数理解析研究所講究録
第1791巻 2012年 107-111

ppOpen-AT: ポストペタスケール時代の数値シミュレーション基盤ソフトウェア *ppOpen-HPC* のための自動チューニング基盤

ppOpen-AT: An Auto-tuning Infrastructure for Numerical Simulation Infrastructure *ppOpen-HPC* Towards To Post Petascale Era

東京大学・情報基盤センター　片桐　孝洋(Takahiro Katagiri)
Information Technology Center,
The University of Tokyo

1. はじめに

先進計算機のハードウェア構成がますます複雑化している．非均質メモリ，多階層キャッシュ，チップ上に多数の計算機要素（コア）をもつマルチコアCPU，そして，GPUに代表される演算アクセラレータ，を搭載した計算機が普及している．ここ数年以内のごく近い将来に，CPUとGPUを混載した計算機環境（**非均質（ヘテロジニアス）計算機環境**と呼ぶ）が主流になると予想されている．

このように複雑化した計算機環境では，工学や理学で必要となる科学技術計算プログラムの最適化が，高性能計算を専門としない科学者やエンジニアにとって困難になる．この状況は，応用数理学者にとっても例外ではない．たとえば，数理的な手法を使ってある手法の収束を2倍効率良くしたとしても，その提案手法の演算が非均質計算機環境に不向きであり従来実装の2倍演算効率が悪くなれば，提案手法の実際の効果は無い．したがって，数値計算法を提案するにも，計算機環境の特徴を知っておく必要がある．

そこで我々は，大規模並列計算機上での実用的なシミュレーションコード開発と演算最適化を支援するソフトウェア基盤 *ppOpen-HPC* を開発している．ここでの“pp”とは，post-petaの略語であり，ペタスケール計算機の後に来る計算機環境のことを指す．ポストペタ環境では，非均質な計算ノードを持つことが予想されている．つまり，マルチコアCPUと，演算アクセラレータであるGPUの混合構成となることが予想されている．

2. *ppOpen-HPC* とAT基盤 *ppOpen-AT*

2.1 *ppOpen-HPC* 概要

ppOpen-HPC が対象とするのは，現在主流の数値計算法によるプログラムである．主にFEM (Finite Element Method)，FDM (Finite Difference Method)，BEM (Boundary Element Method) およびDEM (Discrete Element Method) といった数値計算法を対象としている．

一方，自動チューニング（Auto-tuning (AT)）技術が，先進的計算機環境での性能チューニングで必須となるといわれている．*ppOpen-HPC* はAT技術を採用して開発される．*ppOpen-HPC* のためのAT基盤ソフトウェアを *ppOpen-AT* と呼ぶ．

図1に *ppOpen-HPC* における *ppOpen-AT* の位置づけとその機能をのせる．

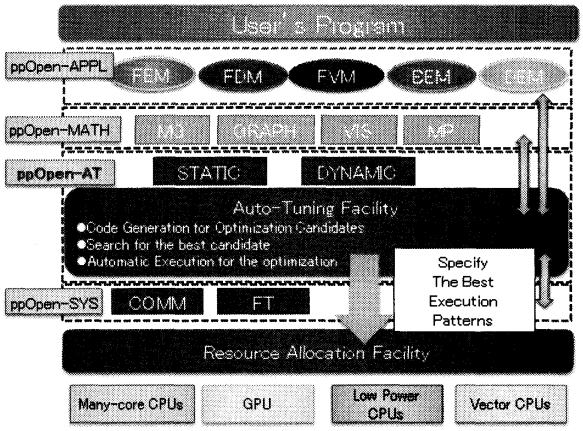

図1　*ppOpen-HPC* における *ppOpen-AT* の位置づけとその機能

2.2 *ppOpen-AT* 概要

ppOpen-AT は，*ppOpen-MATH* と *ppOpen-SYS* に AT 機能を提供する．*ppOpen-AT* の AT 機能を利用することで，計算機資源選択—ここでは CPU と GPU の自動切り替えの機能—を実現する．

図 1 の計算機資源選択は，*ppOpen-AT* の AT 機能と，ユーザ環境ごとに異なるコンパイラ最適化機能を合わせて使うことにより，*ppOpen-AT* 専用プリプロセッサを通して，コード最適化機能の 1 つとして実現される．このことで幅広い計算機環境での AT を可能にする．*ppOpen-AT* の最適化対象は，*ppOpen-HPC* で実装されている，（1）数値計算法カーネル；（2）プログラム；（3）実装方式；に限定する．主にデータアクセスパターンの最適化を行う．*ppOpen-APPL* の原始コード（C，C++，Fortran90）が対象であり，コンパイラの「静的最適化」の範疇である．この機能を *ppOpen-AT / STATIC* と記載する．将来的には，実行時最適化機能の提供も検討している．実行時最適化機能を *ppOpen-AT / DYNAMIC* と記載する．

