5J-05

マルチGPU上での畳み込みニューラルネットワークにおけるハイブリッド並列処理

Hybrid Parallelization for Convolutional Neural Network on Multi-GPU

綿貫 幸† 吉田 明正†
Yuki Watanuki Akimasa Yoshida

1 はじめに

深層学習は多くの分野で活用されているが，性能向上を目的とした訓練データやパラメータの大規模化による学習時間の増加が課題となる．マルチGPUを用いた深層学習の並列処理手法として，データ並列とモデル並列がある．データ並列は単純な実装で学習時間を短縮することができ，モデル並列は学習に必要なメモリ容量を抑えつつ，パイプライン処理の適用によって学習を高速化することができる．本稿では，代表的な深層学習モデルであるCNNに対して，マルチGPU環境において各GPUに複数ステージを割り当てるモデル並列を適用し，データ並列と組み合わせたハイブリッド並列による高速化を図る．画像分類CNNのマルチGPU向け並列プログラムをCUDAとOpenMPを用いて実装し，NVIDIA Tesla K80搭載サーバ上で性能評価を行い，提案手法の有効性を確認した．

2 マルチGPUによる深層学習の並列処理

本章では，深層学習の並列処理手法について述べる．

2.1 データ並列

データ並列は，1つの学習モデルを複数のデバイスに複製することで，複数のバッチを並列に処理する手法である．2GPUによるデータ並列の実装例を図1に示す．図のように各GPUに割り当てられたミニバッチの逆伝播までが完了した後，1つのGPUに勾配を集約し平均化する．その値によって更新されたパラメータを各GPUに複製すると，次のミニバッチが投入され，学習が進行する．データ並列はモデルに手を加える必要がなく，容易に深層学習の高速化を達成することが可能である．しかし，モデル全体を複製する手法であるため，大規模なパラメータを持つモデルにおいては，メモリ不足や通信時間の増加が起こり得る．また，学習時の実質のバッチサイズがデバイス数に比例して増大するため，過剰に並列性を高めると精度劣化を招く．

2.2 モデル並列

モデル並列は，1つの学習モデルを複数のステージに分割し，各ステージをデバイスに割り当てる手法である．これにより，大規模なモデルを実装する際の要求スペックを低減することができる．さらに，パイプライン型モデル並列は，ミニバッチをマイクロバッチに分割してパイプライン処理を適用することで，深層学習の高速化を達成する[1][2]．図2は，モデルを2つのステージに分割し，2つのマイクロバッチを並列に処理するパイプライン型モデル並列の実装例である．図のようにマイクロバッチを並列に処理し，最後尾のマイクロバッチの逆伝播完了時に勾配を平均化し，ステージ毎にパラメータを更新する．このとき，デバイス間の通信はステージ間の入出力データの受け渡しに限られるため，データ並列と比較して通信時間の抑制が期待できる．パイプライン処理による速度向上率は，より負荷が大きいステージの処理時間に依存するため，各ステージ間の負荷が均等になるように分割位置を決定する必要がある．

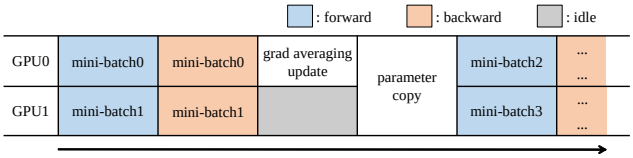

図1 2GPUによるデータ並列.

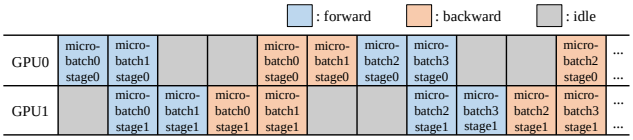

図2 2GPUによる2分割パイプライン型モデル並列.

