CGLAアクセラレータにおけるレーン数に応じた演算完了の待機方法によるLLM推論の高速化

浦山 健太郎[1,a)]　中井 平蔵[1,b)]　宗片 吉史[1,c)]　安藤 拓翔[1,d)]　中島 康彦[1,e)]

概要：AIの利用拡大に伴い，計算処理の高速化とエネルギー効率の向上が求められている．粗粒度線形アレイ（CGLA）に基づくアクセラレータIMAX3では，並列に演算を実行できる処理系であるレーンを複数使用することで高速化が期待できる．しかし，実行を制御するCPUの処理が，全体性能を制約する可能性がある．本研究では，2コアを持つCPUで最大8レーンを制御するシステムを対象に，演算の完了待ちに伴うCPUの使用に着目した．各レーンに1つのスレッドを割り当て，複数レーン構成でスレッドを休止させる条件を変更し，使用レーン数と適切な休止条件の関係を調査した．その結果，測定した条件の中で実行時間が最短となる休止条件はレーン数によって異なり，従来条件に対して4レーンで7.54%，8レーンで10.91%のLLM推論実行時間の短縮した．

Reducing LLM Inference Time on a CGLA Accelerator by Selecting the Waiting Method for Computation Completion According to the Number of Lanes

Kentaro Urayama[1,a)]　Heizo Nakai[1,b)]　Yoshifumi Munakata[1,c)]　Takuto Ando[1,d)]
Yasuhiko Nakashima[1,e)]

Abstract: With the growing use of AI, faster and more energy-efficient computation is required. IMAX3, an accelerator based on a coarse-grained linear array (CGLA), executes computations in parallel on multiple lanes, but CPU processing for controlling the lanes can limit overall performance. On an IMAX3 system in which a dual-core CPU controls up to eight lanes with one thread per lane, we vary only the threshold that determines whether a thread sleeps before polling for computation completion. In LLM inference with two, four, and eight lanes, the threshold yielding the shortest execution time differed with the number of lanes, and selecting a suitable threshold reduced the execution time by 7.54% with four lanes and 10.91% with eight lanes compared with the original threshold.

1. はじめに

LLMをはじめとするAIの利用拡大に伴い，データセンターの電力需要が増大している．世界におけるデータセンターの電力消費量は，2025年の約485 TWhから2030年には約950 TWhへ増加すると予測されている[1]．このため，増大する計算需要に対応するには，必要な処理性能を維持しながら，計算に要するエネルギーを削減することが重要となる．

こうした電力需要の増大に対応するため，計算処理のエネルギー効率を高めるアーキテクチャの研究が進められている．その一つとして，粗粒度再構成可能アレイ（CGRA）がある．一般的なCPUでは，演算の実行に加えて命令の取得や解読などの処理に電力を消費する．一方，CGRAでは，演算を複数の演算器（PE）に割り当て，演算器間の接続を利用してデータフローとして実行する．これにより，命令ごとのフェッチやデコードに伴う処理を削減するとと

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology, Nara, Japan
[a)] kenta.axis@gmail.com
[b)] nakai.heizo.nk9@naist.ac.jp
[c)] munakata.yoshifumi.nz5@naist.ac.jp
[d)] ando.takuto.aq5@naist.ac.jp
[e)] nakashim@is.naist.jp

ⓒ2026 Information Processing Society of Japan

もに，演算結果を PE 間で直接受け渡すことでデータ移動に伴うエネルギー消費を抑えられる [2].

CGRA の一種である粗粒度線形アレイ（CGLA）に基づく IMAX（In-Memory Accelerator eXtension）は，PE とローカルメモリモジュール（LMM）を線形に配置し，データの再利用と連続的なパイプライン実行によって効率的な演算を実現する [3], [4]．LLM 推論を対象とした既存研究では，ASIC 化を想定した電力と性能の見積りにより，GPU と比較して高いエネルギー効率を実現できる可能性が示されている [5]．IMAX では，演算器とローカルメモリから構成され，それぞれが並列に演算を実行できる処理系をレーンと呼ぶ．レーンは CPU におけるコアのような単位であり，レーン数を増やして演算を分担させることで，アクセラレータの演算時間の短縮が期待できる．ただし，CPU が各レーンへデータを受け渡して演算を起動するため，レーン数を増やすと CPU の処理も増える．

