NVIDIA DEEP LEARNING INSTITUTE

第100回お試しアカウント付き並列プログラミング講習会
「REEDBUSH スパコンを用いたGPUディープラーニング入門」

ハンズオン#2:
マルチGPUによる高速化体験

山崎和博
NVIDIA, ディープラーニング ソリューションアーキテクト

AGENDA

ハンズオン#2のテーマ: 分散学習

タスク#1: マルチGPUでの学習

タスク#2: マルチノードでの学習

ハンズオン#2のゴール

複数のノードを利用できるようになる

- Reedbushの複数ノードを利用してジョブを流す方法を把握する
- マルチGPU/マルチノードでの学習方法を理解する

NVIDIA DEEP LEARNING INSTITUTE

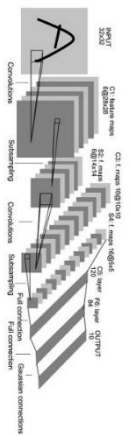
ハンズオン#2のテーマ: 分散学習

分散学習

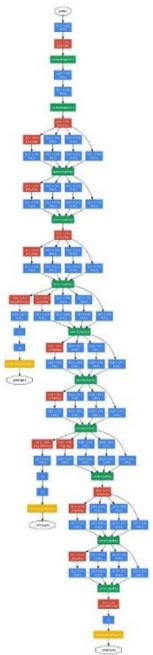
モデル/データの大規模化

モデルの複雑化

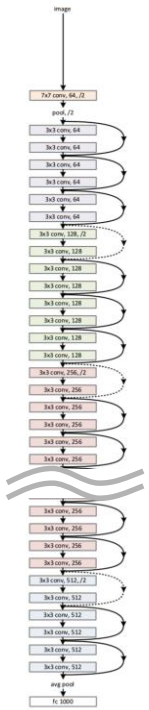
LeNet
5layers



GoogLeNet
24layers



ResNet
152layers



?

データの拡大

?

