社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－HPC－115 (4)
2008／5／20

CUDAによる定常反復 Poisson ベンチマークの高速化

小川 慧† 青木 尊之††

SIMD 型のアクセラレータである GPU を用い、格子系の流体計算などで最も計算負荷の高い Poisson 方程式の高速化を試みた。定常反復法である Point Jacobi 法で Poisson 方程式を計算する理研（姫野）ベンチマークテストに対し、CUDA でプログラミングすることにより 1 枚の GPU で 30.6 GFLOPS の演算性能が得られた。また、複数 GPU を用いた計算では、CPU 側のメモリ上に作成したバッファを介して GPU 間の通信を行い、4 GPU を用いて最大 93.6 GFLOPS の演算性能が達成された。

Acceleration of Point-Jacobi Poisson Solver by using GPU

Satoi Ogawa† and Takayuki Aoki††

The Poisson equation which is the major time consuming of CFD (Computational Fluid Dynamics) is solved on the GPU as a SIMD-type accelerator. The Riken (Himeno) benchmark problem uses the Point-Jacobi method to solve the Poisson equation and the performance of 30.6 GFLOPS is achieved when we use a nVIDIA GeForce 8800 Ultra card. The GPU code is described by CUDA and 4-GPU parallel computing scores 93.6 GFLOPS.

1. はじめに

GPU(Graphics Processing Unit) はグラフィックス処理だけでなく、SIMD 型アクセラレータとして汎用計算に用いる試みが GPGPU(General-Purpose computation on GPU) または GPU コンピューティングとして注目されている。GPU は CPU に比べて広いバンド幅を持つビデオメモリと高速な浮動小数点演算器を搭載した SP(Streaming Processor) を持つハードウェアである。これまでの GPU コンピューティングは頂点ユニットおよびシェーダーユニットがプログラム可能であることを利用、汎用計算をグラフィックス機能に置き換えて実行させる必要があり、cg 言語と呼ばれる言語で記述する必要があった。一昨年、CUDA[1] と呼ばれる GPGPU 用のフレームワークが nVIDIA 社よりリリースされ、グラフィックス機能を全く意識することなしに C 言語の拡張としてプログラミングが可能になり、GPU を汎用計算の演算デバイスとして利用しやすい環境が整ってきた。GPU コンピューティングは既に幾つかの成果が奥山らの研究[2]、坂牧らの分子動力学加速[3] など報告されている。特に天体物理の N 体粒子計算では 256GFLOPS という実行性能[4] が報告されており、GPU の演算性能を十分に引き出すことに成功している。一方、格子計算は粒子計算と比較して計算精度が高いために理工学のさまざまな分野で重要な解析ツールとして広く使われているが、GPU を利用した高速計算は殆ど報告されていない。非圧縮性流体は MAC(Marker-and-Cell) 法などのセミ陰解法と呼ばれる手法で計算されることが多く、圧力について Poisson 方程式を解く必要がある。電磁気学でも静電ポテンシャルは Poisson 方程式に支配される。楕円型方程式である Poisson 方程式は空間格子上で離散化されると大規模線形一次連立方程式になり、係数行列は規則的な疎行列になる。非圧縮性流体計算では、計算時間の大半が Poisson 方程式の計算に費やされるため、大規模疎行列計算を高速化する研究が精力的に行われてきた。本論文では、一般座標系における Poisson 方程式を 2 次精度中心差分法で離散化した式を定常反復法である Point Jacobi 法で計算する理研ベンチマーク問題を GPU で高速化することを試みる。Point Jacobi 法は安定性と収束性が低いために現在では殆ど用いられない行列解法であるが、ベンチマーク問題として非常に多種類の CPU やプロセッサに対するデータが存在し、GPU で高速化の基準が見つけ易い。また他の利点として、反復計算が 1 ステップ前の値に基づいて行われるため、隣接計算のデータ依存性がなく、GPU の並列計算に向いているという点がある。GPU と同様な SIMD 型のアクセラレータである ClearSpeed[6] を使って理研ベンチマークテス

† 東京工業大学大学院理工学研究科
Tokyo Institute of Technology
†† 東京工業大学学術国際情報センター
Global Scientific Information and Computing Center, Tokyo Institute of Technology

トを加速する研究[7]では、カード上のメモリ転送速度が遅いために十分に演算性能を引き出すことができないことが報告されている。本研究は平成20年3月の理研シンポジウムの中で行われた理研ベンチマークコンテストに応募した内容に基づいている。

