2008－ARC－179 (14)
2008／8／6

Cell およびGPGPU の性能比較評価

西村涼平[†1] 菅原豊[†1]
入江英嗣[†1] 平木敬[†1]

近年，半導体の製造技術の向上によって，1チップに多数のプロセッサコアを集積することが可能になった．これを実現するために，多くの場合，チップを計算能力当たりで簡素にできるSIMDアーキテクチャが採用されている．

この流行に沿ったアーキテクチャとして，Cellプロセッサや GPU が挙げられる．この2つのアーキテクチャは，マルチコアで SIMD という点では共通しているものの，細部においては様々な差がある．我々は，これらの差がメモリレイテンシの隠蔽やプログラミングの複雑さなどにおいてどのように表れてくるかを，行列積，FFT，ソーティング，そして ZIP ファイルのパスワードクラッキングの4つのアプリケーションを使って調べた．

Comparison and Evaluation of Performance of Cell and GPGPU

Ryōhei NISHIMURA,[†1] Yutaka SUGAWARA,[†1]
Hidetsugu IRIE[†1] and Kei HIRAKI[†1]

Recently, improvement of manufacturing technology of semiconductors has enabled to accumulate a lot of processor cores to one chip. In order to realize this, in a lot of cases, the SIMD architecture that can enable a chip to be simple per computing ability is adopted.

We mention the Cell processor and GPUs as the architectures in accordance with this trend. These architectures are common in points of multicore and SIMD, but they are different in various particulars. We investigated how these differences appear in concealment of memory latency and complexity of programming using the four applications of matrix multiplication, FFT, sorting and password cracking of ZIP files.

1. 背　景

近年，プロセッサのスカラ処理性能の向上が難しくなってきている．そのため，プロセッサの性能を向上させるために，マルチコア化を進め，並列処理性能を向上させるというのが，近年のプロセッサの流行である．また，同様の理由により，比較的単純なアーキテクチャで高い性能を出すことができるSIMD命令を充実させ，ベクトル処理性能を高めるというのもまた，最近のプロセッサの流行である．

これらの流行を特に進めているアーキテクチャとして，Cellプロセッサ[7]や各種GPUが挙げられる．これらのプロセッサは，せいぜいピーク性能が100 GFLOPS強に留まっている汎用CPUに対して，数百〜一千GFLOPSという高いピーク性能を売り物にしている．これらを用いた研究も近年盛んであり，これらを比較した研究というのはいくつか見受けられる[8)5)]が，あまり多くはないのが現状である．

一方，プロセッサの処理性能の進歩に比べ，メモリの処理性能の進歩は遅い．そのため，先に挙げたようなマルチコアSIMDプロセッサが高いピーク性能を持つようになっても，メモリが性能のボトルネックになるケースが多くなってきている．メモリのボトルネックの原因の一つである，レイテンシの問題に対して，計算の裏でメモリ転送を行う方法や，インターリーブ[6]などの方法が考えられている．マルチコアSIMDプロセッサでは，先に挙げたCellは前者を採用し，また，GPUのGeForce 8000シリーズ[3]は後者を採用し，メモリレイテンシの隠蔽を図っている．

2. Cellプロセッサ

Cellプロセッサは，テレビゲーム機に搭載することを主な目的として，高い処理性能を目指したマルチコアSIMDプロセッサである．Cellプロセッサは，汎用CPUのPPEと呼ばれるコアと，複数のSPEと

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, the University of Tokyo

クロック	3.2 GHz
SPE の数	6 基
ローカルストア	各 256 KB
メモリ容量	256 MB
メモリ帯域	25.6 GB/秒
OS	Fedora 8
開発環境	CellSDK 3.0.0.3