本研究で対象とする IMAX3 は，2 コアを持つ CPU で最大 8 レーンを制御するシステムである．各レーンに 1 つのスレッドを割り当てるため，8 レーン実行時には 8 つのスレッドが 2 つの CPU コアを共有する．しかし，既存評価では 2 レーンを超える構成で LLM 推論全体の性能が低下し，CPU によるデータ転送処理や実行制御が並列化の効果を制限することが報告されている [5]．そこで本研究では，アクセラレータの演算完了を待つ間の CPU 使用に着目し，スレッドを休止させる条件が全体の性能に与える影響を使用レーン数ごとに評価することで，多レーン実行時の性能向上を目指す．この課題は，今後レーン数を増やす際の性能拡張性を確保する上でも重要である．

IMAX では，CPU が状態レジスタを繰り返し読み取るポーリングによって，演算や DMA 転送の完了を確認する．ポーリングは完了を速やかに検出できる一方，待機中も CPU を使用する．これに対し，指定した時間だけスレッドを休止させる関数である usleep を用いると，待機中の CPU を他のスレッドの処理に割り当てられるが，休止と再開には時間を要する．従来の実装では，演算時間の見積値が一定の閾値を超える場合にスレッドを休止させ，その後にポーリングを行う．使用レーン数に応じた条件の妥当性を検討するため，本研究では演算時間の見積方法と休止させる時間の求め方を維持し，休止するかどうかを判断する閾値のみを変更する．本研究の貢献は以下のとおりである．

- 多レーン実行時に CPU が演算結果をコピーする区間の長時間化を観測し，待機中の CPU 使用と共有資源の競合という観点から性能問題を整理した．
- 従来条件に対して 4 レーンで 7.54%, 8 レーンで 10.91% の実行時間短縮を観測し，測定範囲内で最短となる休止条件が使用レーン数によって異なることを示した．

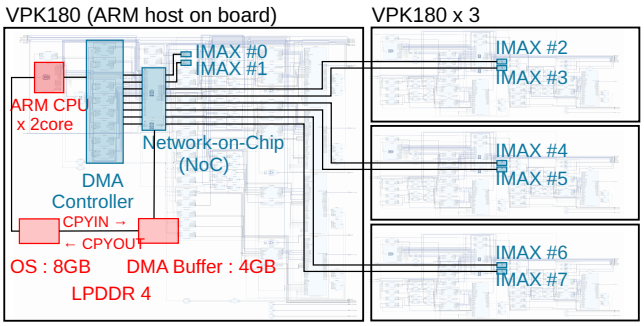

図 1　IMAX3 のハードウェア構成

2. 対象システムと演算の完了を待つ仕組み

2.1 IMAX3 と LLM 推論の処理

図 1 に IMAX3 の構成を示す．4 枚の VPK180 に各 2 レーンを実装し，マスターボード上の 2 コアを持つ CPU から合計 8 レーンを制御する．各レーンでは，PE に設定した演算をパイプラインとして連続的に実行する．LMM は演算データや中間結果を保持し，ソフトウェアが配置を管理する．

図 2 の右側に，IMAX3 で 1 回の演算を実行する際の処理の流れを示す．LLM 推論では，各層の重み行列と入力ベクトルの積を IMAX で実行するたびに，この流れを繰り返す．まず，CPU が，演算に用いる入力データを主記憶上の DMA バッファへコピーする．本稿では，この処理を CPYIN と呼ぶ．また，CPU は，演算結果を受け取る DMA バッファ上の領域をゼロで初期化する．この処理を BZERO と呼ぶ．次に，DMA コントローラが入力データを DMA バッファから LMM へ転送し，IMAX が演算を行う．演算が終わると，DMA コントローラが演算結果を LMM から DMA バッファへ転送する．最後に，CPU が演算結果を DMA バッファからアプリケーションのメモリ領域へコピーする．本稿では，この処理を CPYOUT と呼ぶ．CPU は，これらのコピーと初期化に加えて，DMA 転送と演算の起動，および状態レジスタによる完了の確認を担当する．

