[研究成果]

大規模計算システムにおけるBCMの性能評価

†‡小松 一彦　†¶曽我 隆　†‡江川 隆輔　§‡滝沢 寛之　†小林 広明
†東北大学サイバーサイエンスセンター
§東北大学情報科学研究科
¶NEC システムテクノロジー株式会社
‡JST CREST

*BCM(Building Cube Method)*は，大規模計算システムでの流体シミュレーションを念頭に設計された次世代CFD(Computational Fluid Dynamics)ソルバーである．BCMでは等間隔直交格子を採用しており，流体シミュレーションに必要な演算を容易に均等分割することができるため，大規模計算システムを用いた並列計算に適している．本報告では，様々な大規模計算システムの特徴を考慮してBCMの実装・最適化を行い，その性能評価に基づき実効性能を議論する．

1. 緒言

1960年代以降，コンピュータを用いた数値計算が流体力学におけるシミュレーションや解析に用いられてきた．一般的にCFDの分野では，複雑な形状を再現するために，境界適合格子(boundary-fitted mesh)や非構造格子(unstructured mesh)が広く用いられる．これらの格子を用いることにより，流体解析の対象となる形状を忠実に再現することが可能なため，より正確なCFDを行うことができる．しかしながら，近年，飛行機や高速鉄道，自動車などのように，非常に形状が複雑な物体がCFDの主な対象となっている．そのため，非構造格子を用いたCFDにおいては，モデルデータからの格子作成などの処理を行う前処理，複雑なモデルの流体解析計算，その解析結果に基づく可視化やデータ抽出などの後処理に非常に膨大な時間が必要となる．また，大規模計算システムを用いた並列処理に必要となる分割方法も複雑化し，演算負荷を均等に保つのが難しいため，非構造格子に基づく大規模CFDの効率的な並列処理は困難になっている．

この問題を根本的に解決するために，次世代のCFDソルバーであるBCMが提案されている[1,2,3]．BCMは大規模計算システムを用いて様々な流体現象を効率的にシミュレーションするために，格子の形状が単純な直交格子(Cartesian mesh)を採用している．これにより，複雑化・煩雑化した前処理・流体解析・後処理を，単純にすることができる[4]．さらに，BCMはCFDに必要となる演算を容易に均等分割することができるため，大規模計算システムにおいて負荷のバランスを保った効率的な並列処理を実現できる．

本報告では，大規模計算システムを用いてBCMの性能評価・解析を行い，BCMの特性を明らかにする．近年の大規模計算システムの性能向上は著しく，多様な大規模計算システムが登場している．汎用プロセッサを搭載したスカラ型大規模計算システムや描画処理用プロセッサ(Graphics Processing Unit, GPU)を計算に応用するアクセラレータ型大規模計算システム，大量の計算を同時に処理するベクトル型大規模計算システムなど，プロセッサのアーキテクチャやシステム構成によりその特徴が異なる．それぞれの大規模計算システムの特徴を考慮してBCMの実装と最適化を行い，その性能評価・解析を行うことで，BCMの特徴を明らかにし，更なる高速化の検討を行う．

2. 等間隔直交格子を用いたBuilding Cube Methodの概要

BCMは，高密度格子を必要とするような複雑な物体周りの大規模3次元流体解析のために設計されている．図1に示すように，BCMではCFDの解析領域を***cube***と呼ばれる部分領域に分割し，さらにそれぞれのcubeを***cell***と呼ばれる高密度の等間隔直交格子で分割する．cubeのサイズは物体の形状や流れの特徴に応じて決められる[4]．格子の形状が非構造格子のように複雑ではな

いため，BCM のアルゴリズムをシンプルにすることができ，前処理や流体解析，後処理の高速化が期待できる．

BCM は演算を容易に均等分割することができるため，並列処理に適している．cube ごとの計算は完全に独立しており，各 cube の計算に必要となる演算量とデータ量は同一である．そのため，cube 単位で計算を分割することにより，均等な演算に分割することができる．さらに，cube 内の隣接する cell 同士には依存関係があるが，各 cell の計算に必要となる演算量とデータ量は同一である．そのため，cell 同士の依存関係を解消することができれば，更なるデータ並列性を抽出することが可能になる．

