計算工学講演会論文集 Vol. 30 (2025 年 6 月)
日本計算工学会

格子ボルツマン法の Cerebras CS-2 単体PE実装と性能測定

Single PE implementation and performance measurement of Lattice Boltzmann Method on Cerebras CS-2

酒井宏己 [1)]　折原冴保 [2)]　下川辺隆史 [3)]　宮島敬明 [4)]
Koki Sakai, Saho Orihara, Takashi Shimokawabe and Takaaki Miyajima

[1)]明治大学 理工学研究科 情報科学専攻 （〒 214-0033 神奈川県川崎市多摩区東三田 1-1-1, E-mail: ee217044@meiji.ac.jp）
[2)]明治大学 理工学研究科 情報科学専攻 （〒 214-0033 神奈川県川崎市多摩区東三田 1-1-1, E-mail: ee217045@meiji.ac.jp）
[3)]東京大学 情報基盤センター（〒 113-0032 東京都文京区弥生 2-11-16, E-mail: shimokawabe@cc.u-tokyo.ac.jp）
[4)]明治大学 理工学研究科 情報科学専攻（〒 214-0033 神奈川県川崎市多摩区東三田 1-1-1, E-mail: takaaki.miyajima@cs.meiji.ac.jp）

While recent computers have improved in computing performance, the ratio of memory bandwidth to computing performance, expressed as Byte/Flops values, has deteriorated. The Cerebras CS-2, which is designed for high-performance computing and has the world's largest chip, is expected to reduce and solve this problem. The Lattice Boltzmann Method (LBM) is a computational method used in computational fluid dynamics. It models fluid movement as a collection of microscopic particles and simulates fluids' macroscopic behaviour by calculating the movement and collisions of these particles. We port and implement LBM on a single PE of CS-2 and measure performance.

Key Words : *Wafer-Scale Computing, Cerebras CS-2, Lattice Boltzmann Method, performance measurement*

1. はじめに

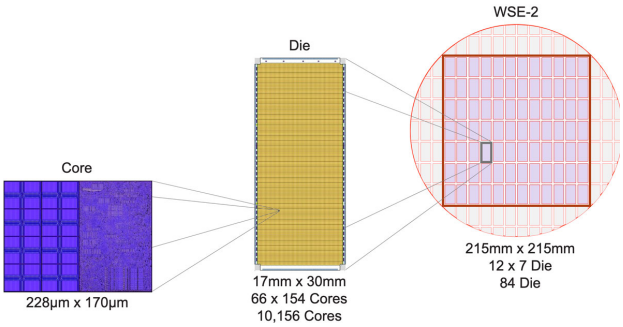
科学技術計算の分野では、気象予測や数値流体力学などの多様な分野で複雑な計算モデルの正確な計算が求められ、計算規模が肥大化している。こうした大規模科学技術計算には Graphics Processing Unit（GPU）が主に利用されている。近年の高性能計算機システムは、GPU やアクセラレータを搭載した数百から数千の計算ノードを集約したものになっている。しかし、ムーアの法則に従うようなスケールアップは物理的限界に達しはじめ、計算機間の通信がボトルネックとなり計算性能のスケールアウトも難しくなってきている。特に、Byte/Flops 比で示されるメモリ帯域と演算性能の比率は悪化し続けており、その問題を回避するために計算機の構造はさらに複雑化している。また、計算を複数の計算ノードに分割して並列処理する際は、ノード間のデータ転送がボトルネックとなり、並列化効率が低下する。このように、現在のシステム構成はさらなるスケールアップとスケールアウトに解決すべき課題が多い。これらの課題に対し、パッケージング技術、マイクロアーキテクチャ、プログラミングモデルなど、さまざまな観点から研究開発が進められている [1]。

