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　元 年　 ６ 月 １９ 日現在

機関番号：３２６８２
研究種目：基盤研究(C)（一般）
研究期間：2016～2018
課題番号：１６Ｋ００１７４

研究課題名（和文）マルチプラットフォーム向けJava粗粒度並列処理コードの自動生成に関する研究

研究課題名（英文）Study on Automatic Code Generation of Java Coarse Grain Parallel Processing for Multi-Platform

研究代表者
吉田　明正（Yoshida, Akimasa）

明治大学・総合数理学部・専任教授

研究者番号：６０２７７８４５

交付決定額（研究期間全体）：（直接経費）　　3,600,000円

研究成果の概要（和文）：マルチコアプロセッサはサーバからスマートフォンに至るまで，高い計算性能を達成するために利用されている．マルチコアプロセッサの性能を最大限に引き出すためには対象プログラムに内在する並列性を最大限に利用することが必要であり，本研究では，Javaプログラムを対象として，粗粒度並列処理を実現するタスク駆動型粗粒度並列処理を提案した．また，メニーコアプロセッサを有効利用するために，ローカルタスク協調実行方式を開発した．これらの手法は，本研究で開発した並列化コンパイラに実装されており，x86，POWER，ARMのマルチプラットフォームで行った性能評価の結果から，Java粗粒度並列処理の有効性が確認された．

研究成果の学術的意義や社会的意義
本研究はマルチプラットフォーム環境において，Java処理系に備わっているFork/Join Framework（並列処理ソフトウェア）を利用して，粗粒度タスク並列処理を実現するタスク駆動型粗粒度並列処理手法を提案している．本手法は，並列処理の実行環境に依存せず，Javaプログラムの並列性を最大限に活用できることに優位性がある．また，本手法は高性能なサーバからPCやAndroidスマートフォンに至るまで，容易に利用可能なシステムソフトウェア技術であり，学術的意義に加えて社会的意義も大きい．

研究成果の概要（英文）：Many-core processors are used to realize high-performance computing for servers and smartphones. In order to maximize the performance of multicore processors, it is required to utilize the inherent parallelism of target programs. Therefore, this research proposed the task-drive coarse grain parallel processing that realizes the coarse grain parallelization for Java programs. Also, in order to use many-core processor efficiently, this research proposed the local task cooperative execution method to use not only coarse grain parallelism but also local task parallelism. These schemes are implemented on the developed parallelizing compiler. From the results of performance evaluation on multi-platform environment such as x86, POWER and ARM, effectiveness of the Java coarse grain parallelization was confirmed.

研究分野：並列処理

キーワード：粗粒度タスク並列処理　ローカルタスク協調実行　並列化コンパイラ　Java　Fork/Join Framework　メニーコア　POWER8　ダイナミックスケジューリング

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９、ＣＫ－１９（共通）

１．研究開始当初の背景

コンピュータ高速化技術として，マルチコアプロセッサを用いた並列処理は必須のテクノロジーとして注目されており，高性能なスーパーコンピュータから，PC，タブレット，組み込みシステムに至るまで広く普及している．このようなマルチコアプロセッサを利用したシステムにおいて高い実効性能を達成するには，さまざまな並列処理技術を必要としており，現在までに，ループ並列処理，および，粗粒度タスク（ループやサブルーチン）レベルの並列性を利用する粗粒度タスク並列処理が必要とされている．

本研究のベースとする階層統合型実行制御を用いた粗粒度タスク並列処理では，まず，マクロタスクと呼ぶ粗粒度タスクにプログラムを分割し，マクロタスク間の依存関係を表現したマクロタスクグラフを生成する．その後，ダイナミックスケジューラが，全階層のマクロタスクの実行開始条件を統一的に管理し，均質なコアに割り当てて並列処理を行う．これにより，全階層の並列性を最大限に利用することが可能である．

並列処理の対象言語としては， PC や組み込みシステム等のソフトウェア開発の現場において Java 言語が広く利用されるようになってきており，Java プログラムによる並列処理への期待がより一層高まっている．Java プログラムの並列処理に関する研究は，OpenMP/MPI のような API を提供する Parallel Java，async や finish のような X10 の言語拡張を取り入れた Habanero-Java 等が提案されている．これらは言語拡張を前提としており，複数階層にまたがった粗粒度並列性を実行開始条件の形で統一的に管理する方式とは異なる．

