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

Cerebras CS-2 を用いた 疎行列ベクトル積の測定

Measuring Sparse Matrix-Vector multiplication Using Cerebras CS-2

折原冴保 [1)]　酒井宏己 [2)]　宮島敬明 [3)]
Saho Orihara and Koki Sakai and Takaaki Miyajima

[1)]明治大学 理工学部 情報科学科 （〒 214-0033 神奈川県川崎市多摩区東三田 1-1-1, E-mail: ee217043@meiji.ac.jp）
[2)]明治大学 理工学部 情報科学科 （〒 214-0033 神奈川県川崎市多摩区東三田 1-1-1, E-mail: ee217044@meiji.ac.jp）
[3)]明治大学 理工学部 情報科学科（〒 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）

With the development of the HPC field, deep learning, and fluid dynamics simulations, the need for faster execution of large-scale calculations has increased rapidly in recent years. GPUs are used to efficiently perform these calculations, but bottlenecks such as bandwidth limitations in communication performance and latency in memory access still exist. We have been experimentally validating the Cerebras CS-2, which can solve these two issues at the architectural level. In this study, we conduct a feasibility study of sparse matrix-vector multiplication, a fundamental computation kernel in scientific computing. We measured runtime breakdown of processing time and strong scaling of SpMV on CS-2.

Key Words : *Cerebras CS-2, SpMV, Parallel Computing*

1. はじめに

かつては困難とされていた高精度なシミュレーションや深層学習が、 近年の計算技術の向上によって可能になりつつある。それに伴い、基礎的な計算カーネルを高速に処理するニーズが増加している。疎行列ベクトル積（Sparse Matrix-Vector multiplication: SpMV）は、工学や自然科学をはじめとする幅広い分野で繰り返し利用される代表的な計算カーネルの一つである。SpMV の最大の特徴は行列が多数のゼロ要素を含む疎行列である点にある。そのため、計算効率の向上には非ゼロ要素の格納方法や効率的なメモリアクセスなどが必要になる。特に、行列の格納形式に由来する間接参照が計算のボトルネックとなっている。これまでにも、Graphic Processing Unit（GPU）を用いた高速化の研究が行われてきた [1][2]。高性能計算の研究分野においては、GPU や他のアクセラレータを搭載する計算ノードを数百台から数千台ほど接続するような、従来のスケールアウト型手法には限界があることが知られている。その主な要因として、ノード間通信のレイテンシが計算に比べて非常に長いことが挙げられる。

Cerebras CS-2 は、これらの問題を解決することを目的として設計された深層学習用アクセラレータである。その最大の特徴は、300mm の単一ウェハーで構成された巨大な並列プロセッサ Wafer-Scale Engine 2（WSE-2）を搭載している点である。WSE-2 には約 85 万個の処理要素（Processing Element: PE）が 2 次元メッシュ上に配置され、相互に接続されている [3]。各 PE は、演算処理用のコアと 48KB のローカル SRAM を備えた計算ユニット（Computing Engine: CE）と、通信を担うルータから成る。SRAM メモリへのアクセスおよび PE 間通信は、ともに 1 クロックサイクルという非常に低遅延で実現されている。DRAM メモリのような大容量かつ高レイテンシなメモリを持たないため、処理はすべてローカル SRAM で行われ、間接参照の様なメモリアクセスにも強いと考えられる。このように、 従来の計算機とは異なる設計思想に基づくハードウェアアーキテクチャを採用することで、通信レイテンシを大幅に削減する。その高い並列計算性能により、科学技術計算への応用も進められている [8][9]。基礎的な演算性能の解析の試みが進められている [5] が、計算カーネルの評価は十分に進んでいない。

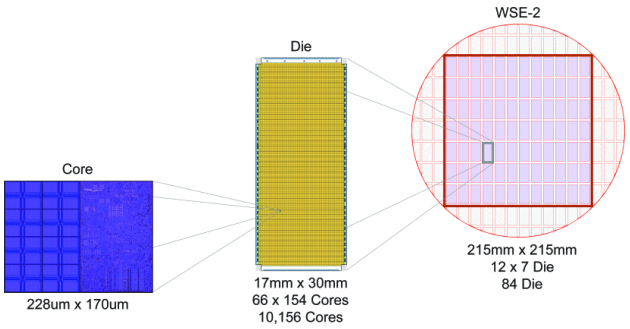
そこで本研究では、Cerebras CS-2 向けに実装された既存の SpMV プログラムの性能測定をシミュレータを用いて行う。本研究の主な貢献は以下の 2 点である。