図 2 に BCM による非圧縮性流体解析のフローチャートを示す[3,4]．BCM では，流れの基礎方程式に非圧縮性 Navier-Stokes 方程式を用いており[5,6,7]，スタガード配置(staggered arrangement)の要素に対して有限差分法(finite difference scheme)による部分段階法(fractional-step method)で計算を行う．部分段階法では，流体解析を時間ステップごとに，仮速度場計算ステージ，圧力場計算ステージ，速度場計算ステージの主に 3 つのステージに分けている．各ステージには場の計算と隣接する cube 間とのデータ交換が含まれている．

これらのステージの中で，SOR(Successive over-relaxation)法を用いてポアソン方程式を解く圧力場の計算が最も支配的である．これは，隣接する 6 つの cell を用いたステンシル計算を，圧力場の差分が収束するまで，全ての cube の全ての cell に対して繰り返し計算を行う必要があるためである．

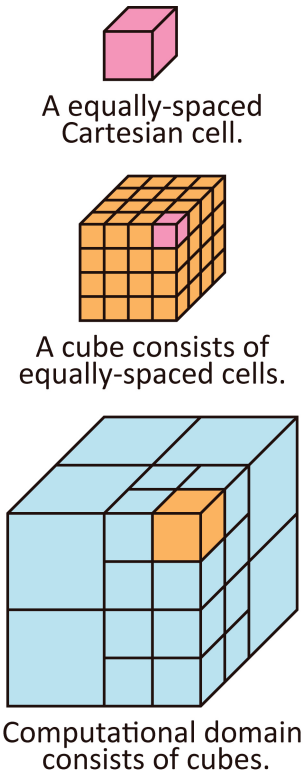

図 1　BCM における計算格子

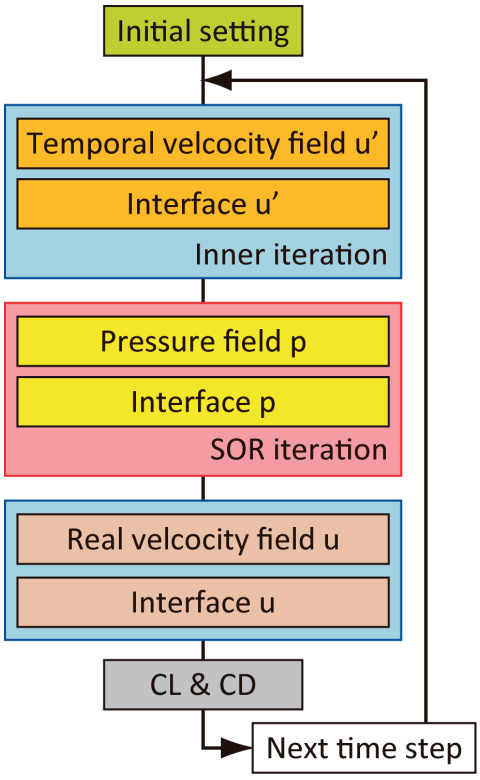

図 2　BCM における流体解析の流れ

3. 大規模計算システムへのBCMの実装

大規模計算システムによる並列処理によってBCMの計算時間を短縮させるためには，プロセッサのアーキテクチャやシステム構成を考慮した実装と最適化が必要になる．本章では，表1に示すプロセッサを搭載する大規模計算システムの概要を述べ，それぞれの大規模計算システムへのBCMの実装・最適化について述べる．

表1　大規模計算システムに搭載されるプロセッサの諸元

システム	理論演算性能 (Gflops/s)	メモリバンド幅(GB/s)	コア数	オンチップメモリ	B/F値
Nehalem EP	46.93	25.6	4	256KB L2/core 8MB shared L3	0.55
Nehalem EX	74.48	34.1	8	256KB L2/core 24MB shared L3	0.47
FX-1	40.34	40.0	4	6MB shared L2	1.0
SR16000 M1	245.1	128	8	256KB L2/core 32MB shared L3	0.52
SX-9	102.4	256	1	256KB ADB	2.5
Tesla C1060	78	102	1	16KB/SM	1.3

