LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

Evaluation of JVM Garbage Collectors for CPAchecker

Tobias Maget

Supervisor: Prof. Dr. Dirk Beyer
Mentor: Dr. Philipp Wendler

Research Context – JVM Garbage Collection

Garbage collection as part of JVM's automatic memory management:

Research Context – JVM Garbage Collectors

Garbage Collector	Operating Principle	Multiple Collection Threads	Generational
Serial Garbage Collector	Stop-the-world	No	Yes
Parallel Garbage Collector	Stop-the-world	Yes	Yes
Garbage First Garbage Collector	Mostly concurrent	Yes	Yes
Z Garbage Collector	Fully concurrent	Yes	Since JDK 21
Shenandoah	Fully concurrent	Yes	No

Research Context – JVM Garbage Collectors

Each garbage collector with its own individual tuning options

→ **Impact of different configurations on the performance of CPAchecker?**

Overview

THESIS GOAL

Identify garbage collection configurations that enhance the performance of CPAchecker

2 main use cases:

- Evaluating results in a scientific environment
- Achieving results rapidly and with minimal memory usage

EVALUATION MEASURES

- CPU time
- Wall time
- Peak memory consumption

METHODOLOGY

We benchmarked

- 96 garbage collection configurations
  - ➢ 762 355 verification runs
  - ➢ 2 063 days of CPU time
- different analyses of CPAchecker

Experiment Setup

SOFTWARE

- BenchExec
- JDK 17
- Ubuntu 22.04.4
- Setup of SV-COMP24:
  - timelimit: 15 min
  - hardtimelimit: 16 min
  - memlimit: 15 GB
  - cpuCores: 4
  - -heap 10 GB
  - Identical configuration

VERIFICATION TASKS

- Subset of tasks of SVCOMP24
- Tasks of SVCOMP-24
- Common analyses:
  - K-induction
  - Value Analysis
  - Predicate Analysis

(no hardtimelimit and different configuration)

HARDWARE

BenchCloud

- Set of “tc” machines

  Intel Core i7-10700
- Set of “apollon” machines

  Intel Xeon E3-1230v5

Reproduction package available

Experimental Evaluation - Structure

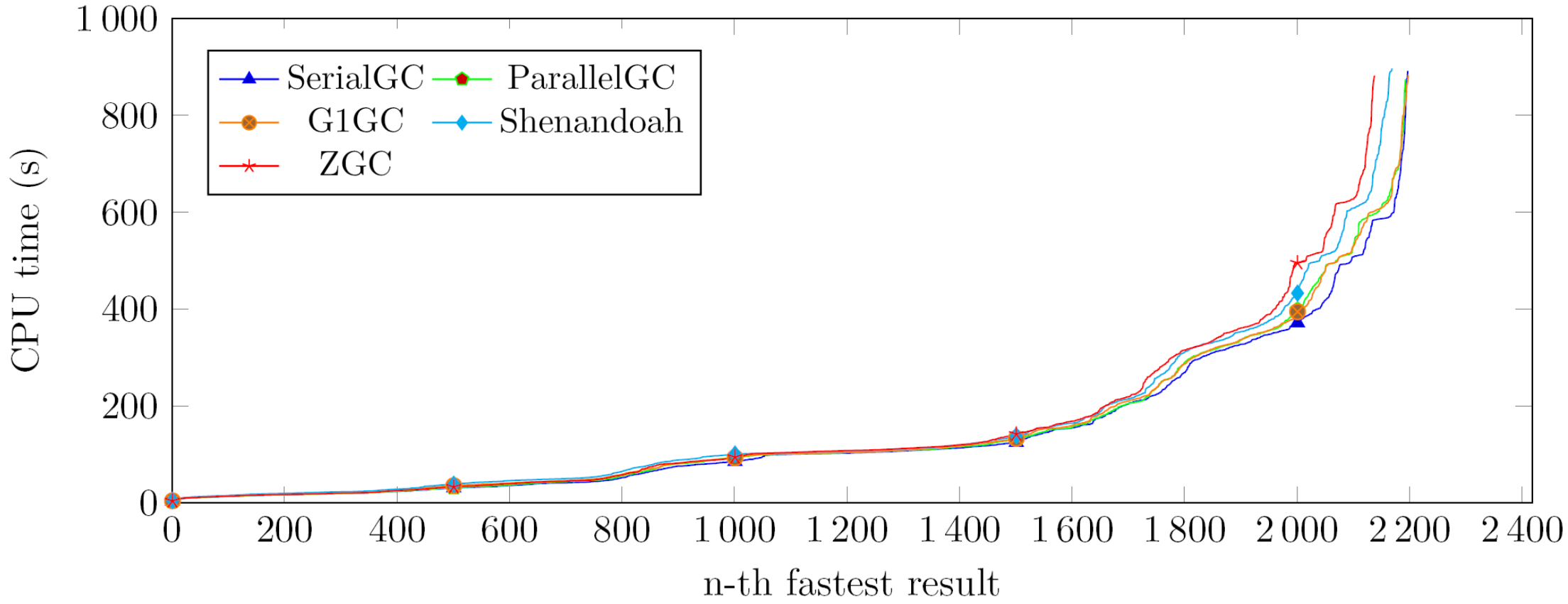
EXPERIMENTS	BENCHMARKS
Selecting garbage collectors	5 configurations
Tuning garbage collector	
• Total and initial heap size	21 configurations
• Generation size	32 configurations
• Number of parallel threads	12 configurations
• Number of concurrent threads	
• Throughput goal	2 configurations
• Maximum pause time goal	2 configurations

