Vol.2009-ARC-184 No.25
2009/8/5

GeForce GTX 280 vs. Cell

西村 涼平[†1]　入江 英嗣[†1]　平木 敬[†1]

近年，GPU の性能が TFLOPS を超える高いものになっている一方，グラフィックス用途における高性能な GPU の需要が頭打ちになっている．そのため，処理能力とともに，可能な処理の範囲を拡大している GPU を汎用計算に用いる，GPGPU がホットなテーマとなってきている．

一方，この GPGPU と競合するものに，1 チップに多数の SIMD プロセッサを集積した Cell プロセッサがある．こちらも，数百 MFLOPS の処理能力に広いメモリ帯域を持ち，HPC の分野で注目を集めている．

われわれは，最新アーキテクチャの GPU である GeForce GTX 280 と Cell プロセッサを，行列積，FFT，ソーティング，ZIP ファイルのパスワードクラッキング，HPC Challenge の RandomAccess，そして Levenshtein 距離の算出という，6 つのアプリケーションを用いて比較を行った．その結果，一部のアプリケーションを除いて，GeForce GTX 280 の性能が優れていることが示された．

GeForce GTX 280 vs. Cell

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

Recently, on the one hand, performance of a GPU has been higher than a TFLOPS, on the other hand, demand of GPUs of high performance for graphics has peak. Then, GPGPU that uses GPUs increasing in ability and possible range of processing for general-purpose computing has been the hot theme.

On the other hand, there is the Cell processor that accumulates many SIMD processors into one chip as what competes GPGPU. It also has performance of hundreds MFLOPS and wide memory bandwidth and has been paid attension to on a field of HPC.

We compared GeForce GTX 280 of the GPU of the up-to-date architecture and the Cell processor using the 6 applications: matrix multiplication, FFT, sorting, password-cracking of ZIP files, RandomAccess of HPC Challenge and calculation of the Levenshtein distance. As a result, it was shown that the performance of GeForce GTX 280 was superior except the part of the applications.

1. 背　　景

半導体の製造技術は，依然として Moore の法則[15] に従ったペースで進歩している．この進歩に乗る形で，Graphics Processing Unit (GPU) も進歩を続けており，2008 年 6 月にはついにその性能は 1 TFLOPS を突破した．その GPU のアーキテクチャは，1 チップに多数の演算器を集積し，高い並列計算の処理能力を持つ，といったものが主流である．これは，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というハードウェア的な事情が，グラフィックス処理が並列性の高い処理であるというソフトウェア的な事情にマッチしているという背景がある．用途は限定的ながらも，このように目覚ましい性能向上を遂げている GPU であるが，その性能をグラフィックス処理のみならず，汎用計算にも用いようという General Purpose GPU (GPGPU)[13] が近年注目されている．

この GPGPU に対抗するものとみなされているのが，1 チップに最大 9 基の 4 way SIMD 演算器を集積した Cell プロセッサ[17] である．これは汎用計算に使われることを狙って設計されたプロセッサであり，GPU に匹敵する高い処理能力を持つ．この Cell プロセッサは，スーパーコンピュータのランキングである TOP 500 における 2008 年 11 月付での世界最速のスーパーコンピュータ，「Roadrunner」に使われているプロセッサである．

我々は，最新アーキテクチャの GPU である GeForce GTX 280 を Cell プロセッサと比較して，より優れていることを示し，GPGPU が汎用計算において十分な性能を持っていることを示す．

2. 各プロセッサについて

2.1 GeForce GTX 280

GeForce GTX 280 は，2008 年 6 月に発表された，1 チップに高い処理能力を詰め込む方針で設計された，NVIDIA 社のハイエンド GPU である．GPGPU 言語である CUDA[7] に対応し，簡単に汎用計算に転用することができる．Streaming Multiprocessor (SM) と呼ばれるユニットを 30 基搭載し，これらはそれぞれ Streaming Processor (SP) と呼ばれる演算ユニットを 8 基搭載する．SP 1 基が，整数演算，および積和演算の可能な単精度浮動

