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　２９　年　　６　月　２１　日現在

機関番号：　１２１０２
研究種目：　挑戦的萌芽研究
研究期間：　2015～2016
課題番号：　１５Ｋ１２００６

研究課題名（和文）メニーコアと大容量主記憶を持つ計算機向けのビッグデータの並列処理方式

研究課題名（英文）A Parallel Processing Scheme for big data targeting a computer with many CPU cores and a large capacity main memory.

研究代表者

加藤　和彦（KATO, kazuhiko）

筑波大学・システム情報系・教授

研究者番号：９０２２４４９３

交付決定額（研究期間全体）：（直接経費）　　2,600,000円

研究成果の概要（和文）：メニーコアと大容量主記憶持つ計算機をターゲットとしたビックデータ処理の高速処理手法の研究開発を行った。有用なビックデータ処理の多くが，並列性を引き出しやすい性質をもつ数学的性質を有することを利用し，データの分割と並列処理を行い，中間処理結果を，新開発したロック不要並列マージ処理を行うことによって結果を得られるようにした。プログラミング言語Scalaを用い，同言語のマルチスレッド並列化支援を機能を駆使して実装した。実機を用いた実験を行い，高速なビッグデータ並列処理を行えることを確認した。

研究成果の概要（英文）：We have conducted research and development on a high-speed processing method for big data processing targeting a computer with many CPU cores and a large capacity main memory. Many of useful big data processing has mathematical properties to derive parallelism. By using that, we proposed a parallel processing scheme to data partition and processing, and the intermediate results are merged in an almost lock-free manner, which enables full-parallelism. We implemented the scheme using the Scala programming language. We conducted experiments using real machines and confirmed that high-speed big data parallel processing can be performed efficiently with the scheme.

研究分野：　システムソフトウエア

キーワード：　ビッグデータ　メニーコア　並列処理　B木　B+木

１．研究開始当初の背景

今日，サーバー計算機からクライアント計算機まで，プロセッサのマルチコア化が進んでいる。既に 1 プロセッサあたり 10 コア以上の商用量産マイクロプロセッサが現実のものとなっており，数 10 から 100 コア以上のコア数を持つメニーコア計算機が容易に入手可能な時代となりつつある。

メニーコアを操るソフトウェア技術は，従来，SMP (Symmetric Multiprocessing) 計算機と呼ばれていた共有メモリ型並列機の並列プログラミング技術の系譜上にあると考えられ，少なからぬ技術開発の歴史を有する。SMP 計算機は，以前から OS が提供するマルチプロセス機能、および，SMP 計算機の登場と同時期に開発されたマルチスレッド機能と相性のよい計算方式であり，マルチコアも同様であると考えられるが，これだけマルチコアが至る所に普及するようになった今日においても，マルチコア／メニーコアを有効活用する並列プログラミングと並列実行は，未だ成熟の域に達していない。

並列処理対象の問題を，互いに独立性が高く，かつ，高並列実行出来る部分処理に分解できるときは，並列プログラミングは容易化し，並列処理による性能向上も得やすい。これは，逐次処理プログラムは一般に，並列処理プログラムよりもプログラム開発を行いやすく，また，実行時にプログラム間の同期や通信等の協調処理を少なくすることが出来るためである。その反対に，このような分解が容易には行えないとき，並列プログラミングは容易でなく，並列化による性能向上も得にくい。現実のアプリケーションでは，そのような分解が容易でないアプリケーションが多く，そのために，メニーコア用アプリケーションプログラムの設計や再設計は容易でなく，複雑なプログラム構造，複雑な同期・通信機構の呼び出しが必要であり，またそれに伴い，効率的なメニーコア並列処理が難しくなっていると考えられる。

２．研究の目的

本研究では，複雑な構造を持ち，動的にその内容や構造が変化し得る,比較的大きなメモリ上の共有データを，メニーコア計算機にて並列処理する場合を対象とし，その並列プログラミングを見通しよく行い，かつ，効率のよい並列処理を可能とする方式を研究開発する。並列処理の効率の尺度としては単位時間あたりの処理量、すなわち,スループットの向上を当面のターゲットとする。コア数が大きくなってもスケーラビリティが失われないことを目指す。

このような処理対象は，前述の「分解」が一般に難しい処理である。本研究では，そのような処理であっても，ある条件を満たす場合は「分解」が可能であることに着眼する。共有データをパブリック領域にあるデータ（パブリックデータと呼ぶ）と見なしたとき，パブリックデータへの更新をプライベートスペース上のデータ（プライベートデータと呼ぶ）において並列に行う。プライベートデータは適宜，一貫性制約を保ちつつ，後からまとめてパブリックデータに反映させる。このような処理を行っても，対象処理のセマンティクスが変わらない，もしくは，許容可能な場合を条件とする。