Experimental Evaluation - Structure

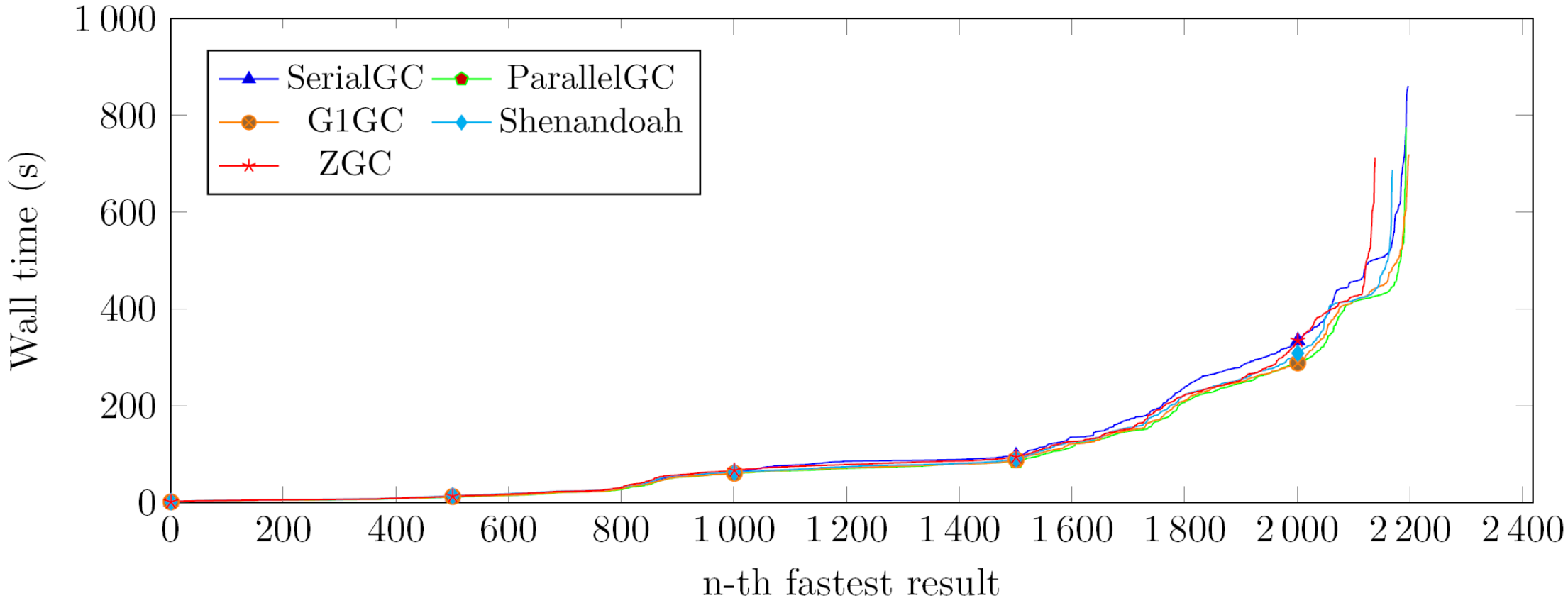
EXPERIMENTS	BENCHMARKS
Logging Information	6 configurations
Garbage First Garbage Collector	18 configurations