ppOpen-AT / STATIC は以下の機能をもつ：

(1) オリジナルの C，C++，Fortran90 コードから自動生成された CUDA コード，もしくは OpenCL コードに対する最適化機能；

(2) DEM，FDM，および BEM のコードに対し，特定のディレクティブを使うことによる直接の GPU コード生成機能；

2.3 計算機資源選択のための機能

ppOpen-AT が提供する計算機資源選択機能を実現するには，ABCLibScript のオリジナルなディレクティブ機能[1]に対し拡張が必要である．この拡張は，非均質環境における計算機資源選択を実現するものである．以下の 2 通りの方法を前提とする：

(1)ABCLibScript 専用の拡張ディレクティブ “allocate(auto)” を利用する方法[2]；

(2)ABCLibScript オリジナルのディレクティブ “select region” を利用し，CPU コードと GPU コードの 2 種を開発者が用意したうえで，コード切り替え機能を AT 化する方法；

本稿では，(2) の方法について説明し，予備評価を行う．

3. 予備評価

3.1 計算機環境

本稿で利用した，CPU と GPU の混載環境の概要を以下に述べる．

- CPU 環境：
  Xeon W3520 (Nehalem Quad-core, 2.67GHz)．主メモリ：12GB．OS: CentOS5.5 x86_64.
- GPU 環境：
  Tesla C2050．CUDA toolkit ： version3.2. GPU Driver 260.19.26.
- コンパイラ：
  PGI コンパイラ version 11.0-0．OpenMP によるスレッド並列化．PGI アクセレータ機能により CUDA コードを自動生成する．コンパイラオプション：“-Msafeptr=all -fastsse -lm -mp -ta=nvidia -Mlist -Minfo=accel,mp”

3.2 GPU コードについて

本予備評価では，*ppOpen-AT / STATIC* により選択される GPU コード（CUDA コード）は，PGI アクセレータ機能により自動生成される CUDA コードを利用した．なお [2] の方式では，ABCLibScript が PGI アクセレータのディレクティブを自動生成するので，ユーザが PGI アクセレータのディレクティブを記述しなくても CUDA コードが自動生成されることに注意する．

PGI アクセレータ機能を使うことなく CUDA コードを開発者が用意すれば，より高性能な GPU コードを利用できる．だがここでの目的は，高性能な GPU コードを自動生成することではなく，CPU コードと GPU コードを自動選択する計算機資源選択の AT 機能を *ppOpen-AT / STATIC* により提供し，その有効性を示すことにある．このことで，AT 機能付きのソフトウェアを低コスト—ここでは少ない開発工数—で開発ができることを示す．

3.3 問題の詳細

扱うシミュレーションプログラムは，京都大学の岩下武史氏のグループが *ppOpen-HPC* の提供機能の一つとして開発しているコードである．境界要素法（Boundary Element Method, BEM）を利用した静電場の問題である．

上記シミュレーションの問題は，地上から 50cm のところに半径 25cm の金属球を配置し，金属球に +1V の電位を与えたときに金属球表面に誘起され

る電荷を求めるものである．空間内の電場，ポテンシャル（電位）の値が分かる．

このシミュレーションでは連立一次方程式を解く．そのため用いられている数値解法は，前処理なしのBiCGStab法である．連立一次方程式の係数行列は「密行列」として扱われている．いくつかのシミュレーションでは係数行列は疎行列となるが，BEMでは境界接触面の影響がほぼすべての要素に影響を及ぼすという特性から，係数行列がほぼ密行列になる．なので，係数行列を密行列として扱うことはBEMでは特殊ではない．

このシミュレーションにおける主演算は，BiCGStab法の中で用いる「密行列-ベクトル積」である．

本問題から作られるテスト用の行列サイズは，以下の4種である：

- Small：600 x 600,
- Medium：2400 x 2400,
- Medium-Large：15,000 x 15,000,
- Large: 21,600 x 21,600.

3.4 AT記述の方法

想定する*ppOpen-AT/STATIC*によるATの実現方法は，CPUコードとGPUコードで異なる．図2に，CPUコードにおける「行列-ベクトル積ルーチン」の*ppOpen-AT/STATIC*記述を載せる．

```
#pragma oat install unroll (row, col) region start
#pragma oat name MyMatVec
#pragma oat varied (row, col) from 1 to 4
#pragma omp parallel for private(col)
for (row=0; row < dim; row++){
  tmp = 0.0;
  for (col=0; col < dim; col++){
    tmp = tmp + mat[row*dim+col] * p[col];
  }
  q[row] = tmp;
}
#pragma oat install unroll (row, col) region end
```

