Vol.2011-HPC-130 No.16
2011/7/27

SPEC ベンチマークプログラムの CUDA による並列化の検討

平 勇樹[†]　木村啓二[†]　笠原博徳[†]

[†]早稲田大学

近年，GPU を汎用的な科学計算に用いる手法である GPGPU が注目されている．GPU は CPU と比べて高速な演算性能を持っているが，GPU の高い並列性を生かすためには並列性を持ったプログラムの選択と最適化が不可欠である．Doall のような単純な計算においては性能を出しやすいが，漸化計算（Reduction）のような並列性が低くなる計算では最適化を行わなければ性能を生かし切ることが出来ない．
本研究報告では，並列性の高い汎用 SMP 用ベンチマークである SPEC OMPL2001 331.art を評価対象とし，最適化によって GPU の性能がどれほど得られ，データサイズによってどれほど性能向上の差があるかを調査する．
GPGPU のための並列アーキテクチャ CUDA を用いてベンチマークプログラム 331.art を最適化し評価を行ったところ，12 コアでの並列実行の総計算時間と比べて 1.90 倍の速度向上を得た．また，配列サイズが 200 万個以上の漸化計算であればデータ転送帯域を有効に使うことが出来，CPU での並列実行より高速に動作させられることが確認できた．

Examination of Parallelization by CUDA in SPEC benchmark program

Yuki Taira[†], Keiji Kimura[†] and Hironori Kasahara[†]

[†]Waseda University

Recently, GPGPU which means a technique of General Purpose computing on GPU has attracted attention. GPU has a high-speed computing performance compared with CPU. Although in order to utilize a high parallelism that GPU have well, it is necessary to select a program with parallelism and optimize the program. It is easy to give high performance in the simple calculation such as Doall, but cannot make use of performance if you don't optimize a low parallelism compute such as Reduction.
In this paper, we set a target for SPEC OMPL2001 331.art which has high parallelism and evaluate how much performance is provided by GPU optimize and evaluate how much difference will appear by changing data size of arrays.
In this paper, we got speed-up of 1.90 times compared with the total calculation time of parallel execution in 12 cores. We can execute faster than parallel execution in 12 cores when we set a target as a Reduction which access to 2 Million data array.

1. はじめに

CPUが年々速度増加を遂げるのと同時に, GPU(Graphic Processing Unit)の計算速度は同年代のCPUより速い計算速度を保ちながら速度増加を遂げている．CPUの性能に関しては，CISCO社のIntel Xeon X5680 を搭載したブレードサーバーがLinpackにおいて 146.8GFlops [1]を達成した．Intel Xeon X5680 のメモリ転送帯域はメーカーの公称値によると 32GB/sである．これに対しGPUの性能に関しては，メーカー公称値によるとNVIDIA Tesla C2050 の単精度演算性能は 1.03TFlops，メモリ転送帯域は 144GB/sである．このように，演算能力ではCPUの 7.02 倍，メモリ転送帯域では 4.5 倍と，同時期のCPUとGPUではGPUの方がより高速な演算を行うことが出来る．このような速度差は，CPUとGPUでは行うべき用途が違うことによって生まれている．CPUは分岐予測やout-of-order実行等により，複雑な制御フローを持つ様々なプログラムの処理に向いている．その一方でGPUは，簡易な計算コアを多数搭載し，さらに高バンド幅のメモリを接続することで，大量のデータに対する比較的規則的な処理に適したアーキテクチャとなっている．その一方で，GPUの各演算コアは最小限の制御ロジックのみを持つ．

近年，NVIDIA 社の CUDA によって GPU のプログラム開発の生産性は大きく向上し，GPU の高い計算能力を利用する環境が整ってきたが，GPU の高並列性を生かし性能を発揮するには高い並列性を持ったプログラムの選択と最適化が不可欠である．前述のように GPU は最小限の制御ロジックしか持たないため，Doall のような単純な計算においては性能を出しやすいが，if 文を多用した計算や，漸化計算（Reduction）のような各コアの演算結果を 1 つのスカラ値に縮約するような計算では特殊な最適化を行わなければ性能を活かし切ることが出来ない．