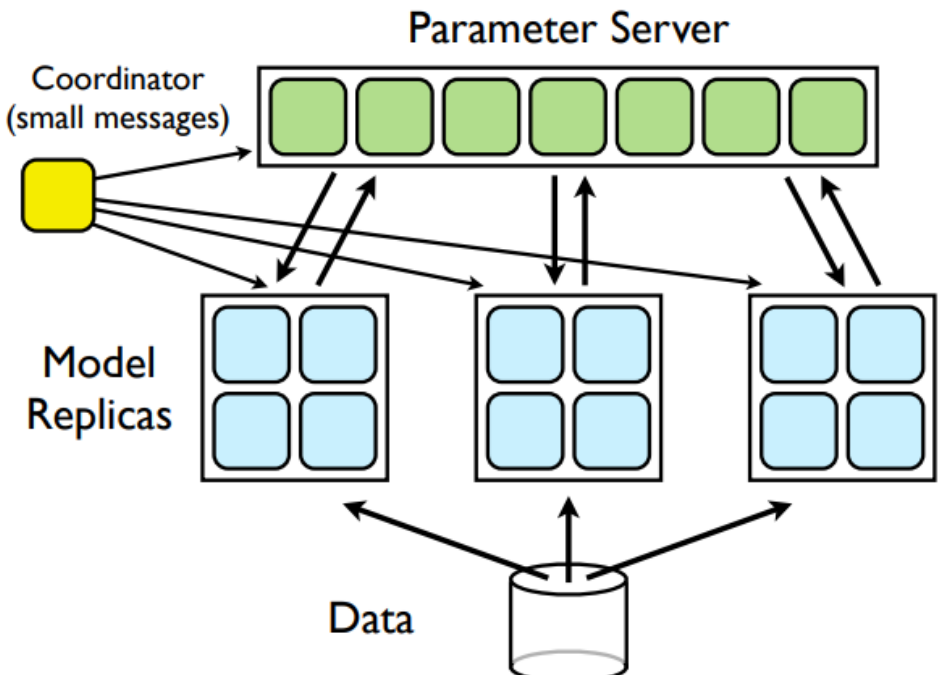
ImageNet
256x256, 1M imgs

MNIST
28x28,60K imgs

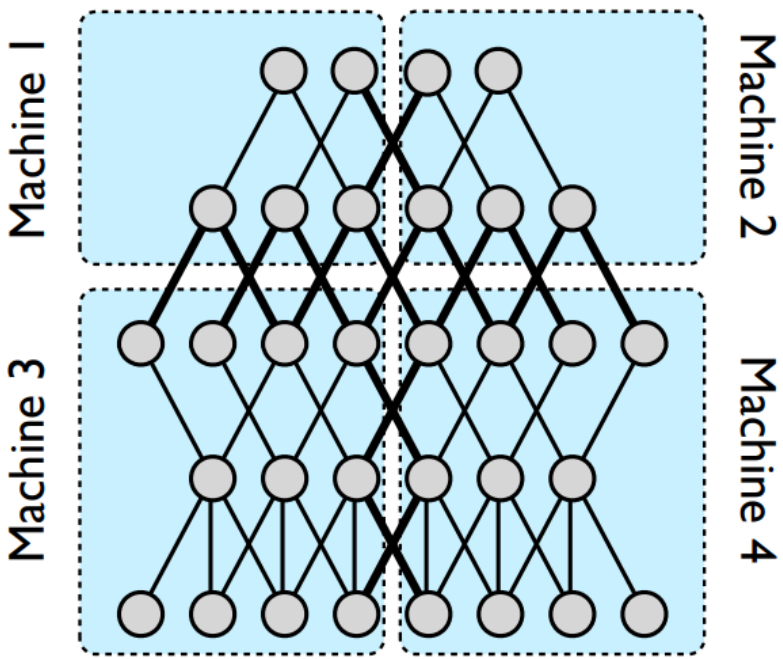
NVIDIA DEEP LEARNING INSTITUTE

分散学習

マルチGPU/マルチノードで分散して学習



データ並列（data parallelism）



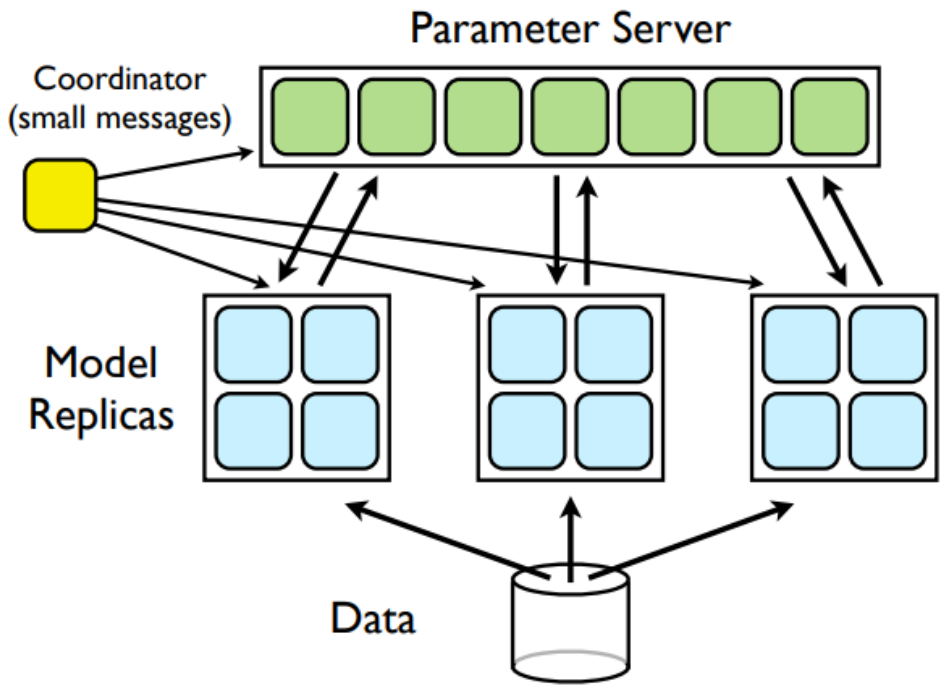
モデル並列（model parallelism）

Large Scale Distributed Deep Networks より

NVIDIA DEEP LEARNING INSTITUTE

分散学習

マルチGPU/マルチノードで分散して学習



データ並列（data parallelism）

- 複数GPUで同じネットワーク（パラメータ）を共有
- データを分割してそれぞれ勾配計算し、パラメータを更新

→ **学習時間の短縮が期待できる**

Large Scale Distributed Deep Networks より

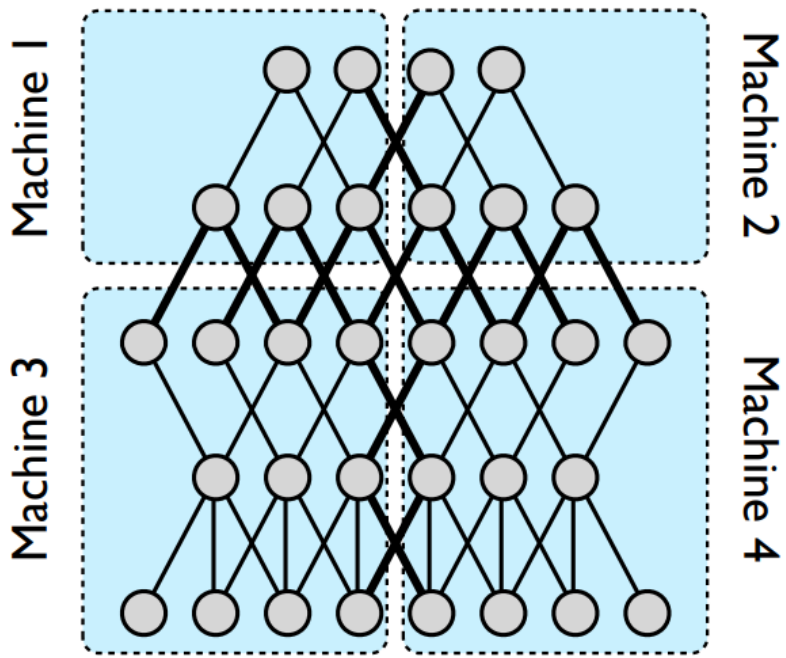
NVIDIA DEEP LEARNING INSTITUTE

分散学習

マルチGPU/マルチノードで分散して学習

- ネットワーク（パラメータ）を複数GPUで分散して保持
- 同じデータに対し、各GPUが協調してパラメータを更新

→ **1GPUに乗り切らない大規模なネットワークを処理できる可能性**



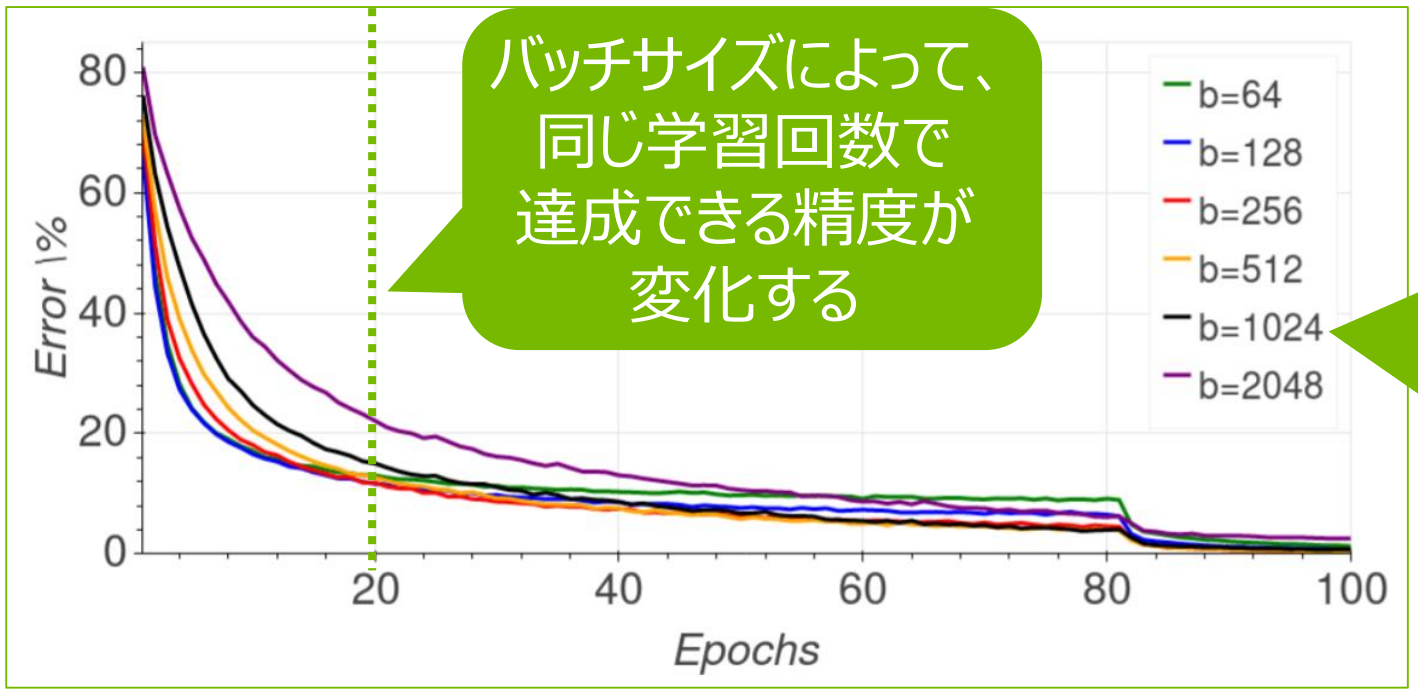
モデル並列（model parallelism）

Large Scale Distributed Deep Networks より

NVIDIA DEEP LEARNING INSTITUTE

分散学習における注意点

バッチサイズと精度の関係



バッチサイズによって、
同じ学習回数で
達成できる精度が
変化する

分散学習すると、
トータルのバッチサイズ
は大きくなりがち
→ 問題になる
可能性あり

https://arxiv.org/abs/1705.08741 より

NVIDIA DEEP LEARNING INSTITUTE

分散学習における注意点

バッチサイズと精度の関係

学習率やバッチサイズを調整することで、精度低下を回避する研究が多数

- Train longer, generalize better: closing the generalization gap in large batch training of neural networks
- Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour
- LARGE BATCH TRAINING OF CONVOLUTIONAL NETWORKS
- Don't Decay the Learning Rate, Increase the Batch Size

……が、今日は細かいところに踏み込みません

https://arxiv.org/abs/1705.08741 より

NVIDIA DEEP LEARNING INSTITUTE

タスク#1: マルチGPUでの学習

マルチGPUで学習ジョブを流す

簡単な例で、ジョブの流し方を把握する

以下の作業を完了させる。

1. train_cifar_multi_gpu.pyに「PUT YOUR CODE」という箇所があるので、書き換えてプログラムを完成させる
2. スクリプトの修正完了したら実行
    1. qsub -j oe run_multi_gpu_training.sh
