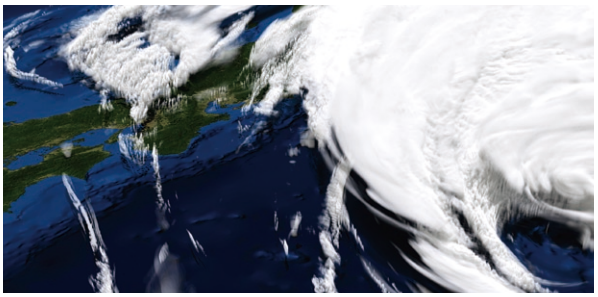
e-Science Journal
東京工業大学 学術国際情報センター
Vol. 12

TSUBAME ESJ.



将来のスーパーコンピュータにおける エクストリームなビッグデータ処理にむけた TSUBAME2での取り組み
Extreme Big Data with TSUBAME2 and Beyond

OpenACCを用いた全球雲解像モデル NICAM力学コアの大規模GPU計算
A Global Atmosphere Simulation on a GPU Supercomputer using OpenACC: Dry Dynamical Core Tests

気象計算のための GPUコンピューティング・フレームワーク
High-productivity Framework on GPU-rich Supercomputers for Weather Prediction Code

http://www.gsic.titech.ac.jp/TSUBAME_ESJ

将来のスーパーコンピュータにおけるエクストリームなビッグデータ処理にむけたTSUBAME2での取り組み

佐藤 仁* 上野 晃司** 白幡 晃一** 社本 秀之** 松岡 聡*
*東京工業大学 学術国際情報センター **東京工業大学大学院 情報理工学研究科

現状の「ビッグデータ」処理の多くはウェブサーバ由来のHDDやギガビットイーサネットが搭載されたクラウド基盤上でのMapReduce処理などが行われている。一方で、現代の最新鋭のスーパーコンピュータでは、GPUアクセラレータや不揮発性メモリデバイスなどの新しいコモディティデバイスの搭載が進んでおり、既存のビッグデータ処理の性能を大幅に改善できると期待される。しかし、このようなGPUアクセラレータや不揮発性メモリデバイスなどが多数搭載された大規模な計算環境で動作するソフトウェアの開発については、現状ではまだあまり進んでいない。本稿では、将来のエクストリームスケールなスーパーコンピュータやクラウドデータセンターでのビッグデータ処理にむけたTSUBAME2での研究開発の動向として、Graph500、GPUアクセラレータによるMapReduce処理、大規模分散ソート、などの具体的な事例ついて紹介する。

1 はじめに

近年、ソーシャル・ネットワーク、バイオインフォマティクス、モノのインターネット(IoT)などの様々な分野において「ビッグデータ」が注目されており、ペタバイト～ヨッタバイトスケールの大規模なデータに対するスケーラブルで高速な処理を行う要求が増大している。既存の「ビッグデータ」処理の多くはウェブサーバ由来のHDDやギガビットイーサネットが搭載されたクラウド基盤上でのHadoopによるMapReduce処理などが行われている。一方で、現代の最新鋭のスーパーコンピュータでは、GPUアクセラレータや不揮発性メモリデバイスなどの新しいコモディティデバイスの搭載が進んでおり、既存のビッグデータ処理の性能を大幅に改善できると期待される。(図1) 例えば、GPUアクセラレータは、汎用的なワークロードに向くCPUと比較して、規則性のある特定のワークロードにおいては高いピーク性能とバンド幅性能を達成することができる。また、不揮発性メモリデバイスは、従来のDRAMと比較するとスループットやレイテンシなどの性能は劣るものの、低価格コストで、省電力、大容量である利点がある。これらの新しいデバイスは、将来のエクストリームスケールなスーパーコンピュータやクラウドデータセンターなどのビッグデータ処理を指向する大規模システムにおいても様々な利点がある。実際、TSUBAME2では、このような将来のシステムを見越して、いち早く、GPUアクセラレータや高性能なInfinibandネットワーク、フラッシュメモリなどを導入し、大規模なデータの高速処理にも対応可能なスーパーコンピュータを設計・開発し、運用している。しかし一方で、GPUアクセラレータや不揮発性メモリデバイスなど、これらのデバイスなどが多数搭載された大規模な計算環境で動作するソフトウェアの開発については、現状ではまだあまり進んでいない。本稿では、将来のエクストリームスケールなスーパーコンピュータやクラウドデータセンターでのビッグデータ処理にむけたTSUBAME2での研究開発の動向として、Graph500、GPUアクセラレータによるMapReduce処理、大規模分散ソート、などの具体的な事例について紹介する。

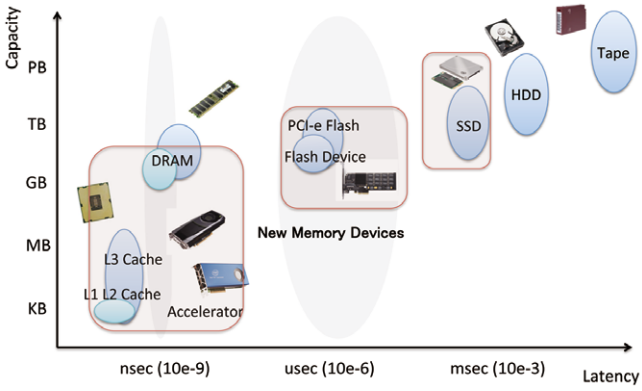


図1 階層的なメモリデバイス

2 グラフと高性能計算

2.1 スーパーコンピュータ向けのビッグデータ処理カーネルとしてのGraph500

グラフは、辺と頂点で記述された連結されたオブジェクトを表現するための基本的な数学的表現である。ヘルスケア、システム生物学、ソーシャル・ネットワーク、ビジネスインテリジェンス、電力網などの様々な重要アプリケーションがグラフを用いてモデル化されている。(図2) さらに、近年、このようなアプリケーション分野において様々なデータが大量に生成されるようになったため、大規模なグラフに対する高速処理の要求が非常に大きくなっており、高性能計算での重要なカーネルのひとつとなってきている。実際、従来のLinpackによるスーパーコンピュータの計算処理の性能を競うTop500リストに加え、近年、大規模なグラフ処理を行うことでスーパーコンピュータのビッグデータ処理能力を競うGraph500リストが登場してはじめている。図3にGraph500ベンチマークの概要を示す。現在のベンチマークではスケールフリーや直径の小ささなどの現実のネットワークをモデル化したクロ

ネッカーグラフに対して幅優先探索（BFS）を行った際の実行時間を計測する。幅優先探索とは、ある頂点から隣接している頂点への探索を繰り返し、グラフ内で隣接した頂点を全探索するアルゴリズムである。しかし、現状では、分散メモリで構成されたスーパーコンピュータ上での実行に適した幅優先探索のアルゴリズムには多くの最適化の余地がある。そこで我々は、大規模計算環境でのビッグデータ処理のハードウェア・ソフトウェアに関連する問題点を明らかにし将来のスーパーコンピュータの設計へ活かすことを目的とし、Graph500に関する研究開発を進めている。

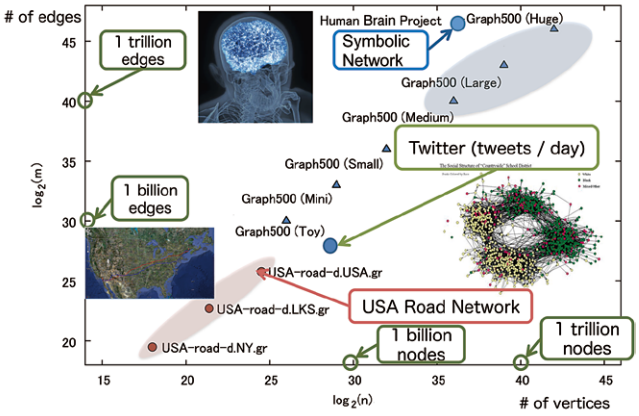


図2　様々な大規模グラフ

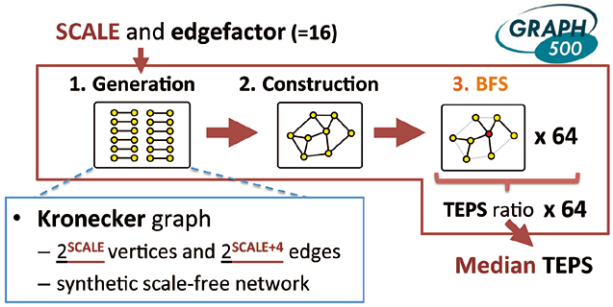


図3　Graph500の概要

2.2　スケーラビリティの問題

スーパーコンピュータでは、数千〜数万台の計算ノードを使った並列分散アルゴリズムが求められるが、BFSの並列分散アルゴリズムを設計するのは容易なことではない。単純なデータ構造や探索アルゴリズムでは、計算ノードの台数が数百〜数千台規模に増えたときネットワーク性能に律速されて全体の性能が頭打ちになってしまう。計算ノード数が数千〜数万台規模においても計算ノード数に比例した性能を得て、かつ、効率よく処理を行うためには、以下の条件を満たすアルゴリズム・データ構造が必要となる。

- 大規模に分割したときでも、メモリ使用量とアクセスコストをともに小さくできるグラフ探索向け疎行列のデータ構造
- 通信ノード数と通信データ量がともに低く抑えられる通信アルゴリズム
- グラフの分割数が増えても探索エッジ数を小さく保つ探索アルゴリズム

Graph500の参照実装では、データ構造や通信アルゴリズムなどの問題により、数百台の計算ノード規模で性能が頭打ちになってしまう。大規模並列分散向けBFSの最新の研究として、2次元分割のWaveを使った手法[3]があるが、この方法は、高速化に有効で他の実装でも広く用いられているHybrid BFSの手法を使っておらず、最適な方法ではない。また、Hybrid BFSを使った分散並列アルゴリズムとして、[4]が提案されているが、これもデータ構造や通信アルゴリズムの問題により、千台の計算ノード程度で性能が頭打ちになっている。