2.2 ポーリングとスレッドの休止

図 2 の左側は，完了割り込みを使う一般的なブロッキング I/O の一例である．OS が要求元スレッドを休止させ，デバイスの完了通知を受けて実行可能状態へ戻す．これに対し，IMAX3 では CPU が状態レジスタをポーリングして完了を確認する．usleep は，呼び出したスレッドを指定した時間だけ休止させる関数であり，ポーリングの前に呼び出すと，CPU が状態レジスタを読み続けることを防ぐことができる．これにより，CPU を実行可能な別のスレッドに割り当てられるとともに，状態レジスタへのアクセスも減る．待機中の消費電力を減らせる可能性もあるが，本稿

ⓒ2026 Information Processing Society of Japan

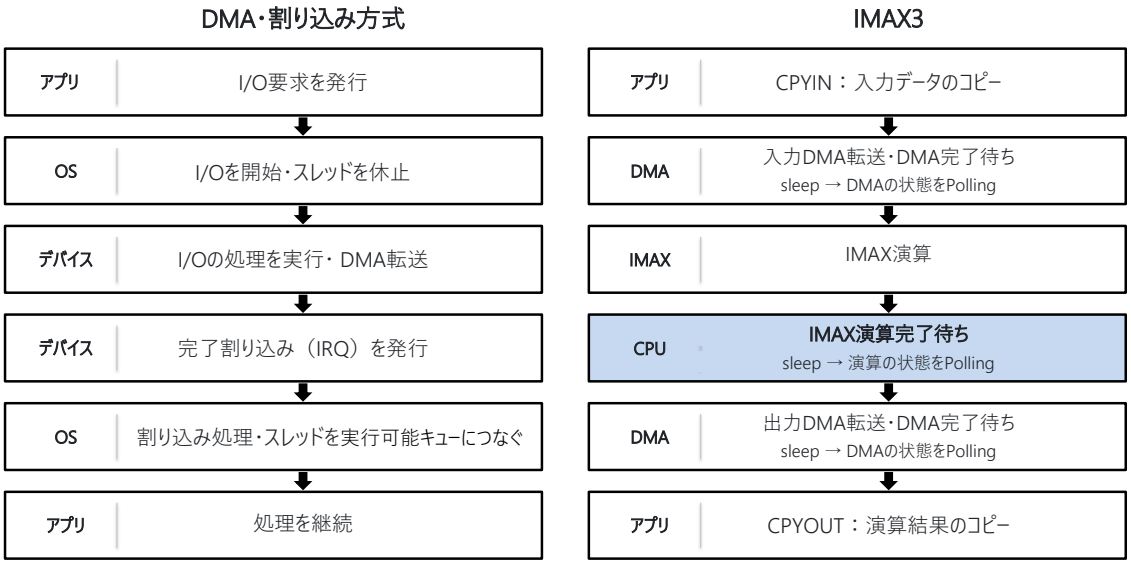

図 2　一般的な割り込み方式と IMAX3 の処理順序

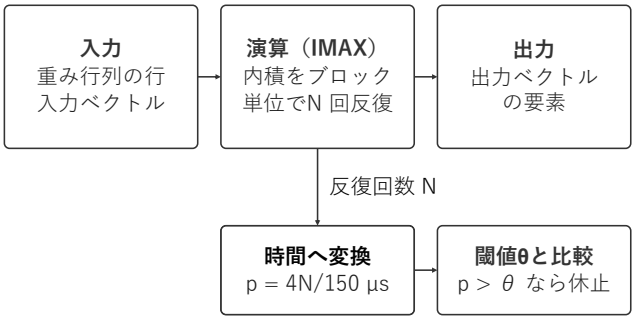

図 3　演算ループ回数の求め方と時間への変換

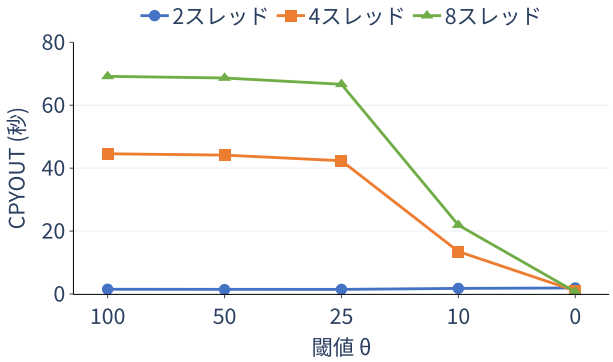

図 4　閾値 θ を変更したときのスレッド数と CPYOUT の関係

では消費電力を測定していない．また，要求した時間と，呼出しから実際に復帰するまでの時間は同じではなく，呼出し処理，スケジューリングなどの影響を受ける [6]．したがって，休止による CPU 使用の削減と，休止・再開や完了検出の遅れによる負荷の両方を考慮する必要がある．本研究では，演算を起動した後に完了を待つスレッドを休止させる条件を変更する．