3. 結果を確認

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所

```
91    # Set up a dataset and prepare train/test
92    print('Using CIFAR10 dataset: {}'.format(
93    train, test = dataset_cifar10.load_cifar
94    train_iter = """PUT YOUR CODE"""
95    test_iter = chainer.iterators.SerialIterator(test, args.batchsize,
96                                                 repeat=False, shuffle=False)
97
98    # Make a model.
99    class_labels = 10
100   model = L.Classifier(models.VGG.VGG(class_labels))
101
102   # Make a specified GPU current
103   master_gpu_id = """PUT YOUR CODE"""
104   chainer.cuda.get_device_from_id("""PUT YOUR CODE""").use()
105
106   # Make optimizer.
107   optimizer = chainer.optimizers.MomentumSGD(args.learnrate)
108   optimizer.setup(model)
109   optimizer.add_hook(chainer.optimizer.WeightDecay(5e-4))
110
111   # Set up a trainer
112   devices_list = {'main': """PUT YOUR CODE"""}
113   devices_list.update({"""PUT YOUR CODE""" for i in range(1, args.n_gpus)})
114   print(devices_list)
115
116   updater = training."""PUT YOUR CODE"""
117   trainer = training.Trainer(updater, (args.epoch, 'epoch'), out=args.out)
118   evaluator = extensions.Evaluator(test_iter, model, device="""PUT YOUR CODE""")
```

マルチGPU向けに
データ用iteratorを変更

メインのGPUとして
使うものを指定

デバイス(=GPU)の
リストを作成

マルチGPU用の
更新ロジックを指定

テストで使う
GPUを指定

NVIDIA DEEP LEARNING INSTITUTE

CHAINERのAPI

学習の実行

- ネットワークの更新ロジックを指定
  - chainer.training.updaters.MultiprocessParallelUpdater(iterators, optimizer, devices)
  - 基本はシングルGPUのクラスと同じ使い方
    - GPUへのモデル転送は明示的に実施する必要なし
  - 学習データのイテレータをGPU数と同じ数準備する
    - デバイス名をキーとする辞書(dict)を作成
    - メインのGPUのみデバイス名を「main」にする必要あり

