水工学シリーズ　11-A-3

GPUスパコンを用いた
大規模流体計算の最前線

東京工業大学　学術国際情報センター　教授

青木　尊之

土木学会
水工学委員会・海岸工学委員会

2011年8月

GPUスパコンを用いた大規模流体計算の最前線

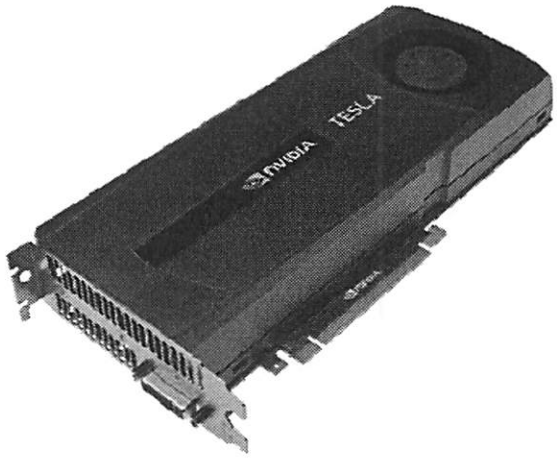
Large-scale Computational Fluid Dynamics on GPU-rich Supercomputer

青木尊之

Takayuki Aoki

1．はじめに

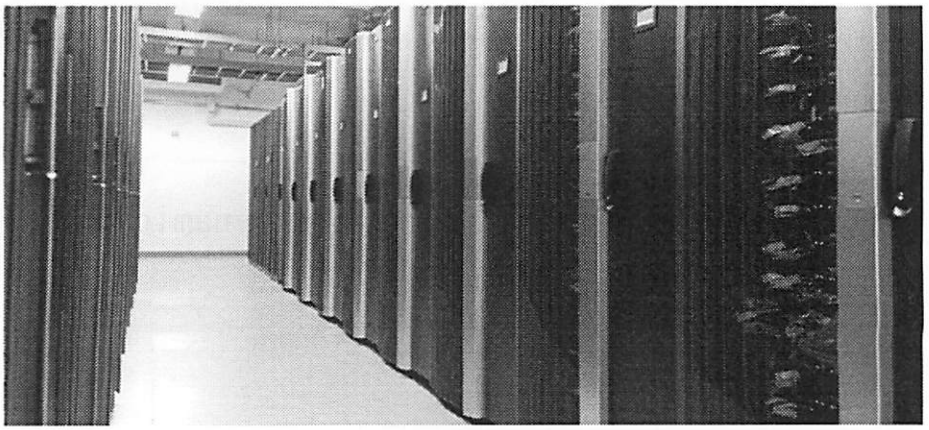
殆ど全てのパソコンにはグラフィクス・ボードやグラフィクス機能をもったCPUやチップセットが搭載されている。そのグラフィクス機能を支えるのが画像処理プロセッサGPU(Graphics Processing Unit)である。より高速に、より美しく、より精細な画像を表示することが求められ続け、その結果として描画機能が飛躍的に向上してきた。そして、GPUを画像表示だけではなく汎用計算に使う試み「GPUコンピューティング」[1]（GPGPU(General-Purpose computing on GPUs)と同義）が2000年頃から行われるようになった。2006年にNVIDIA社が自社のGPUに対してGPUコンピューティング用の統合開発環境としてCUDA(Compute Unified Device Architecture)[2,3]をリリースしたことが大きな節目となる。それまではCg (C for Graphics) 言語やHLSL(High Level Shader Language) により画像処理の機能を汎用計算に置き換えてプログラミングする必要があったが、CUDAの登場によりC言語でプログラミングすることができるようになり、一気にGPUコンピューティングが広まり始めた。さらに2009年にはCUDAのFORTRAN版もリリースされ、GPUの性能向上に合わせた頻繁なバージョン・アップとともに機能もますます向上している。2009年末には特定のGPUだけでなく、AMD社、Intel社のCPU, GPUやCellなどでも同じプログラムが動作するような標準化として策定されたOpen CL[4]も正式にリリースされ、GPUコンピューティングがより普及する環境が整ってきている。



図－１　NVIDIA社グラフィクス・ボード

GPUはポリゴンに対する幾何学的描画処理を高速化することに特化して開発されてきたため、単体で1 TFLOPSを超えるようなピーク演算性能を持つ。また、GPUコンピューティングはパソコンのグラフィクス市場で展開する製品を流用するため、利用コストが極めて低く、手元のパソコンにも装着できる、という大きな特徴がある。さらに、GPUは消費電力当たりの演算性能が高いため、スパコンの高性能化と低消費電力化に向けたアクセラレータとして広く認識されるようになっている。2008年には東京工業大学学術国際情報センターが680個のGPU (NVIDIA Tesla S1070) をスパコンTSUBAME 1.2に導入し、世界から大きな注目を集めた。NVIDIA社は2010年に倍精度浮動小数点演算性能の向上やECCメモリに対応したFermiコアを搭載したGPU（図－１）をリリースしており、GPUをHPC(High Performance

Computing)の分野で利用する条件が整った。2011年6月のスパコンTop500のランキングでは、2位と4位に中国のGPUスパコンが入り、5位に総合演算性能2.4PFLOPSの東京工業大学学術国際情報センターのTSUBAME 2.0（図－2）がランクインするなど、多数のGPUマシンがスパコンの上位に位置している。



図－2 東京工業大学のスーパーコンピュータ TSUBAME 2.0

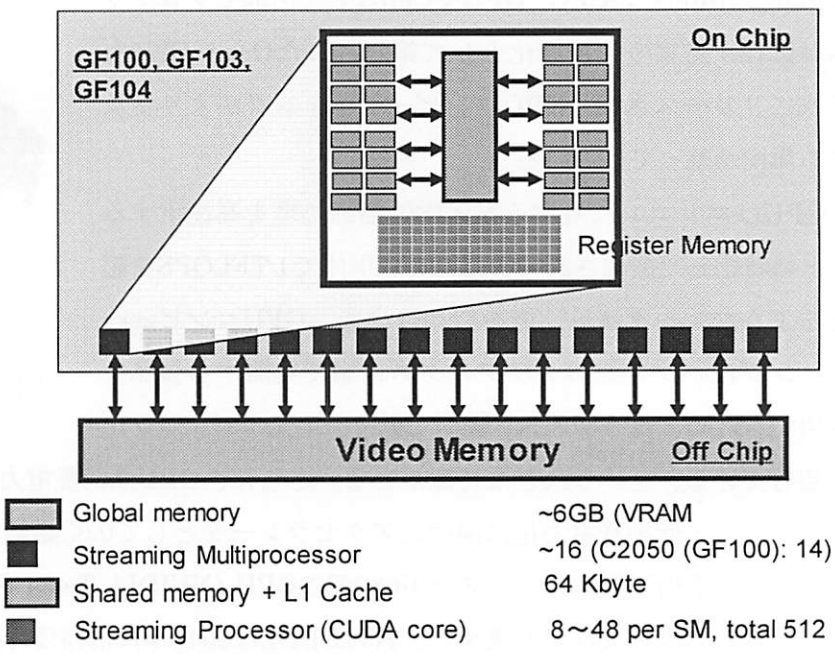
初期の頃のGPUコンピューティングは重力多体問題への適用で注目を集めた。GRAPEなどの専用計算機と同様に演算負荷の高い部分にGPUをアクセラレータとして利用し、高い実行性能を引き出すことができた。GPU がClearSpeedやGRAPEなど従来のアクセラレータと大きく違う点は、アクセラレータ・ボード上に搭載されるメモリに対して広いバンド幅を持つところである。そのため、GPUコンピューティングは演算負荷が極端に高いような限定された分野での計算だけではなく、物理、化学、金融、データ処理等の様々な分野のアプリケーションへの適用が進んでいる。