表 1 Cell プロセッサの実験環境

呼ばれる SIMD 演算に特化したコアから成り立っている．SPE はそれぞれ 256 KB の「ローカルストア」と呼ばれるスクラッチパッドメモリを持ち，メインメモリへのアクセスはローカルストアを介してのみ行うことができる．「ダブルバッファリング」と呼ばれる，ローカルストア上にバッファを 2 枚用意しておき，片方のバッファがメインメモリにアクセスしている裏でもう片方のバッファで計算を行うという方法で，メモリレイテンシを隠蔽することができる．SPE は 128 bit の SIMD 演算を行うパイプラインを持ち，ローカルストアへのアクセスを行うパイプラインと合わせて，2 Way のスーパースカラを構成する．これらのスーパースカラパイプラインとは別に，メインメモリとローカルストアの間のデータ転送を行うメモリコントローラを持つ．計算のレイテンシは大きいが，レジスタを 128 本と豊富に持ち，複数のデータに対する処理を並べて書くことで，レイテンシを隠蔽するプログラミングができるようにしてある．

我々は，実験環境として，ソニーコンピュータエンタテイメント社の PLAYSTATION 3 に，Fedora 8 Linux と Cell SDK 3.0.0.3 をインストールして用いた．PLAYSTATION 3 の Cell プロセッサは 3.2 GHz で動作し，Fedora 8 からは 1 基の PPE と 6 基の SPE を扱うことができる．メインメモリとしては XDR メモリを 256 MB 搭載し，25.6 GB/秒の帯域を持つ．SPE のピーク性能は 153.6 GFLOPS である．

3. GeForce 8800

GeForce 8800 は，高い処理性能を謳って 2006 年に登場した GPU であり，GPGPU 用言語である CUDA[3] に対応している．GeForce 8800 は「ストリーミングプロセッサ」と呼ばれる演算ユニットを多数搭載している．ストリーミングプロセッサは 8 つで「ストリーミングマルチプロセッサ」と呼ばれる処理単位を構成し，GeForce 8800 は，このストリーミングマルチプロセッサを単位としてスケジューリングを行う．ストリーミングプロセッサ 1 基に 1 基のスカラ演算器が搭載されているほか，ストリーミングマルチプロセッサに対し 2 基のベクトル/超越関数演算器が

コアクロック	1.5 GHz
ストリーミングプロセッサ	128 基
シェアードメモリ	16 KB × 16
ビデオメモリ容量	768 MB
ビデオメモリ帯域	103.68 GB/秒
ホスト CPU クロック	2.6 GHz
ホスト CPU コア数	2
ホスト CPU L2 キャッシュ	1 MB × 2
メインメモリ容量	4 GB
メインメモリ帯域	12.8 GB
ホスト OS	Fedora 7
ホスト・GPU 間接続	PCI-Express 1.0
開発環境	CUDA 1.1

表 2 GeForce 8800 の実験環境

割り当てられている．ストリーミングマルチプロセッサはそれぞれ 16 KB の「シェアードメモリ」と呼ばれるスクラッチパッドメモリを持つが，ビデオメモリに直接アクセスすることもできる．各ストリーミングプロセッサはマルチスレッディングに対応し，インターリーブを行うことでビデオメモリのレイテンシを隠蔽する．

我々は，実験環境として，GeForce 8800 Ultra を用いた．GeForce 8800 Ultra は 1.5 GHz で動作するストリーミングプロセッサを 128 基搭載し，また，ビデオメモリとして 103.68 GB/秒の帯域を持つ GDDR3 メモリを 768 MB 搭載する．GeForce 8800 Ultra のピーク性能は，576 GFLOPS である．ホストの PC は，CPU がデュアルコアの Athlon 64 X2 5200+ （2.6 GHz，L2 キャッシュは 1 MB × 2），メモリがデュアルチャネルの DDR2-800 （帯域は 12.8 GB/秒） 4 GB，OS が Fedora 7 Linux であり，GeForce 8800 Ultra との間は PCI-Express 1.0 で接続されている．開発環境には，CUDA 1.1 を用いた．

4. 行 列 積

巨大な行列の乗算は，コンピュータのベンチマークにしばしば用いられるアプリケーションである．また，スーパーコンピュータのランキングである TOP500 の指標に用いられる Linpack ベンチマークの中核をなす計算でもある．このアプリケーションの特徴として，プロセッサのピーク性能に近い性能が出るということが挙げられる．