NVIDIA DEEP LEARNING INSTITUTE

メインのGPU？

ChainerのMultiprocessParallelUpdaterにおけるData Parallelism

シングルGPUの場合

Forward | Backward | Up

計算した勾配を
全GPUで共有して
モデル更新

マルチGPUの場合

Forward | Backward

Forward | Backward

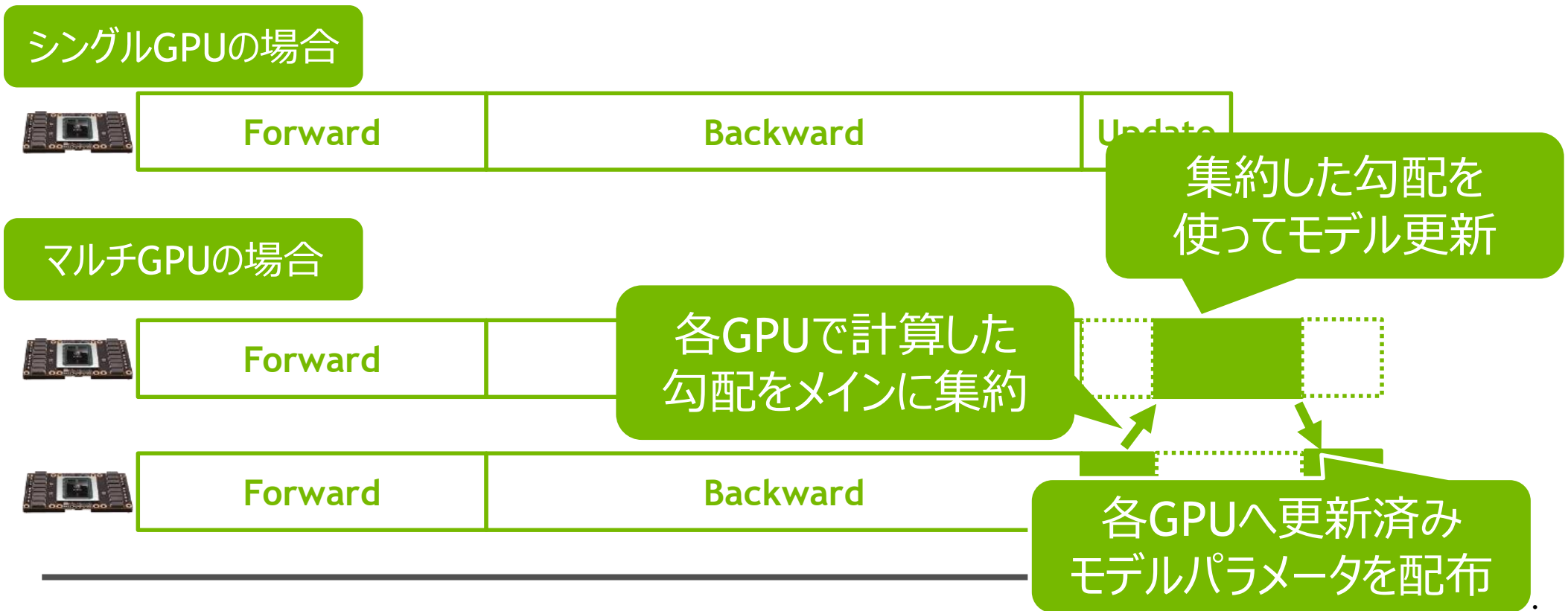
Update

1iteration

NVIDIA DEEP LEARNING INSTITUTE

メインのGPU？

ChainerのMultiprocessParallelUpdaterにおけるData Parallelism



※ ChainerMNとは、異なる計算モデルを使っているようです

CHAINERのAPI

データの分割

- chainer.datasets.split_dataset_n_random(dataset, n)
  - 学習データのイテレータ作成時に使用
  - データをランダムにGPU数分割

