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

平成　３０　年　　５　月　２８　日現在

機関番号：　１４３０３
研究種目：　基盤研究(C)（一般）
研究期間：　2015～2017
課題番号：　１５Ｋ００００７６

研究課題名（和文）スレッドレベル並列投機実行の制御を支援する順序付きトランザクショナルメモリの研究

研究課題名（英文）Study on Ordered Transactional Memory Supporting the Thread-Level Parallel Speculation

研究代表者

平田　博章（HIRATA, Hiroaki）

京都工芸繊維大学・情報工学・人間科学系・准教授

研究者番号：９０２７３５４９

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）：逐次プログラムを並列化するために多くの技術が開発され、実用化されているが、並列化できないプログラムもまだ多く存在する。そのようなプログラムを並列化するため、本研究では投機的メモリ（SM）という概念を提案し、そのプロトタイプシステムを開発した。SMでは、プログラマがスレッドの並列投機実行を明示的に指定する。SMシステムは、スレッドが投機的に読み書きしたメモリデータを管理し、データに矛盾が生じたことを検出すると、スレッドを実行し直す。データの矛盾が頻繁に発生しなければ、プログラムの実行時間を短縮でき、そのための学術的な基盤を確立した。

研究成果の概要（英文）：Many techniques for parallelizing a sequentially coded program have been developed and put to practical use. But still many program codes cannot be parallelized because it is impossible to assure that the candidate program for parallel execution produces the same results as the original program. To parallelize such programs, we have proposed speculative memory (SM) and developed an SM prototype system. With SM, programmers can specify the parallel and speculative execution of threads explicitly in their programs. The SM system manages the memory data that are speculatively read or written by the threads running in parallel. When the system detects inconsistent memory accesses, it recovers the computational state of the program and restarts the execution. Unless such inconsistencies often occur, the total execution time of the program can be shorter. Consequently, we could establish the fundamental of parallel speculation.

研究分野：　情報工学

キーワード：　計算機システム　ハイパフォーマンスコンピューティング　投機実行　スレッドレベル並列処理　メモリシステム

１．研究開始当初の背景

(1) コンピュータアーキテクチャの分野では、特に 1990 年以降は、（マシン）命令レベルの並列性を活用して高速化を図る技術が盛んに研究され、半導体技術の進歩と相まって大きな発展を遂げた。その後、この命令レベル並列処理による性能向上の限界が見え始めると、次第に、命令レベルよりもさらに粗粒度のスレッドレベルの並列性の活用へと技術的動向が変化した。その結果、今日では、パソコン用のコンピュータに搭載するマイクロプロセッサにおいても、複数のプロセッサ（コア）を設けてスレッドレベル並列処理を行うマルチプロセッサ（マルチコア）構成（マルチスレッドアーキテクチャ）を採用することが一般的となっている。しかし、この変化は、最新のコンパイラやプロセッサを用いることで逐次プログラムの実行時間が短縮される状況から、スレッドレベルの並列プログラムを記述しなければ実行時間の短縮が望めない状況への変化と捉えられる。

(2) このような技術的動向を背景に、本研究代表者および分担者は、スレッドレベル並列実行アーキテクチャに関する研究を行ってきた。

(3) まず、逐次プログラムを従来の並列化手法の枠を超えた方法で並列化を行い、プログラム実行時間の短縮とプログラマの負担軽減を両立することを目的に、プロセッサの性能評価に用いる SPEC ベンチマークを用いて、「本質的に並列実行可能なアルゴリズムを採用しているにもかかわらず、単一スレッド実行を前提としてプログラミングしたソースプログラムに対して、これをコンパイラによって並列化しようとする場合に顕在化する並列化に対する阻害要因」を定性的かつ詳細に調査した。その結果から、逐次プログラム中のループに対して、ループ本体に関数呼び出しを含むなどの理由で静的には並列化不可能な場合であっても、自動的にスレッドを生成して投機的に並列実行することにより、スレッド間に存在する依存関係の 8 割以上を除去するメモリリネーミング機構を開発するに至った。

