発表概要

弱いメモリモデル環境における Java volatile 最適化手法の評価

緒方 一則[1,a)]　堀井 洋[1]　堀江 倫大[1]

2018年11月1日発表

Java の volatile 変数はノンブロッキングなアルゴリズムや ConcurrentHashMap などのライブラリを実装する際によく使用される．Java 言語仕様では，1 つの volatile 変数へのアクセスは逐次一貫性を持つと定めている．したがって，弱いメモリモデル環境では JVM 内や JIT 生成コード中に適切なメモリバリアを挿入する必要があり，不適切に処理されれば再現性の低いバグを引き起こすことになる．一方，メモリバリアはオーバヘッドが高いので，このオーバヘッドを減らすことは Java ランタイムの重要な最適化手法の 1 つである．POWER アーキテクチャなどのプロセッサでは，複数のメモリ同期命令を適切に組み合わせてメモリバリアを実装する．メモリ同期命令の組合せ方は，書き込み側とメモリ側のメモリアクセス・パターンによって変わる場合があり，それに応じてオーバヘッドも変わる．本研究では，POWER アーキテクチャ用の OpenJDK と Open J9 において volatile 変数アクセスのメモリバリア実装の最適化手法を比較し，Java プログラムの実行速度への影響を評価した．

Presentation Abstract

Evaluating Optimization Techniques for Java Volatile Variables on Weak Memory Model Platforms

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

Presented: November 1, 2018

Java volatile variables are commonly used to implement non-blocking algorithms and libraries, such as ConcurrentHashMap. Java language specification requires all accesses to a volatile variable need to be sequentially consistent. This means Java VM and JIT compiled code need to put memory fences appropriately on the weak memory model platforms, otherwise Java programs cause hard-to-reproduce intermittent problems. However, memory fences often cause large overhead, so reducing the fence overhead is one of effective optimization techniques for Java. Some processor architectures, such as the POWER architecture, provide multiple memory synchronization instructions. An appropriate combination of those instructions for implementing a memory fence depends on how threads access memory, and the overhead can vary accordingly. In this research, we investigated how memory fence for volatile variables are efficiently implemented in OpenJDK and Open J9 for the POWER platform and evaluated how the difference of implementation affects performance of Java programs.

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

[1] 日本 IBM 株式会社東京基礎研究所
IBM Research, Chuo, Tokyo 103–8510, Japan
[a)] ogatak@jp.ibm.com