NVIDIA DEEP LEARNING INSTITUTE

データ分割？

各GPUへの分配

特に何もしないと、
1epochで処理される
データ量がGPU数倍に

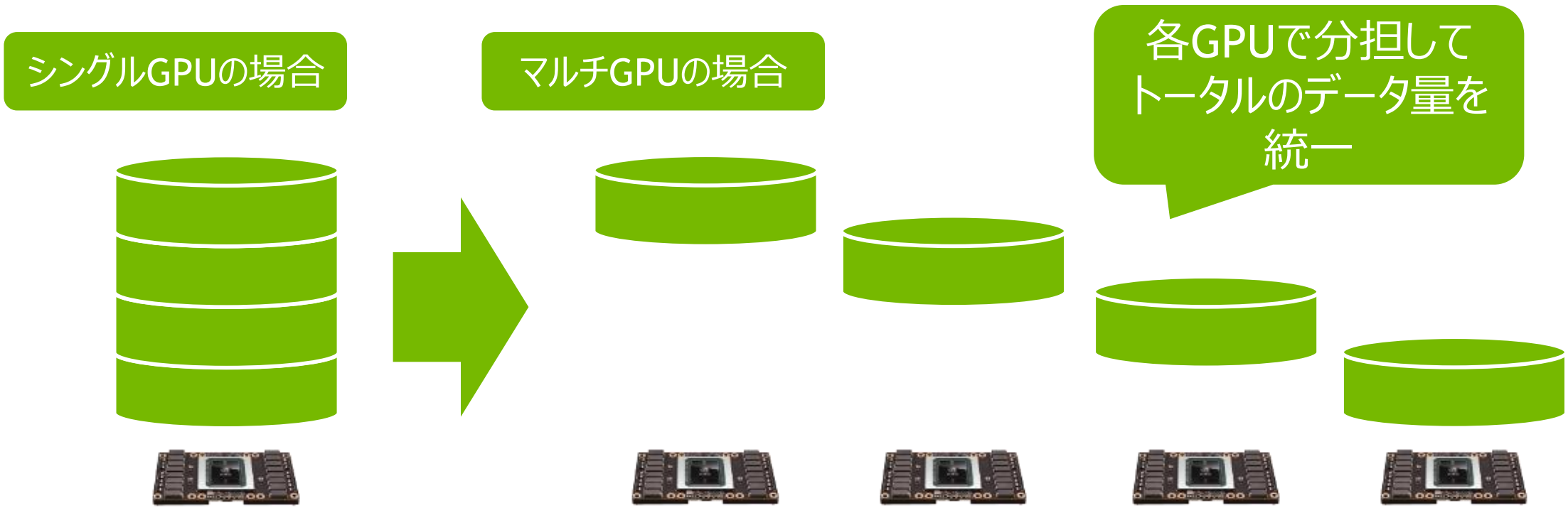
シングルGPUの場合

マルチGPUの場合

NVIDIA DEEP LEARNING INSTITUTE

データ分割？

各GPUへの分配



NVIDIA DEEP LEARNING INSTITUTE

スクリプトの修正を試みましょう

コンソールに戻ってください！

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所(答え; 1/2)

```
train_iter=[chainer.iterators.SerialIterator(sub_train, args.batchsize) \
            for sub_train \
            in chainer.datasets.split_dataset_n_random(train, args.n_gpus)]
```

```
master_gpu_id=0
.get_device_from_id(master_gpu_id).
```

```
91    # Set up a dataset and prepare train/test data iterator.
92    print('Using CIFAR10 dataset: {}'.format(args.dataset))
93    train, test = dataset_cifar10.load_cifar10(args.dataset)
94    train_iter = """PUT YOUR CODE"""
101
102   # Make a specified GPU current
103   master_gpu_id = """PUT YOUR CODE"""
104   chainer.cuda.get_device_from_id("""PUT YOUR CODE""").use()
105
106   # M
                                    umSGD(args.learnrate)
                                    WeightDecay(5e-4))
                                    ODE"""}
113   devices_list.update({"""PUT YOUR CODE""" for i in range(1, args.n_gpus)})
114   print(devices_list)
115
116   updater = training."""PUT YOUR CODE"""
117   trainer = training.Trainer(updater, (args.epoch, 'epoch'), out=args.out)
118   evaluator = extensions.Evaluator(test_iter, model, device="""PUT YOUR CODE""")
```


NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所(答え; 2/2)

```
devices_list={'main': master_gpu_id}
devices_list.update({'gpu{}'.format(i): i for i in range(1, args.n_gpus)})
```

```
111     # Set up a trainer
112     devices_list = {'main': """PUT YOUR CODE"""}
113     devices_list.update({"""PUT YOUR CODE""" for i in range(1, args.n_gpus)})
114     print(devices_list)
115
116     updater = training."""PUT YOUR CODE"""
117     trainer = training.Trainer(updater, (args.epoch, 'epoch'), out=args.out)
118     evaluator = extensions.Evaluator(test_iter, model, device="""PUT YOUR CODE""")
```

```
device=0
```

```
updater=training.updaters.MultiprocessParallelUpdater(
train_iter, optimizer, devices=devices_list)
```


NVIDIA DEEP LEARNING INSTITUTE

マルチGPUでの学習実行結果

```
Start a training script using single GPU.
# Minibatch-size: 32
# epoch: 4

Using CIFAR10 dataset: /lustre/gt00/share/lecture/20180531_dl_intro/dataset//cifar10_256px_5p.pickle.gz
epoch       main/loss   validation/main/loss  main/accuracy  validati
1           3.56997     9.0989                0.107595       0.141016
(中略)
4           2.37658     2.14666               0.199119       0.227344                  193.16
Throughput: 51.76725472300107 [images/sec.] (10000 / 193.17230657697655)
Start a training script using multiple GPUs.
# GPUs: 2
# Minibatch-size: 32
# epoch: 4

Using CIFAR10 dataset: /lustre/gt00/share/lecture/20180531_dl_intro/dataset//cifar10_256px_5p.pickle.gz
{'main': 0, 'gpu1': 1}
/lustre/app/acc/cuda9/chainermn/ofed4.2/1.2.0/lib/python3.6/site-
packages/chainer/training/updaters/multiprocess_parallel_updater.py:141: UserWarning: optimizer.lr is changed to 0.025 by
MultiprocessParallelUpdater for new batch size.
  format(optimizer.lr))
epoch       main/loss   validation/main/loss  main/accuracy  validati
1           3.03633     2.34811               0.107813       0.106641
(中略)
4           2.83194     2.33467               0.09375        0.100781                  114.008
Throughput: 87.71081415935855 [images/sec.] (10000 / 114.011026985012)
```

シングルGPUだと
51.8image/sec.

同条件で2GPU使うと
87.7image/sec.と高速に

マルチGPUでの学習実行結果

```
Start a training script using single GPU.
# Minibatch-size: 32
# epoch: 4

Using CIFAR10 dataset: /lustre/gt00/share/lecture/20180531_dl_intro/dataset//cifar10-256px.5n.pickle.gz
epoch       main/loss   validation/main/loss  main/accuracy  validati
1           3.56997     9.0989                0.107595       0.14101
(中略)
4           2.37658     2.14666               0.199119       0.227344                193.16
Throughput: 51.76725472300107 [images/sec.] (10000 / 193.17230657697655)
Start a training script usi
# GPUs: 2
# Minibatch-size: 32
# epoch: 4

Using CIFAR10 dataset: /lu
{'main': 0, 'gpu1': 1}
/lustre/app/acc/cuda9/chainermn/ofed4.2/1.2.0/lib/python3.6/site-
packages/chainer/training/updaters/multiprocess_parallel_updater.py:141: UserWarning: optimizer.lr is changed to 0.025 by
MultiprocessParallelUpdater for new batch size.
  format(optimizer.lr))
epoch       main/loss   validation/main/loss  main/accuracy  validati
1           3.03633     2.34811               0.107813       0.106641
(中略)
4           2.83194     2.33467               0.09375        0.100781                114.008
Throughput: 87.71081415935855 [images/sec.] (10000 / 114.011026985012)
```

