Chainer MN を用いた深層学習の性能評価

塙　敏博

東京大学情報基盤センター

１．はじめに

本稿は、2017 年 12 月に実施された「Reedbush-H 大規模 HPC チャレンジ」において、GPU クラスタに対応した深層学習(Deep Learning)フレームワークである Chainer MN を用いて ImageNet の画像分類および seq2seq による機械翻訳の学習について性能評価を実施した報告である。

２．Chainer MN

近年、GPU を用いた深層学習(Deep Learning)の研究が盛んに行われており、深層学習を実現するフレームワークも多く開発されている。その中で Preferred Networks 社(PFN)が中心となって開発しているオープンソースの Chainer[1]は、Python で実装されており、define-by-run モデルによって簡便な記述で自由度の高いネットワークを記述することができることや、NVIDIA 製の GPU を用いた場合に高い性能を実現しているため、広く使われているフレームワークの一つである。その一方で、深層学習には大量の教師データを用いて学習させることが必要である。よく知られている画像分類データセットとして ImageNet があるが、100 万枚を超える画像データにカテゴリを示すアノテーションが付けられたものである[2]。しかし GPU 単体では、最新の GPU を使っても学習に 1 週間以上必要になってしまう。

Chainer MN は、複数ノードの GPU を用いてデータ並列に学習を行う、Chainer の追加パッケージである[3,4]。ミニバッチを各 GPU に分散させて独立に学習を行い、各 GPU における学習によって得た勾配を全 GPU 間で Allreduce 通信をして、全体の学習結果に反映させる。しかし、並列化によってスループットを上げることは容易だが、相対的にミニバッチサイズが大きくなるため得られるモデルの精度が悪化することが知られている。そのため、PFN による試行では、128 GPU (GeForce GTX TITAN X)を用いて GPU 毎のバッチサイズを 32 に抑えることにより、ResNet-50 のモデル[6]を用いて、4.4 時間で 70%以上の精度を達成している[5]。

そこで、この大規模 HPC チャレンジでは、(実施当時)最新バージョンの Chainer MN を Reedbush-H 全系（最大 120 ノード、合計 240 NVIDIA Tesla P100 GPU)を用いた時の ImageNet の画像分類、および seq2seq による機械翻訳を対象に主にスケーラビリティの評価を実施した。

３．Reedbush-H における準備

Reedbush-H にはあらかじめ、Chainer および ChainerMN もインストールされている。しかし、この分野は非常に研究開発の進捗が速く、実施した 2017 年 12 月現在、インストール済みの Chainer は 2.0.0、Chainer MN は 1.0.0b2 だったのに対し、すでに Chainer は 3.1.0, Chainer MN は 1.0.0 になっていた。バグフィックスおよび性能改善が随時行われているため、なるべく新しいバージョンを使用したいが、システムにインストールされた環境を更新する頻度には限界がある。

そこで、今回はインストール済みの Anaconda3 環境をベースに、Python を含めた最新の環境

を手元に構築することとした。詳細な手順は以下の通りである。

1. CUDA モジュールをロード: `module load cuda/8.0.44-cuDNN7`
   NVIDIA が提供する GPU 利用ライブラリ CUDA 8.0.44 と同時に深層学習用ライブラリ cuDDN7 と GPU 間の通信を最適化した通信ライブラリ NCCL(NVIDIA Collective Communications Library)も同時に利用可能になる。
2. デフォルトの Intel MPI から OpenMPI モジュールに切り替え： `module switch intel-mpi openmpi-gdr/2.1.1/intel`
   Intel MPI では GPU メモリ間のデータを MPI で送受信できない。また MVAPICH2 ではエラーになるため、OpenMPI 2.1.1 を選択した。
3. Anaconda3 モジュールをロード: `module load anaconda3`
4. Anaconda 環境を構築するにあたって、ホームディレクトリを Lustre ファイルシステムに変更： `export HOME=/lustre/gi99/i12345`
   Lustre ファイルシステムでないと、計算ノードから参照できないためである。
5. Anaconda 環境を create, activate する。ここでは chainerMN という環境名にする ： `conda create -n chainerMN python=3; source activate chainerMN`
6. Cupy をインストール: `pip install -U cupy --no-cache-dir -vvv`
   Cupy は numpy 互換の GPU 向け実装であり、Chainer の GPU 実行において重要な役割を果たしている[7]。
7. 6 と同様に、cython、chainer, chainermn の順にインストールする。

以上の設定は、実行時にジョブスクリプト中でも指定する必要があることに注意が必要である(1〜5 項)。

４．Reedbush-H における Chainer MN の実行

図 1 に、Reedbush-H でのジョブスクリプトの例を示す。図 1 中の中段については、前述の環境設定を反映したものである。Mpirun のオプションには、GPU Direct for RDMA (GDR)を有効にするため `--mca btl openib_want_cuda_gdr 1` を指定した[1]。

