B-039

GPU 用自動並列化コンパイラを用いた Fortran プログラムの高速化手法の提案

A Proposal of the Acceleration Technique using Automatic GPU Parallelizing Compiler for Fortran Codes

田中　裕也 †　吉見 真聡 *　三木 光範 *
Yuya Tanaka　Masato Yoshimi　Mitsunori Miki

1 はじめに

近年，アルゴリズム内の並列性の高い部分を GPU で処理することで，計算時間を短縮させる研究が広く行われている．科学技術計算の分野においては，大規模な行列処理を含んだプログラムが多く見られ，GPU を用いた並列化による実行時間の短縮が見込まれる．

しかし，整備された開発環境が提供されている現在においても，プログラムの実装やそのチューニングに要する開発コストは依然として高い．例えば，プログラム中の並列化する領域を決定するためには，長時間の作業が必要である．データ転送をはじめとするオーバヘッドを考慮した上で，CPU で処理するよりも実行時間が短縮できるか検討したり，並列化するループ内部の手続きに関する制限についても考慮したりしなくてはならない．

科学技術計算のプログラムには，Fortran 言語で記述されたものも多く存在する．それらのプログラムについて，追加の開発を行わずに GPU を用いて実行時間が短縮できることが望ましいと考える．

本研究では，Fortran プログラムを対象に，GPU による並列化を行う領域の抽出と，実行時間計測を自動化するシステムを用いる高速化手法を提案する．実際にシステムを実装し，ベンチマークプログラムの並列化を通してその評価を行った．

2 提案する高速化手法

2.1 概要

本手法では，PGI 社が開発，販売を行っている，PGI Accelererator[1] コンパイラを用いる．このコンパイラは，OpenMP[2] に似た指示行を用いるインタフェースを有している．ソースコードに GPU に処理を行わせるための指示行を追加すると，指定されたループブロックが GPU で並列処理される．

2.2 並列化領域の抽出方法

Fortran 言語のソースコードに存在するループのネスト構造を解析し，図 1 に示すような解析木を作成する．解析木の各ノードには，ループブロックの開始行と終了行が保存されている．

2.3 並列化領域の実行時間計測方法

作成した解析木を深さ優先順でたどり，各ノードに対応するループブロックを並列化指示行で囲んだソースコードを生成する．生成したソースコードをコンパイル，実行し，その実行時間を計測する．実行時間が最も短かったノードに対応するループブロックを，最も効率の良い並列化領域として判断する．

2.4 システムの実装

提案した高速化手法を実現するシステムを，Python 言語で新たに作成した．本システムは，主に次のような動作を行う．

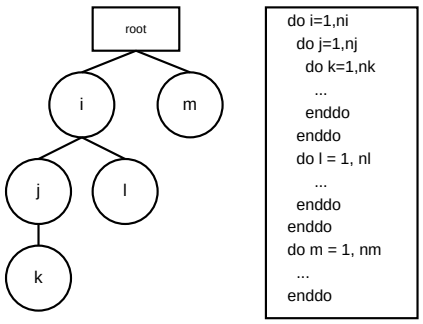

図 1　作成する解析木の例

1. Fortran 言語のソースコードを解析し解析木を作成
2. 解析木を元に，並列化指示行を挿入したソースコードを出力
3. 出力されたソースコードをコンパイル
4. コンパイルされたプログラムの実行時間を 10 回計測し平均を計算

図 2 に，システムが行う処理の流れを示す．

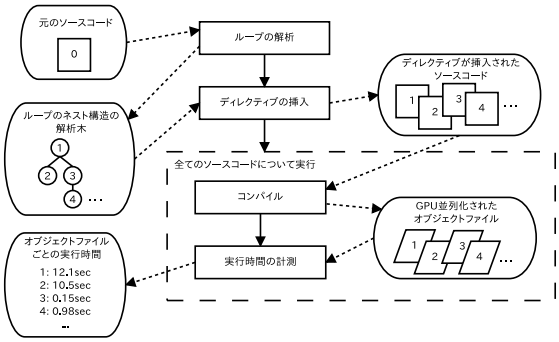

図 2　作成したシステムによる処理の流れ

3 システムの評価

3.1 評価方法

作成したシステムを用いて，インターネット上で公開されているベンチマークプログラムの並列化領域の最適化を行った．生成されたソースコードのそれぞれについて，実行時間を計測した．用いたベンチマークプログラムは，次に示すとおりである．

- 姫野ベンチマーク Fortran90 版 [3]
- 積分の計算を行うベンチマーク：intgl4[4]
- 行列積の計算を行うベンチマーク：matmul[5]