1. Cerebras CS-2 における疎行列ベクトル積の強スケーリング性能測定
2. 総所要時間と各 PE におけるタスク別の所要時間の測定とボトルネック解析

本論文の構成は次の通りである。第 2. 節では、Cerebras CS-2 のアーキテクチャ概要について述べる。第 3. 節では、SpMV の概要とその特徴を説明する。第 4. 節では、本研究において実装した疎行列演算アルゴリズムについて詳述し、第 5. 節では提案手法の実行環境および測定結果を示し、性能評価を行う。最後の第 6. 節にて本研究の結論を述べる。

2. Cerebras CS-2 システム

本節では、WSE-2 および、PE の構成と特徴、及びプログラミングモデルについて述べる。



図–1 WSE-2 の物理的構造

(1) WSE-2

Cerebras CS-2 は、第 2 世代のウェハースケールエンジン WSE-2 を搭載している [4]。この WSE-2 は、図 1 に示すように 215mm × 215mm のウェハー上に 84 個のダイを搭載しており、各ダイには 10156 個の PE が集積されている。WSE-2 全体で合計 853104 個の PE が実装されており、これらは 2 次元メッシュ状に配置され、相互に接続されている [3]。ユーザが計算に使用可能な PE の総数は 745500 個（x 方向 750 個 × y 方向 994 個）である。各 PE は非同期で動作しており、メモリ一貫性は保証されないため、科学技術計算を行う際は分散メモリアーキテクチャとして扱われる。また、PE の動作周波数は 850MHz であり、科学技術計算の実行時の CS-2 システム全体の消費電力は 16.5kW となる。

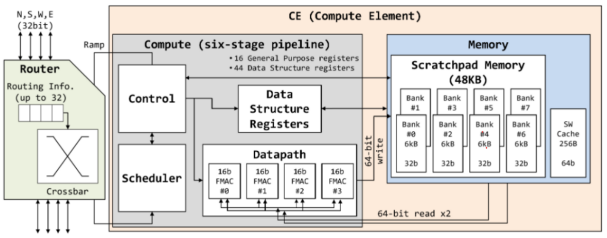
(2) プロセッシングエレメント（PE）

PE の構成を図 2 に示す。各 PE は、コンピューティングエンジン（Computing Engine: CE）と、データの送受信を担うルータから構成されている。PE1 つの回路面積は 38,000μm^2 であり、このうち半分は 48KB の SRAM メモリ、残りの半分はおよそ 110,000 個のスタンダードセルで構成された CE で占められている。SRAM メモリは、32bit 幅の 8 つのシングルポートバンクから構成されている。1 サイクルあたり 64bit の読み出しを 2 つ、書き込みを 1 つ行える。PE の動作周波数は 850MHz で、ピーク時の消費電力は 30mW に抑えられている。

ルータは、4 方向（東・西・南・北）に接続される 32bit 双方向インタフェースと、PE 内部のメモリに接続される 1 方向（Ramp）を含めた計 5 ポートで構成されている。PE 間のデータ転送に要する通信遅延は 32bit/1 サイクルで、通信用バッファの最小化、低コストかつロスの少ないフロー制御が実現されている。ルータは静的にルーティングされ、物理的に同一のリンク上でも複数のルートを有効化することが可能である。これにより、最大 24 個の独立した静的ルートを設定でき、それぞれのルートはカラーと呼ばれる。

(3) プログラミングモデル

Cerebras SDK を用いることで、WSE のマイクロアーキテクチャに対応した低レイヤのコードを記述できる。汎用的な並列計算を CS-2 で行うために、専用の開発環境として Cerebras SDK が用意されており、これは他の GPU プラットフォームにおける HPC SDK や HIP と類似した役割を果たす [5]。本 SDK を使用することで、各



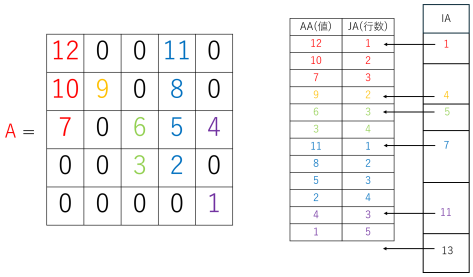
図–2 プロセッシングエレメント（PE）の構造

コアをターゲットとしてコードを記述でき、計算資源およびメモリ資源を明示的に利用できる。SDK では、CS-2 はデバイス、ホスト側の CPU クラスタはホストと呼ばれる。プログラミングモデルは、他のアクセラレータと同様のアーキテクチャを踏襲している。具体的には、ホストシステムから CS-2 に対してプログラムを送った後、実行を開始する。なお、ホストシステム側の制御プログラムは Python で記述される。デバイス側のアプリケーションは、Cerebras Software Language（CSL）と呼ばれる C 言語に類似したドメイン固有言語で記述する。CSL を用いることで、各 PE に同一のタスクを割り当てるだけでなく、PE ごとに異なる処理も実行できる。本 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。