†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

	GeForce GTX 280	Cell on PS3 Linux	Core 2 Quad Q9400 @ホスト環境
クロック周波数 (GHz)	1.296	3.2	2.66
コア数	30 SMs	7 SPEs（1 基は OS が使用）	4
1 Hz、1 コアあたりの浮動小数点演算処理数	24（1 SM あたり）	8	8
ピーク浮動小数点演算性能 (GFLOPS)	933.12	153.6（PPE 及び SPE 1 基を除く）	85.12
1 コアあたりの実行可能スレッド数	512	1	1
1 コアあたりの浮動小数点レジスタ	32 bit × 16384	128 bit × 128	128 bit × 16（プログラムから見える SSE レジスタ）
ピークメモリ帯域 (GiB/s)	141.696	25.6	12.8
メモリ容量 (MiB)	1024	256	4096
1 コアあたりのオンチップメモリ容量 (KiB)	16	256	32 (L1 キャッシュ)
コア間接続	クロスバースイッチ	リングバス	バス
消費電力 (W)	236	64[⋆1]	95
OS 及びデバイスドライバ	NVIDIA 公式ドライバ 185.18.08	Linux 2.6.27.21	Linux 2.6.27.21
コンパイラ	CUDA 2.2	GCC 4.1.1	GCC 4.3.2

表 1 各プロセッサの比較

小数点演算を行う演算器を持つほか，SM 毎に 2 器の超越関数演算器と 1 器の倍精度浮動小数点演算器を持つ．SM はそれぞれ 16 KiB の Shared Memory と呼ばれるスクラッチパッドメモリを持つが，ビデオメモリへのアクセスは，Shared Memory やキャッシュを介さずに，直接アクセスする形となる．なお，読み込み専用の Constant Memory 及び Texture Memory と呼ばれるビデオメモリの領域は，キャッシュを介したアクセスとなる．各 SM は，レジスタファイルを共有するマルチスレッディングに対応し，インターリーブ[11] を行うことでメモリアクセスのレイテンシを隠蔽する．

CUDA によって生成された命令は，SM を単位にスケジューリングされ，4 クロック分繰り返して実行される．つまり，CUDA の命令発行の単位は 32 スレッドであり，この単位を Warp と呼ぶ．この仕様により，SM あたりのスレッドの数が 32 の倍数のとき，最も効率よく計算を行える．メモリアクセスは，Half Warp と呼ばれる 16 スレッドを単位として行われ，ビデオメモリは，32 bit，64 bit，または 128 bit 単位で Half Warp 内のスレッドが連続してアクセスしたとき，Shared Memory は，Half Warp 内のスレッドが，32 bit 単位のバンクに対して競合しないようにアクセスしたとき，それぞれ高速にアクセスできる．また，Shared Memory と Constant Memory は，Half Warp 内のスレッドが同じデータを読み出す時も，高速にアクセスすることができる．一方，Texture Memory は，Warp 内のスレッドが同時にアクセスするメモリアドレスが近い場合に，高速にアクセスすることができる．

GeForce GTX 280 の詳しい仕様は，表 1 に記してある．これを，同じ表に示してある，Fedora 10 をインストールしてあるホスト環境に接続して，実験を行った．ホストと GPU との間は，PCI Express 2.0 x16（ピーク帯域 8 GB/s）で接続されている．

2.2 Cell プロセッサ

Cell プロセッサは高い処理能力を目指して設計された，ヘテロジニアスマルチコアプロセッサであり，HPC，テレビゲーム，PC のアクセラレーションなどに広く使われている．Cell プロセッサは，PPE と呼ばれる汎用 CPU コアと，SPE と呼ばれる，複数の 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 KiB の Local Store と呼ばれるスクラッチパッドメモリを持ち，SPE のメインメモリへのアクセスは，メインメモリ・Local Store 間の DMA 転送によってのみ行われる．SPE のパイプラインは，128 bit の SIMD 演算をスループット 1 クロックで処理できる．整数演算及び浮動小数点演算を行うパイプラインと，Local Store アクセス及びシャッフルを行うパイプラインとで，2 Way のスーパースカラを構成する．

