ABCI

ABCIミニキャンプ

ABCIの使い方・お役立ち情報

2021年5月12日

国立研究開発法人 産業技術総合研究所
情報・人間工学領域
ディジタルアーキテクチャ研究センター

目　次

1. 計算リソース
   - ABCI 2.0と計算ノード(A)
   - 利用可能な資源の説明
2. 計算ノード(A)の使い方
   - 計算ノード(A)の資源タイプ
   - 計算ノード(A)利用時の注意点
3. ミニキャンプでの基本操作・お役立ち情報
   - ジョブの投入 (qsub, qrsh)
   - NVIDIA A100を使用するPytorch, TensorFlowの導入事例
   - クラウドストレージの利用、など

ABCI 2.0へのアップグレード

2021年5月10日13時よりABCI 2.0の運用を開始

https://www.aist.go.jp/aist_j/news/au20210510_2.html

ABCI 2.0全体概要

今回追加部分

計算リソース

計算ノード(A) x 120台
- CPU / Intel Xeon Platinum 8360Y Processor 2.4 GHz, 36 Cores x 2
- GPU / NVIDIA A100 SXM4 x 8
- Memory / 512GiB
- NVMe SSD / Intel SSD DC P4610 1.6TB x 2
- Interconnect / InfiniBand HDR x 4

計算ノード(V) x 1088台
- CPU / Intel Xeon Gold 6148 x 2
- GPU / NVIDIA V100 SXM2 x 4
- Memory / 384GiB
- NVMe SSD / Intel SSD DC P4600 1.6TB x 1
- Interconnect / InfiniBand EDR x 2

メモリインテンシブノード x 10台
- CPU / Intel Xeon 6132 x 2
- Memory / 768GiB
- SSD / Intel SSD DC S4500 1.9TB x 1
- Optane SSD / Intel Optane SSD P4800X 1.5TB x 2
- Interconnect / InfiniBand EDR x 2

共有ファイルシステム

DDN SFA14KX (w/ SS8462 Enclosure x 10) x 3
- 12TB NL-SAS HDD x 2400
- 3.84TB SAS SSD x 216