行列積計算のアルゴリズムには，ブロッキングのアルゴリズム[1] を用いた．巨大な行列を，高速なメモリに収まる大きさの小さなブロックに分割し，分割統治法によって答えを求めるアルゴリズムである．

我々は，2048×2048 の大きさの行列の乗算にかかった時間を測定した．

	Cell	Cell（最適化無し）	8800	8800（メモリ転送抜き）
実行時間（秒）	0.138	1.53	0.221	0.130
性能（GFLOPS）	124	11.2	77.7	132
コード行数	306	109		37

表 3 行列積の測定結果

4.1 Cell プロセッサのプログラム

ブロックの大きさは 64 × 64 に設定した．これは，Cell プロセッサのメインメモリ転送命令が一度に扱えるデータの大きさが 16 KB であるところから決まった値である．最も内側のループはアンローリングを行った．

また，最適化を行ったプログラムとは別に，ダブルバッファリングやループアンローリングといった最適化を行わないプログラムも用意し，両者を比較した．

4.2 GeForce 8800 のプログラム

ブロックの大きさは 32 × 32 に設定した．これは，シェアードメモリの大きさより決まった値である．最も内側のループはアンローリングを行った．

4.3 測定結果

Cell プロセッサでは，実行に 0.138 秒かかり，このときの性能は 124 GFLOPS である．これは，ピーク性能の 81.4%が出ている計算になる．これを見る限り，Cell プロセッサの性能をかなり引き出せているように見えるが，最適化を行わないプログラムの場合，実行に 1.53 秒かかってしまう．このときの性能は 11.2 GFLOPS であり，ピーク性能の 7.31 %しか引き出せていない計算になる．このように，最適化を行わないと，Cell プロセッサのパフォーマンスを十分には引き出すことができないのがわかる．最適化プログラムの SPE プログラムの行数は 306 行であり，非最適化プログラムの SPE プログラムの行数は 109 行である．

GeForce 8800 では，実行に 0.221 秒かかった．これより，性能は 77.7 GFLOPS と求まる．ただし実際は，メインメモリとビデオメモリの間の転送にある程度時間がかかっている．この転送を除いた実行時間は 0.130 秒であり，性能は 132 GFLOPS である．GeForce 8800 の単精度浮動小数加算器と乗算器の数の比は 1 : 2 であるので，加算と乗算の比が 1 : 1 の行列積計算で演算器が常に動き続けると仮定した場合，ピーク性能は 384 GFLOPS となる．ストリーミングプロセッサはスーパースカラでないので，計算とメモリ転送を同時に行えず，我々のプログラムでは積和算とメモリ転送の比がおよそ 2 : 3 であるので，メモリ転送を除いたピーク性能は 153.6 GFLOPS となる．これに対する，メモリ転送を除いた性能の比は，86.0 %であり，GeForce 8800 の性能を引き出せていることがわかる．プログラムからホスト部分のコードを除いた行数は 37 行であり，Cell プロセッサに比べて簡潔な記述で性能を引き出せていることがわかる．

5. FFT

高速フーリエ変換 (FFT)[?),4)] は，画像，音声，動画などの圧縮のアルゴリズムや，コンピュータのベンチマークとして使われる円周率計算などに広く使われる計算である．我々は，複素数 2^{19} 要素からなる 1 次元 FFT の計算を行った．

アルゴリズムとして，Cooley-Tukey 型のアルゴリズムに，行列転置のアルゴリズム[?)] を加えて用いた．行列転置のアルゴリズムの基数は，各アーキテクチャのスクラッチパッドメモリの大きさに合わせて設定した．

我々は，1000 回 FFT の計算を行い，計算にかかった時間の平均の測定した．

5.1 Cell プロセッサのプログラム

行列転置のアルゴリズムを使って 3 回に分けて計算をし，基数は 1 回目が 2^7，2 回目が 2^6，3 回目が 2^6 というように設定した．三角関数のテーブルは，添字のビットの桁を反転させたものを用意し，3 つのテーブルを引いた値から加法定理で元の三角関数の値を計算できるようにした．また，添字をシフトしなくてもよいように，あらかじめ添字をシフトしたテーブルを数パターン用意した．行列積同様，最も内側のループはアンローリングした．