本稿では、GPUスパコンTSUBAMEを使うことにより、従来では行うことが困難であった大規模計算として、高解像度のメソスケール気象モデル、格子法による気液二相流のシミュレーションを紹介する。

2．ＧＰＵのアーキテクチャとプログラミング・モデル

GPUのアーキテクチャは汎用CPUと異なり、NVIDIA社の2011年の最新GPUは図－3のように1チップ当たり500個以上の演算プロセッサ（CUDAコア）を搭載している。さらに8～32個のCUDAコアが1つのストリーミング・マルチプロセッサを構成していて、そこにはL1キャッシュと共有メモリがある。GPUを搭載したボード上にビデオ・メモリがあり、GPUからビデオ・メモリへ100GB/secを超える高速なアクセスが可能なことやメモリが階層的構造になっている点も特徴的である。

GPUコンピューティングでは、多数の演算プロセッサを効率的に使うようにプログラミングすることが高い実行性能を達成するために必須である。このため、データ並列性の高い問題に対してGPUコンピューティングを適用



図－3　NVIDIA の Fermi コアGPUのアーキテクチャ

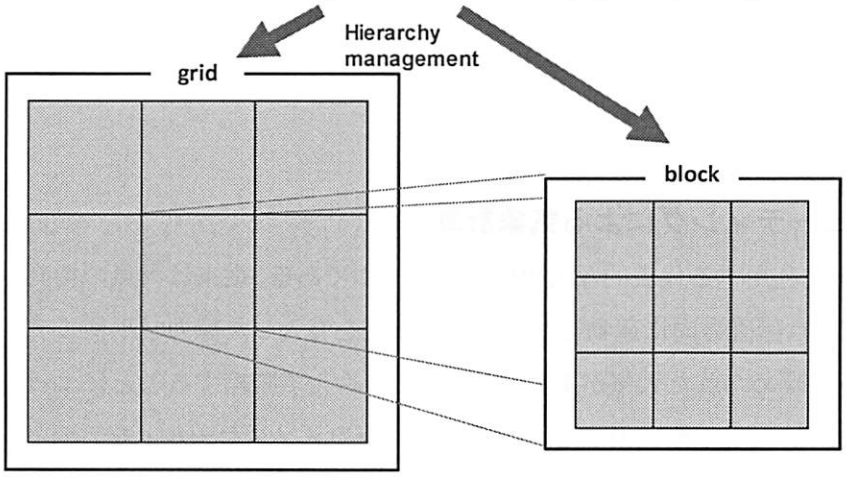
することが非常に有効である。マルチコアのCPUではコア数とほぼ同数のスレッドを実行させることが多いが、GPUの場合は高速にスレッドを切り替えるハードウェア・コントローラーが多数搭載されているため、数100個の演算ユニットに対して数10,000以上の数のスレッドを実行しても切り替えのオーバーヘッドが殆どなく、このような多数のスレッドで計算することにより始めて十分な性能を引き出すことができる。GPUコンピューティングのプログラミングは、ストリーミング・マルチプロセッサ単位ではSPMD(Single Program Multiple Data)であり、その中では32個のスレッド毎にSIMD(Single Instruction Multiple Data)として実行される。このようなアーキテクチャを前提にプログラミングすることにより、高い実行性能を引き出すことができる。

3．細粒度超多スレッド並列計算

CUDA によるGPUコンピューティングのプログラムの構造は、GPU上にメモリを確保したり、CPUとGPU間でデータ転送するようなランタイムAPIと、GPUのスレッドで実行させる内容を記述するGPUカーネル関数からなる。CUDAは標準的なC言語やFortran言語にGPUコンピューティング用の拡張がなされている。スレッドの識別番号を読み出せる変数などが宣言なしで利用できる。スレッドは、ハードウェア的には GPU の中の１つのプロセッサで実行される。

CUDA では数10万を超えるような多数のスレッドを、グリッドとブロックという概念で管理している。まず、ブロックの中で実行される 3次元（3変数）の数で複数のスレッドを管理し、それらはハードウェアでは１つのストリーミング・マルチプロセッサに割り当てられる。さらに、複数のブロックをグリッドの中で3次元の数で管理するという階層的構造を持つ。それらの指定は非常に簡単で、図－４のように通常のC言語プログラムの関数呼び出しと同じように記述し、

```
CPU:  function(a, b, c, . . . );
GPU:  function<<< grid,  block >>>(a, b, c, . . . );
```



図－４　CUDA のグリッドとブロックによるスレッドの階層的管理

<<< grid,　block　>>>で挟んだ中の2変数で実行するスレッド数を指定している。図－４では、ブロックの中に9個のスレッドがあり、グリッドの中に9個のブロックがあるので、スレッドの総数は81となる。GPUコンピューティングの本質は、超多スレッドを同時に実行する並列計算であり、そのような並列性を見出すことが重要になる。幸い科学技術計算の殆どに並列性があり、GPUの実行性能を十分に引き出すことができる。要素数がNの1次元配列のコピーを例にとる。CPUで普通に計算する場合は、

```
for(j = 0; j < n; j++) A[j] = B[j];
```

のように書くが、CUDAのプログラムは

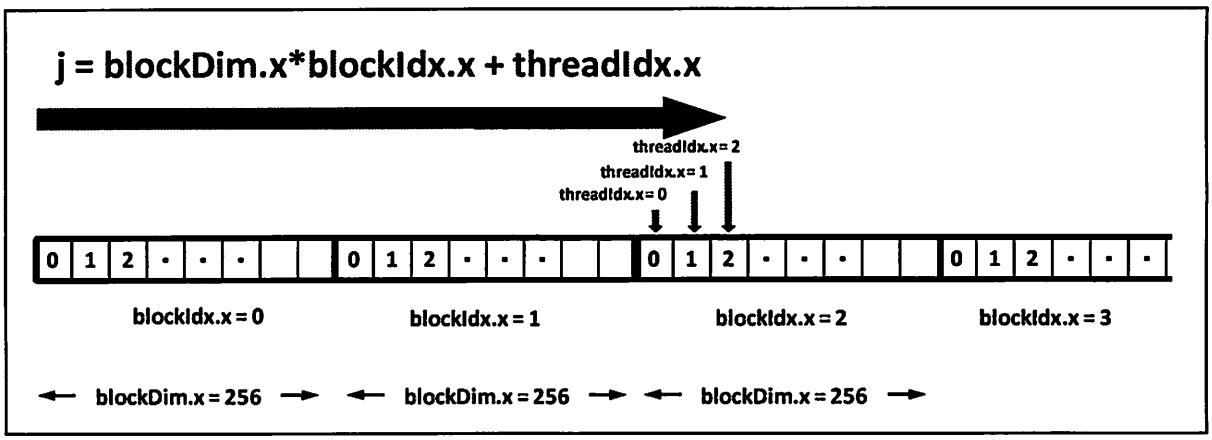
```
__global__  void  copy(float *A, float *B)
{
int j = blockDim.x*blockIdx.x + threadIdx.x;

A[j] = B[j];
}
```