Cerebras CS-2 は、深層学習における大規模言語モデルの学習ワークロードを高速化するために設計されたアクセラレータである。CS-2 は、300mm の半導体ウェハ全体を一つのチップとして使用した世界最大のチップである Wafer-Scale Engine (WSE-2) を搭載する。WSE-2 は 850,000 以上の演算コア（PE, Processing Element）を持ち、各計算コアは二次元メッシュトポロジで接続されている。WSE-2 全系での実効演算性能は半精度と単精度浮動小数演算でそれぞれ最大 2.98 と 1.02 PFlops を、メモリバンド幅はそれぞれ最大 8.94 と 8.93 PB/s と非常に高い計算性能と実効メモリ帯域幅を有する。CS-2 を用いた科学技術計算の試みが始まっている [2] [3] [4] [5] が、そのプログラミングモデルは、分散並列かつ非同期の処理を実装するもので、これまでのものとは大きく異なる。さらに、二次元メッシュトポロジへのタスクのマッピングの研究も進んでいない。

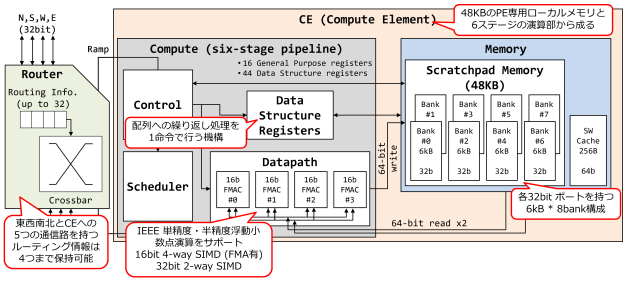
格子ボルツマン法（LBM, Lattice Boltzmann Method）は、数値流体力学の分野で用いられる計算手法の一つである。流体を微小な粒子の集まりとして近似し、各粒子の並進と衝突とを粒子の速度分布関数を用いて逐次計算することで、流体の大規模な動きを再現する。LBM では計算領域を小さな格子に分割し、各格子点で粒子の速度分布を計算する。粒子は時間の経過とともに格子点間を移動し、他の粒子と衝突して運動量やエネルギーを交換する。このステップを繰り返すことで、流体の流れをシミュレートする。LBM は、複雑な形状の境界条件や気液二相流を扱うのに適しており、工学や物理学のさまざまな分野で利用されている。また、計算が並列化しやすいため、大規模なシミュレーションにも向いている。

本研究の最終目的は、CS-2 の大規模科学技術計算への適用可能性を理解することである [6]。本論文では、CS-2 上に流体流体力学の数値計算手法の 1 つである格

© 一般社団法人日本計算工学会



図–1 WSE-2 と PE の物理的構造 [7]



図–2 WSE-2 の PE の内部構造

子ボルツマン法（LBM, Lattice Boltzmann Method）を実装し、その性能を測定した。今回は 1 つの PE で正しく LBM の計算を行えるようなナイーブな実装について述べる。得られた性能をもとに、全 PE を用いた場合の性能を推定し、ルーフラインモデルによる性能評価を実施した。

2. Cerebras CS-2

Cerebras CS-2 は、世界最大のチップ Wafer Scale Engine-2（WSE-2）を搭載する大規模な深層学習モデル向けのアクセラレータである [7]。筐体は高さ 26 インチで、標準的なデータセンターラックの 3 分の 1 のサイズである。CS-2 はホスト CPU クラスタに 100 Gbps Ethernet × 12 レーンで接続されたネットワークアタッチドアクセラレータとして設計されている。また、独自の電力供給アーキテクチャを採用し、標準部品とインタフェースを組み合わせることで、最大 23,000W の電力供給を実現している。システムの左上には、9+3 の冗長構成を持つ 12 基の標準電源が配置されており、安定した電力供給を可能にする。

(1) PE のアーキテクチャ

CS-2 はスケールアップを最大化するために、1 つのチップ内に計算を集約するアプローチを採用している。図 1 に WSE-2 と PE の物理的構造を示す。WSE-2 チップは、750×995 個の PE が二次元メッシュトポロジで接続され、総計で 40GB のオンチップメモリを物理的に搭載する。各 PE は簡素な計算コア（CE, Compute Elment）、48KB のローカルメモリ、およびルータで構成される。その面積は 38,000 μm^2 で、110,000 個のスタンダードセルで構成される。動作周波数は 850MHz で、ピーク時の消費電力は 30 mW に抑えられている。従来の GPU ではメインメモリを共有する設計が採用されているが、メインメモリとプロセッサとの物理的距離が動作周波数やアクセスレイテンシのボトルネックとなる。これに対し、CS-2 はメモリを CE の直近に分散配置することで、メモリ帯域をデータパスの計算帯域と同等の高さに引き上げている。

