科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　１１　日現在

機関番号：　１２６０１
研究種目：　若手研究(B)
研究期間：　2012～2013
課題番号：　２４７０００２４

研究課題名（和文）ＧＰＵプログラム最適化のための指示文を用いた自動チューニング機構の開発

研究課題名（英文）Development of Auto-Tuning framework using pragmas for optimization of GPU program

研究代表者
大島　聡史（OHSHIMA, Satoshi）

東京大学・情報基盤センター・助教

研究者番号：４０５７００８１

交付決定額（研究期間全体）：（直接経費）　　3,000,000円、（間接経費）　　900,000円

研究成果の概要（和文）：GPUプログラムの最適化における難しさや手間を軽減するための研究を実施した。特に利用者が容易に扱えるように指示文やわかりやすいAPI関数を用いて利用できるようにすることを念頭に置いて研究と開発を行った。具体的には、主に
1.GPU上で実行されるプログラム部（GPUカーネル）の性能評価や評価結果を活用しやすくすることを目指して、プロファイラ情報を活用するための機構を提案し、実装を行った。
2. CPU-GPU間のデータ移動に着目して計算結果の正当性確認やテスト実行コスト削減を行う機構について検討し、実装を行った。

研究成果の概要（英文）：This project researched to reduce the difficulty and labor of optimization of GPU program. Especially, in order to make the product useful for users, this project attached a high value to achieve the purpose by using simple pragmas and API functions.
1. In order to make performance evaluation and utilization of the result of the performance evaluation, th is project proposed and developed a framework to utilize some information of the profiler.
2. This project studied and developed a framework which confirm the computation of GPU and reduce the cost of testing by focusing on the data transfer between CPU and GPU.

研究分野：　総合領域

科研費の分科・細目：　情報学、ソフトウェア

キーワード：　GPGPU　自動チューニング

１．研究開始当初の背景

数値シミュレーションなど計算科学の分野においては様々な並列計算ハードウェアが利用されている。現在最も広く普及している並列計算ハードウェアはマルチコア CPU であり、4 から 8 程度の計算コアを搭載したマルチコア CPU が多く用いられている。マルチコア CPU の市場は Intel (Xeon)、AMD (Opteron)、IBM (Power)の 3 陣営が特に大きなシェアを持っており、「京」コンピュータなどを持つ富士通やベクトルプロセッサの技術を持つ NEC などが残る若干のシェアを分け合っている。

また既存のマルチコア CPU よりも多数の計算コアを備えたメニーコアプロセッサについても注目が集まっている。具体的には Intel 社が Knights シリーズと呼ばれるメニーコアプロセッサを一部の研究機関などに提供しつつある。（その後、2012 年末から 2013 年初頭にかけて Xeon Phi シリーズとして発表・発売され、2014 年にはある程度普及が進んでいる。）

一方で近年は GPGPU（GPU コンピューティング）も重要な役割を担っている。GPGPU は画像処理用に開発されたハードウェアを汎用計算に活用する技術であり、高い並列度の並列処理による高い演算性能や高性能なメモリによる高いメモリ性能を活用することで、対象問題（アプリケーション）によっては既存の CPU よりも高い性能を得ることができる。GPU を搭載した計算機、いわゆる GPU ワークステーションは多くの大学や研究機関等にて活用されている。GPU をスーパーコンピュータの計算ユニットとして活用するいわゆる「GPU スパコン」も国内外にて普及しており、例えば 2012 年 6 月の TOP500 リストでは全 500 システム中の約 1 割が GPU を計算ユニットとして搭載している。また、主要な GPU ベンダーの 1 つである米国 NVIDIA 社が 2020 年頃に達成が見込まれている ExaFLOPS 級のスーパーコンピュータを GPU 技術を用いて構成しようとしているなど、GPU への期待は大きい。