2.3　演算ループ回数と休止判定

対象とする演算は，Q2_K，Q3_K，Q6_K の形式で量子化された重み行列と入力ベクトルの積である．1 回の演算起動では，重み行列のうち担当する行と入力ベクトルが入力となる．IMAX は，各行と入力ベクトルの内積を量子化ブロック単位で繰り返し計算し，その結果を出力ベクトルの担当要素として出力する．この繰返しの回数を演算ループ回数 N と呼ぶ．N は，内積方向のブロック数と担当する行数から，演算を起動する前に求められる．

休止するかどうかの判定では，まず N を時間に換算する．従来の実装は，1 回の反復に 4 クロックを要し，動作周波数が 150 MHz であると仮定して，演算時間の見積値 p と，usleep で休止させる時間 s を

$$p = \frac{4N}{150}\ \mu\mathrm{s}, \qquad s = \frac{p}{2}\ \mu\mathrm{s} \tag{1}$$

として求める．次に，p を閾値 θ と比較する．p が θ を超える場合は，演算に時間がかかるとみなし，usleep で s だけの休止を要求してから演算の完了をポーリングする．p が θ 以下の場合は，演算がすぐに終わるとみなし，ポーリングする．したがって，θ を小さくするほど休止する演算が増え，$\theta = 0$ では p が正となるすべての演算で休止する．θ は見積値 p と比較する値であり，休止時間そのものではない．入力から時間への変換までの流れを図 3 に示す．

2.4　関連研究

SSD への I/O 完了を待つ処理を対象とした研究では，

ⓒ2026 Information Processing Society of Japan

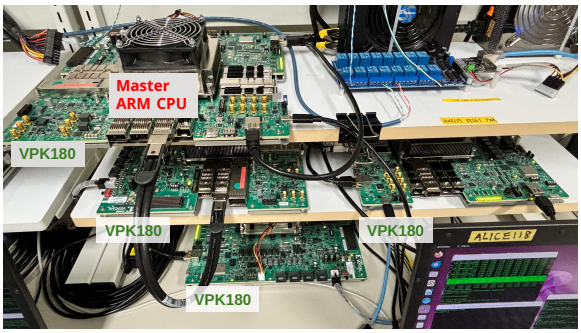

図 5　評価に用いた 4 枚の VPK180 の実機

表 1　ハードウェア構成

項目	構成
CPU	Arm Cortex-A72（2 コア）
アクセラレータ	IMAX3，最大 8 レーン，145 MHz
PE 数	64 / レーン
メモリ容量	OS 用 8 GB，DMA 用 4 GB

ポーリングは CPU の競合下で性能が低下し，休止してからポーリングする従来の方式は I/O の遅延による変動に追従できないことが指摘されている．この研究で提案された DPAS は，直近の I/O から休止時間を調整し，ポーリング，割込み，およびこの方式を実行中に切り替える [7]．また，複数の利用者が CPU を共有する GPU のクラウド環境では，CPU の共有が CPU と GPU の間の同期待ちに干渉して性能を低下させ，その影響はポーリングによる同期の方が大きいことが分析されている [8]．これらは，完了の待ち方を選ぶ際に，デバイスの処理時間と CPU の実行条件を併せて考慮する必要があることを示している．

IMAX3 では，GGML を用いた LLM の実行が評価されている [9]．また，Stable Diffusion の量子化した内積の計算では，カーネルの実行時間が 2 レーンまでは効率よく短縮される一方，3 レーン以上では短縮が鈍化し，その要因として，2 コアを持つ CPU によるデータ供給と実行制御の限界が考察されている [10]．LLM 推論を対象とした総合評価でも，2 コアを持つ CPU によるデータ転送と実行制御が，2 レーンを超える構成の性能向上を制約すると指摘されている [5]．本研究では，CPU のコア数を変えずに，演算の完了を待つスレッドを休止させる条件を変更する．既存の演算時間の見積りを利用し，CPU による演算結果のコピーに要する時間，CPU への割当て待ち時間，および LLM 推論全体の実行時間を比較することで，使用するレーン数に適した待機条件を検討する．