DDN SFA14KX (w/ SS9012 Enclosure x 5) x 1
- 7.68TB SAS SSD x 216

DDN ES7990X (w/ SS9012 Enclosure x 2) x 3
- 18TB NL-SAS HDD x 801

DDN ES400NVX x 3
- 7.68TB NVM2 SSD x 69

ABCIクラウドストレージ

HPE Apollo 4510 Gen10 x 24
- 12TB SATA HDD x 1440

インタラクティブノード / ゲートウェイノード

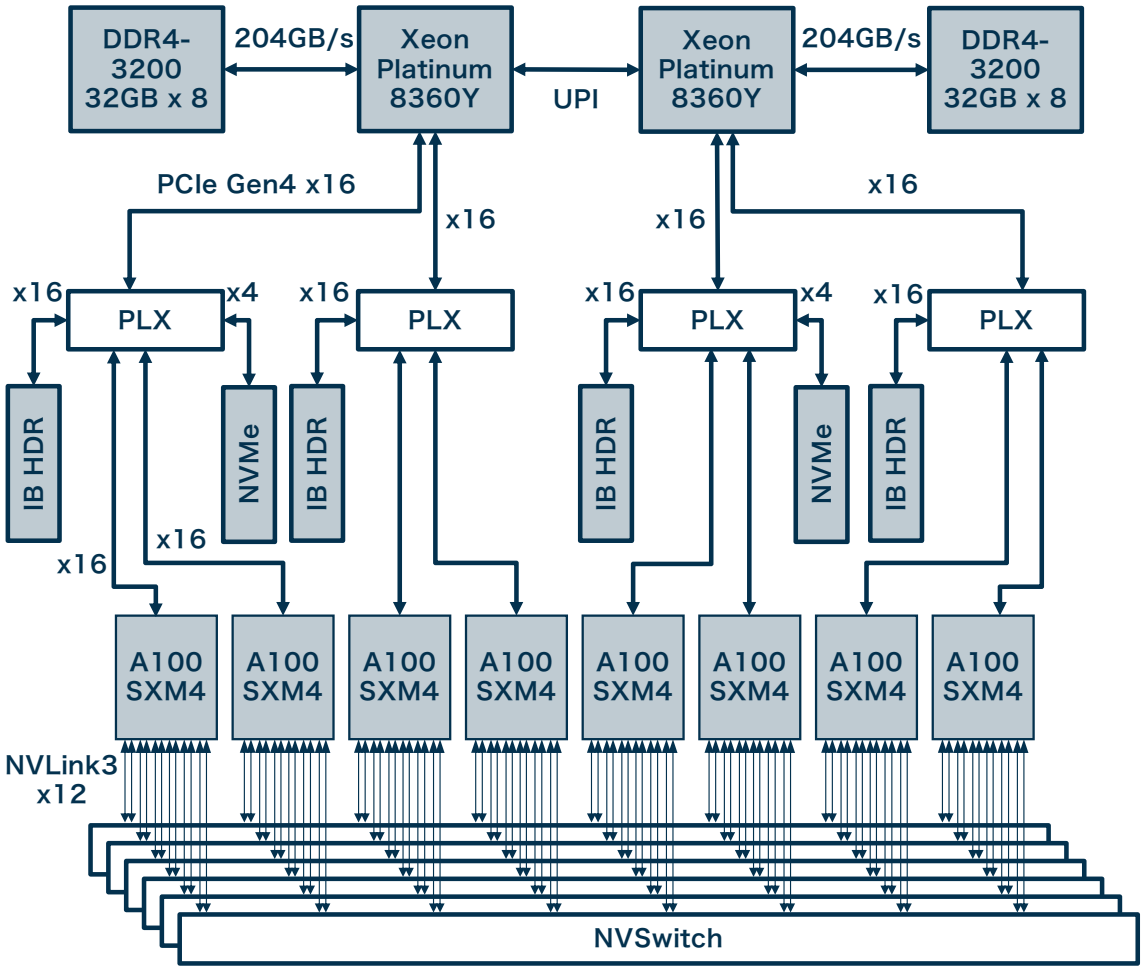
ファイアウォール

SINET5

計算ネットワーク (InfiniBand EDR/HDR)

サービスネットワーク (10GbE)

ABCI Compute Node (A)

