Vol.2018-HPC-166 No.7
2018/9/27

AI橋渡しクラウドABCIの性能評価

佐藤　仁[1,a)]　溝手　竜[1]　滝澤　真一朗[1]

概要：2018 年 8 月に稼働を開始した産総研 AI 橋渡しクラウド ABCI は，従来の高性能計算のみに特化した典型的なスーパーコンピュータとは異なり，高性能計算の要素に加えて AI/ビッグデータ処理に特化した設計が行われている．本稿では，ABCI のシステムのうち，GPU を搭載した計算ノードに対し AI/ビッグデータ処理に関する性能評価を行った結果を報告する．

1. はじめに

近年，AI やビッグデータ処理においても高性能計算の必要性が著しく高まっている．とりわけ，深層学習は，自動車の自動運転，製造業，ロボット，医療，創薬，金融など様々な分野への応用が期待されており，アルゴリズム (Algorithm Theory) の進展だけでなく，大量のビッグデータ (Big Data) を蓄えるストレージ技術や，それらのデータに対して高速に処理する計算能力（Computation) など三位一体となった解決が求められている．

AI 橋渡しクラウド ABCI [1] は，世界最高水準の 0.550 EFlops(FP16) の機械学習処理能力，37 PFlops(FP64, Peak), 19.88 PFlops(FP64, Real) の高性能計算能力及び 12.054 GFlops/Watt の省電力性を備え，アルゴリズム・ビッグデータ・計算能力の協調による高度な AI/ビッグデータ処理を可能にする大規模クラウド基盤であり，2018 年 6 月末に産業技術総合研究所に導入され，2018 年 8 月に稼働を開始した．ABCI は，AI の研究開発のためのオープンなリーディングインフラストラクチャとして，画像認識・音声認識・自然言語処理等の種々の機械学習アルゴリズムやデータモデルの高度化，自動車・ロボットの自動運転・制御のアルゴリズム開発，創薬向け化合物推定，音声対話・自動翻訳のアルゴリズム開発等，幅広い分野でのアプリケーション創出が期待されている．

ABCI は従来の高性能計算のみに特化した典型的なスーパーコンピュータとは異なり，高性能計算の要素に加えて AI/ビッグデータ処理に特化した設計が行われている．一方で，実機での具体的な性能値は明らかではない．本稿は，ABCI のシステムのうち，GPU を搭載した計算ノードに対し AI/ビッグデータ処理に関する性能評価を行った結果を報告する．

[1] 国立研究開発法人産業技術総合研究所
[a)] hitoshi.sato@aist.go.jp

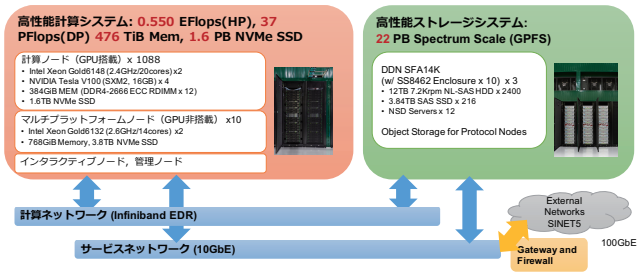


図 1　ABCI の全体構成

2. AI橋渡しクラウド ABCI

2.1 概要

図 1 に ABCI の全体構成を示す．ABCI は，GPU を搭載する 1088 台の計算ノードをメインとして，多目的用途の 10 台のマルチプラットフォームノード，ログイン用途の 4 台のインタラクティブノード，その他管理サーバ等からなる高性能計算システム，22PB の Spectrum Scale (GPFS) のストレージ領域からなる大容量ストレージシステム，Infiniband EDR からなる計算用途の計算ネットワーク，10 Gbps Ethernet からなる外部接続用途のサービスネットワークを含むネットワークインターコネクト等から構成される．本稿では特にベンチマークに関連する部分について焦点を当てて説明する．

