平成27年度 スーパーコンピューター若手・女性利用者推薦利用報告

大島聡史
東京大学情報基盤センター

1 はじめに

本報告者は平成27年度のスーパーコンピューター若手・女性利用者推薦制度(以下、本制度)において、前期課題として「ppOpen-ATを用いたOpenACCプログラムの最適化に関する研究」、後期課題として「ppOpen-ATを用いたOpenACCプログラムの最適化：全体最適化と指示文拡張に向けて」の採択を受け、平成27年5月から9月および11月から12月に「GPUクラスタ」を利用した。本制度にて実施した内容を含む研究の成果は国内外のいくつかの研究会等[1, 2, 3, 4]にて発表済である。これらの研究発表の内容をまとめて再構成したものを、本制度の利用報告とする。

2 研究の背景

画像処理を目的に開発されたGPU (Graphics Processing Unit)は、高い演算性能やメモリ転送性能を持つため、今日では科学技術計算や機械学習、ビッグデータ処理など様々な用途に利用されている。特にこれらの分野で広く用いられているGPUとしてはNVIDIA社のGeForceシリーズおよびTeslaシリーズがあげられる。これらを活用して高い性能を得るためのGPUプログラミング環境として、CUDA[5]という既存のマルチコアCPUやメニーコアプロセッサとは異なるプログラミング環境がNVIDIA社から無償で提供されている。CUDAを用いたプログラミングは、CUDA登場以前に用いられていたGPUプログラミング手法と比べれば各段に使いやすいものであるものの、マルチコアCPUやメニーコアプロセッサで用いられているOpenMPやMPIとは異なるものであり、習得や利用がやや難しい。そのため、GPUを含む様々な並列計算機環境で利用可能な共通のプログラミング環境の登場が望まれていた。

OpenACC[6]は、GPUを含む様々な並列計算機環境向けのプログラムを指示文を用いた平易な記述で統一的に行えるようにするべく提案された並列化プログラミング環境である。OpenACCの言語仕様や使い方はOpenMPに似た点が多いため、既存のOpenMP利用者にとっては習得や利用の難易度が低い。また対象とする処理の内容によっては、OpenACCを用いてCUDAに匹敵する性能を持つGPUプログラムを作成することも可能である。そのためGPU向け並列化プログラミング環境としてOpenACCへの期待は大きい。しかし、OpenACCを用いてGPUの持つ性能を十分に引き出すためには、OpenMPを用いてCPU向けの最適化プログラミングを行う場合とは異なる最適化の知識や技術、手間が必要であることも事実である。

一方、プログラムの性能最適化を行う手法の一つとして自動チューニング(Auto-tuning, AT)技術への注目が高まっている。本報告者らは指示文を用いてパラメタ最適化などを容易に行うことができるAT言語**ppOpen-AT**[7, 8, 9]の開発を行っており、すでにマルチコアCPUやメニーコアプロセッサ上で動作する様々なプログラムの性能最適化においてその有効性を確認している。しかしppOpen-ATはOpenACCプログラムを対象として開発されてこなかったため、

OpenACCプログラムに対しても利用できるのかや有効性があるのかは確認されていなかった。そこで本制度の利用を含む一連の研究においては、ppOpen-ATがOpenACCプログラムの最適化にも利用可能であるのか、利用不可能であるならどのような修正が必要なのか、利用可能である(修正により利用可能となる)場合にはどの程度の有効性があるのか、ppOpen-ATにどのような変更や機能追加があればOpenACCプログラムの最適化に今より有用となるのか、といった点について調査や提案および評価を実施している。

3　OpenACC最適化プログラミング

3.1　OpenACC プログラムの最適化

OpenACCを用いた並列化プログラミングはいくつかの指示文を挿入するだけで行える容易なものであるが、高い性能を得るためには対象となるハードウェアの特徴を考慮した適切な指示文の挿入が必要不可欠である。そこで本章では主なOpenACCプログラムの最適化方法について述べ、次章では既存のppOpen-ATがこれらの最適化手法に対して貢献できるかどうかを検討する。

3.2　ループの並列度に対する最適化

OpenMPやOpenACCにおいてループの並列化を行う際、対象となるループのイタレーション数よりも計算コア数が多い場合には何も行う仕事のない計算コアが生じてしまい、性能効率の低下を招く。そのため対象となるループイタレーション数は基本的に大きい方が良いとされる。今日のマルチコアCPUやメニーコアプロセッサは、1システムあたり最大数百のスレッドを同時に用いることが可能である。これはすなわちループイタレーション数が数百なければ対象計算機環境の性能を発揮できないことがあることを意味する。一方、GPUは最大数千の計算コアを搭載している。そのためOpenMPを用いて実装されていた既存の並列化済プログラムをOpenACCに移植する場合、ループイタレーション数の不足が主な原因で十分な性能が得られない可能性がある。

