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　３１　日現在

機関番号：　１２６０１
研究種目：　挑戦的萌芽研究
研究期間：　2015～2017
課題番号：　１５Ｋ１２０３３

研究課題名（和文）自らを進化させ未知の計算環境に適応するソフトウェア自動チューニング機構方式の研究

研究課題名（英文）Research on Software Autotuning Mechanism that evolves to unknown computing environments

研究代表者

須田　礼仁（SUDA, Reiji）

東京大学・大学院情報理工学系研究科・教授

研究者番号：４０２５１３９２

交付決定額（研究期間全体）：（直接経費）　　2,700,000円

研究成果の概要（和文）：自動チューニングは、ソフトウェアにあらかじめ可変性を仕込み、この可変性をソフトウェア自身に調整させて、様々な計算環境で良好な実行性能を目指す。本研究では、既存のプログラムに対して、事後的に可変性と調整機能を組み込むことにより、新しい計算環境や新しい高性能手法が登場しても、それを既存のプログラムに組込み自動チューニングができる仕組みを目指して研究した。
我々はチームメンバーが開発してきた Xevolverというコード変換システムを活用することで、自動チューニングを想定していないプログラムに可変性と自動チューニング機構を組み込む手法を明らかにした。ただし原プログラムの分析の必要性が明らかになった。

研究成果の概要（英文）：Autotuning is a technology aiming to attain good execution performance on various computational environments by preparing variabilities within software and letting the software itself control the variabilities. In this research, we aimed to develop methodology to infuse variabilities and control mechanism which are unintended or even unknown to existing codes, to attain autotuning even if novel computational environments and novel variabilities become newly known.
We have shown that, by using Xevolver, which is developed by our team members, we can infuse variabilities and autotuning mechanisms which is unknown to the original code. However, it became clear that we need to fully analyze the original code before applying such infusions.

研究分野：　高性能・並列数値計算

キーワード：　自動チューニング　高性能計算　コード変換　最適化

１．研究開始当初の背景

自動チューニングは、ソフトウェアにあらかじめ可変性を仕込んでおき、この可変性をソフトウェア自身に調整させて、多様な計算環境において良好な実行性能を達成することを目指す技術的パラダイムである。ここで可変性には、単純な変数の値の設定から、関数やサブルーチンの選択、ループ変換などの実行順序の変更、アルゴリズムやデータ構造の変更、さらには、並列化や GPU 化などのプラットフォーム特有のコーディングまで多様なものが考えられる。自動チューニング機構を有するソフトウェアは、新しい計算環境においても、優れた性能を自動的に実現するものを選択できるのであるが、それはあくまであらかじめ準備された可変性の範囲内に限られる。しかし GPU や FPGA など、従来の CPU とは全く異なるアーキテクチャが次々と出てきており、そのような新たなアーキテクチャの性能を最大限に引き出すためには、それ以前には知られていなかったような可変性を組み込む必要がある。そのためには自動チューニングが組み込まれたソフトウェアであっても、新しい計算機に対してソフトウェアを作り直すということが必要ということになってしまっていた。

２．研究の目的

上記の問題に対し、本研究では、ソフトウェアに新たな可変性とそれを調整する機能を事後的に、自動的に取り込む新たな自動チューニング方式を目指した。すなわち、これにより未知のハードウェア、未知の計算環境においても、自動的に良好な性能を達成できることを目指した。

３．研究の方法

上記の目的を達成するため、まず従来の自動チューニングの方式を見直すことから始める計画を立てた。

これに続いて、滝沢らが開発してきたユーザ定義可能かつカスタマイズ可能なコード変換フレームワークである Xevolver を用いて、片桐らが開発してきた自動チューニング記述言語 ABCLibScript および ppOpen-AT を参考にし、高性能な可変性と自動チューニング機構を原プログラムから分離し、事後的に付加する仕組みの実現を目指すこととした。これにより、上記で分析した従来の自動チューニングの方式のうちいくつかの事例について、原プログラムに対して、可変性、自動チューニング機構を事後的に付加するプログラム再構成を行うこととした。ここから、提案するパラダイムを実現する設計方式および構築方式を探り当てることを目指した。

また、これに並行して、高性能なソフトウェア実現のためのプログラム可変性の事例を収集し、それを事後的に付加する Xevolver の変換ルールの収集を目指した。これを参照することにより、提案するパラダイムの実効性がより高くなることを期待した。

４．研究成果

まず最初に、我々の従来研究において Xevolver で生成したコードについて、無駄を削減し、高性能を達成するようなコード変換を、Xevolver で記述することを行った。これにより、プログラムを高性能化する変換を事後的に付加することができることを実証した（明らかに性能が改善するタイプの変換であったため、自動チューニング機能は付加しなかった）。

