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

2007－ARC－172 (2)
2007－HPC－109 (2)
2007／3／1

CBE^{TM} を用いた大規模流体計算の高速化

新井佑介[†] 澤井 涼[††] 山口佳樹[††]
丸山 勉[††] 安永守利[††]

Cell Broadband Engine (CBE) は，マルチメディアデータの高速でリアルタイムな処理を目的として開発されたプロセッサである．CBE は，汎用プロセッサコアである PowerPC Processor Element(PPE) と，高速演算に特化した Synergistic Processor Element(SPE) の 2 種類のプロセッサコアを持つ，非対称マルチコアプロセッサである．CBE は，マルチメディア系の処理を想定して設計されたプロセッサであるが，その高い演算性能は，科学技術計算にも応用できると考えられる．そこで本研究では，CBE を用いて流体シミュレーションなどに応用される格子ガスオートマトン法を実装し，検証を行った．実装して得られた速度向上は，約 1000 万格子規模の FHP-III モデルにおいて，マイクロプロセッサ（Pentium4 3.4GHz）と比較し，約 21 倍であった．

Improvement in large size fluid dynamics simulation with CBE^{TM}

Yusuke Arai,[†] Ryo Sawai,[††] Yoshiki Yamaguchi,[††]
Tsutomu Maruyama[††] and Moritoshi Yasunaga[††]

The Cell Broadband Engine (CBE) is developed as a high-performance multimedia processing environment. CBE is a heterogeneous multicore processor that incorporates the PowerPC Processor Element (PPE) and Synergistic Processor Elements (SPEs). Each SPE is a special purpose RISC processor with 128-bit SIMD capability. In this paper, we describe a new computation method for FHP-III model with CBE. FHP-III model is used for simulating fluid dynamics. The approach is very suitable for simulating very complicated shapes. However, the need to compute a large number of grids makes it unrealistic to simulate on a desktop computer. In our implementation, the speedup of FHP-III model is about 21 times compared with Pentium4 3.4GHz when there are about 10 millions lattices.

1. はじめに

Cell Broadband Engine(CBE) は，高解像度の動画やマルチチャネルの音声のリアルタイム処理，またマルチメディア系の処理や分散コンピューティング環境を想定し設計されたプロセッサである[1)]．9 つのプロセッサコアを搭載し，それらが最高動作周波数 4.6GHz で動作する．動作周波数を 3.2GHz としたときのピーク性能は，単精度浮動小数点数演算で 204.8GFlops，倍制度では 14.6GFlops である．これは，同じプロセステクノロジを利用した Pentium4 (3.8GHz) のピーク性能（単精度 15.2GFlops）と比較して約 13 倍であり，単一プロセッサとして非常に高いパフォーマンスを秘めていると言える．しかし，CBE はこれらのマルチメディアの用途だけではなく，科学技術計算に応用できると考えられる．本研究では，流体シミュレーションなどに応用される格子ガスオートマトン法を実装し，検証を行った．

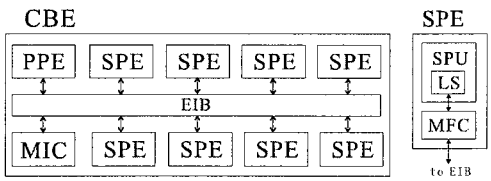

図 1 CBE(左) と SPE(右) の構成[1)]
Fig. 1 Implementation of CBE(L) and SPE(R)

† 筑波大学第三学群情報学類
College of Information Sciences, Third Cluster of Colleages, University of Tsukuba
†† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba

2. Cell Broadband Engine (CBE)

2.1 CBE の概要

CBE は，1 つの PowerPC Processor Element(以

下，PPE)，8つのSynergistic Processor Element(以下，SPE)の計9つのプロセッサコアを持つマルチコアプロセッサである．PPEは汎用プロセッサであり，CBE全体の制御を行う．OSもPPE上で動作する．一方，SPEは高速演算に特化されたプロセッサコアで，システム制御命令を持っていないが128ビットのSIMD演算が可能といった特徴を持っている．図1にCBEの構成を示す．