この条件を満たし，かつ，実用的な処理を行える処理対象として，データベースのインデックス構造として広く実用に供している B 木構造の並列処理を考える。B 木の並列処理方式は，B 木の提案者である R. Bayer ら自らによって提案され（参考文献 [1]），その並列処理方式は今日でも有効である。B 木はエレガントなデータ構造ではあるものの，逐次版プログラムであっても複雑性は高く，また，lock/unlock に基づく並列処理方式のプログラムはさらに複雑で，高い並列性を引き出すことは難しい。我々は既に，B 木の更新操作が前述の条件を満たすと仮定したとき，提案方式によって並列 B 木操作を見通しよくプログラミングし，さらに，Bayer らの方式よりもスループットとスケーラビリティの高い並列処理が可能であるとの先行実験結果を得ている。

<u>特色・独創的な点</u>

研究代表者らはこれまで長年に渡って，分散処理研究，および，二次記憶上の複雑なデータ構造（B 木のようなインデックス構造がその典型）に関する研究を行ってきた。マルチコア／メニーコアのハードウェア技術は広く使われるようになりつつあるが，未だに並列ソフトウェア技術がハードウェア環境の変化に追いついていないことに不満を感じ，メニーコアが提供する潤沢なハードウェア資源を十分に活用しながらも，逐次処理プログラムの開発に近い並列プログラムの生産性を提供できる方法を探っていた。その中で，以前からデータベースの分野で知られるトランザクション概念を並列処理に適用した，ソフトウェア・トランザクション・メモリ（STM）と呼ばれる技術に有望性を感じて試験的な研究を試みた。しかし，本研究が対象とするような，複雑で動的で巨大性のあるデータ構造に対しては，少なくとも現在の STM 技術では，有効に機能させることは難しいとの試行結果を得た。しかしその試行過程の中で，我々が以前行っていた「サステーナブルシステム」と我々が名付けた研究において使用していた概念をメニーコア並列処理に適用することで，通信や同期等の協調処理を処理の途上で省略しても，全体として矛盾のない計算が可能であるとの着想を得た。サステーナブルシステムでは，分散システムの構成要素（計算機ノードやネットワーク）に

部分的な障害が発生し，ネットワーク分断のような事態が発生しても，（一般に複数の）部分システムにおいて計算を続行し，障害回復時にマージ処理を行うことで，システム全体の一貫性を保持する。このように，元々分散システムのために開発された技術を，視点を変え，メニーコアという並列計算環境に適用するというアイデアが本研究の特色であり，独創的な点である。

予想される結果と意義

本研究の最大の意義は，条件付きではありながらも，その有用性は十分に広いと考えられる対象領域において，メニーコア処理技術をユーザに提供することにある。そのことを通じ，未だそのプログラミングが容易ではなく，ハードウェア資源を十分に活かし尽くすことが容易ではないメニーコア環境の適用範囲を現在よりも大きく広げることを目指す。

データの高スループット処理を，高いソフトウェア生産性のもとで行う技術としては，Google 社の MapReduce システムやその類似システムが有名である。MapReduce は，クラスタシステムという分散計算環境を実行環境としている点で，本研究の想定実行環境とは異なるが，概念レベルで考えると，本研究は MapReduce の一般化と考えることができる。MapReduce では，Map ステップが本研究のプライベートデータに対する分散処理，Reduce ステップが本研究のパブリックデータに対する分散処理に相当する。本研究では分散処理の部分を，メニーコアにより並列処理する。プロセッサノード間のネットワーク通信は，コンピュータ筐体内の CPU-メモリ-バス間通信に相当する。

MapReduce はクラスタマシン上で実行され，高いスケーラビリティを有するが，本アプローチではスケーラビリティに関しては，メニーコアのハードウェア的制約を受けるものの，今後のコア数の増加は期待され，また，マシンの管理コストは大幅に削減される。MapReduce は，対象処理を Map と Reduce というデザインパターン」に落とし込まねばならない。このために，B 木のような，再帰性を有するデータ構造の並列処理は一般に困難である。本アプローチはより汎用的であり，特定のオペレーションに依存するものではないため，その応用範囲はより広く，柔軟性が高い。

参考文献[1] R. Bayer et al. Concurrency of operations on B-trees. Acta Informatica, 9(1), 1977, pp. 1-21.

３．研究の方法

メニーコアによる並列処理を見通しよくプログラミングし，高スループット，高スケーラブルな性能を引き出す方法を開発する。提案手法を Cave and Court Computing (CCC) と名付ける。CCC では，マルチスレッドによる並列計算を cave モードと court モードの二モードに分解する。cave モードでは，スレッド間の同期制御を不要とし，同期オーバヘッドなしで計算を行う。cave モードにおいて各スレッドは，グローバルデータの参照と，ローカルデータの参照・更新のみが可能である。court モードでは，スレッド間同期をとりながら，各スレッドが生成したローカルデータをグローバルデータに反映させる。グローバルデータのデータ構造の工夫と，cave/court モードの各操作の最適化，およびモード切り替えタイミングの動的最適化等の手法を総合して高性能性を引き出す。