図 2 に PE の内部構造を示す。CE に専用の 48 KB の SRAM スクラッチパッドメモリ を搭載し、32 ビット幅の 8 つのシングルポートバンクに分割することで、1 サイクルあたり 64 ビットの読み出しを 2 回、64 ビットの書き込みを 1 回可能にしている。CE からローカルメモリへのアクセス遅延は 1 サイクルであるため、高い B/F 値と高い電力効率を実現している [8]。PE は、演算を担当する Computing Element（CE） と、PE 間のデータ交換を担うルータで構成される。CE には 48KB のローカルメモリと 6 ステージのインオーダー演算パイプラインを備え、両者は 64bit の読み出しポート 2 本と書き出しポート 1 本で接続されている。ローカルメモリは 48KB のスクラッチパッドメモリを持ち、SIMD 演算機を搭載する。メモリ階層は単一であり、データアクセスの遅延を最小限に抑えている。CE は大きく 4 つの機能ブロックで構成され、その中核であるデータパス部分には、IEEE 規格に準拠した半精度浮動小数点演算をサポートする積和演算機（FMAC ユニット） を 4 基搭載している。これにより、4-way 16bit または 2-way 32bit の SIMD 演算が可能となる。各メモリは独立したアドレスを持ち、共有型メモリは存在しないが、コア間のデータ共有は専用のパラレル線を介して行われる。各コアは、この SRAM にわずか 1 クロック・サイクルでアクセスできるため、従来のアーキテクチャに見られるオフチップ DRAM へのアクセスと比べて低レイテンシである。また、頻繁に使用されるデータ構造向けに 256 バイトのソフトウェアキャッシュがデータパスの近傍に配置されている。PE 間は二次元メッシュトポロジで接続され、通信には静的ルーティング手法を採用し、データフロー型の計算機としても利用可能である。これにより、メモリアクセスとデータ転送の遅延を大幅に低減し、多くの問題を通信制約型から計算制約型へと転換している。

(2) PE 間ネットワーク

二次元メッシュトポロジのカタログ値はインターコネクト帯域幅 220PB/s、メモリ帯域幅 20 PB/s である。HPC アプリケーションでは、隣接する計算機間の通信が頻繁に発生する場合、通信性能が計算性能の制約となることが多い。各 PE はルータを介して 4 つの隣接 PE と直接接続され、1 クロックサイクルでデータを転送できる。各データにはヘッダーが付与され、専用のスタティック・ルートが設定される。このアーキテクチャでは、各コアに統合されたルータが重要な役割を果たしている。このルータは、プロセッササイクルを消費することなく、事前に設定されたルーティング設定を利用して、介入することなく必要に応じてデータを受け

© 一般社団法人日本計算工学会

渡すことが可能である。データを直接 PE 間で送受信でき、いったんメモリに書き込む必要はない。この仕組みにより、通信オーバーヘッドを大幅に削減し、1 サイクルという低レイテンシを実現している。隣接 PE 以外の遠隔 PE へのデータ通信は、複数の PE を介したものとなる。対角にある PE 間の通信が最も遠くなり、750 × 995PE を介することになるが、PE 間の通信が 1 サイクルであるため、理論的には 1745 サイクル で通信が行える。これは従来のハードウェアのノード間通信レイテンシ（1ms 弱）と比べると大幅に短い。