2.2 Synergistic Processor Element(SPE)

SPEは，Synergistic Processor Unit(以下，SPU)とMemory Flow Controller(以下，MFC)により構成されている．SPEの構造を図1に示す．

SPUは，SIMD型のRISCアーキテクチャのプロセッサユニットである．SPUは，高性能な浮動小数点数演算ユニットと26段の深いパイプラインによって高速な処理を可能としている．また，ハードウェアで分岐予測機構を持たず，インオーダーの命令発行などによって構造を単純化し，動作周波数の向上や低消費電力化を実現している．

SPUは，128ビット幅の汎用レジスタを128個持っている．演算は，128ビット単位で行われ，これによってSIMD処理を行う．Local Storageと呼ばれる256KBのメモリをプログラムとデータが置かれる内部メモリとして持っている．Local Storageは，MFCを介したDMAによって，外部とのデータ転送を行うことが出来る．

3. 格子ガスオートマトン法

格子ガスオートマトンとは，セルラオートマトン[2]を流体解析に適応しようとしたものであり，流体を粒子の集まりとして捉え，時間・空間・および速度について完全に離散化して取り扱うモデルである(図2)．1973年に提案されたHPPモデル[3]（図2(左)）は，格子の回転対称性が不十分であったため，流体の挙動を正確にシミュレートできなかった．そこで，Frisch等によって提案されたFHPモデル[4]が二次元非圧縮性流れの解析に用いられるようになった（図2(右)）．以上から，本論文でも実装対象はFHPモデルとした．

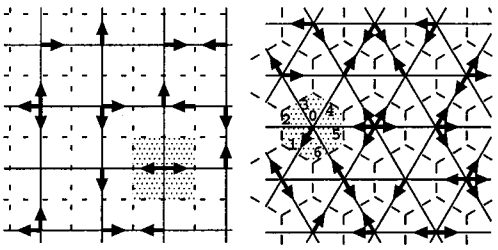

図2 格子構造，HPPモデル(左)とFHPモデル(右)
Fig. 2 Lattice structure, HPP(L) and FHP(R)

ここでFHPモデルについて考えると，ある一つの格子（図2の影がついている部分．HPPモデルでは四角形，FHPモデルでは六角形）には，六方向の速度を持つ粒子（図2(右)の1〜6の数字）と格子内で静止している粒子（図2(右)の0の数字）の計七個が存在できる．移動方向をiとし速度を$\mathbf{c}_i$とすると，$\mathbf{c}_i$(i=1〜6)は以下の式で表される．

$$\mathbf{c}_i = \left(cos\frac{(5-i)\pi}{3}, sin\frac{(5-i)\pi}{3} \right) \qquad (1)$$

また，上記$\mathbf{c_i}$を用いて，ある時刻tにおける二次元座標$\mathbf{x}$の状態（$n_i(\mathbf{x},t)$）は以下の様に表せる．

$$n_i(\mathbf{x}+\mathbf{c}_i, t+1) = n_i(\mathbf{x},t) + \Delta_i(n) \qquad (2)$$

この時，$\Delta_i(n)$は衝突による$n_i(\mathbf{x},t)$の変化量であり，± 1と0の何れかの値を取る関数である．図3にFHPの衝突側を示す．

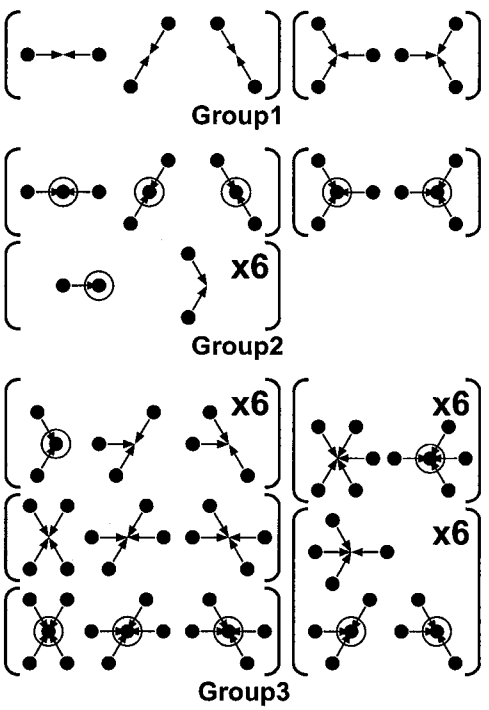

図3 FHPの衝突則
Fig. 3 FHP collisions

括弧内はそれぞれある格子中の全粒子の一状態を表しており，衝突により同じ括弧に含まれる他の状態に変化する．図3中に"**x6**"とあるのは，同様の衝突則が六種類あることを意味している．