3.　多レーン実行時の問題と改善方法

3.1　CPYOUT の長時間化という観測

図 4 に，2・4・8 スレッドでの CPYOUT と閾値の関係を示す．測定条件は第 4 節に示す．従来の閾値 100 では，スレッド当たりの平均累積時間が，2 スレッドの 1.51 秒に対し，4 スレッドで 44.57 秒，8 スレッドで 69.17 秒となった．閾値 25 以上では同様に，スレッド数の増加とともに CPYOUT が長時間化する．これらはプロセス開始からのウォームアップを含む計測値であり，後述する T_{MAIN} とは計測範囲が異なる．この観測は，アクセラレータ側の並列演算だけでは性能向上が得られず，CPU によるコピーや実行制御が制約となる可能性を示しており，本研究の出発点である．ただし，スレッド数と使用レーン数を同時に変えているため，CPU のスケジューリング競合だけが原因とは断定できず，共有 DMA，メモリ帯域，バスなどの競合や，処理の進行順序の変化も候補である．

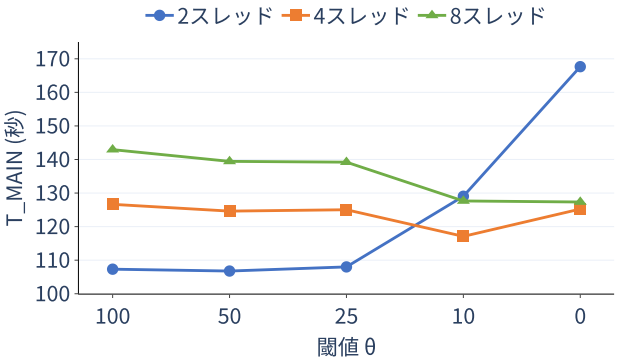

図 6　2・4・8 スレッドにおける閾値と T_{MAIN} の関係

3.2　原因の仮説と変更内容

本研究では，演算の完了を待つスレッドのポーリングが CPU を使用し続け，別のレーンを制御するスレッドのコピーや次の演算の起動を遅らせることを考える．本評価の測定では，従来の閾値 100 のとき，Q2_K，Q3_K，Q6_K の演算の完了を待つ間に休止は一度も行われていなかった．閾値を下げて休止する機会を増やすと，あるレーンの演算中に，別のレーンの入力準備や CPYOUT を進められる可能性がある．一方，短い演算でも頻繁に休止すると，休止・再開や完了検出の遅れが全体の実行時間を増やす可能性がある．そこで，式 (1) の見積値 p と比較する閾値 θ のみを変更し，演算内容，演算ループ回数と見積値の求め方，休止させる時間の求め方，DMA 転送の完了を待つ処理は変更しない．閾値は各実行中に固定し，従来値 100 と，0，10，25，50 を比較する．

4.　評価

4.1　目的と実験条件

本評価の目的は，演算の完了を待つ間にスレッドを休止させるかどうかを判断する閾値を変えることで，多レーン構成の LLM 推論を従来条件より高速化できるかを検証し，実行時間が最短となる閾値がレーン数によってどう異なるかを明らかにすることである．併せて，Sleep と CPYOUT の累積時間，CPU 実行時間および CPU への割当て待ち時

ⓒ2026 Information Processing Society of Japan

間の変化から，CPU の競合が性能を制約しているという仮説を検証する．

表 1 にハードウェア構成，図 5 に実験システムの外観を示す．AMD Versal Premium VPK180 を 4 枚使用し，1 枚をマスターボード，残り 3 枚をスレーブボードとする．各ボードに IMAX3 を 2 レーンずつ実装し，マスターボード上の Arm Cortex-A72 の 2 コアで最大 8 レーンを制御する．OS には 64 ビット Arm 向け PetaLinux を使用し，各レーンにユーザ空間の制御スレッドを 1 つ割り当てる．

LLM 推論には IMAX 向けに拡張した llama.cpp を用い，Gemma-2-9B-it を Q2_K 形式で量子化したモデルを実行する．入力プロンプトは “Prime numbers smaller than ten”，生成トークン数の上限は 32 とする．モデル，入力プロンプト，生成トークン数の上限およびスレッド数以外の実行オプションは，全条件で共通とする．このモデルの計測対象には，Q2_K に加えて Q3_K および Q6_K の処理が含まれる．使用レーン数とスレッド数を一致させた 2・4・8 レーン構成に対し，閾値を 0，10，25，50，100 とする計 15 条件を各 1 回実行し，従来の閾値 100 に対する実行時間の変化を評価する．4・8 レーンを主な改善対象とし，2 レーンは比較対象とする．