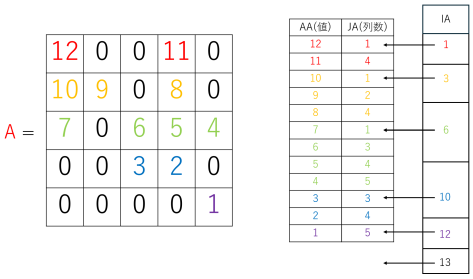
3. 疎行列ベクトル積

多数のゼロ要素を含む行列（疎行列）とベクトルを掛け合わせ、新たなベクトルを生成する演算を疎行列ベクトル積（Sparse Matrix-Vector Multiplication: SpMV）と呼ぶ。SpMV は一般に $y = \mathbf{A}x$ で表される。ここで、$\mathbf{A}$ は疎行列、x は入力ベクトル、y は出力ベクトルである。工学や自然科学をはじめとする幅広い分野で利用される、代表的な計算カーネルの一つである。

疎行列の非ゼロ要素数は、行列全体の 1%程度となる場合もあり、行列をそのまま密行列として格納するとメモリ使用率と計算効率が大幅に低下してしまう。そのため、CSC（Compressed Sparse Column）形式や CSR（Compressed Sparse Row）形式と呼ばれる疎行列の格納形式が用いられる。CSC 形式は図 3 に示すとおり、行列の列方向に注目して格納する形式であり、以下の 3 つの配列によって構成される。(1) 非ゼロ要素の値を格納する配列、(2) 各非ゼロ要素が行列のどの行に位置するかを示す行インデックス配列、(3) 各列における最初の非ゼロ要素が、非ゼロ要素全体の中で何番目に現れるかを示すポインタ配列である。また、CSR 形式は図 4 に示すとおり、行列の行方向に注目して格納する形式であり、以下の 3 つの配列によって構成される。(1) 非ゼロ要素の値を格納する配列、(2) 各非ゼロ要素が属する列を示す列インデックス配列、(3) 各行における最初の非ゼロ要素が、非ゼロ要素全体の中で何番目に現れるかを示すポインタ配列である。これらの形式を用いることでデータを圧縮して格納できるため、メモリ使用量を大幅に低減できるが、間接参照を含んだ非効率なメモリアクセスとなってしまう。



図–3 CSC（Compressed Sparse Column）形式



図–4 CSR（Compressed Sparse Row）形式

SpMV の高速化に関する先行研究は多数存在している。たとえば、先行研究 [6] では、メモリトレース情報と実行命令数を用いて、任意のメモリアーキテクチャ上での実行時間を推定するツール”PMNet”を提案している。また、先行研究 [7] では、115 種類の疎行列を GPU 上で実行し、SpMV の性能モデルの比較を行っている。その結果、GPU による実行においては Padding の補正や、小規模行列のデータ保管形式が計算時間を左右すること、また適切な負荷分散が実現された場合には、キャッシュヒット率よりも非ゼロ要素の空間的局所性やインデックス転送の抑制が実行性能に大きな影響を与えることが示されている。