以上を踏まえて，本研究の目的は，階層統合型実行制御による粗粒度タスク並列処理をマルチプラットフォームで実現するために，Fork/Join Framework を取り入れたマルチプラットフォーム向け並列 Java コードの仕様を作成，かつ，その並列化コンパイラを開発し，複数の並列プラットフォーム上での性能評価により，その有効性を確認することとする．

２．研究の目的

高性能なスーパーコンピュータから組み込みシステムに至るまでマルチコアを用いた高速化が普及している．マルチコア向けのソフトウェア開発言語としては，高性能なサーバから Android タブレットに至るまでマルチプラットフォームに対応可能な Java 言語が注目されている．本研究では，マルチコア上での Java プログラムの並列処理基盤として，階層統合型実行制御を用いた粗粒度タスク並列処理を採用し，新たに Java Fork/Join Framework を導入したマルチプラットフォーム向け並列 Java コードを開発する．加えて，本研究ではその並列プログラムを自動生成する並列化コンパイラを開発し，x86 系サーバ，ARM 系 Android タブレット，POWER8 系サーバで性能評価を行い，異なる命令セットと OS の並列プラットフォーム上においてマルチプラットフォーム向け並列 Java コードの実効性能を検証する．

３．研究の方法

(1)マルチプラットフォーム向け並列 Java コードの仕様作成

マルチプラットフォーム向け並列 Java コードは，対象並列コンピュータの命令セットに依存しない並列コードを想定しており，それらにより粗粒度タスク並列処理を実現する．この並列 Java コードの仕様作成にあたり，先行研究により Java Fork/Join Framework を用いて実装するアプローチの有効性が示されており，本研究においても Fork/Join Framework を採用する．本研究では実用的な Java プログラムに対応できるような仕様を作成する．

階層統合型実行制御を用いた粗粒度タスク並列処理では，階層型マクロタスクグラフを生成し，マクロタスク（MT）を階層的に定義する．その後，ダイナミックスケジューラが実行可能条件を満たしたすべての階層のマクロタスクを統一的にコアに割り当てて処理する．

Fork/Join Framework は，Java SE 7 や Android API Level 21 で導入された ExecutorService インタフェースを実装した並列処理フレームワークであり，粗粒度タスク並列処理におけるダイナミックスケジューラの役割を果たす．Fork/Join Framework による並列プログラムでは，main()メソッドにおいてスレッドプールを生成する．スレッドプール内に生成された各ワーカースレッドは独自のワーカーキューを持ち，それぞれが Fork されたタスクを実行することにより並列処理が行われる．このフレームワークは，コア数が増大した場合においても，ワークスティーリングによりロック競合の問題に対応することができる．

Fork/Join Framework を用いたマルチプラットフォーム向け並列 Java コードの仕様を確定し，手動で生成した並列コードを用いて，マルチコアプロセッサシステム上で予備評価を行う．予備評価を考慮して，マルチプラットフォーム向け並列 Java コードの仕様を確定する．

(2)マルチプラットフォーム向け並列 Java コードを自動生成する並列化コンパイラの開発

仕様を決定したマルチプラットフォーム向け並列 Java コードを自動生成する並列化コンパイラの開発を行う．応募者は現在までに，階層統合型実行制御を伴う粗粒度タスク並列処理の Java スレッドコードを生成する並列化コンパイラを開発しており，その並列化コンパイラをベースに本研究の並列化コンパイラを開発する．

並列化コンパイラの構成は，並列処理の対象となる指示文付 Java プログラムを入力とし，並列化コンパイラは字句解析，構文解析，意味解析，Fork/Join 用実行条件解析，Fork/Join 用並

列コード生成を行い，Fork/Join 実装のスケジューラを含んだ並列 Java コード（マルチプラットフォーム向け）を自動生成する．

生成されたマルチプラットフォーム向け並列 Java コードは,階層統合型実行制御による粗粒度タスク並列処理を，さまざまな並列プラットフォーム上で実現することが可能であり，高い実効性能が期待される．特に，本研究では，階層統合型実行制御のダイナミックスケジューリング処理コードを，一般的なライブラリである Fork/Join Framework により実現しており，コア数の多い並列プラットフォームにおいても，ワークスティーリングによるタスクマイグレーション（移動）の機能によって，スケジューリングオーバヘッドを小さく抑えられる．