一方，我々は汎用CPUとGPU等のアクセラレータを混載したヘテロジニアス・マルチコア用の自動並列化コンパイル手法を研究・開発してきた[2][3]．このようなヘテロジニアスなシステムにおいて，CPUとアクセラレータに効率よくタスクを割り当てるためには，CPUとアクセラレータで実行可能な処理の特性およびそれらの処理の性能差がどの程度のものか見積もることが特に重要となる．

CPUとGPUの性能差評価に関しては，特に[4]において各演算カーネルの処理特性とメモリバンド幅等の面から詳細な評価が行われた．またSPEC CPUベンチマークを利用した評価の例としては，SPEC OMP2001 Swim，SPEC CPU2000 MgridあるいはSPEC CPU2006 BWAVESを例に，GPU用メモリ最適化を適用した研究がある[5][6][7]．特に[7]はgrid sizeやデータレイアウトの面でGPUにそれほど適していないプログラムであり，また本稿が対象とする 331.artのように多数の漸化計算を持つ．

本稿では，将来のヘテロジニアス・マルチコア用自動並列化コンパイラへの GPU の適用の基礎データ取得を目的として, 漸化計算を多く含む汎用 SMP 用ベンチマーク

に対して漸化計算の最適化を行い性能評価をする．対象とするベンチマークは SPEC OMPL2001 で提供されている 331.art である．このプログラムは本来汎用 SMP 上で動かすことを前提として作られたベンチマークであるため大きなデータサイズを持ち，既に OpenMP によって並列化され高い並列性を持っていることが分かっている．このような汎用 SMP 用途として作られた高い並列性を持ったプログラムに対し GPU 計算の実装を行い性能計測することで，CPU と GPU の性能特性を比較検討する．また，データセットを変更することでデータセットのサイズと GPU の性能向上の関係を評価する．

以下，本稿の構成は，2 節で対象プログラムである SPEC OMPL2001 331.art の概要と，その CUDA による GPU 並列化について述べ，3 節で性能評価とその考察を行う．最後に 4 節でまとめとする．

2. SPEC OMPL2001 331.art

2.1 331.art の概要

SPEC (Standard Performance Evaluation Corporation)は，コンピュータの性能を公平に評価するために設立された非営利団体である．SPECは様々なシステムの性能を用途別に評価するため，いくつかのベンチマークを提供している．SPEC OMP2001 は，OpenMP ベースのアプリケーションの性能評価を行うベンチマークである．また，SPEC OMP にはSPEC OMPM2001 とSPEC OMPL2001 の 2 つのバージョンがある．OMPL2001 ではOMPM2001 と比べてより大きなサイズのワーキングセットを利用する．そのため，OMPL2001 ではよりハイパフォーマンスな汎用SMPサーバーなどの性能を測る際に利用される[8].

今回評価対象として選択した art は，ニューラルネットワークを利用した熱画像の判定アプリケーションである．art アプリケーションは，熱画像をトレーニング画像として入力し，ニューラルネットワークをトレーニングする．その後，図 1 のようなスキャン画像から一部を取り出し，ニューラルネットワークを用いて画像判定を行う．計算した値が閾値を超えたとき，画像がマッチしたと判定する．この判定を取り出す位置を少しずつ変えながら行い，マッチする箇所を探し出す．

2.2 331.art のプログラム構成と CUDA による GPU 並列化の方針

331.art は，メモリ領域の初期化部分，ニューラルネットのトレーニング部分，及びニューラルネットを用いた画像判定部分から構成されている．このうち，SPEC OMP では画像判定部分の計算時間が 99%以上を占める．画像判定部分についてより細かく見ていくと，SPEC OMPL2001 の設定において，最も外側にある scan_recognize()関数内のループが 20000 ループである．その内側で呼び出される match()関数では条件分岐によるループがあり，プログラム中の合計で 25779 回ループ内処理が実行される．

