様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　５ 年　 ６ 月　 １ 日現在

機関番号：１２６０１
研究種目：基盤研究(C)（一般）
研究期間：2019～2022
課題番号：１９Ｋ１１９０４

研究課題名（和文）マネージド言語のための不揮発性メモリを用いた永続化ヒープ

研究課題名（英文）Persistent Managed Heap on Non-Volatile Memory

研究代表者
鵜川　始陽（Ugawa, Tomoharu）

東京大学・大学院情報理工学系研究科・准教授

研究者番号：５０４２３０１７

交付決定額（研究期間全体）：（直接経費）　　3,400,000円

研究成果の概要（和文）：本研究では不揮発性メモリをプログラミング言語で高度に抽象化する研究を行った．マネージド言語でプログラマが不揮発性メモリを明示適に扱わずにプログラミングできるようにするために，直行永続化というモデルを使った．本研究では，オブジェクトの複製を不揮発性メモリ上に作ることによりオブジェクトを永続化する方法を提案し，広く使われているJava仮想機械であるOpenJDKに実装した．この手法のオーバヘッドは，同時期に他のグループによって行われた同様の研究と同程度だった．この他，不揮発性メモリ向けのGCや，C++向けの永続化トランザクションライブラリやチェックポインティングライブラリを開発した．

研究成果の学術的意義や社会的意義
本課題では不揮発性メモリという新しいメモリデバイスを使って，ソフトウェアやハードウェアの障害に備えてソフトウェアが処理しているデータを保存する技術を研究した．不揮発性メモリを明示的に扱うプログラムを記述しなくても，常に必要なデータが自動的に保存され，かつ，保存している間もソフトウェアが実行を続けることができる仕組みを提案し，試作した．この技術はデータベースのようなシステムの実装で利用されることを想定しており，データベースの高速化や，ソフトウェアに合わせた小型のデータベースの構築を容易にすることにつながる．

研究成果の概要（英文）：We studied high level abstractions of non-volatile memory in programming languages. We used the orthogonal persistency model to allow the programmers to write programs without handling non-volatile memory explicitly. We proposed to create replicas of objects in non-volatile memory to make the objects persistent and implemented it in OpenJDK, which is a widely used Java virtual machine. Our system had a similar overhead to the other work done by another group separately. We also developed GC for non-volatile memory and C++ libraries to support persistent transactions and checkpointing.

研究分野：プログラミング言語、システムソフトウェア

キーワード：不揮発性メモリ　マネージドランタイム　並行処理　メモリ管理　省電力計算

※科研費による研究は、研究者の自覚と責任において実施するものです。そのため、研究の実施や研究成果の公表等については、国の要請等に基づくものではなく、その研究成果に関する見解や責任は、研究者個人に帰属します。

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

不揮発性メモリは電源を切っても内容が保持されるという特性を持つ記憶装置で，研究開始当初は次世代のメモリとして注目されていた．研究を開始した 2019 年にインテル社から製品の販売が始まり，本格的に普及しはじめるという期待があった．学術的な面では，2010 年ごろから不揮発性メモリを使ったアルゴリズムやプログラミング言語の意味論，プログラミング言語処理系の研究が少しずつ行われるようになってきており，研究を始めた 2019 年頃にはある程度の基礎的な知見が蓄積されていた．