(4) また、並列プログラムに対しては、スレッド間の同期に関するプログラマの負担を軽減する目的で、トランザクショナルメモリ（以下、TM）の研究を行ってきた。大きな期待が寄せられ、国内外で盛んに研究成果が報告されていたが、投機実行の成功時と失敗時のどちらか一方で必ずオーバヘッドを伴っていた。そこで、本研究代表者らは、成功/失敗時のどちらにおいてもオーバヘッドが生じない TM の新たな実現方式を開発した。

(5) 本研究は、上記(3)と(4)の 2 つの研究から着想した発展的な研究である。

２．研究の目的

(1) 本研究は、研究代表者および分担者がそれまで行ってきた研究を融合・発展させて大規模な投機実行を可能とすることにより、従来の限界を超えて大きな並列処理効果を引き出す技術を確立することを目的とする。

(2) 一般に、投機実行とは、高速化を目的として（研究代表者および分担者がそれまでに行ってきた研究も含めて）ハードウェアのみによって実現する。しかし、投機実行の規模（投機実行するマシン命令数）が大きくなると、スレッドのコンテクストを管理しなければならなくなり、それをハードウェアのみで実現することは不可能である。そこで、本研究では、これまでの「ハードウェアのみで制御する」という範疇を超え、ソフトウェア機能も利用してスレッドの投機実行を実現することによって、大規模なスレッドレベルの並列投機実行を可能とする。

３．研究の方法

(1) 本研究では、あらかじめ複数のスレッドを生成しておき、ループのイタレーション間に依存関係が無いものと仮定して、それらを投機的に実行する。依存関係によるハザードを検出した場合には、そのスレッドのそれまでの実行結果を破棄して、再実行を行う。このような大規模スレッドレベル並列投機実行方式を実現するための概念を「順序付き TM」として提案し、これを用いるプログラムとのインタフェース仕様を策定する。

(2) 「順序付き TM」を実装するためのアーキテクチャレベルの概略設計を行う。ハードウェアとソフトウェアとの機能分担を検討し、各モジュールの仕様を明確化する。

(3) 上記の概略設計に基づいて、「順序付き TM」を実現するマルチスレッドライブラリを開発する。

(4) 上記のマルチスレッドライブラリを用いたプロトタイプシステムで性能評価を行い、「順序付き TM」の有効性を検証する。

４．研究成果

(1) 本研究の目的とする大規模並列投機実行方式を実現するために順序付き TM が提供すべき機能を検討し、新たに「投機メモリ（以下、SM）」としてその仕様を設計した。例えば、図 1（a）に示すループ（ここでは、プログラムを C 言語風に記述することとする）を並列化する場合、プログラマは図 1(b)のようにプログラムを記述する。これをコンパイラによって、図 1(c) および図 1(d)のように、SM を用いたプログラムに再構造化する（ただし、本研究ではこのコンパイラの開発は対象外としているため、現時点では手作業で図 1(a)ま

たは図 1(b)のプログラムを、図 1(c) および図 1(d)のプログラムに書き換える)。図 1(c)において名前が「sm_」で始まる関数は SM のライブラリ関数であり、元のループのループ本体は、図 1(d)に示すように新たな関数として記述する。

```
for( i = 0 ; i < n ; i++ ) {
    .....
    loop body
    .....
}
```

(a) ソースプログラム

```
speculative_for( i = 0 ; i < n ; i++ ) {
    .....
    loop body
    .....
}
```

(b) 投機実行用ソースプログラム

```
for( i = 0 ; i < n ; i++ ) {
    th = sm_next_thread();
    sm_push_arg(th,i);
    sm_start_speculation(th,sub);
}
sm_wait_until_committed();
```

(c) 再構造化したプログラム

```
void sub(int i)
{
    .....
    loop body
    .....
    sm_terminate(SM_CONTINUE);
}
```

(d) ループ本体の関数化

図 1 投機実行用プログラムへの変換

図 1 で使用した SM ライブラリ関数の機能を表 1 に示す。なお、プログラム起動時の初期スレッドをメインスレッドと呼び、ループを並列投機実行する前に、メインスレッドが複数のスレッド（以下、これをサブスレッドと呼ぶ）を予め生成しておくものとする。図 1(c)のコードはメインスレッドが実行し、図 1(d)の関数はサブスレッドが実行する。