性能評価では，Java Grande Forum Benchmark 等の Java プログラムにマクロタスクを定義する指示文を加え，その Java プログラムを開発した並列化コンパイラの入力として，マルチプラットフォーム向け並列 Java コードを自動生成する．その並列 Java コードは Fork/Join Framework を伴って実装されており，さまざまな並列プラットフォーム上で実行可能である．マルチコアプロセッサシステムを用いて性能評価を行い，異なる命令セットアーキテクチャ上でマルチプラットフォーム並列 Java コードの有効性を検証する．

４．研究成果

(1)マルチプラットフォーム向け並列 Java コードの仕様作成

初年度にあたる平成２８年度は,マルチプラットフォーム向け並列 Java コードの仕様を作成した．本並列 Java コードは，階層統合型実行制御を用いた粗粒度タスク並列処理を実現するものであり，階層的に定義されたマクロタスク（粗粒度タスク）の並列性が，実行可能条件として表現している．また，本並列 Java コードには，Fork/Join Framework を伴うダイナミックスケジューリングのためのマクロタスク実行管理コードも含まれている．

Fork/Join Framework は，Java SE 7 や Android API Level 21 で導入された並列処理フレームワークであり，粗粒度タスク並列処理におけるダイナミックスケジューラの役割を果たす．Fork/Join Framework による並列プログラムでは，各ワーカースレッドが独自のワーカーキューを持ち，Fork されたタスクを実行することにより並列処理が行われる．

以上のように，平成２８年度には，マルチプラットフォーム向け並列 Java コードの仕様を確定し，マルチコアプロセッサシステム上で予備評価を行い，マルチプラットフォーム向け並列 Java コードの有効性を確認した．

(2) マルチプラットフォーム向け並列 Java コードの仕様確定とプロトタイプの並列化コンパイラの作成

平成２９年度は，マルチプラットフォーム向け並列 Java コードの仕様を確定し，その並列化コンパイラ（プロトタイプ）を作成した．本並列 Java コードは，階層統合型実行制御を用いた粗粒度タスク並列処理を実現するものであり，階層的に定義されたマクロタスク（粗粒度タスク）の並列性が,実行可能条件として管理されている.Fork/Join Framework は JVM(Java Virtual Machine) 環境あるいは Android 環境で利用可能な並列処理フレームワークであり，粗粒度タスク並列処理におけるダイナミックスケジューラの役割を果たす．

さらに，平成２９年度は，メニーコア上のための高並列プログラムの並列 Java コード長を短縮するコードコンパクション手法も新たに開発した.本手法により並列 Java コード長が大幅に短縮され，並列処理時間が短縮されることが確認されている．

以上のように，平成２９年度には，マルチプラットフォーム向け並列 Java コードを自動生成するプロトタイプの並列化コンパイラを開発し，マルチコアおよびメニーコア上で性能評価を行い，マルチプラットフォーム向け並列 Java コードの有効性を確認した．

(3)マルチプラットフォーム向け並列 Java コードの自動生成

最終年度の平成３０年とは,本研究の目的としている x86 系や POWER8 系のようなマルチプラットフォームを対象とした並列 Java コードを自動生成するための並列化コンパイラ(システムソフトウェア）の開発を行った．開発した並列化コンパイラでは，粗粒度タスクを定義する指示文を加えた Java プログラムを入力として，Java Fork/Join Framework を利用するマルチプラットフォーム向け並列 Java コードの自動生成を可能にしており,並列プログラムの生成に伴うユーザ負担を軽減することが可能である．この並列化コンパイラにより生成された並列 Java コードは，全階層の粗粒度タスクを実行開始条件により統一的に管理し，Fork/Join Framework のタスクスケジューラにより，粗粒度タスクをコアに動的に割り当てることが可能である．特に，本年度はメニーコアのようなコア数の多い並列システムにおいて，高い実効性能を達成するために，通常の粗粒度タスクだけでなく，粗粒度タスク内のローカルタスクを協調して実行させる方式を新たに提案した．