4.2 指標と計測範囲

閾値の変更に伴う処理時間と CPU 使用状況の変化を調べるため，各処理区間の経過時間と，スレッドごとの CPU 実行時間および CPU への割当て待ち時間を計測する．経過時間は，Linux 上で利用できる時間計測関数を用いて各処理区間の開始時刻と終了時刻から求める．CPU 実行時間と CPU への割当て待ち時間は，Linux のスケジューラの統計情報を使用して求める [11]．指標の T_{MAIN} は，プロンプトの処理から生成の終了までの推論の実行時間である．処理区間の分析には，Q2_K，Q3_K，Q6_K の IMAX 処理区間と，その中の CPYIN，BZERO，CPYOUT，Sleep の経過時間を用いる．Sleep は，演算の完了を待つ間に usleep を呼び出してから実際に復帰するまでの経過時間である．各区間の値は，各スレッドの経過時間を実行全体で合計し，スレッド数で割ったスレッド当たりの平均累積時間である．IMAX 処理区間から上記 4 区間を差し引いた残りをその他とする．その他には，DMA 転送とその完了を待つ時間や，休止せずに演算の完了をポーリングする時間などが含まれる．

4.3 LLM 推論全体の実行時間

図 6 に全 15 条件の T_{MAIN} を示す．4 レーンでは従来の閾値 100 の 126.64 秒に対し，閾値 10 で 117.09 秒となり，7.54%短縮した．8 レーンでは閾値 100 の 142.93 秒に対し，閾値 0 で 127.33 秒となり，10.91%短縮した．4 レーンでは閾値 0 の 125.23 秒より閾値 10 が短く，25 以上では 124.62〜126.64 秒となった．8 レーンでは閾値 0 と 10 がそれぞれ 127.33 秒，127.65 秒であり，25 以上の 139.20〜142.93 秒より短かった．比較対象の 2 レーンでは閾値 50 の 106.76 秒が最短で，閾値 100 との差は小さい．各構成の最短値では 2 レーンが最速となった．

表 2 閾値 0 の Q2_K における休止 1 回当たりの平均時間

	2 スレッド	4 スレッド	8 スレッド
要求した時間 (μs)	4.75	4.75	4.75
復帰までの時間 (μs)	64.86	77.76	190.70

4.4 Sleep・CPYOUT と CPU 使用の変化

図 7 に，4・8 スレッドの IMAX 処理区間の内訳を示す．4 スレッドでは，閾値を 100 から 0 へ下げると Sleep が 0 秒から 65.50 秒へ増え，CPYOUT は 44.57 秒から 1.03 秒へ減った．中間の閾値 10 では Sleep が 33.71 秒，CPYOUT が 13.51 秒であった．8 スレッドでも，閾値 100 から 0 への変更に伴い Sleep は 0 秒から 89.30 秒へ増加し，CPYOUT は 69.17 秒から 0.71 秒へ減少した．閾値 25・50 で休止したのは Q3_K の一部の演算だけであり，Sleep は小さく，CPYOUT も閾値 100 と同程度であった．一方，閾値 0 では短い演算でも休止する．表 2 に，モデルの量子化形式である Q2_K について，閾値 0 で休止 1 回当たりに要求した時間と，usleep を呼び出してから復帰するまでの時間の平均を示す．要求した時間は 4.75 μs であったのに対し，復帰までの時間は 2 スレッドでも 64.86 μs であり，スレッド数が多いほど長くなった．

図 8 に，CPU 実行時間と CPU への割当て待ち時間を示す．閾値 100 の CPU への割当て待ち時間は，2・4・8 スレッドで 1.32 秒，60.47 秒，102.67 秒であり，スレッド数とともに大きく増えた．4 スレッドの閾値 10 では 41.90 秒，8 スレッドの閾値 0 では 75.55 秒に減少し，CPU 実行時間も減少した．