FHPモデルにはI〜IIIがあり，その違いは採用する衝突則の違いである．FHP-Iでは，図3のGroup1の衝突則（5通り）のみが，FHP-IIではGroup1と2の

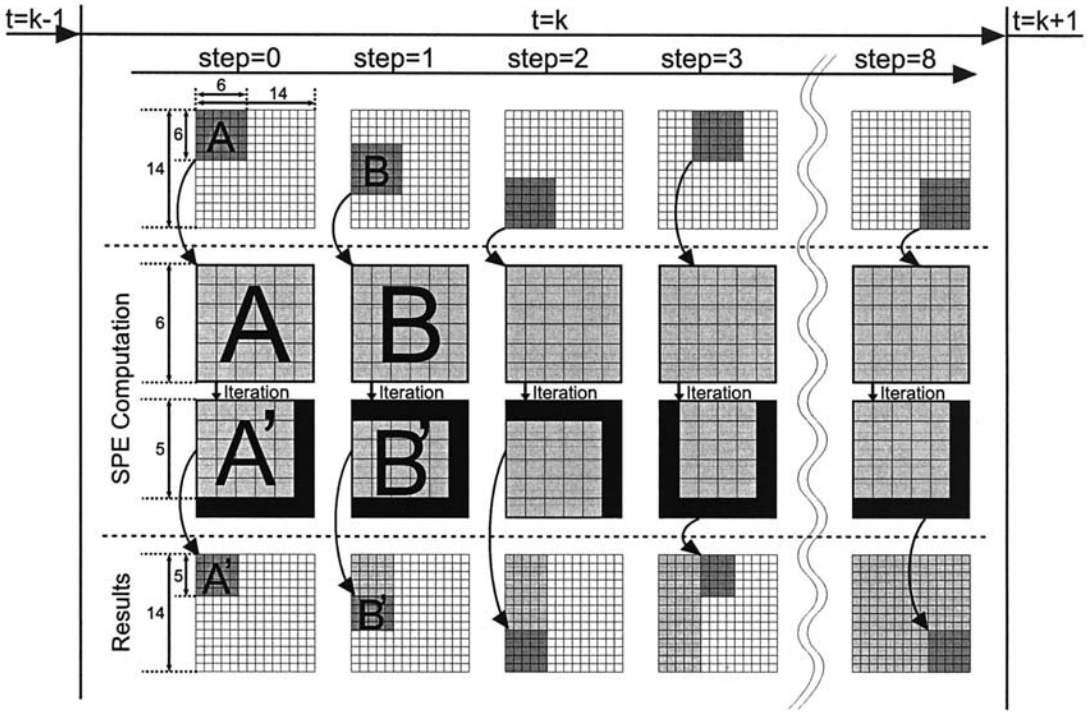

図 4　シミュレーション空間の分割と SPE による演算結果の格納

衝突側（22 通り）が，FHP-III では Group1〜3 の全衝突則（76 通り）がそれぞれ考慮されている．FHP-I は衝突則が 5 通りしかなく非常に簡単であるが，静止粒子の取り扱いがないため，体積粘性がゼロとなるなどの問題がある[5]．そこで本論文では FHP-III を取り扱うものとする．

4. 格子ガスオートマトンの実装

4.1 シミュレーション空間の分割

本論文では，FHP-III モデルを対象とし，シミュレーション空間は 2 次元としている．各格子点において必要となる情報は 6 個の移動粒子と 1 個の静止粒子の存在の有無であり，粒子を存在する (1)，存在しない (0) で表現すれば，各格子点は 7bit で表現できる．本論文では実装の都合上，各格子点を 1byte で表現することにした．