２．研究の目的

しかし，それまでに研究開発されていた多くのシステムでは，不揮発性メモリが十分に抽象化されておらず，抽象度の高い記述ができるプログラミング言語でも不揮発性メモリを意識してプログラムを記述する必要があった．特に，不揮発性メモリ上のデータの完全性(integrity)はプログラマの責任となっていた．不揮発性メモリはプロセスの仮想アドレス空間にメモリマップして利用し，他のアドレスにマップされている DRAM と併用する．このとき，不揮発性メモリがマップされたアドレスに書き込まれたデータだけが永続化され，再起動後に復旧に利用できる．しかし，研究開始当時に発表されていたシステムでは，不揮発性メモリの領域に保存したオブジェクトが参照するオブジェクトが全て不揮発性メモリ上にあることは自動的には保証されなかった．これによって，プログラミングが煩雑になり誤りやすくなるだけでなく，内部でオブジェクトを作ったり，オブジェクト間の参照を付け替えたりする既存のライブラリが利用できなくなる．ライブラリ関数の内部で不揮発性メモリ上のオブジェクトから DRAM 上のオブジェクトへの参照を作る可能性があるからである．
この問題の他にも，不揮発性メモリ上の不要になった領域を回収して再利用するガーベージコレクション(GC)は一般的な DRAM 用のアルゴリズムをそのまま利用しており，DRAM に比べアクセスが遅い不揮発性メモリに適したものではなかった．さらに，システムクラッシュの後の復旧時に，膨大なデータを不揮発性メモリから DRAM にコピーする必要があり，復旧に時間がかかるという問題もあった．復旧の間はプログラムを実行することができず，そのため，システムクラッシュ時のダウンタイムが長くなる．
そこで，本研究では，以下の 3 点を目的とした．
(1) 永続化されるデータの完全性が自動的に実現され，制約なく既存ライブラリが使えるような不揮発性メモリのプログラミング言語での抽象化
(2) 不揮発性メモリに適した，不揮発性メモリへのアクセスが少ない GC
(3) 復旧を完了する前にプログラムの実行を開始できる，インクリメンタルな復旧処理
なお，(3)は本研究の立案から研究着手までの間に他グループによって達成されたため，本研究では(1)，(2)に取り組んだ．

３．研究の方法

まず，(1)不揮発性メモリを利用するソフトウェアの研究で主流だった，プログラムが明示的に不揮発性メモリを利用するのをサポートするフレームワークを C++ライブラリとして作る方法を研究し，不揮発性メモリを利用するソフトウェアシステムに対する知見を深めた．その後，(2)永続化データの完全性が自動的に実現されるプログラミングモデルである直行永続化(orthogonal persistency)を，不揮発性メモリを使って実現する研究を行なった．この研究では，考案したアルゴリズムを広く利用されている Java 仮想機械(Java VM)である OpenJDK に実装して評価した．さらに，(3)その Java VM を使って不揮発性メモリの GC の研究を行なった．