3 ハイブリッド並列による深層学習の高速化

本章では，本稿で提案するハイブリッド並列の手法と，その実装方法について述べる．

3.1 データ並列とモデル並列の併用

本稿では，データ並列とパイプライン型モデル並列を併用することで，深層学習の高速化を実現し，かつ大規模なモデルに対応可能なハイブリッド並列の手法を提案する．4GPU上で提案手法によるハイブリッド並列を適用した実装は，図3のようになる．まず2GPU上でパイプライン型モデル並列を実装し，パイプラインを複製することで2バッチを同時に処理するデータ並列を実装している．また，各GPUに2つのステージが割り当てられるようにモデルの分割数を増やすことで，パイプライン処理のさらなる速度向上を図っている[3]．ここで，2.2節で述べたように，パイプライン処理の効率を高めるためには，ステージ間の負荷の不均衡を考慮する必要がある．そこで本稿では，CNN各層の順伝播における実行時間を計測し，順伝播のパイプライン処理時間が最短となるように分割位置を決定する[3]．

3.2 画像分類CNNへのマルチGPU並列処理の実装

本稿では，C++で実装された画像分類CNN[4]に対して提案手法を適用する．CNNの各層の処理にはCUDAカーネルとcuBLASライブラリ関数を使用しており，各処理はGPU上で実行される．マルチGPU向けの並列処理の実装には，OpenMPを使用する．順伝播を開始する直前で，`parallel`指示文により4スレッドの並列

†明治大学大学院先端数理科学研究科ネットワークデザイン専攻
Network Design Program, Graduate School of Advanced Mathematical Sciences, Meiji University

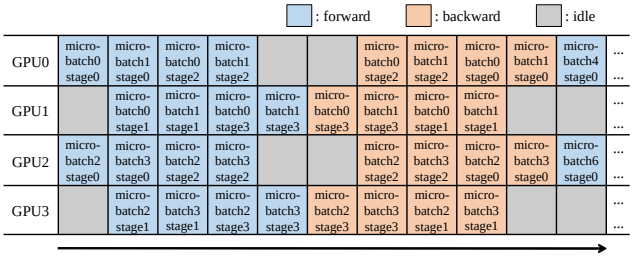

図 3　提案手法による 4GPU 上でのハイブリッド並列.

領域を作成し，スレッド番号による分岐でマルチ GPU 並列処理を行う．パイプライン処理の工程が進む毎に `#pragma omp barrier` による同期処理を挟み，逆伝播完了時にはステージ毎に勾配を平均化し，更新したパラメータの複製を行う．

4　NVIDIA Tesla K80 搭載サーバ上での性能評価

本章では，性能評価について述べる．

4.1　性能評価環境

性能評価に用いるマルチ GPU サーバの構成を，表 1 に示す．本性能評価では，CIFAR-10 データセット [5] を用いた 10 クラスの画像分類 CNN プログラムに提案手法を適用し，学習時間を測定する．

表 1　性能評価に用いるマルチ GPU サーバの構成.

マシン	NVIDIA Tesla K80 搭載サーバ
プロセッサ	Intel Xeon E5-2680 v3 (2.5GHz, 12Core×2)
メモリ	64GB
GPU	NVIDIA Tesla K80×2 (GK210×4)
OS	CentOS 6.9
CUDA Toolkit	9.1
g++	5.3.1

4.2　マルチ GPU 上でのハイブリッド並列 CNN の性能評価

図 4 は，バッチサイズを 100 として 20 エポックまで学習した CNN の訓練データ正解率を示す．ここで，MP はモデル並列，DP はデータ並列を適用した実装である．20 エポック完了時の精度を 1GPU 実行と比較すると，4 分割 MP+2DP の実装は 2.67%低下しているが，これはデータ並列の適用によりバッチサイズが 2 倍になっているためと考えられる．