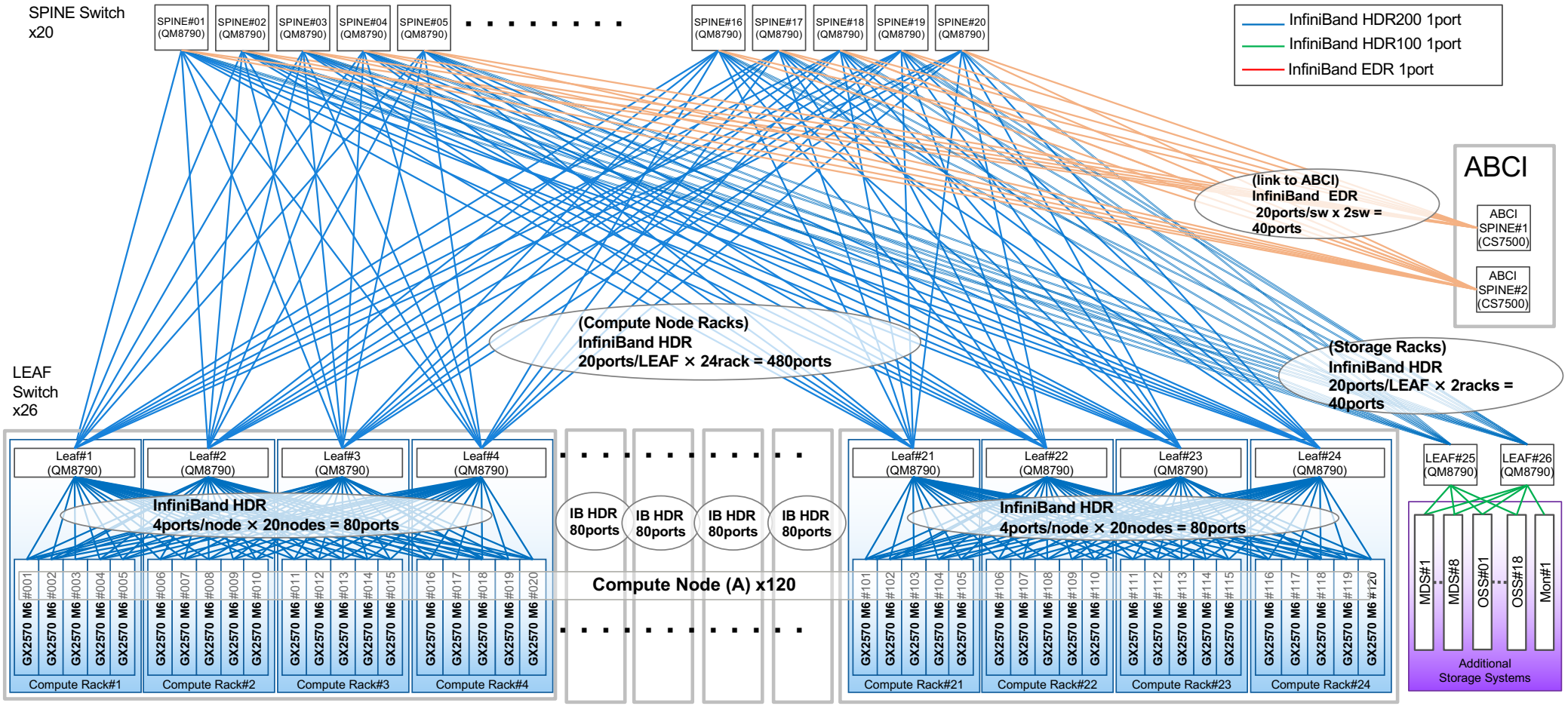


FUJITSU PRIMERGY GX2570 (4U) x 120 nodes	
CPU	Intel Xeon Platinum 8360Y Processor 2.4 GHz, 36 Cores
GPU	NVIDIA A100 SXM4 (40GiB HBM2) x8
Memory	512GiB DDR4 3200MHz RDIMM
NVMe SSD	Intel SSD DC P4510 2TB x2
Interconnect	InfiniBand HDR (200Gbps) x4

	Compute Node (V)	Compute Node (A)	Scale-up
HPC (FP64)	34 TF	156 TF	x 4.6
DL Training (FP32/TF32)	68 TF	1.25 PF	x 18
DL Training (FP16/BF16)	500 TF	2.5 PF	x 5
FP16 w/ Sparsity	500 TF	5 PF	x 10
GPU memory	64 GiB	320 GiB	x 5
Injection BW	200Gbps	800 Gbps	x 4

ABCI Compute Network (A)

Full-bisection & SHARPv2-enabled network



AIST ABCI 2.0 Upgrade: 計算リソース増強の観点

ABCI 2.0 (2021 May-)

ABCI 1.0 (2018Q2-)
550 PF (FP16), 37.2 PF (FP64)
476 TiB Memory, 1.74 PB NVMe SSD

Compute Nodes (V) x 1088

GPU	NVIDIA Tesla V100 SXM2 x 4
CPU	Intel Xeon Gold 6148 (2.4GHz/20cores) x 2
Memory	384 GiB
Local Storage	Intel SSD DC P4600 (NVMe) 1.6TB x 1
Interconnect	InfiniBand EDR x 2 (25 GB/sec)

+

ABCI Expansion (2021Q2-)
300 PF (FP16), 19.3 PF (FP64)
97.5 TiB Memory, 384 TB NVMe SSD

Compute Nodes (A) x 120

GPU	NVIDIA A100 x 8
CPU	Intel Xeon SP (Ice Lake) x 2
Memory	512 GiB
Local Storage	Intel SSD DC P4610 (NVMe) 1.6TB x 2
Interconnect	InfiniBand HDR x 4 (100 GB/sec)

Precision	ABCI 1.0	今回追加	ABCI 2.0 (合算)	スケールアップ	用途
FP32/TF32	75 PF	150 PF	**225 PF**	x 3	高精度DL
TF32 w/ Sparsity	↑(*1)	300 PF	**375 PF**	x 5	高精度DL
FP16/BF16	550 PF	300 PF	**850 PF**	x 1.55	低精度DL
FP16/BF16 w/ Sparsity	↑(*1)	600 PF	**1.15 EF**	x 2.09	低精度DL
FP64	37.2 PF	19.3 PF	**56.5 PF**	x 1.52	シミュレーション

(*1) V100はSparsityをサポートしないため参考値。

ミニキャンプで利用する計算機資源

- ABCIグループ gaa50123 を利用してジョブ投入ください
  - 5/31まで1チーム当たり「250ポイント」を上限に、計算ノードを利用可能

```
[aaa12345xx@es-a1 ~]$ qsub -g gaa50123 -l rt_AF=1 sample.sh
        # gaa12345: グループ名, rt_AF=1 : 計算ノード(A)1台, sample.sh: ジョブスクリプト
```