FFT のデータと，計算用の一時格納用の領域は，Linux のヒュージページの機能を用いて確保した領域に置く．ヒュージページの機能を使わないと，計算の途中で TLB ミスが頻発して，性能低下の原因になる．ヒュージページを使う場合と使わない場合とで比較を行う．

5.2 GeForce 8800 のプログラム

行列転置のアルゴリズムを使って 10 回に分けて計算を行った．基数は，1 回目から 9 回目までが 4，10 回目が 2 である．三角関数のテーブルは，Cell プロセッサのものと同じく添字のビットの桁を反転させたものを用いたが，テーブルの数は，Cell プロセッサでは 3 つであったのに対して，GeForce 8800 では 2 つとした．また，こちらはあらかじめ添字をシフトした

	Cell	Cell（スモールページ）	8800	8800（メモリ転送抜き）
実行時間（ミリ秒）	1.48	1.65	4.55	1.78
性能 (GFLOPS)	33.7	30.4	10.9	28.0
メモリ転送量 (GB/秒)	15.8	14.3	–	43.9
メモリ転送量対ピーク比	.619	.558	–	.423
コード行数		710		467

表 4　FFT の測定結果

テーブルは用意せず，テーブルを引くときに添字をシフトした．三角関数のテーブルは，GPU 側から変更できない代わりにキャッシュメモリを持ち高速な「コンスタントメモリ」に転送して計算した．

5.3 測定結果

Cell プロセッサでは，実行に 1.48 ミリ秒かかった．これより，性能は 33.7 GFLOPS と求まる．また，メモリ転送量を求めると，15.8 GB/秒となる．メモリ転送はピークの 61.9 % であり，ここがボトルネックになっている．一方，ヒュージページを用いない場合は，実行時間は 1.65 ミリ秒と，遅くなっていることがわかる．このときの性能は 30.4 GFLOP で，メモリ転送量は 14.3 GB/秒，ピークの 55.8 %に落ち込んだ．FFT の計算は，メモリに飛び飛びにアクセスするので，ヒュージページの効果が分かりやすく表れた．SPE プログラムの行数は 710 行である．

GeForce 8800 では，FFT を 4.55 ミリ秒で計算した．ここから性能を計算すると，10.9 GFLOPS となる．ただし，実際はメインメモリとビデオメモリ間の転送がボトルネックになっていて，転送を除いた実行時間は 1.78 ミリ秒であり，性能は 28.0 GFLOPS である．メモリ転送量は，ピークの 42.3 %である 43.9 GB/秒であり，ここがボトルネックである．性能が Cell プロセッサに比べて伸びないのは，Cell プロセッサが全データ領域を 3 回読んで書いているのに対して，GeForce 8800 はスクラッチパッドメモリが小さいため，全データ領域を 10 回も読んで書いているために，メモリ転送量が増えてしまっているのが原因である．GPU 部分のプログラムの行数は 467 行である．

6. ソーティング

ソーティングは，古くからさまざまなアルゴリズムが研究されている問題である．場合に応じてアルゴリズムを使い分けることが，高速に問題を解く鍵である．我々は，Cell プロセッサと GeForce 8800 という，異なったアーキテクチャでソーティングを行う場合に，それぞれどのようなアルゴリズムが最適かを探し，ここからアーキテクチャの特徴を炙り出した．

我々が扱ったソーティングアルゴリズムは主に，計算量が $O(N \log N)$ のマージソートと，計算量が $O(N(\log N)^2)$ のバイトニックソート[2] の 2 つである．これらのアルゴリズムの補助として，分割統治法の問題サイズが小さいときに，計算量が $O(N^2)$ の選択ソートを併用した．