しかしここで課題に直面することになった。上記の実装では数種類の最適化を実装するのに、250 行ほどの Xevolver の変換ルールの記述が必要であった。この事例では、原プログラムが自動生成されたコードであったため、非常に定型的な書き方がなされており、それを仮定することにより、一般的なコードに対する変換ルールよりもずっと単純なもので済んでいるのであった。人間が書いたコードでは、記述の自由度が高く、人それぞれが自分の癖を持って記述することになっているであろうから、それに対応する変換ルールは、各段に複雑になることが危惧されることになった。他方で、複数の人間が協働的にコーディングする際には、一定のコーディングルールを用いて書き方に制約を設けることもよく行われている。このような場合には、そのコーディングルールを前提とすることで、多少変換ルールの記述が簡単になる可能性が期待される。なお、グループでコーディングする際に詳細なコメントを付けることもよく行われるが、これは我々の目的には何の役にも立たず、むしろ複雑なコメントの場合（Fortran で継続行の途中にコメント行を入れるなど）には邪魔になる可能性もある。

次に、事前に自動チューニング機構を入れることを想定されていないソフトウェアとして、物理学者が開発したテンソル計算の実用コードをお借りして、これに Xevolver で可変性を組み込むことに取り組んだ。コードは比較的素直に記述されているのであるが、Fortran コンパイラに先立ち cpp を通すコーディングとなっていたため、そのままでは Xevolver に入力することができなかった。そこで当該コードで使われている cpp の機能を Xevolver で再実装し、手動でオリジナルコードを修正することで、Xevolver を通すことができるようにした。その後は順調に進み、いくつかのループの変換を組み込んで、最大で 20% の性能向上を実現する可変性を組み込むことができた。

ここでも、想定外の課題が見つかった。すなわち、Fortran であっても cpp などのプリプロセッサを通していたり、各種コンパイラの独自拡張などに基づく実装が実際には多数あるということであった。Fortran のコ

ンパイラごとの独自仕様の問題は大変古い課題であるが、ここでもそれが再燃したことになる。我々の提案する手法を用いるためには、(1) きわめて標準的な Fortran の書き方に準拠してもらうか、そのように書き直してもらう、(2) そうでない記述が必要であれば、標準的な Fortran から必要な記述を変換により生成してもらう、(3) Xevolver で同等の機能を提供する、などの準備が必要であるということが明らかになった。Xevolver は極めて広範なコード変換を独自に定義することができるので、Xevolver で処理できる形でまず記述してもらい、コンパイラ独自拡張は変換後に適用するような形が、我々にとっては都合がよい。もしくは、Xevolver のパーサを非常に緩いものにして、標準 Fortran では受け付けないような構文も受け付けられるようにするという方法も考えられる。

ここまでは可変性を事後的に組み込む研究であったが、最後に自動チューニング機構を事後的に組み込む研究を行った。様々な手法について検討を行ったが、結論的には、次のような 3 ステップを踏んで自動チューニング機構を組み込むのがよいであろうという結論になった。第 1 ステップでは、原プログラムに対して、Xevolver を用いて、どこでどのような自動チューニングを行うかを示す指示行を挿入する。Xevolver ではコード変換だけではなく、(Xevolver が認識するフォーマットで書かれた) 指示行についても追加・削除・変換などの操作が自由にできるため、これは比較的簡単に実現できる。第 2 ステップでは、自動チューニングのための可変性の実装と、性能測定やパラメタ調整の機能を、やはり Xevolver の指示行を用いて記述する。ここでは、Xevolver の機能を活用することで、変換ルールそのものは記述せず、自動チューニング機構のために付加する実行命令、サブルーチン、データ構造、モジュールなどを、通常の Fortran コードとほぼ同等の形で、指示行とともに記述すればよい。最後の第 3 ステップでは、第 2 ステップで記述した自動チューニング機構を第 1 ステップで指示行を組み込んだ原プログラムに適用する。ここでは、Xevolver の柔軟な機能が役に立つ。具体的にはこの第 3 ステップは 2 つのサブステップからなり、第 1 サブステップでは、第 2 ステップで記述した指示行つきのコードから、第 1 ステップで構成したプログラムを変換するルールを生成する。第 2 サブステップではこのルールを実際に適用することにより、自動チューニング機構を原プログラムに組み込むことができる。本研究ではこれらの変換を実現するためのインタフェースとして、第 1 ステップおよび第 2 ステップの指示行の仕様を定義し、また第 3 ステップの変換ルールの設計まで行った。ただしここまでが限界で、これらを実装して自動チューニング機構を事後的に組み込むところまでは達成できず、今後の課題として残った。