メモリレイテンシを隠蔽する方法の一つとして，Local Store 上にバッファを 2 枚用意しておき，片方のバッファで DMA 転送を行っている裏で，もう片方のバッファのデータに対して計算を行うという，ダブルバッファリングのテクニックがある．また，SPE の命令スケジューラはインオーダ実行であり，単精度浮動小数点演算のレイテンシが 7 クロックであるなど，SPE の命令レイテンシは全体的に長いため，SPE の性能を引き出すためには，複数のデータに対する依存性の無い命令を明示的に並べるソフトウェアパイプラインのテクニックを用い，命令レイテンシを隠蔽する必要がある．

我々の実験環境は，ソニーコンピュータエンタテイメント社の PLAYSTATION 3 に，Fedora 10 と IBM Cell SDK 3.1 をインストールして用いた．詳しい仕様は，前節で示した表に記してある．

⋆1 4 GHz で 80 W[21] であることより，消費電力はクロック周波数に比例するとして計算

3. 実験に用いたアプリケーション

3.1 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．

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．

行列積計算のアルゴリズムには，GeForce GTX 280 と Cell プロセッサで，共にブロッキングのアルゴリズム[4] 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．

GeForce GTX 280 のプログラムにおいては，片方の行列を転置した上で計算を行う，Volkov のアルゴリズム[20] を利用した．ブロックの大きさは，Volkov 氏のプログラムで使われていた，64×16 の値を利用した．1 Streaming Multiprocessor あたりのスレッド数は，Volkov 氏のプログラムと同じ 64 である．

Cell プロセッサのプログラムでは，ブロックの大きさを 64×64 に設定した．これは，ピーク性能に近い速度で処理を行っても十分な大きさであるとともに，Cell プロセッサの DMA メモリ転送命令が一度に扱えるデータの大きさが 16 KiB であるところから，決まった値である．

3.2 FFT

高速フーリエ変換 (FFT)[9] 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．

我々は，1000 回 FFT の計算を行い，計算にかかった時間の平均を測定した．また，比較用に，GeForce GTX 280 のホスト環境でも，計算を行って時間を測定した．

アルゴリズムとして，Stockham[19] アルゴリズムを用いた．これは，大きな FFT を複数回のの小さな FFT に分割して計算するアルゴリズムである．小さな FFT の計算には，Cooley-Tukey 型のアルゴリズムを用いた．GeForce GTX 280 では，4 回の 2^4 要素の FFT と，1 回の 2^3 要素の FFT に分割して計算を行った．Cell プロセッサでは，1 回の 2^7 要素の FFT と，2 回の 2^6 要素の FFT に分割して計算を行った．

GeForce GTX 280 のプログラムは，1 Streaming Multiprocessor あたりのスレッド数を、2^4 要素の FFT では 128、2^3 要素の FFT では 256 に設定した．各スレッドは，要素をレジスタにロードし，小要素の FFT を行った後，Shared Memory を用いて，スレッド間でシャッフルを行い，データをストアする．三角関数の値は，超越関数演算器が生成する値を用いたため，高速ではあるが，精度がやや悪いのが難点である．

Cell プロセッサのプログラムは，64 KiB のバッファを 3 枚用意し，バッファ0 とバッファ 1 をロードと FFT のメイン処理に使い，2^6 要素の FFT では計算に使ったバッファをそのままストア，2^7 要素の FFT では，その後シャッフル処理を行い，出力をバッファ2 に格納し，このバッファ2 をストアする．2^7 要素の FFT では，64 本のレジスタを用いて，8 並列で 16 要素の FFT を行った後，一旦レジスタを Local Store に書き戻し，再び 64 本のレジスタを用いて 16 並列で 8 要素の FFT を行い，もう一度 Local Store にレジスタを書き戻し，最後にシャッフル処理を行って，Local Store をメインメモリに書き出す．2^6 要素の FFT では，64 本のレジスタを用いて，2 並列で 64 要素の FFT を行う．三角関数の計算は，添字のビット桁を反転させたテーブルを 1 つと，2 つに分割した反転させないテーブルを，PPE で計算した後 SPE に転送して準備し，3 つのテーブルの値を加法定理で合成した．なお，計算に使うメインメモリの領域は，TLB サイズ 16 MiB のメモリとして確保した．