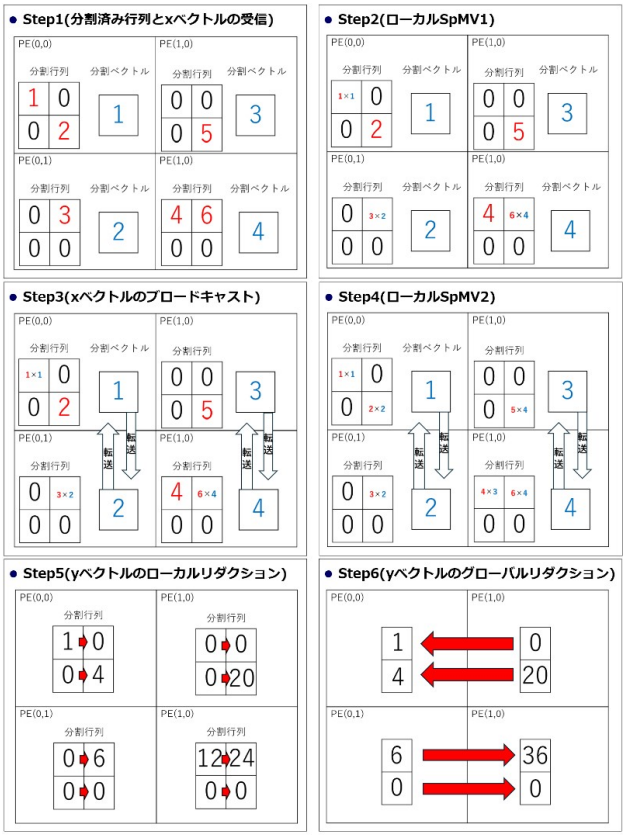
4. 疎行列ベクトル積の分散並列処理アルゴリズム

本節では、Cerebras SDK がサンプルプログラムとして提供している SpMV の分散並列処理プログラム”Hypersparse SpMV”[11] について述べる。なお、本プログラムの分散並列処理アルゴリズムは、[10] を元にしていると思われる。このプログラムでは、ホスト側とデバイス側に機能を分離して動作する構成となっている。ホスト側では、疎行列およびベクトル分割を行った後に、分割されたデータをデバイス側へ送信する。デバイス側では、ホスト側から受信したデータをもとに計算し、算出された結果をホスト側へ返す。本節では、Hypersparse SpMV の全体的な処理の流れを、ホスト側アルゴリズムおよびデバイス側アルゴリズムの 2 つに分けて記す。

(1) ホスト側アルゴリズム

ホスト側では元の疎行列 **A** とベクトル x を分割しデバイス側へ送る。処理の流れは以下の通りである。

1. **疎行列 A と x ベクトルの分割**：デバイスに送信するためのデータを準備として、**A** および x を矩形に分割する。分割方法は使用する PE 数で **A** を分割する単純なもので、非ゼロ要素の位置や負荷分散などは考慮していない。また、**A** は CSC 形式で格納し南北方向への通信に適した形としている。
2. **比較用結果の作成**：デバイス側の計算結果が正確な値であるかを判定するために、比較用の基準値をホスト側であらかじめ作成する。この基準値は、最終的な計算結果の検証時に参照される。
3. **分割データの送信**：分割済みの行列および x ベクトルを、ホスト側からデバイス側へ送信する。この処理により、デバイス側での並列計算に必要な入力データがすべて揃う。
4. **計算開始の指示**：ホスト側からデバイス側に対して、SpMV の計算を開始する。この命令により、デバイス側での並列計算が実行される。
5. **計算結果の受信**：デバイス側で算出された計算結果を受信する。
6. **計算結果の確認**：受信した計算結果とホスト側での計算結果と比較する。



図–5 疎行列ベクトル積の分散並列処理アルゴリズム

(2) デバイス側アルゴリズム

デバイス側では、ホスト側から送信された分割済みの疎行列および x ベクトルを用いて、SpMV を並列処理する。以下に、詳細な処理の流れをステップに分けて示す。また、本処理の概要を図 5 に示す。

1. **分割済み行列と x ベクトルの受信**: ホスト側で分割済み疎行列と x ベクトルを、各 PE がホストから受信する。
2. **ローカル SpMV 1**: 各 PE は受信した疎行列と x ベクトルに対し、ローカルな SpMV を行う。
3. **x ベクトルのブロードキャスト**: 各 PE は自身が保持する分割済みベクトルを南北方向の全 PE にブ

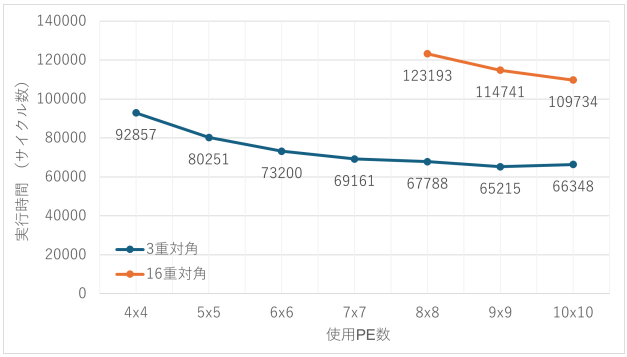
ロードキャストする。

4. **ローカル SpMV 2**: 他の PE から受信した分割済み x ベクトルを用いて、各 PE がローカル SpMV の続きを実行する。
5. **y ベクトルのローカルリダクション**: 全てのローカル SpMV の完了後、各 PE は自身の部分ベクトル y に対してリダクション処理を行う。
6. **y ベクトルのグローバルリダクション**: 部分リダクションで得られた結果を東西方向に送信しながら、各 PE で加算処理を行い、結果を集約する。

(3) CSL による実装：Hypersparse SpMV プログラム

Hypersparse SpMV は以下のタスク関数を組み合わせて (2) 節で述べたアルゴリズムを実現している。タスクは主にデータ送受信タスクと計算タスクに分類される。以下にタスク名とその役割を示す。タスク名の先頭に付された番号は、(2) 節の各ステップと対応している。

