GPUを用いた並列共用スーパーコンピュータの構築

Shared Parallel GPU Supercomputer Construction

白﨑　博公*

Hirokimi Shirasaki*

*玉川大学工学部機械情報システム学科，〒194-8610　東京都町田市玉川学園 6-1-1

*Department of Intelligent Mechanical Systems, College of Engineering, Tamagawa University, Tamagawa-Gakuen 6-1-1, Machida, Tokyo 194-8610

Abstract

In this study, the parallel GPU supercomputer is constructed.　Using the GPU, the high-performance computing (HPC) system can be comparatively constructed cheaply and the calculation performance per energy consumption (Green) becomes better or acceptable.　Here, we construct the programming language CUDA (Compute Unified Device Architecture) environment for the NVIDIA GPU.　The computer is composed of two Intel Xeon (6 cores) CPUs and four Nvidia Tesla C2070 (448 CUDA cores) GPUs.　The OS is 64 bit Linux (Cent OS) which makes it possible to access remotely from other PCs.　We install the MATLAB, Open ACC and so on for use with the GPU.　Next, we calculate the electromagnetic field by the FDTD method, and show that the calculation time becomes more than 20 times.

Keywords: Supercomputer, Parallel computing, HPC, GPU, CUDA, Linux server, Cent OS.

１．はじめに

東工大の「TSUBAME」は、GPU (Graphics Processing Unit) を取り入れたスパコンである[1]。GPUは、数百の演算コアが集約されており、並列的な演算能力が高く、かつ消費電力が少ない特徴を持つ。OSはLinuxで、アクセス権があれば、東工大では１年生から、遠隔ログインでアクセスして利用できる。当大学でも、院生や教員のためのスパコンの導入の必要性を感じていた。スパコンの重要な利用法は、身の回りの物理現象や工学設計の数値シミュレーションであり、現代社会では重要な役割を担っている。著者も、1998年より続けているスキャトロメトリ解析において、マルチコアCPUやメニーコアGPUを用いた並列処理で、数値計算時間の短縮を行ってきた[2-4]。

本研究では、院生や教員が自由に使える高速並列計算機を学内に導入し、並列GPUを搭載した環境構築を目的としている。そして、並列GPUの効率的利用による計算の高速化により、数値シミュレーションの研究レベルを向上させることを第１の目的としている。また、ホームページによる計算機の使用状況の公開や、成果の発表、定期的な利用法についての講習会開催により[2]、研究のアクティビティの宣伝に貢献することを第２の目的とする。

本論文では、構築したサーバを有効利用してもらうために、まず並列GPUを搭載したサーバの利点、運用法について述べる。そして、構築したサーバのスペック、環境構築について述べる。次に、並列計算システムの開発環境として、CUDAやマルチGPUなどについて述べる。最後に、FDTD

（時間領域差分）法を用いて、GPU を用いることにより、計算速度が大幅に早くなることを述べている。

2．GPU と CUDA プログラミング

2.1　GPU について

GPU（Graphics Processing Unit）は、PC やワークステーション等の画像処理を担当する主要な部品の一つである[5]。VPU（Visual Processing Unit）という名称もある。コンピュータシステムにおいて画像表示を担当する ASIC に等しいグラフィックコントローラから発展したものであり、ジオメトリエンジンなどの専用ハードウェアを用いた画像データ処理を行う集積回路をさしている。現在の高機能 GPU は高速の VRAM と接続され、グラフィックスシェーディングに特化した演算器を複数搭載するマイクロプロセッサとなっている。そもそも、従来より GPU は比較的単純な計算を、多数のデータに並列に繰り返し実行するのが得意な構造になっているため、科学技術関連の解析やシミュレーションと相性が良い。近年 GPU メーカーでは GPU コンピューティングを意識して、GPU の設計を当コンピューティングに対し、より適した構造に改良したり、その支援となるソフトウェアを提供したりして、GPU コンピューティング用途の開拓に積極的な背景がある。

NVIDIA の社の GPU は Fermi もしくは Kepler というアーキテクチャが採用されている。図 1 は NVIDIA 社の Tesla などに使用される Fermi アーキテクチャの模式図である。