2. 理研ベンチマークテストの内容

一般座標系において、従属変数 p に関する3次元Poisson方程式

$$\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}+\frac{\partial^2 p}{\partial z^2} + \alpha\frac{\partial^2 p}{\partial xy}+\beta\frac{\partial^2 p}{\partial xz}+\gamma\frac{\partial^2 p}{\partial yz}=\rho \quad (1)$$

を2次中心差分法で離散化すると

$$\begin{aligned}&\frac{p_{i+1,j,k}-2p_{i,j,k}+p_{i-1,j,k}}{\Delta x^2}\\ &+\frac{p_{i,j+1,k}-2p_{i,j,k}+p_{i,j-1,k}}{\Delta y^2}\\ &+\frac{p_{i,j,k+1}-2p_{i,j,k}+p_{i,j,k-1}}{\Delta z^2}\\ &+\alpha\frac{p_{i+1,j+1,k}-p_{i-1,j+1,k}-p_{i+1,j-1,k}+p_{i-1,j-1,k}}{4\Delta x\Delta y}\\ &+\beta\frac{p_{i+1,j,k+1}-p_{i-1,j,k+1}-p_{i+1,j-1,k}+p_{i-1,j,k-1}}{4\Delta x\Delta z}\\ &+\gamma\frac{p_{i,j+1,k+1}-p_{i,j+1,k-1}-p_{i,j-1,k-1}+p_{i,j-1,k-1}}{4\Delta y\Delta z}\\ &=\rho_{i,j,k} \quad (2)\end{aligned}$$

になる。ここで α,β,γ は座標変換のメトリック、ρ は生成項で静電場ならば電荷密度、非圧縮性流体ならば速度の divergence である。理研ベンチマークテストは、式(2)をPoint Jacobi法で解く問題であり、C言語およびFortran言語のソースコードが準備されている。そこでは、反復計算中に変化しない係数行列を全て配列として保持し、反復計算の度に参照するコードとなっている。計算規模がSサイズ($65\times65\times129$)、Mサイズ($129\times129\times257$)、Lサイズ($257\times257\times513$)、XLサイズ($513\times513\times1025$)の4種類が準備されている。コード可変で独自の高速化チューニングが許されるのはSサイズの問題である。$65\times65\times129$ の1配列は2.18MBにしかならないが、**表1**に示すとおり、理研ベンチマークテストは係数を全てこのサイズの配列12個に格納し、2つの配列が1つ前のステップの値と現ステップの値の格納に割り当てられ、読み書きの対象となる。この12個の配列は格子点間の相互参照がないためにCPUキャッシュが効かず、メモリバンド幅が広いベクトル型スーパーコンピュータが非常に有利と言われている。従属変数 p に関しては**図1**のように多数の隣接参照があり、CPUの場合は計算

表1 理研ベンチマークで使用される代表的な配列
Table 1 Arrays used in Riken Benchmark

Name	Property
a0, a1, a2	Read only
b0, b1, b2	Read only
c0, c1, c2	Read only
bnd	Read only
wrk1	Read only
p	Read & Write
wrk2	Read & Write

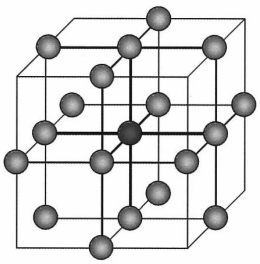

図1 理研ベンチマークテストの変数 p の隣接参照
Fig. 1 Stencil of Pressure Calculation

サイズに対応したキャッシュが効くので、キャッシュ・メモリのサイズが計算速度に大きく影響する。理研ベンチマークテストは初期条件を設定したあと、Point Jacobi反復計算を3回行い、その後に500回の反復計算を行う部分の計算速度を計測するようになっている。本論文でもPoint Jacobi計算のみをGPUで計算させることによりCPUでの計算と比較して高速化を行う。本研究に用いた計算機環境を**表2**に示す。本論文で用いたGPUの詳細を**表3**に示す。SP数は

表2 実験に用いた PC の仕様
Table 2 Specification of used PC

CPU	AMD Phenom 9600(2.3GHz)
Memory	DDR2-800 SDRAM 4 GByte
Graphics	MSI NX8800ULTRA-T2D768E-HD-OC x 2
	ELSA GD988-768ERU x 2
OS	CentOS 5.1, CUDA SDK 1.1