  - ただし、ポイントは自己管理でお願いします。
  - 計算ノード(A) x 10の予約を作成済み（AR_ID: 3686）。主に並列ジョブを投入する際にご利用ください

```
[aaa12345xx@es-a1 ~]$ qsub -g gaa50123 -ar 3686 sample.sh
        # gaa12345: グループ名, 3686 : 予約ID, sample.sh: ジョブスクリプト
```

- クラウドストレージ
  - 最大10TBのクラウドストレージ/チームを利用可能
  - 5/31まで。期間内に別の場所（他ABCIグループのグループ領域 or クラウドストレージ）にデータを移動ください

2. 計算ノード(A)の使い方

計算ノード(A)の資源タイプ

2種類の資源タイプを利用可能

＊Spotで最大64、On-demandで最大4

資源タイプ	資源タイプ名	説明	割り当て物理CPUコア数	割り当てGPU数	メモリ(GiB)	ローカルストレージ(GB)	資源タイプ課金係数	割当可能な個数
Full	rt_AF	ノード占有GPU利用	72	8	480	3440	3.00	1−64 (1−4)*
AG.small	rt_AG.small	ノード共有GPU利用	9	1	60	390	0.50	1

- On-demand (インタラクティブジョブ), Spot (バッチジョブ)にて利用可能
- ジョブ投入時には資源タイプ名と数量を指定 (例: -l rt_AF=1)

ジョブ毎の実行時間・ノード時間積制限

	On−demand（上限/デフォルト）	Spot（上限/デフォルト）
rt_AF	12:00:00/1:00:00	72:00:00/1:00:00
rt_AG.small		168:00:00/1:00:00
ノード時間積（rt_AFのみ）	12	288

計算ノード(A)利用時の注意点 (1/2)

- 既存の計算ノード(V)とはOSが異なるため、環境の再構成が必要
  - 計算ノード(V): CentOS 7.5
  - 計算ノード(A): RHEL 8.2
- 計算ノード(A)からは、グループ領域には /groups/GROUP_NAMEでアクセス
  - 計算ノード(A)で利用するデータはHOME, /groups/GROUP_NAME以下に置いてください
    - 従来の /groups1/GROUP_NAME, /groups2/GROUP_NAMEにあるデータにはアクセスできない
- 計算ノード(A)用のプログラムは専用のインタラクティブノードで開発
  - インタラクティブノード名: es-a1, es-a2
    - OSはRHEL 8.2、GPUは無いが計算ノード(A)と同じバイナリが動作
  - アクセス方法
    - 従来のesへのアクセス方法と同じ方法で、es -> es-aに変更
    - 従来のes (es1, es2)にログイン後、ssh es-a

計算ノード(A)利用時の注意点 (2/2)

- 計算ノード(A)を使用するプログラムではCUDA 11.xを利用
  - CUDA 10.x以下はNVIDIA A100が対応するCompute Capability 8.0に非対応
- 計算ノード(A)ではSHARP v2の利用が可能
  - 集団通信における演算の一部をInfiniBandスイッチにオフロードし、性能向上する技術
  - 現在はルーティング設定に問題があり、十分に性能発揮できない
    - 6月のメンテナンスにて解決予定
  - 現時点では運用でのサポート対象外

3.ミニキャンプでの基本操作・お役立ち情報

バッチジョブ(qsub)

1） インタラクティブノード(es)にログインして、sample.shを実行。

```
[aaa12345xx@es1 ~]$ qsub -g gaa50123 -l rt_AG.small=1 sample.sh
```

 # gaa12345: グループ名, rt_AG.small=1 : 計算資源タイプ（NVIDIA A100 x 1）, *sample.sh:* ジョブスクリプト

```
Your job 151645 ("sample.sh") has been submitted
```

2） バッチジョブの状況を参照。

```
[aaa12345xx@es1 ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          jclass    slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
151646  0.25586 sample.sh  aaa12345xx   r     01/20/2019 15:16:53 gpu@g0002                                  10
```

3） バッチジョブの出力（ホームディレクトリ）。

```
[aaa12345xx@es1 ~]$ ls -l
-rw-r--r-- 1 aaa12345xx gaa50123       172  1月 20 15:17 sample.sh.e151646  #エラー出力ファイル（数字はジョブ番号）
-rw-r--r-- 1 aaa12345xx gaa50123         0  1月 20 13:51 sample.sh.o151235  #正常出力ファイル（数字はジョブ番号）
```

予約ノードの指定法： サブコマンド（ -ar *ar_id* ）

 # *ar_id* は「予約ID」（４桁の数字、ミニキャンプでは「 3686 」）

予約ID は、「利用ポータル」の「ノード予約・キャンセル」または、qrstatコマンドで参照できる。

優先度を上げるには： サブコマンド(-p -400)