(3) 開発環境

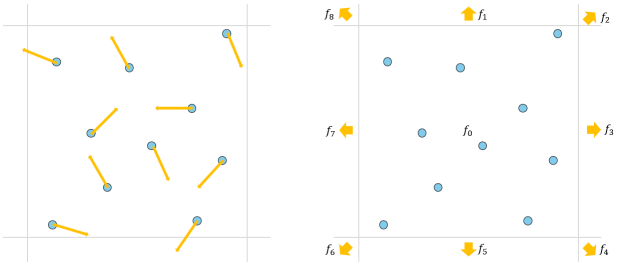
CS-2 上での汎用計算には Cerebras SDK を用いる。本 SDK は、プログラマが WSE-2 のマイクロアーキテクチャを直接ターゲットとする低レイヤのコードを書くことができ、本稿の LBM の実装にもこれを利用した。SDK では CS-2 をデバイス、ホスト CPU クラスタをホストと呼び、デバイス側は Cerebras Software Language（CSL）と呼ばれる専用プログラミング言語でプログラムされる。CSL は標準的な構文と機能を維持しながら、CS-2 に最適化されたコードを簡単に作成できるよう、多くのコアにまったく同じことを行わせることや、独自のタスクを割り当てたりできる。また、データ構造記述子（Data Structure Descriptors: DSD）と呼ばれる、テンソルへのアクセスパターン記述などに使用される特殊な記述子が用意されている。DSD とそれを操作するビルドイン関数を組み合わせることで、1 つのハードウェア命令でテンソルに対して所々の繰り返し演算を 1 つのハードウェア命令で実行できる。この柔軟性により、アイドルサイクルを排除して効率的な計算ができる。また、SDK には動作検証用のシミュレータが用意されている。シミュレータでは計算の所要時間がサイクル数の形で確認できるが、CS-2 の実機よりも低い値が得られる。本稿ではこれを用いて性能測定を行った。

3. 格子ボルツマン法

(1) 概要

格子ボルツマン法 (lattice boltzmann method: LBM) は、流体を有限個の速度をもつ多数の仮想粒子の集合体で近似し、各粒子の衝突と並進を粒子の速度分布関数を用いて逐次計算し、その速度分布関数のモーメントから巨視的流れ場を求める数値計算方法である。すなわち、流体を仮想粒子の集合体として近似し、その仮想粒子を並進と衝突を連続して起算することで、流体をシミュレーションする数値計算法である。近年、計算機の急速な進化により、膨大な格子点を扱う大規模計算が可能になった結果、理工学のさまざまな分野で高解像度計算や広域計算が盛んに行われるようになっている。特に、GPU などのアクセラレータはステンシル計算に適していることから、物理シミュレーションにおける大規模計算の分野で幅広く活用され、流れ場のシミュレーションなど数多くの成功例を生み出している [9]。

LBM は、格子に固定された計算点を使用して、各粒子の衝突と並進を計算する。図 3 のように、粒子を格子状にとらえることで、計算を効率的かつ簡単に行うことができる。格子点は格子状にとらえた粒子の存在する場所を表し、粒子は格子の方向に沿って一定のステップ幅で移動する。この仕組みにより、次の移動先や衝突の計算が統一され、数式が簡略化できる。また、格子モデルでは時間刻み幅が固定されており、一貫性を保ちやすく誤差の蓄積が抑えられるため、数値解法の安定性が向上する。さらに、格子状に空間を分割して連続的な流体を離散化し、計算機上で数値的に扱うことが可能になる。この方法は、並列計算にも適しており、格子点ごとの計算が独立しているため、スーパーコンピュータや GPU を利用して効率的に高速計算を行うことができる。物理的な直感や実装のしやすさも LBM の利点であり、粒子が格子の方向に移動し格子点で衝突するという仕組みは、物理現象を簡略化した直感的なモデルとして理解しやすく、プログラム作成も容易になる。格子を使えば壁や障害物などの境界条件をシンプルに設定できるため、壁にぶつかる粒子の反射などを自然に再現できる。これらの特徴から、LBM は複雑な流れシナリオや複雑な形状を扱うことができるため、航空宇宙や自動車、建築など幅広い分野で応用され始めている。