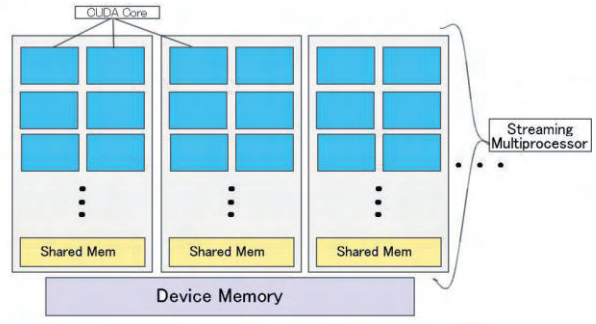

図 1　Fermi アーキテクチャ

GPU は Streaming-Multiprocessor(SM) と Device Memory によって構成されている。SM は Shared Memory と CUDA Core によって構成されており、CUDA Core については多数搭載できる様式である。この CUDA Core の搭載数が GPU の性能を決める指標の一つとなる。

2.2　GPGPU と GPU コンピューティング

GPGPU や GPU コンピューティングは、コンピュータグラフィックスの生成やビデオ圧縮・再生などで必要とされる演算・処理を高速に実行する機能を、科学技術研究等で用いられる大規模数値計算その他に応用する技術である。安価なグラフィックスカードでも高価な並列計算機に匹敵する高速処理を可能とする長所があることにより注目されている技術である。

GPGPU コンピューティングと GPGPU とを正式に区別することは難しいが、より汎用的な HPC（High Performance Computing）の実現に向けた取り組みを行っているのが GPU コンピューティングである。一般にグラフィックス・カード上のビデオ・メモリを利用するのが GPGPU、メイン・メモリを利用するのが GPU コンピューティングと区別される。

2.3　CUDA プログラミング

CUDA（Compute Unified Device Architecture）とは、NVIDIA GPU 製品において汎用的なプログラムを並列計算するための統合開発環境である[6]。この環境下では、第 1 に、GPU を多数のスレッドが並列実行可能であるコプロセッサとみなす。第 2 に、CPU やメインメモリを“ホスト”、GPU を“デバイス”として扱う。図 2 は、CUDA プログラムの流れを示す。まず、カーネル関数をホスト側で呼び出し、デバイス側のメモリを確保してから、ホストのデータをデバイス側に送る。次に、デバイスで計算を実行する。計算が終了したら、結果をホスト側に返し、CPU で結果を表示できるように処理する。最後に、デバイス側のメモリを解放し、計算終了となる。

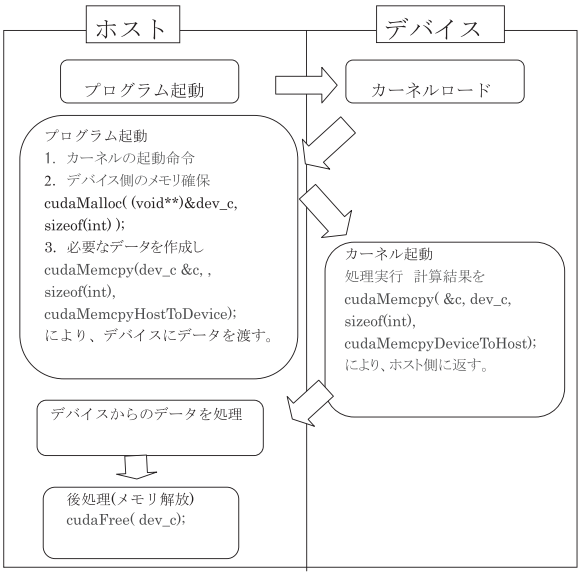

図２　CUDAプログラムの流れ

2.4　スレッド、ブロック、グリッドの概念

CUDAでは、新しい単位を当てはめる。昇順に挙げると、「スレッド」、「ブロック」、「グリッド」となる。

スレッドはスカラプロセッサ上で動作するプログラムの単位である。スレッド自体は、ホスト側から起動される。実行されるプログラムはすべて同じだが、タイミングは別となる。このプログラムがタイミングを揃えずに実行されることを、非同期の動作という。CUDAのプログラミングで特筆すべきは、多数の処理がスレッドとして並列に動作しながらも、処理がすべて非同期であることである。それを同期にするにはプログラム上での関数として syncthreads がある。この関数により、非同期で動作していたプログラムの同期をとることができる。