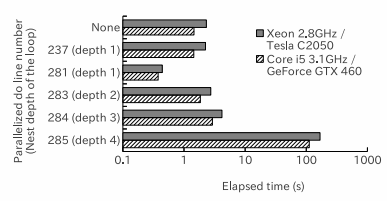

図 3　姫野ベンチマークの計測結果

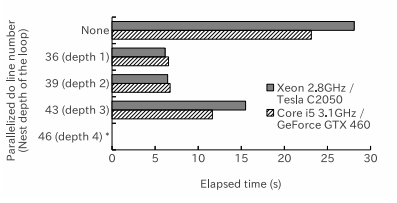

図 4　intgl4 ベンチマークの計測結果

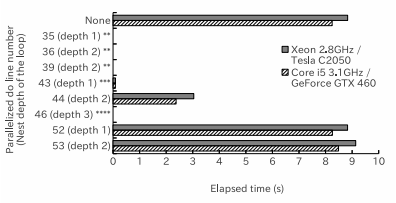

図 5　matmul ベンチマークの計測結果

表 1　評価用マシンのスペック

	マシン 1	マシン 2
CPU	Xeon W3530 2.8GHz	Core i5 2400 3.1GHz
メモリ	6GB	8GB
GPU	Tesla C2050	GeForce GTX 460
OS	Linux 2.6.26 x86_64	Linux 2.6.38 x86_64
コンパイラ	PGI Accelerator 2010 (10.9)	
オプション	-Minfo=accel,inline -fastsse -Minline=size:1000,levels:10,reshape -ta=nvidia,cuda3.1,time	

各ベンチマークには，プログラムの開始から終了までの経過時間を計測して出力するように変更を加えた．実験に用いた環境では，GPU 初期化時間に 1.5〜3 秒ほどの時間を消費していたため，各ベンチマークプログラムの実行開始部分にベンチマークと無関係なループおよび並列化指示行を配置し，強制的に GPU 初期化を行った後，実行時間の計測を開始するようにした．

姫野ベンチマークについては，おおよその実行速度に応じて繰り返し回数を変更するようなプログラムが含まれていたため，この処理を削除し繰り返し回数を 20 回に固定した．

評価に用いたマシンのスペックと，コンパイラとそのオプションについて，表 1 に示す．

3.2　評価結果

それぞれのベンチマークプログラムについて，自動並列化によって得られたオブジェクトファイルの実行時間を，図 3，図 4，図 5 に示した．

姫野ベンチマークについては，ヤコビの反復法を行う jacobi 関数の中に，4 重のループが含まれている．その最も外側にあたるループが 281 行目にある．図 3 を参照すると，その部分を並列化領域を指定する場合に実行時間が大幅に短縮されていることがわかる．

積分を行うベンチマークについては，intg4a 関数に 4 重のループが含まれている．その最も外側にあたるループが 39 行目にある．図 4 を参照すると，その部分を並列化領域を指定する場合に実行時間が短縮されていることがわかる．なお，“ * ” で示した部分の結果が 0 であるが，コンパイルされたプログラムがメモリを占有したため，強制終了を行った．

行列積を行うベンチマークについては，図 5 を参照すると，43 行目の 3 重ループを並列化領域に含めると実行時間が大幅に短縮されていることがわかる．なお，“ *** ” が示した部分の結果は，マシン 1 が 0.090，マシン 2 が 0.088 である．“ ** ” と “ **** ” で示した部分の結果が 0 であるが，データ並列性のない手続きが存在し，並列化できなかったためである．PGI Accelerator を用いて並列化を行う際には，並列化領域内のループブロックについて，次の項目をはじめとする条件が課せられている．

- データ並列性があること
- 関数呼び出しの禁止
- ループの出口が 1 つであること

4　まとめと今後の展望

本研究では，Fortran プログラムの GPU 並列化において，並列化領域を自動的に抽出し，実行時間を時間計測するシステムを使った高速化手法を提案した．提案した手法を実現するシステムを実装し，ベンチマークプログラムを並列化して評価した．この手法を用いることで，ベンチマークプログラムの並列化が自動的に行われ，実行時間が短縮されるような並列化領域を得ることができた．また，並列化領域の指定によっては，性能が大きく変化することも実験結果からわかった．

本研究においては並列化領域の数を高々 1 つに限定していたが，より大きな規模のプログラムでは，複数の並列化領域の組み合わせによる最適化が必要になると考えられる．今後の研究で，組み合わせの最適化についても行うように改善したい．

参考文献

[1] PGI Accelerator.
http://www.pgroup.com/resources/accel.htm.

[2] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science Engineering, IEEE*, Vol. 5, No. 1, pp. 46–55, jan-mar 1998.

[3] 姫野ベンチマーク.
http://accc.riken.jp/HPC/HimenoBMT.html.

[4] N. Tajima’s fortran benchmark tests (Ver.2).
http://serv.apphy.fukui-u.ac.jp/ tajima/bench/.

[5] Fortran Benchmarks (University of Western Ontario).
http://www.stats.uwo.ca/faculty/aim/epubs/benchmark/fortran.htm.