また、get_local_rank_ompi は、wrapper スクリプトであり、各 GPU が接続されたソケットに MPI プロセスをバインドするような設定を付け加えている。図 2 に実行時のイメージを示す。具体的には、1 MPI プロセスが 1 GPU を制御することになり、各 MPI プロセスは GPU が接続された各ソケットに固定する必要があるため、`numactl --cpunodebind=$OMPI_COMM_WORLD_LOCAL_RANK --localalloc` のように指定している。

加えて、NCCL の最適化として効果があった以下のオプションを環境変数として指定している。

- `export NCCL_SHM_DISABLE=1`
  GPU Direct for RDMA が有効なので、ノード内も共有メモリ(SHM)は使わない方が高速
- `export NCCL_NET_GDR_READ=1`
  READ にも GPU Direct for RDMA を有効にした方が高速

また、ImageNet では 100 万の画像ファイルを扱うため、IME (Infinite Memory Engine)を読み込みキャッシュとして使用した。わずかに効果は見られたが、32 ノード、64GPU のときに 2%

[1] 実際には、ほとんどが MPI ではなく NCCL による通信になるためあまり意味はない。

```
#!/bin/sh
#PBS -q h-regular
#PBS -l select=32:mpiprocs=2
#PBS -l walltime=04:00:00
#PBS -W group_list=gi99

cd $PBS_O_WORKDIR
module load cuda/8.0.44-cuDNN7 anaconda3
module switch intel-mpi openmpi-gdr/2.1.1/intel
export HOME=/lustre/gi99/i12345
source activate chainerMN

mpirun --mca mtl ^mxm --mca coll_hcoll_enable 0 ¥
--mca btl_openib_want_cuda_gdr 1 ¥
--mca mpi_warn_on_fork 0 ./get_local_rank_ompi ¥
python train_imagenet.py … >& log-${PBS_JOBID}
```

図 1 ジョブスクリプトの例

表 1 実行環境

ChainerMN 1.0.0
Chainer 3.1.0
python 3.6.1, CUDA 8, cuDNN7, cupy 2.1.0
通信関係: OpenMPI 2.1.1, NCCL v2

程度の性能向上に留まっている。

５．ImageNet の結果

CNN (Convolutional Neural Network)の典型的な例として、ImageNet の ILSVRC2012 データセットについて ResNet-50 により画像分類を学習させた結果について示す。実行環境は表 1 に示す通りである。また、問題設定としては、Preferred Networks によるもの[3,4]と同じ条件を採用している。

図 3 に、各エポックにおける実行時間と、そのときの validation accuracy を示す。ミニバッチサイズは ChainerMN のデフォルト値で 32 であり、いずれも 100 エポックまでの学習を行った。ただし、32 ノード 64 GPU の場合には 100 エポックの終了までに 14300 秒を要しており、グラフの範囲に収まっていない。30 エポックごとに学習率をそれまでの 1/10 に減少させることで、高い accuracy を得ることに成功している。

Accuracy は、32 ノード 64 GPU で 72.0%, 64 ノード 128 GPU で 72.2%、120 ノード 240 GPU で 71.7%を得た。実行時間を比較すると、120 ノード 240 GPU では、32 ノード 64 GPU の 3.54 倍、64 ノード 128 GPU の 1.77 倍の性能が得られており、良好な strong scaling を示している。

６．Seq2seq の結果

次に、RNN (Recurrent Neural Network)の例として、seq2seq により機械翻訳の学習を行った結果を示す。実行環境は ImageNet と同様である。問題設定は、ChainerMN の Data-parallel seq2seq example におけるものと同一にしており、WMT15 における French-English ペアの学習を