2.3　上野のアルゴリズム

Hybrid BFSアルゴリズムは、探索する頂点数を 大幅に削減するために、従来手法の、探索済みの頂点から未探索の頂点を探索する手法と未探索の頂点から探索済みの頂点を探索する手法をハイブリッドに切り替えながらBFSを行う手法である。上野は、Beamerらの並列分散Hybrid BFSアルゴリズム[5]を基盤として使用しつつ、通信データ量、メモリ使用量、計算量を減らして、数千〜数万台の計算ノード規模におけるスケーラビリティを実現するため、以下のような各種手法を適用したBFSの実装を開発した。

- Bitmapを使った疎行列表現
- 頂点濃度に応じてBitmapと疎ベクトルから選択
- グラフ探索に適したデータ構造
- 共有メモリを使ったノード間通信データ量

2.4　性能評価

TSUBAME2を1024ノードまで使って性能評価を行った結果を図4に示す。単位は、一秒間に処理した辺の総数を表すGTEPS（Giga Traversed Edges Per Second）である。参考に前回のグランドチャレンジ（2012年9月）までに得られた性能を示した。前回までに得られた性能は、GPUを使って高速化されているが、Hybrid BFSの手法を使用していない。その性能と比較してHybrid BFSと各種最適化により、1280 GTEPSを達成し、2.78倍の性能向上を示した。

また、図5にTSUBAMEにおけるGraph500のBFS実装のこれまでの性能向上を示す。2011年に100GTEPSを達成してGraph500リストにおいて3位を獲得して以降、継続的にスパコン上での幅優先探索の計算方法を研究してきた結果、現在では1280GTEPSと約13倍の性能向上を達成している。

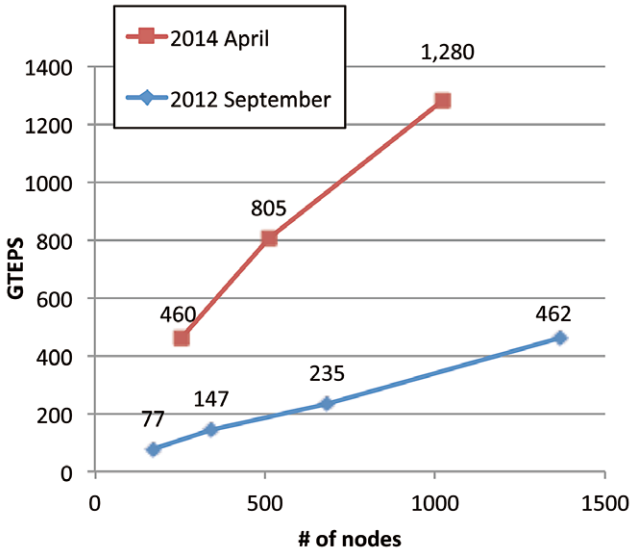


図4　TSUBAME2におけるGraph500の性能

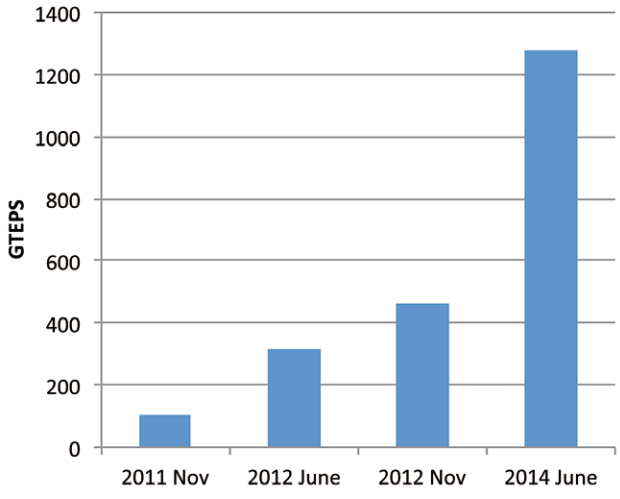


図5　TSUBAME2におけるこれまでの性能向上

GPUアクセラレータを考慮したMapReduce　3

3.1　MapReduceとメモリ階層の深化

大規模データ処理の分野では、MapReduceプログラミングモデル[6]が登場し、数千台規模のクラウド上のクラスタ型計算機上で局所性と耐故障性を考慮したスケーラブルなペタバイト-ヨッタバイト級のデータ並列処理を可能にしている。MapReduceでは、分散したkey-valueのペアデータに対して局所性を考慮した統一的な操作を並列で適用するために、並列データ処理のプロセスをMap、Shuffle、Reduceの3つの処理に分解する。そして、Map処理で入力データとなるkey-valueペアから中間データとなるkey-valueペアを生成し、Shuffle処理で同じkeyに対してvalueのリストを生成し、最後にReduce処理で中間データをShuffleすることにより得られたkeyとvalueのリストから最終出力となるkey-valueのペアデータを生成する。MapReduceは局所性を考慮したデータ並列処理を提供するという点でメモリ空間が階層的に分かれているGPUアクセラレータへも有効な手法であると考えられる。一方、数千〜数万台のアクセラレータを搭載したスーパーコンピュータへMapReduce処理を適用した事例は少ない。

3.2　HAMAR: メモリ階層の深化を考慮したスーパーコンピュータ向けデータ並列処理フレームワーク

そこで、我々は、現在、数千〜数万のアクセラレータないし不揮発性メモリデバイスを搭載したスーパーコンピュータ上でスケーラブルなデータ並列処理を目指したソフトウェア基盤としてHAMAR（Highly Accelerated Data Parallel Processing Framework for Deep Memory Hierarchy Machines）の開発を進めている。現在のHAMARの実装では（図6）、処理する対象のデータを動的にいくつかのチャンクに分割し、CPUとGPU間のデータ転送とGPUアクセラレータ上でのMap、Shuffle、Reduce処理をできるだけオーバーラップさせることで、GPUアクセラレータに搭載されたメモリの容量を超えるデータセットに対しても効率のよい処理が可能になっている。また、GPUベースの外部ソートなどOut-of-coreな基本アルゴリズムの実装を多く取り入れている。

3.3　ケーススタディ：GIM-V（Generalized Iterative Matrix-Vector multiplication）

HAMARを用いた実装のケーススタディとして、MapReduce型の大規模グラフ処理であるGIM-V（Generalized Iterative Matrix-Vector multiplication）と呼ばれる反復行列ベクトル積の一般化されたアルゴリズム[7]を実装している。いま、i、j が{1, ..., n}の値

をとるとし、$M = (m_{i,j})$ を $n \times n$ の行列、$v = (v_i)$ を大きさ n のベクトルとする。ここで、オペレータ $\times_G$ を導入することにより、GIM-Vアルゴリズムを次のように定義できる。

$$v' = M \times_G v$$
$$\text{where } v'_i = assign(v_i, combineAll_i(\{x_j \mid j = 1..n, \text{ and } x_j = combine2(m_{i,j}, v_j)\}))$$

上式は、*combine2*、*combineAll*、*assign* という3つのオペレータを用いることにより表現される（図7）。上の操作を、PageRank、Random Walk with Restart、Connected Componentといったグラフアルゴリズムの実装によって定義された収束条件を満たすまで反復処理する。これらのアプリケーションは上の3つのオペレータを定義することによって実現できる。

3.4 性能評価

Hamarを用いて、行列ベクトル積の一般化をMapReduceへ適用したGIM-VによりPageRankアルゴリズムを実装し、TSUBAME2.5の1024ノード（12288 CPUコア、3072 GPU）を用いて大規模な実証実験を行った。デバイスメモリの容量を超えるグラフデータ（171.8億頂点、2749億枝からなる大規模グラフ）を処理する場合に、1ノード当たり3GPUを使用した場合、2.81 GEdges/sec（1秒当たりに処理した枝数、47.7GB/sec）の性能となり、CPU上での実行に対して2.10倍の高速化を確認した。また、ウィークスケーリングの性能を計測した結果、1024ノード（3072GPU）を使用した場合に、1ノード（3GPU）を使用した場合に対して686倍の性能向上を示し、良好なスケーラビリティを示すことを確認した。（図8）