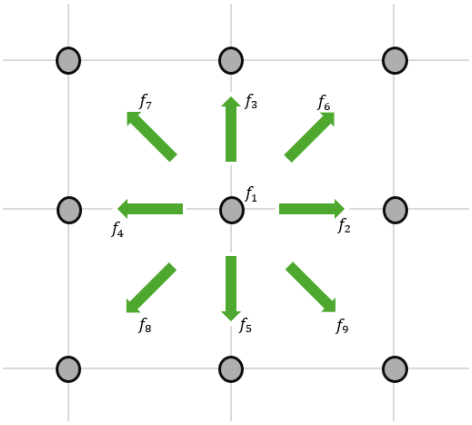
図–3 流体を分布関数でとらえる

LBM のアルゴリズムの構造やデータ依存性は並列計算に適している。アルゴリズムの構造として、並進と衝突の計算が 1 つの格子点ごとに完結するため、計算領域を独立した小領域に分割できる。データの依存関係の発生元となる空間的依存性は隣接する格子点のみであり、時間的依存性も前ステップの格子点のデータしか必要しない。ほかにも、領域を拡大して格子点数を増加させても、計算を均等に分割しやすいためスケーラビリティにも優れている。問題点として、LBM は比較的低い計算強度をもつことが多いため、メモリアクセスがボトルネックとなる場合が多い。以上の理由から、CS-2 は他の GPU と比べてコア数が非常に多く、コア間の通信のレイテンシも小さいため、スケーラブルな LBM の計算が期待できる。

(2) D2Q9 モデル

本研究では、LBM の平面二次元計算において図 4 に示す 2 次元 9 方向型格子（D2Q9 モデル）を使用する。D2Q9 モデルでは、各格子点に 9 つの速度ベクトル c_i を持つ分布関数 f_i が定義される。

© 一般社団法人日本計算工学会



図–4 D2Q9 モデル

$$\boldsymbol{c}_i = \begin{cases} (0,0), & i = 1, \\ (\pm 1, 0), (0, \pm 1), & i = 2,3,4,5, \\ (\pm 1, \pm 1), & i = 6,7,8,9, \end{cases} \tag{1}$$

LBM はボルツマン方程式を基に流体をモデル化し、D2Q9 モデルにおいて離散化することで基本的な計算ステップが導出される。

(3) 計算ステップ

a) 衝突（collision）

LBM は格子ボルツマン方程式より粒子分布関数 f_i の時間発展を解く。格子点ごとに粒子の分布関数を局所平衡状態に向かって緩和させる衝突モデルとして BGK モデルを採用する。このモデルは、次のような方程式で表される。

$$f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}[f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)] \tag{2}$$

ここでは、左辺は衝突計算後の分布関数であり、τ は緩和時間を表している。この式は、各格子点において、粒子の分布関数 f_i は時間とともに局所平衡分布関数 f_i^{eq} に向かって減衰することを表している。

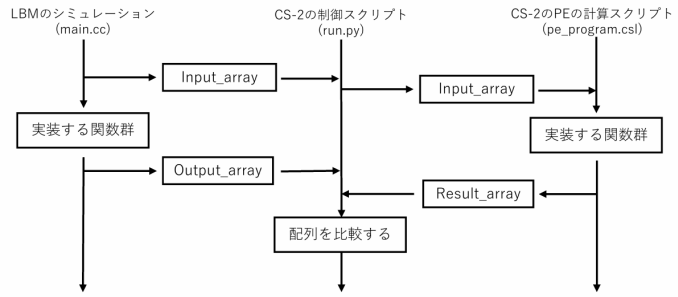
このモデルでは衝突後の分布関数 f_i を求めるために、局所平衡分布関数 f_i^{eq} を計算する。D2Q9 モデルの場合、マクロな流体変数である密度 ρ および流速 u を用いて次のように定義される。

$$f_i^{eq} = w_i \rho [1 + 3\boldsymbol{c}_i \cdot \boldsymbol{u} + \frac{9}{2}(\boldsymbol{c}_i \cdot \boldsymbol{u})^2 - \frac{3}{2}\boldsymbol{u} \cdot \boldsymbol{u}] \tag{3}$$

ここで、右辺の密度 ρ および 速度 $\boldsymbol{u}$ は、f_i から求められるものであり、定数 w_i は、D2Q9 モデル では、

$$w_i = \begin{cases} \frac{4}{9}, & i = 1, \\ \frac{1}{9}, & i = 2,3,4,5, \\ \frac{1}{36}, & i = 6,7,8,9, \end{cases} \tag{4}$$

となる。



図–5 移植の手順

b) 並進（streaming）

並進では、各格子点に存在する分布関数 f_i を、それぞれの速度ベクトル c_i に従って次の時間ステップで隣接した格子点に移動させる。基本式は次の通りである。

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta x, t + \Delta t) = f_i^*(\boldsymbol{x}, t) \tag{5}$$