となる。**blockDim.x**はブロックの中のスレッドの数、**blockIdx.x**はグリッドの中の当該ブロックのID番号、**threadIdx.x**は当該ブロックに中の当該スレッドのID番号であり、これらはビルトイン変数である。これらの変数を使って、配列のインデックスを図－５のように計算している。カーネル関数**copy**の実行は、

```
copy <<<N/256,  256 >>> (A, B);
```

のように行う。



図－５CUDA プログラミングのスレッド識別番号と配列インデックスの対応

CUDA のプログラミングでは、CPU で計算する場合に使っていた for ループが消失し、N 回ループを繰り返す代わりにN個のスレッドが実行されている[3]。

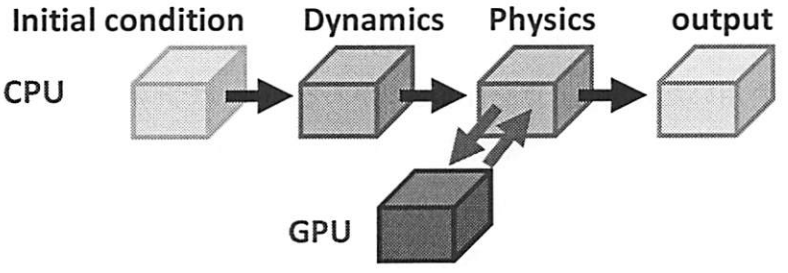
４．ＧＰＵコンピューティングによる気象計算

気象計算はスパコンを利用する代表的な大規模計算の一つである。従来は大気の鉛直方向の圧力勾配と重力が釣り合っていて、大気は鉛直方向に運動しないという仮定を導入した静力学平衡モデルで計算が行われていたが、最近は地表付近で暖められた大気が上昇し、水蒸気が冷えて凝縮することによって雲が発生するプロセスの重要性が指摘され、静力学平衡を仮定しない非静力平衡のメソスケール・モデルとよばれる計算が行われるようになっている。積乱雲などの典型的なサイズは10km程度であり、雲を解像するためにより狭い格子間隔とより多くの格子点数を使う大規模計算が必要となっている。さらに、気象予報の目的に対しては、できるだけ短時間で計算を終了させる必要があるため、気象計算ではスパコンを如何に効率良く利用し、大規模計算を行うかが大きなテーマとなっている。

気象計算は力学過程と呼ばれる風速や気圧、湿度などの予報変数に対する流体力学的な運動の計算と、物理過程と呼ばれる水蒸気の凝縮や雲形成、降雨などの局所的な物理変化のモデリング（パラメタリゼーション）に対する計算に分けられる。力学過程では浮動小数点演算よりも圧倒的にメモリアクセスに時間がかかり、ど

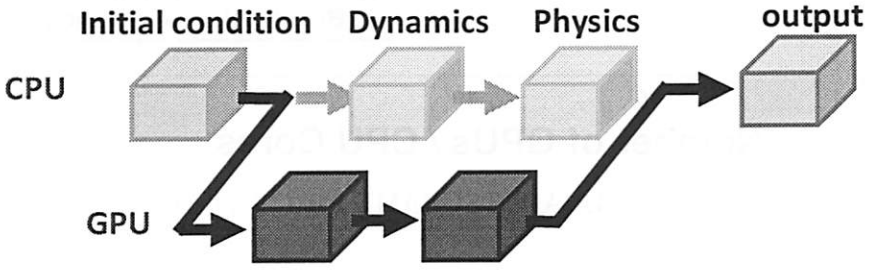
の計算機でもピーク演算性能に対して十分高い実行性能は得られない。一方、物理過程はさまざまな経験的なモデリングやパラメータを多く含み、一部の計算は非常に負荷の高い浮動小数点演算を必要とする。

米国大気研究センターを中心に開発されている次世代メソスケール大気モデル WRF (Weather Research and Forecasting)は世界標準になりつつある研究用のオープンソースのコミュニティ・コードであり、執筆時点で世界第3位のスパコン Jaguar上でも実行され 50TFLOPSの実行性能が得られたことを報告している。WRF[5]の開発グループはいち早くGPU を利用する取り組みを行わっている。WRFでは物理過程における計算負荷の高いモジュールの一部を GPU に移植し高速化を図った[6]。計算の全体は従来通り CPU 上で実行し、GPU 化したモジュールの部分を計算する際には、先にCPUのメモリからGPUのメモリに計算に必要なデータを転送しておく。GPUで計算した結果はGPUのメモリに出力されるため、CPUで計算を続行させるためにはGPUからCPUのメモリへのデータ転送も必要になる。WRFの時間積分の毎ステップでこのようなCPU-GPU 間の通信が頻繁に発生し、この通信時間がボトルネックとなる。GPUに移植したモジュール単体ではCPUの 20 倍の高速化に成功しているが、計算全体では 30% 程度の速度向上に留まり、GPU の持つ本来の性能を十分に発揮できない結果となった[6]。GPU コンピューティングにより高い実行性能を達成するには、可能な限りCPU-GPU 間の通信を排除する必要がある。そのためには、気象モデルの時間積分ループの中の全てのサブルーチン（関数）をGPU化する必要がある。



図－６物理過程の一部をGPU計算で加速する気象計算のフロー

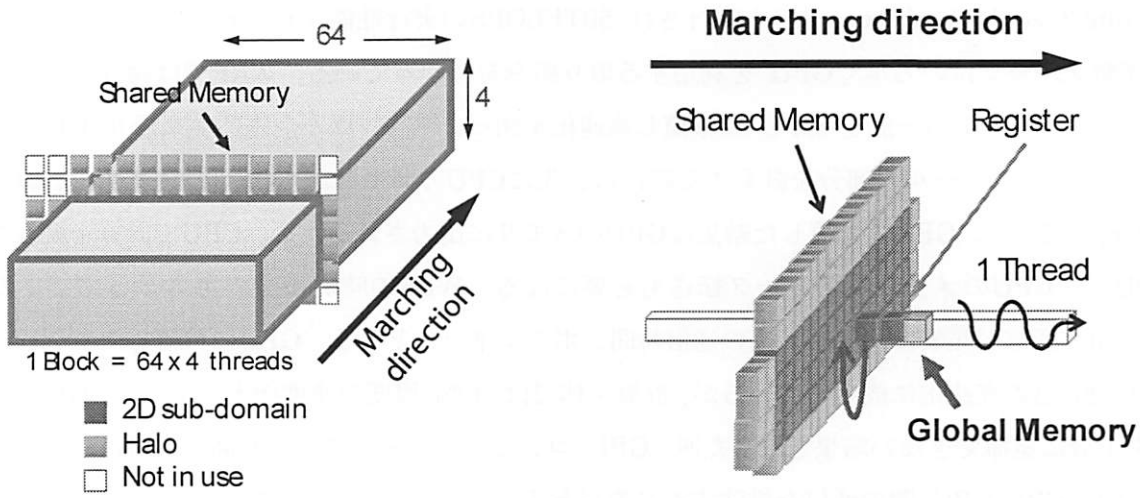
我々は気象庁数値予報課と協力し、次期気象予報モデルとして開発しているASUCA[7]の物理過程と力学過程の全てのモジュールのGPU化を行った。GPU化のためにはASUCA のコード全体を一からCUDAに書き直す必要がある。物理過程の各モジュールは単体のGPU化が可能であり、移植は比較的容易である。一方、力学過程の計算は隣接格子点へのアクセスを伴うため、予報変数は終始GPUボードのビデオ・メモリ上に確保しておき、一度にGPU化する必要がある。力学過程の計算は圧縮性流体力学の方程式に基づいているため演算よりメモリアクセスが支配的である。GPU のビデオ・メモリへのアクセスはCPUのメインメモリへのアクセスと比較すると数倍以上高速であるため、GPUは力学過程においてもCPUより十分高速な計算が期待できる。