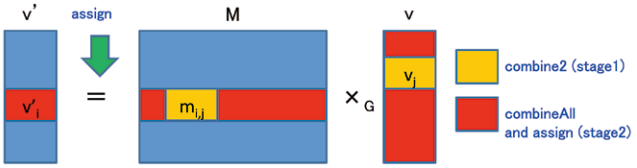


図7 GIM-Vの概要

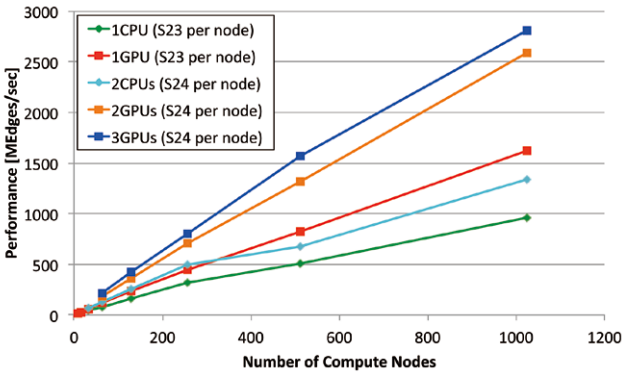


図8 TSUBAME2上でのGIM-Vのスケーラビリティ

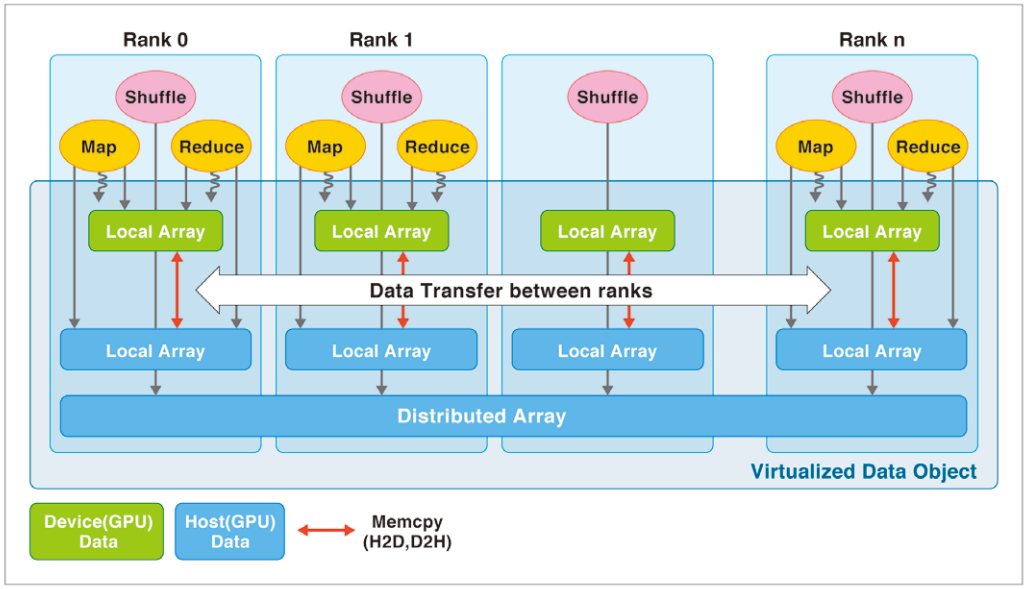


図6 HAMARの概要

GPU アクセラレータを考慮した大規模分散ソート 4

4.1 大規模分散メモリ環境でのソートアルゴリズム

ソートアルゴリズムは最も基本的なアルゴリズムの一つであり、ゲノム解析や多体問題など様々なアプリケーションのカーネルに使用されている。近年、このようなアプリケーションでの急激なデータの増加に伴い、膨大な量のデータをソートするために分散システム向けのソートアルゴリズムが数多く提案されてきた。その中でも、データの通信量を減らす事によって、高速にソートできるアルゴリズムとして、Splitter-based parallel sorting algorithmというアルゴリズムが広く知られている。Splitter-based parallel sorting algorithmでは、後述するように、スプリッターを導入することでデータの通信コストを減少させ、高速化に成功したが、相対的に計算処理のコストが高まり、全体の処理時間の大部分を占めるようになることが問題となっている。

4.2 Splitter-based parallel sorting algorithm

Splitter-based parallel sorting algorithmでは、次のような流れでソートが行われる（図9）。まずはじめに、それぞれのプロセスに配置されたデータを独立にソートする。その後、データの転送時の基準となるスプリッターを選択する。選択したスプリッターに合わせてプロセス間で通信を行い、データの転送を行う。最後に、マージ操作で、プロセス上のデータをソートされた状態にする。分散システム用のマージソートや基数ソートではデータ転送フェイズが数多く行われるが、このアルゴリズムではスプリッターを元にしたデータ転送によって通信回数を減らすことで通信コストを下げている。

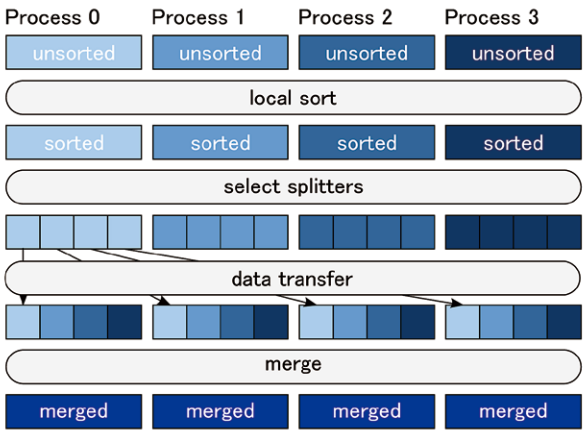


図9 Splitter-based parallel sorting algorithm

4.3 Splitter-based parallel sorting algorithmのGPUを使用した実装

Splitter-based parallel sorting algorithmの一つであるHykSort[8]を拡張し、GPUアクセラレータによる高速化を行った。実装では、最もコストの大きい計算処理である、各プロセス上でのソート処理をGPU上で行うように変更した。GPUメモリはDRAMと比べると非常に小さいため、GPUメモリを溢れるようなサイズのデータをソートする時には、複数個にチャンクを分けてソートを行い、最後にソートされたチャンクをマージすることで対応している。

4.4 性能評価

GPUアクセラレータを用いたhyksortの性能評価はTSUBAME2.5上で最大1024ノードを使用して行った。各ノード上ではソケットあたり1つのプロセスが配置され、各プロセスが2GBの64bit整数のデータを所有している状況で開始する。弱スケーリングの実験では、2048プロセス（1024ノード、2048GPU）の時に0.25TB/s程のスループットが得られた。これはCPU 1スレッドのみの実装と比べると3.61倍、CPU 6スレッド並列のものと比べると1.40倍の性能となっている（図10）。

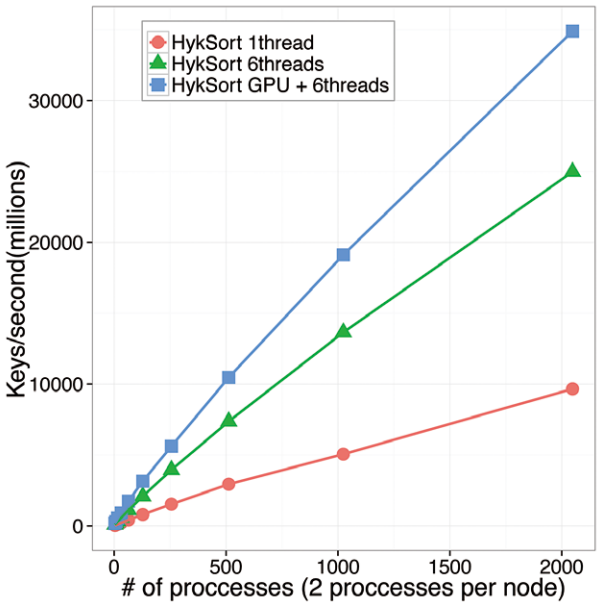


図10 GPUを考慮したhyksortの実行結果

おわりに 5

本稿では、将来のエクストリームスケールなスーパーコンピュータやクラウドデータセンターでのビッグデータ処理にむけたTSUBAME2での研究開発の動向として、Graph500、GPUアクセラレータによるMapReduce処理、大規模分散ソート、などの具体的な事例についての紹介を行った。これらの事例は2016年頃に導入されるTSUBAME3.0などのエクストリームなビッグデータ処理を指向したスーパーコンピュータのアーキテクチャの設計などに活かされる予定である。

謝辞

本研究の一部はJST CREST「ポストペタスケールシステムのおける超大規模グラフ最適化基盤」「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」、JSPS科研費 課題番号「26011503」、「26540050」、及び、TSUBAMEグランドチャレンジ大規模計算制度の支援による。

参考文献

[1] Graph500: http://www.graph500.org/

[2] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, Zoubin Ghahramani, “Kronecker graphs: An approach to modeling networks”, The Journal of Machine Learning Research, Vol11, p985-1042, 2010.

[3] Fabio Checconi, Fabrizio Petrini, Jeremiah Willcock, Andrew Lumsdaine, Anamitra Roy Choudhury, and Yogish Sabharwal, “Breaking the speed and scalability barriers for graph exploration on distributed-memory machines”, Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12), Article No.13, p13:1-13:12, 2013.

[4] Scott Beamer, Krste Asanović, and David Patterson, “Direction-optimizing breadth-first search”, Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12), Article No.12, p12:1-12:10, 2012.

[5] Scott Beamer, Aydin Buluc, Krste Asanović, and David Patterson, “Distributed Memory Breadth-First Search Revisited: Enabling Bottom-Up Search”, Proceedings of IPDPSW'13, p1618-1627, 2013.

[6] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: simplified data processing on large clusters”, Communications of the ACM, Vol. 51, Issue 1, p107-113, 2008.

[7] U. Kang, Charalampos E. Tsourakakis, and Christos Faloutsos, “PEGASUS: A Peta-Scale Graph Mining System Implementation and Observations”, Proceedings of the 9th IEEE International Conference on Data Mining (ICDM '09), p229-238, 2009.

[8] Hari Sundar, Dhairya Malhotra, and George Biros, “HykSort: a new variant of hypercube quicksort on distributed memory architectures”, Proceedings of the 27th international ACM conference on International conference on supercomputing, p293-302, 2013.

OpenACCを用いた全球雲解像モデルNICAM力学コアの大規模GPU計算

八代 尚* 成瀬 彰** 吉田 龍二* 丸山 直也* 富田 浩文*
* 理化学研究所・計算科学研究機構 ** エヌビディア

気象・気候シミュレーションは多くの場合メモリ律速であり、GPUの積極的な利用が期待されている。一方でアプリケーションが大規模かつ複雑であるため、ソースコードの書き換えには多大な時間と労力を必要とする。そこで我々はOpenACCを用いたGPU最適化を進め、GPU専用のソースコードをほとんど増やすことなく、全球雲解像モデルNICAMの流体力学計算部分全体をGPU上で計算することを可能にした。これをもとにTSUBAME 2.5上で大規模計算を行い、OpenACCを用いたディレクティブベースのGPUプログラミングにおいてもGPUのメモリ転送性能を十分に引き出し、良好なウィークスケーリング性能を発揮できることが示された。計算は最大で67億格子（水平3.5kmメッシュ、160層）に対して2560個のGPUを用いて行われ、最大60TFLOPSの演算性能を得た。

1 はじめに

台風の発生過程解明や気候システムの諸要素の研究など学術的な側面だけでなく、天気予報や気候変動予測のような人間社会と密接に関わる課題においても、気象・気候の数値シミュレーションは大きな役割を果たしている。
大気現象には様々な時間・空間スケールが存在し、小さなスケール（数m）から全球スケールの現象までを統合的に扱い、スケール間の相互作用をよりよく表現するには、現在よりもさらに細かい空間解像度でのシミュレーションを実現することが求められている。これはさらに強力なスーパーコンピューターを活用する必要があるということである。