3.3 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，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．また，比較用に，FFT 同様にホスト環境でも計算を行った．

GeForce GTX 280 と Cell プロセッサとで共に，ソーティングアルゴリズムは，計算量が $O(N(\log N)^2)$ のバイトニックソート[6] を主に用いた．計算量は $O(N \log N)$ のアルゴリズムに比べて不利であるが，並列化に向いているアルゴリズムであり，かつ係数も小さいため，今回の問題サイズでは，我々が調べた中で最も有利なアルゴリズムであった．バイトニックソートの補助として，小問題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．

GeForce GTX 280 のプログラムは，7 種類のカーネルを使い分け，1 Streaming Multiprocessor あたりのスレッド数は，128，256，512 の 3 通りを使い分けた．GeForce GTX 280 では，選択ソートを使わずに，バイトニックソートのみの計算となる．

Cell プロセッサのプログラムは，最初に，問題サイズ 8 の選択ソートを，SIMD 命令を用いて並列に解く．その後，データの並びを整理したのちに，バイトニックソートを開始す

る．SPE 上のバッファは，64 KiB のものを 2 枚用意している．

3.4 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 bit 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 bit 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，ホスト環境のプログラムも用意する．

GeForce GTX 280 のプログラムは，1 Streaming Multiprocessor あたりのスレッド数を 192 に設定して解く．Shared Memory に，処理中のパスワードの文字列と，各文字の計算済みのキーを配置し，Constant Memory に ZIP ファイルから抽出したヘッダと CRC を，Texture Memory に 8 bit の CRC を計算したテーブルを，転送して用いる．

Cell プロセッサのプログラムと，ホスト環境のプログラムは，8 bit の CRC を計算したテーブルを用意し，CRC を高速に計算できるようにしてある．

3.5 RandomAccess

このアプリケーションは，スーパーコンピュータの性能を測るためのベンチマークである，HPC Challenge の一項目である．この処理を簡単に示すと，図 1 のようになる．ただし，N は 2 の冪乗である．

我々は，$N = 2^{20}$ としてこの計算を行い，かかった時間を測定した．ホスト環境でも同様にプログラムを組み，測定を行った．

GeForce GTX 280 のプログラムでは，1 Streaming Processor あたりのスレッド数を 256 に設定した．このプログラムは，アトミック演算命令を用いて，メモリの更新処理を行う．アトミック演算命令は 32 bit のものしかないが，XOR の計算であるので，アトミック XOR を 2 回使えば用は足りる．

Cell プロセッサのプログラムでは，各 SPE がアクセスするメモリの範囲をあらかじめ定めておき，計算で求まったメモリアドレスが範囲外の領域だった場合は，SPE 間の DMA メモリアクセスを用いて通信を行い，処理を担当の SPE に投げる，という方法をとって，メモリの一貫性を確保している．

ホスト環境のプログラムは，1 スレッドのみの単純なプログラムである．マルチスレッドのプログラムも試してみたが，今回の環境では 1 スレッドのみの場合がもっとも高速であった．

3.6 Levenshtein 距離

本年度の SACSIS[3] における，GPU Challenge[2] および Cell Challenge[1] という 2 つのプログラミングコンテストにおいて，「Levenshtein 距離[12] を求めよ」という同じ問題を，CUDA および Cell プロセッサで解く，という課題が出された．これは，本研究において貴重なデータとなるものである．

Levenshtein 距離とは，2 つの文字列の間に定義される，距離空間上の整数値の一種であり，以下のように定義される．

- 2 つの文字列を，1 文字の追加・1 文字の削除・1 文字の交換のいずれかの操作を繰り返して，同じ文字列にする一連の操作を考えたとき，この操作の最短回数が Levenshtein 距離である．