3.1 スカラ型大規模計算システムにおけるBCMの実装

Intel Nehalem EPクラスタやIntel Nehalem EXクラスタ，富士通FX-1，日立SR16000 M1は，それぞれNehalem EP，Nehalem EX，SPARC64VII，IBM Power 7といった多数の汎用型スカラプロセッサを搭載するスカラ型大規模計算システムである．表1に示すような複数のコアを持つスカラプロセッサを1つまたは複数搭載し1ノードを構成し，このノードをInfinibandなどの高速なネットワークで多数接続することで，スカラ型大規模計算システムが構築されている．スカラプロセッサは複数のデータに対して同じ演算を実行することが可能なSIMD(Single Instruction Multiple Data)命令や，局所性の高いデータへのアクセスレイテンシを削減するための大容量オンチップキャッシュメモリを備えている．

BCMをスカラ型大規模計算システムで実装するためには，多数のスカラプロッセッサの有効活用が重要となる．BCMの高いcube並列性を利用し，cubeを計算システムのノード，そしてノード内のプロセッサへ，プロセッサ内のコアへと階層的に割り当てることで，多数のコアを用いた並列処理を行う．cube毎の演算量は完全に同一であるため，ノード間，プロセッサ間，コア間の演算は均等となり，多数のスカラプロセッサを利用した効率的な計算が可能になる．スカラプロセッサにおける大容量キャッシュメモリを効率的に利用するために，データへのアクセスが連続であり局所性が高い通常のSOR法を用いる．また，SIMD命令を利用するために，コンパイラによる最適化を行っている．より効率的にSIMD命令を利用するためには，スカラプロセッサごとにそれぞれ定義されているインラインアセンブラなどを用いた実装が必要になる．

3.2 ベクトル型大規模計算システムにおけるBCMの実装

NEC SX-9は大規模な*SMP(Symmetric Multi Processing)*ノードから構成されるベクトル型大規模計算システムである．各SMPノードは102.4Gflops/sのベクトルプロセッサを16個搭載している．SX-9ベクトルプロセッサでは256要素の同一演算を同時に処理することができる．

SX-9による効率的な処理を実現するためには，BCMの並列性をできる限り抽出し，できるだけ多くの要素を同時に演算する必要がある．そのため，SX-9におけるBCMの実装には，cubeの並列

性だけでなく，cell の並列性も利用可能な，Red-Black SOR 法を採用する．Red-Black 法では隣接する cell 同士の依存関係を解消するために，cube 内のすべての cell を図 3 に示すような red グループと black グループに分ける．red グループの cell 同士は依存関係がないため，red グループのすべての cell の圧力場を同時に計算することが可能となる．同様に，red グループの圧力場の計算が終わった後に，black グループのすべての cell の圧力場を同時に計算する．この Red-Black SOR 法により，cube の並列性だけでなく cell の並列性も利用して，複数の cell の圧力場の計算をベクトルプロセッサで同時に処理することができる．

しかしながら，一般的にベクトル処理による Red-Black SOR 法では，同時に処理できる要素数が減ってしまいベクトル演算性能が低下してしまう可能性がある．これを抑制するために，マスクテーブルを用いて red グループと black グループを判別することにより，ベクトル演算に十分な要素数を確保する．

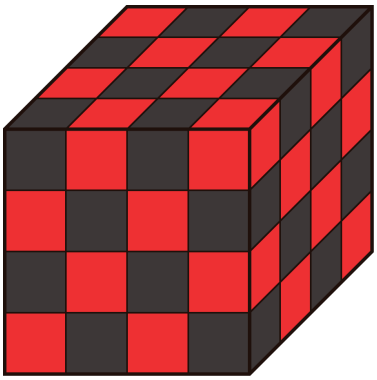

図 3　Red cell グループと black cell グループ

```
・・・
      !cdir ON_ADB(prs)
      do icolor=1, 2
            do icell=icolor,max_cell*max_cell*max_cell,2
                  if(mask(icell,1,1,icolor).eq.1) then
                        p0   = prs(icell,1,1)
                        pxm = prs(icell-1,1,1)
                        pym = prs(icell-max_cell,1,1)
                        pzm = prs(icell-max_cell*max_cell,1,1)
                        pxp = prs(icell+1,1,1)
                        pyp = prs(icell+max_cell,1,1)
                        pzp = prs(icell+max_cell*max_cell,1,1)
      ・・・
```