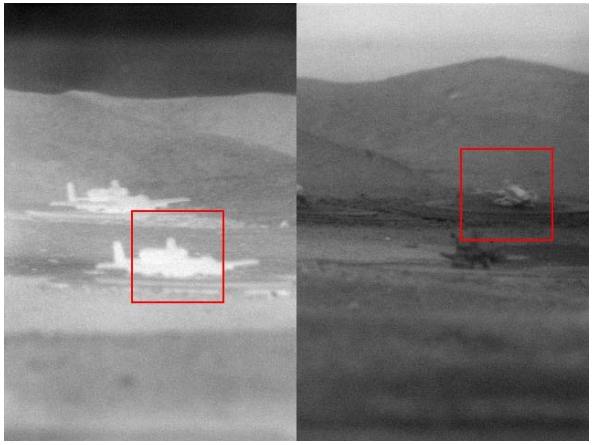

図 1 スキャン画像と特定された熱画像の位置

match()関数からさらに最も内側にある compute_values_match()関数が呼び出され，呼び出される回数は合計で 92599 回である． art のオリジナルコードでは，OpenMP によって最も外側のループである scan_recognize()内の 20000 ループが for ループのダイナミックスケジューリングによって並列化される．この並列化手法では，計算をするスレッドの数だけの計算用配列が必要となる．CPU のスレッド数は今回の CPU 評価用 Xeon X5680 であれば 12 スレッド，一般的なサーバーでも数百スレッドしかなく，1 つのスレッドあたり必要な計算用配列は 1MB 程度なので，メモリの容量は十分足りていることになる．この並列化によって CPU はほぼ完全に並列化され，スレッド数を増やせば増やすほどリニアに速度を向上することが出来る．

ここで，このプログラムの CUDA 化について考える．CUDA の並列化では常にワープ単位では同じ処理が行われる必要があるため，今回のようなループの各イタレーションによって処理結果が大きく違うようなプログラムではダイナミックスケジューリングのような手法を取り入れることは避けるべきである．OpenMP の並列化と同じ箇所で並列化を行おうとすると，スレッド毎の処理はループの回転数によって処理数が変わってしまうため，並列性を大きく下げる原因となってしまう．このため，この 20000 回転のループを 20000 スレッドで置き換えてカーネル実行を行う，といった手法をとることができない．

ここで，画像判定部分のうちどの関数が一番重い処理になっているかを計測した．計測には Xeon X5680 を用いて，1 スレッドでの 1 ループ分の計算から計測を行った．この結果から，compute_values_match()が処理の 99%以上を占め，この計算が art のプ

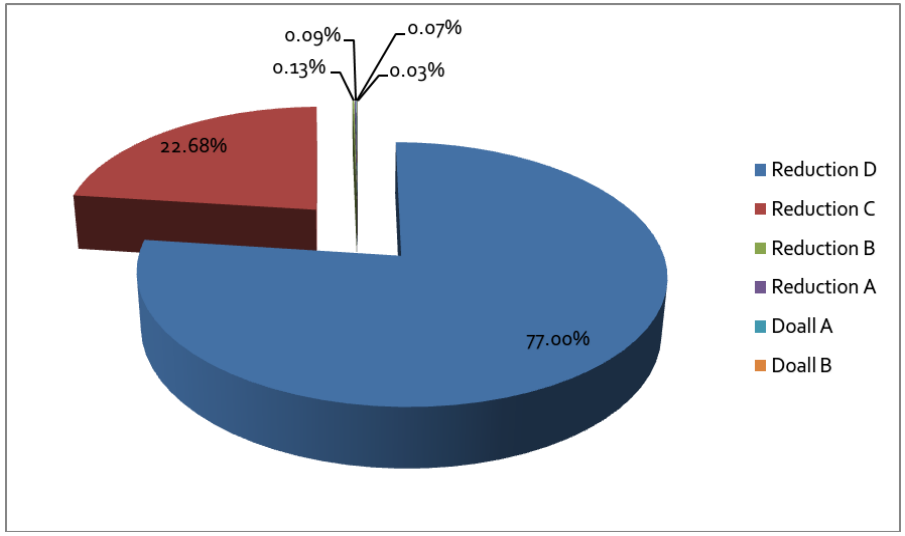

図 2　compute_values_match()関数内の計算時間内訳

ログラムの中で一番時間の掛かる計算だということが特定された．

さらに compute_values_match()関数の構成を細かく分解していくと，この関数は Doall-A, Doall-B, Reduction-A, Reduction-B, Reduction-C, 及び Reduction-D という 6 つの並列化可能ループから構成されていることがわかった．そこで，これらのループに対して CPU の計算による Reduction，Doall の計算時間を測定した．その結果を図 2 に示す．この結果から，Reduction-D および Reduction-C に関して重点的に高速化を行うことで，並列化の高速化を行うことができることがわかった．