図2　CPUコードのAT記述

図2では，“#pragma oat”で始まる指示行（プラグマ）が，*ppOpen-AT/STATIC*へのAT指示である．ATのタイミングは「インストール時」(install)であり，ソフトウェアのインストール時にATを一度行い，その後，そのAT結果を利用した実装選択を実行時に行うことを意味している．

図2では，密行列-ベクトル積を構成する2重ループのループ導入変数 row, col のそれぞれに1段から4段までのアンローリングを適用したコードを自動生成する．実行時間を実測の上，最高速となる実装方式を自動選択する．自動生成するコードの種類は，ここでは，4x4=16種類（varied (row, col) from 1 to 4）である．

図2では，さらにOpenMPのディレクティブが適用されている．OpenMPでスレッド並列化されたうえ，アンローリングを適用したコードになる．ここでは，2次元配列 mat を1次元化して利用している．これは，この環境では1次元化したほうが高速であったため，CPUコードでは1次元実装を採用した．一方，PGIアクセレータによりCUDA化したGPUコードは，2次元配列を用いた実装のほうが，1次元化した実装より高速であった．そのため，2次元配列を用いた実装を利用した。図2のATは，このシミュレーションでは，初めに行うATである．

図2の次に行うATは，BiCGStab法のメインループに対するCPUコードとGPUコードの切り替えとなる．このAT記述を，図3に示す．

```
#pragma oat select region start
#pragma oat name SelectGPU
#pragma oat select sub region start
  pbicgstab(dim, a, rhs, sol, tor, max_steps);
#pragma oat select sub region end
#pragma oat select sub region start
  pbicgstab_gpu(dim,a, rhs, sol, tor, max_steps);
#pragma oat select sub region end
#pragma oat select region end
```

図3　CPUとGPUとの切り替えAT記述

図3により，CPUコードとGPUコードの時間を実測し，適する方法を選択するAT機能を有するコードが自動生成される．

3.5 性能評価結果

図2と図3のAT機能について，*ppOpen-AT/STATIC*専用プリプロセッサにより自動生成されるコードについて性能評価した．CPUコードは，は8スレッド並列実行である点に注意する．

最初のATである，CPUでの行列-ベクトル積のAT効果を図4にのせる．図4では，8スレッド並列実行，N=600（Small）での実行時間を示している．コンパイラ最適化のみで明示的にアンローリン

グをしていない実行時間はiusw=1の時であり，それは0.095秒であった．ATにより，明示的にアンローリングを施したコードが選ばれた．この場合は，iusw=16の時であり，これは，外側ループ4段，内側ループ4段，アンローリングしたコードである．iusw=16の時の実行時間は，0.061秒であった．したがって，ATを搭載することで，約1.55倍の速度向上が達成できた．

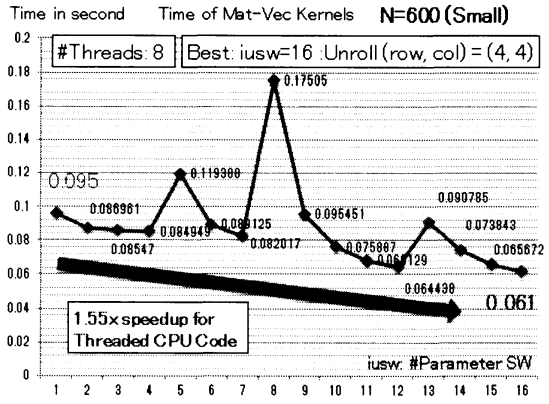

図4　CPUの行列-ベクトル積のAT効果（BEM）

2番目に行うATである，CPUとGPU選択の結果について表1に示す．なお，CPUでの行列-ベクトル積は図4で示した最適化がされている．表1の結果はSmall, Medium, Medium-Large, およびLargeともに最適な実装が選択されている．

表1　CPU-GPU切り替えAT効果（BEM）

Size	CPU Code [Second]	GPU Code [Second]	Speedup
Small	0.002	0.022	0.09x
Medium	0.128	0.125	1.02x
Medium-Large	7.96	6.99	1.13x
Large	-	Out of Memory	-

表1から，問題サイズSmallではCPU実行がGPU実行より約10倍高速であることがわかる．しかし，問題サイズMedium以上になると，GPUコードが高速となることがわかる．したがって，*ppOpen-AT / STATIC*によるCPU-GPU資源切り替えATは，この環境では有効といえる．

PGIアクセレータ機能によるCUDA化に際し，配列matのCPUからGPUへの転送を最小化している．これは，PGIアクセレータのプラグマにより，明示的にデータ転送のタイミングを指定することで実現される．したがって，ユーザがPGIアクセレータの機能を熟知していないと実現できない．しかし，*ppOpen-HPC*の利用方法を考えると，*ppOpen-HPC*開発者であるライブラリ開発者が熟知していればよいわけで，*ppOpen-HPC*を利用するユーザ（エンドユーザ）はPGIコンパイラやAT機能を一切知る必要はないといえる．すなわち，*ppOpen-HPC*は「ライブラリ」としてエンドユーザから利用されるので，エンドユーザは実装詳細について知る必要はない．