2-a. **ローカル計算（compute_local）**: 各 PE がホスト側から受け取った分割行列及び、分割ベクトルを用いて計算を行うタスク。
3-a. **北方向送信（tx_north）**：各 PE が保持する分割ベクトルを北方向に送信するタスク。
3-b. **南方向送信（tx_south）**: 各 PE が保持する分割ベクトルを南方向に送信するタスク。
3-c. **北方向受信（rx_north）**: 北側 PE から分割ベクトルを受信するタスク。
3-d. **南方向受信（rx_south）**: 南側 PE から分割ベクトルを受信するタスク。
4-a. **北からのベクトル計算（compute_south）**: 各 PE が自分が保持する分割行列と、自分より北側に隣接する PE から受け取った分割ベクトルを用いて計算を行うタスク。
4-b. **南からのベクトル計算（compute_north）**: 各 PE が自分が保持する分割行列と、自身より南側に位置する PE から受信した分割ベクトルを用いて計算を行うタスク。
5-a. **ローカルリダクション（start_reduce）**: 各 PE 内部での計算結果を集約するリダクションタスク。
6-a. **東方向送信（tx_east）**: 集約された部分計算結果を東方向に送信するタスク。
6-b. **西方向送信（tx_west）**: 集約された部分計算結果を西方向に送信するタスク。
6-c. **東方向受信（rx_east）**: 東側 PE から部分計算結果を受信するタスク。本タスクには、集約処理を担う reduce_local_west というタスクが含まれている。本研究では、この演算部分を除外することで、通信処理のみに要する時間を算出した。
6-d. **西方向受信（rx_west）**: 西側 PE から部分計算結果を受信するタスク。本タスクには、集約処理を担う reduce_local_east というタスクが含まれている。本研究では、この演算部分を除外することで、通信処理のみに要する時間を算出した。
6-e. **西リダクション（reduce_local_west）**: 同じ行にある PE の部分計算結果を加算して西方向に集約するタスク。
6-f. **東リダクション（reduce_local_east）**: 同じ行にある PE の部分計算結果を加算して東方向に集約するタスク。



図–6 3 重対角行列と 16 重対角行列における強スケーリング

5. 疎行列ベクトル積の性能測定

本節では、シミュレータを用いた SpMV の性能測定として、各タスクの所要時間の内訳と強スケーリングの性能について述べる。使用したソフトウェア環境は Cerebras SDK v1.2.0、時間測定には SDK が提供するサイクルカウンタを利用した。SpMV はあるアプリケーションの計算の一部として利用されることが一般的であるため、所要時間の測定は、デバイス側のみを対象とし、ホスト側での行列とベクトルの分割とそれらの転送処理は測定の対象外としている。測定に用いた行列の特性を表 1 に示す。どちらの行列もサイズは 1000 × 1000 要素とした。

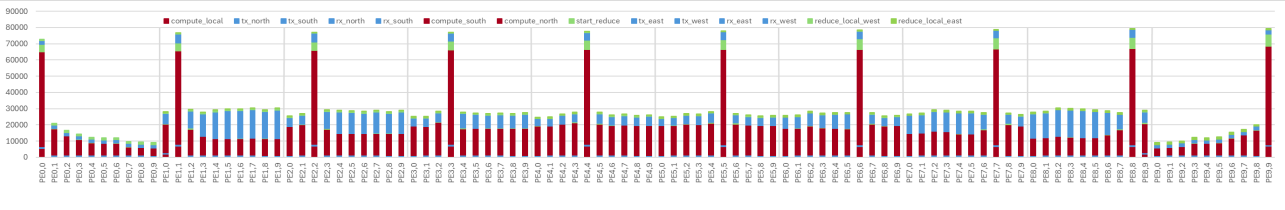
表–1 測定に用いた行列の特性

行列	非ゼロ要素数	スパース性
16 重対角行列	16928	1.69%
3 重対角行列	3996	0.40%

(1) 強スケーリング

強スケーリングの評価には、表 1 に示す 3 重対角行列および 16 重対角行列を用いた。3 重対角行列については、使用する PE 数を 4 × 4 から 10 × 10 まで 1 ずつ増やして測定を行った。一方、16 重対角行列については、メモリ容量の制約により、8 × 8 から 10 × 10 までの PE 数で測定を行った。なお、各 PE において処理の開始および終了時刻が異なるため、全 PE のうち最も早く処理を開始した PE の開始サイクル数を全体の開始時刻とし、最も遅く処理を終了した PE の終了サイクル数を全体の終了時刻と定義した。その差分を全体の並列処理の所要時間とした。

図 6 に 3 重と 16 重の対角行列の強スケーリングの測定結果を示す。3 重対角行列では、9×9 PE までは徐々に計算時間が減少しているが、10×10 PE では計算時間が減少していない。16 重対角行列では、測定点が少ないが、PE 数の増加につれて計算時間も減少している。また、表 2 に、16 重対角行列での計算タスクと通信タスクの全所要時間に占める割合を示す。PE 数の増加によって通信の所要時間が増加する一方で、計算時間の減少量がそれを上回っていることが、全所要時間の減少の要因と考えられる。