また、OpenMPにおける並列度の指定はスレッド数の指定という一階層の指定であり、スレッド数については多くの場合、利用する物理コア数または物理コア数の数倍(Hyper ThreadingなどのSMT機能により同時実行可能な数まで)にあわせれば良い性能が得られる。一方OpenACCでは並列度をgang, worker, vectorの三階層で指定することが可能であり、GPU上のコアがどのようなコアグループを構成しているかにあわせて適切な値を指定する必要がある。さらに高速なスレッド切替によりメモリアクセスのレイテンシを隠蔽するため物理コア数を越えた並列度が必要である。これはCUDAプログラミングにおけるグリッドやスレッドの数の最適化に対応している。また並列化対象が階層的なループの場合にはcollapse節を用いてループの階層を減らすことも有効となり得る。

3.3　メモリアクセスに関する最適化

CPU(OpenMP)とGPU(OpenACC)では高いメモリアクセス性能を得やすいプログラムの構造が異なる。

OpenMPによるCPU向けの並列化プログラミングにおいては、スレッド毎にアクセスするメモリの範囲がある程度離れていることが望ましい。これは複数のスレッドが近い範囲のメモリにアクセスしようとするとキャッシュやメモリのアクセスに競合が発生して性能が低下するためである。

一方GPUについては、同一コアグループに属する計算コア同士などある範囲の計算コアが連続するもしくは近接するメモリをアクセスする(コアレスなメモリアクセスをする)方が良い。これはメモリアクセスをまとめて行う仕組みがあるためであり、CPUとは異なる重要な特徴である。

一般的なソースコード記述レベルの最適化方法に対応させて考えた場合、階層的なループの順序入れ替えや、多次元配列の走査順序の変更、Structure of Arrays (SoA) / Array of Structures (AoS) の選択などの最適化において、CPUとGPUでは異なる実装やパラメタを選択することで高い性能が得られる可能性がある。OpenACCにはこれらについて自動的に最適なものを選んでくれるような機能はないため、プログラマ自身の手による最適化記述が必要である。

3.4 データ転送やその他の最適化

CPUとGPUは得意とする処理が異なる。対象とする処理がCPUで実行する方が高速なものなのかGPUで実行する方が高速なものなのかは処理の内容に依存し、さらに処理を行うプロセッサをCPUからGPUへ、またはGPUからCPUへ変更する際にはデータの転送も必要となることがある。そのため、高い性能を得るためには、対象となる処理を実行するプロセッサを適切に選択することも重要である。

また現在のGPUはハードウェアキャッシュを搭載しているため、CPUなどと同様にブロック化などのアルゴリズムが効果を得られることがある。ブロック化などの最適化手法を利用する際には、ブロックサイズなどの最適化が必要になることが多い。

4 ppOpen-ATのOpenACC対応と既存のAT機能の有効性に対する評価

4.1 ppOpen-AT の OpenACC 対応および修正

ppOpen-ATはJST-CREST「自動チューニング機構を有するアプリケーション開発・実行環境：ppOpen-HPC」の構成要素の1つである、自動チューニングのための言語である。

ppOpen-ATの主な特徴としては、OpenMPやOpenACCのように指示文を用いてプログラムを最適化するものでありソースtoソースでプログラムを書き換えたり生成したりすることC言語とFortranに対応すること、インストール時・実行起動前・実行時の3タイミングでのATを行えること、OSレベルのATとは異なり専用のOSが動作している運用スパコンなどでも利用しやすいことなどがあげられる。またppOpen-ATの備える基本的なAT機構は、与えられた選択肢(変数候補群やサブカーネル群など)に基づく候補コード群をあらかじめ用意しておき、問題サイズなどのパラメタに対してどの候補コード群を選べば最も高い性能が得られるかを求める仕組みを備えている。

ppOpen-ATはOpenACC向けに開発されてきたものではないため、本研究の最初の課題はppOpen-ATの備える既存の機能がOpenACCプログラムに対しても大きな問題なく適用できるかであった。実際にppOpen-ATを用いてOpenACCプログラムに対するコードの書き換えを

図 1　変数選択機能

図 2　アンローリング機能

行った結果、いくつか細かい修正を行えばOpenACCコンパイラが問題を起こさずにプログラム変換処理を行えることが確認された。なおここで必要となった修正は、その後ppOpen-ATのバージョンアップ時に採り入れられている。