5. 考察

5.1 レーン数によって有利な閾値が異なる理由

2 コアに対して 4・8 スレッドを割り当てる構成では，ポーリングで演算の完了を待つスレッドが，別のレーンのコピーや実行制御と CPU の実行機会を奪い合う．待機中のスレッドが休止すれば，あるレーンの演算中に別のレーンのコピーや演算の起動を進めるという，パイプライン的な処理の重なりを活用できる．閾値を下げた際の CPYOUT と CPU への割当て待ち時間の減少は，この仮説と整合する．8 レーンでは他のレーンの処理が多く，閾値 100 でも CPU への割当て待ち時間が 102.67 秒に達しており，休止しないことによる完了検出の早さが生かされにくい．そのため，すべての演算で休止させる閾値 0 が最も短くなったと考えられる．ただし，閾値 10 との差は小さく，0 を普遍

ⓒ2026 Information Processing Society of Japan

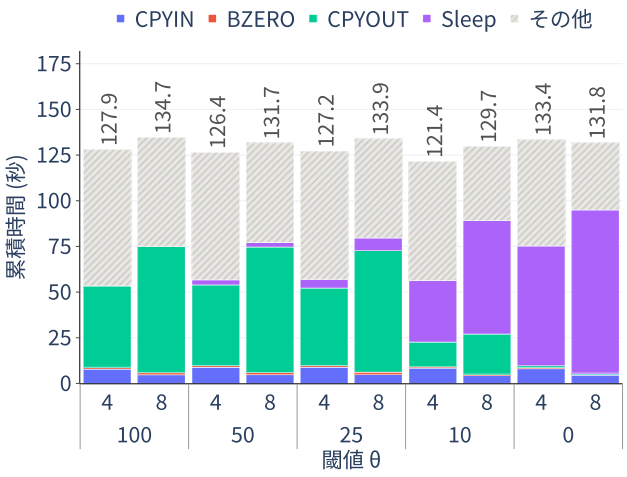

図 7 4・8 スレッドにおけるアプリケーション側から見た IMAX 処理時間の内訳（スレッド当たりの平均累積時間）

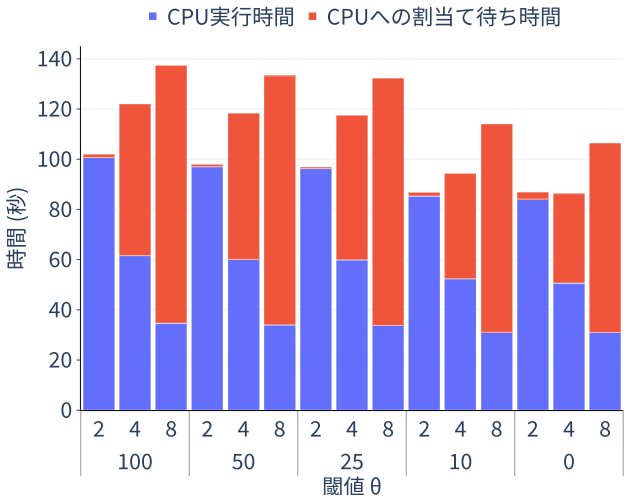

図 8 2・4・8 スレッドにおけるスケジューラ側から見た CPU 実行時間と CPU への割当て待ち時間（スレッド当たりの平均累積時間）

的な最適値とは結論付けられない．一方，表 2 に示したように，休止は呼び出してから復帰するまでに，要求した時間を大きく上回る時間を要する．閾値を 10 から 0 に下げて新たに休止するのは見積値 p が 10 以下の短い演算であり，休止中に演算が終わっても，復帰するまで完了を検出できない．4 レーンでは，このような短い演算を休止せずに待ちつつ，長い演算では休止して CPU を譲る閾値 10 が最も短くなったと考えられる．2 レーンではスレッド数とコア数が等しく，休止によって CPU を譲る相手がほとんどないため，休止させずに待つ方が速く，休止・再開の負荷が閾値 0 での大幅な増加につながったと考えられる．このように，測定した範囲では最適な閾値がレーン数によって異なり，レーン数が多いほど休止させる方が有利になる傾向が得られた．