GPU を活用するプログラムの作成（GPU プログラミング）には CUDA(CUDA C)が多く用いられている。CUDA は NVIDIA 社の GPU 向けの開発環境であるため利用できる対象が限定されるものの、言語仕様上は C/C++とほとんど変わらない（いくつかの指示子等が加わるのみ）こと、新アーキテクチャの投入サイクルが速い GPU の最新機能を十分に利用できること、チュートリアルや最適化方法などの資料が多く揃っていること、既存の CPU 向けのプログラムと共存させやすい（GPU 上で実行させたい部位のみを CUDA 化すれば良い）ことなどから広く普及している。

GPU の普及が進む一方で、GPU プログラミングの難しさや手間が問題となっている。確かに CUDA は現実的なコストで高性能なアプリケーションを作成可能とした。CUDA を用いたプログラム最適化の方法や方針についてもある程度の資料が揃っている。しかしながら、プログラム最適化における考え方（最適化の方針やアルゴリズム）や最適化パラメタが CUDA と既存の CPU では異なるため、実装や最適化の手間や難しさが大きいという問題が生じている。より高い性能をより容易に利用できるようにするためには、最適化プログラミングをサポートする工夫や仕組みが必要である。

一方、プログラムの最適化や最適化手法の選択を効果的に行うための技術としてソフトウェア自動チューニング(AT)の研究が進められている。AT はソフトウェアに可変性を取り入れることで既知の情報を用いた人力による最適化では困難な最適化を達成するものである。AT をうまく利用すれば、利用する計算機の特徴（アーキテクチャ）や実行する対象問題の特徴、入力データの特徴などに合わせた最適なプログラム構成を行い、より良い性能を得られる可能性を高めることができる。そのため、複雑化する並列計算ハードウェアを有効に活用するための技術として、近年、AT への注目は国内外を問わず非常に高まっている。

２．研究の目的

プログラム最適化を容易にし、また高い性能を得られるようにする方法としては、例えば、プログラミング言語やモデル自体を改良すること、開発フレームワークを用いること、共通機能をライブラリ化することなど様々なアプローチが存在する。それぞれに利用のしやすさ、学習コスト、汎用性、継続性などに一長一短の特徴があり、全てを満たすような完璧な手法を開発することは非常に難しい。

本研究で着目するのは指示文を用いたプログラミング環境である。すなわち、既存のプログラムコードに対して情報を付加し、その情報に基づいてトランスレータにてソースコードの付加や改変を行ったり、コンパイラの挙動を変化させたりするものである。

既存の指示文を用いたプログラミング環境としては OpenMP が広く知られている。OpenMP は指示文により指定されたプログラム部位をスレッドにより並列実行するものであり、主にループ処理のスレッド並列化に用いられている。具体的な OpenMP 処理系の実装はコンパイラごとに異なるものの、例えば以下のような処理系によってプログラムの並列化を行うことができる

- トランスレータ（ソースコード書き換え機構）によって、並列化指示子の挿入された部位にスレッドの生成や管理を行うコードを挿入する

- さらに、ループの並列化が指示されているならば、対象のループ構造を書き換え、ループを分割して各スレッドに割り当てるための処理を挿入する

指示文を用いたプログラミングの利点としては、（言語仕様にもよるが）指示文を無視すれば既存のプログラムとして利用できること、段階的なプログラムの移行がしやすいこと（元となるプログラムを大きく書き換えずに少しずつ適用できること）などがあげられる。

そこで本研究では GPU 向けの指示文を用いた自動チューニング機能付きプログラミング環境の開発を行う。ただし、既存の CPU 向けプログラムに対して指示文を加えて GPU 化を行うのではなく、GPU プログラムの細かなパラメタ調整やアルゴリズム選択の手間を削減することを主眼として、GPU 向けに書かれたプログラムに対して指示文を加えるものとする。

本研究では主なユーザ層（対象とするユーザの要求）として以下を想定する：

1．既存の CPU 向けアプリケーションを GPU 化するうえで、主要な計算カーネルを GPU 向けに書き換える程度のプログラム変更は受け入れる（CPU 向けプログラムを自動的に GPU 化することは本研究の対象外である）
2．細かな最適化の手間と記述量を削減し、可読性を保ちたい