図 4　SX-9 における圧力場計算の疑似コード

また，SX-9 の性能をさらに引き出すために，プロセッサに搭載されているオンチップメモリである ***ADB(Assignable Data Buffer)*** を活用する．ADB は 256KB の容量を持つソフトウェア制御が可能なオンチップキャッシュメモリである．プログラマに指示されたデータに一度アクセスされると ADB に保存され，次のアクセスの際に ADB からデータを取得することができる．図 4 に示すように，圧力場のステンシル計算の際に 7 回再利用されるデータを ADB に保存する ON_ADB ディレクティブをソースコードに挿入する．これによって，メインメモリと ADB の両方から別々

のデータをベクトル演算器に供給することができるため，より高い実効メモリバンド幅を実現し，BCM の計算を高速に実行できることが期待できる．

3.3 アクセラレータ型大規模計算システムにおける BCM の実装

代表的なアクセラレータ型大規模計算システムの 1 つである *GPU(Graphics Processing Unit)* クラスタでは，1 つまたは複数の GPU を搭載したノードが多数接続されている．GPU は *CUDA(Compute Unified Device Architecture)* では，数百の *SP(Stream Processors)* からなるメニーコアのプロセッサと考えることができる[5]．CUDA では SP が *SM(Stream Multiprocessor)* としてグループにまとめられ，SM の中の SP が同時に動作する．

GPU クラスタでは，1 つの GPU に数百コアも搭載されているため，システム全体のコア数は膨大になる．また，GPU はメモリアクセス中に他の演算を実行することで，メモリアクセスレイテンシを隠蔽することが可能なため，GPU の高い性能を引き出すためには並列処理が可能な処理を可能な限り増やす必要がある．そのため，GPU クラスタへの実装においても cube と cell の両方の並列性を抽出可能な Red-Black SOR 法を用いる．図 5 に示すように，cube をサブグループに分けて，各ノードへ割り当てる．割り当てられた cube をさらに GPU の SM に割り当てる．cube 中に含まれる各 cell の計算を thread に割り当て，SP で並列に実行する．各 cell の演算量は同一であるため，複数ノードの GPU を用いた効率的な大規模並列処理が期待できる．

GPU の高い演算性能をさらに引き出すためには，演算の割り当てのみではなく SP へのデータ供給が非常に重要となる．GPU のメモリは階層構造になっており，*共有メモリ(shared memory)* は各スレッドから共有されている．共有メモリの容量は小さいが，メモリアクセスレイテンシは非常に短い．一方，*大域メモリ(global memory)* は，容量が大きいオフチップメモリであり，メモリアクセスレイテンシは長い．最大限に共有メモリを活用し，大域メモリへの効率的なアクセスを行うなど，これらの両方のメモリを適切に利用することで GPU の性能を引き出すことができる．

容量の小さな共有メモリを効率的に利用するために，図 6 に示すように，共有メモリ上にリングバッファを構築する．ステンシル計算に利用される cell を平面単位でリングバッファに保存することによって，大域メモリへのアクセス数を削減するだけでなく，限られた共有メモリの容量を効率的に利用することが可能になる．