2.2 計算ノード

図 2 に計算ノードの構成を示す．計算ノードは，GPU を搭載する典型的なスーパーコンピュータと同様の構成であり，演算装置として最新鋭の CPU と GPU を搭載し，大容量の主記憶装置，広帯域低遅延なネットワークインターコネクトやフラッシュストレージ等から構成される．CPU は Intel Xeon Gold 6148 プロセッサ（Skylake-SP, 2.4GHz, 20

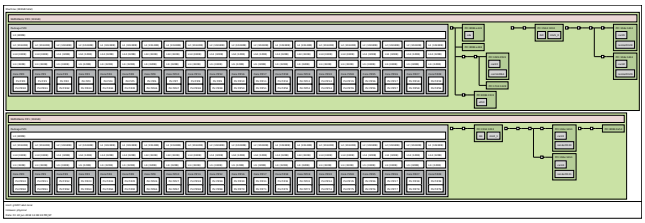


図 2　計算ノードの構成

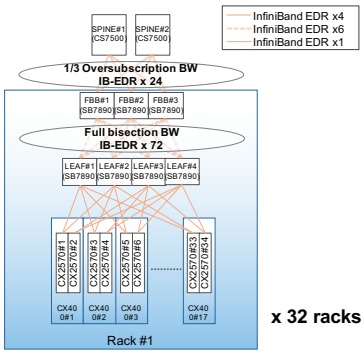


図 3　1 ラック内のネットワーク構成

cores）を 2 基搭載し，DDR4-2666 ECC RDIMM 32 GiB モジュールを 12 枚により容量 384 GiB, バンド幅 256 GB/s となる．GPU は NVIDIA Tesla V100 (SXM2, 16 GiB) を 4 基搭載する．ネットワークインターコネクトは Infiniband EDR の HCA(Host Channel Adapter) を 2 ポート有し，理論インジェクションバンド幅は片方向 200 Gbps, 双方向 400 Gbps となる．また，ローカルストレージとして，容量 1.6 TB の Intel DC P4600 の NVMe SSD を有する．

計算ノードは 34 台を束ねて 1 ラックとして構成され，次節の 2.3 で述べるように，ラック内で理論インジェクションバンド幅が 200 Gbps でフルバイセクションバンド幅となるように Fat Tree 構成の Infiniband EDR ネットワークで接続される．従って，典型的なユースケースではラック単位で分散深層学習の処理を大量に流すことを意図しており，複数ラックに跨る計算ノードを用いる大規模ジョブを実行する際はネットワーク通信の最適化が必要となる．

2.3　ネットワークインターコネクト

計算ノードやマルチプラットフォームノード，インタラクティブノード，管理サーバ等のその他のノード，大容量ストレージシステム等は Infiniband EDR で構成される Fat Tree 構成のネットワークで相互接続されている．計算ノード同士は，1 ラック内の計算ノード間は，**図 3** で示すように，理論インジェクションバンド幅が 200 Gbps でフルバイセクションバンド幅となるように接続され，ラックを跨ぐ計算ノード間は 1/3 のオーバーサブスクリプションバンド幅となるように接続されている．また，マルチプラットフォームノード，インタラクティブノード，管理サーバ等のその他のノードや大容量ストレージシステムはフルバイセクションバンド幅となるよう接続されている．一部のノードは外部接続用のネットワークであるサービスネットワークに 10Gbps Ethernet で接続されているが本稿では詳細を省く．

2.4　ストレージ

ABCI は性能，容量，用途に応じて，複数のストレージ階層を設けている．

1 つ目の階層は，計算ノードに接続された NVMe SSD からなるローカルストレージである．計算ノードには，ローカルストレージとして，容量 1.6TB の Intel DC P4600 の NVMe SSD が接続されており，サイズの小さなファイルへの大量の I/O に向いている．また，これらのローカルストレージは，ジョブスケジューラが計算ノードを割り当てられた際に，分散ファイルシステムである BeeGFS [2] OnDemand (BeeOND) [3] により共有ストレージを構成し，シングルシステムイメージでボリュームを利用することができる．