3．研究の方法

本研究の主な開発項目は以下の通りである：

1．GPU カーネル実行時に選択できる値の最適化を行う機能
2．最適化対象部のみを繰り返し実行することで全体の最適化時間を削減する機構
3．プロファイリング・解析による分析に基づく GPU に特化した性能評価機能・デバッグ機能・サジェスチョン機能

研究期間は 2 年であり、初年度にはプログラムの書き換え機能とチューニング時間削減の機能を中心に研究を実施する。次年度には、初年度の開発内容を改善するとともに、GPU に特化した性能評価機能・デバッグ機能・サジェスチョン機能の開発を実施する。基本的な実装の技術や性能評価対象プログラムについては、自らが参加している他のプロジェクト等で用いたものを参照・発展させて効率的に進める。

プログラムを書き換える機能については、自身が OpenMP の並列化対象部分を GPU 上で実行させる機構 OMPCUDA の開発で用いた既存の処理系（Omni OpenMP コンパイラ）を活用する。機能の齟齬等によりうまく行かない場合には、必要に応じて正規表現によるテキスト処理などをも活用して実現可能性と有効性の確認を行うことにする。

4．研究成果

研究を進めるにあたり、主な開発項目としてあげた 3 点をまとめ直した。すなわち、

1．プロファイル結果をプログラム中で再利用するためのフレームワーク（API と指示文）を提案・実装した
2．性能評価・サジェスチョン（アドバイス）のための枠組み（API と指示文）を提案した
3．最適化対象部のみを繰り返し実行し効率よく最適化する機構（API と指示文）を提案した

の 3 点である。以下、各項目について成果の概要を記す。

1 にあげたプロファイル結果の活用については、CUDA に備えられているプロファイラ関連の API をまとめ直してプログラム実行時に参照（取得）できるような API を実装した。またこれに対応する指示文を提案し、単純なテキスト処理により指示文から API へと変換する仕組みを実装した。提案した主な指示文とその意味は以下の通りである：

```
#pragma exprof begin 名前 (設定情報)
  名前付け、開始、情報出力先の指定など
#pragma exprof end 名前
  名前単位での終了
#pragma exprof exec default
  デフォルトの実行関数
#pragma exprof exec if(条件)
  条件による実行制御
#pragma exprof getval 名前 変数名
  プロファイル情報の取得
```

これらの指示文を用いて記述されたプログラムは、実装された処理系によってプロファイラ関連の API を扱うソースコードに変換される。さらに変換されたソースコードを CUDA コンパイラでコンパイルし、実装したライブラリとリンクすることで想定されているような機能が達成される仕組みとなっている。

以上の提案により、利用者（アプリケーションの最適化を行おうとするプログラマ）は単純な実行時間だけではなく、その実行時間を裏付ける要素にまでさかのぼってのチューニングが可能となった。例えばパラメタの最適化などを行う際に、性能を左右するパラメタ、例えば対象範囲内の命令実行数やキャッシュヒット具合などの「プロファイラを用いると知ることができる内部情報」を確認しながら動的な最適化を行うことが可能となる。ただし、この機能を十分に活かした最適化事例の作成は不十分であり、十分に高い有効性があることを明らかにするためには今後さらなる評価が必要であると言える。

2,3 にあげた枠組み・機構については、利用者の指示した部分における配列の値の取得や比較、該当範囲のプロファイラ結果の比較などを行う為の仕組み（API と指示文）を提案した。これらの機能は既存の CPU プログラムを GPU 化する際に GPU カーネルの最適化作業に集中しやすくするための補助的な機能を提供するものである。例えば、動作確認のためのデータセットのハンドリング（入力データの生成や出力結果の確認）を補助したり、該当部分以外の計算を省略したりといった機能を持つソースコードが簡単に生成される。これによりアプリケーションプログラマは、本来取り組むべき GPU カーネルの性能向上に集中しやすくなる、バグに気付きやすくなるといった効果が期待される。