我々は，2^{20} 要素の単精度浮動小数点数のソートを行い，かかった時間を測定した．

6.1 Cell プロセッサのプログラム

選択ソートは，SIMD 命令を用いて，4 つ飛ばしのデータがソートされるようにプログラムした．4 つ飛ばしでソートされたデータは，マージソートで整列される．

マージソートのプログラムは，要素数が 64 以下（SIMD 命令を用い，256 要素を処理）の場合は選択ソートを用いた．また，バイトニックソートのプログラムは，同じく要素数が 64 以下の場合は選択ソートを用いた．

マージソート，バイトニックソート共に，並行するデータに対する命令を並列に並べ，命令のレイテンシを隠蔽できるようにプログラミングした．

6.2 GeForce 8800 のプログラム

マージソートのプログラムは，選択ソートを用いずに，最初からマージソートを用いた．また，バイトニックソートのプログラムは，要素数が 8 以下の場合は選択ソートを用いた．

バイトニックソートの比較の距離がスレッド数である 512 未満の場合は，データをシェアードメモリに移してから計算を行った．

6.3 測定結果

Cell プロセッサによるマージソートは，105 ミリ秒で計算を終了した．メモリ転送量は，ピークの 4.07 %である 1.04 GB/秒であり，ここはボトルネックではない．SPE は SIMD 計算しかできないので，スカラ計算が必要になるマージソートでは，SPE の性能を生かしきることはできないことがわかる．SPE プログラムの行数は 901 行である．

GeForce 8800 によるマージソートの実行時間は，518 ミリ秒であった．メインメモリとビデオメモリの間の転送を除くと，実行時間は 459 ミリ秒であり，メモリ転送量は，ピークの 0.328 %である 0.340 GB/秒であった．メモリ転送量に比べて性能が出ていない理

		Cell	8800	8800（メモリ転送抜き）
マージソート	実行時間（ミリ秒）	105	518	459
	メモリ転送量 (GB/秒)	1.04	–	0.340
	メモリ転送量対ピーク比	.0407	–	.00328
	コード行数	901		73
バイトニックソート	実行時間（ミリ秒）	103	101	24.6
	メモリ転送量 (GB/秒)	13.9	–	55.9
	メモリ転送量対ピーク比	.532	–	.539
	コード行数	1760		95

表 5 ソートの測定結果

由は，ソートが進んでくるとプログラムの並列性が落ち，GeForce 8800 が苦手な並列性の低い計算となるからである．GPU 側のプログラムの行数は 73 行である．

Cell プロセッサによるバイトニックソートの実行時間は，103 ミリ秒であった．こちらのメモリ転送量は 13.9 GB/秒であり，ピークの 54.2 %である．こちらはメモリ転送がボトルネックになっているとみてよいだろう．マージソートと違って，SIMD 命令を使ったソートが容易であるので，メモリ帯域を使いきることができた．SPE のプログラムの行数は 1760 行である．

GeForce 8800 によるバイトニックソートは，101 ミリ秒で計算を終了した．メインメモリとビデオメモリの間の転送を除くと，実行時間は 24.6 ミリ秒である．ここから，メモリ転送量は 55.9 GB/秒であり，ピークの 53.9 %である．こちらもメモリ転送がボトルネックとなっていることがわかる．バイトニックソートは，最初から最後まで高い並列性を保ったまま処理ができるので，GeForce 8800 で高速に処理ができる．ホスト側のコードを除いたプログラムは 95 行である．

7. ZIP ファイルのパスワードクラッキング

ZIP ファイルのパスワードクラッキングは，整数演算を用いた暗号の計算を用いたアプリケーションである．ZIP ファイル内の各ファイルのエントリのヘッダを復号化し，8 ビットの CRC と照らし合わせることでパスワードをチェックできるので，これを逆用してクラッキングに使う．ZIP ファイル内に複数のファイルが存在するとき，すべてのファイルのパスワードが同じだと仮定すると，ファイルを 1 つずつ順番にチェックしていくことで，ファイルが 1 つ増えると精度が 8 ビット詳しくなり，現実的な精度でパスワードが求まる．