2 つめの階層は，大容量ストレージシステム上に構成される並列ファイルシステム Spectrum Scale (GPFS) である．10 台のエンクロージャ (SS8462) が接続された DDN SFA 14K を 3 セット用いることにより 22 PB からなる共有ストレージ領域を構成している．典型的な POSIX ファイル I/O の他，MPIIO などの並列 I/O によるバースト I/O に向いている．計算ノードやマルチプラットフォームノード，インタラクティブノード等からネットワークインターコネクトを介してシングルシステムイメージでボリュームを利用することができる．

3 つめの階層は，大容量ストレージシステムの Spectrum Scale (GPFS) の一部に OpenStack Swift を用いて構成されるオブジェクトストレージである．ABCI の計算ノードや ABCI 外のインターネット上の計算機から Amazon S3 互換のインターフェースによるオブジェクトへのアクセスを提供することができる．今回は，オブジェクトストレージに対するベンチマークは行っていないため，本稿では詳細を省く．

3.　ベンチマークによる性能評価

ABCI のシステムのうち，GPU を搭載した計算ノードに対し AI/ビッグデータ処理に関する性能評価を行った．ベンチマークとしては，演算性能，メモリバンド幅性能，ネットワーク通信性能，ストレージ I/O 性能，そして，分散深層学習の性能について行った．

計算ノードの OS は CentOS 7.4.1708 で構成され，Linux のカーネルは v3.10.0 である．特別な記載がない限り，各種ベンチマークプログラムのビルドには OS のデフォルトのコンパイラである gcc v4.8.5 を用いた．

3.1 演算性能

演算性能を計測するために BLAS(Basic Linear Algebra Subprograms) ライブラリの GEMM(General Matrix Multiplication) カーネルを GPU と CPU を対象に実行した．GPU に対しては，CUDA Toolkit 9.2.88.1 に付属する cuBLAS [4] を用いて倍精度 (FP64)，単精度 (FP32)，半精度 (FP16) の GEMM を実行した．CPU に対しては，Intel Math Kernel Library 2018.2.199 の CBLAS [5] を用いて倍精度 (FP64)，単精度 (FP32) の GEMM を実行した．

図 4 に単体の GPU での実行結果を示す．x 軸は行列のサイズ ($M = N = K$) を表し，y 軸は演算性能 [TFlops] を表す．単精度と半精度 (HGEMM(FP16)) の実行では CUDA Core を用いた実行の他に，Tensor Core を用いた実行も行った．CUDA Core を用いた場合，演算性能の理論値は，倍精度で 7.8 TFlops，単精度で 15.7 TFlops，半精度で 31.3 TFlops (= 1530 MHz × 80 SMs × 64 cores/SM × 4 ops/clock / 1000000) となるが，結果より各々最大で 7.2 TFlops ($M = N = K = 3264$)，14.2 TFlops ($M = N = K = 9216$)，29.8 TFlops ($M = N = K = 13056$) となり，概ね理論値に対して各々 92.3 %，90.4 %，95.2 %の性能となることを確認した．さらに，Tensor Core を用いた場合は，演算性能の理論値は半精度で 125 TFlops となるが，結果より最大で 106 TFlops ($M = N = K = 13056$) となり，概ね理論値に対して 84.8 %の性能となることを確認した．GPU に関しては，TensorCore に向く FP16 の FMA(Fused Multiply-Add) 演算の場合に特に高速化されることが伺える．今回，FP32 に対しても TensorCore による実行を行っているが内部で FP16 に変換して演算するため精度に問題がある可能性があるのに注意されたい．

次に，**図 5** に 1 socket の CPU での結果を示す．CPU の実行では，SSE4.2, AVX, AVX2, AVX512 による SIMD 最適化を行った．CPU 単体の演算性能の理論値は，AVX512 の最適化を行い Turbo Boost を用いた場合，倍精度で 1.4 TFlops (= 2200 MHz × 20 cores × 32 ops/clock / 1000000)，単精度で 2.8 TFlops (= 2200 MHz × 20 cores × 64 ops/clock / 1000000) となるが，各々最大で 1.1 TFlops，2.3 TFlops となり，概ね理論値に対して各々 78.6 %，82.1 %となることを確認した．1 ノード全体となる 2 socket の CPU の実行では，**図 6** のようになり，AVX512 の最適化を行った場合に各々最大で 2.1 TFlops，4.3 TFlops となり，概ね理論値に対して各々 75.0 %，76.8 %となることを確認した．CPU に関しては，概ね妥当な性能であることが伺える．