以上の提案により、既存のプログラム最適化よりも容易で効果的な最適化が可能となると考えられる。現時点では「性能に悪影響があると容易に疑うことの可能ないくつかの事例」の検出のみに対応している。様々な事例に対応するにはサンプル数や事例が不足している。今後はさらに実験と評価を重ね、より良い効果の得られそうな状況を多く認識できるようにしていきたいと考えている。

本研究課題におけるアクティビティについて、学会発表[2]や[6]にて報告を行った。また要素技術としての GPU 向け最適化プログラミングおよび AT について[1],[3],[4],[5]にて発表した。

一方、当初の予定では開発期間中にソフトウェアの公開を行いフィードバックを得るなどして開発の精度を上げる予定であった。しかしながら、いくつかの技術的な困難にぶつかったため開発が遅延し、ソフトウェアの公開を行うことができなかった。今後改めて公開を行い成果の発展を行う予定である。

プログラムの書き換えを行う機能については、当初の予定では既存の OpenMP 処理系を元にして機能を実現する予定であったが、改めて詳細に調査した結果、実装されていた機能の都合上、今回の実装のベースとして使うのは非常に難しいことが判明した。そのため、正規表現を用いた単純な書き換えにより提案機能を実現するにとどまってしまった。その後の調査の結果、ROSE や Xevolver といったコンパイラ（フレームワーク）を用いることで当初想定していた機能が実現できる可能性があるとわかったが、本開発期間内には機能の実現まではできなかった。これについては今後の検討・実装課題とする。

プログラムに対してサジェスチョンを行う機能については、本プロジェクトの開発期間中に、CUDA を開発している NVIDIA 自身が高機能な GUI デバッガ・プロファイラ（nsight）の目玉となる機能の一つとして GPU カーネルのプロファイル情報を用いたサジェスチョン機能を提供しはじめた。そのため、今後は開発内容と NVIDIA 自身によって提供されている機能との比較や差別化についても考慮する必要がある。

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 6 件）

1. 大島聡史: GPU を用いた疎行列ベクトル積計算の最適化，日本応用数理学会 2012 年度年会，稚内全日空ホテル，8 月 28 日（火）-9 月 2 日（日）(2012).
2. 大島聡史: GPU プログラム最適化のための指示文を用いたチューニング機構の開発，第 4 回 自動チューニング技術の現状と応用に関するシンポジウム，東京大学 山上会館，12 月 25 日（火）(2012).
3. Satoshi Ohshima: Performance Evaluation of SpMV on Current Parallel Processors -- CPU vs GPU vs MIC --, HPC in Asia session at ISC'13, Congress Center Leipzig, Leipzig, Jun 17 (2013).
4. 大島聡史，金子勇，片桐孝洋: Xeon Phi における SpMV の性能評価，情報処理学会 研究報告(HPC-140)，7 月 24 日発行 (Vol.2013-HPC-140 No.33)，SWoPP2013 北九州，北九州国際会議場，7 月 31 日(水)-8 月 2 日(金) (2013).
5. 大島聡史，金子勇，片桐孝洋: メニーコアアーキテクチャ向けの SpMV 最適化と自動チューニング，日本応用数理学会 2013 年度年会，アクロス福岡，9 月 9 日(月)-11 日(水) (2013).
6. 大島聡史: GPU プログラム最適化のための指示文を用いたチューニング機構の開発，第 5 回 自動チューニング技術の現状と応用に関するシンポジウム，東京大学 山上会館，12 月 25 日（水）(2013).

〔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
http://www.cspp.cc.u-tokyo.ac.jp/ohshima/gpupragma/

６．研究組織
(1)研究代表者
大島聡史（OHSHIMA, Satoshi）
東京大学・情報基盤センター・助教
研究者番号：40570081