不揮発性メモリ向けオブジェクト永続化アルゴリズムのモデル検査器 Spin を用いた検証

1245115　飯干 寛幸　【 ソフトウェア検証・解析学研究室 】

Verification of an Algorithm for Persistent Object in Non-Volatile Memory Using Spin Model Checker

1245115　Hiroyuki IIBOSHI　【 Software Verification and Analysis Lab. 】

1　はじめに

近年，DRAM と同じようにバイト単位のアクセスが可能で，かつ計算機の電源が失われてもデータが残る不揮発性メモリが注目されている．不揮発性メモリを使ったシステムではメモリへ書き込むだけでデータを**永続化**できる．ただし，不揮発性メモリはキャッシュを介してアクセスすることに注意が必要である．キャッシュが書き戻されるタイミングはプログラマには分からない．その他め，データの整合性を保つためには，キャッシュラインを書き戻す命令を使ってプログラマが永続化順序を制御する必要がある．

プログラマの負担を軽減するために，Java のようなマネージド言語でランタイムシステムが代わりに永続化順序を管理するシステムが研究されている [1]．そのようなシステムでは，全てのデータを永続化するのではなく，オブジェクト単位で選択的にデータを永続化する．

オブジェクト永続化アルゴリズムは，ランタイムシステムが Java のオブジェクトを永続化するためのアルゴリズムである．一般に，DRAM 上のオブジェクトを不揮発性メモリ上へコピーすることでオブジェクトを永続化する．オブジェクトの永続化中にユーザプログラムがオブジェクトの値を書き換える可能性があるため，オブジェクト永続化アルゴリズムには適切な排他制御が必要であり，作るのが難しい．そこで，形式的手法を使った検証によってオブジェクト永続化アルゴリズムの正しさを確かめることが考えられる．モデル検査器 Spin[2] は，プログラムを抽象化したモデルに基づいて，特定の性質がモデルのあらゆる実行経路で成り立っているか網羅的に調べるツールであり，並行プログラムの検査にも使われる．

本研究では，オブジェクト永続化アルゴリズムの一つである**複製による永続化** [1] のアルゴリズムの正当性を，Spin を使った検査で確かめる．

2　複製による永続化

複製による永続化は，オブジェクトの永続化後に DRAM 上のオブジェクトと不揮発性メモリ上のオブジェクトの両方を保持する．DRAM 上のオブジェクトを揮発性コピー，不揮発性メモリ上のオブジェクトを永続化コピーと呼ぶ．ユーザプログラムは揮発性コピーから値を読み出し，揮発性コピーと永続化コピーの両方に値を書き込む．計算機の電源が失われたときには永続化コピーだけが残るので，DRAM に書き込まれた値を他のスレッドが読み出す前に不揮発性メモリに書きこまなければ不整合な状態になる可能性がある．

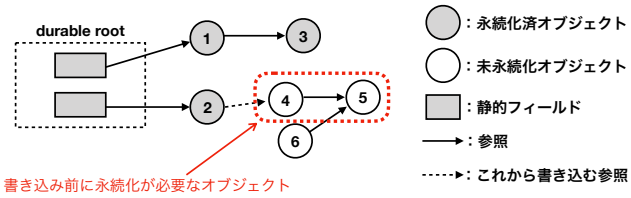

図 1　オブジェクトの永続化が必要になる例

複製による永続化では，static 変数の中から永続化する変数の集合をプログラマが指定する．この変数の集合を **durable root** という．そして，durable root から参照を辿って到達可能なオブジェクトをランタイムシステムが永続化する．永続化コピーを持つオブジェクトが永続化コピーを持たないオブジェクトを参照すると，計算機の電源が失われたときに参照先がないオブジェクトができてしまう．したがって，durable root から到達可能なオブジェクトはすべて永続化コピーを持たなければならない．例えば，図 1 では durable root から参照を辿って到達できるオブジェクト 1，2，3 はすでに永続化されている．ユーザプログラムがオブジェクト 2 から 4 への参照を書き込もうとした場合，オブジェクト 4，5 は durable root から到達可能になるので，参照の書き込みよりも先に永続化する必要がある．オブジェクト 6 は参照の書き込みが終わった後でも durable root から到達不可能なので永続化する必要がない．