表 1　SMライブラリ関数の仕様

sm_next_thread()
アイドル状態のサブスレッドの識別子を返す。

sm_push_arg(th,arg)
サブスレッドthに渡す引数argを設定する。

sm_start_speculation(th,func)
サブスレッドthを起動する。これにより、サブスレッドthは関数funcの投実行を開始する。

sm_wait_until_committed()
起動された全てのサブスレッドが、それらの起動時に指定された関数の実行を終了するのを待つ。

sm_terminate()
データの無矛盾性チェックを行う。矛盾が生じていればアボートし、矛盾が生じていなければコミットして終了する。

このように、SM ライブラリを用いることで自然に並列投機実行が行えるように SM のインタフェース仕様を策定した。

(2)　SM の実装についてアーキテクチャレベルで検討を行った。オペレーティングシステムが提供する（投機的でない）スレッドを利用して投機実行を行うためのスレッド管理機能および制御構造を明らかにした。また、SM の実装に関して、データの無矛盾性検査のポリシ、スレッド間の同期方式、データのバージョン管理、などにおける選択肢を比較検討した。

(3) 上記(2)の結果を基に、SM を実現するマルチスレッドライブラリを開発した。オペレーティングシステムのスレッド機能を直接利用せずに POSIX Threads (PThreads) をベースに開発したため、性能の点では最適化やチューニングの余地はあるが、移植性が高く、様々なコンピュータで利用することができる。SM を用いた並列投機実行方式の研究用プロトタイプシステムとして、十分な実験環境を整えることができた。

(4) 上記(3)のプロトタイプシステムで性能評価を行った。二分探索木の生成処理を投機実行を行うことで並列化した場合の性能向上比を図 2 に示す。

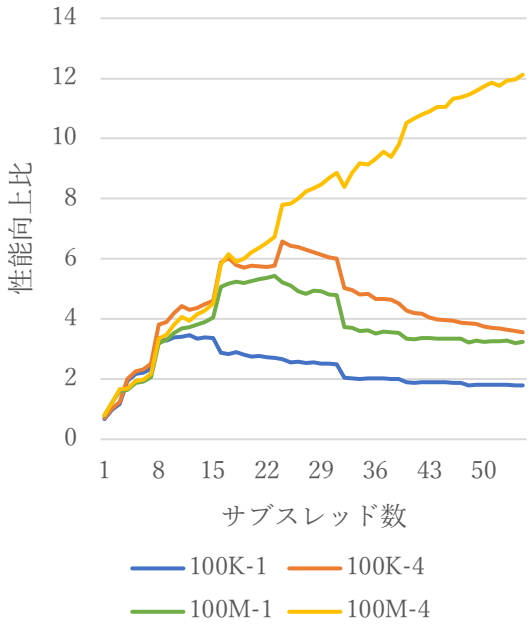

図 2 並列投機実行による性能向上比

図 2 のグラフでは、横軸がサブスレッド数、縦軸が逐次プログラムの性能（実行時間の逆数）を 1 としたときの性能向上比、をそれぞれ表す。データ数が 10 万個の場合（100K-1）と 1 億個の場合（100M-1）で性能評価を行った。いずれの場合もサブスレッド数を増やすことで性能が向上するが、増やしすぎると性能向上比は低下する。これは並列処理粒度が細かいのに対してサブスレッド間での同期処

理にかかるオーバヘッドが増大したことに起因する。そこで、ループの 4 個のイタレーションをループアンローリングによって 1 つにまとめることで並列処理粒度を粗くした（100K-4、100M-4）。データ数が 1 億個の場合（100M-4）はループアンローリングを施さない場合（100M-1）よりも高い性能が得られ、また、サブスレッド数の増加とともに性能向上比も向上する。一方、データ数が 10 万個の場合（100K-4）は、ループアンローリングを施さない場合（100K-1）よりも高い性能は得られるものの、依然としてサブスレッド数を増やしすぎると性能向上比の低下が見られる。解析の結果、この原因は投機実行失敗時のアボートのペナルティによるものであることが判明した。