性能評価では，保有している x86 系サーバ，本研究予算で購入した POWER8 系サーバを用いて性能評価を行い，マルチプラットフォーム向けの並列 Java コードの有効性を検証した．性能評価プログラムとしては，Java Grande Forum Benchmark Suite 等を採用している．性能評価の結果から,本手法により生成された並列 Java コードはマルチプラットフォーム環境下でも高い実効性能が達成されることが確認されており，その有効性が確認された．

(4) 研究成果の国内外における位置づけとインパクトと今後の展望

本研究の研究成果は，情報処理学会論文誌の論文 3 件，及び，情報処理学会システムアーキテクチャ研究会や全国大会等の学会発表 6 件となっている．

本研究は,サーバからスマートフォンに至るマルチプラットフォームを対象とした Java の粗粒度並列処理コードを自動生成するという独創性の高い研究であり，国内外の関連研究と比べても提案手法の優位性が確認されている．

今後の展望としては,本研究の成果をより実用的なプログラムに適用することが挙げられる．

５．主な発表論文等

（研究代表者には下線）

〔雑誌論文〕（計 3 件）

(1)Akimasa Yoshida, Akira Kamiyama, Hiroki Oka,
A Task-Driven Parallel Code Generation Scheme for Coarse Grain Parallelization on Android Platform,
情報処理学会論文誌 Advanced Computing Systems, 査読有, Vol.10, 2017, pp.1-12.
https://ipsj.ixsq.nii.ac.jp/ej/

(2)Akimasa Yoshida, Akira Kamiyama, Hiroki Oka,
A Task-driven Parallel Code Generation Scheme for Coarse Grain Parallelization on Android Platform
情報処理学会 Journal of Information Processing, 査読有, Vol.25, 2017, pp.426-437.
DOI:10.2197/ipsjjip.25.426

(3)岡　宏樹, 吉田明正,
メニーコア上でのローカルタスク協調実行を伴うタスク駆動型粗粒度並列処理,
情報処理学会論文誌 Advanced Computing Systems, 査読有, Vol.12, 2019, 印刷中.
https://ipsj.ixsq.nii.ac.jp/ej/

〔学会発表〕（計 6 件）

(1)岡　宏樹, 吉田明正,
Android 搭載ヘテロジニアスマルチコアにおける Fork/Join Framework を用いた粗粒度並列処理,
情報処理学会第 15 回情報科学技術フォーラム，C-033，2016 年 09 月 07 日，富山.

(2)橋本里菜, 吉田明正,
マルチコア上での Fork/Join Framework を用いた再帰プログラムの粗粒度並列処理,
情報処理学会第 15 回情報科学技術フォーラム，C-034，2016 年 09 月 07 日，富山.

(3)岡　宏樹, 吉田明正,
メニーコア上での粗粒度並列処理におけるコードコンパクション,
情報処理学会システムアーキテクチャ研究会研究報告，2017-ARC-227-38，2017 年 7 月 28 日，秋田.

(4)村上　茜, 吉田明正,
マルチコア上での Java インスタンスメソッドのタスク駆動実行による粗粒度並列処理
情報処理学会第 16 回情報科学技術フォーラム，C-013，2017 年 9 月 14 日，東京.

(5)岡　宏樹, 吉田明正,
メニーコア上でのローカルタスク協調実行を伴う Java プログラムのタスク駆動型粗粒度並列処理,
情報処理学会システムアーキテクチャ研究会研究報告，2018-ARC-232-24，2018 年 8 月 1 日，熊本.

(6)山端大揮, 岡　宏樹, 吉田明正,
POWER8 上でのローカルタスク協調実行を伴うタスク駆動型粗粒度並列処理,
情報処理学会第 81 回全国大会，5L-02，2019 年 3 月 15 日，福岡.

〔その他〕

ホームページ
明治大学 総合数理学部・先端数理科学研究科 吉田研究室 研究内容
http://www.isc.meiji.ac.jp/~yoshidalab/research.html

６．研究組織

(1)研究分担者

なし

(2)研究協力者

なし

※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、研究者個人に帰属されます。