一方、近年のスーパーコンピューターの演算性能はGPGPUやメニーコアコプロセッサのようなアクセラレータに依る部分が多く、その比率は増加の一途を辿っている。このような背景から、気象モデルも積極的にアクセラレータを利用することが求められている。気象モデルは要求するByte/FLOP比が一般的に高く、アーキテクチャの演算性能よりもメモリバンド幅に律速される。この点でも、CPUより高いメモリ転送性能をもつアクセラレータの利用は有効である。

しかし、気象モデルのソースコード規模は非常に大きく（例えば、NICAMは数10万行）、またモデル自体が大気力学、雲微物理学、接地微気象学、陸水学、大気放射学などの様々な分野で研究開発されている小モデルの集合体である。そのためソースコードのメンテナンスや大規模な書き換えには大きな困難を伴う。

近年、ディレクティブベースでのアクセラレータ利用を可能にするプログラミング規格として、OpenACCが登場した。これにより、CUDA C/Fortranなどを用いたソースコードの大幅な書き換えを行うことなく、既存のソースコードにディレクティブを追加することによってCPU-アクセラレータ間のデータ転送とアクセラレータ上での演算実行が実現する。本稿では、既に気象・気候計算の第一線で実際に活用されている大規模化・複雑化した気象モデルに対し、OpenACCを適用することによって、コードのポータビリティや可読性を失うこと無く、GPU上での計算を実現し、その性能評価を行った結果を報告する。

2 全球雲解像モデル NICAM とその力学コア

全球雲解像モデルNICAM（Nonhydrostatic ICosahedral Atmospheric Model）[1][2]は、高並列計算機上で高解像度の全球シミュレーションを行うために開発された気象・気候モデルである。現在は海洋研究開発機構、東京大学大気海洋研究所、理研計算科学研究機構の研究者が中心となって開発を進めている。言語はFortran90で記述されている。NICAMは静力学近似を用いず、完全圧縮流体の方程式系に対して有限体積法を用いて離散化を行っている。水平方向には球面上で準一様な解像度とするため正20面体格子系を用いている。有限体積法のような格子法の利用はこれまで全球モデルで広く採用されてきた球面調和関数を用いるスペクトル法と異なり、計算ノード間の広域通信を必要とせず、隣接通信だけで済むという利点がある。NICAMを含む気候・気象モデルの計算は大まかに2つに分類される。一つは流体力学方程式系の計算部分（力学過程）であり、NICAMでは運動方程式等を有限体積法で解くためのステンシル計算を主に行う部分である。力学過程は要求するByte/Flop比が大きく、また隣接通信が頻繁に行われる。もう一つは物理過程と呼ばれ、水の相変化を扱う雲微物理過程、太陽光や赤外放射による加熱冷却を扱う大気放射過程、現状の格子間隔では解像されないサブグリッドスケールの境界層乱流の効果などを含む。これらの諸過程は一般的に力学過程よりも要求B/F比が低く演算律速である。またほとんどの過程は3次元大気の中で水平方向には依存関係をもたないため、大気を水平方向にプロセス分割する場合は通信を必要としない。

気候・気象アプリケーションをGPU上で高速に動作させるためには、力学過程、物理過程のほとんどすべての部分をGPU最適化させる必要がある。これは演算密度がソースコード内に一様で、高速化するべきホットスポットが明瞭でないためである。このような状

況をふまえ、我々はまず力学過程の完全なGPU化を進めることにした。本研究で用いたNICAM-DCは、NICAMの力学過程を抽出したパッケージであり、BSD-2ライセンスで配布されている（http://scale.acis.riken.jp/nicamdc/）。NICAM-DCは演算カーネル群ではなく通信やファイルI/Oをすべて有しており、全球大気モデルの力学コアに関する各種テストケースを実行し評価することが出来る。

OpenACC の適用 3

OpenACCディレクティブの適用にあたり、基本的な最適化の指針を以下に示す。

- メモリアロケーションと演算部分の分離の徹底：GPU上で計算するために必要なメトリクス項などの配列は、必ずセットアップ部分でメモリ確保を行い、演算部分から排除する。
- 更新されない配列はすべてGPU上に常駐させる：上記の配列は時間発展しないので、セットアップ時にpresent_or_copyin節を用いてあらかじめ転送を行う。
- カーネルの非同期実行：各ループに対しカーネル節を適用する際に、袖通信のタイミングやデータ依存性のある部分を明確にし、極力async節を付加した実行を行う。
- 袖通信の最適化：NICAMの格子配置は正20面体を用いており、隣接通信の相手や格子位置は複雑である。そのため、袖通信の際に送るデータは必ずpack / unpackを行っている。GPU-Host間のデータ転送量を削減するため、このpack/unpackの作業はGPU上で行い、隣接通信に必要なデータのみをHost側に転送し、再受け取りを行う。
- 特異点格子の計算：マスターノードで行われる特異点格子の計算は演算量が少ないためGPUには転送せずCPUで行う。GPUのカーネル実行を非同期化しているため、CPUが行う特異点格子の計算はGPU実行とオーバーラップされ、その実行時間は隠蔽される。またCUDA等を用いた場合、通常の格子と特異点の計算カーネルをサブルーチン単位で分離することになり、メンテナンスが困難になることが予想されるがOpenACCの場合は同じソースコードに共存することが出来る。
- 計算結果出力の扱い：気象モデルでは風速や気温、診断値のような解析に用いる2次元・3次元変数の時系列を取得するため、計算途中の値をファイルに出力することが一般的である。この結果出力のための鉛直内挿、時間方向の平均化や診断値の計算も、GPU・Host間のデータ転送量を削減するためGPU上で行い、数十〜数百ステップごとのファイル出力のタイミングでHost側に転送し、書き出しを行う。

NICAMのデータ構造は（ij、k、l）=（水平格子、鉛直格子、領域分割単位）であり、多くの場合で水平格子が最もサイズが大きい。GPU上での実行に際し、このデータ構造を変更する必要はなかった。ただし、いくつかのステンシル演算カーネルではスカラー機用のキャッシュラインチューニングとして、小さいサイズの配列次元を最内に変更した箇所があったため、OpenACC適用時に再度変更された。これは、アーキテクチャによってデータ順序の異なるAoS（Array of Structure）版とSoA（Structure of Array）版の2バージョンを使い分ける必要があることを示唆している。しかし、切り替えるコードの分量としてはそれほど大きくないため、メンテナンス性の観点から見ても十分に対応可能であると判断される。以上のことをふまえ、挿入されたディレクティブ行および元コードに変更が施された行は2000行程度であった。これはNICAM-DCの総コード行数58000行の5%に満たない量であり、十分なポータビリティとメンテナンス性を保持したまま、GPUへの対応が完了したといえる。

TSUBAME2.5 を用いた実行性能測定 4

4.1 演算性能

TSUBAME2.5を用いて、CPUのみでの実行とGPUを利用した実行を行い、演算性能の評価を行った。CPUとGPUの性能比較の方法についてはいくつか方法があるが、NICAMはメモリ律速のアプリケーションであるため、図1に示すノード単位での比較を行った。

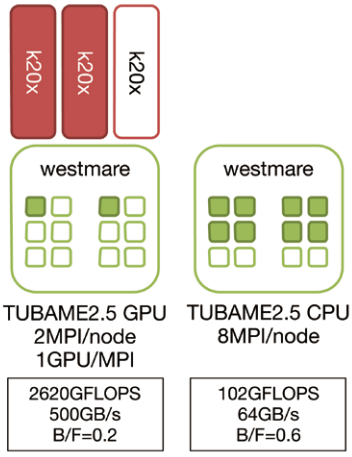


図1 TSUBAME2.5でNICAM-DCを実行した際の1ノードあたりの構成。それぞれGPU使用時（左）、GPU非使用時（右）を示す。

このとき、CPUのみの実行ではCPU向きのAoS版のカーネルを用い、MPIプロセス分割数を4倍に増やして行った。GPUを用いた実行ではTSUBAME2.5のノードに3枚搭載されているGPUカードのうち、

2枚を用いている。ノードあたりのプロセス数、GPU利用数が限られているのは、NICAMのMPI分割数に制限があり、$10x4^n$の公約数しかとれないためである。性能計測にあたり、演算量およびメモリ転送量をあらかじめ取得しておき、実行時にはタイマー計測で実行時間を取得した。電力消費量については、TSUBAME2.5がサービスとして提供するノード単位の電力消費情報のためのツールを利用した。コントロール実験として演算量とメモリ転送量の計測に用いた際の問題サイズは水平解像度56km、鉛直160層（格子数2600万）であった。実験には全球大気モデルの力学コアテストケースの一つであるJablonowski and Williamson (2006) [3]の傾圧不安定波実験（ただし60ステップ）を用い、結果取得のためのファイル出力も行った。セットアップを除くメインループの1ノード、1ステップあたり総演算量は420GFLOP/stepであり、メモリ転送量は2.3TB/stepであった。結果を図2に示す。

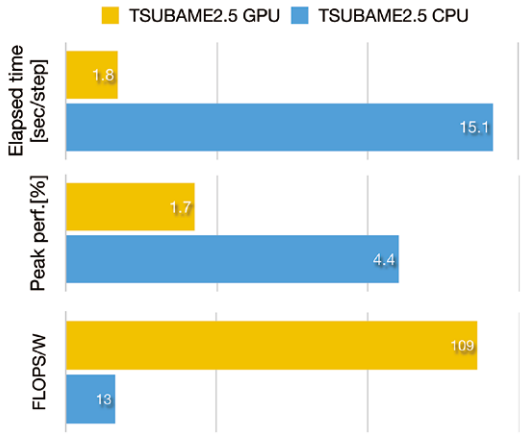


図2 TSUBAME2.5でNICAM-DCを実行した際の実行性能。それぞれメインループ1ステップあたりの所要時間（上）、演算ピーク性能比（中）、ワットあたりFLOPS（下）を示す。