図－７　物理過程と力学過程の全てをGPU計算する気象計算のフロー

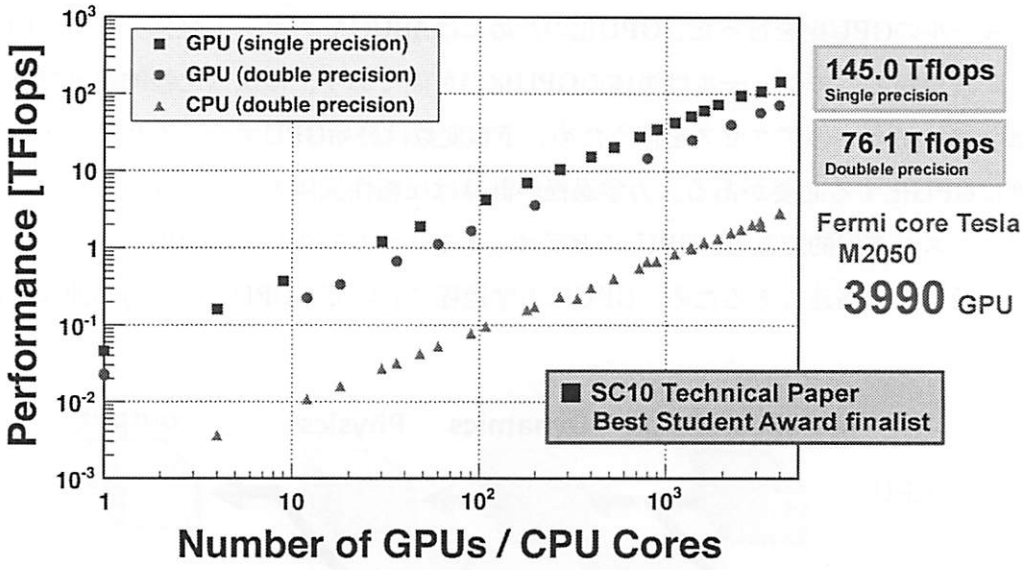
我々はFORTRANプログラミング言語で記述されたASUCAのコードを一旦 C/C++ 言語に書き直し、その

後、CUDAに書き換えている。さらにストリーミング・マルチプロセッサ内の共有メモリをキャッシュ的に使うアルゴリズムやレジスタを有効利用するなどの多数の技法を導入し、最終的にIntel CPU Xeon X5670 の1ソケット（6コア）に対して、NVIDIA GPU Tesla M2050 の1ソケット(448 CUDAコア)が12倍高速に計算できることを示している[8]。



図－８　力学過程のShared メモリおよびレジスタの利用による高速化

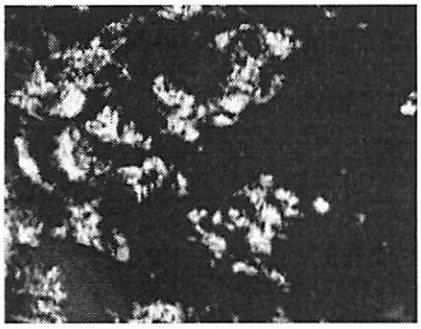
一つの問題点はGPUボード上のビデオ・メモリはせいぜい数GBであることであり、実際の気象モデルを動作させるには複数GPUを用いた大規模計算を行う必要がある。CPUで計算するときと同じように単体GPUカードのメモリで計算可能なサイズにまで計算領域の分割し、それぞれを各GPUに割り当てる。現時点のCUDAではGPUのメモリ間で直接データ通信を行うことができず、CPU側のメモリを介して通信する必要がある。大規模計算では、このGPU間のデータ通信が大きなオーバーヘッドになるため、通信と計算のオーバーラップの技術も開発されている。これらにより、ASUCAのGPU版はTSUBAME 2.0 の 3990個のGPUを使って145TFLOPSという非常に高い実行性能を達成している[9]。



図－９　気象モデルASUCA のTSUBAME 2.0 におけるGPU計算の実行性能

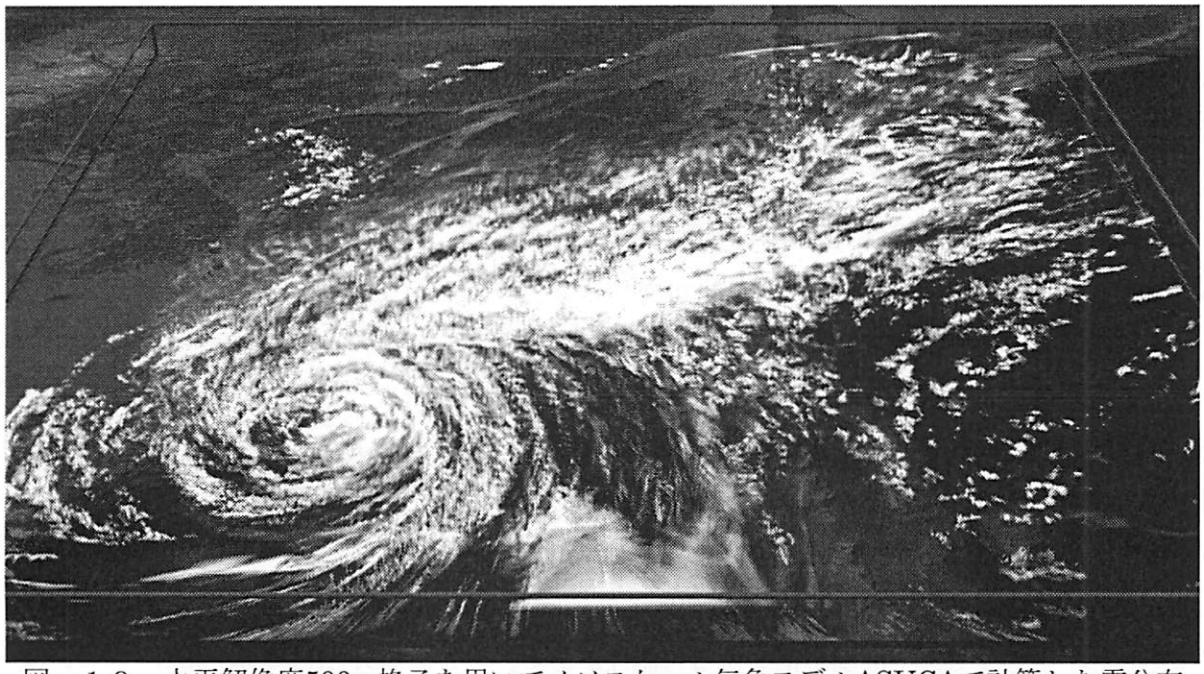
図－１０は気象庁メソモデル(MSM)のデータを初期条件・境界条件とし、TSUBAME 2.0 の 437 GPU を用いてGPU版のASUCAにより4792 × 4696 × 48（水平500m格子）の格子で、初期時刻 2009年10月6日