ブロックはスレッドの集合体である。１つのブロックには最大 512 スレッドがまとめられている。また、このスレッドの表現を最大 3 次元まで拡張できる。

グリッドは、ブロックをさらに集合させたものである。グリッド内のブロックは、２次元で表現される。１グリッド内の最大のブロック数は、65535 である。

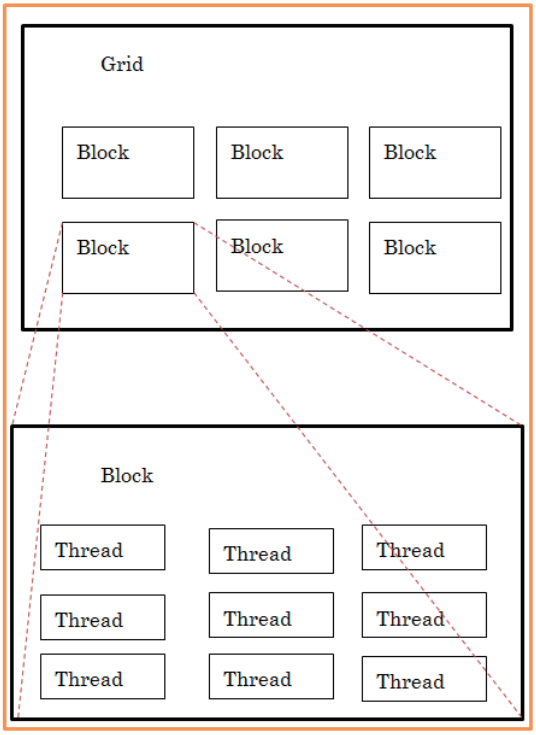

図３　スレッド、ブロック、グリッドの構成

また、グリッドの上位に、デバイスといわれる仕組みがあり、GPU 本体のことを意味している。しかし、現在グリッドとデバイスとの間に明確な定義式がないので、「１グリッド＝１デバイス」と考えるのが適切である。このグリッド、ブロック、スレッドの構成を図３で示す。

また、スレッドは32スレッドごとに動作することになっている。この32スレッドを１つと見なす場合、これを「ワープ」と呼称する。よって、スレッドの処理を32として「ワープ」を基準とすることで、計算の区切りが良くなる。

３．並列 GPU 計算サーバ構築

ここでは，並列 GPU 計算サーバの構築手順や、選定したマシンのスペックおよびOSについて述べる。また、インストールした商用ソフトウェアや利用法について述べる。

3.1　CentOS の採用

本サーバ構築の際、Linux OSとして、Ubuntu、Fedora、CentOSを試した。安定性と信頼

表１　構築サーバ「 Tachyon 」のスペック

OS	CentOS 6.2
プロセッサ（ CPU ）	Xeon 6C X5690 × 2
プロッセッサ周波数	3.46GHz
主記憶容量（ RAM ）	DDR3-1333 8GB ECC REG ×12　合計96GB
グラフィックボード	Tesla C2070 × 4
CUDAバージョン	5.0
CUDAコア数	448
CUDAコア周波数	1.15GHz
GPU 専用メモリ合計	6GB
補助記憶装置（内臓）	3.5SATA 1000GB HDD ×2 合計 2TB
ドメイン名	tachyon.tamagawa.ac.jp

性、使用感を判断基準としてCentOSを用いた。CentOSとは Red Hat Enterprise Linux との完全互換を目指したフリーの Linux ディストリビューションである。そのためCentOSは、高い安定性と信頼性を誇っているのでサーバOSを主目的として開発されている。加えてサードパーティのリポジトリを用いてマルチメディアツール等をインストールすることもできるため、デスクトップOSとして用いることも可能である。