→ **In total: 37 garbage collection parameters**

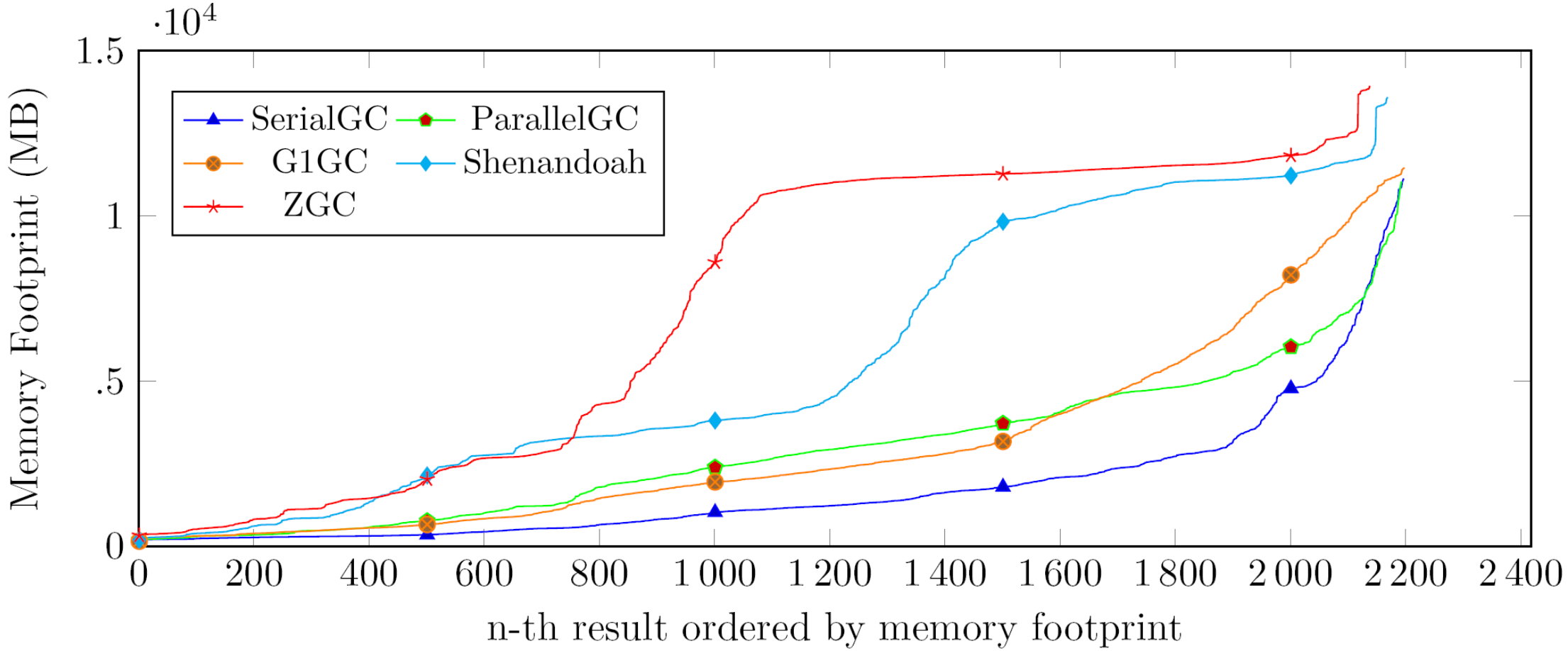
Results – Selecting Garbage Collectors



Results – Selecting Garbage Collectors



Results – Selecting Garbage Collectors



Results – Heap Sizing

- No experiments with total heap size (`-Xmx`)
- Initial heap size = total heap size (`-Xms=-Xmx`)
- Expanding the heap: `-XX:MinHeapFreeRatio`