15UTCから計算した9時間後の雲の分布を可視化している。GPUコンピューティングにより、実運用を目指している気象予報モデルがCPUで計算するよりも10倍以上高速に計算でき、さらに10倍以上少ない消費電力で計算できることが示された。



水平500m格子で計算した雲分布

水平5km格子で計算した雲分布



図－１０　水平解像度500m格子を用いてメソスケール気象モデルASUCAで計算した雲分布

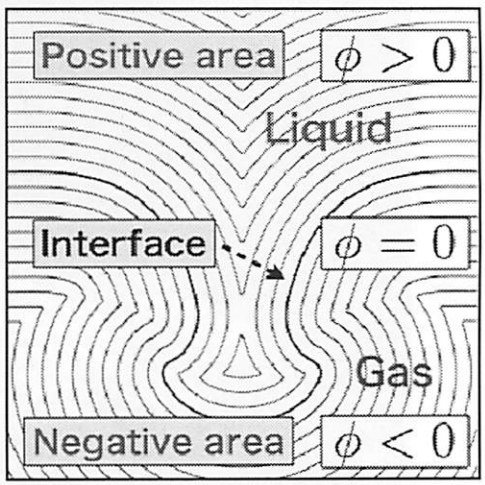
５．気液二相流のＧＰＵ計算

最近、ハリウッド映画でも水と空気が激しく入り混じるような流れを流体計算で行い、コンピュータグラフィクスで処理することで、実写ではできないようなシーンを作り出している。そこでは科学技術計算より高精細で大規模な計算が行われていることに驚かされる。GPUコンピューティングは当初は重力多体問題での粒子計算で成功したことが影響してか、気液二相流などのGPU計算に対しても、SPH(Smoothed Particle Hydrodynamics)法などの粒子法が使われていた。粒子法はカーネル半径という影響範囲内の粒子と相互作用して粒子に及ぼされる力を求め粒子を移動する計算を行っている。3次元計算になると、低次精度にも拘わらずカーネル半径内の粒子数が増え、ランダムなメモリアクセス、演算量、計算精度という関連から効率が悪くなる。特に半陰解法では圧力のPoisson方程式の非ゼロ要素数が多くなり、疎行列解法および分散メモリ環境での並列計算という観点からも効率低下の原因となる。また、気液界面での非物理的振動の発生や単一粒子が界面から離れた場合の精度（スプラッシュに似ているが非なるもの）、強過ぎる数値粘性（十分に渦を表現できない）等の問題がある。

有限差分法、有限体積法、有限要素法などの格子法での計算は、隣接格子点へのアクセスとなり、有限差分法では高次精度スキームの導入で高精度かつ高効率な計算を行うことができる。ハリウッドの映画制作においても、リアルな水のシーンなどは粒子法から格子法での計算に変わってきている。格子法で気液二相流を計算する代表的な計算手法として界面捕獲法がある。気体と液体を密度や粘性などのパラメータが異なるだけの同じ流体として統一的に計算し、気体と液体を区別するために識別関数を導入し界面を表現する手法である。また、界面で密度比が1000倍近く変化する流体に対するNavior-Stokes方程式を非圧縮性流体として解くので、速度の発散がゼロ（$\nabla \cdot u = 0$）という条件を満足するために解く圧力のPoisson方程式が行列解法という観点で解き難くなる欠点がある。ここでは、格子法による気液二相流に必要な全ての部分をGPUで計算し、CPUで計算では困難であった高解像度・大規模気液二相流計算の結果を示す。

５．１　気液界面の捕獲法

気液界面は2次元面であるが、これを表現するために3次元の識別関数の断面を利用する手法が良く使われる。3次元の識別関数は情報量として冗長に思えるが、気泡の合体・分離などのトポロジー変化を容易に表現することができる。代表的な識別関数としてLevel Set 関数を使う方法[10]や気体・液体の流体体積率を用いるVOF(Volume of Fluid)法等が挙げられる。Level Set 関数は界面からの符号付距関数であり、例えば図－１１のように液体側の領域は界面からの距離に正の符号を付け、気体側の場合は距離に負の符号を付ける。こうすることにより界面はLevel Set 関数の0の等値面として表現される。なめらかなプロファイルを持つため、曲率等を精度よく求めることができる利点がある。Level Set 法（特にParticle Level Set 法）は界面形状を表現する精度は高いが、本来保存されるべき気体・液体の体積の保存が保証されず小さな気泡や液滴が消失する可能性がある。一方、VOF法は気体・液体のそれぞれの体積を保存するが、格子サイズに近い曲率半径を持つ気液界面形状を精度よく表現することができない。ここでは、THINC[11] WLIC法[12]というVOF法ベースで気液界面の広がりを阻止するアンチ拡散が入った手法で界面を捕獲し、表面張力や接触角を評価するためにLevel Set 法を導入している。



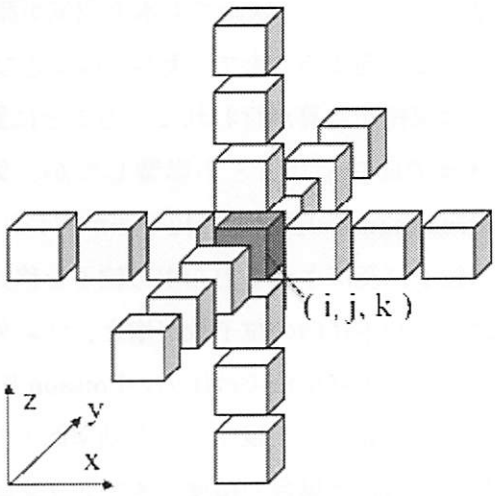
図－１１　Level Set 関数の等高線表示

表面張力および壁との接触角については、BrackbillのCSF (Continuous Surface Force) モデルにより有限幅に力を分散させ解いている。

５．２　非圧縮性Navior-Stokes方程式の計算

５．２．１　移流計算

Navior-Stokes方程式の移流項の計算、Level Set 関数の再初期化計算には、5次精度のWENO スキームを用いる。単調性の確保が高波数成分をフィルタリングしていて、両方の計算にける安定性の確保に貢献している。5次精度WENOスキームのステン