3.2 メモリバンド幅性能

GPU と CPU のメモリバンド幅を計測するために STREAM ベンチマークを実行した．STREAM ベンチマークは，

Copy: 配列のコピー $a(i) = b(i)$

Scale: 配列とスカラ値の乗算 $a(i) = q * b(i)$

Add: 配列の加算 $a(i) = b(i) + c(i)$

Triad: Scale と Add の組み合わせ $a(i) = b(i) + q * c(i)$

を計測する．今回は，バージニア大学のサイトより入手したコード [6] を基に，GPU に対しては OpenACC により並列化を行い，CPU に対しては OpenMP により並列化を行った実装を用いて計測した．GPU に対する OpenACC による並列化では，文献 [7] で述べられているように，C 言語版のオリジナルのコードを基に OpenACC の指示文を追加した．コンパイルには PGI Compiler 18.5 を使用し，コンパイルオプションには，`-O2 -acc -ta=tesla:cc70` を指定している．ここで，`-acc` は OpenACC を有効，`-ta=tesla:cc70` は Nvidia Tesla V100 GPU 向けの最適化のためのオプションである．CPU の OpenMP による並列化はデフォルトの gcc v4.8.5 を用いた．

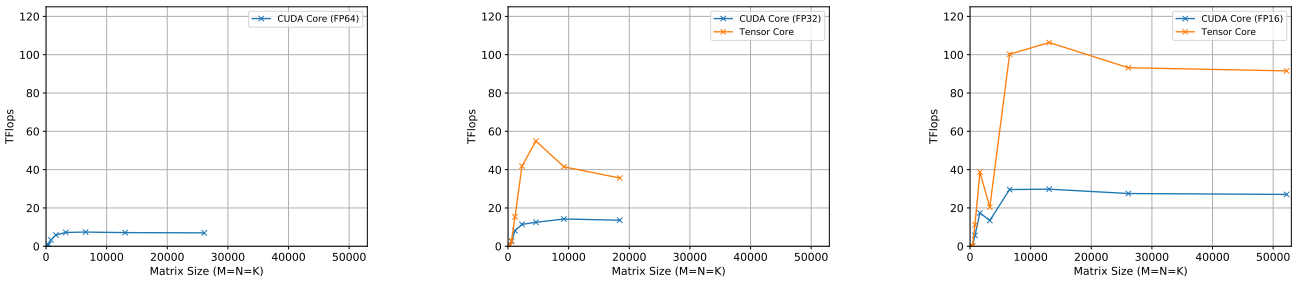
図 7 に結果を示す．GPU 単体のメモリバンド幅の理論値は 900 GB/sec であるが，**図 7(a)** に示す単体の GPU の実行では，Triad で 840 GB/sec となり，理論値に対して 93.3 %の性能を達成することを確認した．また，CPU のメモリバンド幅の理論値は 1 socket あたり 128 GB/sec，2 socket で 256 GB/sec であるが，**図 7(b)** に示す CPU の実行では，Triad で 1 socket あたり 89.9 GB/sec，2 socket で 115 GB/sec となり，理論値に対して各々 70.2 %，44.9 %となることを確認した．

3.3 ネットワーク通信性能

3.3.1 OSU Micro-Benchmarks

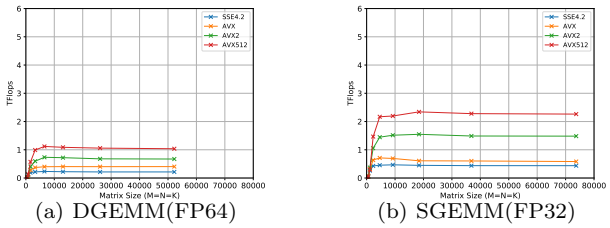
ネットワーク通信性能を計測するために OSU Micro-Benchmarks [8] を実行した．今回は，OSU Micro-Benchmarks v5.4.2 を用いて，計算ノードの CPU ホスト間及び GPU デバイス間のネットワークのバンド幅，遅延を計測した．使用した MPI 実装は，OpenMPI v2.1.3 で，1 台の計算ノードに 1 プロセスを割り当て 2 ノード間で計測を行った．