Levenshtein 距離を求める方法として，広く用いられる方法は，2 つの文字列の 1 文字 1 文字を，行列の各列各行に対応させ，動的計画法で問題を解く方法である．我々は，この方法を用いて，計算を行った．

本来であれば，両コンテストのトップのチームのソースコードを参考にしたいところなのであるが，GPU Challenge 成績上位者のプログラムのソースコードが手に入らないため，我々が両コンテストに参加した時のプログラムを使い，比較を行う．

比較に関しては，ツールキット付属の，9 番の問題セットを使い，実行にかかった時間を測定した．

GeForce GTX 280 のプログラムは，動的計画法の行列を 128×128 の小ブロックに分割し，小ブロックが 1 つの Streaming Multiprocessor に対応するように処理を行った．

Cell プロセッサのプログラムは，動的計画法の行列を 64×64 の小ブロックに分割し，小ブロックが 1 つの SPE に対応するように処理を行った．小ブロックをこのサイズにすることによって，小ブロック内の計算を 8 bit で行うことができ，16 並列の SIMD 命令を使うことができる．

4. 結　　果

4.1 行 列 積

実行結果は，表 2 に示されたとおりである．ピーク性能に対して，GeForce GTX 280 の

```
void
init(unsigned long long t[])
{
  int i;
  for (i = 0; i < N; i++) {
    t[i] = i;
  }
}
void
update(unsigned long long t[])
{
  int i;
  unsigned lont long ran;
  for (i = 0; i < N * 4; i++) {
    ran = (ran << 1) ^
      (((signed long long) ran < 0) ?
        7ULL : 0);
    t[ran & (N - 1)] ^= ran;
  }
}
int
main()
{
  unsigned long long t[N];
  init(t);
  update(t);
}
```

図 1 RandomAccess の処理

	GeForce GTX 280	Cell
実行時間（ミリ秒）	59.5 (46.8)	140
処理速度 (GFLOPS)	289 (367)	123
ワット性能 (GFLOPS/W)	1.22 (1.56)	1.92

表 2 行列積の測定結果（() 内はホスト GPU 間転送抜き）

	GeForce GTX 280	Cell
実行時間（秒）	1.97 (0.483)	1.68
処理速度 (GFLOPS)	25.3 (103)	29.6
メモリ転送速度 (GiB/s)	1.98 (40.4)	6.98
ワット性能 (MFLOPS/W)	107 (436)	462

表 3 FFT の測定結果（() 内はホスト GPU 間転送抜き）

成績が振るわないように見える．この原因は 2 つあり，1 つは，GeForce GTX 280 の演算器の構成であり，Streaming Processor に 32 bit 積和算演算器を，Streaming Multiprocessor の超越関数演算器に 128 bit 積算演算器を搭載し，これらが同時に動く．一方，行列積計算に必要な計算は積和算のみであるので，超越関数演算器は休むことになる．このギャップが，ピーク性能との差の原因の 1 つになっている．もう 1 つの原因は，インターリーブのメモリアクセスを用いた場合，メモリレイテンシの隠蔽はできても，メモリスループットはそのまま実行速度に跳ね返ってくる．GeForce GTX 280 のメモリ転送速度は高速であるが，それ以上に処理能力も高いため，ブロッキングを使っていても，メモリスループットが無視できなくなっている．

一方，Cell プロセッサは，演算パイプラインは 1 本で，これが積和算を行うため，休んでいる演算器は無い．また，メモリ転送の裏で計算処理を行えるため，メモリスループットも無視することができる．このため，ピーク性能に近い性能が実現できている．

4.2 FFT

実行結果は，表 3 に示されたとおりである．ホスト GPU 間転送込みになると GeForce GTX 280 の性能が落ちるが，4×10^{20} バイトのデータに対し，計算量は 47.5×10^{20} FLOP しかないためである．これを無視しても，GeForce GTX 280 と Cell プロセッサの性能の