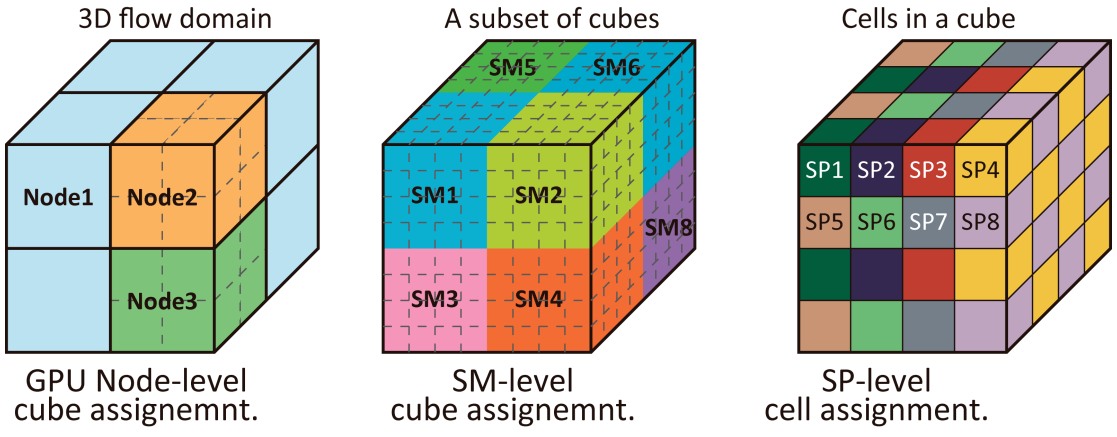

図 5　GPU クラスタ大規模計算システムのおける計算割当

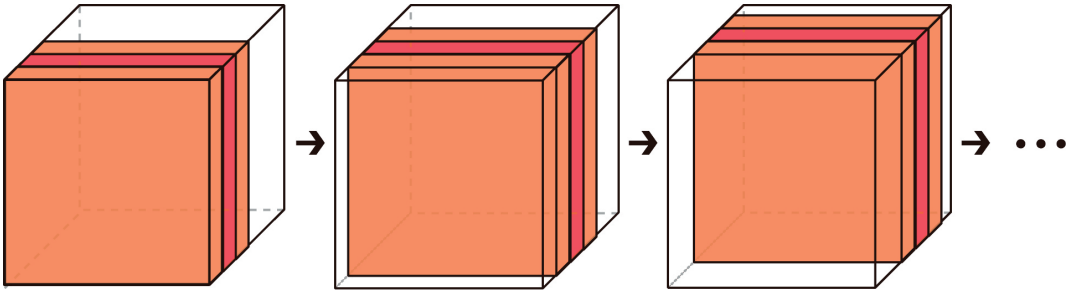

図 6　リングバッファを用いた効率的な共有メモリの活用

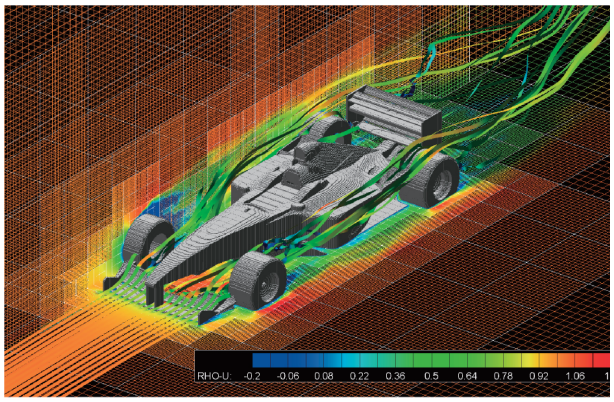

図 7　F1 モデル(1 億 cell)

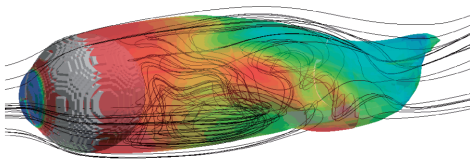

図 8　球体モデル(500 万 cell)

4.　大規模計算システムにおける BCM の性能評価

表 1 に示す大規模計算システム上で，BCM による 3 次元テストモデル周りの流体解析を実行して，その実効演算性能を評価した．3 次元テストモデルには図 7 に示す F1 モデルと図 8 に示す球体モデルを用いた．F1 モデルは複雑で大規模なモデルデータを想定しており，約 1 億 cell の格子からなる．球体モデルは比較的小規模なモデルデータを想定しており，約 500 万 cell の格子からなる．