ここで，本研究で対象とするシミュレーション空間が 1000 万個の格子点を持つと考えると，計算結果を格納するに必要な記憶領域として約 10MB が要求される．本研究で使用する CBE は，7 個ある各 SPE の中に 256KB の Local Storage を持っているが合算しても 1792KB(=7×256) の記憶領域しか持たないため，全シミュレーション空間を記憶することができない．また，Local Storage は演算結果だけでなくプログラムにも利用されるため，CBE 内で記憶領域として利用できる領域は 1792KB からかなり減少してしまう．そこで本論文では，全シミュレーション空間は主記憶上に記憶することとし，各 SPE で記憶できる大きさ (約 220KB 程度) まで領域を論理的に分割して計算することで以上の問題を解決している．以上のシミュレーション空間分割手法とその分割された空間をどのように SPE で演算するかの概要を図 4 に示す．この図において，分割して得られた正方形の領域は小領域と呼ぶことにする．また，全シミュレーション空間は 196(=14×14) 格子点とし小領域は 36(=6×6) 格子点としている．

次に，図 4 において小領域を 1 個の SPE で順に演算していく方法について説明する．図 4 において，step は処理の順番を表し，矢印方向（step=0 から step=8 の方向）に処理が進んでいく．よって，step=0 より演算が開始され step=8 でシミュレーション空間の全格子点の演算が終了し，シミュレーションの単位時間 t が 1 インクリメントされて次の時間のシミュレーションが行われる．

ここで，SPE について着目すると，step=0 の時に全空間左上の小領域が SPE の Local Storage に読み込まれる．SPE では読み込まれた小領域（図 4 の A

領域）について演算を行うが，演算量に対するデータ転送を減らすため繰り返し演算（図4のIteration）を行う．このとき，小領域の境界となる格子については，境界を跨いだ隣の格子点の新しい演算結果を得ることができないため，繰り返し演算において正しい結果を得ることができない．このため演算を繰り返しが多くなると，正しい演算結果を返す領域が狭まっていく．図4において，A領域（6×6格子点）に1回の繰り返し演算を適用したため正しい結果を得られた領域がA'領域（5×5格子点）に減少していることがわかる．

SPEでの演算が終了すると，その結果をメモリに格納する．このとき，繰り返し計算により得られた正しい結果のみが格納され，黒く塗りつぶされている領域（境界面にあるため正しく計算できなかった領域）は破棄される．これをこの後，step=1~8まで繰り返し，全演算結果を出すことができる．

この手法において，繰り返し演算を行うことでデータ転送のボトルネックを軽減できる一方，計算の無駄が生ずることが問題となる．例えば，シミュレーション空間においてA領域では2辺が他の領域と境界を共有しているのに対して，B領域では3辺を共有しているため無駄な計算量が多くなっている．全空間を広く設定し4辺を共有する小領域が増加する場合，繰り返し回数の設定を誤るとデータ転送に対する効果より破棄される演算コストが高くなるため全体のパフォーマンスが低下してしまう．そこで本論文では，プログラムサイズ（約22KB）から逆算し，小領域の大きさを488×488とした．次に，現在までの実装の知見より[8]，繰り返し演算の回数を32回として実装を行った．

4.2 メモリアクセスとメモリ使用量の削減

複数のSPEを利用する場合，図4の各小領域を順に割り当てていくことになる．ここで，複数のSPEを用いた並列計算考える．演算が終了したSPEにより未読の隣接する小領域のデータに結果の上書きが行われると，次回以降の演算時に正しいデータが読み込めなくなるため，シミュレーション全体で正しい計算結果を維持できなくなるという問題がある．このためデータ不整合を生じないような手順を導入するか，データの配置方法の何れかについて考える必要がある．本論文では，主記憶が十分に存在することから読み込むデータとは別の場所にデータを格納することで，この問題を回避している．しかし，記憶領域を2倍にしたり一時記憶領域を設けるのは領域及び結果を再配置するなどののコストが大きいため好ましくしくない．そこで，本研究では，これらのコストの削減を考慮し，図5のようにデータ領域を取ることにした．

図5は，t=k，t=k+1のメモリの状態を示している．斜線部は，有効なデータを表しており，Xはシミュレーション空間の横幅，Yは同じく縦幅である．図5において，t=kの時点でデータが右側に格納されている．marginと書いてあるところには，無効なデータ