GPUは5ノード10MPIプロセス、CPUは5ノード40MPIプロセスで計測を行い、ノードあたりの結果として示してある。図より、OpenACCを通してGPUを利用した際の実行時間はCPUのみを利用した時よりも、およそ8倍高速に実行された。これは図1に示した構成での、メモリ転送性能の差とおおむね一致している。CPU、GPU実行においてどちらもピークメモリ転送性能の約50%を利用できており、ディレクティブによるプログラミングスタイルをとるOpenACCが十分に性能の高い実行コードを生成出来ていることを示している。一方、演算ピーク性能比という観点では、よりByte/FLOP比の小さいGPUでピーク性能比が悪い結果となった。NICAM-DCの主要部分である力学コアはNICAMの中でも要求B/F比が高い部分であり、強くメモリ転送性能に律速される。GPUの演算性能はまだまだ活用出来る余地が残っており、シミュレーション結果の物理的な性能や演算量を評価しながら、力学コアのスキーム・アルゴリズムの変更や実数演算精度の見直しを行うことが今後必要であると考える。消費電力あたりの演算性能では、実行時間とおおむね同程度である、約8倍の効率向上が確認された。

4.2 スケーリング性能

図3にウィークスケーリング性能の計測結果を示す。ウィークスケーリングではモデルの平解像度を56kmから3.5kmまで変化させることで、ノードあたりの問題サイズを固定して5ノード10MPIプロセスから1280ノード2560MPIプロセスまで増加させた。ただし、CPUのみの実験では1280ノード10240MPIプロセスでの実行は行っていない。図よりわかるように、CPU、GPU実験共に良好なスケーリング性能が得られている。1ステップあたりの実行時間でみても、5ノード実験に対し1280ノード実験で30%程度所要時間が伸びるのに留まっている。このとき、得られた演算性能は46.5TFLOPSであった。所要時間が増加する主な要因は、CPU実行の場合は主にネットワーク通信時間の増加であるのに対し、GPU実行の場合は計算結果出力であった。CPUの場合はフラットMPIで実行しているために、ネットワーク通信の輻輳がMPIプロセス数の少ないGPU実行時より起こりやすいと推察される。一方、GPUではファイル出力を行う変数の3次元配列全体をGPUからホストに転送する必要があり、袖通信のためのデータ転送と比べても転送量が圧倒的に多く、律速となっていることがわかった。時系列データは多くの場合倍精度である必要がないので、今後はGPU上での単精度化や、GPU上でのデータ圧縮を適用出来るよう改良する予定である。このテストケースではファイル出力の間隔をシミュレーション時間で15分に1回としているが、現実的な実験ではより長い間隔で出力されることが多い。ファイル出力間隔を12時間とした場合は、演算性能は1280ノードの場合で60TFLOPSまで向上した。

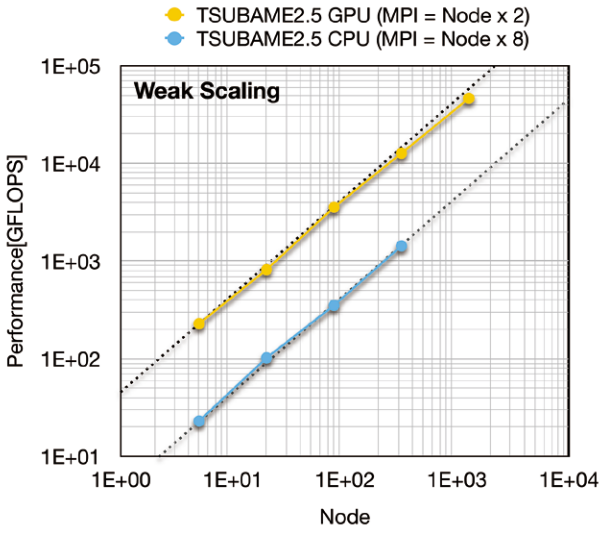


図3 TSUBAME2.5でNICAM-DCを実行した際のウィークスケーリング性能。

図4にストロングスケーリング性能の評価結果を示す。ストロングスケーリングでは水平解像度56kmのまま、5ノードから1280ノードまで増加させた。このときプロセスあたりの水平格子数は16900gridから100gridまで減少している。図より、CPUのみの実行と比較して、GPU実行の場合はノード数の増加に伴って急速にスケーリング性能が低下していることがわかる。これはGPU-Host間のデータ転送時間の割合が増大することに加えて、水平格子数の減少に伴い並列性が減少し、GPUが十分な演算効率を発揮出来なくなったためである。特にデータ転送に関しては、1回の袖通信のために行う一連の動作に含まれる種々のレイテンシが無視出来ないため、通信量よりも通信回数を減らすことが効果的であることがわかった。また、課題実施時点のPGIコンパイラとGPUドライバのバージョンではサポートされていなかったpinned memory機能の活用も、GPU-Host間の転送時間削減に有効であると考えられる。ストロングスケール性能は十年～百年（10^6-10^7ステップ）にわたる気候実験を行う際に重要であり、今後重点的に改良を進める必要があると考える。

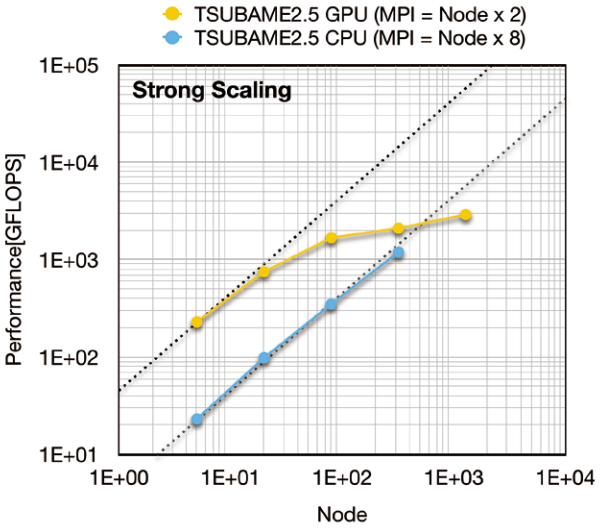


図4 TSUBAME2.5でNICAM-DCを実行した際のストロングスケーリング性能。

まとめと今後の課題 5

OpenACCを用いて、全球高解像大気モデルの力学コア全体をGPU上で計算し、メモリ転送性能に見合う実行性能を得ることに成功した。ソースコードの変更量はモデル全体の行数と比較して5%以下であり、高いポータビリティを同時に達成した。スケーリングについては、ウィークスケーリングで良好な結果を得た一方、ストロングスケーリングではGPU-Host間の転送がより律速することがわかった。本課題の結果は、OpenACCを用いることで、ソースコード規模が大きく複雑なアプリケーションコードを容易にアクセラレータ上で動作させられることを実証するものである。今後はOpenACCの適用をさらに物理諸過程の各コンポーネントに拡張し、フルパッケージのGPU対応を進める。力学コアの更なる高速化に向けて、時間発展スキームの変更や計算結果の精度を考慮した単精度・倍精度計算の混合を検討する必要が考えられる。ストロングスケーリング性能の向上については、通信回数の削減を目指したアルゴリズムの見直しを行うことが重要である。

謝辞

本研究はTSUBAMEグランドチャレンジ大規模計算制度を利用して実施させて頂き、一部は多国間国際研究協力事業「エクサスケール地球システムシミュレーションのための20面体モデル」から支援を頂いた。記して謝意を表す。

参考文献

[1] Satoh, M., T. Matsuno, H. Tomita, H. Miura, T. Nasuno and S. Iga (2008) : Nonhydrostatic Icosahedral Atmospheric Model (NICAM) for global cloud resolving simulations. J. Comp. Phys., the special issue on Predicting Weather, Climate and Extreme events, 227, 3486-3514.

[2] Tomita, H. and Satoh, M. (2004) : A new dynamical framework of nonhydrostatic global model using the icosahedral grid. Fluid Dyn. Res., 34, 357-400.

[3] Jablonowski C, Williamson DL. A baroclinic instabilitiy test case for atmospheric model dynamical cores. Quart. J. Roy. Meteor. Soc. 2006; 132(621C):2943–2975.

気象計算のための GPUコンピューティング・フレームワーク

下川辺 隆史　青木 尊之　小野寺 直幸
東京工業大学・学術国際情報センター

気象計算はスパコンで実行される重要なアプリケーションの一つであり、GPUスパコンで高速化されてきている。GPUスパコンで効率的に実行するためには、複雑な最適化手法を導入する必要がある。これらをアプリケーションに簡単に導入することが可能なGPUコンピューティング・フレームワークを開発し、これを用いて次期気象予報のために気象庁で開発されている非静力気象モデルASUCAをGPUスパコンへ実装した。フレームワークによって、実装の複雑な通信を隠蔽するオーバーラップ手法等を簡便にASUCAへ導入することが可能となり、高い生産性・可搬性を実現しながら、高い実行性能を達成することに成功した。

はじめに 1

ここ数年、高い浮動小数点演算能力と高い メモリバンド幅および電力効率のよい Graphics Processing Units（GPU）が汎用計算に多く用いられるようになってきた[1,2,3]。数値予報の分野においても、GPUによる高速化は大きな注目を集めている。気象庁で開発が進められている次世代高分解能局地モデルASUCAでは、アプリケーションの全計算をGPU上で行う方法でGPU化し、東京工業大学のGPUスパコンTSUBAMEで高性能計算が行われた[1,2,5]。また、メソスケール気象モデルWRF（Weather Research and Forecasting）においてもGPUにより高速化されたことが報告されている[6,7]。

このようにGPU計算では高い性能が得られることが期待されるものの、GPU用プログラミングは、NVIDIA社製GPUに特化したCUDAや複数のマルチコアプロセッサに対応した言語 OpenCL などを用いる必要があり、さらにプロセッサの性能を引き出すためには個々のアーキテクチャを意識したプログラミングを行い、機種固有の最適化手法を導入する必要がある。このような問題を解決するために、高い抽象度により生産性を向上させ、可搬性を備えたいくつかのプログラミングモデルが提案されている[8]。