4. 格子ボルツマン法の 1PE ナイーブ実装

格子ボルツマン法の CS-2 への移植と実装に際し、C++で記述されたプログラム（CPU 版）を CSL で書き直し（CSL 版）、関数ごとに検証を行った。各関数の役割は次の通りである。

1. *streaming_collision*：並進と衝突の計算を行う
2. *boundary*：境界条件を設定する
3. *swap*：与えられた配列の中身を入れ替える

本稿は 1PE で正しく動作することに焦点を絞り、複数 PE 実装に向けた予備評価や問題点の洗い出しを行う。実装には Cerebras SDK v1.2.0 を利用し、時間測定には SDK が提供するサイクルカウンタを利用し、所要時間は得られた所要サイクル数を動作周波数 850MHz で割った値とした。

(1) 実装手順

関数を一度に全て CSL で実装すると、不具合の特定に時間を要する可能性が高い。そのため、今回は CPU 版の各関数を順に CSL に移植して検証を行う単体テストの形式で実装を進めた。大まかな移植の手順を図 5 に示す。CPU 版の各関数が計算に必要とする入力配列と、関数によって更新される出力配列をバイナリデータとして保存した。CSL 版の関数に入力配列を入れ、計算結果を出力配列と比較して検証を行った。CPU 版は全ての計算を単精度浮動小数点で行っているため、検証に際して許容する誤差は小数点以下第 6 位とした。これ以下の誤差は、丸め誤差や計算の順序により誤差により発生したものと考えられる。なお、CPU 版に存在する可視化やログ出力の処理は移植の対象外とした。

すべての関数において単体テストを実施した後は、いくつかの関数を組み合わせて結合テストを実施した。この段階でも、関数間でデータの受け渡しが正確に行われること、そして出力結果が同様に許容誤差内に収まることを確認した。結合テストでは、個々の関数が独立して正確に動作していても、複数の関数が連携する際

© 一般社団法人日本計算工学会

に不整合や計算誤差が発生する可能性があるため、注意が必要である。結合テストを実施した後は、実際にシミュレーションループを数周行い、発生した誤差がループを重ねても変化しないことを確認した。

Listing 1 に CSL 版の collision 関数の主要部を示す。CSL は C 言語をベースに開発されているため、記述の容易さという利点を有している。

Listing 1 衝突処理の CSL 実装

```
// 局所平衡分布関数の用意
var feq: [NQ] f32;

// 密度・速度の用意
var rhos: f32 = 0.0;
var us : f32 = 0.0;
var vs : f32 = 0.0;

// (3)式の局所平衡分布関数の計算
lbm_rho_velocity(&fs, &rhos, &us, &vs);
lbm_feq(rhos, us, vs, &feq);

// 緩和時間とその逆数の計算
// (2)式の右辺の第二項の定数
const tau: f32 = 3.0*(vis[ix]/(c_ref*c_ref*dt))+0.5;
const omega: f32 = 1.0 / tau;

// (2)式の衝突の計算
for (@range(u16, NQ)) |l| {
    fs[l] = fs[l] - omega * (fs[l] - feq[l]);
}
```

(2) 実装結果

本稿では円柱周りの流れを対象として、格子点数を 18×18 セルとして、検証と測定を行った。CS-2 の各 PE は 1 PE あたり 48 KB のローカルメモリを持つため、今回の実装では 1PE あたり最大 18×18 の格子点数を計算できた。前節に述べたシミュレーションループを 10 周した後、分布関数を保持している配列で誤差が発生した。誤差の値を表 1 に示す。発生した誤差の最大値は 2.33e-06 であったため、許容範囲内に収まっている。

表–1 CPU 版と CSL 版で発生した誤差

配列名	不一致要素	最大絶対差
f_i	7 / 2601 (0.269%)	1.87e-06
f_i^*	14 / 2601 (0.538%)	2.33e-06

boundary 関数と streaming_collision 関数、swap 関数のサイクル数とループ全体の計算にかかったサイクル数、演算性能と MLUPS を用いた性能を表 2 に示す。得られたサイクル数から演算性能を次の式で見積もる。

$$\text{演算性能} = \frac{\text{動作周波数 [Hz/s]}}{\text{サイクル数 [cycles]}} \times \text{浮動小数点演算数 [Flops]} \quad (6)$$