	GeForce GTX 280	Cell
実行時間（ミリ秒）	7.20 (5.42)	24.1
処理速度（G 比較/s）	15.3 (20.3)	4.57
メモリ転送速度 (GiB/s)	1.09 (33.2)	2.92
ワット性能 (M 比較/s/W)	64.8 (86.0)	71.4

表 4 ソーティングの測定結果（() 内はホスト GPU 間転送抜き）

差は，ピーク性能およびピークメモリ帯域の差よりも小さくなっているが，この理由としては，Cell プロセッサが，豊富な Local Store を生かして効率的なメモリの転送を行っていることと，本数の多いレジスタを生かしてシャッフル処理を省略できていることが挙げられる．FFT の大きさを変えると，また違った傾向が出る可能性があるが，これを調べることはは今後の課題である．

4.3 ソーティング

実行結果は，表 4 に示されたとおりである．GeForce GTX 280 が他のアーキテクチャを圧倒している．ただし，表には示していないが，Cell の初期化のコストが 14 ミリ秒強であり，これを無視すれば GeForce GTX 280 に迫ることが予想される．このコストが無視できる程度に問題サイズを大きくした場合については，今後の研究課題である．

このコストを無視して考えた場合，興味深いのは，ボトルネックになっている点が，GeForce GTX 280 ではメモリアクセスであるのに対して，Cell プロセッサでは CPU の処理能力である，[⋆1]という点である．これは，Cell プロセッサは大きな Local Store を持ち，大量のデータを転送してきて，そこで大量の処理を行える，という利点があるためである．

4.4 ZIP ファイルのパスワードクラッキング

実行結果は，表 5 に示されたとおりである．GeForce GTX 280 が 3 倍強速い，と，ここまでのアプリケーションと似た傾向が出ている．GeForce GTX 280 がもう一押し足りない理由の一つは，いくら Texture Memory が高速とはいえ，アクセスが頻繁なため，Texture Memory へのランダムアクセスが原因となって足を引っ張っている，という点である．

4.5 RandomAccess

実行結果は，表 6 に示されたとおりである．Cell プロセッサの場合，アクセスするメモリアドレスが確定してから DMA 転送命令を発行し，データの到着を待って計算を行う，という処理の流れにならざるを得ない．そのため，一点では，メモリ転送をソフトウェアで制御する Cell プロセッサの弱点が露呈した形であり，またある一点では，マルチスレッディングによってメモリレイテンシを隠蔽できる，GeForce GTX 280 の戦略が有効に機能していると言える．

⋆1 実行時間 10 ミリ秒で計算した処理速度は 11.0 G 比較/s，比較 1 回につき 3 命令以上実行

	GeForce GTX 280	Cell
実行時間（秒）	0.237	0.770
処理速度 (Mword/s)	362	111
ワット性能 (Mword/s/W)	1.53	1.73

表 5 ZIP ファイルのパスワードクラッキングの測定結果

	GeForce GTX 280	Cell
実行時間（ミリ秒）	27.0	207
メモリ転送速度 (GiB/s)	2.31	0.302
ワット性能 (MiB/s/W)	10.0	4.83

表 6 RandomAccess の測定結果

	GeForce GTX 280	Cell
実行時間（ミリ秒）	477	218
ワット性能 (10^3/s/W)	8.88	71.7

表 7 Levenshtein 距離の測定結果

4.6 Levenshtein 距離

実行結果は，表 7 に示されたとおりである．このアプリケーションにおいては，Cell プロセッサが GeForce GTX 280 を大きく上回っている．この原因として，この計算に必要な精度は 8 bit であり，8 bit の計算を 32 bit の計算と同じ回数しかこなせない GeForce GTX 280 が，8 bit の計算を 32 bit の計算の 4 倍の回数こなせる Cell プロセッサに後れをとる形になっている．