ポイントは通常の1.5倍

ジョブをプログラム（スクリプト）化 -> 多数のジョブを一度に投げられる

インタラクティブジョブ(qrsh)

1） インタラクティブノード(es)にログインして実行。

```
[aaa12345xx@es1 ~]$ qrsh -g gaa50123 -l rt_AF=1 -l h_rt=01:00:00
```

 # gaa12345: グループ名, rt_AF=1 : 計算資源タイプ（計算ノード(A) を1個）, *h_rt=01:00:00*（最大1時間確保）

2） インタラクティブジョブの状況を参照。

```
[aaa12345xx@es1 ~]$ qstat
job-ID  prior   name       user         state  submit/start at     queue                  jclass    slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
151646  0.28027 QRLOGIN    aaa12345xx   r      01/21/2019 09:39:43  gpu@g0371                        10
```

予約ノードの指定法：　サブコマンド（ -ar *ar_id* ）

 # *ar_id* は「予約ID」（４桁の数字、ミニキャンプでは「 3686 」）

予約ID は、「利用ポータル」の「ノード予約・キャンセル」または、qrstatコマンドで参照できる。

ジョブ実行時に、インタラクティブに計算ノードへアクセス可能 -> デバッグ等に便利

NVIDIA A100を利用するPyTorch導入例

導入例: PyTorch 1.8.1 + CUDA 11.1 + Python 3.8.7

```
[aaa12345xx@es-a1 ~]$ module load python/3.8/3.8.7
[aaa12345xx@es-a1 ~]$ python3 -m venv ~/lib/pyenv/pytorch-cna
[aaa12345xx@es-a1 ~]$ source ~/lib/pyenv/pytorch-cna/bin/activate
[aaa12345xx@es-a1 ~]$ pip install --upgrade pip setuptools
[aaa12345xx@es-a1 ~]$ module load cuda/11.1/11.1.1
[aaa12345xx@es-a1 ~]$ pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111 torchaudio==0.8.1 -f
https://download.pytorch.org/whl/torch_stable.html
```

参考: https://pytorch.org/get-started/locally/

ジョブスクリプト例: sample.sh

```
#!/bin/sh
#$-l rt_AG.small=1
#$-cwd
#$-l h_rt=01:00:00

source /etc/profile.d/modules.sh
module load python/3.8/3.8.7
module load cuda/11.1/11.1.1
source ~/lib/pyenv/pytorch-cna/bin/activate

wget -O mnist-main.py https://raw.githubusercontent.com/pytorch/examples/master/mnist/main.py
python mnist-main.py
```

ジョブ投入例

```
[aaa12345xx@es-a1 ~]$ qsub -g gaa50123 sample.sh
```

NVIDIA A100を利用するTensorFlow導入例

導入例: TensorFlow 2.4.0 + CUDA 11.0 + Python 3.8.7

```
[aaa12345xx@es-a1 ~]$ module load python/3.8/3.8.7
[aaa12345xx@es-a1 ~]$ python3 -m venv ~/lib/pyenv/tensorflow-cna
[aaa12345xx@es-a1 ~]$ source ~/lib/pyenv/tensorflow-cna/bin/activate
[aaa12345xx@es-a1 ~]$ pip install --upgrade pip setuptools
[aaa12345xx@es-a1 ~]$ module load cuda/11.0/11.0.3
[aaa12345xx@es-a1 ~]$ pip install tensorflow==2.4.0
```

ジョブスクリプト例: sample.sh

参考: https://www.tensorflow.org/install/source?hl=ja#gpu

```
#!/bin/sh
#$-l rt_AG.small=1
#$-cwd
#$-l h_rt=01:00:00

source /etc/profile.d/modules.sh
module load python/3.8/3.8.7
module load cuda/11.0/11.0.3
source ~/lib/pyenv/tensorflow-cna/bin/activate

# pip install tensorflow_datasets tensorflow-model-optimization
# git clone git@github.com:tensorflow/models.git
cd models
PYTHONPATH=`pwd` python official/vision/image_classification/mnist_main.py ¥
--distribution_strategy=one_device --num_gpus=1
```

ジョブ投入例

```
[aaa12345xx@es-a1 ~]$ qsub -g gaa50123 sample.sh
```

NCCLでSHARPを利用 (1/4)