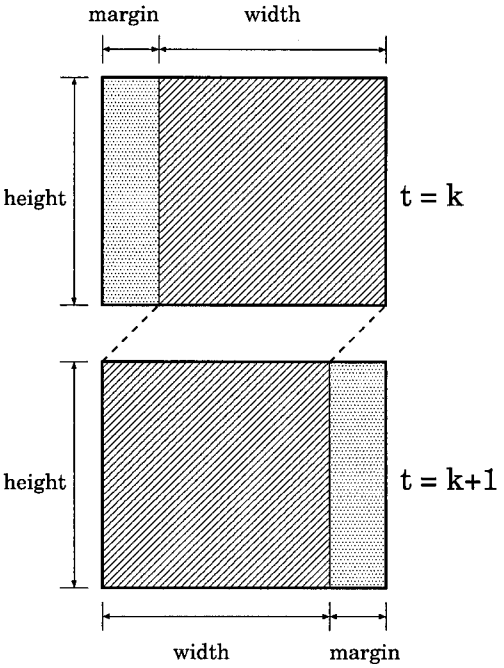

図5 データのメモリ上での配置

しか存在しない．このとき，次の計算は左端のデータから行う．左端のデータ(斜線部の左端)を読み込み，計算結果の書き込みはメモリ領域の左端から格納していく．シミュレーション空間全体の計算が終わったとき，データは左側に格納されている(図5，t=k+1)．次の計算は，データの右端から始め，同様に計算結果をメモリ領域の右端から格納していく．このように交互に記憶領域を移動させることによって，データの整合性を保ちながらメモリサイズの無駄を省いた．本論文では，marginの幅は小領域の幅の半分とした．

5. 評価結果と検証

演算結果のスナップショットを図6に，計算時間及び速度向上比について表1に示す．評価実験は，シミュレーション空間は約1000万格子，世代数は約1万のシミュレーションを行った．ソフトウェア実行環境はkernel2.6.17，gcc4.0.4であり，ソースコードも冗長な部分を可能な限り省きCBEが行っているビット演算のみを忠実に再現するよう努めた．gccの最適化オプションはO2を利用している．また，表1において，timeは1世代あたりの計算時間である．

ここで，表1に(SSE)と書いてあるが，これはSSEを用いた並列計算を利用した結果である．XeonとPentium4のSSEを用いた実装では，SPEのプログラムをそのまま移植したものを利用している．SSEを使用

図 6　32000 世代のスナップショット
Fig. 6　Snapshot of LGA, 32000 iterations

表 1　CBE による速度向上率
Table 1　Speedup gain

	Xeon	Xeon(SSE)	P-4	P-4(SSE)	CBE
clock(GHz)	3.06		3.4		3.2
# of core	1		1		7
L2 Cache	512KB		1MB		7×256KB
DRAM(GB/s)	4.2		6.4		25.6
time(msec)	1282.14	2175.69	940.323	2375.38	43.7324
sppedup	0.733	0.432195	1.00	0.396	21.5

していない場合との差異は，SSE を使用していないプログラムでは，格子点の状態に応じて分岐命令で処理を選択しているのに対し，SSE を使用しているプログラムでは，SPE と同様に命令分岐を出来る限り使わず，論理演算のみで計算を行うようにした．これは，SSE を用いる場合は 16 格子を並列に計算することが出来る代わりに，各格子の状態に応じて分岐をするというのが困難であり，また速度面でも不利になると考えたからである．表 1 より，CBE により約 21 倍の速度向上が得られたことがわかる．また，第 4.1 小節で説明したように，重複して計算している部分がかなりあるため実性能は更に上である．具体的には，一度のリードで 32 世代の繰り返しを可能にしたため，図 4 の黒い領域の幅が 32 格子分あるため，約 10%の無駄を生じている．このためこれを加味して速度向上率を考えると，トータルで約 23.7 倍の高速化を実現したとも考えることができる．