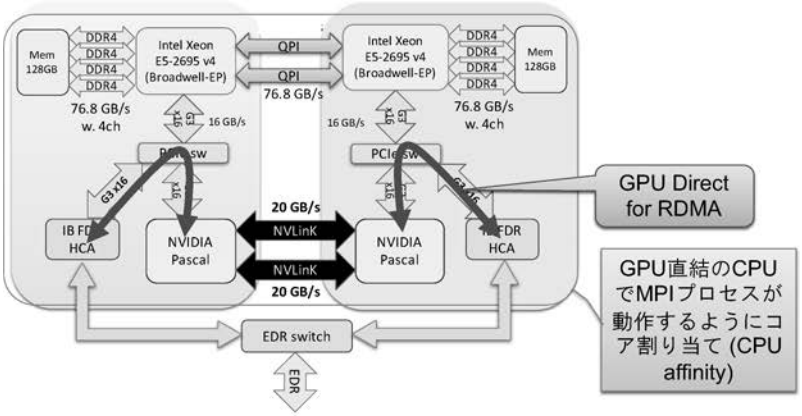

図 2　MPI プロセスの affinity 設定

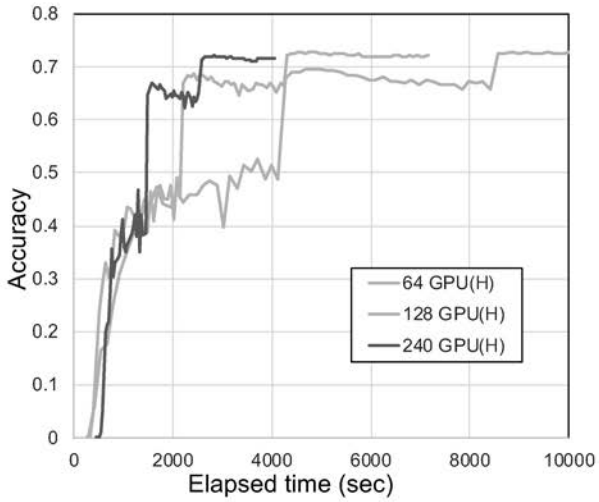

図 3 ImageNet における学習曲線（64 GPU において学習終了は 14300 秒）

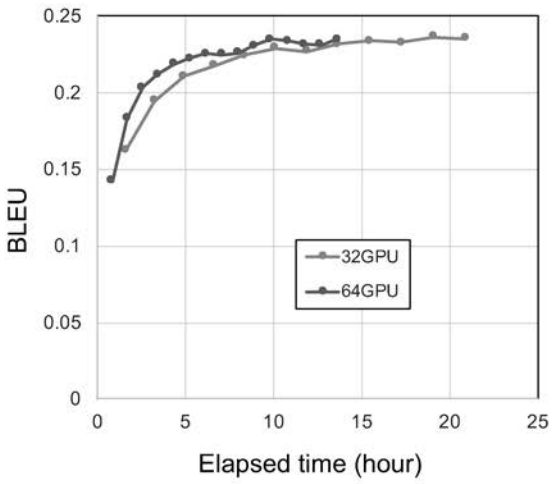

図 4　Seq2seq における学習曲線

（15 epoch まで実行、32 GPU では 12 epoch で 24 時間に達し打ち切り）

対象にしている[9]。15 エポックまでの学習を行った。

図 4 に、各エポックにおける実行時間と学習結果の評価指標である BLEU (BiLingal Evaluation Understudy)を示す。まず 32 ノード 64 GPU での実行では、12 エポックを超えたところで 24 時間に達し、実行を打ち切った[2]。64 ノード 128 GPU では 15 エポックの学習を行うことができた。いずれも BLEU としては 23.5%の値を得ており、妥当な結果である。一方、120 ノード 240 GPU での実行を行ったが、スクリプトのバグによって正しく動作しなかった[3]。

7．おわりに

Chainer MN を用いて、Reedbush-H システム全系の 240 GPU に至るまで高い strong-scalability が得られることを示した。学習結果についても、ノード数が増加して全体のバッチサイズが増えていても、学習率の調整が有効に働き、GPU 数が異なる場合でも同一エポック数ではほぼ同等の精度が得られていることがわかる。

本稿の内容が Reedbush-H, Reedbush-L における深層学習に興味をお持ちの方々へのヒントになれば幸いである。

謝辞

Preferred Networks 社の福田 圭祐、鈴木 脩司の両氏には多くの有益なご助言をいただきました。深く感謝いたします。また、本研究の一部は、学際大規模情報基盤共同利用・共同研究拠点、および、革新的ハイパフォーマンス・コンピューティング・インフラの支援による(課題番号: jh170036-DAH)。

参 考 文 献

[1] Chainer, a powerful, flexible, and Intuitive Framework for Neural Networks, https://chainer.org

[2] ImageNet, http://www.image-net.org

[3] マルチノードでの分散学習パッケージ ChainerMN 正式版 v 1.0.0 をリリース、https://www.preferred-networks.jp/ja/news/pr20170901

[4] Akiba T, Fukuda K, Suzuki S, ChainerMN: scalable distributed deep learning framework, arXiv: 1710.11351, 2017

[5] 秋葉 拓哉、「ChainerMN による分散深層学習の性能について」https://research.preferred.jp/2017/02/chainermn-benchmark-results/

[6] Kaiming He et al., “Deep Residual Learning for Image Recognition”, CVPR 2016.

[7] Cupy, http://cupy.chainer.org/

[8] 塙 敏博,「Reedbush スーパーコンピュータを用いたディープラーニング」, GPU テクノロジーカンファレンス, 2017 年 12 月

[9] WMT15 Translation Task, http://www.statmt.org/wmt15/translation-task.html

[2] データセットを直接ロードする代わりに IME 上に置いたキャッシュを利用しているが、それでも training を始めるまでのセットアップに 1 時間以上を要しており、13 エポック実行が終わる前に 24 時間に達している。グラフでは training を始めてからの経過時間を示している。

[3] その後 ChainerMN の最新版を取り込むなどして正しく動作するようになったが、大規模 HPC チャレンジ中には実行が終了しなかった。