とは言っても，GeForce GTX 280 がダブルスコアで負ける，というのは，この点を差し

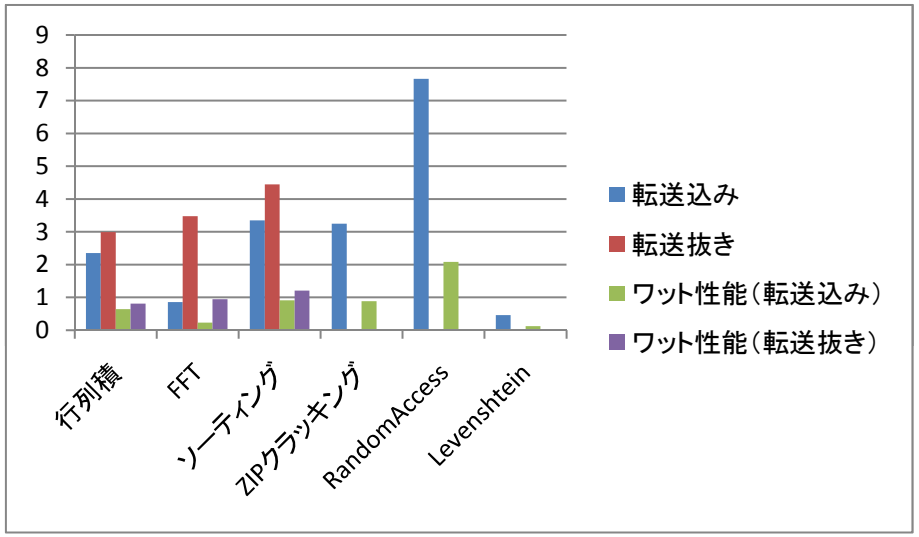

図 2 対 Cell プロセッサ比実行結果一覧

引いても，3 倍程度の高速化を為している他のアプリケーションに比べて，性能が十分に出ていないと言える．この原因については調査中であるが，GPU Challenge の上位陣のプログラムが参考になる可能性はある．

なお，GPU Challenge および Cell Challenge の結果においても，Cell プロセッサのほうが高速だという結論が導き出せる．

5. 関 連 研 究

GPGPU や，Cell プロセッサのような，ヘテロジニアス環境における並列プログラミングを行うフレームワークとして，OpenCL[16] がある．NVIDIA や Intel，AMD といった GPU メーカーや，Cell プロセッサを開発した IBM などがワーキンググループに参加しており，今後の普及が期待される．

AMD は，自社の GPU を使って GPGPU を行うために，スタンフォード大学で開発された Brook 言語[8] を拡張した，Brook+言語[10] を使った GPGPU 開発環境を提供している．

Scherl ら[18] は，コンピュータ断層撮影法の処理を GeForce 8800 GTX で実装し，Cell プロセッサと比較している．その結果，Cell プロセッサに対する実装よりも少ない労力で，Cell プロセッサ以上の処理速度を実現したと結論している．しかし，焦点を当てたポイントが実装そのものであり，比較を目的としたものではない，という点で，我々の研究とは異なる．

Agarwal ら[5] は，モンテカルロ法を用いた金融計算を，Cell プロセッサと GeForce 8800 GTX 上で，Cell SDK，CUDA，RapidMind SDK[14] を用いて実装し，それぞれを比較している．幅広いアプリケーションに対して比較をしている本研究と異なり，ある一つのアプリケーションに焦点を絞り，非常に詳細な比較を行っている．

6. 結　　論

測定結果を見てみると，標準的なアプリケーションでは，性能面で GeForce GTX 280 が 3 倍程度の差をつけて Cell プロセッサに勝っていることがわかる．ピーク性能比である 6 倍強ほどは差がついていないが，これはさまざまな要因によって，Cell プロセッサがピーク性能に近い性能を出しやすい設計になっているためである．

この要因の一つには，Shared Memory と Local Store の容量の差が挙げられる．メモリ帯域がボトルネックになるようなアプリケーションにおいて，Cell プロセッサは，大きな Local Store を使ったメモリ転送量の削減によって，GeForce GTX 280 との差を縮めている．

