JOHANNES KEPLER
UNIVERSITÄT LINZ

Master's Thesis

Throughput Barrier Exploration for the Garbage-First Collector

Student:	Jevgēnijs Protopopovs
Supervisors:	DI Thomas Schatzl (Oracle Labs) Prof. Hanspeter Mössenböck
Begin:	1. Oktober 2022

o.Univ.-Prof. Dr.
Hanspeter Mössenböck
Institute for System Software

T +43 732 2468 4340
F +43 732 2468 4345
hanspeter.moessenboeck@jku.at

Secretary:
Karin Gusenbauer
Ext 4342
karin.gusenbauer@jku.at

The Hotspot virtual machine (VM) is a core component of the OpenJDK project. It provides automatic dynamic memory management for its hosted applications using different garbage collection algorithms [1].

The current default collector, G1 [2], is a generational, incremental, mostly concurrent, stop-the-world evacuating garbage collector which tries to keep user-defined pause-time goals within its stop-the-world pauses.

The current implementation implements fairly complicated zone post-write-barriers [3] in order to move maintenance work for remembered sets for incremental collection out of the pause. This mechanism is called refinement in G1. This reduces pause times, but comes at a significant cost of up to 10% in throughput compared to other throughput oriented collectors like Hotspot's Parallel GC [4].

In many cases, the user-defined pause time goal is set very lenient or not at all (i.e. using the default of 200ms) that can be easily met by G1 already. This means that a significant amount of potential throughput while keeping the user goal is lost.

The goal of this thesis is to explore behavior and optimization opportunities of a more throughput-oriented alternate mode of G1 by removing the concurrent refinement, keeping all other attributes (concurrent marking, incremental collection) to (re-)gain throughput while staying within pause time goals.

The scope of this thesis is as follows:

- Implement a prototype that changes the current post-write barrier to a variant of the Parallel GC card-table based post-write barrier within the G1 framework.
  The young collection garbage collection algorithm needs to be adapted to this change as the current code exploits the availability of concurrent refinement.

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

- Evaluate throughput and latencies compared to existing G1 and Parallel GC of variants of this hybrid with at least two different throughput-oriented post write barriers on several workloads (DaCapo [5], SPECjbb [6], Renaissance [7]).
- Implement and evaluate a prototype that can switch between these barriers at stop-the-world pauses (by simply throwing away existing code) to demonstrate that both options can coexist. Experiment with potential switch-over strategies.

Since this work is done in close cooperation with Oracle and the OpenJDK project, with potential contributions to the OpenJDK project resulting from this work, this requires the student to understand and sign an Oracle Contributors Agreement [8] at the start of the thesis work.

The progress of the project should be discussed at least every two weeks with the supervisor. A time schedule and a milestone plan must be set up within the first 3 weeks. It should be continuously refined and monitored to make sure that the thesis will be completed in time.

The final version of the thesis should be submitted not later than 30.09.2023.

References:

[1] https://docs.oracle.com/en/java/javase/18/gctuning/available-collectors.html#GUID-F215A508-9E58-40B4-90A5-74E29BF3BD3C

[2] David Detlefs, Christine Flood, Steve Heller, and Tony Printezis. 2004. Garbage-first garbage collection. In Proceedings of the 4th international symposium on Memory management (ISMM '04). ACM, New York, NY, USA, 37-48. DOI=http://dx.doi.org/10.1145/1029873.1029879

[3] Xi Yang, Stephen M. Blackburn, Daniel Frampton, and Antony L. Hosking. 2012. Barriers reconsidered, friendlier still! SIGPLAN Not. 47, 11 (November 2012), 37–48. https://doi.org/10.1145/2426642.2259004

[4] https://docs.oracle.com/en/java/javase/18/gctuning/parallel-collector1.html#GUID-DCDD6E46-0406-41D1-AB49-FB96A50EB9CE

[5] Blackburn, S. M., Garner, R., Hoffman, C., Khan, A. M., McKinley, K. S., Bentzur, R., Diwan, A., Feinberg, D., Frampton, D., Guyer, S. Z., Hirzel, M., Hosking, A., Jump, M., Lee, H., Moss, J. E. B., Phansalkar, A., Stefanovic, D., VanDrunen, T., von Dincklage, D., and Wiedermann, B. The DaCapo Benchmarks: Java Benchmarking Development and Analysis, OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-Oriented Programing, Systems, Languages, and Applications, (Portland, OR, USA, October 22-26, 2006)

[6] https://www.spec.org/jbb2015/

[7] Aleksandar Prokopec, Andrea Rosà, David Leopoldseder, Gilles Duboscq, Petr Tůma, Martin Studener, Lubomír Bulej, Yudi Zheng, Alex Villazón, Doug Simon, Thomas Würthinger, and Walter Binder. 2019. Renaissance: benchmarking suite for parallel applications on the JVM. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019). Association for Computing Machinery, New York, NY, USA, 31–47. https://doi.org/10.1145/3314221.3314637

[8] https://oca.opensource.oracle.com/