不揮発性メモリを用いた Java オブジェクト永続化のオーバヘッドの調査

1210375　松本 康太郎　【 ソフトウェア検証・解析学研究室 】

1　はじめに

不揮発性メモリ（NVM）は，電源が喪失してもデータを保持することが可能な主記憶装置である．この特性を利用して，クラッシュ後にプログラムの状態を復元させることができるシステムを構築できる．ただし，メモリへのアクセスはキャッシュを介して行われるため，書き込みを永続化させるにはキャッシュラインを NVM に書き戻す必要がある．Java のようなマネージド言語では自動でメモリ管理を行う機構が備わっており，メモリについてプログラマが意識しなくて良い．そのため，オブジェクトが自動的に永続化される仕組みを持つ永続化ヒープが開発されている．

Shull らは，オブジェクトを自動的に永続化するプログラミングモデルを提案し，Java を用いてその実装を示した [1]．しかし，オブジェクトを永続化しない場合に比べて，どの程度遅くなるかは示されていない．我々は，Shull らのプログラミングモデルを実現する新しいアルゴリズムを開発している．本研究では，我々のアルゴリズムのうち NVM へのアクセスを伴う処理のオーバヘッドやその原因の調査を行う．

2　提案手法

Shull らのプログラミングモデルでは，オブジェクトを選択的に永続化する．永続化の対象であるオブジェクトへの書き込みは，プログラムと同じ順序で永続化される．

これを実現する我々のアルゴリズムでは，全てのオブジェクトは DRAM 上に存在し，Java のプログラムは DRAM 上のオブジェクトを使って実行する．永続化対象になったオブジェクトは，コピーを NVM 上に作成する．永続化されたオブジェクトへの書き込みは DRAM と NVM の両方に行い，読み込みは DRAM からのみ行う．書き込みの際は必要なキャッシュの書き戻しを自動的に行う．

3　オーバヘッド調査のための実装

本研究では，全てのオブジェクトを永続化の対象とした時の，ヒープの永続化，つまり，NVM 上のコピーの作成と NVM にコピーを持つオブジェクトへの書き込みのオーバヘッドを調べる．この実験では，DRAM 上に新しいオブジェクトが生成される際に，同時に NVM 上にコピーを作成する．この仕組みを OpenJDK 13 に実装した．JIT（Just-In-Time）コンパイラは使用せず，インタプリタの実行時間を計測した．また，NVM への書き込みを排他制御していない．

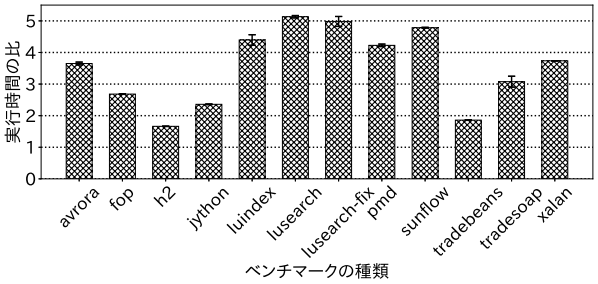

図 1　DaCapo ベンチマークの実行時間

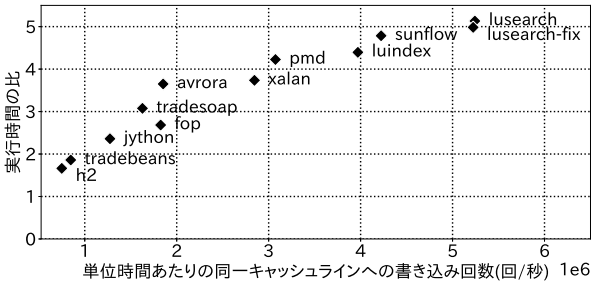

図 2　同一キャッシュラインへの書き込み頻度と実行時間

4　実験

実用的なプログラムの実行時間を調べるために，DaCapo Benchmark Suite を用いた実験を行った．変更を加えていない元の VM を 1 に正規化したときの実行時間の比を，図 1 に示す．NVM にコピーを作成して必要なキャッシュの書き戻しを自動的に行った場合，最大で 410%のオーバヘッドが観測されたが，プログラムによりオーバヘッドにばらつきが見られた．その原因を調べるために，単位時間あたりの連続で同一キャッシュラインに書き込みを行った回数を測った（図 2）．単位時間あたりの同一キャッシュラインへの書き込み回数が増加すると，実行時間の比も増加していることが分かった．

5　まとめ

本研究では，オブジェクトの永続化に必要な処理のうち NVM へのアクセスを伴う部分を OpenJDK に実装し，全オブジェクトの永続化にかかるオーバヘッドを調査した．実験の結果，DaCapo ベンチマークで観測されたオーバヘッドは最大で 410%であり，同一キャッシュラインへの連続アクセスが多いプログラムは遅くなる傾向があった．

参考文献

[1] T. Shull, J. Huang, and J. Torrellas. AutoPersist: An Easy-to-Use Java NVM Framework Based on Reachability. In *PLDI'19*, pages 316–332, 2019.