また，図 5 は，各実装方法において 1 ミニバッチ分の学習に関わる GPU メモリ使用量の概算値を示す．数値は各層の入出力，正解ラベル，学習パラメータ，勾配，最適化に利用するモーメントの合計サイズとして算出している．データ並列を適用している場合は，パラメータ更新時に集約される勾配を含む．図 5 より，1GPU あたりの最大メモリ使用量を 1GPU 実行時と比較すると，2 分割 MP+2DP の実装では 72.5%，4 分割 MP+2DP の実装では 61.4%に抑えられている．

さらに，表 2 は，各実装におけるエポック平均学習時間と 1GPU 実行に対する速度向上比を示す．提案する 4 分割 MP ＋ 2DP の実装は，1GPU 実行比で 1.838 倍の速度向上が得られ，2 分割 MP+2DP の実装と比較すると，実行時間を 87.4%に短縮することができた．以上の結果から，提案手法によるマルチ GPU 上でのハイブリッド並列処理について，有効性が確認された．

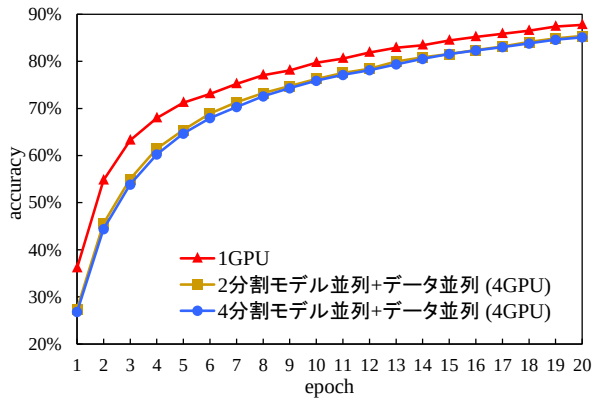

図 4　訓練データの正解率.

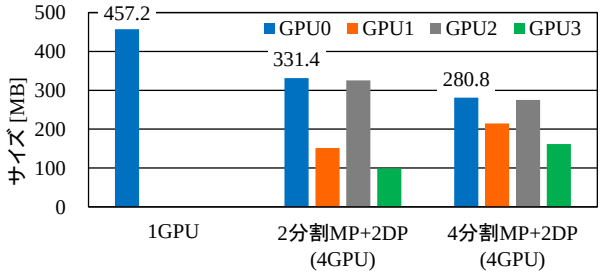

図 5　1 ミニバッチの学習に関わる GPU のメモリ使用量.

表 2　学習時間（エポック平均）.

	エポック平均実行時間 [s]	速度向上比 [倍]
1GPU	8213.1	1.000
2 分割 MP+2DP(4GPU)	5115.0	1.606
4 分割 MP+2DP(4GPU)	4469.2	1.838

(注)MP：モデル並列，DP：データ並列.

5　おわりに

本稿では，マルチ GPU 上での深層学習の高速化を目的として，データ並列とパイプライン型モデル並列の併用によるハイブリッド並列を提案した．画像分類 CNN プログラムに対してマルチ GPU 並列処理を CUDA と OpenMP により実装し，提案手法を適用した．マルチ GPU サーバ上での性能評価の結果，4 分割モデルを用いたハイブリッド並列は，2 分割モデル並列を用いたハイブリッド並列と比較して，GPU のメモリ使用量を抑えた上で実行時間を 87.4%に短縮することができており，提案手法の有効性が確かめられた．

参考文献

[1] Huang, Yanping, et al. GPipe: Efficient training of giant neural networks using pipeline parallelism, Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp.103-112, 2019.

[2] Narayanan, Deepak, et al. PipeDream: generalized pipeline parallelism for DNN training, Proceedings of the 27th ACM Symposium on Operating Systems Principles, pp.1-15, 2019.

[3] 綿貫幸, 吉田明正. マルチ GPU 上での畳み込みニューラルネットワークにおけるモデル分割配置, 情報処理学会研究報告, 2023-HPC-188-7, 2023.

[4] 藤田毅. C++で学ぶディープラーニング, マイナビ出版, 2017.

[5] The CIFAR-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html.