以上の結果から, CUDA 化する箇所は Reduction-A～D 及び, Doall-A とした. Doall-B は CUDA 化によって得られる恩恵が少ないことと, 不必要なデータの受け渡しが発生してしまうため CPU 上で行う．

2.3 331.art における漸化計算の CUDA による GPU 並列化

ここで, Reduction-C 及び Reduction-D ループの CUDA 実装方法について触れていく．Reduction C は 2 重ループから構成されており，外側のループは 10000 ループ，内側のループは 2000 ループである．さらに外側のループでは同時に Doall 計算が行われている．Reduction-C は 2000 個のデータ配列から総和を取る計算であり，この計算を 10000 個のデータ配列群に対して行う．CUDA カーネルは外側のループに対して作成し，内側の 2000 ループによる Reduction はカーネル内の for ループによって計算した．

Reduction-D も同様に 2 重ループによって構成されているが，外側のループは 2000 ループ，内側のループは 10000 ループである．また，このループには他の Doall 計算が含まれていない．Reduction-D は 10000 個のデータ配列から総和を取る計算で，この計算を 2000 個のデータ配列群に対して行う．CUDA カーネルは 2 重ループ全体に対して作成し，NVIDIA により提供されている並列漸化計算の実装方法に基づいて計算を行った．この NVIDIA による漸化計算手法に加えて，今回さらに 10000 個のデータによる漸化計算を 1 回のカーネル呼び出しで終了出来るようにした．これは, Reduce 6 による複数加算を 1 スレッドあたり 16 個のデータに対して行い，さらに残りの 625 個のデータに対して 5 個ずつデータを扱うことで実現した．

また，データアクセスが極力少なくなるよう必要なデータのみホスト側へ返すこと，データアクセスの際にはできる限りコアレスアクセスとなるようにすることに留意してカーネルを実装した．

また，データサイズを変更することによってループ回数が変更される．OMPL2001 では実行時オプション objects=2000 によって 2000 個の漸化計算を行うように設定されているが，OMPM2001 では objects=1000 となっており，半分のデータサイズとなる．さらに，SPEC2000 の art では objects=10 で動作させている．これらの中間サイズについても手動で設定を行い，データサイズと性能向上の関係を評価する．

2.4 最適化限界とメモリ転送帯域

漸化計算はその大半がデバイスメモリからのデータ読み出しとループオーバーヘッドである．基本的に読み出したデータは再利用されず，読み出したデータは 1 回の漸化計算にのみ使われる．このため，いかにループオーバーヘッドを隠蔽し，最大メモリ転送帯域速度でデータの読み出しを行うかということが最適化の焦点となる．

読み出すデータ総量は 10000×objects×2×8Byte であるため，計算時間で割ることによって 1 秒当たりのメモリ転送速度を調べることができる．なお，Tesla C2050 の最大メモリ転送帯域は 144GB/s である．

3. 性能評価

3.1 評価手法

評価アプリケーションの SPEC OMPL2001 331.art を用いる．331.art の実行にあたってそれぞれのデータセットの ref に記述された設定を用いている．

CPU と GPU の性能評価では, 331.art 全体の処理を通した処理時間の計測を行った．時間計測には，SPEC OMPL2001 331.art により用意されている既存の機能を使用している．このため，GPU の時間計測は，GPU のイニシャライズや GPU へのデータ転送などに掛かる時間も全て含めた時間となっている．

