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　６ 年　 ６ 月 ２２ 日現在

機関番号：１３１０１
研究種目：基盤研究(C)（一般）
研究期間：2019～2023
課題番号：１９Ｋ１１８９３

研究課題名（和文）マルチコア並列計算に対応した関数型言語処理系の実現

研究課題名（英文）Development of Functional Language Capable of Parallel Computation on Multicore Processors

研究代表者
上野　雄大（Katsuhiro, Ueno）

新潟大学・自然科学系・准教授

研究者番号：６０５５１５５４

交付決定額（研究期間全体）：（直接経費）　　3,300,000円

研究成果の概要（和文）：C言語やFORTRANに匹敵する並列計算性能を安定して発揮できる関数型言語処理系を実現することを目指して研究を行い、以下2点の主要な成果を得た。
（１）投入されたCPUコア数と同じ並列度を持つ自動的メモリ管理アルゴリズムを開発した。また、その正しさを形式モデルを通じて証明した。
（２）そのアルゴリズムの最適化された実装をSML#コンパイラに搭載し、全世界に無償で公開した。研究代表者の知る限り、SML#は、実用的なフルスケールのStandard ML処理系の中では、世界で初めてマルチコアCPUの理論上の並列計算性能を引き出すことに成功したコンパイラである。

研究成果の学術的意義や社会的意義
本研究成果の学術的な意義は，自動的メモリ管理方式の改善により、今日広く普及しているマルチコアCPUを用いて高性能な並列計算を行うことができる、フルスケールの関数型言語コンパイラを実現できたことにある。関数型言語は並行・並列処理の記述に向いていると言われており、並行・並列計算モデルや自動並列化に関する研究が古くから行われている一方で、理論上の並列計算性能を実現することができる関数型言語処理系は、いくつかの実証実験的な処理系を除いて、これまで存在しなかった。本研究成果により、高性能な並列計算が可能な関数型言語処理系が日本発で全世界に無償で提供されたことの社会的意義は大きい。

研究成果の概要（英文）：We conducted research on theory and implementation of a functional programming language that is capable to parallel computation on multicore processors similarly to C and FORTRAN. Major achievements includes the following:
(1) We have developed a fully concurrent and parallel garbage collection (GC) algorithm that does not stop any thread. Its amount of parallelism is automatically increases when the number of threads increases. We proved the correctness of the algorithm through a formal model.
(2) We have optimized and implemented the GC algorithm and equip the SML# compiler with it. The implementation is widely available on Internet as an open-source software. To the best of our knowledge, SML# is a world first practical full-scale Standard ML compiler that achieves full-scale parallel performance of multicore processors.

研究分野：プログラミング言語

キーワード：関数型言語　コンパイラ　ガベージコレクション　動的型付け　SML#

様　式　Ｃ－１９、Ｆ－１９－１（共通）

１．研究開始当初の背景

マルチコアCPUはすでに広く普及しており、大型計算機から組み込み機器に至るまで、計算能力を有するあらゆるデバイスに搭載されている。これら多数のコアを持つプロセッサを最大効率で駆動するための並列プログラミング技術は、今日では必須である。その一方で、高効率な並列プログラミングを行うための基盤技術は、マルチコアCPUが普及し高性能化している現状に対して、十分に成熟しているとは言えない。高い計算効率が求められる分野では、古くから用いられている C/C++言語や FORTRAN を用いて、オペレーティングシステムが提供するスレッドを通じてCPUとメモリを緻密に制御する、職人芸的なプログラミングが主流である。

関数型プログラミング言語による並列計算は、この現状を打破し、並列プログラミングの難しさを大きく改善する可能性がある技術のひとつである。関数型言語と並列計算の相性が良いことは古くから主張されており、関数型言語をベースとした並列計算モデルや自動並列化の研究が80年代から綿々と続けられている。その研究成果のいくつかは、主要な関数型言語処理系にすでに実装されている。しかしながら、それらの実装はマルチコアCPUに対応しておらず、期待される計算性能は全く得られていない。もし、高性能な並列処理が現実に可能な関数型言語処理系を実現することができるならば、現代的な並列計算を含むソフトウェアの生産性を飛躍的に向上させる可能性がある。さらに、関数型言語の基礎研究で培われてきた並列計算モデルや自動並列化理論と現実の並列計算能力が結び付き、スーパーコンピューティングと関数型言語理論の複合領域の創生などにも繋がり得ると期待できる。