3.2 サーバ構築

アカウント登録した学生や教員が、ネットワークを通して学部内で使用可能な並列 GPU 計算機サーバを構築した。サーバとして、 GPU ワークステーション（ NEC 製108T d -4G ）を導入した。サーバ名は、 Tachyon （タキオン）とした。スペックを、表１にまとめた。

このマシンは、 GPU による高い演算能力を主目的としており、 GPU として nVIDIA Tesla C2070 を４枚搭載している。また、 CPU に６コアのインテル Xeon 6C X5690 を２個（12スレッド）、メモリを96GB搭載しているので、 CPU のみでも並列計算ができる。図４に、サーバの内部写真を示した。左下半分に Tesla が４枚、左上半分に、 CPU が２個、メモリが12枚刺さっている様子を写している。

3.3　環境構築

図４　GPU サーバ「 Tachyon 」

開発環境としてCUDAやOpenGLを中心としたGPGPU 環境を導入した。

マシンをサーバとして使用するためにIPアドレスの固定を行い DNS サーバ、ゲートウェイ、フィルタリング等の通信環境を設定した。

CUDA環境導入では、CUDAパッケージとグラフィックドライバのインストールを行った。そして、CUDA関数が用いられるようにライブラリ等の環境変数の設定をbashrc実行ファイルに対して行った。CUDA のコンパイル命令である nvcc を用いて、CUDA の deviceQuery 命令をコンパイル実行することで GPU の動作環境を調べた。

3.4　運用法

運用する側のWindows PCに、 Xming やcygwin等の X Sever ソフトを導入した。これにより、ファイル転送することなくサーバ上のグラフィックスを、手元のWindows PCに表示することができる。また、TeraTarm等のコンソールソフトによるｓお行うことで、サーバにログインすることができ、サーバ内のリモート操作が可能な状態に至る。

3.5　導入ソフトウェア

表２に GPU サーバを有効利用するために導入した商用ソフトウェアを示した。

１番目のMATLABに関しては、マルチコアプロセッサや、 GPU を備えたコンピュータの利点を生かせるようにアプリケーションを変換する高度なプログラミング機能がある Parallel Computing

Toolbox を導入した。

２番目に、既存のプログラムを変更することなく、指示文を挿入するだけで（ディレクティブ・ベース方式）GPU を利用した高速計算が可能になる OpenACC を導入した。また、指示文ベースの GPU 化コードジェネレータHMPPも導入した。これらは共に、フランスCaps社によるコンパイラである。

３番目に、12コアある CPU での並列計算プログラムを効率よく作成するために、インテル C++ Composer XE を導入した。このソフトでは、並列化に欠陥が生じた部分をチェックすることができる。

表２　導入した商用ソフト

MATLAB	Simulinnk Parallel Computing Toolbox Optimization Toolbox Signal Processing Toolbox
CAPS コンパイラ	HMPP OpenACC
インテル C++ Composer XE	インテルC++ コンパイラ インテル　MKL 11.0 インテル　IPP 7.1 インテル　TBB 4.1 インテル　デバッガー Eclipse 開発環境への統合

４．　２次元 FDTD 解析

ここでは、２次元並列FDTDプログラムを用いて、 CPU のみの場合と、 GPU を用いた場合の計算速度の比較を行う。また、計算領域を分割して、マルチ GPU プログラミングによる数値計算も行う。