また，データセットを変えた時の性能評価に関しては，SPEC OMPL2001 により用意されているプログラムを利用して，各データセットの実行時オプションによりデータサイズを変化させている．これは，OpenMP 化されているという点以外では，データサイズ毎に配布されているプログラムに違いが無い為である．データセットとして予め用意されていない中間サイズのデータセットに関しては，実行時オプションを変更することによりサイズを変更している．また，その他のオプションに関しては SPEC OMPM2001 と同じものを使用する．

漸化計算のデータセットによる性能向上評価では Reduction-D の計算のみに着目し，Reduction-D を 1 回計算するのに掛かる時間を計測した．また，Reduction-D の計算時間からメモリアクセス転送速度を求め，メーカーが公表する理論値との比較を行った．

評価環境は，CPU が Intel Xeon X5680×2，GPU が NVIDIA Tesla C2050 である．それぞれの性能は図 3，図 4 の通りである．

CPU のみでの計算では，基本的に OpenMP オプションをつけて測定する．CUDA での計算では，OpenMP オプションをつけずに測定する．

コンパイラには，CPU でのコードは Intel C Compiler 12.0 を使用する．また，CUDA コードのコンパイルは nvcc を用いる．どちらのコンパイラも-O3 オプションをつける．CPU コードは最大限の速度向上を得るため，-parallel オプション，-xhost オプションをつけてコンパイルする．また，CUDA コンパイラのオプションとして-arch=sm13 を指定する．このオプションをつけることで倍精度計算がサポートされる．本論文の評価では，全て倍精度での計算を行う．

CPU のみでの計算では，基本的に OpenMP オプションをつけて測定する．CUDA での計算では，OpenMP オプションをつけずに測定する．

コア数	6
クロック周波数	3.33GHz
キャッシュサイズ	12MB
最大メモリ転送帯域	32GB/s

図 3　Xeon X5680 の諸元

CUDA コア数	448
CUDA コア周波数	1.15GHz
最大メモリ転送帯域	144GB/s

図 4　Tesla C2050 の諸元

コンパイラには，CPU でのコードは Intel C Compiler 12.0 を使用する．また，CUDA コードのコンパイルは nvcc を用いる．どちらのコンパイラも-O3 オプションをつける．CPU コードは最大限の速度向上を得るため，-parallel オプション，-xhost オプションをつけてコンパイルする．また，CUDA コンパイラのオプションとして-arch=sm13 を指定する．このオプションをつけることで倍精度計算がサポートされる．本論文の評価では，全て倍精度での計算を行う．

3.1 CPU と GPU の性能評価

331.art 全体での処理時間の計測を行い，CPU と GPU の性能を評価した．**エラー! ブックマークが自己参照を行っています。**にその結果を示す．SPEC OMPL2001 において OpenMP によって 12 コアで並列化された場合の処理時間と最適化された 1CPU+1GPU の処理時間を比較すると，1.90 倍の性能向上が得られた．

この図において，SPEC OMPL2001 における最適化されていない 1CPU+1GPU による CUDA の実行処理時間は，12 コアで実行される処理時間よりも 4.81 倍長く掛かり，遅くなってしまった．つまり，漸化計算を多く含むプログラムにおいて単純に CUDA による漸化計算を実装するだけでは CPU の速度より速く計算することは出来ず，漸化計算の最適化が CPU の速度を超えるには必要不可欠だということがいえる．