図 9 に F1 モデルを用いた BCM による流体シミュレーションの実効性能を示す．この図より，SX-9 が他のスカラ型大規模計算システムに比べ高い実効性能を達成しているのが分かる．ステンシル計算はデータ転送あたりの演算が少ないため，実効メモリバンド幅が BCM の実効性能に影響する．SX-9 は，高い理論メモリバンド幅に加え，ADB を効率的に利用することにより，実効メモリバンド幅を高めることができる．これにより，SX-9 が他のスカラ型大規模計算システムに比べ，高い実効性能を達成することができたと考えられる．

FX-1 の理論メモリバンド幅が Nehalem EP や Nehalem EX の理論メモリバンド幅よりも高いにも関わらず，FX-1 の実効性能は Nehalem EP や Nehalem EX に比べ低くなっている．これは，FX-1 の実効メモリバンド幅が Nehalem EP や Nehalem EX の実効メモリバンド幅よりも低いためである．実効メモリバンド幅を測定するためのベンチマークソフトである STREAM ベンチマークを実行した結果と，Nehalem EP や Nehalem EX はそれぞれ 17.0GB/s，17.6GB/s のメモリバンド幅を達成しているにもかかわらず，FX-1 は 10.0GB/s と実効メモリバンド幅が低いことが分かる．このため，Nehalem EP や Nehalem EX の実効性能が FX-1 の実効性能を上回った．

図 10 に球体モデルを用いた BCM による流体シミュレーションの実効性能を示す. 球体モデルの結果には GPU クラスタによる結果も含まれている. この結果を見ると, GPU の大域メモリの容量が限られるため F1 モデルのような大きなモデルに対しては実行できないが, 球体モデルのようなモデルにおいては, GPU クラスタの性能が SX-9 とほぼ同等, スカラ型大規模計算システムよりも高いことが分かる. これは多数の SP に効率的に演算を割り当てることで, 高いデータ並列処理能力を発揮できたためである. また, 共有メモリを効率的に利用することによって, スカラプロセッサに比べ高い実効メモリバンド幅を引き出すことができたことも要因の 1 つである. もう 1 つの要因としては, 球体モデルのデータが小さいため, SX-9 や他のスカラ型大規模計算システムにおいては十分な性能を引き出すことができないことが挙げられる. 図 9 に示す F1 モデルから得られた実効性能と比較すると, 球体モデルから得られた実効性能が低くなっているのが分かる.