表3 使用した GPU の仕様
Table 3 Specification of used GPUs

Card Name	GPU1 (GPU2)
GPU Chip	GeForce8800Ultra[G92]
Peak Performance [GFLOPS]	384(414)
Number of SP	128
SP Clock [MHz]	1500(1618)
Memory Trasfer Rate [GB/s]	103.68(110.4)
Memory Inferface [bit]	384
Data Rate [GHz]	2.16(2.3)
Video memory [MB]	768

GPU1=ELSA GD988-768ERU
GPU2=MSI NX8800ULTRA-T2D768E-HD-OC

128、メモリインターフェースは 384 bit、理論性能は最大 1 枚 414 GFlops である。

3. 1 つの GPU による高速化

3.1 Shared memory を使用した隣接点計算

理研ベンチマークテストに対する 14 個の S サイズの配列を GPU のビデオメモリ上に確保し、予め cudaMemcpy 関数を用いてデータ転送しておく。ビデオメモリは CUDA では Global memory と呼ばれ、全てのスレッドから共有でアクセスできる。今回用いた GeForce 8800 Ultra の場合、Global memory から Shared memory にデータ転送する速度は、転送量に応じて**図 2** のようになる。Global memory に対する連続アクセスはバースト転送を行う Coalaced Access となる場合の測定結果である。

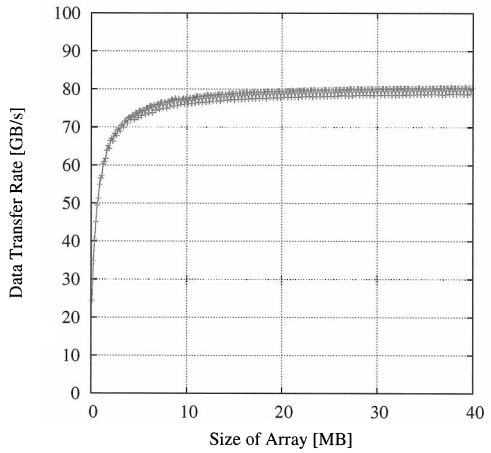

図 2 Global memory と Shared memory 間の実効メモリ帯域幅

Fig. 2 Memory Transfer Rate between Global memory and Shared memory

一方、ランダムメモリアクセスは転送速度が 1/10 に低下してしまう。先に示した通り、理研ベンチマークテストはデータ転送処理が多いため、Coalaced Access ができたとしても実行時間の中のデータ転送時間が占める割合が大きい。

Global memory 上の 1 変数配列 2.18MB の読み出し速度は図 2 から 70[GB/sec] 程度であり、単精度データに対しては 70 ÷ 4=17.5 GWord/sec となる。一方、理研ベンチマークテストは 1 格子点当たり 14 変数を読み書きし、34 回の浮動小数点演算を行う。従って、演算時間が無視できるほど短い（性能が高い）と仮定しても最大演算性能は 17.5 × 34/14 = 42.5 GFLOPS である。しかし、圧力変数 p の 18 回の隣接参照を毎回行うと、計算速度は 17.5 × 34/(14+18) = 18.6 GFLOPS に低下してしまう。圧力変数の隣接参照回数を減らすために block 内で共有できる Shared memory をキャッシュとして利用する。8 個の SP で構成する multiprocessor 内にある Shared memory のサイズが 16kB と少ないため、計算領域を $16 \times 16 \times 8$ の格子点の領域単位に**図 3** のように分割する。1 つの

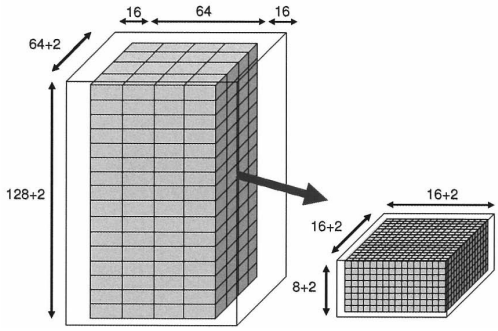

図 3 Global memory と Shared memory 上の配列の構造

Fig. 3 Assign of Block Structure and usage of Shared memory

block がこの計算領域を担当し、全体では $4 \times 4 \times 16$ の block が存在することになる。block 内のスレッドは $(16+2) \times (16+2) \times (8+2)$ の圧力変数を共有する。Shared memory のサイズが担当計算格子より大きいのは、p が隣接参照する際のはみ出しに対するパディングである。この部分のメモリーロードは大半が Coalaced Access ではないため、全体としてのデータ転送速度を低下させるが、転送量が少ないために低下率は低い。