２．研究の目的

本研究の一般的な目的は、マルチコアCPUの並列性能を引き出すことが可能な、フルスケールの関数型言語処理系を実現することである。関数型言語がマルチコアCPUに十分に対応していない大きな理由のひとつは、メモリの使い方にある。関数型言語の基礎理論が想定する実行モデルには、メモリなどの記憶領域は陽に現れない。代わりに、メモリは何らかの方法で自動的に供給・管理されるものとされている。この理論のとおりにメモリのことを考えなければ、関数型言語のプログラムに含まれるほとんどの部分は、互いに独立に評価することが可能である。このことを根拠に、関数型言語は並列計算に向いていると言われている。一方、関数型言語の実装においては、この自動的なメモリ管理は、ガベージコレクション（GC）と呼ばれる処理によって、メモリの割り付けと再利用を行うことで実現されている。GC はプログラムが使用するメモリ全体を管理するため、プログラムのあらゆる部分に影響を及ぼすと同時に、理論上は独立に計算できる各部分に暗黙的な依存関係を与えてしまう。一般に GC アルゴリズムは、大域的な動機を本質的に含む逐次的な処理として設計される。しかも、プログラムの実行の流れはメモリの使い方と相互に関連するため、スレッドと GC は相互作用を持つ。このため、現行の関数型言語の処理系の多くは、スレッドを含む並行・並列計算機能を、処理系ごとに独自に実装している。このようにメモリとスレッド管理が独自実装によって抽象化された結果として、ユーザープログラムによる並列性の制御は極めて難しくなっている。

本研究では、この問題の解決に向けて、関数型言語におけるメモリの使い方に着目し、現実のマルチコア計算機を高い性能で制御できるようにメモリを自動管理する方式を開発することで、マルチコアCPUの並列性能を引き出す関数型言語の実現を目指す。この達成のために本研究で取り組む具体的な学術的課題は以下の2点である。

(1) 関数型言語の暗黙的ながら高いメモリ要求に答えることができる性能を有する並行・並列GCアルゴリズムの構築
(2) 暗黙的なメモリ操作や大域的同期を含まないコードを生成するコンパイル方式の確立

もしこれらの課題を解決することができたならば、研究代表者らがこれまでに研究開発を進め達成してきた関数型言語 SML#の諸機能をそれらと組み合わせ、C 言語で書かれた細粒度スレッドライブラリとの緻密な連携をとることで、C 言語の並列計算性能に匹敵する並列計算性能を有する関数型言語処理系を実現できるはずである。

３．研究の方法

本研究では、上述した 2 点の課題を軸として、研究代表者らがこれまでに研究開発してきたSML#の基礎をなす理論、コンパイル方式、および実装方式をさらに洗練することで、並列計算機の性能を引き出す関数型言語の基礎理論と実装方式を探求する。各課題に対する研究の具体的な方法は以下のとおりである。

第一の課題、すなわち並行・並列 GC アルゴリズムの構築にあたっては、研究代表者らがすでに達成済みの、スレッドを止めない並行 GC アルゴリズムを基礎とする。このアルゴリズムは並列ではなく、また良い性質を残したままこのアルゴリズムを並列化できるかは定かでない。本研究ではこのアルゴリズムを並行かつ並列なものに洗練することを試みる。

第二の課題、すなわち暗黙的なメモリ操作を含まないコードを生成するコンパイル方式の確立に向けては、すでに研究代表者らが自ら開発しその全体を掌握している SML#コンパイラが

生成するコードの分析から始める。なかでも短命で局所的にしか使用されない中間データに着目する。それらは理論上は大域的なメモリプールに割り付けられる必要はないはずである。それらが大域的に割り付けられることが、暗黙的な大域的同期を生み、並列性能を大きく損ねていると推測される。本研究では SML#の分析を通じて改良点を洗い出し、その結果を基礎理論に反映することで、単なる実装上の工夫にとどまらない、抜本的かつ普遍的な解決を根らう。