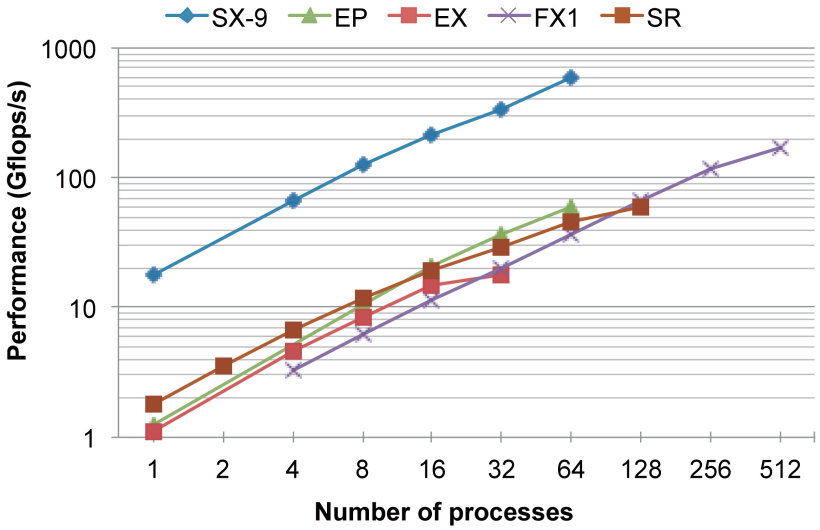

図 9　F1 モデルを用いた場合の BCM の実効性能

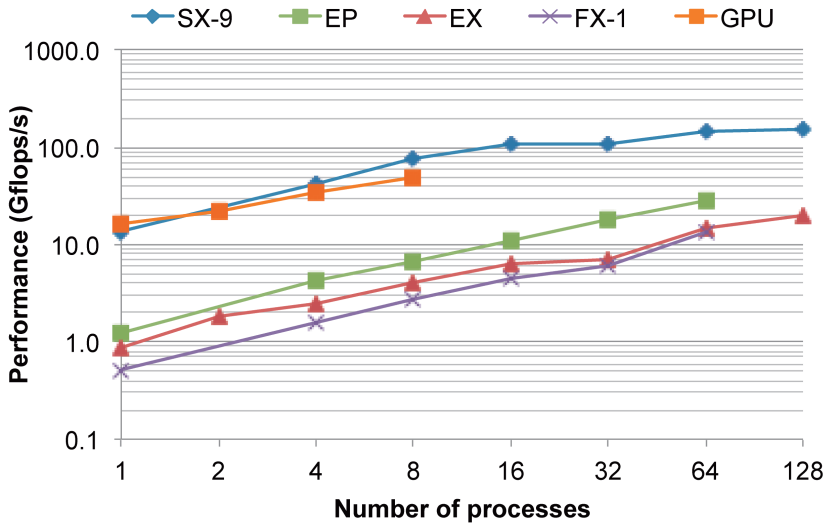

図 10　球体モデルを用いた場合の BCM の実効性能

GPU による流体シミュレーションにおいては，圧力場の計算時間の大部分がデータ転送で占められる．これは，GPU の活用によって演算時間が短縮されたが，ノード内の GPU と CPU や他のノードとのデータ転送に時間がかかるためである． GPU クラスタを用いたさらなる高速化を実現するためには，演算中にデータを転送しデータ転送時間を隠蔽するなどの工夫が必要となる．

図 9 と図 10 において，理論性能に対する実効性能の比率である実行効率を見ると，SX-9 が約 17%と他の大規模計算システムの 1.5〜3.2%と比べ非常に高いことが分かる．これは高い実効メモリバンド幅を実現することによって，ベクトルプロセッサが効率的にベクトル処理を行うことができたためである．

図 11 に示されている F1 モデルにおけるスケーラビリティをみると，すべての大規模計算システムにおいて高いスケーラビリティが実現できているのが分かる．これは，F1 モデルの並列度が高いため，十分な並列処理可能な計算を割り当てることができたことと，ノード間をつなぐネットワークのバンド幅が十分であったためだと考えられる．図 12 に示す球体モデルにおけるスケーラビリティは F1 モデルのスケーラビリティと比べ低い．GPU クラスタにおいては，データ転送のオーバーヘッドが要因でスケーラビリティが低下したと考えられる．他の大規模計算システムにおいては，球体モデルでは並列可能な処理が不足したため，スケーラビリティの低下が見られる．