また，もう一つの要因としては，Cell プロセッサが，ソフトウェア制御のメモリコントローラによって，柔軟なメモリプリフェッチを実現している点が挙げられる．メモリアクセスの多いアプリケーションにおいて，計算の裏でメモリをフェッチすることで，Cell プロセッサは GeForce GTX 280 に比べて計算の効率を上げている．ただし，この長所は，裏返しで「ランダムアクセスに弱い」という欠点を孕んでいる．この点を加味すれば，今回のアプリケーションで大きな弱点の見当たらなかった，マルチスレッディングによるメモリレイテンシの隠蔽，という GeForce GTX 280 の戦略は，また優秀な戦略であると言える．

一方，ワット性能においては，GeForce GTX 280 が Cell プロセッサに劣るケースが多かった．この原因の一つは，Cell プロセッサが，比較的少ないトランジスタ数で実装できる SIMD アーキテクチャを採用しているのに対して，GeForce GTX 280 の SP は独立した命令フロー処理機構を持ち，これを実現するためにトランジスタ数が多くなってしまっている点である．無論，GeForce GTX 280 のアーキテクチャのほうが，柔軟な処理ができるのであるが，反面，消費電力の増加を招いてしまっている．ただし，GeForce GTX 280 の 236 W という消費電力は，GPGPU では使わない，グラフィックス処理に使うトランジスタの

分も含まれているため，GPGPU 処理時における実際の消費電力はこれよりやや少なくなる．GeForce GTX 280 と同一のアーキテクチャを持つが，グラフィックス処理を行わない，Tesla[⋆1]を使って測定，計算した場合は，今後の研究課題である．

これらの点を参考に GeForce GTX 280 のアーキテクチャを改良すれば，より GPGPU に適したアーキテクチャになると，今回の結果から結論付けられる．今後の GPU における，アーキテクチャの改良に期待したい．

参 考 文 献

1) *Cell Challenge 2009*, 2009. http://www.hpcc.jp/sacsis/2009/cell/.
2) *GPU Challenge 2009*, 2009. http://www.hpcc.jp/sacsis/2009/gpu/.
3) SACSIS2009 - 先進的計算基盤システムシンポジウム, 2009. http://www.hpcc.jp/sacsis/2009/.
4) 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.
5) V.Agarwal, Lurng-Kuo Liu, and D.A. Bader. Financial modeling on the cell broadband engine. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pp. 1–12, April 2008.
6) K.E. Batcher. Sorting networks and their applications. *Proceeding AFIPS Spring Joint Computer Conference*, 1968.
7) 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.
8) Ian Buck, Tim Foley, Daniel Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, and Pat Hanrahan. Brook for gpus: stream computing on graphics hardware. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pp. 777–786, New York, NY, USA, 2004. ACM.
9) JamesW. Cooley and JohnW. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comput. 19*, pp. 297–301, 1965.
10) Advanced MicroDevices Inc. Brook+ sc07 bof session. *Supercomputing 2007 Conference*, November 2007.
11) 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.
12) VladimirI. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report8, 1966.
13) David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, and Aaron Lefohn. Gpgpu: general purpose computation on graphics hardware. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, p.33, New York, NY, USA, 2004. ACM.
14) MichaelD. McCool. Data-parallel programming on the cell be and the gpu using the rapidmind development platform. *the GSPx Multicore Applications Conference*, 2006.
15) G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, Vol.86, No.1, pp. 82–85, 1998.
16) Aaftab Munshi. Opencl. http://s08.idav.ucdavis.edu/munshi-opencl.pdf.
17) 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.
18) 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.
19) D.Takahashi. High-performance parallel fft algorithms for the hitachi sr8000. *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, Vol.1, pp. 192–199 vol.1, 2000.
20) Vasily Volkov. Homepage for vasily volkov. http://www.cs.berkeley.edu/volkov/.
21) D.Wang. Isscc 2005: The cell microprocessor. *Real World Technologies*, February 2005. http://www.realworldtech.com/page.cfm?ArticleID=rwt021005084318&p=2.

⋆1 GeForce GTX 280 とほぼ同等の Tesla C1060 の消費電力は 187.8 W である