図 8 にネットワークのバンド幅の性能の結果を示す．x 軸は各プロセスが送信するデータサイズ [Bytes] を表し，y 軸はバンド幅 [GB/sec] を表す．計算ノードからネットワークインターコネクトへは 2 ポートの Infiniband EDR HCA により接続されており，理論値は 1 ポートあたり 12.5 GB/sec である．CPU ホスト間の通信では，データサイズが 1 MiB 以上の場合に概ね理論値に近い性能を達成することを確認した．一方，GPU デバイス間の通信では，GPU Direct による GPU デバイスと Infiniband HCA 間の直接通信 (GDRDMA)，パイプライン転送による通信 (Pipeline)，両者の手法を組み合わせたハイブリッド通信 (Hybrid) の 3 種類の手法を試した．ハイブリッド通信では 30000 bytes を境にアルゴリズムの切り替えが起きる．データサイズが小さい場合，具体的には 8 KiB 以下の場



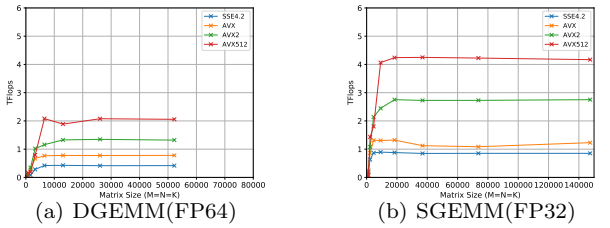
(a) DGEMM(FP64) (b) SGEMM(FP32) (c) HGEMM(FP16)

図 4 GEMM(GPU) の性能



(a) DGEMM(FP64) (b) SGEMM(FP32)

図 5 GEMM(CPU, 1 Socket) の性能



(a) DGEMM(FP64) (b) SGEMM(FP32)

図 6 GEMM(CPU, 2 Sockets) の性能

合は，GPU Direct による直接通信の性能が良好であるが，8 KiB を超えたあたりからパイプライン転送による通信の性能が良好であることを示している．ハイブリッド通信では，アルゴリズム切り替えにより，どのデータサイズにおいても良好な性能を示している．ただ，2 MiB を境に性能が低下しており，この原因は現在解析中である．

次に，**図 9** に遅延の結果を示す．x 軸は各プロセスが送信するデータサイズ [bytes] を表し，y 軸は遅延 [msec] を表す．概ね妥当な性能を示すことを確認した．

3.3.2 NCCL

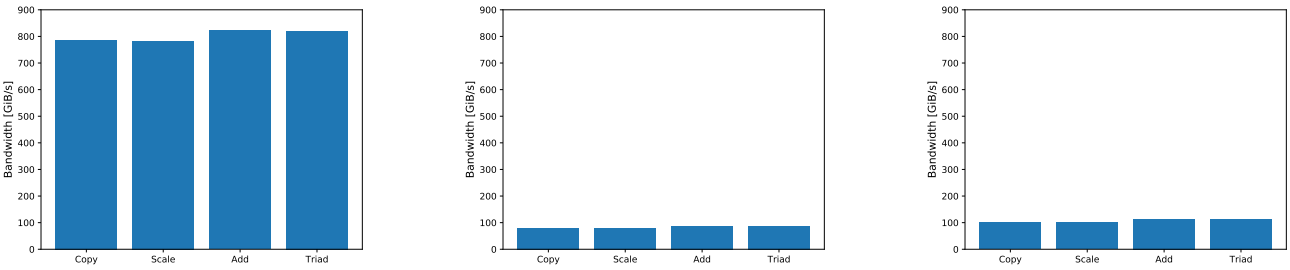
多くの分散深層学習フレームワークのネットワークの集団通信の実装では，NVIDIA Collective Communications Library (NCCL) [9] が広く使用されている．そこで，NCCL による GPU 間の通信性能を計測するために，nccl-tests [10] を用いて AllReduce, AllGather, ReduceScatter を実行した．1 プロセスに 1 台の GPU，1 台の計算ノードに 4 プロセスを割り当て，計算ノードの台数を 1 から 32 台まで変えて計測した．使用した NCCL のバージョンは v 2.2.13-1 である．