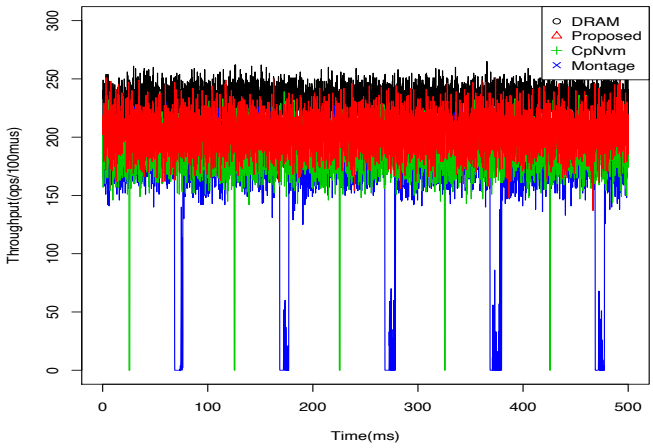

図 1 チェックポインティングを行う memcached の 100μs 毎のスループット．YCSB ベンチマークのワークロード A を使用．

４．研究成果

(1) プログラムが明示的に不揮発性メモリを利用するのをサポートするフレームワーク

一連の不可分なメモリアクセスをデータベースのトランザクションのように扱うトランザクショナルメモリの手法を不揮発性メモリに適用する C++ライブラリである NV-HTM を調査し，不揮発性メモリの利用に関する知見を深めた．その中で，NV-HTM の非効率な点を発見し，改良した．トランザクショナルメモリは，複数のスレッドによる競合アクセスの可能性があっても楽観的に処理を行い，実際に競合があったことを検出すると処理を巻き戻す実行モデルである．それをサポートする CPU 命令も作られ，効率よく不可分操作を実行できる．不揮発性メモリ上にデータ構造を配置する場合，その更新処理の途中でクラッシュするとデータ構造の一部だけが更新され，整合性が取れていない状態のデータが不揮発性メモリに残ることになる．このデータを利用しても実行状態を復旧することはできない．そこで，NV-HTM では不揮発性メモリ上のデータ構造の更新処理をソフトウェアトランザクショナルメモリのトランザクションと融合し，他のスレッドとの競合だけでなく，実行途中でクラッシュした場合でも更新処理を行う前に巻き戻せるようにしている．これを実現するために，トランザクション中のメモリへの書き込みを直接不揮発性メモリ上のデータ構造に書き込むのではなく，一旦 redo ログと呼ばれるログ構造に蓄え，トランザクションの完了時に不揮発性メモリ上のデータ構造に反映させる．Redo ログをデータ構造に反映させる処理は不可分ではないが，その途中でクラッシュしても復旧時に redo ログの反映をやり直すことで，トランザクションを完了させることができる．
この方式では，ユーザプログラムがデータ構造を更新しするとき，不揮発性メモリ上の redo ログに書き込む必要がある．不揮発性メモリへの書き込みは DRAM に比べて遅いことが知られており，それによりプログラムの実行が遅くなる．NV-HTM は，DRAM 上に一次 redo ログを作り，それを専用のスレッドが不揮発性メモリ上の redo ログにコピーする．しかし，このようにしても不揮発性メモリに書き込むデータの総量は減らない．本研究では，不揮発性メモリ上 redo ログを圧縮することで不揮発性メモリへの書き込みを減らした．
さらに，個々のトランザクションの完了時に不揮発性メモリに書き込むのではなく，100ms 程度の頻度でチェックポイントを設け，それまでに蓄積された更新をまとめて不揮発性メモリに書き込むチェックポインティングへの応用も研究した．この研究では，チェックポイントで区切られた世代ごとに redo ログを設け，不揮発性メモリへの書き込みとプログラムの実行を並行して行えるようにし，プログラムを停止することなくチェックポインティングを行うことを可能にした．その結果，既存のチェックポインティング手法では，チェックポインティングのたびにシステム全体を停止する必要があったものを，システムを停止することなくチェックポインティングを行えるようになった．既存のチェックポインティング手法を搭載したシステム CpNvm と Montage との比較を図 1 に示す．図の縦軸は 100μs 毎のスループットを表しており，提案手法(proposed)は常に高いスループットを示しているのに対し，既存手法は定期的にスループットが 0 になっている．この成果は日本ソフトウェア科学会の大会で発表した．

(2) 不揮発性メモリを使った直交永続化の実現

直行永続化は，永続化ルートと呼ばれるプログラマが指定した大域変数と，そこからポインタを辿って到達することのできる全てのオブジェクトを自動的に永続化し，クラッシュ後に復旧できるようにするモデルである．従来はオブジェクトの永続化にファイルシステムやデータベー

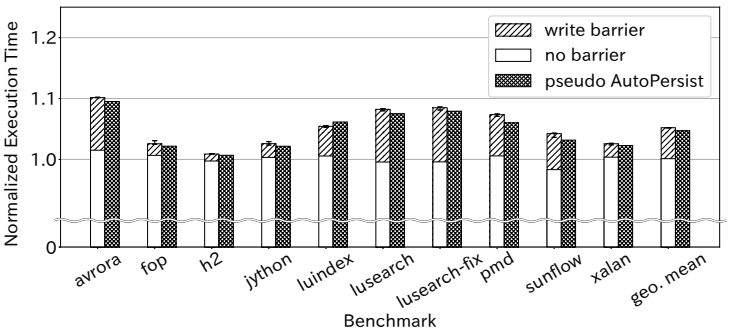

図 3 CCJava と AutoPersist でどのオブジェクトも永続化しないプログラムを実行したときの実行時間．変更していない Java VM の実行時間で正規化してある．

