Presentation Abstract

Reducing Memory Fences in a Copying Garbage Collector of Java for Weak Memory Models

Hiroshi Horii[1,a)] Michihiro Horie[1]

Presented: July 28, 2017

Garbage collectors in OpenJDK are implemented independently of memory models to maximize cross-platform portability. In the default configuration of OpenJDK, a copying garbage collector is responsible for copying live objects and reclaiming memory. Threads avoid redundant copies of objects by calling the compare-and-swap (CAS) API during concurrent access. The semantics of the CAS instruction for each CPU architecture are different. Therefore, the implementations of the CAS API need to guarantee the same semantics across different architecture. OpenJDK was first developed for SPARC and x86, which use strong memory models. To implement the CAS API for weak memory models, such as in POWER and ARM, additional memory fences are necessary. These fences guarantee the same semantic as strong memory models for weak memory models. However, each fence in the implementation incurs some overhead. In this presentation, we study memory accesses in the copying garbage collector of OpenJDK for weak memory models. We identify that the necessary memory fences for the concurrent access of values are called inside the workstealing mechanism. Therefore, the memory fences in the copying garbage collector are unnecessary. We empirically show that removing such fences provide superior performance. In particular, we achieve 20% improvement in performance by reducing the pause time for garbage collections in SPECjbb2015 on POWER8.

This is the abstract of an unrefereed presentation, and it should not preclude subsequent publication.

[1] IBM Research - Tokyo, Chuo, Tokyo 103–8510, Japan
[a)] horii@jp.ibm.com