具体的な方法

本研究では，複雑な構造を持ち，動的にその内容や構造が変化し得る，比較的に大きなメモリ上の共有データのメニーコア計算機による並列処理を，見通しよくプログラミングし，かつ，スケーラブルな性能を引き出す方法を開発する。我々は提案手法を Cave and Court Computing (CCC) と名付けた。“cave and court”とは，オフィス環境等をデザインをするときに用いられる考え方で，cave とは個人が他とのコミュニケーションを絶ち，自分の仕事に没入する，洞窟のような私的空間を指す。court とは，cave における私的な仕事を持ち寄り，仲間と共にコミュニケーションを取って相互の意見を交換し，合議する公的空間を指す。

提案方式 CCC の概略を説明する。メニーコア上で動作するプログラムを，lock/unlock 方式で相互同期操作を行うマルチスレッド・プログラムと考える。各スレッドが共通的にアクセス出来るグローバル共有メモリ領域と，各スレッドのみが独自にアクセスできるローカルメモリ領域を持つとする（グローバル共有メモリしかない場合でも，その一部をローカルメモリ領域として各スレッドに割り当てることが可能なので一般性は失わない）。グローバル共有メモリ領域に置かれるデータをパブリックデータ，ローカルメモリ領域に置かれるデータをプライベートデータと呼ぶ。各スレッドがパブリックデータにアクセスするときは必ず，lock/unlock プリミティブ等を発行して同期制御を行う必要がある。

CCC は以下に述べる二つのアイデアの組合せである。第一のアイデアは，マルチスレッドによる並列計算を cave モードと court モードの二モードに分解し，このモードを交互に繰り返すことである。cave モードにおいて各スレッドは，グローバルデータの read と，自身のプライベートデータの read/write が可能である。court モードにおいて各スレッドは，グローバルデータと自身のプライベートデータの双方 read/write が可能である。ただし，グローバルデータを read/write するときはスレッド間で同期制

御を行う必要がある。つまり，cave モードにおいては，各スレッドは，スレッド間の同期制御を行わない，単なる逐次プログラムとして「全速力」でプログラム実行を行う。court モードにおいては，スレッド間の同期制御を行う，従来の並列プログラム実行を行う。

CCC の第二のアイデアは，cave モードでの実行をできるだけ長くすることができるように，問題領域において求められる結果の等価性を保ちながら，並列処理対象の一貫性制約を緩和する許容性(allowance) があるようなデータ構造とアルゴリズムを選択・設計し，cave モードでの処理と court モードでの処理記述を行う。

４．研究成果

本研究では，メニーコアと大容量主記憶持つ計算機をターゲットとしたビッグデータ処理の高速処理手法の研究開発を行った。有用なビッグデータ処理の多くが，並列性を引き出しやすい 性質をもつ代数的構造(モノイド)と見なせることを利用し，データの分割と並列処理を行い，中間処理結果を，新開発したロック不要並列マージ処 理を行うことによって結果を得られるようにした。

提案方式の詳細化と共に実装を進めた。実装言語として Scala 言語を用い，Scala 言語が提供する future, 並列データ構造等のマルチスレッド並列化支援を機能を駆使して実装した。メニーコア計算機は，マルチコア計算ノードの集合体である。マルチコア計算ノード内はマルチスレッド内の提案並列計算を行い，マルチコアの計算結果を集めて結果をマージするために, 高性能 key-value store である Redis システムを用いて情報集約を行うこととした。ビッグデータのデータとして Wikipedia の全英語データ（約 38GB）を用いた。実装と実機を用いた実験によって，マルチコア環境，および，メニーコア環境の双方で，高速なビッグデータ並列処理を行えることを確認した。

ここで得られた研究成果により，クラスタ計算環境に比べてシステム構築・保守を圧倒的に楽に行えるメニーコア計算機において，ソフトウェアの生産性高く，ビッグデータ処理できる基礎技術を作った。実験による評価の過程で，負荷分散をさらに的確に行うことによって，更なる並列性の向上，同期制御の削除，およびメモリコピー回数の削減を行える可能性があることを見出した。

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 0 件）

〔学会発表〕（計 1 件）

①天沼　駿亮,鷹野　駿介,加藤　和彦,
阿部　洋丈
メニーコアマシン環境における並列 B 木マージを利用したビッグデータ処理方式の検討,
日本ソフトウエア科学会第 32 回大会 2015 年 9 月 11 日,早稲田大学,東京

６．研究組織

(1)研究代表者
加藤　和彦（KATO,Kazuhiko）
筑波大学・システム情報系・教授
研究者番号：90224493

(2)研究分担者
阿部　洋丈（ABE,Hirotake）
筑波大学・システム情報系・准教授
研究者番号： 00456716
長谷部　浩二（HASEBE,Koji）
筑波大学・システム情報系・准教授
研究者番号： 80470045