- NCCL: NVIDIA社が開発するGPU間集団通信ライブラリ
  - バージョン2.6.2よりSHARPをサポート
- ABCIでは、ABCIが提供する各CUDAバージョンに対応するNCCLをサポート
  - CUDA 11.1.1に対応するNCCL 2.8.4-1を利用する場合

```
[aaa12345xx@a0001 ~]$ module load cuda/11.1/11.1.1
[aaa12345xx@a0001 ~]$ module load nccl/2.8/2.8.4-1
```

    - MPIとNCCLをloadする前にCUDAがloadされていると、そのCUDAに対応するMPIとNCCLをloadする
- NCCLでSHARPを利用するには、利用者自身でNCCLのプラグインを導入する必要あり
  - nccl-rdma-sharp-plugins

NCCLでSHARPを利用 (2/4)

nccl-rdma-sharp-plugins インストール例

```
[aaa12345xx@es-a1 ~]$ qrsh -g gaa50123 -l rt_AG.small=1 -l h_rt=01:00:00
[aaa12345xx@a0001 ~]$ module load cuda/11.1/11.1.1
[aaa12345xx@a0001 ~]$ module load openmpi/4.0.5
[aaa12345xx@a0001 ~]$ ompi_info | grep 'Configure command line'
Configure command line: '--prefix=/apps/rhel8/openmpi/4.0.5/gcc8.3.1_cuda11.1.1' '--enable-orterun-
prefix-by-default' '--with-cuda=/apps/cuda/11.1.1' '--with-
ucx=/apps/rhel8/ucx/1.9.0/gcc8.3.1_cuda11.1.1_gdrcopy2.1' '--with-sge'

[aaa12345xx@a0001 ~]$ git clone git@github.com:Mellanox/nccl-rdma-sharp-plugins.git
[aaa12345xx@a0001 ~]$ cd nccl-rdma-sharp-plugins
[aaa12345xx@a0001 ~]$ ./autogen.sh
[aaa12345xx@a0001 nccl-rdma-sharp-plugins]$ ./configure --prefix=<LIBNCCL_NET_PATH> ¥
--with-cuda=$CUDA_HOME --with-sharp=/opt/mellanox/sharp ¥
--with-ucx=/apps/rhel8/ucx/1.9.0/gcc8.3.1_cuda11.1.1_gdrcopy2.1
[aaa12345xx@a0001 nccl-rdma-sharp-plugins]$ make
[aaa12345xx@a0001 nccl-rdma-sharp-plugins]$ make install
[aaa12345xx@a0001 nccl-rdma-sharp-plugins]$ exit
```

UCXのパスを確認

nccl-rdma-sharp-pluginsはUCX 1.9以上に対応。ABCIが提供するCUDAに依存しないOpenMPIは全てUCX 1.8を利用するようビルドされているため、CUDA-awareなOpenMPIを利用し、それが依存するUCX 1.9を利用してビルドする必要がある。

NCCLでSHARPを利用 (3/4)

ジョブスクリプト例:

NCCL Tests

```
#!/bin/sh
#$-l rt_AF=4
#$-cwd
#$-l h_rt=00:20:00

source /etc/profile.d/modules.sh
module load cuda/11.1/11.1.1
module load openmpi/4.0.5
module load nccl/2.8/2.8.4-1

PROG=${NCCL_TESTS_HOME}/build/all_reduce_perf
PROG_ARGS="-b 4 -e 64M -f 2 -g 1 -t 1"
NPPN=8
NMPIPROCS=$(($NHOSTS * $NPPN))
NPPS=$(($NPPN / 2))

mpirun -np $NMPIPROCS --map-by ppr:$NPPS:socket ¥
       -x LD_LIBRARY_PATH=${LIBNCCL_NET_PATH}:${LD_LIBRARY_PATH} ¥
       -x NCCL_DEBUG=INFO ¥
       -x NCCL_IB_HCA=mlx5 ¥
       -x NCCL_COLLNET_ENABLED=1 ¥
       -x NCCL_ALGO=CollNet ¥
       -x SHARP_COLL_LOG_LEVEL=3 ¥
       -x ENABLE_SHARP_COLL=1 ¥
       -x SHARP_COLL_ENABLE_SAT=1 ¥
       ${PROG} ${PROG_ARGS}
```

NCCLでSHARPを利用 (4/4)