現在の実装は，技術的な問題がある．BiCGStab法の反復ループ全体をGPU化できない．この理由は，PGIアクセレータ機能の言語仕様上の制約から生じるものである．CUDAコードを手書きで開発すれば，BiCGStab法の反復ループ全体をGPU化できる．したがって，CUDAを用いるプログラミングをしたうえで，ppOpen-ATを用いるCPU-GPUの切り替えAT化を行えば，さらにGPUコードが有効になり，GPUを用いることでさらなる速度向上が得られるはずである．再度の記載になるが，ここでの目的は，PGIアクセレータでの実装やCUDAコードでの手書き実装にかかわらず，CPU-GPUの切り替えを容易に行える*ppOpen-AT*の機能を提供することにある．この点において，本予備評価で有効性を確認できた．

4. おわりに

本原稿では，*ppOpen-AT / STATIC*の概略と，*ppOpen-HPC*に含まれる境界要素法(BEM)プログラムに対しAT機能を実装する1例を紹介した．

予備評価の結果，BEMで用いられる密行列-ベクトル積に対し，CPU最適化の1例としてOpenMPによるスレッド並列化時のAT例を示した．一方，CPU-GPUの資源切り替えの1例として，PGIコンパイラを用いて自動的にCUDAコードを自動生成する方法によるGPUコードを，AT専用言語*ppOpen-AT / STATIC*を用いて実装する例を示した．双方ともATによる速度向上が確認され，AT方式の有効性が確認できた．

今後の予定として，*ppOpen-HPC*のプログラムに対して，*ppOpen-AT*のAT適用事例と，効果的な新規AT機能を開発していく．特にBEMの例のようなBLAS基本カーネルのATに限定するのではなく，FDMに現れる多様で多くの配列アクセスを含むループ

に適用できるAT方式に適用することを考えている．また，FEMのプログラムにおいて，問題場の物理特性情報，メッシュ結合情報，を活用することで，行列情報だけでは実現できない効率の良いブロック化やアンローリング実装について，新たなAT機能として開発していく予定である．従来から行っているBLASレベルのATではなく，より上位階層（すなわち，アプリケーションに近い階層）でのATを実現していく．このことが，*ppOpen-HPC*プロジェクトでのAT方式の新規性である．

関連研究について述べる．CPU-GPUの資源切り替えに関し，ストリーム処理に限定したAT機能の方式が滝沢ら[3]により提案されている．滝沢らの方法は，ストリーム処理に限定し，少ない試行でGPU性能を予測するモデルを提案し実装している．本アプローチのAT方式は，***ppOpen-HPC***の処理に限定している．CPU-GPU切り替え対象の処理拡大を目指しているため，AT対象はストリーム処理に限定しない．反面としてGPUやCPUの性能モデル化が難しくなる．現在実装されているATの方法は，ユーザがある条件を与えたうえ，全探索するアプローチ（非モデル化ベース）が採用されている．この方法はコスト高と思われるが，モデル化できないような汎用的な処理に対しても，開発者情報を入れることで探索範囲の限定が実現できる．本プロジェクトでは*ppOpen-HPC*で現れる処理に限定し，現実的な時間でATを行うAT方式の実現を目指している．

謝辞: 本研究は，**JST CREST**領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」，**H23**年度採択課題「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」（代表：中島研吾 東大教授）の支援による．

日頃ご議論いただくppOpen-HPCプロジェクトの諸氏に感謝します．また，BEMコードの提供およびサンプルデータの提供に関し，岩下武史氏（京都大学学術情報メディアセンター），美舩健氏（京都大学大学院工学研究科），高橋康人氏（同志社大学理工学部電気工学科）の諸氏に感謝いたします．

参 考 文 献

[1] Takahiro Katagiri, Kenji Kise, Hiroki Honda, and Toshitsugu Yuba, ABCLibScript: A Directive to Support Specification of An Auto-tuning Facility for Numerical Software, Parallel Computing, Vol. 32, No.1, pp.92-112 (2006).

[2] 片桐孝洋，大島聡史，平澤将一，本多弘樹：HxABCLibScript:非均質計算機向け自動チューニング記述言語拡張，情報処理学会研究報告-ハイパフォーマンスコンピューティング（HPC），Vol.2011-HPC-129, No.13, pp.1-8 (2011).

[3] 滝沢寛之，白取寛貴，佐藤功人，小林広明：SPRAT：実行時自動チューニング機能を備えるストリーム処理記述用言語，情報処理学会論文誌：ACS, Vol.1, No.2, pp.207-220 (2008).