図–7 各 PE における全タスクの所要時間

表–2 PE 構成ごとの通信・計算割合

PE 数	通信割合 (%)	計算割合 (%)
8×8	3.85	73.85
10×10	5.25	67.08

(2) 各タスクの所要時間

本測定では、データ送受信タスクと計算タスクの 2 つの処理に焦点を当て、それぞれの所要時間の評価を行った。評価には、表 1 に示す 16 重対角行列のみを使用し、PE 数は 10 × 10 とした。図 7 に 10×10 PE 使用時の各 PE のタスク所要時間を示す。計算系タスクが赤色、通信系タスクが青色、計算系タスクの中でも集約処理を担うものは緑色となっている。また、図 8〜21 に、各タスクの各 PE における所要時間を二次元状にしたヒートマップとして示す。所要時間が小さい場合は薄い黄色、大きい場合は緑色にセルを塗っている。図 9、10、11、12 から南北方向へのデータ送信の時間は固定であるが、受信は送信元からの距離に比例して所要時間が長くなっていることがわかる。また、図 16、17、18、19 より東西方向へのデータ送受信にも同様の傾向が見られる。図 15 からローカルリダクションは集約すべき値を多く持っている対角線上の PE が所要時間が長いことがわかる。図 16、17、18、19、20、21 からリダクションは東西方向への通信と計算を伴うため、最終的に値が集約される東側と西側の PE が所要時間が長いことがわかる。これらの結果はアルゴリズムから予想できるものであるが、本プログラムが正しく動いていることを示している。

図 7 の各タスクの所要時間の測定により、計算タスクに分類される compute_local、compute_south、compute_north が総所要時間に対して支配的であることがわかる。各タスクを詳細に見ると、図 8、13、14 より、ボトルネックとなっている計算タスクでは、対角線上の PE に負荷が集中している事が判明した。これは、本アルゴリズムが行列を分割する際に、非ゼロ要素の位置や負荷分散などを考慮しておらず、元の行列 (表 1) の非ゼロ位置を担当する対角線上の PE に計算負荷が偏ったためである。

6. 結論

本研究では、Cerebras CS-2 上での疎行列ベクトル積プログラム”Hypersparse SpMV”の性能をシミュレータを用いて測定、解析した。評価項目として、2 種類の対角行列を対象に 10×10PE 程度を利用した際の強スケーリングと、その時の総所要時間と各 PE におけるタスク別の所要時間の測定とボトルネック解析を行った。その結果、今回の 3 重対角行列では 9×9 PE までは徐々に計算時間が減少しているが、10×10 PE では計算時間が減少しなかった。また、各タスクの所要時間の測定結果から、行列の分割時に負荷分散が考慮されていないため、元の行列の非ゼロ位置を担当する対角線上の PE に計算負荷が偏っていることがわかった。

5581	1938	213	213	213	213	213	213	213	213
237	6874	210	210	210	210	210	210	210	210
211	237	6877	213	213	213	213	213	213	213
210	213	237	6877	213	213	213	213	213	213
210	208	208	237	6877	213	213	213	213	216
205	208	208	208	208	6638	208	208	208	208
205	208	208	208	208	208	6638	208	208	208
205	208	208	211	211	208	208	6638	208	208
208	205	205	205	205	205	205	205	6635	205
205	205	205	205	205	205	205	205	1854	5375

図–8 ローカル計算の各 PE の所要時間

0	0	0	0	0	0	0	0	0	0
200	200	200	200	200	200	200	200	200	200
206	209	209	209	209	209	209	209	209	209
208	206	209	209	209	209	209	209	209	209
208	208	206	209	209	209	209	209	209	209
219	208	208	206	209	209	206	209	209	209
209	209	208	211	206	209	209	206	209	209
209	209	208	207	206	206	209	209	206	209
209	209	209	209	208	208	206	209	209	206
212	212	212	212	212	212	212	212	212	212

図–9 北方向送信の各 PE の所要時間

192	192	199	199	199	199	199	199	192	192
206	189	189	189	189	189	191	188	189	189
209	207	209	209	209	209	206	208	208	208
209	209	206	209	209	209	209	206	208	208
209	209	209	206	209	209	209	209	206	208
210	209	209	209	209	209	209	209	209	206
209	209	209	209	209	209	209	209	209	209
209	209	209	209	209	209	209	209	209	209
200	180	180	180	180	180	180	180	180	180
0	0	0	0	0	0	0	0	0	0