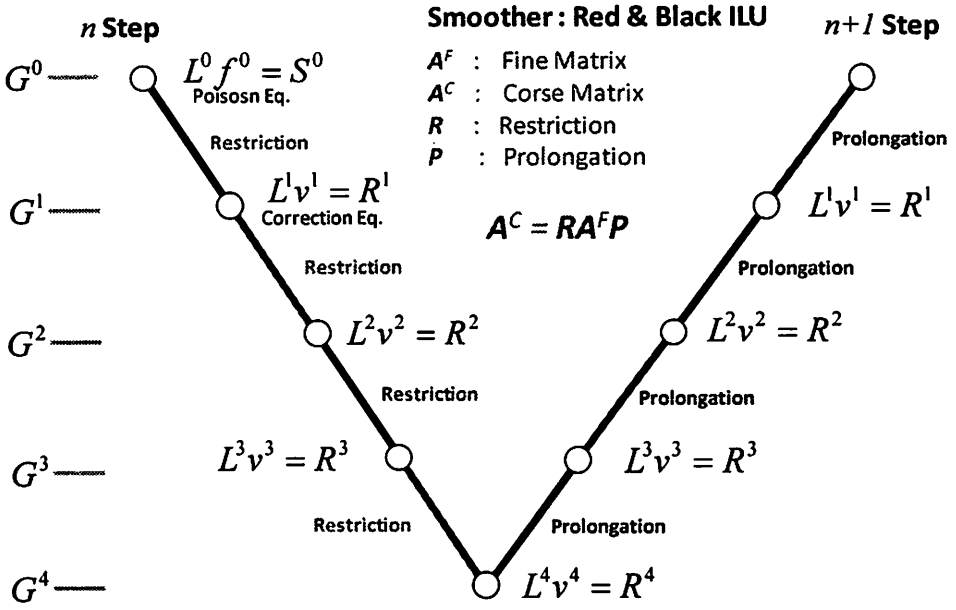


図－１２　5次精度WENO スキームのステンシル

シル・アクセスは図2のようになり、NVIDIA GPU のshared メモリをSoftware Managed Cacheとして使うことによりビデオ・メモリへのアクセスを低減できる。この部分だけなら1GPUで300GFLOPSを超える高い実行性能が得られている。

５．２．２　圧力Poisson方程式の解法

気液二相流計算において最も計算時間を要する部分がPoisson方程式を解くための連立一次方程式の計算である。ここでは構造格子を用いているので非ゼロ要素の位置は規則的な疎行列となるが、気液を統一的に解くために界面に急峻な密度変化が生じ悪条件の疎行列となる。そこで、クリロフ部分空間での反復法であるBiCGSTAB法をマルチグリッド法による前処理と組み合わせた収束性の高い疎行列解法を導入している。マルチグリッド法は並列計算にも適用可能なアルゴリズムであり、Red & Black 法を用いたILU(0) 法をスムーザとして用いることで安定した収束性を確保している。

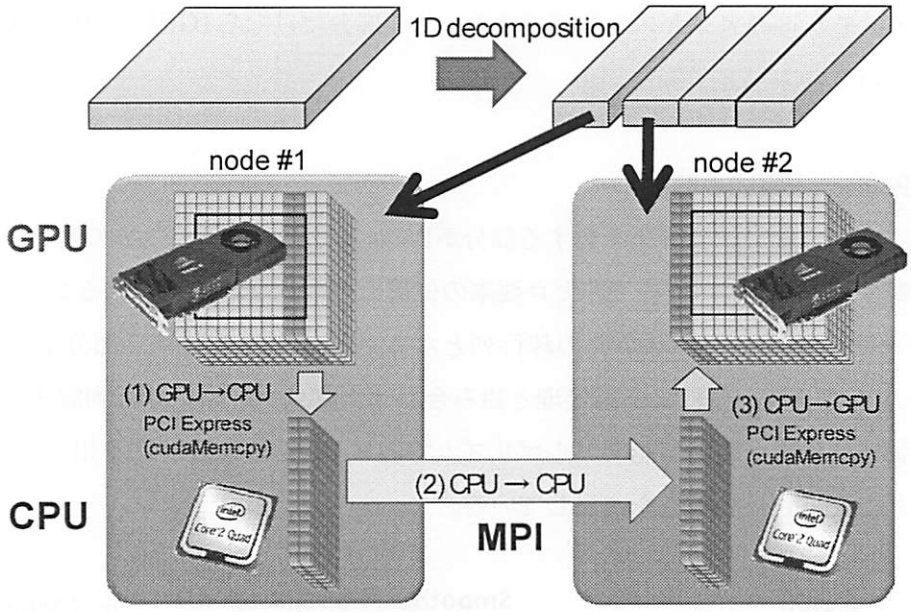


図－１３　マルチグリッド前処理のVサイクル

５．３　気液二相流の複数GPUによる計算

これまでに述べた気液二相流計算のためのコンポーネントを全てGPU計算として実装し、速度、圧力、Level Set 関数、VOF関数と言った従属変数をGPU上のメモリに置くことにより、CPU側からはGPU計算を実行するカーネル関数をcall(実行命令)するだけとなり、CPUとGPUの頻繁なデータ交換を排除することができる。これによりGPU本来の演算性能、メモリバンド幅を有効に利用する計算を実行することができ、単一CPUコアに対して単一GPUで数10倍の実行性能を達成できる。

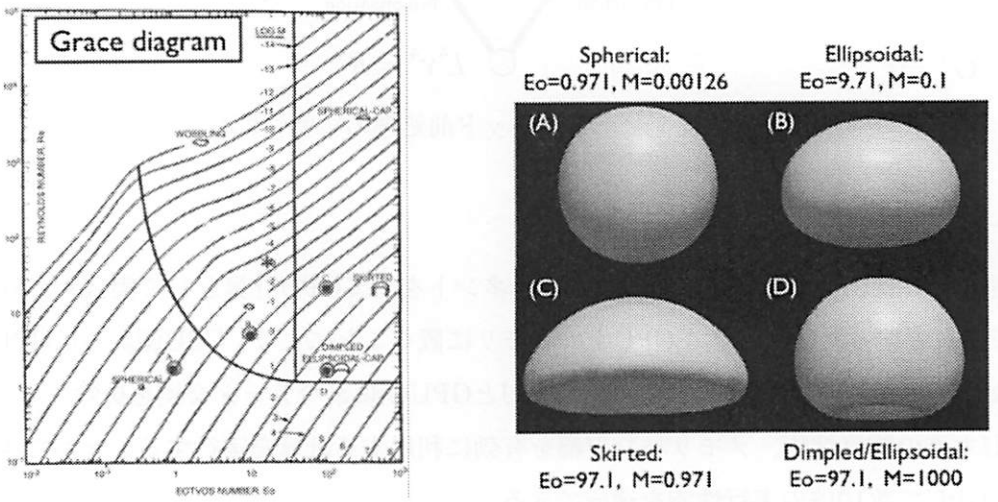
大規模計算に対しては複数のGPUを用い、各GPUのビデオ・メモリのサイズで計算可能になるように領域分割を行い、各GPUは割り当てられた領域のみを計算する。分割領域間のデータ通信が必要になり、GPUからCPU上のメモリを介してMPIライブラリを用いたデータ通信を行う。GPUスパコンでアプリケーションを実行する際には、ノードの演算性能に対してノード間のインターコネクションの性能が不足するため、GPU数を増やすにつれて通信時間が大きなオーバーヘッドになり、計算と通信をオーバーラップすることにより通信時間を隠ぺいするような工夫が必須となる。



図－１４　GPU計算におけるノード間通信

５．４　単一気泡の上昇の検証

気液二相流の基本的な検証として、単一気泡の上昇を計算した。Grace ダイヤグラムによると、単一気泡の上昇は図5のように無次元のEotvos数、Morton数、Reynolds数に応じて球型、楕円型、スカート型、くぼみ付き楕円型に分類される。その典型的なパラメータに対して得られた計算結果を図6に示す。定常的な上昇速度になったときの気泡形状はGrace ダイヤグラム[13]の分類と一致しており、さらに上昇速度は実験値[14]と非常によく一致している。