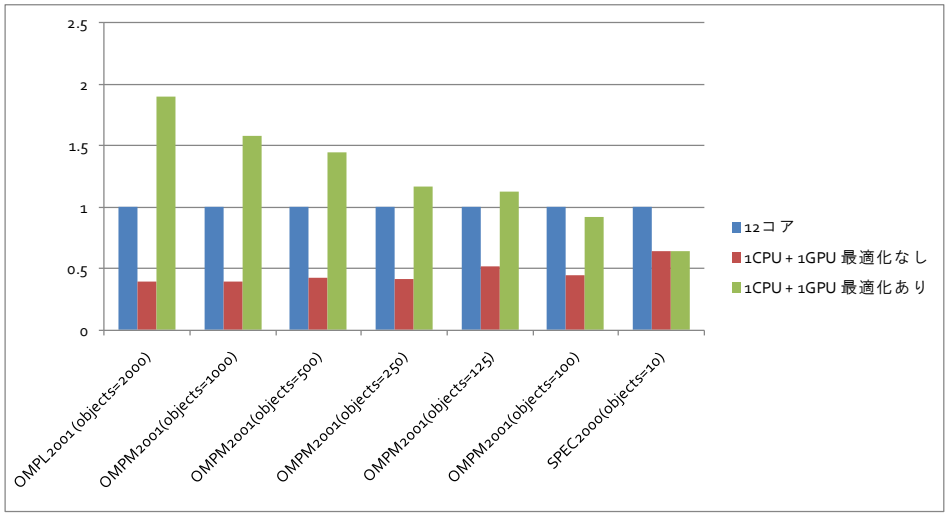

図 5　12 コアを基準とした art アプリケーション全体の速度向上率

また，どの大きさのデータサイズまでなら性能を出すことが出来るか，という点について検討する．予め用意されている SPEC OMPL2001, SPEC OMPM2001, SPEC2000 のデータセットによると，それぞれのデータセットにおける 12 コアの速度で正規化すると，1CPU+1GPU の速度はそれぞれ 1.90 倍，1.58 倍，0.64 倍であった．この結果から，さらに SPEC OMPM2001 のデータサイズ objects=1000 から object = 500, 250, 125, 100 について計測を行った．

この図から，実行時オプション objects=100 以上でなければ 12 コアの速度より遅くなってしまうことが分かる．漸化計算を行う配列は 10000×objects×2 であるため，200 万ほどのデータを持つ漸化計算でなければ高並列化された CPU での実行より遅くなってしまう．

3.2 Reduction-D におけるメモリ転送帯域の比較

対象を Reduction-D に絞り，Reduction-D を 1 回計算するのに掛かる時間を計測した．このときの結果を図 6 に示す．最適化なしと書かれている項目以外は，全て漸化計算について最適化された CUDA カーネルを利用している．

OMPL2001 の計測において，メモリ転送速度は 143.697GB/s であり，理論値の 144GB/s とほぼ同等の値となった．また，OMPM2001 の objects=1000 においても 139.514GB/s と，理論値に近い値が出ている．

しかし，objects の値が小さくなるにつれてメモリ転送速度が遅くなっていった．objects=100 の時点でメモリ転送速度は 90.253GB/s まで下がり，この値は理論値の 0.63 倍である．また，図 5 における objects=2000 と objects=100 における 12 コアとの比較はそれぞれ 1.90 倍と 0.92 倍で，この 2 つを比べると objects=100 のときの比率は 0.48 である．これらの結果より，メモリ転送速度の低下が 331.art アプリケーション全体の速度に影響を及ぼしていることが確認された．

3.3 メモリ転送速度と CUDA カーネル指定時のブロック数

objects の値が低下することによってメモリ転送速度が低下することが 3.3 節の結果により分かった．今回のプログラムでは 1 つのブロックが 10000 個のデータの漸化計算を行うため，objects の値がブロック数に相当する．つまり，メモリ転送速度の低下は十分な数のブロックが指定されていないことから起こっていると思われる．十分な数のブロックが指定されていない場合，デバイスメモリからのデータ転送による待ちが発生した際にループ分岐計算や算術計算を別のブロックで行うことが出来なくなり，メモリ転送速度が低下する．この問題に対応するには，漸化計算を複数のブロックに分割することで 1 つのブロックあたりの計算量を減らし，全体のブロック数を増やす方法が考えられる．しかしこの場合，各ブロックで漸化計算を行うまでの初期化プロセスが発生したり，分割したブロックを統合するブロックが必要になったりする問題が出てくる．