5.2 今後の課題

本研究では，レーン数ごとに閾値を固定して評価しており，実行中に閾値を変更する方式は実装していない．測定した範囲では最短となる閾値がレーン数によって異なったことから，今後は，使用するレーン数や実行中の CPU の状況に応じて，閾値を動的に変更する方式を検討する．また，IMAX では，サーバ環境に向けた試作も進められている [4]．本研究では，レーン数が多いほど休止させる方が有利になる傾向が得られたことから，より多くのレーンを用いる構成では，演算の完了を待つ間の CPU の使い方が性能に与える影響はさらに大きくなると考えられる．そのため，サーバ環境での利用を見据えて，より多くのレーンを制御する方法についても検討する．

6. まとめ

本研究では，2 コアを持つ CPU で最大 8 レーンを制御する IMAX3 を対象に，演算の完了を待つ間にスレッドを休止させるかどうかを判断する閾値を変更し，多レーン実行時の LLM 推論の高速化を図った．レーン数の増加に伴い CPYOUT が長時間化することを観測し，演算の完了を待つスレッドのポーリングによる CPU の競合が，多レーン実行の性能を制約している可能性を示した．測定した範囲で最短となる閾値は 2・4・8 レーンで 50・10・0 と異なり，8 レーンではすべての演算で休止させる方が速く，2 レーンでは休止させずに待つ方が速いという関係が得られた．それぞれのレーン数に適した閾値を選ぶことで，アクセラレータの構成や演算の内容を変えずに，従来の閾値 100 に対して 4 レーンでは閾値 10 で 7.54%，8 レーンでは閾値 0 で 10.91%，LLM 推論の実行時間を短縮した．今後は，使用するレーン数や実行中の CPU の状況に応じて閾値を動的に変更する方式と，サーバ環境を見据えたより多くのレーンの制御方法を検討する．

参考文献

[1] International Energy Agency: Key Questions on Energy and AI, IEA, Paris (2026). 2026 年 4 月 16 日公表.
[2] Liu, L., Zhu, J., Li, Z., Lu, Y., Deng, Y., Han, J., Yin, S. and Wei, S.: A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications, *ACM Computing Surveys*, Vol. 52, No. 6, pp. 118:1–118:39 (online), DOI: 10.1145/3357375 (2019).
[3] Akabe, T., Le, V. T. D. and Nakashima, Y.: IMAX: A Power-Efficient Multilevel Pipelined CGLA and Applications, *IEEE Access*, Vol. 13, pp. 31899–31911 (online), DOI: 10.1109/ACCESS.2024.3524415 (2025).
[4] Ando, T., Eto, Y. and Nakashima, Y.: A Detailed Analysis of LLM Execution on IMAX3 and Initial Evaluation of IMAX4 Prototype for Server Environment, *SASIMI 2025 Proceedings*, pp. 8–13 (2025).
[5] Ando, T., Eto, Y., Takeuchi, A. and Nakashima,

ⓒ2026 Information Processing Society of Japan

Y.: Efficient Kernel Mapping and Comprehensive System Evaluation of LLM Acceleration on a CGLA, arXiv:2512.00335v1 (2025). DOI: 10.1109/ACCESS.2025.3636266.

[6] Linux man-pages Project: *usleep(3)*, Linux man-pages 6.19 (2026). `https://man7.org/linux/man-pages/man3/usleep.3.html`.

[7] Seo, D., Jung, J., Yoon, Y., Chen, P.-X., Joo, Y., Lim, S.-S. and Dutt, N.: DPAS: A Prompt, Accurate and Safe I/O Completion Method for SSDs, *24th USENIX Conference on File and Storage Technologies (FAST 26)*, pp. 381–397 (2026).

[8] Elmougy, Y., Jia, W., Ding, X. and Shan, J.: Diagnosing the Interference on CPU-GPU Synchronization Caused by CPU Sharing in Multi-Tenant GPU Clouds, *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–10 (online), DOI: 10.1109/IPCCC51483.2021.9679439 (2021).

[9] Uetani, H. and Nakashima, Y.: Implementation and Evaluation of LLM on a CGLA, *2024 Twelfth International Symposium on Computing and Networking (CANDAR)*, pp. 252–258 (online), DOI: 10.1109/CANDAR64496.2024.00040 (2024).

[10] Ando, T., Eto, Y. and Nakashima, Y.: Implementation and Evaluation of Stable Diffusion on a General-Purpose CGLA Accelerator, *2025 IEEE 18th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)* (2025).

[11] The Linux Kernel Documentation: Scheduler Statistics, `https://docs.kernel.org/scheduler/sched-stats.html`.

ⓒ2026 Information Processing Society of Japan