(5) データ数が 1 億個の場合は生成する二分探索木のサイズが大きくなるため、投機実行が失敗する比率は 0.02%以下と小さい。そのため、性能向上比はサブスレッド数の増加に伴って向上する。10 万個の場合に投機実行が失敗する比率は 8%以下であり、値としてはそれほど大きなものとは言えない。しかし、1 億個の場合と比べると大きく、また、ループアンローリングによってアボート時のペナルティが大きくなっている。そのため、サブスレッド数を増やしすぎるとアボートによる再実行に要する時間がオーバヘッドとなって性能が低下する。ただし、アボート時のペナルティを削減するための方策についてのアイデアはすでに得ており、今後の研究でその効果を確認する予定である。

(6) 近年はプロセッサとメモリとの性能差がさらに拡大し、そのため、共有メモリアクセスに関する並列処理オーバヘッドも増大する傾向にあるが、本研究により、そのようなオーバヘッドを伴っても、並列投機実行を行うことでさらに性能向上が可能であることが確認できた。SM（順序付き TM）の概念が学術的に有望であることを示した点で、本研究結果の意義は大きい。結論として、本研究により、これまでは並列化不可能であったプログラムを並列化可能にし、プログラムの実行時間をさらに短縮する技術の基礎が確立できた。

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 15 件）

① 山崎寛季、布目淳、平田博章、Parallelizing the Construction of a k-Dimensional Tree、Proceedings of the 3rd International Conference on Big Data, Cloud Computing, and Data Science Engineering (BCD 2018)、査読有、2018

② 布目淳、平田博章、A Data Migration Scheme Considering Node Reliability for an Autonomous Distributed Storage System、Proceedings of the 5th International Conference on Computational Science/ Intelligence and Applied Informatics (CSII 2018)、査読有、2018

③ 藤澤昂平、布目淳、柴山潔、平田博章、Design Space Exploration for Implementing a Software-based Speculative Memory System International Journal of Software Innovation (IJSI)、査読有、Vol.6、No.2、2018、pp.37-49

④ 嶋野真吾、布目淳、横井雄太、柴山潔、平田博章、A Dynamic Configuration Scheme of Storage Tiers for an Autonomous Distributed Storage System、Information Engineering Express、査読有、Vol.3、No.4、2017、pp.91-104

⑤ 一井世界、林昇平、布目淳、平田博章、柴山潔、Performance Evaluation of Delayed-Committing Transactional Memory、Proceedings of the 18th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017)、査読有、2017、pp.445-451

⑥ 藤澤昂平、布目淳、柴山潔、平田博章、A Software Implementation of Speculative Memory、Proceedings of the 18th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017)、査読有、2017、pp.437-443

⑦ 嶋野真吾、布目淳、横井雄太、柴山潔、平田博章、An Autonomous Configuration Scheme of Storage Tiers for Distributed File System、Proceedings of the 18th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017)、査読有、2017、pp.453-458

⑧ 出島貴史、布目淳、平田博章、スレッドレベル並列投機実行のためのデータ依存解析機構、情報処理学会第 79 回全国大会講演論文集、査読無、2017、Vol.1、pp.67-68

⑨ 平田博章、布目淳、柴山潔、Speculative Memory: An Architectural Support for Explicit Speculations in Multithreaded Programming 、Proceedings of the 15th International Conference on Computer and Information Science (ICIS 2016)、査読有、2016、pp.715-721

⑩ 布目淳、平田博章、柴山潔、An Interval Control Method for Status Propagation in an Autonomous Distributed Storage System、 Proceedings of the 15th International Conference on Computer and Information Science (ICIS 2016)、査読有、2016、pp.723-728

⑪ 小路勇気、布目淳、平田博章、柴山潔、A Large-Scale Speculation for the Thread-Level Parallelization 、International Journal of Computer and Information Science (IJCIS)、査読有、2016、Vol.17、No.1、pp.24-32