4.2　ppOpen-AT の備える最適化機能と OpenACC の対応付け

ppOpen-ATの備えるいくつかの最適化機能を用いてOpenACCプログラムに対するコードの書き換えを行い、変換されたプログラムを目視にて確認した。さらにOpenACCコンパイラに与えてコンパイル上の問題を確認し、必要に応じてコードの手動書き換えを行って動作内容や性能を確認した。以下、ppOpen-ATの備える最適化機能の例とOpenACCにてどのように活用できるかの検討内容を示す。

変数選択機能　変数選択機能(図1)は、指定した変数の値を指示文によって与えられた候補群から選択する機能である。任意の変数の値を変化させたコードバリエーションを生成する。本機能は非常に汎用的な機能であり、OpenACCプログラムにおいても有用であることは明らかである。

ループアンローリング機能　ループアンローリング自体はよく知られた最適化手法である。本機能(図2)ではアンローリングの段数を選択する機能である。OpenACCにおけるループのアンローリングは、ループの終了判定回数が減るという点において有用である一方、多数のコアをまかなうために並列度の高いループが必要であるというGPU向け並列化の基本に反する点は問題となる可能性もある。またループアクセスパターンが変化するためコアレスなメモリアクセスが保たれるように注意せねばならない。以上から、本機能はOpenACCプログラムにおいても有用である場合がある、と言える。

ループ変型機能　ループ変型機能はループの融合(Fusion)や消失(Collapse)、分割(Split) といったループ構造を変更させるような候補群を生成する機能である。本機能を用いればメモリアクセス順序の異なるループバリエーションや、ループ内で用いる一時変数の数量が異なるループバリエーションなどを容易に生成できるため、最適なOpenACCプログラム実装を検討するうえで有用であると考えられる。図3は2重ループを1重ループに変換する単純な例である。OpenACCの指示文が挿入された複雑なループ構造を変換する場合、変換後に意図したところにOpenACCの指示文が入った状態となっているかは注意

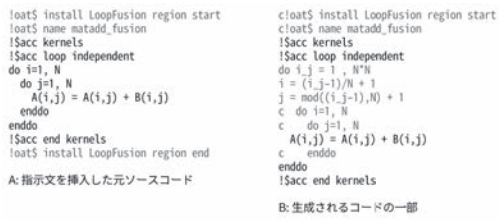

図 3　ループ変型機能

図 4　実装選択機能

せねばならない。例えば、collapse節を指定してあるループが結合/消失されてしまった場合にはコンパイルエラーが発生する。そのような問題を全てプログラム変換機構で防ぐのは難しく、生成されたコードの妥当性は利用者が確認する必要がある。

実装選択機能　変数選択機能1は与えられたサブカーネル群から最適なサブカーネルを選ぶための機能である。上述の他の最適化機能のように新たなコードバリエーションを生成するわけではなく、利用者が用意したコード群の中から最適なものを選ぶ機能である。そのため汎用性は高く、OpenACCプログラムの最適化においても様々な利用が考えられる。例えば同一の処理をCPU向けに記述したサブカーネルとGPU向けに記述したサブカーネルを作成しておき搭載されているCPUやGPUの性能にあわせてより高速な方を選ぶ、といった利用するハードウェアの選択などにも利用することができる。

以上のように、ppOpen-ATの提供する最適化手法はOpenACCプログラムの最適化にも有効性を発揮できることがわかった。

5　ppOpen-ATに対する機能拡張

ppOpen-ATを用いることでOpenACCプログラムの最適化が行えることは確認できたが、従来のppOpen-ATではうまく行うことができないOpenACCプログラム最適化手法も幾つか確認することができた。そこで、それらの最適化手法を有効に活用するべく、ppOpen-ATに対する機能拡張を提案した。本章ではこれらの新たな最適化機能および対応する指示文について述べる。

5.1　指示文内の変数を最適化するための機能

OpenACCプログラムの最適化においては指示文中におけるパラメタ選択が性能に大きな影響を与えることがある。特にgang, worker, vector節からなる並列実行形状の選択や、階層的なループを結合したものとして扱うことのできるcollapse節は、その指示文中で指定する数値が性能に大きな影響を及ぼす。これらのパラメタはppOpen-ATによるAT対象として適当であると考えられる。

しかし既存のppOpen-ATは指示行を単純なコメントとして解釈してしまう仕様であるため、指示文中の変数を最適化することは不可能であった。そこで新たな機能として、指示文内の変