本研究では、直交格子上で実行される気象計算ASUCAを高い抽象度で、高生産にGPUスパコン上に実装するためのGPUコンピューティング・フレームワークを開発する。このフレームワークは、C/C++言語およびCUDAを用い実装され、大規模な気象計算で必要なステンシル計算を簡便に表現するクラスや GPU間通信やノード間通信を簡単に行うクラスを提供する。これにより、高い生産性を実現し、性能を出すためには制約の多いGPUアーキテクチャやGPU間通信の実装を意識することなく、GPUスパコン向けの最適化を施すことが可能である。ユーザは、ステンシル計算のみを記述するため、可搬性があり、本フレームワークを用いることでGPU以外のアーキテクチャでもユーザコードの変更無く実行することができるようになる。

本稿では、本フレームワークの概要と、これを用いた気象計算ASUCAのTSUBAMEを用いた大規模計算の実行性能について紹介する。なお、本研究の詳細は参考文献[4,5]を参照されたい。

気象計算 ASUCA とその GPU による高速化 2

ASUCA（Asuca is a System based on a Unified Concept for Atmosphere）は気象庁か次期の気象予報のための現業コードとして開発を進めている次世代高分解能局地モデルである。ASUCAは一般座標系を採用し、力学過程の方程式系はフラックス形式の完全圧縮・非静力学方程式系である。ASUCAでは鉛直方向の音波関連項のみを陰的に扱うHEVI（Horizontally explicit-Vertically implicit）法を採用している。風速などに比べて伝播速度の速い音波、重力波に関係する項はサブステップを用いて小さい時間刻みで計算し、移流計算や物理過程など、その他の項は大きい時間刻みで計算する。小さい時間刻み、大きい時間刻みともに3段階Runge-Kutta法を用いている。現在では、気象庁で運用中の気象モデルJMA-NHMと同等の雲物理過程が導入されている。

これまでの研究で、気象計算ASUCAのGPU化に取り組み、その全ての計算がGPUへ移植された。全ての予報変数をGPUのデバイスメモリ上に確保し、時間発展ループの内側の全ての計算をGPUで実行する。気象庁において開発されてきたASUCAはFortranで書かれているが、GPUへ実装するにあたり予報変数の配列の次元を変更するために、その検証目的も含めてC/C++言語に書き換え、さらにCUDAでGPUへ実装した。2011年に、GPUスパコンTSUBAMEの3990GPUを利用して、14,368 × 14,284 × 48計算格子に対して行った計算では、145 TFlopsという極めて高い実行性能を達成した[1,2]。

GPU コンピューティング・フレームワーク 3

気象計算ASUCAのGPU化では、GPU上で性能を出すために配列の次元を変更し、GPUアーキテクチャに向けた最適化手法を導入した。また、大規模計算では、通信コストを隠蔽する通信と計算のオーバーラップ手法などの計算手法を導入した。これまでの研究で気象計算はGPUで高速化できることは示されたが、その開発コストは高い。そこで、本研究では、これらの最適化手法を簡便に導入し、高い生産性を実現するフレームワークを開発し、これを用いて気象計算ASUCAを実装する。本フレームワークは、直交格子型の気象計算を対象とし、各格子点上で定義される物理変数（プログラム上は配列となる）の時間変化を計算する。また、当該物理変数の時間ステップ更新は陽的であり、ステンシル計算によって行われる。TSUBAMEに導入されているNVIDIA社製GPUで実行することを目指し、実装には、ホストコードはC/C++言語、デバイスコードはCUDAを用いる。

フレームワーク設計における主な目標を述べる。

- プログラマは格子点上での計算についてのみ記述する。格子全体の処理はフレームワークが行う。格子全体の処理がユーザコードからフレームワークへ分離されGPUに合った最適化手法を隠蔽する。また、バックエンドとして様々なプロセッサを採用することができ、拡張性と高い生産性を持つ。現在、フレームワークでは、GPUおよびCPUで実行可能である。
- フレームワークが提供するテンプレートを用いたC++クラスを用い格子点上の計算を記述する。言語拡張や標準的でないプログラミングモデルを利用すると、既存コードからの乖離も大きく利用しにくい。また、拡張部分がフレームワークを他の環境へ移植する妨げになりうる。その点、C++のテンプレートやクラスは広く使われており、基盤とできる。
- 複数ノードでのGPU計算に対応するため、ノード間およびノード内通信を簡便に記述するクラスを提供する。このクラスを用いることで、プログラマはMPIやOpenMPを明示的に使用した通信を記述すること必要がなくなる。

以上をまとめると、ユーザは、(1)フレームワークの提供するテンプレート関数およびクラスを用い、ステンシル計算を行う関数を記述する。(2)フレームワークの提供するクラスを用いて、GPU間通信を記述する。以下で詳細について記述する。

3.1 ステンシル計算関数の定義

本フレームワークでは、ステンシル計算は、フレームワークの提供するArrayIndex3D等を用い、ファンクタ（関数オブジェクト）として定義し、ステンシル計算関数と呼ぶ。 3次元の拡散計算では、次のように関数を定義できる。

```
struct Diffusion3d {
__host__ __device__
void operator()(const ArrayIndex3D &idx,
       float ce, float cw, float cn, float cs,
       float ct, float cb, float cc,
       const float *f, float *fn) {
fn[idx.ix()] =
  cc*f[idx.ix()]
+ ce*f[idx.ix<1,0,0>()] + cw*f[idx.ix<-1,0,0>()]
+ cn*f[idx.ix<0,1,0>()] + cs*f[idx.ix<0,-1,0>()]
+ ct*f[idx.ix<0,0,1>()] + cb*f[idx.ix<0,0,-1>()];
}
};
```

ArrayIndex3Dは、対象とする配列のサイズ（n_x, n_y, n_z）を保持し、ある特定の格子点を表すインデックス（i,j,k）を設定できる。対象とする配列がfであるとき、idx を ArrayIndex3D のオブジェクトとすると、f[idx.ix()]として使われ、これは配列fの（i,j,k）点の値を返す。ArrayIndex3Dはテンプレートを用いたメンバ関数が定義されており、例えば、idx.ix<+1,0,0>()、idx.ix<−1,−2,0>()とすると、（i+1,j,k）、（i-1,j-2,k）を表すインデックスを返す。テンプレートを用いることで、GPUおよびCPUでのインデックス計算の高速化を図っている。

ステンシル計算関数の第一引数は固定で、計算対象となる格子のインデックス情報を持つidxを受け取らなければならない。関数実行時には、格子点(i, j, k)の値が設定されているため、(i, j, k)を中心としたステンシル計算を関数内に記述する。f、fnは配列へのポインタであり、これに対しステンシルアクセスすることとなる。

拡散係数が空間の関数になっているなど、解析する問題によってはf、fn以外の係数を保持する変数が必要となる。ステンシル計算関数内では、ある格子点を更新するための記述しか表現できないため、空間の関数になっている係数に対してはf、fnと同様に配列として確保し、f、fnと同じようにステンシル計算関数の外から渡す必要がある。

3.2 ステンシル計算関数の実行

本フレームワークは、全ての格子点に計算を行うLoop3D等のクラスを提供する。これを用い、ステンシル計算関数の実行は以下のように行う。

```
Loop3D loop3d(nx+2*mgnx, mgnx, mgnx,
              ny+2*mgny, mgny, mgny,
              nz+2*mgnz, mgnz, mgnz);
loop3d.run(Diffusion3d(), ce, cw, cn, cs,
           ct, cb, cc, f, fn);
```

Loop3Dはステンシル関数を適用する範囲を指定するパラメータで初期化する。Loop3D::run()は任意個の異なる型を引数にとるテンプレート関数として定義されている。C++のテンプレート関数の型推論を利用し、Loop3D::run()は、与えられた全ての引数を第二引数以降に持つファンクタDiffusion3d()を呼び出す。ファンクタは、NVIDIA CUDAの__host__、__device__で定義

することができ、ホスト、デバイス両方へコンパイルされており、`Loop3D::run()`がCPU上のデータに対しても、GPU上のデータに対しても同じ関数を実行する。`Loop3D`内部の実装としては、CPU上で実行する場合はファンクタを全格子点に対して`for`文で実行し、GPU上で実行する場合は内部で生成されるCUDAのグローバル関数に包み、適当なCUDAのblock数、thread数が渡され全格子点に対して実行される。第二引数以降で初めて出てくるポインタがGPUメモリの配列を指すかCPUメモリの配列を指すかを判定し、GPUで実行するかCPUで実行するかを自動的に決定する。

3.3 変数のGPU間通信

複数GPUによる格子を用いた計算では、図1に示す領域分割法を用いる。領域分割法では、隣接領域間の境界領域の交換が必要であり、配列として確保された変数は隣接GPUから送信されたデータを格納する境界領域を持つ。本フレームワークでは、この隣接領域間の境界領域を交換するためのGPU間通信を簡単に記述するためのクラス`BoundaryExchange`を提供する。複数GPUにおける計算を効率的に行うため、フレームワークは、ノード内並列ではOpenMPで並列化し、GPU間の通信を効率的に行うため図2のようにGPUDirect によるpeer-to-peer 通信を用いる。一方、ノード間並列ではMPIを利用する。図3のようにMPIによるノード間通信では、転送のための一時領域をホストメモリに確保している。