図–10 南方向送信の各 PE の所要時間

0	0	0	0	0	0	0	0	0	0
81	80	71	71	71	71	71	71	71	71
160	160	178	165	160	167	160	160	160	160
241	241	241	268	246	246	257	241	241	242
320	320	320	320	356	325	332	345	320	320
401	401	401	401	401	446	409	420	435	401
480	480	480	480	480	483	534	491	506	509
561	561	561	561	561	564	567	624	575	590
640	560	560	560	560	563	566	569	632	581
721	721	721	721	721	724	727	730	736	802

図–11 北方向受信の各 PE の所要時間

802	763	756	749	742	735	728	721	721	721
703	712	675	668	661	654	647	640	640	640
561	615	624	589	587	575	568	561	561	561
480	480	525	534	501	494	487	480	480	480
401	401	401	437	446	415	408	401	401	401
320	320	320	320	347	356	327	320	320	329
241	241	241	241	241	259	268	241	241	241
160	160	160	160	160	160	169	178	160	160
81	81	81	81	81	81	81	81	90	81
0	0	0	0	0	0	0	0	0	0

図–12 南方向受信の各 PE の所要時間

0	0	0	0	0	0	0	0	0	0
243	5587	1940	221	222	219	219	219	219	219
1152	3236	12469	3263	1760	1760	1152	987	987	877
1426	4231	5019	19352	5022	3522	3520	2536	2526	3958
1754	4514	6494	6801	26234	6835	5280	5333	4339	4890
2193	5493	8696	11774	11722	33117	8649	6269	5553	5718
3356	7591	10729	13923	15556	13789	39999	10192	7698	8922
4135	9350	12475	15665	17432	17070	15724	46882	12056	11068
4417	10118	13270	16444	18088	18053	17881	17138	53764	15129
4636	10337	13489	16747	18435	18419	18263	18103	18036	60647

図–13 北からのベクトル計算の各 PE の所要時間

58419	17380	17720	17995	18164	18338	16558	13547	10629	4591
15743	51896	16745	17411	17621	18074	16346	13396	10461	4376
10647	12668	45343	16786	17217	17351	16860	13749	10480	4396
8194	7336	10761	38685	14822	15201	13331	11791	8528	3374
5912	5612	6718	9298	32028	12737	11182	7772	6319	2264
5064	4640	4568	4692	7276	25370	7430	6738	5145	1902
4031	2615	2605	2605	2729	5243	18713	5463	4794	1551
951	961	951	951	951	1323	3279	12055	3493	1261
228	215	215	215	215	215	215	1861	5385	215
0	0	0	0	0	0	0	0	0	0

図–14 南からのベクトル計算の各 PE の所要時間

4678	496	109	109	109	109	109	109	109	117
443	4974	496	109	109	109	109	109	109	117
110	731	5275	487	100	100	100	100	100	108
110	100	731	5585	487	100	100	100	100	108
110	100	100	731	5895	487	100	100	100	108
110	100	100	100	731	6205	487	100	100	108
110	100	100	100	100	731	6515	487	100	108
110	100	100	100	100	100	731	6825	487	108
104	109	109	109	109	109	109	740	7144	456
100	96	96	96	96	96	96	96	727	7445

図–15 ローカルリダクションの各 PE の所要時間

376	1074	1036	1181	1389	1483	4294	8577	12758	0
183	680	1359	1375	1466	1711	4382	8676	12860	0
186	364	877	1634	1840	2272	4901	9313	13864	0
186	364	547	1038	1843	2289	4985	9421	13941	0
186	364	547	728	1164	2921	5629	9963	14245	0
186	364	547	728	911	1362	6058	10141	13439	0
186	364	547	728	911	1092	1525	6500	11084	0
186	364	547	728	912	1164	1420	1686	6740	0
183	342	514	684	856	1028	1330	1964	1870	0
183	364	547	728	911	1095	1420	2104	2791	0

図–16 東方向送信の各 PE の所要時間

0	3394	2556	1790	1093	904	684	512	342	170
0	2857	2358	1658	1026	855	684	512	342	170
0	8099	2257	1850	1296	910	729	545	364	181
0	11112	7845	1517	1236	910	728	545	364	181
0	13919	10626	6016	1356	909	728	545	364	181
0	14660	10183	5595	2774	1193	728	545	364	181
0	14512	9758	5167	2345	1890	1032	545	364	181
0	15019	10265	5387	2730	2038	1873	755	364	181
0	14217	9790	5219	2393	1775	1649	1539	674	170
0	15143	10372	5530	2584	1822	1801	1716	1550	391

図–17 西方向送信の各 PE の所要時間

1974	1680	1467	1260	1048	832	627	420	207	0
1931	1680	1467	1261	1047	841	627	420	207	0
1829	1677	1653	1419	1180	947	707	473	234	0
1815	1876	1438	1419	1180	946	707	473	234	0
1815	1842	1632	1204	1180	946	707	473	234	0
1815	1842	1632	1369	965	946	707	473	234	0
1815	1842	1603	1369	1130	731	707	473	234	0
1815	1842	1611	1369	1130	897	630	473	234	0
1811	1565	1345	1138	925	791	505	169	207	0
1773	1861	1640	1408	1130	910	664	439	197	0