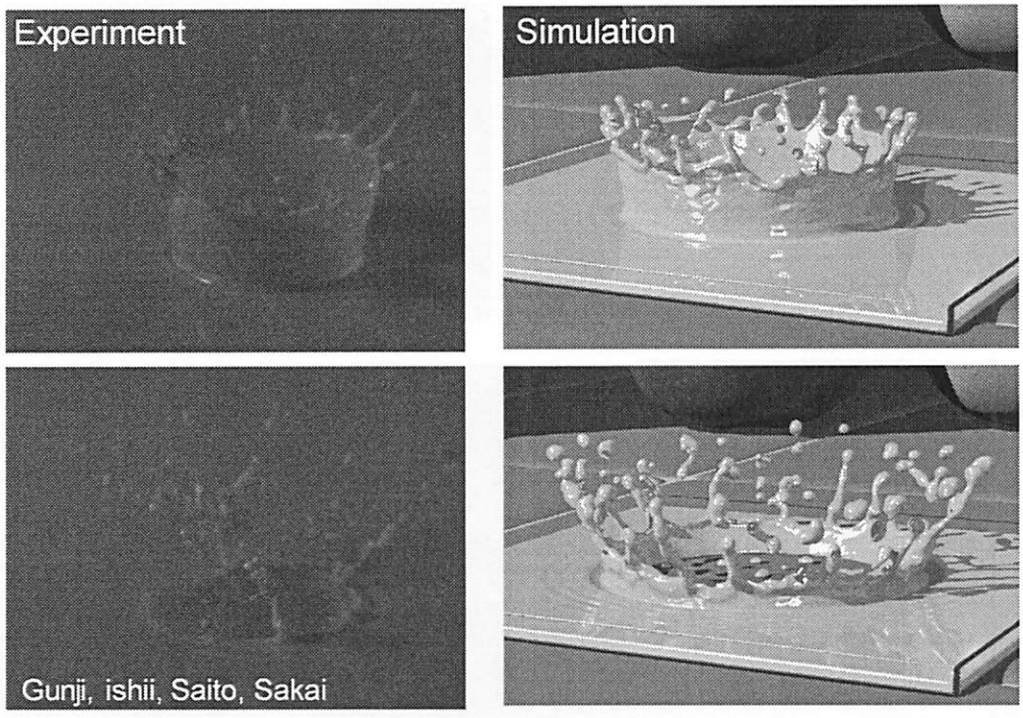


図－１５　無次元パラメータの違いによる上昇する単一気泡の形状変化

５．５　ミルククラウンの検証

静止している比較的浅い液面に液滴を落下すると、ミルククラウンが形成されることが良く知られている。フィンガーがでるメカニズムや本数など、未だに議論が続いているが、実験と同じ粘性、表面張力の下で、同じ速度で液滴を落下させミルククラウンの形成を計算した。落下速度や液の厚さに応じてミルククラウンの形

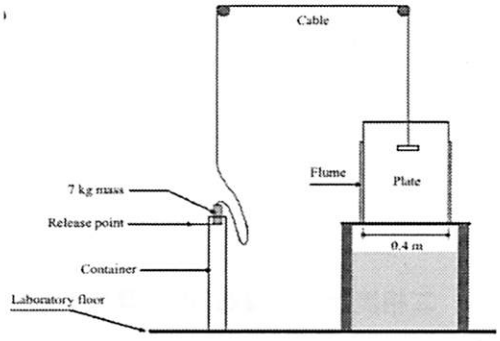
成の様子が大きく変化するが、実験との非常に良い一致が得られている。

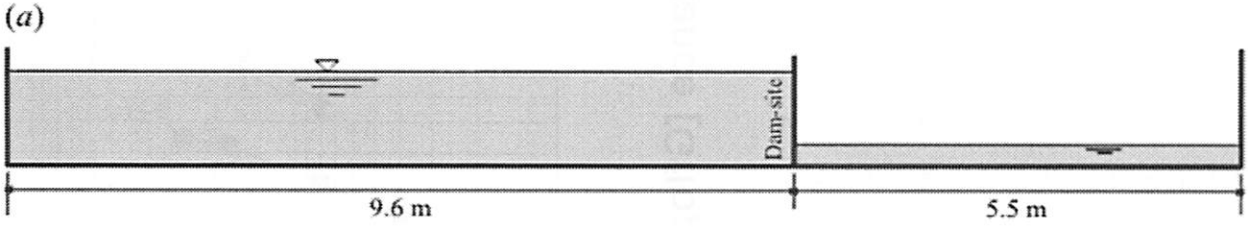


図－１６　ミルククラウンの実験と計算の比較

５．６　ダムブレークの濡れた床への浸水

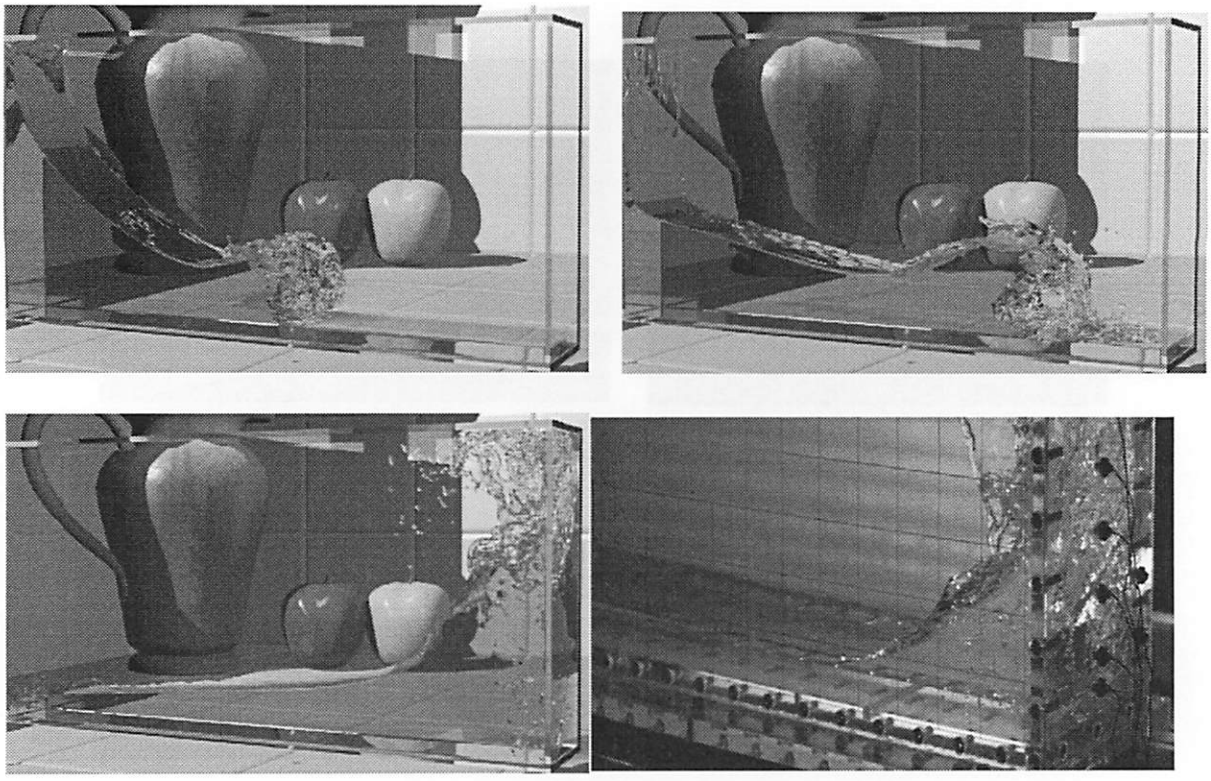
より複雑な気液二相流としてダムブレークの計算を行い、九州大学で行われた実験との比較も行った。通常は溜めておいた水を乾いた床に進水させ、その先端の速度などを実験と比較するが、ここでは濡れた床へ水を浸水させる計算と実験を行う。濡れた床の場合、浸水してくる水に対を塞き止める効果が大きく、後から来る水の速度が先端速度を上回るために進水直後から砕波が起こり浸水波は大きく乱れる。