図 10 に結果を示す．x 軸は 1 台の GPU が割り当てられたプロセスが送信したデータサイズ [Bytes] を表し，y 軸はバンド幅 [GB/sec] を表す．プロセスあたり 1 GiB のデータサイズにおいて，1 ノードでは，AllReduce で 124.12 GB/sec，AllGather で 114.03 GB/sec, ReduceScatter で 110.68 GB/sec であるのに対し，32 ノードでは，AllReduce で 12.12 GB/sec, AllGather で 12.07 GB/sec, ReduceScatter で 12.09 GB/sec となった．1 台の計算ノードには 4 基の GPU が搭載されており，各々の GPU 間は 1 本あたり 25 GB/sec の NVLink が 2 本で接続されている．1 ノードでの実行では，NCCL により計算ノード内の GPU 間の接続を考慮して自動的に 12 本の Ring を生成し，各々の GPU 間では Ring を用いて通信を行っている．一方，複数ノードでの実行では，計算ノード間は，1 ポートあたり 12.5 GB/sec の Infiniband EDR HCA を 2 ポートで介して接続されており，NCCL により計算ノード間のネットワークや計算ノード内の GPU 間の接続を考慮して自動的に 2 本の Ring を生成し，各々の GPU 間では Ring を用いて通信を行っている．このため，1 ノードの実行と複数ノードの実行での性能の大幅な乖離はこのためである．ただし，現状の NCCL では ABCI のネットワークに最適な Ring が生成されていないため，多少の性能向上は期待できると考えている．

3.4 ストレージ I/O 性能

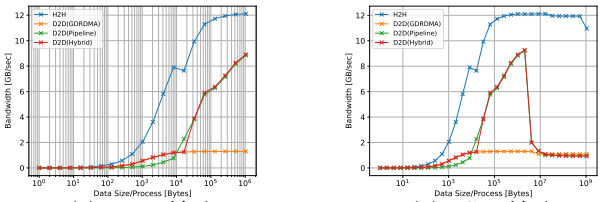
ストレージの I/O スループット性能を計測するために，IOR ベンチマーク [11] を実行した．今回は，計算ノードをクライアントとして，クライアント毎に 8 GiB のファイルを生成し，POSIX による WRITE, READ の逐次 I/O の性能をクライアントとなる計算ノードの台数を 1 から 32 台まで変えながら計測した．対象とするストレージは大容量ストレージシステムに構成された Spectrum Scale (GPFS) と計算ノードの NVMe SSD のローカルストレージを集約して構成された BeeOND とした．

Spectrum Scale に対して実行した結果を**図 11(a)** に示す．x 軸は計算ノードの台数を表し，y 軸はスループット



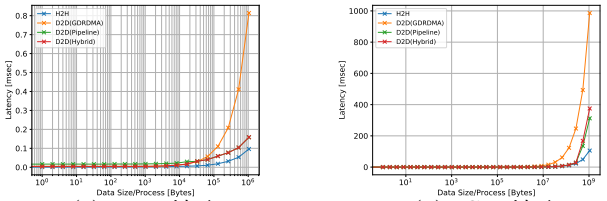
(a) GPU (b) CPU (1 Socket) (c) CPU (2 Sockets)

図 7 STREAM の性能



(a) 1MiB 以下 (b) 1GiB 以下

図 8 OSU Micro Benchmarks によるネットワーク性能 (バンド幅)



(a) 1MiB 以下 (b) 1GiB 以下

図 9 OSU Micro Benchmarks によるネットワーク性能 (遅延)

[GiB/sec] を表す．クライアントとなる計算ノードの台数が増加するにつれ，スループット性能が向上していることが伺える．最終的には，計算ノードが 32 台のとき，READ で 17.8 GiB/sec，WRITE で 23.1 GiB/sec の性能なることを確認した．また，BeeOND に対して実行した結果を**図 11(b)** に示す．クライアントとなる計算ノードの台数が増加しても，スループット性能は一定であった．具体的には，計算ノードが 32 台のとき，READ で 1.16 GiB/sec，WRITE で 2.74 GiB/sec の性能なることを確認した．