3　検査方法

本研究では，複製による永続化のアルゴリズムが次の 2 つの性質を満たしているか Spin を使って調べる．

- 書き込み途中のオブジェクトを除いて，揮発性コピーの値と永続化コピーの値が一致する

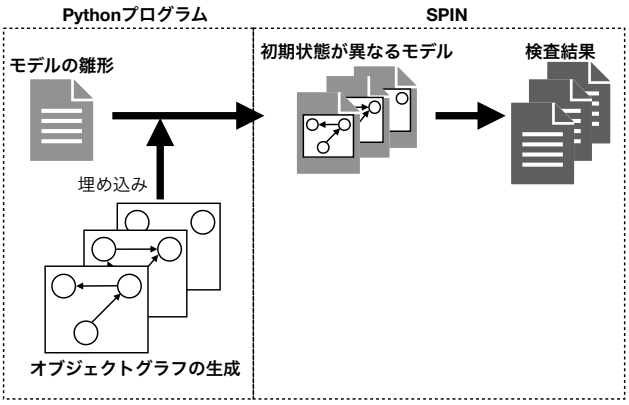

図 2　モデル検査の流れ

- durable root から参照を辿って到達可能なオブジェクトはすべて永続化コピーを持つ

これらの性質は，オブジェクトの参照が書き換えられたときに満たされなくなる可能性がある．そこで，複製による永続化アルゴリズムのうち，参照の書き換えと書き換えに伴うオブジェクトの永続化のモデルを作成した．特定のメモリの状態から実行を始めて，複数スレッドでアルゴリズムに従って任意のオブジェクトへ参照を書き込む．このとき，DRAM と不揮発性メモリの値が常に上記の性質を満たした状態になっていれば，アルゴリズムは正しい．

本研究で作成するモデルは，オブジェクトに参照を書き込む複数のワーカースレッドと上記の性質が満たされているか調べるオブザーバスレッドからなる．ワーカスレッドはアルゴリズムに従って任意のオブジェクトに参照を書き込む．オブザーバスレッドは任意の時点から実行を始めて，durable root から参照を辿って到達できるオブジェクトすべてが 2 つの性質を満たしているか調べる．

オブジェクトを頂点，オブジェクト間の参照関係を辺とみなしたグラフをオブジェクトグラフと呼ぶ．検査開始時点におけるオブジェクトグラフによって，調べることができる実行経路が変わる．ここで，現実的な検査時間やメモリ使用量で調べられる実行経路の数は有限なので，オブジェクトの数やユーザスレッドの数といったパラメータの制限がモデル検査には存在する．すべてのオブジェクトグラフのパターンを 1 回のモデル検査で調べようとすると，一度にメモリを大量に消費するため，小さなパラメータでしか検査できない．できる限りパラメータを大きくとるために，調べる実行経路を検査開始時点のオブジェクトグラフによって分割する．モデル検査の流れを図 2 に示す．初期状態におけるオブジェクトグラフだけが決まっていないようなモデルの雛形を作成し，Python プログラムを使って別途作成したオブジェクトグラフを後から埋め込んでモデルを作成する．

アルゴリズムの特徴を利用して，調べる実行経路の数

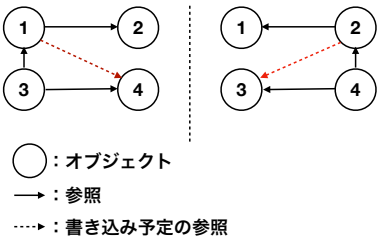

図 3　同型なグラフの例

をさらに減らすことができる．一部の実行経路は，別の実行経路と検査結果が一致することが検査する前にわかる．検査結果が一致する実行経路については，そのうち一つだけを調べればよい．例えば，複製による永続化のアルゴリズムは，オブジェクトを ID によって区別しない．そのため，初期状態とするオブジェクトグラフが異なっても，同型なグラフであれば検査結果は同じである．そこで，オブジェクトグラフを生成するとき，すでに生成したグラフと同型なグラフは生成せず，モデルの数を削減する．図 3 の左のグラフと右のグラフは互いに同型なグラフであり，オブジェクト 1，3 とオブジェクト 2，4 がそれぞれ対応している．右のグラフにおいてオブジェクト 2 から 3 への参照を書き込む場合の検査結果について考える．ワーカスレッドは任意のオブジェクトに書き込むことができるので，左のグラフの検査においてオブジェクト 1 から 4 への参照を書き込む場合の実行経路は必ず検査済みである．左のグラフの検査結果におけるオブジェクトの ID を右のグラフに対応するよう置き換えれば，右のグラフの検査結果が得られる．

4　検査結果

本研究で作成したモデルを使って，パラメータをオブジェクト数が 5 つ，スレッド数が 2 として検査した．その結果，複製による永続化のアルゴリズムにバグは見つからなかった．

5　おわりに

複製による永続化は，durable root から到達可能なオブジェクトがすべて永続化コピーを持ち，揮発性コピーの値と永続化コピーの値が一致するという性質を持つ．本研究では，本研究で検査に用いたパラメータの範囲では，複製による永続化が持つ 2 つの性質が満たされていることをモデル検査によって確かめた．

参考文献

[1] 鵜川 始陽 ら．オブジェクトの到達可能性による永続化をリードバリアを使わずに実現するアルゴリズムとその予備評価. In **日本ソフトウェア科学会第** *38* **回大会講演論文集**, Sep 2021.

[2] G.J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.