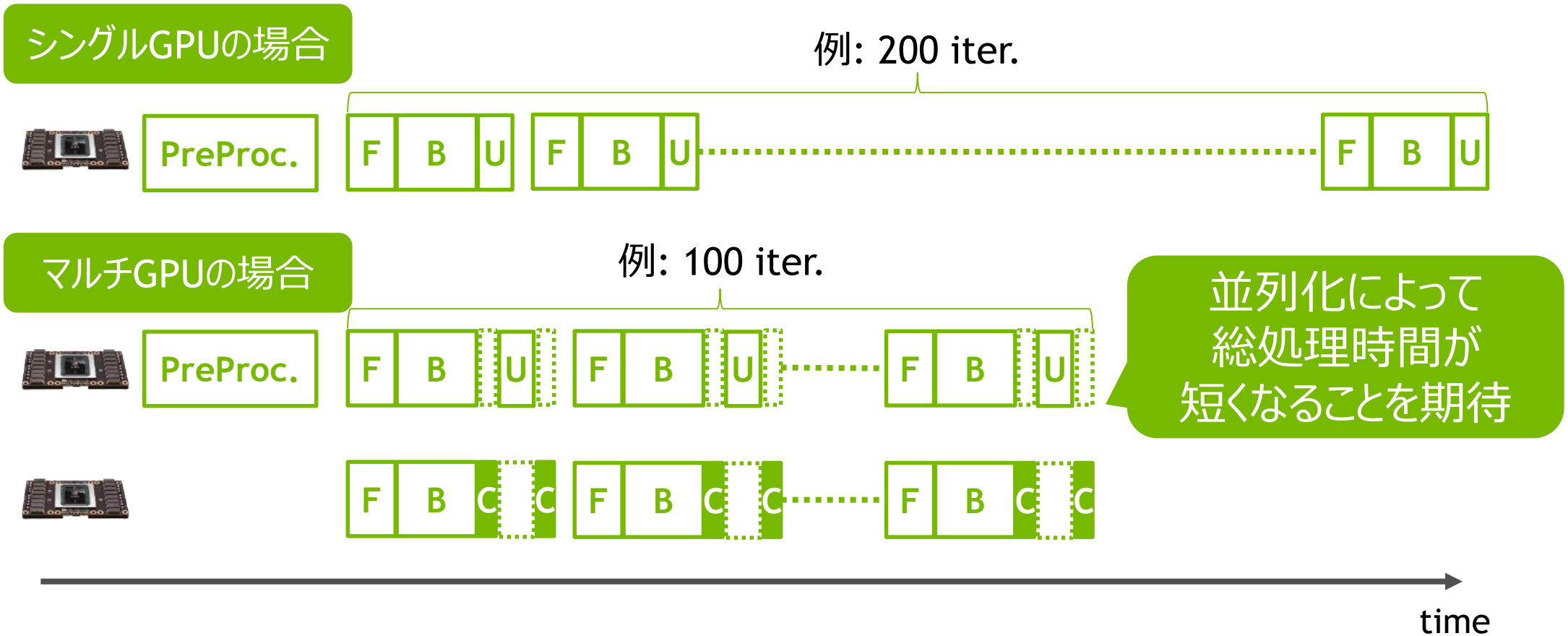
シングルGPUだと
51.8image/sec.

2GPUで約1.69倍？

同条件で2GPU使うと
87.7image/sec.と高速に

マルチGPUによる高速化の詳細

高速化のキモは？



マルチGPUによる高速化の詳細

高速化のキモは？

シングルGPUの場合

PreProc.

例: 20 iter.

F B U F B U ..... F B U

マルチGPUの場合

PreProc.

例: 10 iter.

F B U F B U .....

繰り返し回数が
少ない間は並列化の
効果が見えにくい

F B C C F B C C ..... F B C C

前処理のような
共通して時間を要する
箇所の比率が大きく

time

NVIDIA DEEP LEARNING INSTITUTE

マルチGPUでの学習実行結果

```
Start a training script using single GPU.
# Minibatch-size: 32
# epoch: 50

Using CIFAR10 dataset: /lustre/gt00/share/lecture/20180531_dl_intro/dataset//cifar10_256px_5p.pickle.gz
epoch       main/loss   validation/main/loss  main/accuracy  validati
1           3.69785     2.37116               0.111551       0.100781
(中略)
50          0.344069    1.94607               0.882612       0.417578                  2394.26
Throughput: 52.20790861389455 [images/sec.] (125000 / 2394.2732685279916)
Start a training script using multiple GPUs.
# GPUs: 2
# Minibatch-size: 32
# epoch: 50

Using CIFAR10 dataset: /lustre/gt00/share/lecture/20180531_dl_intro/dataset//cifar10_256px_5p.pickle.gz
{'main': 0, 'gpu1': 1}
/lustre/app/acc/cuda9/chainermn/ofed4.2/1.2.0/lib/python3.6/site-
packages/chainer/training/updaters/multiprocess_parallel_updater.py:141: UserWarning: optimizer.lr is changed to 0.025 by
MultiprocessParallelUpdater for new batch size.
  format(optimizer.lr))
epoch       main/loss   validation/main/loss  main/accuracy  validati
1           3.08727     4.65611               0.107031       0.091015
(中略)
50          0.338263    2.01606               0.888622       0.432031                  1297.95
Throughput: 96.30540731984837 [images/sec.] (125000 / 1297.9541178290383)
```

シングルGPUの場合は
50epochでも変化なし

50epoch回すと2GPUで
96.3images/sec.の約1.84倍

タスク#2: マルチノードでの学習

マルチノードで学習ジョブを流す

簡単な例で、ジョブの流し方を把握する

以下の作業を完了させる。

1. run_multi_node_training.shの実行環境指定を完成させる
2. train_cifar_multi_node.pyに「PUT YOUR CODE」という箇所があるので、書き換えてプログラムを完成させる
3. スクリプトの修正完了したら実行
    1. qsub -j oe run_multi_node_training.sh
4. 結果を確認