3.2 誤差計算

理研ベンチマークテストは各格子点上で残差を計算し、その 2 乗の総和を残差として求めている。上記 16 × 16 × 8 の格子点の残差を Shared memory 上にセーブする。Shared memory 上の総和計算は 256 個のスレッドで reduction を行い 1 つの値にした後、Global memory 上に用意した block 構成に対応した配列 ss[4 × 4 × 16] の 1 要素に転送する。500 回の計算の最後に GPU 上の誤差配列 ss を CPU 側に転送し、CPU 上で ss の 256 要素の総和をシリアルに求める。

3.3 浮動小数点演算の精度

GPU の浮動小数点計算は IEEE に準拠しているが丸めのモードが異なるなど、CPU の浮動小数点計算と誤差が異なる。理研ベンチマークテストの浮動小数点計算の精度検証については、CUDA のエミュレーションは CPU で行うことから、GPU エミュレーションで計算した誤差が CPU で行った結果と一致していることを確認した。

3.4 実行性能

nVIDIA GeForce 8800 Ultra は、G80 の GPU コアであるが、メモリインターフェイスが 384bit あるため G92 世代の GPU カードよりメモリ転送速度が速く、ピーク性能が 100GB/sec を超える。理研ベンチマークテストはメモリ転送速度が速いほど有利であり、GeForce8800 Ultra を 1 枚用いた際の実行性能は 30.6GFLOPS となった。GPU では SP が整数演算も浮動小数点演算も行うが、CPU での計算と比較するために浮動小数点演算部分のみカウントし FLOPS 値を算出している。CPU (AMD Phenom 2.3GHz) で実行速度は 0.976GFLOPS であり、GPU を用いることにより 31.4 倍の高速化が達成されたことになる。

4. 複数 GPU による高速化

4.1 領域分割方法の検討

複数 GPU を用いてさらに理研ベンチマークテストの高速化を試みた。今回用いたマザーボードは MSI K9A2 Platinum で、PCI Express x16 (Generation 2) が 4 スロットある。ただし、**図 4** のように 4 枚の GPU を装着した場合は、それぞれが PCI Express x8 (Generation 2.0) となることが分かっている。GPU

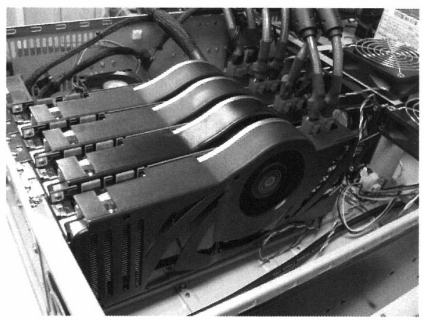

図 4 4 つの PCI-Express x16 幅スロットに装着された GPU 群
Fig. 4 4 GPUs on 4 PCI-Express slots

の演算負荷低減およびメモリバンド幅の増大を狙って、領域分割による並列化を行った。MPI ライブラリを利用した領域分割による並列計算と同様に、分割領域の境界データを隣接領域から取得する必要がある。GPU カードを超えて Global memory の内容を相互に直接参照することができないので、**図 5** のように CPU 側のメインメモリ上にバッファ領域を確保し、そのメモリ領域を介して境界データを反復計算の度に交換している。

計算領域を図 5 のように長軸である z 方向に短冊状に 1 次元分割する理由は、圧力 p の計算の隣接参照が対角線にまで伸びているため、2 次元分割を行うと通信回数が倍に増えるためである。複数の GPU を

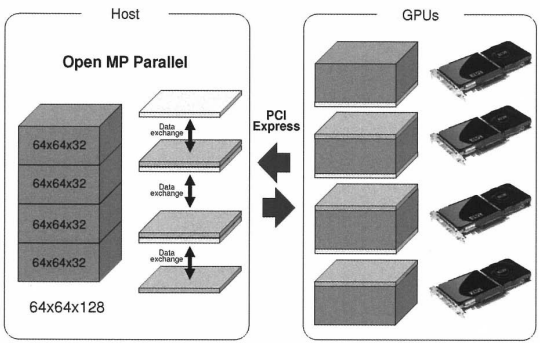

図 5 4 枚の GPU を用い、分割された領域を並列計算する概念図
Fig. 5 Concept of Parallel Computation using 4 GPUs