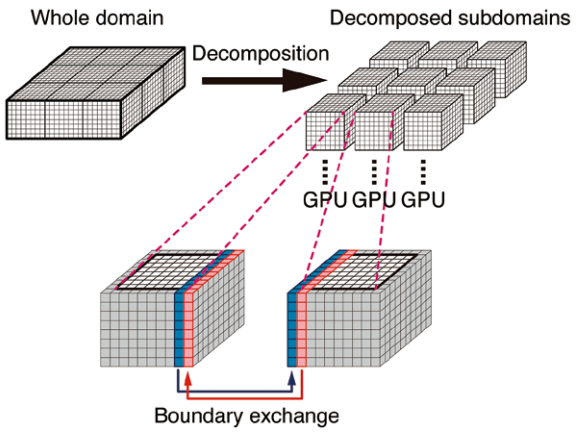


図1 複数GPUによる格子に基づいた計算

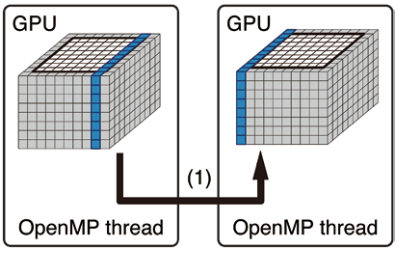


図2 OpenMPスレッドによるノード内GPU間通信。この通信には、(1) GPUDirect によるpeer-to-peer 通信が用いられる。

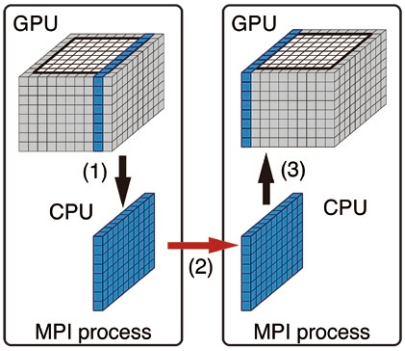


図3 MPI によるノード間GPU間通信。この通信は、(1) GPUからホストへのデータ転送、(2) MPI によるデータ交換、(3) ホストからGPUへのデータ転送で行われる。

クラス`BoundaryExchange`を用いて、以下のようにGPU間通信を行う。

```
BoundaryExchange *exchange = domain.exchange();
exchange->append(array1);
exchange->append(array2);
exchange->append(array3);
exchange->transfer();
```

`domain`はクラス`Domain`のオブジェクトで、計算領域サイズ、隣接ノード番号等を保持している。クラス`BoundaryExchange`は、`Domain`を参照することで計算領域サイズ等を取得している。これらの情報を用いることで、`exchange->append()`で登録された変数`array1`、`array2`、`array3`に関して`exchange->transfer()`で境界領域を転送する。この関数内で、OpenMP、MPIを用いた転送が行われる。

3.4 通信と計算のオーバーラップ手法

大規模計算では、GPU間の通信時間は全実行時間に対して無視できない。通信コストを計算で隠蔽するオーバーラップ手法の導入を大規模計算における性能向上にとって重要となる。

本フレームワークは、カーネル分割によるオーバーラップ手法を提供する[1,2]。この手法は、ある物理変数内でのデータの非依存性を利用する。ある変数のそれぞれの要素は他の要素の計算と独立に計算できるため、1つのGPUが担当する計算領域を境界領域と残りの中心領域に分割して、独立に計算することが可能である。図4にオーバーラップ手法の計算方法を示す。オーバーラップ手法では、中心領域の計算を開始し、それと同時に境界領域の計算に必要なデータを取得するための通信を行う。通信が終了した後、境界領域の計算を行う。これによって、通信の隠蔽を行う。

本フレームワークでは、ユーザコードにカーネル分割によるオーバーラップ手法を適用するために、クラスCompCommBinderを提供する。これを用い、拡散計算では、次のようにオーバーラップ手法を記述できる。

```
BoundaryExchange *exchange = domain.exchange();
exchange->append(f);
CompCommBinder<Loop3D> ccbinder(exchange);
ccbinder.set_post_func(&loop,
    create_funcholder<Loop3D>(Diffusion3d(),
        ce, cw, cn, cs ,ct, cb, cc, f, fn));
ccbinder.set_use_overlapping();
ccbinder.run();
```

CompCommBinderは、Loop3Dによる計算範囲、ステンシル関数とGPU間通信を制御するBoundaryExchangeオブジェクトによって初期化される。CompCommBinderは、Loop3Dを境界領域と中心領域に対応する複数のLoop3Dへ分割し、これらのLoop3Dでステンシル関数を実行する。BoundaryExchangeによる通信を適当なタイミングで行い、この通信を隠蔽する。

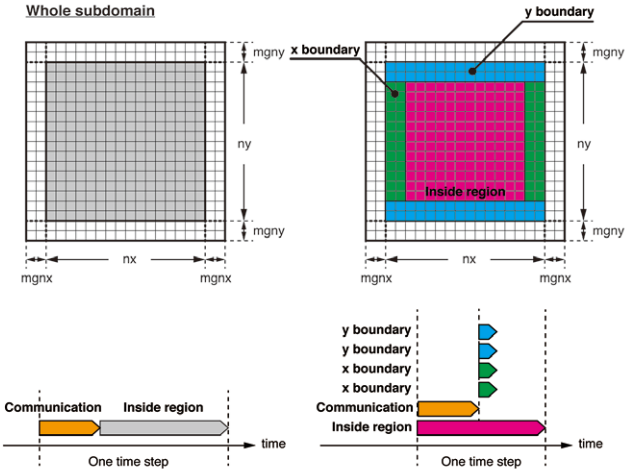


図4 カーネル分割による計算と通信のオーバーラップ手法

フレームワークを用いた気象計算 ASUCA の性能評価 4

本研究では、気象計算ASUCAを本フレームワークを用いて実装した。東京工業大学のTSUBAME2.5に搭載されたNVIDIA Tesla K20X GPUを複数用い、性能測定を行う。

図5に現在の数値予報で使用されている初期値データと境界値データを用いた台風の気象計算を行った例を示す。TSUBAME2.5の672 GPU を用い計算格子 5,376×4,800×57（実際の格子間隔は水平500m）で計算している。

図6に強スケーリングの結果を示す。オーバーラップ手法を用いた場合、用いない場合の性能を示す。参考として1つのGPUを1MPIプロセスで計算するFlat-MPIの結果も合わせて示す。計算格子1536×1280×60および計算格子3072×2560×60を単精度計算した。TSUBAMEの各ノードには3GPU搭載されているが、この計算ではこのうちGPUDirectで通信可能な2GPUを用いている。オーバーラップ手法を用いたことによる性能向上がみられ、計算格子3072×2560×60の512GPU計算では、18.9 TFlopsを達成した。

図7に弱スケーリングの結果を示す。1GPUあたり計算格子768×128×60を用い単精度計算した。強スケーリングの測定と同様、オーバーラップ手法と用いた場合と用いない場合の測定結果を示す。強スケーリングの測定とは異なり、最大の実行性能を得るため、各ノードに搭載された3GPU全てを使用した。オーバーラップ手法を用いることで性能向上がみられ、4,108GPUで209.6 TFlopsを達成した。

