B-016

マルチノードコンピューティングシステムにおける GPGPU の並列処理による高効率処理の実現
Highly efficient processing by parallel processing of GPGPUs in multi-node computing systems

宮川大輝† 李彦志† 菅谷みどり†
Miyakawa Taiki Li Yanzhi Sugaya Midori

1. 背景

近年 GPU，FPGA など異なるアクセラレータを複数持つハードウェアを統合し，アプリケーションを高速かつ低消費電力で計算するマルチノードコンピューティングシステムが提案されている[1, 2]．マルチノードコンピューティングシステムでは，複数の計算機（ノード）のそれぞれが，アクセラレータを搭載したクラスタを構成しており，アクセラレータの特性を活かした大量の計算処理を行う．各ノードは，高速ネットワークを通じてジョブを受信し，高い計算力，応答性を活かした処理を行うことができることから，エッジコンピュータ[3]としても活用されている．

例えば，マルチノードコンピューティングシステムでは，ノードのクラスタの計算力が高いことから，通常複数のジョブを割り当てた計算が期待される．1 度に複数のジョブを割り当てて並列処理をする場合，アクセラレータによっては処理効率が十分に向上しないケースが存在する．

例えば，NVIDIA が提供する GPGPU においては，数個のジョブが割り振られたとしても，1 つのプロセスが逐次実行してしまうと，実行効率が向上しないという課題がある．また，こうした GPGPU では，VRAM 容量が小さい場合，数個のジョブを実施する場合には問題が発生しないが，ある一定以上のジョブを割り当てようとすると，GPU の VRAM 不足による異常終了が発生する場合がある[4]．これらの問題の解決が課題となっている．

2. 目的・提案

本研究では，マルチノードコンピューティングシステムにおける GPGPU の処理効率の向上を目的とする．実現のために，次の 2 つの提案を行う．一つ目は，処理効率の向上である．具体的には，複数のジョブの受け取り，ノードのプログラムの実行を並列処理する手法である．二つ目に，異常終了の抑制を行う手法を提案する．

2.1 処理効率の改善

一つ目の提案として，ノードのプログラムの実行の並列処理手法について述べる．GPGPU の従来実装では，受け取ったジョブの結果を送り返すまでは新たなジョブの受け取りや実行は出来ない問題があった(図 1：上)．そのため，ジョブ受信から GPGPU の結果送信の間で CPU のみが使用され GPU は使用されず，非効率であった．

本実装では，ジョブを受け取るジョブ実行プログラムを 2 つ起動し，最大 2 つまでのジョブを同時に受け取り，並列実行できるようにした．それぞれのプログラムが並列に実行されることで，GPU の未使用時間が減り，GPU の利用効率を改善する．

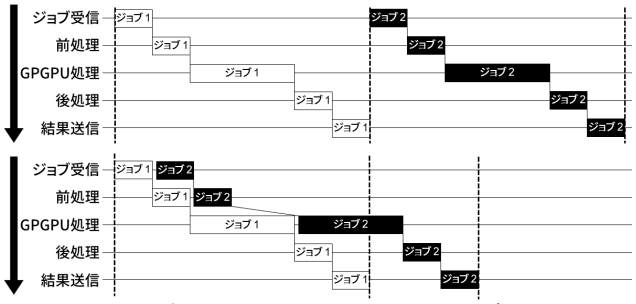

図 1．処理の流れ (上: 従来システム，下: 提案システム)

2.2 異常終了の抑制

二つ目の提案として，異常終了の抑制について述べる．

1 節に述べたように，GPGPU 処理における並列実行の課題の一つに，VRAM 不足によるプログラムの異常終了があげられる．現在の GPU はスワップアウトに対応していないため，複数のプログラムを並列実行すると VRAM の空き領域が無くなって新たな領域確保が出来なくなる．この場合に，プログラムは異常終了してしまうことがある．