CS-2 の動作周波数は 850 MHz で、シミュレーションループ内部の浮動小数点演算は 60960 回であるため、1PE の演算性能は 75.2 MFlops/s となる。なお 浮動小数点演算の回数はコード上の演算回数を手作業で集計した。MLUPS（Mega Lattice Update Per Second）は LBM の分野で一般的に利用される性能指標であり、次の式で計算される。

$$MLUPS = \frac{\text{格子点数 [個]} \times \text{時間ステップ数 [回]}}{\text{実行時間 [}sec\text{]}} \quad (7)$$

表–2 1 タイムステップでの各関数のサイクル数と演算性能

関数名	サイクル数	MFlops/s	MLUPS
boundary	16050	297.2	N/A
streaming_collision	605158	77.8	N/A
swap	67657	N/A	N/A
合計	688865	75.2	0.28

boundary や swap には、浮動小数点計算やメモリアクセスをする箇所が存在しない、または少ないため比較的小さなサイクル数となった。また、streaming_collision は格子点だけではなく LBM の計算方法上、自身を含んだ 9 方向の隣接した格子点に対しても計算が必要となるため、他の関数と比べて大きなサイクル数となった。

また、CS-2 の特徴として PE 間のデータ更新にかかるサイクル数が 1 サイクルであるため、通信オーバーヘッドを隠蔽できると仮定すれば、全 PE を使用した場合の演算性能を見積もることできる。全 PE を使用した場合の推定演算性能は 750 × 995 × 75.2 MFlops/s = 56.1 TFlops/s である。

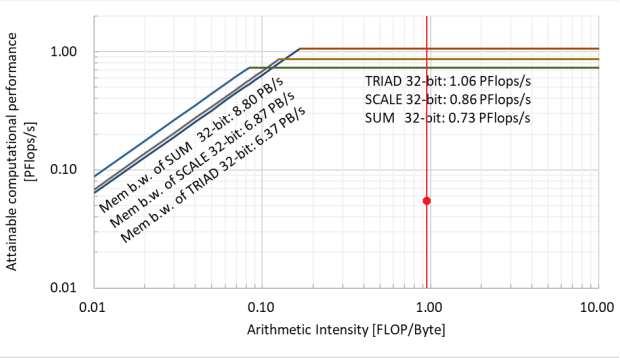
5. ルーフラインモデルを用いた計算性能評価

ここからは、先行研究 [6] で得られた STREAM ベンチマークを元にした CS-2 のルーフラインモデルを使い、1PE ナイーブ実装の性能評価を行う。

ルーフラインモデルとは、プロセッサの計算性能とオフチップメモリの帯域幅との関係を示す目的で考えられたモデルである [10]。浮動小数点性能、演算強度、およびメモリ性能を 2 次元グラフで結び付けることで、視覚的にも理解しやすい性能モデルである。演算強度は、浮動小数点計算の数である Flops とメモリアクセスにかかる Byte 数の比によって計算する。実線で示されている 3 本が、その演算強度において CS-2 が達成可能な実効計算性能の上限を示している。水平線になっている箇所はプロセッサが達成可能な最大浮動小数点性能を示しており、斜線になっている箇所はメモリ帯域幅が制約となっていることを示している。リッジポイントと呼ばれる水平線と斜線の交点より左側であれば、メモリ帯域幅がボトルネックとなって制約を受けて、右側であれば演算性能がボトルネックとなって制約を受けていることが分かる。

図 6 に CS-2 のルーフラインモデルと LBM の演算強度、(2) 節で得られた性能をプロットした結果を示す。streaming_collision の演算強度は $Flops/Bytes = 55350/57600 \approx 0.96$ であった。演算強度からは streaming_collision がメモリ帯域の制約を受けず、0.73〜1.06 PFlops/s の演算性能を達成できると考えられる。しか