いずれの課題についても、その基礎研究で得られた知見を SML#コンパイラの実装に反映することで、実践的な妥当性を含めて検証する。また実践によって得られた分析を基礎研究にフィードバックすることで、基礎と実践の両面から、マルチコア CPU に対応した関数型言語の実現を目指す。

本研究の成果は、オープンソフトウェアである SML#コンパイラの一部として、世界中の誰もが無償で利用可能な形態で社会還元される。

４．研究成果

第一の課題であった並行・並列 GC アルゴリズムの開発に成功し、投入した CPU コア数に対してほぼ期待通りの列計算性能を有する関数型言語処理系の実現を達成することができた。この GC アルゴリズムは、研究代表者らがこれまでに研究開発した並行 GC アルゴリズムの考え方を継承して開発した一方で、最終的な出来上がりはこれまでのアルゴリズムとは大きく異なるものとなった。

本研究成果の並行・並列 GC アルゴリズムには独自の特徴が大きく 2 つある。ひとつは、これまでは不可分なモジュールとして言語ランタイムに実装されていたスレッドスケジューラとガベージコレクタを可分とし、MassiveThreads など既存の細粒度スレッドライブラリを関数型言語からそのまま直接に利用可能にしたことである。この結果、GC モジュールの独立性が高まり言語ランタイムの開発効率が改善されたと同時に、オペレーティングシステム研究の専門家が開発した最先端の細粒度スレッドスケジューラの恩恵を常に享受できるようになった。これは最高の並列計算性能を達成・維持する上で重要な性質である。

もうひとつは、メモリ管理だけを行う専用のスレッドやルーチンを用意せず、すべての CPU コアはユーザープログラムを実行することに加え、メモリ管理の仕事をときどき非同期に行うことである。この構成により、このアルゴリズムは自然に並行なアルゴリズムとなり、しかも起動されたスレッド数（投入された CPU コア数）と同じ並列度で GC が並列処理されることになる。この構成を実現するため、GC アルゴリズムはユーザープログラムの実行中にときどき呼び出されるプロシージャの集まりとして設計された。各 CPU コア上で動いているユーザープログラムは、非同期的に互いにメッセージを交換することで（Doligez-Leroy-Gontier スタイルのハンドシェーキング）大域的な GC を行う。この並行・並列で非同期的な GC アルゴリズムの正しさは、ユーザープログラムのメモリ操作に関する振る舞いの数理的モデルを構築し、そのモデルにおいてオブジェクト集合が持つ性質を表す命題を設定することを通じて、最終的に定理として証明された。

この並行・並列 GC アルゴリズムは、学術的には、メモリ管理分野で著名な国際会議 ISMM 2022 に採択され発表された。また、研究成果の社会還元の点においては、SML#コンパイラの 3.6 版に実装され、オープンソースソフトウェアとして誰でも無償で利用可能な形で公開された。図 1 に示すとおり、本研究成果により、SML# 3.6 は、過去の SML#を含むどの主要な関数型言語よりも大幅な並列計算性能の向上を達成した。

第二の課題であった暗黙的なメモリ操作を含まないコード生成方式の実現については、残念ながら本研究期間内に達成するまでには至らなかった。その主要な要因は、この問題を議論・分析するために必要な、関数型言語のメモリやレジスタの使い方に関しての振る舞いを明確にした基礎理論（操作的意味論）の欠如にあった。一方、本研究期間に、関数型言語の新たな操作的意味の構成法や今後の展開に関する着想を得ることができた。この着想は関数型言語のための GC に代わるメモリ管理機構の研究と統合して、本研究期間後も引き続き研究を展開し推進する予定である。