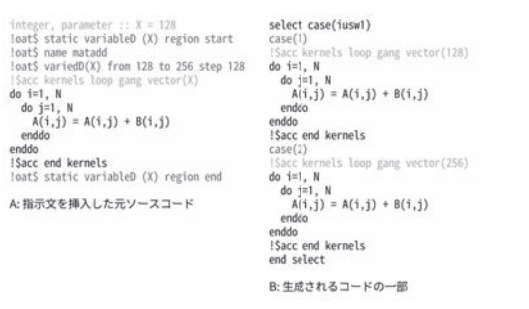

図 5　指示文中の変数を最適化するための拡張機能

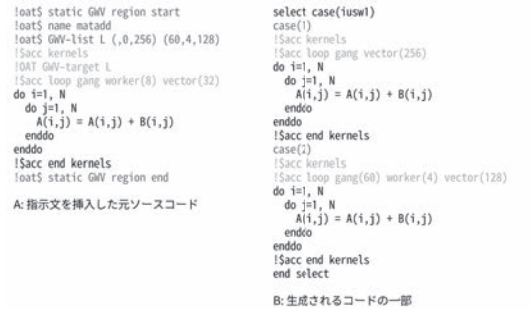

図 6　並列実行形状を最適化するための拡張機能

数を最適化するための記法と機能の追加を提案することにした。ところでppOpen-ATの設計方針として、ppOpen-ATに関する指示文を無視した場合になるべくそのままプログラムをコンパイル/実行可能にするという方針がある。単純に指示文中の値を変数として扱おうとした場合、ppOpen-ATによるソースコードの書き換えが行われない場合に指示文中に未知の文字列が存在することになりコンパイル不可能となってしまう。この点については、対象変数を定数(C言語におけるconst、Fortranにおけるparameter)として宣言しておけば少なくともPGIコンパイラはコンパイルエラーを発生させずに動作することが確認できた。図5に本機能を用いる例を示す。本機能の言語仕様は既存の変数最適化機能とほぼ同じである。

本機能はOpenACCプログラムの最適化のために考案した機能であるが、OpenMPプログラムにおいても有効に機能する可能性がある。例えばOpenMPにもcollapse指示文が存在し、与える数値によっていくつのループをまとめるかを制御することができる。また並列ループ実行時のスケジューリング設定についてもstatic, dynamic, guidedのスケジューリング種別と対応するチャンクサイズを指定することが可能である。本提案機能はこれらのパラメタ最適化にも有用であると期待される。

5.2　並列実行形状を最適化するための機能

指示文内の変数を最適化するための機能を用いることで並列実行形状の最適化が可能となった。一方で、様々なパターンを指定すると最適化のための時間が長くなってしまうことがわかっている。また並列実行形状の指定時には一部の形状情報を省略することも可能であるが、指示文内の変数を最適化する機能では値の省略に対応することができない。そこでより並列実行形状の最適化に特化した機能と記法の追加を提案した。図6に本機能を用いる例を示す。本機能はGWV-list節に続いて書き換え対象となるloop指示文を識別するためのラベルとGang/Worker/Vectorの候補の組を与えて利用する。GWV-targetの直後に存在するloop指示文が書き換え対象である。候補の組となる数値については、順にGang/Worker/Vectorの値を与える。この際、0であればその形状を使わない、1以上の整数であればその値を使用する、その他の値であればデフォルト(数値指定無し)を意味する仕様とした。これにより、柔軟な並列実行形状の指定とパラメタ探索/性能比較が可能となった。

```
!oat$ static GWV region start
!oat$ name spmv
!oat$ GWV-list label (A,B,C) (C,Y,Z), …
!$acc kernels          実際には候補値群を記述する
!oat$ GWV-target label
!$acc loop gang(30) worker(2) vector(64)
DO I=1,N                    AT対象となる箇所
  S=0.0D0
  DO J_PTR=IRP(I),IRP(I+1)-1
    S=S+VAL(J_PTR)*X(ICOL(J_PTR))
  END DO
  Y(I)=S
END DO
!$acc end kernels
!oat$ static GWV region end
```

図 7　疎行列ベクトル積ソースコード

```
• Tesla M2050
    • gang : default, 32, 64
    • vector : from 32 to 512 step 32
• Tesla K40
    • gang : default, 15, 30
    • vector : from 32 to 512 step 32
• FirePro S9000
    • gang : default, 28, 56
    • vector : from 32 to 512 step 32
```

図 8　GPU と対応する Gang/Vector の値

表 1　行列の形状情報

行列名	行数	総比ゼロ要素数
bone010	986,703	71,666,325
Hook_1498	1,498,023	60,917,445
Geo_1438	1,437,960	63,156,690
Serena	1,391,349	64,531,701
audikw_1	943,695	77,651,847
Flan_1565	1,564,794	117,406,044