我々は，5 つのファイルが格納された ZIP ファイルのパスワードを，1 文字から 4 文字まで，95 文字種の範囲で総当たりで調べ，かかった時間を測定した．また，比較のために，GeForce 8800 の実験環境のホストのみで動かしたプログラムも用意する．

7.1 Cell プロセッサのプログラム

SIMD 命令を使い，複数のパスワードを同時にチェックできるようにプログラムを組んだ．また，命令を並列に並べて，命令のレイテンシを隠蔽するテクニックを使った．

7.2 GeForce 8800 のプログラム

他のプログラムでは，ファイルの CRC を見て，間違っていたら次のパスワードに進む，というようなアルゴリズムを使っているが，GeForce 8800 では CRC が合っていても合っていなくても，最後のファイルまで進んでから次のパスワードに進んでいる．これは，GeForce 8800 は，少数のスレッドのみが異なった処理をする，ということが苦手なためである．

7.3 測定結果

Cell プロセッサでは，0.769 秒で計算が終了した．また，GeForce 8800 では，2.33 秒で計算が終了した．Cell プロセッサの SPE プログラムの行数は 555 行，GeForce 8800 の GPU 部分のプログラムの行数は 156 行である．

参考として，GeForce 8800 のホストの Athlon 64 X2 では，計算に 8.42 秒かかる．

8. 結　　論

測定結果を見てみると，どちらのプロセッサも 50 %前後の効率でメモリを使っていることがわかる．計算の裏でメモリ転送を行っても，インターリーブを使っても，メモリの使用効率の点ではそこまで極端な差は出ない．しかし，プログラムの行数では明らかな差が出ているのがわかる．メモリ転送の複雑な指定，ダブルバッファリングの処理などで，Cell プロセッサのプログラムは長くなってしまっている．また，長くなっている原因として，命令のレイテンシの隠蔽のテクニックも挙げられる．一方，GeForce 8800 のプログラムの記述は簡潔である．命令のレイテンシも，インターリーブを行うことで隠蔽できるのが強みである．

これらのことから，GeForce 8800 が採用したイン

	Cell	8800	Athlon 64 X2
実行時間（秒）	0.769	2.33	8.42
コード行数	555	156	–

表 6 ZIP ファイルのパスワードクラッキングの測定結果

ターリーブ方式が，簡潔な記述で高いパフォーマンスを発揮することから，マルチコア SIMD プロセッサのアーキテクチャとして優れていることがわかる．

参 考 文 献

1) W.Abu-Sufah, D.J. Kuck, and D.H. Lawrie. Automatic program transformations for virtual memory computers. In *Proceeding of the 1979 National Computer Conference*, pp. 969–974, June 1979.
2) K. E. Batcher. Sorting networks and their applications. *Proceeding AFIPS Spring Joint Computer Conference*, 1968.
3) I.Buck. Geforce 8800 & nvidia cuda: A new architecture for computing on the gpu. *website of Supercomputing '06 Workshop "General-Purpose GPU Computing: Practice And Experience"*, 2006.
4) JamesW. Cooley and JohnW. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comput. 19*, pp. 297–301, 1965.
5) KevinSkadron DavidTarjan. Multithreading vs. streaming. *MSPC 2008*, March 2008.
6) James Laudon, Anoop Gupta, and Mark Horowitz. Interleaving: a multithreading technique targeting multiprocessors and workstations. *SIGPLAN Not.*, Vol.29, No.11, pp. 308–318, 1994.
7) D. Pham, S. Asano, M. Bolliger, M. N. Day, H.P. Hofstee, C.Johns, J.Kahle, A.Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K.Yazawa. The design and implementation of a first-generation cell processor. In *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, pp. 184–592 Vol. 1, 2005.
8) H. Scherl, B. Keck, M. Kowarschik, and J.Hornegger. Fast gpu-based ct reconstruction using the common unified device architecture (cuda). *Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE*, Vol.6, pp. 4464–4466, 26 2007-Nov. 3 2007.