Results – Heap Sizing

Configuration	Common Correct Tasks	Avg. CPU time (s)	Avg. Wall time (s)	Avg. Peak memory (MB)
ParallelGC	2 123			
`Default`		136	93.6	2 730
`MinHeapFreeRatio=20`		140	91.8	1 420
`MinHeapFreeRatio=40`		137	91.8	1 470
`MinHeapFreeRatio=70`		132	92.5	1 700
`MinHeapFreeRatio=80`		131	92.6	1 870
`MinHeapFreeRatio=90`		130	93.5	1 920

Results – Heap Sizing

Configuration	Common Correct Tasks	Avg. CPU time (s)	Avg. Wall time (s)	Avg. Peak memory (MB)
ParallelGC	2 123			
`Default`		136	93.6	2 730
`MinHeapFreeRatio=20`		140	91.8	1 420
`MinHeapFreeRatio=40`		137	91.8	1 470
`MinHeapFreeRatio=70`		132	92.5	1 700
`MinHeapFreeRatio=80`		131	92.6	1 870
`MinHeapFreeRatio=90`		130	93.5	1 920

Results – Statistical Analysis

Categorical Regression

- Category = configuration of a garbage collector
- Method: Ordinary Least Squares
- Every configuration (in total 7 configurations) was benchmarked 5 times

Thanks to Statistical Consulting Unit StaBLab, Department of Statistics, Ludwig-Maximilians-Universität München, Munich, Germany

Results – Statistical Analysis

Configuration	Coef	[0.025	0.975]
Avg. CPU time (R-squared = 0.998)			
Intercept [G1GC]	146.8	146.6	147.0
G1GCtuned	−0.5	−0.7	−0.2
ParallelGC			
Default	−0.7	−0.9	−0.4
`ParallelGCMinHeapRatio40`	−0.1	−0.4	0.1
`ParallelGCMinHeapRatio80`	−6.8	−7.1	−6.6
`ParallelGCMinHeapRatio80 & ParallelGCThreads=2`	−9.4	−9.6	−9.2
SerialGC	−5.4	−5.7	−5.2

Unit: (s)

Results – Statistical Analysis

Configuration	Coef	[0.025	0.975]
Avg. CPU time (R-squared = 0.998)			
Intercept [G1GC]	146.8	146.6	147.0
G1GCtuned	−0.5	−0.7	−0.2
ParallelGC			
Default	−0.7	−0.9	−0.4
`ParallelGCMinHeapRatio40`	−0.1	−0.4	0.1
`ParallelGCMinHeapRatio80`	−6.8	−7.1	−6.6
`ParallelGCMinHeapRatio80 & ParallelGCThreads=2`	−9.4	−9.6	−9.2
SerialGC	−5.4	−5.7	−5.2

Unit: (s)

Results – Statistical Analysis

Configuration	Coef	[0.025	0.975]
Avg. CPU time (R-squared = 0.998)			
Intercept [G1GC]	146.8	146.6	147.0
G1GCtuned	−0.5	−0.7	−0.2
ParallelGC			
Default	−0.7	−0.9	−0.4
`ParallelGCMinHeapRatio40`	−0.1	−0.4	0.1
`ParallelGCMinHeapRatio80`	−6.8	−7.1	−6.6
`ParallelGCMinHeapRatio80 & ParallelGCThreads=2`	−9.4	−9.6	−9.2
SerialGC	−5.4	−5.7	−5.2

Unit: (s)

Results – SV-COMP24 and other Analyses

- 5 configurations for SV-COMP24, K-Induction, Value Analysis, and Predicate Analysis
  - SerialGC
  - ParallelGC
  - ParallelGC with `MinHeapFreeRatio=80`
  - ParallelGC with `MinHeapFreeRatio=80` and `ParallelGCThreads=2`
  - G1GC

- Quantitative effects of different configurations may vary across different analyses of CPAchecker

However: Effects still recognizable!

Conclusion

- **Use case: Evaluating results in a scientific environment**

  → Recommendation: Serial Garbage Collector

- **Use case: Achieving results rapidly and with minimal memory usage**

  → Recommendation: Parallel Garbage Collector tuned with `MinHeapFreeRatio=80`

- Performance measures are connected
- GC performance was generally robust
- Quantitative effects of different configurations may vary across different analyses of CPAchecker

Literature

D. Beyer, “State of the Art in Software Verification and Witness Validation: SV-COMP 2024”, in TACAS 2024, vol. 14572, Springer, 2024, pp. 299–329. doi: 10.1007/978-3-031-57256-2_15.