NVIDIA DEEP LEARNING INSTITUTE

ジョブスクリプト

マルチノード利用の条件指定

```
#!/bin/bash
#PBS -q (対象ジョブキュー名)
#PBS -l (利用ノード数など)
#PBS -W (利用グループ名)
#PBS -l (実行時間制限)
#PBS -N (ジョブ名)

...
```

利用ノードなどのパラメータは、以下をコロン「:」区切りで指定

- `select` ノード数
- `mpiprocs` mpiプロセス数/ノード
- `ompthreads` スレッド数/プロセス

ノードごとに2GPUで2ノードなので……

例) `-l select=2:mpiprocs=2`

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所(ジョブスクリプト)

```
1  #!/bin/sh
2  #PBS -q h-tutorial
3  #PBS -l xxxxxx
4  #PBS -W group_list=gt00
5  #PBS -l walltime=00:10:00
6  #PBS -N multi_node_training
7
```

ノード数などを指定

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所(PYTHON; 1/2)

```
68      # Setup multi-node communicator.
69      comm = chainermn.create_communicator(args.communicator)
70      device = """PUT YOUR CODE"""
71      assert device >= 0, 'invalid device i
72
73      if comm.mpi_comm.rank == 0:
74          print('=========================================')
75          print('Num process (COMM_WORLD): {}'.format(MPI.COMM_WORLD.Get_size()))
76          print('Using GPUs')
            int('Using {} communicator'.format(args.communicator))
            int('Num Minibatch-size: {}'.format(args.batchsize))
            int('Num epoch: {}'.format(args.epoch))
            int('=========================================')
81
82      # Split and distribute the dataset. Only worker 0 loads the whole dataset.
83      # Datasets of worker 0 are evenly split and distributed to all workers.
84      if """PUT YOUR CODE""":
85          print('Using CIFAR10 dataset: {}'.format(args.dataset))
            ain, test = dataset_cifar10.load_cifar10(args.dataset)
            asize = len(train) * args.epoch
89
            test = None, None
            chainermn."""PUT YOUR CODE"""(train, comm, shuffle=True, max_buf_len=1024 * 1024 * 1024)
91      test = chainermn."""PUT YOUR CODE"""(test, comm, shuffle=True, max_buf_len=1024 * 1024 * 1024)
```

MPIプロセス情報を参照して
どのGPUを割り当てるか指定

元データは親相当の
プロセスのみ
ロードするよう制御

各プロセスに
データを分配

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所(PYTHON; 2/2)

```
92
93      train_iter = chainer.iterators.SerialIterator(train, args.batchsize)
94      test_iter = chainer.iterators.SerialIterator(test, args.batchsize,
95                                                   repeat=False, shuffle=False)
96
97      class_labels = 10
98      model = L.Classifier(models.VGG.VGG(class_labels))
99      if device >= 0:
100         chainer.cuda.get_device(device).use()
101         model.to_gpu()
102
103     # Create a multi node optimizer from a standard Chainer optimizer.
104     optimizer = chainermn."""PUT YOUR CODE"""(
105         chainer.optimizers.MomentumSGD(0.05), comm)
106     optimizer.setup(model)
107     optimizer.add_hook(chainer.optimizer.WeightDecay(5e-4))
108
109     ter = training.StandardUpdater(train_iter, optimizer, device=device)
110     ner = training.Trainer(updater, (args.epoch, 'epoch'), out=args.out)
111
112     eate a multi node evaluator from a standard Chainer evaluator.
113     evaluator = extensions.Evaluator(test_iter, model, device=device)
114     evaluator = """PUT YOUR CODE"""
115
```