表 2　選ばれた最適なパラメタ

行列名	Tesla M2050	Tesla K40	FirePro S9000
bone010	64 / 32	def. / 32	28 / 160
Hook_1498	def. / 32	def. / 32	28 / 224
Geo_1438	64 / 32	def. / 32	28 / 160
Serena	64 / 32	def. / 32	28 / 192
audikw_1	64 / 32	def. / 32	28 / 192
Flan_1565	64 / 32	15 / 32	28 / 160

5.3　有効性の検証

指示文内の変数を最適化するための機能や**並列実行形状を最適化するための機能**の有効性については、行列積などの実験プログラムに対する最適パラメタ探索によって確認できている。

一例として、CRS形式の疎行列(Florida Sparse Matrix Collection[10]の一部)に対する疎行列ベクトル積のシンプルな並列計算において並列実行形状の最適化を行った例を示す。なおこの実験例は[4]にて行ったものと同じである。図7は指示文の挿入されたコード、図8は使用したGPUおよび対応するGang/Vectorの候補群である。今回の実験ではWorkerは利用しなかった。

性能評価対象として用いた行列の構成は表1の通りである。また選ばれた最適なパラメタは表2の通りである。各GPUともにほとんどの行列が同じ形状において最適な性能を得ていたが、一部の行列ではパラメタを変更した方がより高い性能が得られることが確認できている。

6　まとめ

本利用報告では、平成27年度 スーパーコンピューター若手・女性利用者推薦制度(前期および後期) を用いて実施した研究について、実施内容を元に行った対外発表の内容をまとめて再構成する形で紹介した。本制度においては、主にOpenACCプログラムを書き換えてプログラムの動作や性能を確認する用途に「GPUクラスタ」を利用した。本利用報告に書かれた内容を含むOpenACCへの対応がなされたppOpen-ATは、既にプロジェクトのWebページ[11]上にて公開済である。

本研究の今後の課題としては、様々なアプリケーションに適用してさらなる効果の検証や機能の強化を行うこと、ループ変型機能にて書き換えられたループに対してさらに並列実行形状のATを行うような利用方法についての検討、プロファイラなどと連携してATの手間/試行数/時間を減らすことなどがあげられる。

参考文献

[1] 大島聡史, 片桐孝洋, 松本正晴: ppOpen-ATを用いたOpenACCプログラムの自動チューニング, 情報処理学会 研究報告(HPC-150) (2015).

[2] Satoshi Ohshima: Effectiveness of ppOpen-AT for OpenACC, 2016 Conference on Advanced Topics and Auto Tuning in High-Performance Scientific Computing, Feburuary 19 (2016).

[3] Satoshi Ohshima, Takahiro Katagiri, Masaharu Matsumoto: Utilization and Expansion of ppOpen-AT for OpenACC, GTC2016, San Jose, April 4-7 (2016). (poster presentation)

[4] Satoshi Ohshima, Takahiro Katagiri, Masaharu Matsumoto: Utilization and Expansion of ppOpen-AT for OpenACC, The Eleventh International Workshop on Automatic Performance Tuning (iWAPT2016) (In Conjunction with the IEEE IPDPS2016), May 27 (2016).

[5] CUDA Dynamic Parallelism, http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#cuda-dynamic-parallelism

[6] OpenACC Home http://openacc.org/

[7] T. Katagiri, S. Ito, S. Ohshima, "Early experiences for adaptation of auto-tuning by ppOpen-AT to an explicit method," Special Session: Auto-Tuning for Multicore and GPU (ATMG) (In Conjunction with the IEEE MCSoC-13), Proceedings of MCSoC-13, pp.153-158 (2013).

[8] T. Katagiri, S. Ohshima, M. Matsumoto, "Auto-tuning of computation kernels from an FDM Code with ppOpen-AT," Special Session: Auto-Tuning for Multicore and GPU (ATMG) (In Conjunction with the IEEE MCSoC-14), Proceedings of MCSoC-14, pp.253-260 (2014).

[9] T. Katagiri, S. Ohshima, M. Matsumoto, "Directive-based Auto-tuning for the Finite Difference Method on the Xeon Phi," International Workshop on Automatic Performance Tuning (iWAPT2015), Proceedings of IPDPSW2015, pp.1221-1230 (2015).

[10] T.A.Davis, Sparse Matrix collection, http://www.cise.ufl.edu/research/sparse/matrices/

[11] ppOpen-HPC — Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta-Scale Supercomputers with Automatic Tuning (AT) http://ppopenhpc.cc.u-tokyo.ac.jp/ppopenhpc/