これらの研究に加えて、自動チューニングの新たな展開を目指した研究成果を挙げた。第 1 に、近年大きな注目を集めている深層学習を自動チューニングに適用する研究を行った。自動チューニングの重要なテーマに疎行列データ構造の選択がある。疎行列は非零要素が不規則に表れることが多く、局所性が低下し性能劣化につながる。疎行列を表現するデータ構造の工夫により、これを改善することができることがある。本研究では、疎行列の非零パターンを小規模な画像に変換し、特徴量を色情報として埋め込んだ。これを深層学習の入力とした。また、疎行列のデータが深層学習には十分な量にならないので、疎行列に修正を加えて新たなデータを生成し、これをあわせて学習に用いる工夫を加えた。そして、疎行列用のデータ構造や、コンパイラオプションを深層学習で学習させ、よい性能を示す選択が学習できることを示した。この研究は、これまで難しいとされてきた疎行列の非零パターンに依存するチューニングを実現し、しかも高い性能を達成したところに大きな意義がある。

また逆に深層学習に自動チューニングを用いる研究も行った。深層学習では深いニューラルネットワークを用いるが、深いニューラルネットワークほど多様な構造を実装することができ、その選択が機械学習の性能を大きく左右するにも関わらず、ニューラルネットワークの構造の選択は研究者の経験と勘に頼る「匠の技」であった。これに対して本研究ではニューラルネットワークの構造パラメタを自動チューニングで最適化し、良好な性能を達成する構造パラメタを同定できることを示した。

また、計算機アーキテクチャの変更に関わらず汎用的に実現できる自動チューニングとして動的負荷分散を取り上げて研究した。実行状況に対する判断のヒントをユーザが追加することにより、より的確で正しくかつ適応的な振る舞いが達成されることを実証した。また仕事量の見積もりを優先度または重みとして用いることで、ワーカ数 128 の場合を含め有効性を実証した。

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　11　件）

① Hiroshi Yoritaka, Ken Matsui, Masahiro Yasugi, Tasuku Hiraishi, Seiji Umatani, Preliminary Evaluations of Probabilistic Guards for a Work-Stealing Framework, Annual Meeting on Advanced Computing System

and Infrastructure (ACSI2016), 2016年1月18日、九州大学（福岡県福岡市）

② Cui Hang, Shoichi Hirasawa, Hiroyuki Takizawa, Hiroaki Kobayashi, A code selection mechanism using deep learning, MCSoC-16, 2016年9月21日、リヨン（フランス）

③ Hiroshi Yoritaka, Ken Matsui, Masahiro Yasugi, Tasuku Hiraishi, Seiji Umatani, Extending a Work-Stealing Framework with Probabilistic Guards, P2S22016, 2016年8月16日、フィラデルフィア（米国）

④ Takahiro Katagiri, Masaharu Matsumoto, Satoshi Ohshima, Auto-Tuning of Hierarchical Computations with ppOpen-AT, SIAM PP16, 2016年4月12日、パリ（フランス）

⑤ 須田礼仁、複合的自動チューニングのための数理とソフトウェア、第21回計算工学講演会、2016年5月31日、朱鷺メッセ（新潟県新潟市）

⑥ Reiji Suda, Embedded-DSL-Like code generation and optimization of Bayesian estimation routines with user-defined source-to-source code transformation framework Xevolver, International Workshop on Legacy HPC Application Migration, 2017年11月20日、ASPAM（青森県青森市）

⑦ 須田礼仁、複合的自動チューニングのための数理ライブラリの構築、第22回計算工学講演会、2017年5月31日、ソニックシティ（埼玉県大宮市）

⑧ 寄高啓司、八杉昌宏、平石拓、馬谷誠二、優先度並びに重みを用いたワークスティールフレームワークの性能改善、xSIG 2017, 2017年4月24日、虎ノ門ヒルズフォーラム（東京都）

⑨ Yuki Kawarabatake, Mulya Agung, Kazuhiro Komatsu, Ryusuke Egawa, Hiroyuki Takizawa, Use of Code Structural Features for Machine Learning to Predict Effective Optimization, iWAPT 2018, accepted.

⑩ 滝沢寛之、崔航、平澤将一、機械学習によるコード最適化の可能性、第22回計算工学講演会、2017年5月31日、ソニックシティ（埼玉県大宮市）

⑪ Zhen Wang, Ryusuke Egawa, Reiji Suda, Hiroyuki Takizawa, Auto-tuning of Hyperparameters of Machine Learning Models, HPC Aisa 2018, 2018年1月29日、UDXビル（東京都）

〔図書〕（計 0 件）

〔産業財産権〕
なし

〔その他〕
国際研究集会（共催）
- International Workshop on Automatic Performance Tuning, 2016年5月27日、シカゴ（米国）

６．研究組織
(1)研究代表者
須田 礼仁（SUDA, Reiji）
東京大学・大学院情報理工学系研究科・教授
研究者番号：40251392

(2)研究分担者
滝沢 寛之（TAKIZAWA, Hiroyuki）
東北大学・サイバーサイエンスセンター・教授
研究者番号：70323996

(3)連携研究者
八杉 昌宏（YASUGI, Masahiro）
九州工業大学・大学院情報工学研究院・教授
研究者番号：30273759

(4)研究協力者
片桐 孝洋（KATAGIRI, Takahiro）
名古屋大学・情報基盤センター・教授
研究者番号：40345434