図–18 東方向受信の各 PE の所要時間

0	166	346	633	769	987	1192	1439	1641	1849
0	240	284	707	941	1180	1400	1640	1881	1849
0	243	480	538	1023	1207	1430	1673	1910	1888
0	243	480	723	775	1305	1430	1680	1910	1888
0	243	480	723	960	1067	1430	1673	1917	1888
0	243	480	723	960	1203	1255	1702	1910	1888
0	243	480	723	960	1204	1440	1498	1929	1888
0	243	480	723	960	1203	1440	1683	1735	1888
0	240	474	714	948	1188	1422	1663	1896	1965
0	240	474	714	948	1188	1492	1662	1896	2223

図–19 西方向受信の各 PE の所要時間

677	600	527	450	377	308	227	150	77	0
990	600	527	450	377	300	227	150	77	0
1327	933	467	399	334	266	201	133	68	0
1327	962	868	399	334	266	201	133	68	0
1327	962	897	800	334	266	201	133	68	0
1327	962	897	829	735	266	201	133	68	0
1327	962	897	829	764	667	201	133	68	0
1327	962	897	829	764	696	602	133	68	0
1304	1030	957	880	807	730	657	551	77	0
1354	962	897	829	764	696	631	563	469	0

図–20 西リダクションの各 PE の所要時間

0	480	594	670	750	826	906	982	1062	1385
0	73	553	661	738	811	888	961	1038	1385
0	73	150	626	738	811	888	961	1038	1385
0	73	150	223	703	811	888	961	1038	1385
0	73	150	223	300	776	888	961	1038	1385
0	73	150	223	300	373	853	961	1038	1385
0	73	150	223	300	373	450	926	1038	1385
0	73	150	223	300	373	450	523	1003	1385
0	73	150	223	300	373	450	523	600	1065
0	73	150	223	300	373	450	523	600	673

図–21 東リダクションの各 PE の所要時間

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] Changwan Hong, Aravind Sukumaran-Rajam, Bortik Bandyopadhyay, Jinsung Kim, Süreyya Emre Kurt, Israt Nisa, Shivani Sabhlok, Ümit V. Çatalyürek, Srinivasan Parthasarathy, P. Sadayappan: Efficient Sparse-Matrix Multi-Vector Product on GPUs, Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '18), pp.66–79, 2018.

[2] Genshen Chu, Yuanjie He, Lingyu Dong, Zhezhao Ding, Dandan Chen, He Bai, Xuesong Wang, Changjun Hu: Efficient Algorithm Design of Optimizing SpMV on GPU, Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing (HPDC '23), Orlando, FL, USA. ACM, New York, NY, USA, 2023. DOI: https://doi.org/10.1145/3588195.3593002

[3] Cerebras Systems: "Computing with Cerebras," SDK Documentation, accessed January 28, 2025. https://sdk.cerebras.net/computing-with-cerebras

[4] Sean Lie. 2023. Cerebras Architecture Deep Dive: First Look Inside the Hardware/Software Co-Design for Deep Learning. IEEE Micro 43, 3 (May-June 2023), 18–30. https://doi.org/10.1109/MM.2023.3256384

[5] 福岡伶音，宮島敬明: Cerebras CS-2 におけるプロセッシングエレメント間の通信性能評価，HPC 研究会資料，No.197，2024 年.

[6] 萩原汐，児玉宏喜，吉川隆英，幸朋矢，遠藤敏夫: 疎行列演算高速化のためのメモリアーキテクチャ探索，情報処理学会研究報告，2022 年 3 月 3 日. https://ipsj.ixsq.nii.ac.jp/records/217248

[7] 田邊昇，冨森苑子，高田雅美，城和貴: 疎行列ベクトル積性能を決める諸要因，情報処理学会研究報告，2014 年 2 月 24 日. https://ipsj.ixsq.nii.ac.jp/records/98693

[8] 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).

[9] 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).

[10] Erik G. Boman, Karen D. Devine, and Sivasankaran Rajamanickam. 2013. Scalable matrix computations on large scale-free graphs using 2D graph partitioning. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13). Association for Computing Machinery, New York, NY, USA, Article 50, 1–12. https://doi.org/10.1145/2503210.2503293

[11] Hypersparse SpMV - SDK Documentation (1.3.0), https://sdk.cerebras.net/csl/code-examples/benchmark-hypersparse-spmv, Last accessed: 2025 年 4 月 2 日