4.1　FDTD 法とは

FDTD法（Finite Difference Time Domain method）とは、マックスウェルの方程式を時間、空間で差分化し、解析空間の電磁界をリープフロッグアルゴリズムを用いて時間的に更新、出力点の時間応答を得る方法である[7]。従って過渡解あるいは周波数応答を直接求めることができる。通常の差分法と有限要素法では 2階の偏微分を扱うが、FDTDでは１階偏微分により計算を行う。アンテナの解析法にはモーメント法や有限要素法などがあるが、FDTD法は比較的アルゴリズムが簡単で深い知識が無くとも、電磁波解析が容易である。また、優れた精度を持つこと、複雑な物質の解析や材料定数の異なる物質の解析にも適していることなどが知られている。特に誘電体の解析でも誘電率やタイムステップ数などの定数を変えるだけで済む。 FDTD　法は電磁解析をするにあたり電界と磁界の要素が必要であり、電界を求める際半時刻前の半セルずれた磁界を用いて電界を算出して行う。そのため、パラメータが多い為に基本的に計算機のメモリを多く使用し、それに伴い多くの計算時間が必要とされる点が支障となりうる。２次元の問題ならば多くの時間を必要とするわけでもないが、3 次元の問題などを扱う場合などはこの問題を克服しなければならないという短所を持っている。

4.2　CUDA による FDTD 計算速度の比較

FDTD解析を行うために、２次元領域を x 軸と y 軸方向にそれぞれ、 Nx=Ny=511 の格子に分割した。境界で電磁波を吸収するための pml（Perfectly Matching Layer）は、領域を覆うように８格子を確保した。そして、屈折率 1.5 の誘電

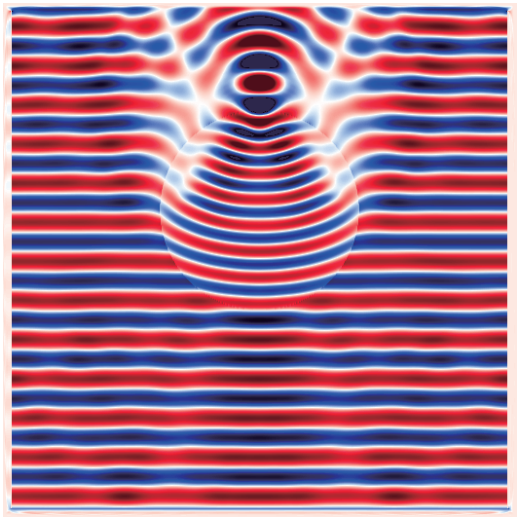

図５　FDTD計算結果（T=5000）

体は、 x= Nx/2, y=3Ny/5 を中心とし、半径Nx/5の内部に置いた。図５には、下の y=0 の場所に、周波数500[THz]の平面波を入射したときの、時間ステップT=5000経過後の電界分布を示した。この場合、計算時間は CPU で 13.2[s] 、 GPUで3.025[s] かかり、4.36倍 GPU の方が早くなった。

4.3 マルチ GPU による FDTD 計算

１つの GPU ではメモリが少ないため、大容量のメモリを使う数値計算では、マルチ GPU が必須となる。マルチ GPU プログラミングとは、 GPU 内並列化とは別に、 GPU 間の並列化を行う方法である。領域を GPU 間で分け、さらに分けた領域を GPU 内で分割し、並列計算することによって高い計算処理速度を得ることができる。ここでは、マルチ GPU で高速化するため、領域を y 方向で３分割し、これらの領域をそれぞれの GPU に割り振り、計算を行った。複数のデバイスのメモリ上で配列を確保するために、 cudaSetDevice 関数を用いた。

図６に、計算結果を示した。領域分割の無い CPU での計算に比べ、 3 領域分割による4GPUでの計算時間の速さは、 25.79 倍となった。 GPU の数を増やすほど計算時間は減少していくが変化は小さくなっていくことがわかった。これは GPU メモリと CPU メモリ間のアクセス時間などの物理的処理にかかる時間が含まれるからであると考えられる。

