors

Elapsed Time Per Thread (seconds)
(Work=10, Heap=64MB*threads)



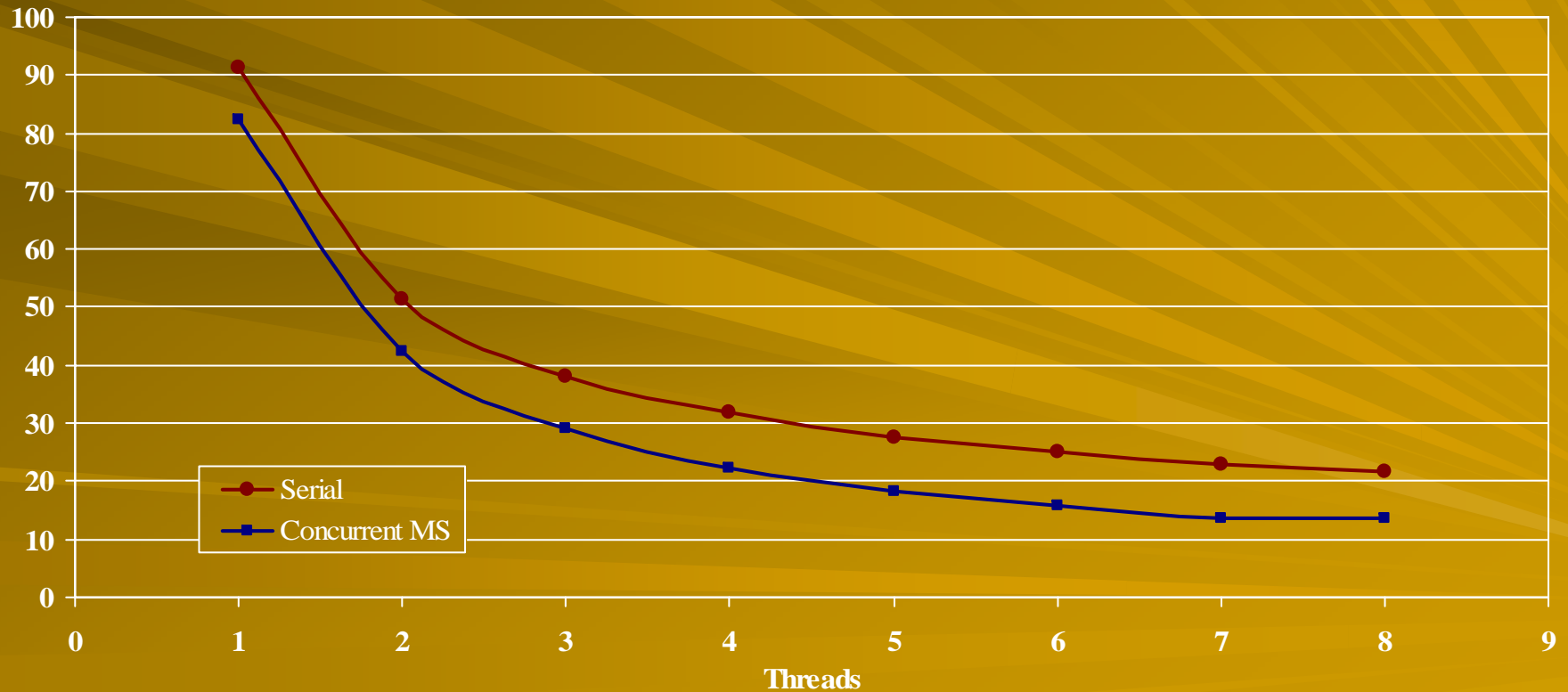
Comparing Results to Modula-3 Benchmarking

Elapsed Time Per Thread (seconds)
(Work=10, Heap=64MB*threads)



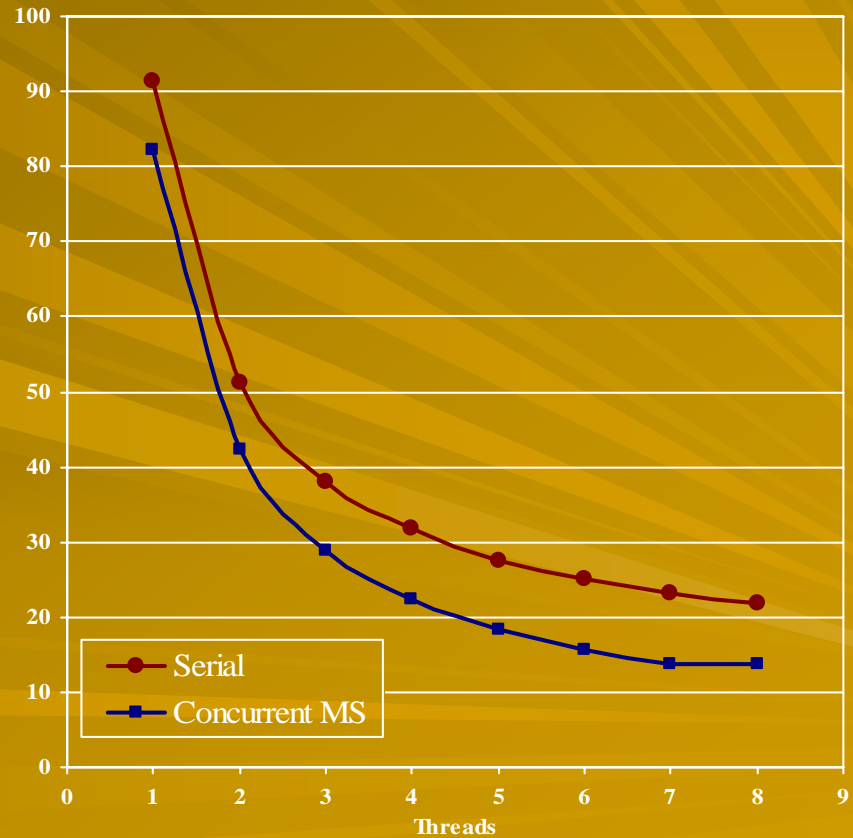
Experimental Results with Java Garbage Collectors

Elapsed Time Per Thread (seconds)
(Work=1000, Heap=64MB*threads)



Comparing Results to Modula-3 Benchmarking

Elapsed Time Per Thread (seconds)
(Work=1000, Heap=64MB*threads)



Conclusions

- The similarity of experimental data to predicted results validates that the multithreaded Java GCOld test suite is accurately ported from Modula-3
 - Viable for additional refinement and further testing

Conclusions

- The Java GC framework incurs significantly more overhead than Modula-3, as timing data for even the concurrent Java GC was at least 2x the duration of m3 garbage collectors with similar constraints
 - Additional testing necessary to determine whether a portion of this overhead rests in the Java benchmark implementation, rather than the Java GC system

Variants

○ Implicit overhead due to Java Byte Code runtime interpretation

- Necessity of “Just In Time” Java compilation and related design considerations

○ Parallelism penalty by using Perl scripting as data mining mechanism

- Could effect timing data when number of benchmark threads \geq number of processors