参考情報

- Using Mellanox SHARP with NVIDIA NCCL
- NCCL Testsのビルド例

```
[aaa12345xx@es-a1 ~]$ qrsh -g gaa50123 -l rt_AG.small=1 -l h_rt=01:00:00
[aaa12345xx@a0001 ~]$ module load cuda/11.1/11.1.1
[aaa12345xx@a0001 ~]$ module load openmpi/4.0.5
[aaa12345xx@a0001 ~]$ module load nccl/2.8/2.8.4-1
[aaa12345xx@a0001 ~]$ git clone git@github.com:NVIDIA/nccl-tests.git
[aaa12345xx@a0001 ~]$ cd nccl-tests
[aaa12345xx@a0001 nccl-tests]$ make MPI=1 MPI_HOME=$OMPI_HOME CUDA_HOME=$CUDA_HOME ¥
NCCL_HOME=$NCCL_HOME
     ./build以下にバイナリが生成される
```

OpenMPIでSHARPを利用 (1/3)

- SHARPを使えるようにビルドされたOpenMPIを /apps/rhel8/openmpi-hcoll に準備中
  - ただし、MPI_Reduceに問題があることを確認
  - 将来的に削除・移動される可能性あり
- 参考: Using Mellanox SHARP with Open MPI

OpenMPIでSHARPを利用 (2/3)

プログラムビルド例: OSU Micro-Benchmarks

```
[aaa12345xx@es-a1 ~]$ qrsh -g gaa50123 -l rt_AG.small=1 -l h_rt=01:00:00
[aaa12345xx@a0001 ~]$ module load cuda/11.1/11.1.1
[aaa12345xx@a0001 ~]$ export OMPI_HOME=/apps/rhel8/openmpi-hcoll/4.0.5/gcc8.3.1_cuda11.1.1
[aaa12345xx@a0001 ~]$ export PATH=${OMPI_HOME}/bin:$PATH
[aaa12345xx@a0001 ~]$ export LD_LIBRARY_PATH=${OMPI_HOME}/lib:$LD_LIBRARY_PATH
[aaa12345xx@a0001 ~]$ export CPATH=${OMPI_HOME}/include:$CPATH
[aaa12345xx@a0001 ~]$ export LIBRARY_PATH=${OMPI_HOME}/lib:$LIBRARY_PATH
[aaa12345xx@a0001 ~]$ export MANPATH=${OMPI_HOME}/share/man:$MANPATH
[aaa12345xx@a0001 ~]$ wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-
5.7.tar.gz
[aaa12345xx@a0001 ~]$ tar zxf osu-micro-benchmarks-5.7.tar.gz
[aaa12345xx@a0001 ~]$ cd osu-micro-benchmarks-5.7
[aaa12345xx@a0001 ~]$ CC=mpicc ./configure --prefix=$OMB_DIR --enable-cuda
[aaa12345xx@a0001 ~]$ make; make install
```

各種環境変数を自分で設定する必要があるが、普段通りのMPIプログラムをビルドする手順と同じ

OpenMPIでSHARPを利用 (3/3)

ジョブスクリプト例:
OSU Micro-Benchmarks

```
#!/bin/sh
#$-l rt_AF=4
#$-cwd
#$-l h_rt=00:20:00

source /etc/profile.d/modules.sh
module load cuda/11.1/11.1.1

export OMPI_HOME=/apps/rhel8/openmpi-hcoll/4.0.5/gcc8.3.1_cuda11.1.1
export PATH=${OMPI_HOME}/bin:$PATH
export LD_LIBRARY_PATH=${OMPI_HOME}/lib:$LD_LIBRARY_PATH

GET_LOCAL_RANK=${OMB_DIR}/get_local_rank
PROG=${OMB_DIR}/mpi/collective/osu_allreduce

NPPN=8
NMPIPROCS=$(($NHOSTS * $NPPN))
NPPS=$(($NPPN / 2))

mpirun -np $NMPIPROCS --map-by ppr:$NPPS:socket ¥
       -x PATH ¥
       -x LD_LIBRARY_PATH ¥
       -x HCOLL_ENABLE_SHARP=3 ¥
       -x SHARP_COLL_LOG_LEVEL=3 ¥
       ${GET_LOCAL_RANK } ${PROG}
```

ABCIクラウドストレージとは

https://docs.abci.ai/ja/abci-cloudstorage/

- Amazon Simple Storage Service (Amazon S3) 互換インターフェースを備えたオブジェクトストレージサービス
- 特徴
  - ① Amazon S3 互換インタフェース
  - ② 計算ノード、ABCI外部の両方からアクセス可能 (ユーザがアクセス制御可能*)
    - ＊ 同一ABCIグループ内、ABCIクラウドストレージアカウント保有者全員、一般公開の３レベルでアクセス制御が可能
  - ③ 暗号化機能を提供 (SSE: Server-Side Encryption機能をサポート*)
    - ＊ 暗号化を有効にしたバケット作成が必要
- アカウントとアクセスキーが必要(「ABCI利用ポータル」から申請)