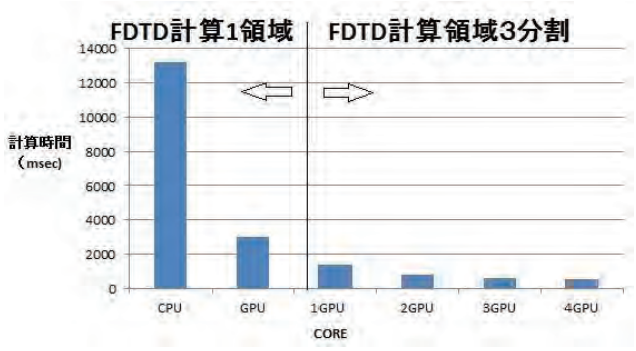

図６　 FDTD 計算結果

4.4 マルチ GPU による Peer to Peer 転送

Peer to Peer とはグラフィックボードのメモリから、直接別のグラフィックボードのメモリとデータのやり取りを行う機能である。このアクセス環境ができると GPU 間転送が速くなる。

cudaDeviceEnablePeerAccess で GPU 間の peer-to-peer アクセスを有効にすることができる。

図７に、Peer-to-Peer アクセスを用いた数値計算結果を示した。Peer-to-Peer アクセスを用いたことで、計算時間は減少した。 GPU 数が増えるにつれて、Peer-to-Peer による計算時間の減少量は縮まっている。しかし、総計算量に対する減少率は上昇していることから、減少効率は GPU 数が増えるにつれて、増加することがわかった。

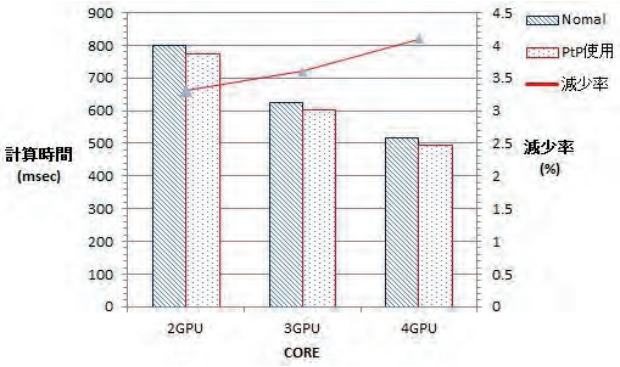

図７　 Peer-to-Peer による FDTD 計算結果

５．おわりに

本論文では、院生や教員が自由に使える高速並列計算機を学内に導入し、並列GPU を搭載した環境構築を目的として研究を行ってきた経過を報告した。また、FDTD法を用いた数値計算により、GPU を用いることで計算速度が早くなることと、その GPU 数を増やすことによって、計算時間が大幅に短縮されることが確認できた。

今後は、構築したサーバへの新たなソフトや環境の導入を通して、より広い分野における計算処理を行うことができるようになる見込みである。その経過を HP 上で公開していく予定である。さ

らに、 4GPU に最適化したマルチGPUによる 2 次元 FDTD や 3 次元 FDTD の並列処理の実現も目指していく。

謝辞

本研究を遂行するにあたり、協力して頂いた当研究室平成24年度卒業研究生の皆様に感謝致します。特に、サーバ構築及び GPU での数値計算に協力して頂いた、鈴木健二郎君に深く感謝致します。

参考文献

1 ） 松岡聡、遠藤敏夫、丸山直也、佐藤仁、滝澤真一郎, “ TSUBAME2.0 の全貌”， TSUBAME ESJ. , 1, pp.02-04 (2010.9) .

2 ） H. Shirasaki, “Scatterometry simulator for multi-core CPU," *Proceedings of the SPIE*, **7638,** pp. 7638V1-7638V6 (2010).

3 ） H. Shirasaki, “Scatterometry simulator using GPU and Evolutionary Algorithm," *Proceedings of the SPIE*, **7971,** pp. 79711T1-79711T7 (2011).

4 ） H. Shirasaki, “Scatterometry simulator development with GPU, real-coded GA and RCWA," *Proceedings of the SPIE*, **8681,** pp. 86811B1-86811B7 (2013).

5 ） 青木尊之、額田彰, “はじめての CUDA プログラミング”, 第 2 章、工学社 (2009) .

6 ） 青木尊之、額田彰, “はじめての CUDA プログラミング”, 第 3 章、工学社 (2009) .

7 ） 白﨑博公他、” Q&A 付き 光学実務資料集～各種応用展開を見据えて～”、 第10章 光学シミュレーション編、 (株) 情報機構 (2006)

2013年 3 月 1 日原稿受付

Received, March 1, 2013