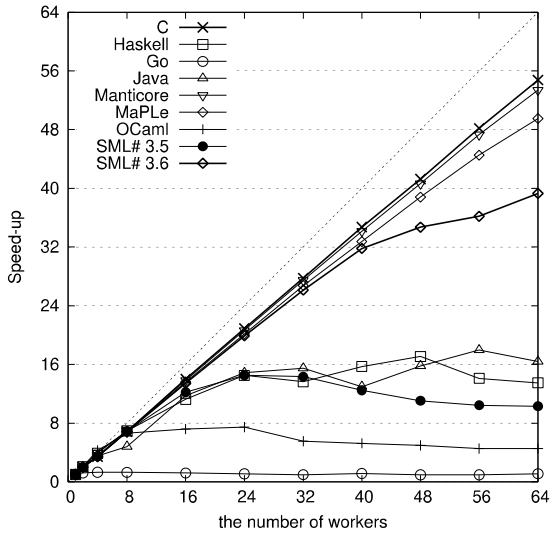

図 1: SML#および並列計算機能を持つ関数型言語における、CPU コア数の増加に対する実装速度の向上比率。ベンチマークには 14-queen 問題のソルバを用いた。対角線（点線）に近いほど良い。「C」はこの計測条件における理想。「SML#3.6」が本研究成果により達成された性能である。SML#3.6 は Manticore や MaPLe などの実験的な関数型言語にはやや劣るものの Haskell や OCaml などの主要なフルスケールの関数型言語には大きく勝っている。

５．主な発表論文等

〔雑誌論文〕　計0件

〔学会発表〕　計12件（うち招待講演　1件／うち国際学会　3件）

１．発表者名 上野雄大
２．発表標題 Minissg: 小さく軽量で規約のない静的Webサイトジェネレータ
３．学会等名 日本ソフトウェア科学会第40回大会
４．発表年 2023年

１．発表者名 佐藤 季樹, 上野 雄大
２．発表標題 型別名を保存する型推論アルゴリズム
３．学会等名 第26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（PPL 2024）
４．発表年 2024年

１．発表者名 Katsuhiro Ueno, Atsushi Ohori
２．発表標題 Concurrent and parallel garbage collection for lightweight threads on multicore processors
３．学会等名 ISMM '22: ACM SIGPLAN International Symposium on Memory Management（国際学会）
４．発表年 2022年

１．発表者名 Katsuhiro Ueno
２．発表標題 SML#: Toward the ideal interoperability between languages and systems
３．学会等名 MoreVMs'23（招待講演）（国際学会）
４．発表年 2023年

１．発表者名
上野雄大，石垣凌

２．発表標題
プラレールによるプログラミングの可能性

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 PPL 2023

４．発表年
2023年

１．発表者名
藤野遼河，上野雄大

２．発表標題
プログラムの意味解析技術の音楽の意味解析への応用に向けて

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 PPL 2023

４．発表年
2023年

１．発表者名
山上隼司，菊池健太郎，上野雄大，大堀淳

２．発表標題
アトム変数を用いた名目単一化の実装

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
Atsushi Ohori, Katsuhiro Ueno

２．発表標題
A compilation method for dynamic typing in ML

３．学会等名
The 19th Asian Symposium on Programming Languages and Systems（国際学会）

４．発表年
2021年

１．発表者名
上野雄大

２．発表標題
自然なデータ表現を持つ多相型言語のLLVM IRへのコンパイル方式

３．学会等名
日本ソフトウェア科学会第37回大会

４．発表年
2020年

１．発表者名
上野雄大，大堀淳

２．発表標題
SML#の並列処理機能とその性能

３．学会等名
日本ソフトウェア科学会第37回大会

４．発表年
2020年

１．発表者名
大堀淳，上野雄大

２．発表標題
関係代数を基礎とするプログラムに現れる名前解析システム

３．学会等名
日本ソフトウェア科学会第37回大会

４．発表年
2020年

１．発表者名
大堀淳，上野雄大，高城光平

２．発表標題
外部データの解釈を文脈ごとに与える動的型付け機構

３．学会等名
日本ソフトウェア科学会第37回大会

４．発表年
2020年

〔図書〕　計1件

１．著者名 大堀淳，上野雄大	４．発行年 2021年
２．出版社 共立出版	５．総ページ数 242
３．書名 SML#で始める実践MLプログラミング	

〔産業財産権〕

〔その他〕

SML#プロジェクト
https://smlsharp.github.io

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

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