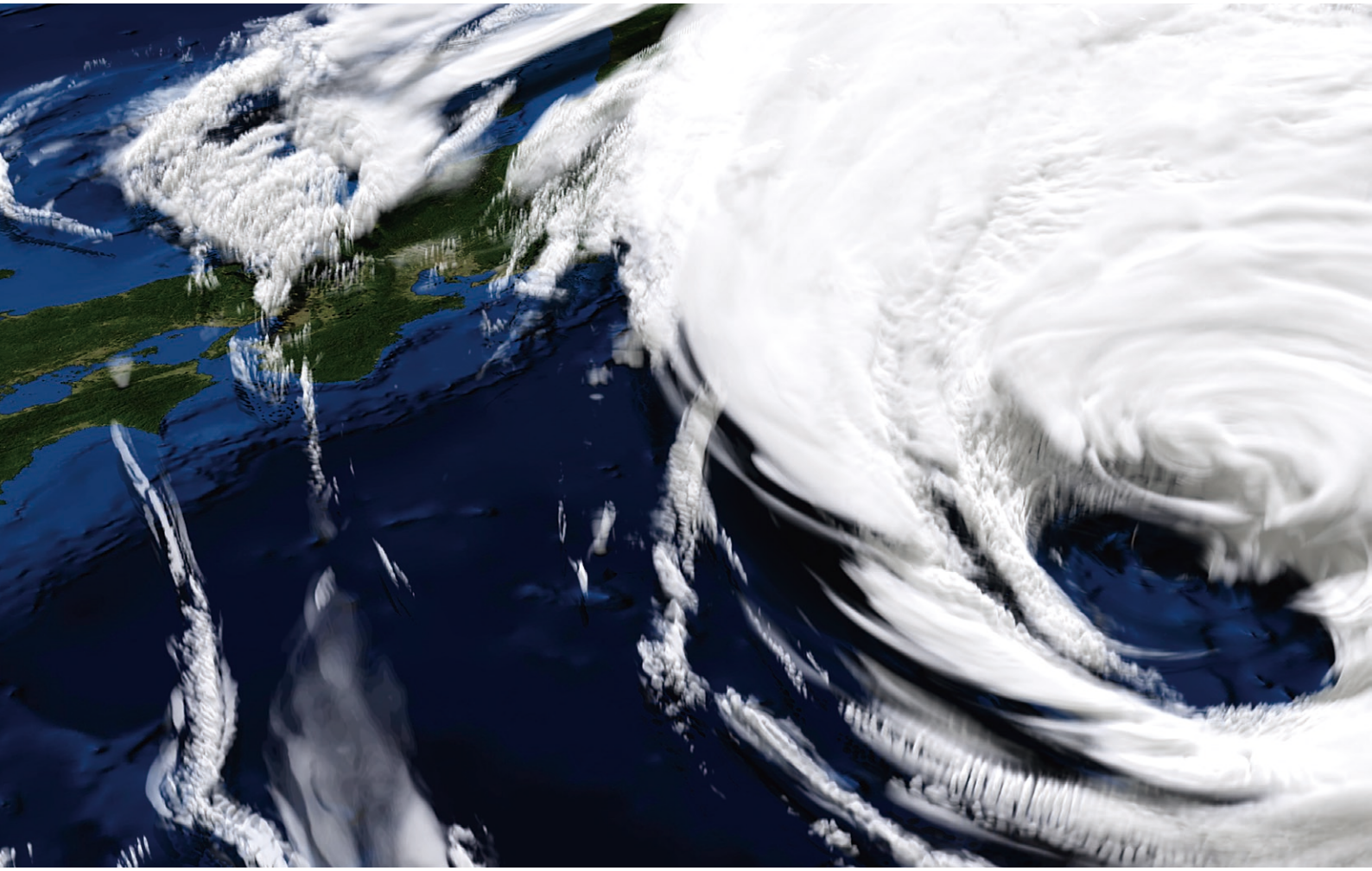


図5 TSUBAME2.5の672GPUを用いた台風の気象計算の例。
計算格子5,376×4,800×57を用い水平解像度500mで計算している。

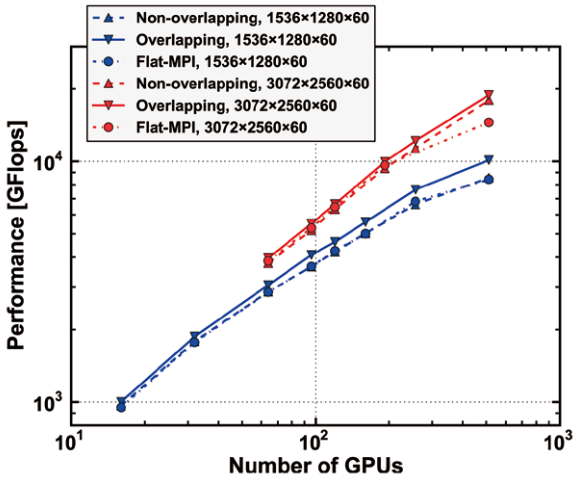


図6 ASUCAの実行性能（強スケーリング）

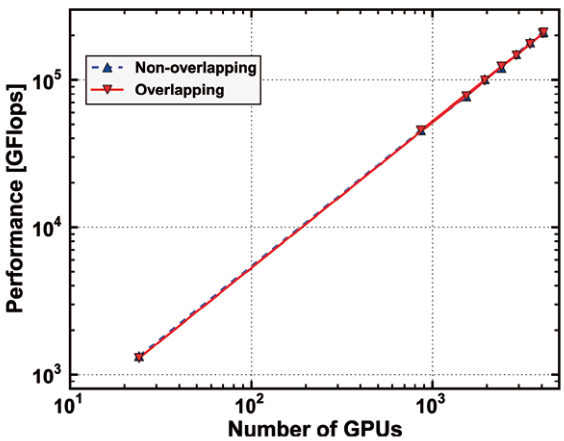


図7 ASUCAの実行性能（弱スケーリング）

まとめ 5

気象計算コードのためのGPUコンピューティング・フレームワークを開発した。フレームワークは、ステンシル計算を簡便に記述するクラスとそれをGPUおよびCPUで実行するクラスを提供する。これにより可搬性があり、高い生産性を実現している。また、フレームワークは、大規模GPU計算で必須となる隣接GPU間の通信を簡便に記述するクラスを提供する。これを用いることで、ノード内、ノード間の通信を簡潔に記述できる。大規模計算では、通信を計算で隠蔽するオーバーラップ手法が重要である。フレームワークはこの手法を簡単に導入できるクラスを提供している。フレームワークで実装した気象計算コードASUCAは、強スケーリング、弱スケーリングともにオーバーラップ手法導入による性能向上が見られた。

謝　辞

ASUCAのオリジナルコードを提供していただきGPU版の開発に協力していただいた気象庁 室井ちあし氏、石田純一氏、河野耕平氏、原旅人氏に深く感謝する。

本研究の一部は科学研究費補助金・若手研究（B）課題番号25870223「低消費エネルギー型GPUベース次世代気象計算コードの開発」、科学研究費補助金・基盤研究（B）課題番号23360046「GPU スパコンによる気液二相流と物体の相互作用の超大規模シミュレーション」、科学研究費補助金・基盤研究（S）課題番号26220002「ものづくりHPCアプリケーションのエクサスケールへの進化」、科学技術振興機構CREST「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」、学際大規模情報基盤共同利用・共同研究拠点、および、革新的ハイパフォーマンス・コンピューティング・インフラから支援を頂いた。 記して謝意を表す。

参考文献

[1] Takashi Shimokawabe, Takayuki Aoki, Chiashi Muroi Junichi Ishida, Kohei Kawano, Toshio Endo, Akira Nukada, Naoya Maruyama, and Satoshi Matsuoka, "An 80-Fold Speedup, 15.0 TFlops, Full GPU Acceleration of Non-Hydrostatic Weather Model ASUCA Production Code," in Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC'10, New Orleans, LA, USA, Nov 2010.

[2] Takashi Shimokawabe, Takayuki Aoki, Junichi Ishida, Kohei Kawano, and Chiashi Muroi, "145 TFlops Performance on 3990 GPUs of TSUBAME 2.0 Supercomputer for an Operational Weather Prediction," Procedia Computer Science, Volume 4, Proceedings of the International Conference on Computational Science, ICCS 2011, 2011, Pages 1535-1544.

[3] T. Shimokawabe, T. Aoki, T. Takaki, A. Yamanaka, A. Nukada, T. Endo, N., Maruyama, S. Matsuoka: Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer, in Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC'11, IEEE Computer Society, Seattle, WA, USA, Nov. 2011.

[4] 下川辺隆史, 青木尊之, 小野寺直幸, "複数GPUによる格子に基づいたシミュレーションのためのマルチGPU コンピューティング・フレームワーク" 2014年ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2014 , 東京, 2014年1月 . (最優秀論文賞、IEEE Computer Society Japan Chapter 優秀若手研究賞)

[5] Takashi Shimokawabe, Takayuki Aoki and Naoyuki Onodera "High-productivity Framework on GPU-rich Supercomputers for Operational Weather Prediction Code ASUCA," in Proceedings of the 2014 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC'14, New Orleans, LA, USA, Nov 2014. (to appear)

[6] J. C. Linford, J. Michalakes, M. Vachharajani, and A. Sandu, "Multi-core acceleration of chemical kinetics for simulation and prediction," in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC'09.New York, NY, USA: ACM, 2009, pp. 1–11.

[7] P. Johnsen, M. Straka, M. Shapiro, A. Norton, and T. Galarneau, "Petascale WRF simulation of hurricane sandy deployment of NCSA's Cray XE6 Blue Waters," in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC'13. New York, NY, USA: ACM, 2013, pp. 63:1–63:7.

[8] N. Maruyama, T. Nomura, K. Sato, and S. Matsuoka, "Physis: an implicitly parallel programming model for stencil computations on large-scale GPU-accelerated supercomputers," in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC'11. New York, NY, USA: ACM, 2011, pp. 11:1–11:12.

● TSUBAME e-Science Journal vol.12
2014 年 9 月 22 日　東京工業大学 学術国際情報センター発行 ©
ISSN 2185-6028
デザイン・レイアウト：キックアンドパンチ
編集：　TSUBAME e-Science Journal 編集室
　　青木尊之　渡邊寿雄　佐々木淳　仲川愛理
住所：　〒 152-8550 東京都目黒区大岡山 2-12-1-E2-6
電話：　03-5734-2085　FAX：03-5734-3198
E-mail：tsubame_j@sim.gsic.titech.ac.jp
URL：　http://www.gsic.titech.ac.jp/

TSUBAME

vol. 12

TSUBAME 共同利用サービス

**『みんなのスパコン』TSUBAME共同利用サービスは、
ピーク性能 5.7PFlops、18000CPUコア、4300GPU搭載
世界トップクラスの東工大のスパコンTSUBAME2.5を
東工大以外の皆さまにご利用いただくための枠組みです。**

課題公募する利用区分とカテゴリ

共同利用サービスには、「学術利用」、「産業利用」、「社会貢献利用」の3つの利用区分があり、さらに「成果公開」と「成果非公開」のカテゴリがあります。
ご利用をご検討の際には、下記までお問い合わせください。

TSUBAME 共同利用とは…

他大学や公的研究機関の研究者の**学術利用**［有償利用］

民間企業の方の**産業利用**［有償・無償利用］

その他の組織による社会的貢献のための**社会貢献利用**［有償利用］

共同利用にて提供する計算資源

共同利用サービスの利用区分・カテゴリ別の利用課金表を下記に示します。TSUBAMEにおける計算機資源の割振りは口数を単位としており、1口は標準1ノード（12 CPUコア, 3GPU, 55.82GBメモリ搭載）の3000時間分（≒約4ヵ月）相当の計算機資源です。
1000 CPUコアを1.5日利用する使い方や、100 GPUを3.75日利用する使い方も可能です。

利用区分	利用者	制度や利用規定等	カテゴリ	利用課金（税抜）※
学術利用	他大学または研究機関等	共同利用の利用規定に基づく	成果公開	1口：120,000円
産業利用	民間企業を中心としたグループ	「先端研究基盤共用・プラットフォーム形成事業」に基づく	成果公開	トライアルユース（無償利用）
				1口：120,000円
			成果非公開	1口：480,000円
社会貢献利用	非営利団体、公共団体等	共同利用の利用規定に基づく	成果公開	1口：120,000円
			成果非公開	1口：480,000円

※　平成 26 年度の利用課金です。最新の利用課金については、下記 URL をご参照ください。
http://www.gsic.titech.ac.jp/kyodou/kakin

産業利用トライアルユース制度（先端研究基盤共用・プラットフォーム形成事業）

東工大のスパコンTSUBAMEを、より多くの企業の皆さまにご利用いただくため、初めてTSUBAMEをご利用いただく際に、無償にてご試用いただける制度です。

（文部科学省 先端研究基盤共用・プラットフォーム形成事業による助成）

詳しくは、下記までお問い合わせください。

お問い合わせ

●東京工業大学　学術国際情報センター　共同利用推進室
●e-mail　kyoyo@gsic.titech.ac.jp　Tel. 03-5734-2085　Fax. 03-5734-3198
詳しくは http://www.gsic.titech.ac.jp/tsubame/をご覧ください。

GSIC
Global Scientific Information and Computing Center