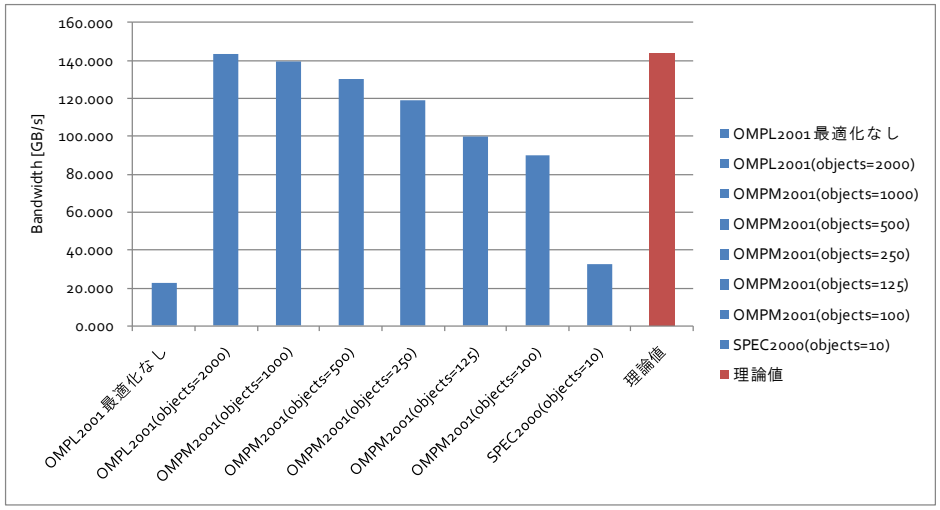

図 6　Reduction D におけるメモリ転送速度

実際に 1 つのブロックを 25 個のブロックに分割し，さらに 25 個のブロックによる Reduction を行うという方法をとったところ，メモリ転送速度は objects=100 のとき 83.370GB/s となった．これは，ブロックの分割を行った場合の 0.92 倍である．ブロックの分割を行うと，デバイス上でさらにカーネルを呼び出す処理が必要になるためオーバーヘッドが増えてしまう．さらに，分割の方法によっては上手く処理を行うためにダミーのデータを追加しなければならず，余計な処理が追加されてしまう．今回のプログラムに関しては，データ数が少なかった場合にブロック数を分割するという方法では速度向上することが出来なかった．

4. まとめ

本稿では，SPEC OMPL2001 331.art を対象として，漸化計算（Reduction）が処理の大半を含むプログラムにおいて GPGPU 計算における漸化計算の最適化を行い，汎用 SMP 用ベンチマークとして高並列化された CPU での実行との比較を行った．12 コアを利用した CPU との比較においては， CUDA による最適化されたプログラムによって 1.90 倍高速に動作させることができた．

また，データセットによる性能向上比較として複数のデータセットによる計測を行い，200 万ほどのデータを持つ漸化計算において高並列化された CPU での実行と同等

の実行時間となることが確認できた.

今回の結果により, 実際に提供されている SMP ベンチマークプログラムにおいても GPU による計算が高並列化された CPU の計算より高速に動作することが確認出来た. 今後は, より様々な特徴を持ったプログラムについて検討を進める予定である.

参考文献

1) http://www.cisco.com/web/JP/news/pr/2010/025.html
2) 和田康孝 等, "ヘテロジニアスマルチコア上でのスタティックスケジューリングを用いた MP3 エンコーダの並列化", 情報処理学会論文誌コンピューティングシステム, Vol. 1, No. 1, pp.105-119, Jun. 2008.
3) Y.Yuyama, et al. , "A 45-nm 37.3 GOPS/W Heterogeneous Multi-Core SOC", IEEE International Solid State Circuits Conference (ISSCC 2020), Feb. 2010
4) V.W.Lee, et al. , “Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU”, Proc. of Intl. Symposium on Computer Architecture (ISCA ‘10), 2010
5) W.Yi, et al. , “A Case Study of SWIM: Optimization of Memory Intensive Application on GPGPU”, Proc. of 3rd Intl. symposium on Parallel Architectures, Algorithms and Programming, Dec. 2010
6) G.Wang, et al. , “Program Optimization of Stencil Based Application on the GPU-accelerated System”, IEEE Intl. symposium on Parallel and Distributed Processing with Applications, Aug. 2009
7) G.Ruetsch, et al. , “A CUDA Fortran Implementation of BWAVES”, PGI Insider, Sep. 2010
8) H.Saito, et al. , “Large System Performance of SPEC OMP2001 Benchmarks”, Proc. of the workshop on OpenMP (WOMPEI 2002), May. 2002