ABCIクラウドストレージの使い方

https://docs.abci.ai/ja/abci-cloudstorage/usage/
https://docs.abci.ai/portal/ja/03/#manage-cloudstorage

- AWS-CLIモジュールのロード

```
[username@es1 ~]$ module load aws-cli
```

- 認証情報などの設定

```
[username@es1 ~]$ aws configure
 AWS Access Key ID [None]: ACCESS-KEY-ID
 AWS Secret Access Key [None]: SECRET-ACCESS-KEY
 Default region name [None]: us-east-1
 Default output format [None]:(入力不要)
```

- 各　種　操　作

AWS-CLI構文 aws [options] <command> <subcommand> [parameters]

例　　aws --endpoint-url https://s3.abci.ai s3 ls
s3: コマンド、ls: サブコマンド

バケットの作成：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 mb s3://dataset-summer-2012
make_bucket: dataset-summer-2012
```

バケットの一覧：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 ls
2019-06-15 10:47:37 testbucket1
2019-06-15 18:10:37 testbucket2
```

オブジェクトのリスト：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 ls s3://testbucket1
2019-07-05 17:33:05 4 test1.txt
2019-07-05 21:12:47 4 test2.txt
```

データのコピー：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 cp ./images/0001.jpg s3://dataset-c0541/
upload: images/0001.jpg to s3://dataset-c0541/0001.jpg
```

データの移動：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 mv annotations.zip s3://dataset-c0541/
move: ./annotations.zip to s3://dataset-c0541/annotations.zip
```

オブジェクトのリスト：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 ls s3://testbucket1
2019-07-05 17:33:05 4 test1.txt
2019-07-05 21:12:47 4 test2.txt
```

ローカルディレクトリとクラウドストレージの同期：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 sync ./sensor2 s3://mybucket/
upload: sensor2/0002.dat to s3://mybucket/0002.dat
upload: sensor2/0004.dat to s3://mybucket/0004.dat
```

オブジェクトの削除：

```
[username@es1 ~]$ aws --endpoint-url https://s3.abci.ai s3 rm s3://mybucket/readme.txt
delete: s3://mybucket/readme.txt
```

ABCI利用中にSSHセッションを切れないようにする

https://docs.abci.ai/ja/faq/#q-ssh

KeepAliveを適用するには、利用者の端末でシステムのssh設定ファイル(/etc/ssh/ssh_config)、またはユーザ毎の設定ファイル(~/.ssh/config)に、オプション ServerAliveInterval を60秒程度で設定してください。

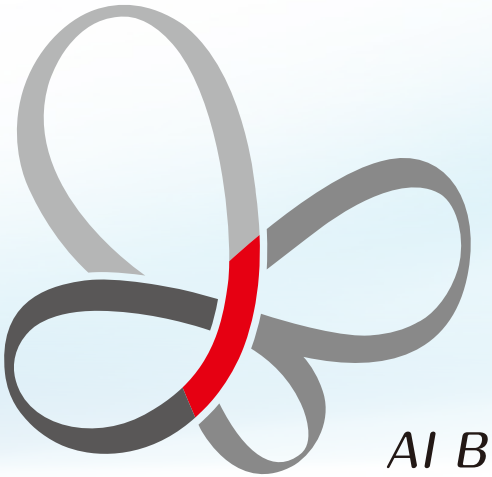
```
[username@userpc ~]$ vi ~/.ssh/config
[username@userpc ~]$ cat ~/.ssh/config
(snip)
Host as.abci.ai ServerAliveInterval 60
(snip)
[username@userpc ~]$
```

ABCI参考サイト

より詳細な情報については、以下を参照下さい。

- ABCIユーザガイド
  https://docs.abci.ai/ja/
- ABCI利用ポータル
  https://portal.abci.ai/user/
- ABCI利用ポータルの利用ガイド
  https://docs.abci.ai/portal/ja/
- ABCIユーザサポート
  https://abci.ai/ja/how_to_use/user_support.html
- ABCI利用法に関するFAQ
  https://abci.ai/ja/how_to_use/yakkan.html#faq9
- クイックレファレンス
  https://abci.ai/ja/how_to_use/data/ABCI_referrence.html

AIST



ABCI

AI Bridging Cloud Infrastructure

https://abci.ai/