次に，ファイルシステムのメタデータ性能を計測するために，mdtest [11] を実行した．1 ノードあたり 2 プロセスを割り当て，プロセスあたり 25000 個のファイルないしディレクトリを対象にメタデータ操作を行った．使用した mdtest のバージョンは v 1.9.3 である．

図 12 に結果を示す．x 軸は計算ノードの台数を表し，y 軸は IOPS [KIOPS] を表す．Spectrum Scale ではノード数が増えるにつれメタデータ性能が向上するが，BeeOND ではノード数が増えても一定のメタデータ性能を示すことを確認した．

3.5 分散深層学習

分散深層学習の性能を計測するために ImageNet のデータセット [12] に対して ResNet-50 [13] による学習を行った．今回は，ChainerMN [14] を用いて，64 台の計算ノード (256 GPU) で実行した．ChainerMN は v1.3.0 を使用し，内部的に Chainer v4.2.0, CuPy v4.2.3, mpi4py v3.0.0, Python v3.6.5 を使用している．パラメタ等の設定は [15] の手法と同等であり，バッチサイズを GPU あたり 32，全体で 8192 とし，学習率を 0.1 から warm up scheduling で 5 epoch で $0.1 \cdot \frac{k \cdot n}{256}$ (ただし，$k = 256$ は GPU の台数，$n = 32$ は GPU あたりのバッチサイズ) と線形に増加させた後，30, 60, 80 epoch 毎に 0.1 倍とし，Momentum SGD (momentum=0.9) の最適化を用い，Weight Decay を 0.0001 として 100 epoch の学習を行った．

図 13 に結果を示す．x 軸は実行時間 [sec] を表し，y 軸は学習 (Training) 及び検証 (Validation) データに対する精度を表す．特別な最適化を行わず，また，検証の処理を含めて，100 epoch を実行に 4124 秒かかり，学習データに対する精度は 75.6 %，検証データに対する精度は 74.5 %程度であった．概ね良好な性能であることが伺える．

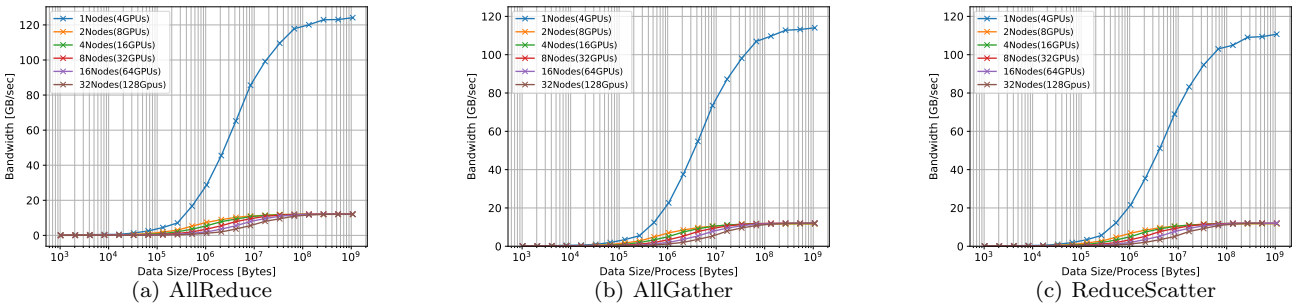
4. おわりに

本稿では，ABCI のシステムのうち，GPU を搭載した計算ノードに対し AI/ビッグデータ処理に関する性能評価を行った結果を行った．今後の課題としては，より大規模な構成での実アプリケーションを対象にした詳細な性能評価などが挙げられる．