制御するために、CPU 側で使用する GPU の数だけ Open MP のスレッドを走らせ、GPU のカーネル関数を実行させる。

4.2 実行性能

表 4 に結果を示す。S サイズの計算を 4 GPU で行った結果は 51.9 GFLOPS で、4 枚の GPU を用いて約 1.7 倍しか速くなっていないことが分かる。これは PCI-Express Bus を介して GPU の Global memory と CPU 側のメインメモリ間のデータ転送が遅いことが最大の理由である。**図 6** は、GPU の Global memory と CPU のメモリへの転送速度の実測値である。S サイズの計算では交換する境界領域のデータのサイズが 32kB と小さいためオーバヘッドが大きく転送速度が低いことが分かる。M サイズの問題、L サイズの問題を同様に複数 GPU で計算した結果が表 4 である。境界領域のデータ交換に所用する時間が計算時間に対して相対的に小さくなったためであり、L サイズの問題では 4 枚の GPU を用いて約 3 倍の性能向上が得られた。

表 4 GPU による Poisson ベンチマーク高速化結果
Table 4 Parallel performance of Riken Benchmark using GPUs

Size	# of GPUs	Performance[GFlops]	Time[sec.]
S	1	30.6	0.268
	2	42.5	0.193
	4	51.9	0.158
M	1	29.4	2.328
	2	53.7	1.275
	4	83.6	0.8190
L	1	-	-
	2	-	-
	4	93.6	5.974

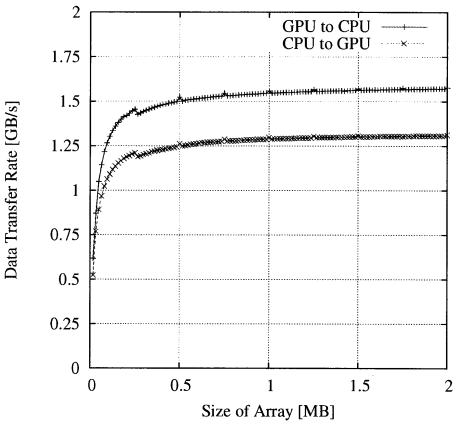

図 6 GPU と CPU 間のデータ転送実効速度
Fig. 6 Data Transfer Rate between GPU and CPU

5. おわりに

これまで、GPU で格子計算を加速する報告は余り報告されてこなかったが、理研ベンチマークテストという Poisson 方程式を定常反復法（Point Jacobi 法）で解く問題を GPU で加速することができた。GPU を 1 枚使って 30.6 GFLOPS の性能を引き出すことができ、メモリバンド幅律速であるにもかかわらず、CPU での計算の 30 倍以上の高速化が達成された。また、複数枚の GPU を使ったカードを超えた並列計算も行い、4 GPU を用いて 93.6 GFLOPS が得られ、これはベクトル型スパコン SX-8 の 15 CPU 分に相当する性能が 1 台の PC の筐体で達成されたことになる。

謝辞 本研究の一部は、日本学術振興会 グローバル COE プログラム「計算世界観の深化と発展」及び、科学技術振興機構 CREST「ULP-HPC：次世代テクノロジーのモデル化・最適化による低消費電力ハイパフォーマンスコンピューティング」の支援による。

参考文献

1) nVIDIA CUDA : http://www.nvidia.com/cuda/
2) 奥山倫弘、伊野文彦、萩原兼一：CUDA を用いた GPU による全点対最短経路問題の実装比較, pp.58 , HPCS2008 (2008)
3) 坂牧隆司、成見哲、泰岡顕治：ビデオゲーム用ハードウェアを用いた高速分子動力学シミュレーション, 101P, 第 21 回分子シミュレーション討論会 (2007)
4) Hamada T., Iitaka T. :*The Chamomile Scheme : An Optimized Algorithm for N-body simulationson Programmable Graphics Processing Units*, http://arxiv.org/abs/astro-ph/0703100v1 (2007)
5) 理研ベンチマーク：http://accc.riken.jp/HPC/HimenoBMT/index.html.
6) Clearspeed：http://www.clearspeed.com
7) 西川由里、鯉渕道紘、吉見真聡、天野英晴：ClearSpeed 製コプロセッサの並列ベンチマークによる性能評価と性能向上手法の提案：情報処理学会研究報告 2007-HPC-109(pp257-262, March 2)(2007)