© 一般社団法人日本計算工学会



図–6 CS-2 のルーフラインモデルと本実装で得られた性能

し、全 PE を使用した場合の推定演算性能は 56.1TFlop-s/s であり、上限値の 7.6% ほどしか達成できていない。性能が低い理由として、本関数が十分にベクトル化されていないことが上げられる。ルーフラインモデルの作成に用いた STREAM ベンチマークはベクトル命令を最大限利用したものである。それに対し、本関数は図 1 に示すようなナイーブ実装であるため、CS-2 の演算性能を引き出せていないと考えられる。なお、渡辺ら [11] の報告によれば、スーパーコンピュータ 富岳の 1 ノードで D3Q15 モデルで 691GFlops を達成している。モデルが異なるため直接的な比較はできないが、この性能はピーク性能 1395GFlops の 49.5%に相当し、本実装で得られた対ピーク性能比よりも大幅に高い。

6. まとめ

本研究では、数値流体力学の計算手法の 1 つである格子ボルツマン法を Cerebas CS-2 に移植・実装した。1 つの PE で正しい数値計算を行えるようなナイーブ実装では、75.2 MFlops/s、0.28MLUPS の性能となった。また、得られた性能から全 PE を用いた場合の性能を推定し、ルーフラインモデルで性能評価を行った。今後は、最適化を行ったうえで、複数 PE を用いた並列化を進めていき、理論性能とどれほどの性能差が発生するかを確認していきたい。

謝辞

本研究は、国立研究開発法人科学技術振興機構（JST）先端国際共同研究推進事業（ASPIRE）JPMJAP2341 の支援を受けたものである。本研究は JSPS 科研費 24K14972 の助成を受けたものです。

参考文献

[1] S. Matsuoka, H. Amano, K. Nakajima, K. Inoue, T. Kudoh, N. Maruyama, K. Taura, T. Iwashita, T. Katagiri, T. Hanawa and T. Endo: “From flops to bytes: Disruptive change in high-performance computing towards the post-moore era”, Proceedings of the ACM International Conference on Computing Frontiers, CF ’16, New York, NY, USA, Association for Computing Machinery, p. 274–281 (2016).

[2] M. Jacquelin, M. Araya-Polo and J. Meng: “Scalable distributed high-order stencil computations”, Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC ’22, IEEE Press (2022).

[3] M. Woo, T. Jordan, R. Schreiber, I. Sharapov, S. Muhammad, A. Koneru, M. James and D. Van Essendelft: “Disruptive changes in field equation modeling: A simple interface for wafer scale engines”, https://arxiv.org/abs/2209.13768 (2022).

[4] M. Orenes-Vera, I. Sharapov, R. Schreiber, M. Jacquelin, P. Vandermersch and S. Chetlur: “Wafer-scale fast fourier transforms”, Proceedings of the 37th ACM International Conference on Supercomputing, ICS ’23, New York, NY, USA, Association for Computing Machinery, p. 180–191 (2023).

[5] R. Sai, F. P. Hamon, J. Mellor-Crummey and M. Araya-Polo: “Matrix-free finite volume kernels on a dataflow architecture”, Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC ’24, IEEE Press (2024).

[6] 福岡伶音, 宮島敬明:“Cerebras CS-2 を用いたリダクション処理の実装と性能評価”, Technical Report 10, 第 188 回ハイパフォーマンスコンピューティング研究発表会 (2023).

[7] S. Lie: “Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning : Cerebras systems”, pp. 1–34 (2022).

[8] T. Miyajima, R. Matsuzaki and L. Fukuoka: “Stream benchmark on cerebras wse-2 (poster)”, ISC High Performance 2024 Research Paper Proceedings (39th International Conference), No. 10 (2024).

[9] N. Onodera, Y. Idomura, Y. Hasegawa, H. Nakayama, T. Shimokawabe and T. Aoki: “Real-time tracer dispersion simulations in oklahoma city using the locally mesh-refined lattice boltzmann method”, BOUNDARY-LAYER METEOROLOGY, **179**, 2, pp. 187–208 (2021).

[10] S. Williams, A. Waterman and D. Patterson: “Roofline: an insightful visual performance model for multicore architectures”, Commun. ACM, **52**, 4, p. 65–76 (2009).

[11] S. Watanabe and C. Hu: “Performance evaluation of lattice boltzmann method for fluid simulation on a64fx processor and supercomputer fugaku”, International Conference on High Performance Computing in Asia-Pacific Region, HPCAsia ’22, New York, NY, USA, Association for Computing Machinery, p. 1–9 (2022).

© 一般社団法人日本計算工学会