図－１８　濡れた床へのダムブレーク実験のセットアップ

実験との比較は必ずしも同一条件ではないが、計算は砕波の過程を良く再現している。また、砕波や壁との衝突により小さな気泡が巻き込まれている様子も良く捉えられている。巻き込み際の水面形状は初期の水面の高さの比に依存し、水高の比が大きいほど噴流の角度が浅く進行方向に巻き込む。

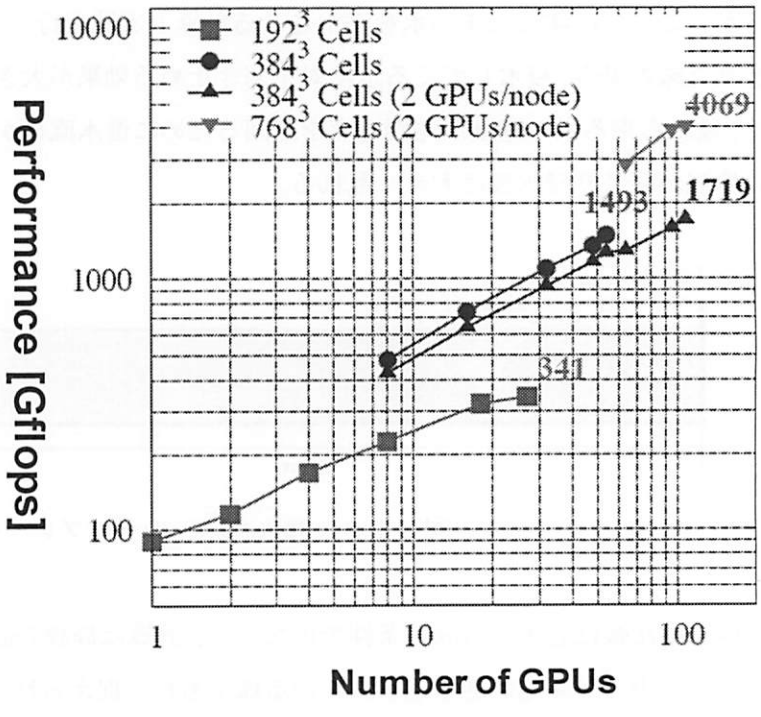


図－１９　GPU によるダムブレーク計算と実験の比較

計算条件は72cm×12cm×36cm ののの計算領域に1.8cm の水面を設定し、初期に幅15cm の水柱を設定し重力加速度を9.8m/s²とした。気相および液相の物性値はそれぞれ空気、水の値を用いた。576× 96×288 格子を用いて計算した結果を図－１９に示す。崩壊初期の段階で水柱の下部先端から噴流が発生し、巻き込みの先端が激しく乱れ、砕波が生じる過程を再現できた。

５．７　二相流計算の複数GPU計算

単一GPUのビデオ・メモリでは計算できない大規模計算を行うためと、計算の高速化を図るために複数GPUで二相流を計算する。直交格子上での計算を対象としているため３次元領域分割による並列化が有効であり、分割された各領域にそれぞれGPUを割り当てることで複数GPUの並列計算が可能となる。割境界近傍３ステンシル分の格子の計算では隣接領域のGPU のビデオ・メモリ上にある格子点へのアクセスを伴うため、ここでもGPU 間のデータ転送が必要になる。ここではTesla S1070が搭載されているTSUBAME 1.2の60 ノード(120GPU :2GPU/node) のうち1GPU



図－２０　二相流計算における複数GPUの実行性能

～108GPU を用いて計算を行った。計算格子192×192×192, 384×384×384, 768×768×768, のサイズのニ相流計算に対し、複数GPUを用いた計算の実行性能を図－２０に示す。108GPU 並列まで強スケーラビリティであり768^3格子のケースでは108GPU を用いて4 TFLOPSの実行性能を達成している。

参考文献

[1] GPGPU URL http://gpgpu.org/

[2] NVIDIA , 2010: CUDA Programming Guide 3.2,http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUDA_C_Programming_Guide.pdf (2010)

[3] 青木尊之, 額田彰著, 2009: はじめてのCUDAプログラミング, 工学社, 249pp.

[4] Khronos group: Open CL URL http://www.khronos.org/opencl/

[5] J. W. Skamarock, J. B. Klemp,J. Dudhia, D. O. Gill, D. M. Barker, M. D. Duda, Huang, X. Y., Wang, W., and Powers,J. G., 2008: A Description of the Advanced Research WRF Version 3, National Center for Atmospheric Research (2008).

[6] J. Michalakes, and M. Vachharajani: GPU acceleration of numerical weather prediction, in IPDPS. IEEE, pp. 1–7 (2008).

[7] J. Ishida, C. Muroi, K. Kawano and Y. Kitamura: Development of a new nonhydrostatic model "ASUCA" at JMA, CAS/JSC WGNE Research Activities in Atmospheric and Oceanic Modeling (2010)

[8] T. Shimokawabe, T. Aoki, C. Muroi, J. Ishida, K. Kawano, T. Endo, A. Nukada, N. Maruyama and S. Matsuoka: An 80-Fold Speedup, 15.0 TFlops Full GPU Acceleration of Non-Hydrostatic Weather Model ASUCA Production Code, in SC '10: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. New York, NY, USA: ACM. (2010)

[9] T. Shimokawabe, T. Aoki, J. Ishida, K. Kawano, C. Muroi: 145 TFlops Performance on 3990 GPUs of TSUBAME 2.0 Supercomputer for an Operational Weather Prediction, First International Workshop on Advances in High-Performance Computational Earth Sciences: Applications and Frameworks (IHPCES), Singapore (2011).

[10] M. Sussman, P. Smereka and S. Osher: J. Comp.Phys., Vol. 114, pp.146-159 (1994)

[11] F. Xiao, Y. Honma and T. Kono: Int. J. Numer.Method. Fluid., Vol. 48, pp.1023 (2005)

[12] K. Yokoi: J. Comp. Phys., Vol. 226, pp.1985-2002 (2007)

[13] J.R.Grace: Transactions of the Institution ofChemical Engineers, Vol. 51, pp.116-120 (1973)

[14] M. van SintAnnaland, N.G.Deen andJ.A.M.Kuipers: Chemical Engineering Science,Vol. 60, pp.2999-3011 (2005)