一方，シミュレーション空間全体を表現するために約 10MB のメモリを使用しているため，Xeon と Pentium4 ではキャッシュを活用できていない．今回 CBE に対して使った空間分割の方法を用いて Xeon と Pentium4 のキャッシュを効率良く利用することで，計算速度を向上させられる可能性がある．Xeon および Pentium4 のキャッシュサイズは，SPE の Local Storage のサイズよりも大きく，同じシミュレーション空間であれば分割数が少なくてすむなど，計算効率の改善効果は高いと考えられる．また，著者らは SPE が十分な性能を出していないことについて検証を行い，我々の実装方法にも問題があり，性能改善できる余地があることを確認している．これは，重複領域を計算するというだけでなく，SPE で SIMD を利用している場合に隠れた多くの無駄な計算があることが分かったからである．著者らは SPE の最高性能を引出すため，全計算を如何に 16 並列の SIMD で計算するかと DMA 転送を如何に少なくするか，のみに着目して実装を行った．ところが，格子ガスオートマトンでは，

シミュレーション空間内に存在する粒子数が少ないため (格子数に対し，粒子数が 7%程度)，粒子が存在しない格子点の数の方が粒子の存在する格子点よりも非常に多い．粒子が存在しない場合は処理を全く行わず次の格子の計算に続くのだが (SSE を使用していない場合)，SPE で計算する場合は 16 並列の SIMD 型で計算を行っているためある一格子に粒子が存在すると残りの 15 格子についても（無駄であっても）同時に計算する必要がある．これは，Xeon および Pentium4 の SSE を使用した場合と使用していない場合の計算速度の差として顕著に現れている．このため，計算が省略している部分についても計算性能を費やしており，これが大きな性能低下の原因だと考えられる．SPE のパイプラインが 26 段と深く現在最適なコーディングができていないという問題点も確かにあるが，本論文の実装においては上記理由の方が大きな影響を与えている．

6. おわりに

本論文では，格子ガスオートマトン法の FHP-III モデルを実装し，シミュレーション空間が約 1000 万格子，世代数が約 1 万世代のシミュレーションで，CBE の性能評価を行った．CBE(3.2GHz) の Pentium4(3.4GHz) に対する速度向上比は約 21 倍程度であり，SPE の強みである数値演算を全く使用せずに，十分な性能向上を得られることを示した．CBE での性能向上が見られた一方で，Xeon および Pentium4 については，最適化ならびに性能評価が不十分であった．また，SSE を使用していない場合では，格子点の状態によって処理内容が変わるため，シミュレーション条件によって計算量が変わる．すなわち，より正確な評価を行うためには，様々なシミュレーション条件において実験を行わなければならない．

今後の課題として，先に述べた Xeon と Pentium4 での実装の最適化と，種々のシミュレーション条件における実験，評価が挙げられる．また，CBE での実装においては，必要のない無駄になっている計算を可能な限り省き，さらに PPE でも計算を行うことで，計算速度の向上させて行く予定である．

謝辞 本研究の一部は株式会社東芝，ならびに株式会社東芝セミコンダクター社の協力により行われた．

参考文献

1) 東芝レビュー，Vol.61，No.6，pp.9-15，Jun. (2006).
2) 加藤恭義，他: **セルオートマトン法**, 森北出版株式会社, (1998).
3) J. Hardy, et al.: *Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions*, Phys. Rev. A, vol.13 no.5, pp.1949-1961. (1976).
4) U. Frish, et al.: *Lattice-gas automata for the Navier-Stokes equation*, Phys. Rev. Letters, vol.56 no.14, pp.1505-1508. (1986).
5) U. Frish, et al.: *Lattice gas hydrodynamics in two and three dimensions*, Complex Systems, vol.1 no.4, pp.649-707. (1987).
6) 日本工業標準調査会: **乱数発生及びランダム化の手順**, 日本規格協会, JIS Z 9031, (2001).
7) Sony: *Synergistic Processor Unit(SPU) Instruction Set Architecture*, Ver.1.1, Jan. (2006).
8) 新井佑介，他 *Cell Broadband Engine* **による格子ガスオートマトンの実装**, 電子情報通信学会コンピュータシステム研究会，信学技報 Vol.106, No.436, pp.81-86(CPSY2006-54), 2006 年 12 月.