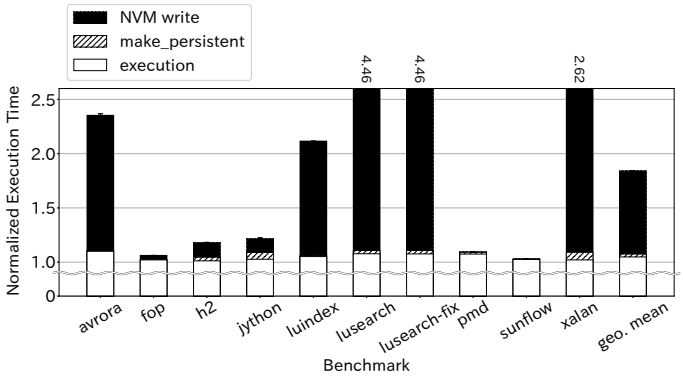

図 2 CCJava で全ての static 変数を永続化ルートに指定した時の実行時間．変更していない Java VM の実行時間で正規化してある．

スを使っており, オブジェクトを直列化して保存しなければならず, オーバヘッドが大きかった．本研究では, データストレージに不揮発性メモリを使い, データ構造を直接保存するようにした．それでも，不揮発性メモリのアクセスは DRAM に比べて遅いため，我々が GC の研究で培ったオブジェクトの複製を作り，本体と複製を同期された状態に保つ技術を応用して，DRAM にオブジェクトの本体を置き，その複製を不揮発性メモリに作る方法をとった．
直行永続化のモデルでは, 全てのオブジェクトは永続化されていない状態で作られる．それらは，永続化ルートから到達可能になった瞬間に永続化される．本研究のとったオブジェクトを複製する方式では，オブジェクトに書き込むとき，オブジェクトが永続化されていれば本体と複製の両方を更新しなければならない．オブジェクトは非同期に実行される別のスレッドの動作によって，あるスレッドがそのオブジェクトに書き込んでいる途中で永続化ルートから到達可能になる可能性がある．その場合，書き込みの途中で複製されることになる．そのような場合でも，正しく複製に書き込みが行われるようなオブジェクトへの書き込みのプロトコルを開発した．このプロトコルは，日本ソフトウェア科学会の大会で発表し，高橋奨励賞を受賞した．
開発したオブジェクトを複製する手法を，広く使われている Java VM である Open JDK に実装した CCJava を開発して評価を行なった．我々の研究の最中に，別の研究グループからも不揮発性メモリを使って直行永続化を実現するシステム AutoPersist の提案がなされた．AutoPersist はオブジェクトを複製するのではなく，不揮発性メモリに移動させるものだった．我々は AutoPersist の一部を OpenJDK に実装し，それと比較した．実装した AutoPersist (Pesudo AutoPersist)は，オブジェクトへの書き込み時に行われる処理のオーバヘッドだけを再現したものであり，オブジェクトは永続化後も DRAM に置かれている．そのため，実際の AutoPersist よりは高速に動作していると考えられる．図 2 は永続化ルートを指定せず, その結果どのオブジェクトも永続化されない実行の実行時間を示している．この結果が示すように，一部だけ実装した AutoPersist とほぼ同等だった．また，図 3 は全ての static 変数を永続化ルートに指定した場合の実行時間である．プログラムによっては永続化しない場合の数倍の実行時間がかかっており，その内訳から，永続化されたオブジェクトへの書き込みによって引き起こされる不揮発性メモリへの書き込みの時間が課題だと分かった．この成果は，メモリ管理に関する著名な国際会議 ISMM に採択された．

(3) 不揮発性メモリの GC

不揮発性メモリのアクセスは遅く，特に書き込みは遅いことが知られている．一方で，典型的なマークスイープ GC は，ルート集合からポインタを辿ることで到達可能なオブジェクトの集合を同定し，その補集合を回収する．この時，到達したオブジェクトに印をつける処理があり，これはオブジェクトへの書き込みを伴う．さらに，オブジェクトが持つポインタを得るために，オブジェクトの内容を読み出す．これらの処理は，オブジェクトが不揮発性メモリに配置されていると，時間がかかり効率が悪い．
本研究では CCJava のための，効率の良い不揮発性メモリ GC を開発した．この GC は，CCJava ではオブジェクトを複製しているので，DRAM にも対応するオブジェクトが存在しているはずで，オブジェクトが回収可能になるタイミングも DRAM 上の対応するオブジェクトと一致するはずという観測に基づく．提案した GC は，OpenJDK が備える DRAM 上の通常の GC を拡張して行われ，通常の GC が DRAM 上のオブジェクトを訪問した時に，それに対応する不揮発性メモリ上の複製にも到達した印を付ける．さらに，不揮発性メモリ上の複製の到達した印を付ける領域は，DRAM 上にビットマップとして用意する．これにより，不揮発性メモリにアクセスすることなく不揮発性メモリの GC を行うことができる．この成果はプログラミング言語に関する国内のワークショップで発表した．