D. Beyer, T. A. Henzinger, and G. Théoduloz, “Configurable Software Verification: Concretizing the Convergence of Model Checking and Program Analysis”, in CAV 2007, vol. 4590, Springer, 2007, pp. 504–518. doi: 10.1007/978-3-540-73368-3_51.

D. Beyer and M. E. Keremoglu, “CPAchecker: A Tool for Configurable Software Verification”, in CAV 2011, vol. 6806, Springer, 2011, pp. 184–190. doi: 10.1007/978-3-642-22110-1_16.

D. Beyer, S. Löwe, and P. Wendler, “Reliable benchmarking: requirements and solutions”, in Int. J. Softw. Tools Technol. Transf., vol. 21, no. 1, 2019, pp. 1–29. doi: 10.1007/S10009-017-0469-Y.

Z. Cai et al., “Distilling the Real Cost of Production Garbage Collectors”, in IEEE ISPASS 2022, IEEE, 2022, pp. 46–57. doi: 10.1109/ISPASS55109.2022.00005.

E. W. Dijkstra et al., “On-the-fly garbage collection: an exercise in cooperation”, in Lecture Notes in Computer Science, vol. 46, Springer, 1975, pp. 43–56. doi: 10.1007/3-540-07994-7_48.

C. Flood and R. Kennke, “JEP 189: Shenandoah: A Low-Pause-Time Garbage Collector”, 2014. Retrieved 2024-07-26. Available: https://openjdk.org/jeps/189.

C. H. Flood et al., “Shenandoah: An open-source concurrent compacting garbage collector for OpenJDK”, in PPPJ 2016, ACM, 2016, pp. 13:1–13:9. doi: 10.1145/2972206.2972210.

R. H. Inc., “shenandoahArguments.cpp”, Retrieved 2024-07-26. Available: https://github.com/openjdk/jdk/blob/master/src/hotspot/share/gc/shenandoah/shenandoahArguments.cpp.

R. E. Jones, A. L. Hosking, and J. E. B. Moss, *The Garbage Collection Handbook: The art of automatic memory management*, Chapman and Hall/CRC, 2011. Available: http://gchandbook.org/.

S. Karlsson, “JEP 439: Generational ZGC”, 2021. Retrieved 2024-07-26. Available: https://openjdk.org/jeps/439.

P. Lidén and S. Karlsson, “JEP 333: ZGC: A Scalable Low-Latency Garbage Collector”, 2018. Retrieved 2024-07-26. Available: https://openjdk.org/jeps/333.

Literature

J. Masamitsu, “JEP 291: Deprecate the Concurrent Mark Sweep (CMS) Garbage Collector”, 2015. Retrieved 2024-07-26. Available: https://openjdk.org/jeps/291.

S. Oaks, *Java Performance, 2nd Edition*, O’Reilly, 2020. Available: https://www.oreilly.com/library/view/java-performance-2nd/9781492056102.

Oracle, “Java HotSpot™ Virtual Machine Performance Enhancements”, Retrieved 2024-07-26. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/vm/performance-enhancements-7.html.

Oracle, “Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide, Release 11”, 2024. Retrieved 2024-07-26. Available: https://docs.oracle.com/en/java/javase/11/gctuning/hotspot-virtual-machine-garbage-collection-tuning-guide.pdf.

T. Schatzl, “JEP 363: Remove the Concurrent Mark Sweep (CMS) Garbage Collector”, 2019. Retrieved 2024-07-26. Available: https://openjdk.org/jeps/363.

A. Shipilev, “JEP 318: Epsilon: A No-Op Garbage Collector”, 2014. Retrieved 2024-07-26. Available: https://openjdk.org/jeps/318.

P. Wendler, “Towards Practical Predicate Analysis”, PhD thesis, University of Passau, 2017. Available: https://opus4.kobv.de/opus4-uni-passau/frontdoor/index/index/docId/509.

M. Williams, “Java Garbage Collection Basics”, Retrieved 2024-07-26. Available: https://www.oracle.com/webfolder/technetwork/Tutorials/obe/java/gc01/index.html.