回避のため，本設計・実装では，あるプロセスが GPU を使用している間は，排他ロックを適用し，GPU での処理は同時に 1 プロセスのみに制限することとした．排他ロック中に他のプロセスが GPU を使用しようとすると処理が一時停止され，ロックが解除されると処理が再開される．

本提案システムの構成図と処理フローを図 2 に示した．

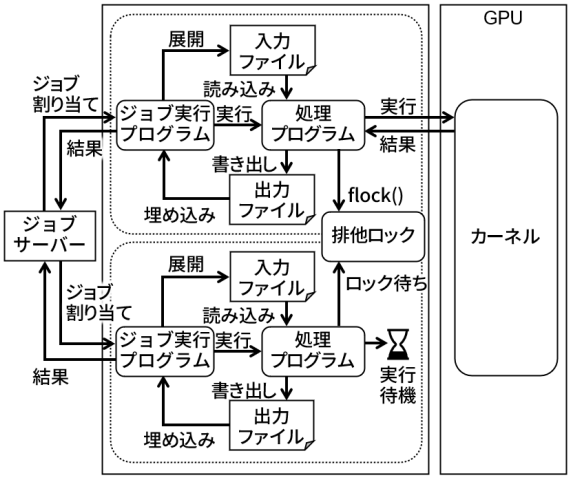

図 2．提案システム構成図

3. 評価

同時に 1 つのジョブのみ割り当てる従来システムと，上記 2.1.2，2.1.3 を適用した提案システムとの比較を行った．

† 芝浦工業大学
Shibaura Institute of Technology, 3-7-5 Toyosu, Koto-ku, Tokyo 135-8548, Japan

評価システムの構成を表 1 に示した．

表 1：実験システム構成

CPU	NVIDIA Carmet ARM®v8.2，6 コア
GPU	NVIDIA Volta™ アーキテクチャ
メモリ	8GB LPDDR4X
OS	NVIDIA Jetson Linux 32.7.1 (Jetpack 4.6.1)
Ethernet	1000 BASE-T Ethernet

3.1 実験 1

評価実験では，畳み込みニューラルネットワークによる画像の超解像プログラム [5] を使用した．

超解像プログラムでは，入力は画像データ，出力は超解像処理後の画像データである．1 回の実験につき，ジョブを 10 個連続して発行する．1 つ目のジョブの送信時から，全ジョブが返ってくるまでの時間を計測する．実験は 10 回行い，平均値と最大値を求める．本実験を，異なる解像度の画像により，従来システムと提案システムそれぞれで実施し，実験結果を図 3 に示した．

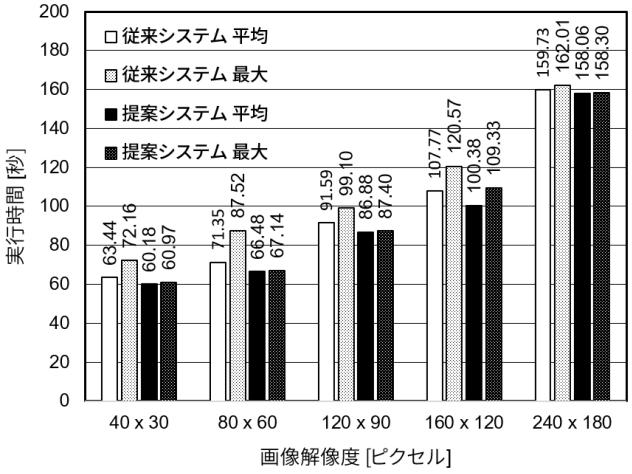

図 3．実験 1 における実行時間の比較

結果より，従来システムと比較して，提案システムの実行時間の平均値が約 1～7%，最大値が約 2～23%短縮されたことが分かった．

3.2 実験 2

次に，サイズの大きいジョブの処理による違いを明らかにするため，動画エンコードプログラム FFmpeg [6] を用いて実験を行った．

FFmpeg では，入力ファイルは動画データ，出力ファイルはエンコードされた動画データとし，1 回の実験につき，ジョブを 5 個連続して発行する．時間計測と統計処理は実験 1 と同様とする．本実験を，いくつかの解像度の動画を用いて，従来システムと提案システムそれぞれで行う．