(4) NUMA 計算機での不揮発性メモリ上へのオブジェクトの配置戦略

NUMA 計算機では，メモリが複数の計算ノードに分散しており，遠隔の計算ノードが持つメモリへのアクセスには時間がかかる．そのため，スレッドがアクセスするデータは，それを実行している計算ノードのメモリに格納されていることが望ましい．通常のメモリについては OS が仮想アドレスを物理アドレスにマップする際に適切なヒューリスティクスに基づいて計算ノードのメモリを割り当てるが，不揮発性メモリは計算ノード毎に異なる仮想アドレスにマップする必要がある．そのため，プロセスが明示的にどの計算ノードにデータを割り当てるかを決めることになる．
本研究では，当初の研究計画に加えて，このような場合にどの計算ノードにメモリを割り当てるのが適切かを CCJava の場合について調査した．CCJava では，オブジェクトは当初は DRAM 上にのみ作られ，永続化ルートから到達可能になった時に，不揮発性メモリに複製が作られる．この複製の領域を(1)複製元の DRAM 上のオブジェクトが存在する DRAM と同じ計算ノードの不揮発性メモリに割り当てる場合と(2)複製処理を行うスレッドを実行している計算ノードの不揮発性メモリに割り当てる場合を比較した．その結果，(2)の方が高速であることが分かった．(1)の方式では，DRAM 上のオブジェクトは OS がヒューリスティクスに基づいて決定した計算ノードに配置されているため，その結果を利用することができ高速と予想していたが，実験結果はそれに反するものだった．

５．主な発表論文等

〔雑誌論文〕　計1件（うち査読付論文　1件／うち国際共著　0件／うちオープンアクセス　0件）

１．著者名 松本 康太郎，高田 喜朗，鵜川 始陽	４．巻 38
２．論文標題 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	５．発行年 2021年
３．雑誌名 コンピュータソフトウェア	６．最初と最後の頁 2_14 - 2_19
掲載論文のDOI（デジタルオブジェクト識別子） 10.11309/jssst.38.2_14	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

〔学会発表〕　計9件（うち招待講演　0件／うち国際学会　1件）

１．発表者名 中田 昌輝，佐藤 重幸，鵜川 始陽
２．発表標題 不揮発性メモリを活用したjemallocのチェックポインティングと復元
３．学会等名 第25回プログラミングおよびプログラミング言語ワークショップ
４．発表年 2023年

１．発表者名 松本 康太郎，長安 尚之，鵜川 始陽，高田 喜朗，岩崎 英哉
２．発表標題 不揮発性メモリを用いた複製に基づく永続化のためのリカバリー機構
３．学会等名 第64回 プログラミング・シンポジウム
４．発表年 2023年

１．発表者名 中田 昌輝，鵜川 始陽，佐藤 重幸
２．発表標題 プログラムを停止させない定期的な不揮発性メモリへのチェックポインティング
３．学会等名 日本ソフトウェア科学会第39回大会
４．発表年 2022年

１．発表者名
Kotaro Matsumoto, Tomoharu Ugawa, Hideya Iwasaki

２．発表標題
Replication-based Object Persistence by Reachability

３．学会等名
2022 ACM SIGPLAN International Symposium on Memory Managemet（国際学会）

４．発表年
2022年

１．発表者名
長安尚之，鵜川始陽，松本康太郎，岩崎英哉

２．発表標題
複製に基づく永続化を行うシステムにおける不揮発性メモリ管理手法

３．学会等名
第24回プログラミングおよびプログラミング言語ワークショップ

４．発表年
2022年

１．発表者名
松本康太郎，高田喜朗，鵜川始陽

２．発表標題
不揮発性メモリにオブジェクトの複製を作るJava VMにおけるvolatileフィールドの永続化

３．学会等名
第24回プログラミングおよびプログラミング言語ワークショップ

４．発表年
2022年

１．発表者名
中田昌輝，鵜川始陽

２．発表標題
不揮発性メモリを用いた高い応答性を持つ定期的チェックポインティング

３．学会等名
第24回プログラミングおよびプログラミング言語ワークショップ

４．発表年
2022年

１．発表者名 鵜川始陽，松本康太郎，岩崎英哉
２．発表標題 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
３．学会等名 日本ソフトウェア科学会 第38回大会
４．発表年 2021年

１．発表者名 飯干 寛幸，松本 康太郎，鵜川 始陽
２．発表標題 不揮発性メモリを使ったデータ永続化システムNV-HTMの評価
３．学会等名 第148回 システムソフトウェアとオペレーティング・システム研究会
４．発表年 2020年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考
研究協力者	岩崎　英哉 (Iwasaki Hideya) (90203372)	明治大学・理工学部・専任教授 (32682)	
研究協力者	高田　喜朗 (Takata Yoshiaki) (60294279)	高知工科大学・情報学群・教授 (26402)	

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関