⑫ 小路勇気、布目淳、平田博章、柴山潔、スレッドレベル並列投機実行のためのデータ値予測機構、第14回情報科学技術フォーラム(FIT2015)講演論文集、査読無、2015、Vol.1、pp.243-244

⑬ 一井世界、田代早紀、布目淳、平田博章、柴山潔、Hardware Transactional Memory with Delayed Committing、Proceedings of the 3rd International Conference on Applied Computing and Information Technology (ACIT 2015)、査読有、2015、pp.161-168

⑭ 小路勇気、布目淳、平田博章、柴山潔、A Large-Scale Speculation for the Thread-Level Parallelization 、Proceedings of the 3rd International Conference on Applied Computing and Information Technology (ACIT 2015)、査読有、2015、pp.169-175

⑮ 嶋野真吾、布目淳、平田博章、柴山潔、An Information Propagation Scheme for an Autonomous Distributed Storage System in iSCSI Environment 、Proceedings of the 3rd International Conference on Applied Computing and Information Technology (ACIT 2015)、査読有、2015、pp.149-154

〔学会発表〕(計12件)

① 山崎寛季、Parallelizing the Construction of a k-Dimensional Tree、The 3rd International Conference on Big Data, Cloud Computing, and Data Science Engineering (BCD 2018)、2018

② 布目淳、A Data Migration Scheme Considering Node Reliability for an Autonomous Distributed Storage System 、Proceedings of the 5th International Conference on Computational Science/ Intelligence and Applied Informatics (CSII 2018)、2018

③ 林昇平、Performance Evaluation of Delayed-Committing Transactional Memory 、The 18th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017)、2017

④ 藤澤昂平、A Software Implementation of Speculative Memory 、The 18th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017)、2017

⑤ 横井雄太、An Autonomous Configuration Scheme of Storage Tiers for Distributed File System、The 18th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017)、201

⑥ 出島貴史、スレッドレベル並列投機実行のためのデータ依存解析機構、情報処理学会第79回全国大会、2017

⑦ 平田博章、Speculative Memory: An Architectural Support for Explicit Speculations in Multithreaded Programming、The 15th International Conference on Computer and Information Science (ICIS 2016)、2016

⑧ 布目淳、An Interval Control Method for Status Propagation in an Autonomous Distributed Storage System、 The 15th International Conference on Computer and Information Science (ICIS 2016)、2016

⑨ 小路勇気、スレッドレベル並列投機実行のためのデータ値予測機構、第14回情報科学技術フォーラム(FIT2015)、2015

⑩ 田代早紀、Hardware Transactional Memory with Delayed Committing、The 3rd International Conference on Applied Computing and Information

Technology (ACIT 2015)、2015

⑪ 小路勇気、A Large-Scale Speculation for the Thread-Level Parallelization、The 3rd International Conference on Applied Computing and Information Technology (ACIT 2015)、2015

⑫ 嶋野真吾、An Information Propagation Scheme for an Autonomous Distributed Storage System in iSCSI Environment、The 3rd International Conference on Applied Computing and Information Technology (ACIT 2015)、2015

〔図書〕(計０件)

〔産業財産権〕

○出願状況 (計０件)

名称：
発明者：
権利者：
種類：
番号：
出願年月日：
国内外の別：

○取得状況 (計０件)

名称：
発明者：
権利者：
種類：
番号：
取得年月日：
国内外の別：

〔その他〕
ホームページ等

６．研究組織
(1)研究代表者
平田　博章　(HIRATA, Hiroaki)
京都工芸繊維大学・情報工学・人間科学系・准教授
研究者番号：９０２７３５４９

(2)研究分担
布目　淳　(NUNOME, Atsushi)
京都工芸繊維大学・情報工学・人間科学系・助教
研究者番号：６０３３５３２０

柴山　潔　(SHIBAYAMA, Kiyoshi)
京都情報大学院大学・その他の研究科・教授
研究者番号：７０１２７０９１

(3)連携研究者
(　　)

研究者番号：

(4)研究協力者
(　　)