謝辞 本研究の一部は JSPS 科研費 18K11332 及び NEDO 次世代人工知能・ロボット中核技術開発の助成を受けて実施した．

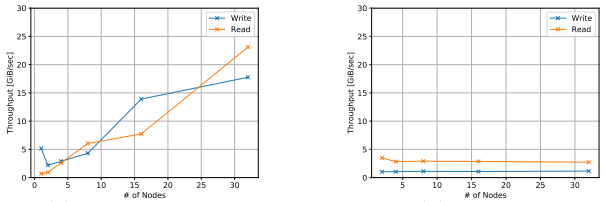
参考文献

[1] 小川宏高，松岡聡，佐藤仁，高野了成，滝澤真一朗，谷村雄輔，三浦信一，関口智嗣：世界最大規模のオープン AI インフラストラクチャ AI 橋渡しクラウド （ ABCI ）の概要，情報処理学会研究報告 Vol.2018-HPC-165 No.19, pp. 1–7 (2018).
[2] ThinkparQ: BeeGFS, https://www.beegfs.io/



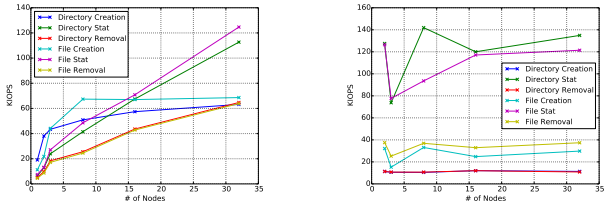
(a) AllReduce (b) AllGather (c) ReduceScatter

図 10 nccl-tests の性能



(a) Spectrum Scale (b) BeeOND

図 11 IOR の性能



(a) Spectrum Scale (b) BeeOND

図 12 mdtest の性能

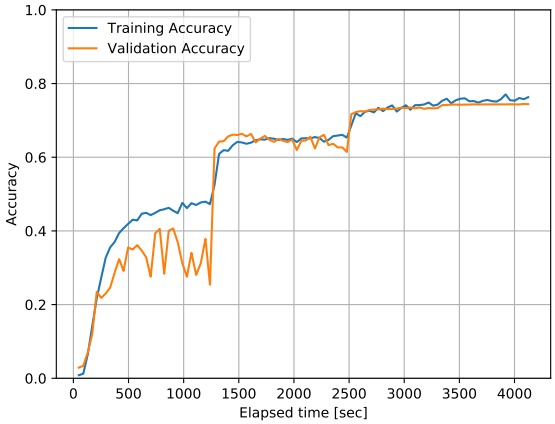


図 13 分散深層学習の性能

content/.

[3] ThinkparQ: BeeOND, https://www.beegfs.io/wiki/BeeOND.

[4] NVIDIA: cuBLAS, https://developer.nvidia.com/cublas.

[5] Intel: Intel Math Kernel Library, https://software.intel.com/en-us/mkl.

[6] McCalpin, J. D.: STREAM: Sustainable Memory Bandwidth in High Performance Computers, https://www.cs.virginia.edu/stream/.

[7] 塙敏博, 星野哲也, 中島研吾, 大島聡央, 伊田明弘：GPU 搭載スーパーコンピュータ Reedbush-H の性能評価，情報処理学会研究報告 Vol.2017-HPC-159 No.9，pp. 1–6 (2017).

[8] Panda, D. K.: OSU Micro Benchmarks, http://mvapich.cse.ohio-state.edu/benchmarks/ (2001-2017).

[9] NVIDIA: NVIDIA Collective Communications Library (NCCL), https://developer.nvidia.com/nccl.

[10] NVIDIA: nccl-tests, https://github.com/NVIDIA/nccl-tests.

[11] hpc: HPC IO Benchmark Repository, https://github.com/hpc/ior.

[12] Fei-Fei, L.: ImageNet, http://www.image-net.org/.

[13] He, K., Zhang, X., Ren, S. and Sun, J.: Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778 (online), DOI: 10.1109/CVPR.2016.90 (2016).

[14] Akiba, T., Fukuda, K. and Suzuki, S.: ChainerMN: Scalable Distributed Deep Learning Framework, *Proceedings of Workshop on ML Systems in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1–7 (2017).

[15] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Kaiming, Y. J. and Facebook, H.: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, *arXiv.org e-Print archive*, pp. 1–12 (2017).