本実験の結果を図 4 に示した．

結果より，従来システムと比較して，提案システムの実行時間の平均値が約 6～24%，最大値が約 4～54%短縮されたことが分かった．

実験 1，2，いずれにおいても，実行時間の短縮が見られた．平均値の差は解像度によらずほぼ一定であった．最大値を比較すると，大きな差が出た箇所があった．ジョブの実行状態を観察していると，既存システムでは，ジョブサーバーからジョブが割り当てられる際に遅延が生じることがあり，提案システムでは，ジョブの割り当て中も別のジョブの処理が実施されていた．提案した並列処理により遅延が改善されたと考えられる．また，異常終了の抑制については，複数のプログラムを並列実行しても異常が発生せず，提案手法の有効性を確認することが出来た．

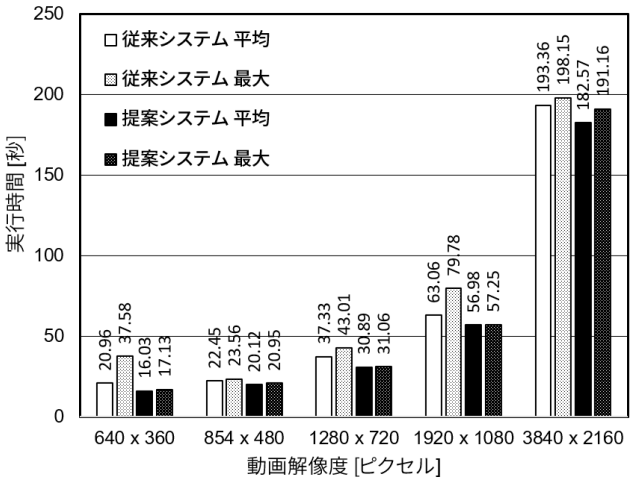

図 4．実験 2 における実行時間の比較

4. おわりに

本研究では，処理効率の改善と，異常終了の抑制手法を提案し，評価により有効性を示した．

今後の課題としては，以下の 3 点があげられる．

1 つ目は，現段階のシステムは，CPU 側の並列実行数が固定であり，GPU 側のメモリ不足による異常終了を抑制するため逐次実行としている点である．これらの制限を動的に変更する手法を考案し，より高効率にアプリケーションを処理できるように改善したいと考える．

2 つ目は，現段階のシステムでは，ノード単体の処理効率のみを考慮している点である．他のノードの状態に合わせたスケジューリングを行い，システム全体の処理効率を高める手法を提案したいと考える．

3 つ目は，他の指標を使った評価も必要であるという点である．システムを評価する際には，1 つの指標だけでは不十分だと考えられる．各ジョブの平均応答時間や消費電力など，複数の指標による評価手法を提案したい．

謝辞

本研究は，JST，CREST，JPMJCR19K1 の支援を受けたものです．

参考文献

[1] Zhe Fan, Feng Qiu, Arie Kaufman, Suzanne Yoakum-Stover, “GPU Cluster for High Performance Computing,” SC '04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing.

[2] Miho Yamakura, Ryousei Takano, Akram Ben Ahmed, Midori Sugaya, Hideharu Amano “A Multi-tenant Resource Management System for Multi-FPGA Systems”, IEICE Trans. Information and Systems, Vol.E104-D,No.12,pp.-,Dec. 2021.

[3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," in IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1125-1142, Oct. 2017, doi: 10.1109/JIOT.2017.2683200.

[4] Kaibo Wang, Xiaoning Ding, Rubao Lee, Shinpei Kato, Xiaodong Zhang, GDM: device memory management for gpgpu computing, ACM SIGMETRICS Performance Evaluation Review, Volume 42, Issue 1, June 2014

[5] BMP-Super-Resolution, https://github.com/yeonzi/BMP-Super-Resolution

[6] FFmpeg, https://ffmpeg.org/