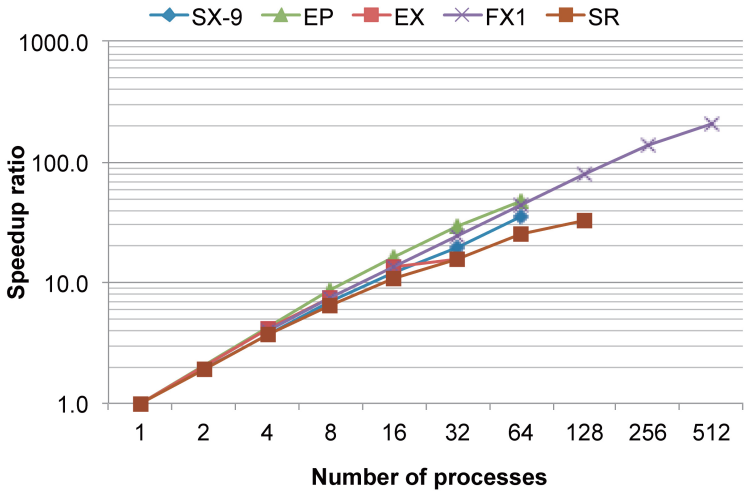

図 11　F1 モデルにおける BCM のスケーラビリティ

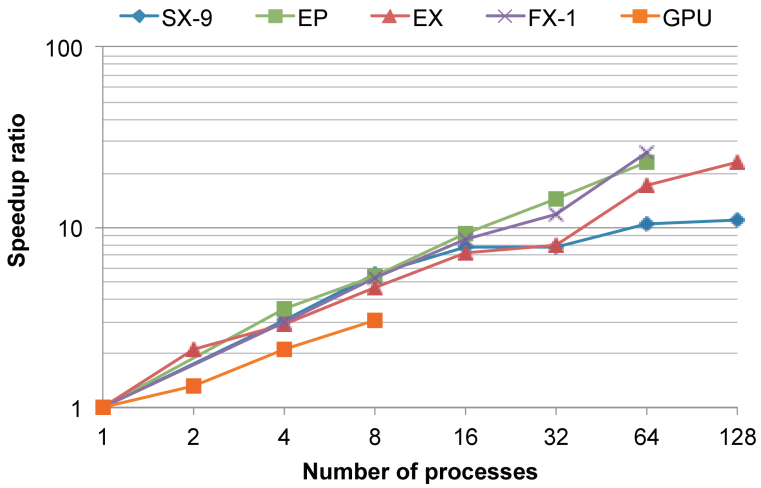

図 12　球体モデルにおける BCM のスケーラビリティ

5. 結言

本報告では，スカラ型・ベクトル型・アクセラレータ型の大規模計算システムを考慮したBCMの実装と最適化を述べ，その性能評価を通じて実効性能を議論した．それぞれの大規模計算システムの性能を引き出すために，大規模計算システムに搭載されているプロセッサのアーキテクチャやシステム構成を考慮してBCMの実装を行った．特に，それぞれの大規模計算システムにおけるプロセッサとそのオンチップメモリを効率的に利用するための最適化を行った．SX-9, Nehalem EPクラスタ，Nehalem EXクラスタ，FX-1，SR16000 M1，GPUクラスタを用いて性能評価を行った結果，実効メモリバンド幅がBCMの性能に大きな影響を及ぼしていることが明らかになった．これにより，SX-9のような高メモリバンド幅の大規模計算システムがBCMによる流体シミュレーションの高速化に適していることが分かった．

謝辞

本研究を進めるにあたりJAXA 中橋和博博士，金沢工業大学 佐々木大輔講師，東京農工大学大学院 高橋俊助教，NEC 撫佐昭裕博士，東北大学サイバーサイエンスセンター関係各位には大変有益なご助言をいただいた．また，本研究は，北海道大学情報基盤センター，東北大学サイバーサイエンスセンター，名古屋大学情報基盤センター，ドイツ シュトゥットガルト大学HLRSのスーパーコンピュータを利用することで実現することができた．また，本研究の一部は，文部科学省科研費研究(S)(21226018)と文部科学省科研費若手研究(B)(23700028)と科学技術振興機構(JST)戦略的創造研究推進事業(CREST)の助成を受けている．

参考文献

[1] Nakahashi, K., “High-density mesh flow computations with pre-/post-data compressions,” AIAA paper, pp. 2005–4876, 2005.
[2] Takahashi, S., Ishida, T., Nakahashi, K., Kobayashi, H., Okabe, K., Shimomura, Y., Soga, T., Musa, A., “Study of high resolution incompressible flow simulation based on cartesian mesh,” AIAA paper 47th AIAA Aerospace Sciences Meeting, pp. 2009–563, 2009.
[3] Takashi, S., “Study of large scale simulation for unsteady flows,” Ph.D. thesis, Tohoku University, 2009.
[4] Ishida, T., Takahashi, S., Nakahashi, K., “Efficient and robust cartesian mesh generation for building-cube method,” Journal of Computational Science and Technology **2**(4), pp. 435–445, 2008.
[5] Kim, J., Moin, P., “Application of a fractional-step method to incompressible navier-stokes equation,” Journal of Computational Physics 59, pp. 308–323, 1985.
[6] Perot, J.B., “An analysis of the fractional step method,” Journal of Computational Physics 108, pp. 1–58, 1993.
[7] Dukowicz, J.K., “Approximate factorization as a high order splitting for the implicit incompressible flow equations,” Journal of Computational Physics 102, pp. 336–347, 1992.