マルチノード実行向けに
最適化ロジックをラップ

マルチノード環境で適切に
評価が動くようラップ

NVIDIA DEEP LEARNING INSTITUTE

CHAINERMNのAPI

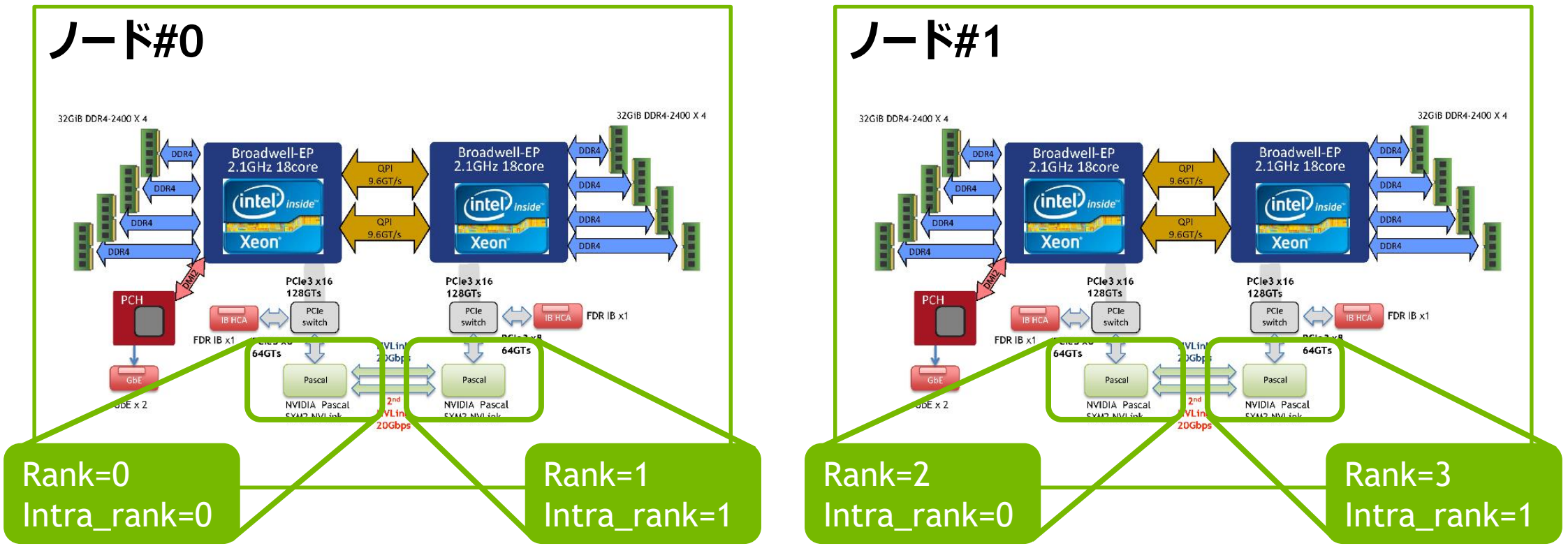
環境情報の取得

- `communicator.rank`
  - ワーカーID(プロセスID)を取得
- `communicator.intra_rank`
  - ノード内のプロセスIDを取得
- ノード内2プロセスで、2ノード使用の場合の例
  - ノード1: rank=0&intra_rank=0, rank=1&intra_rank=1
  - ノード2: rank=2&intra_rank=0, rank=3&intra_rank=1

NVIDIA DEEP LEARNING INSTITUTE

マルチノード実行時の状況

GPUとプロセス: ノード内2プロセス、2ノードの例

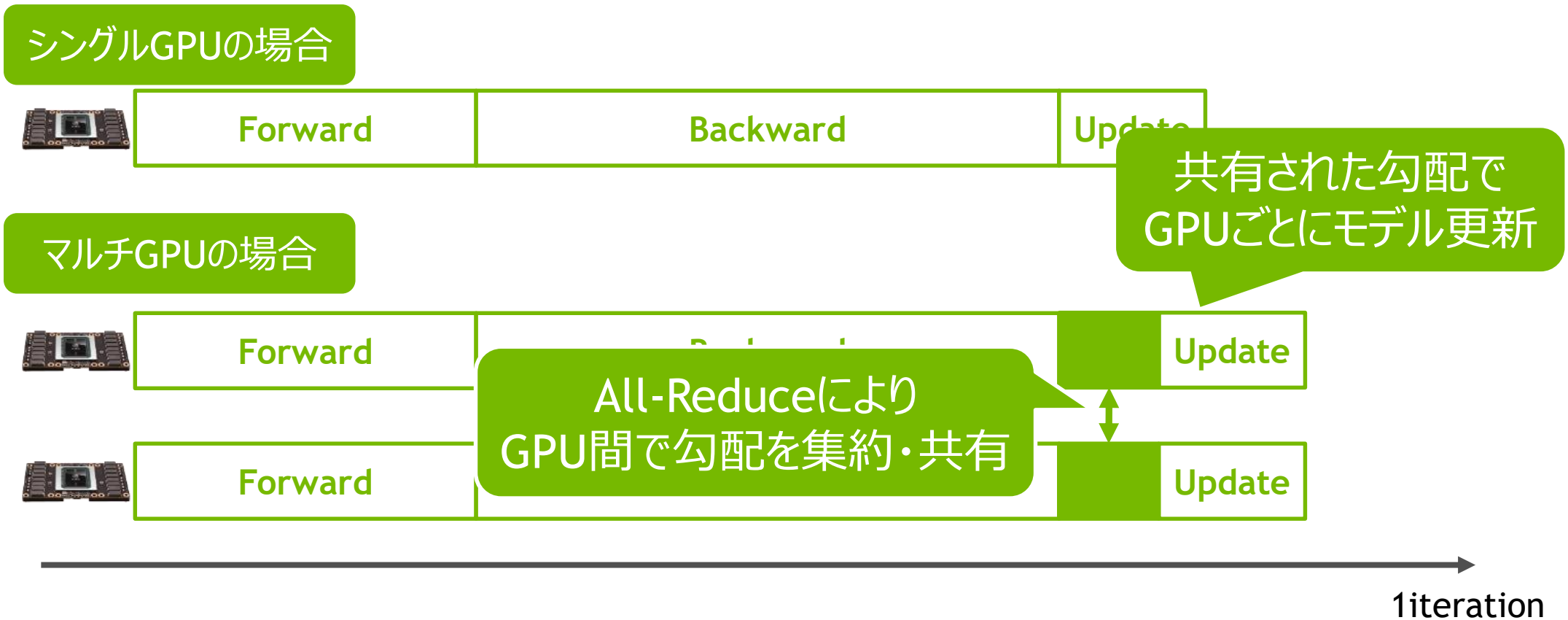


各プロセスのintra_rankの数値を見て、どのGPUを使うかなどを決める

NVIDIA DEEP LEARNING INSTITUTE

CHAINERMNでの分散ロジック

All-Reduceの利用



NVIDIA DEEP LEARNING INSTITUTE

CHAINERMNのAPI

分散学習用のAPI

- `chainermn.scatter_dataset(dataset, comm, shuffle)`
  - データセットを分割して、各プロセスに配布
- `chainermn.create_multi_node_optimizer(actual_optimizer, communicator)`
  - マルチノード環境で動作するよう、optimizerをラップ
- `chainermn.create_multi_node_evaluator(actual_evaluator, communicator)`
  - マルチノード環境で動作するよう、evaluatorをラップ

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの修正を試みましょう

コンソールに戻ってください！

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所(答え; ジョブスクリプト)

```
1  #!/bin/sh
2  #PBS -q h-tutorial
3  #PBS -l select=2:mpiprocs=2:ompthreads=1
4  #PBS -W group_list=gt00
5  #PBS -l walltime=00:10:00
6  #PBS -N multi_node_training
```

#PBS -l select=2:mpiprocs=2:ompthreads=1

```
27
28  mpirun \
29      --mca pml ^yalla --mca mtl ^mxm --mca coll ^hcoll --mca btl_openib_want_cuda_g
30      --mca mpi_warn_on_fork 0 \
31      ./get_local_rank_ompi.sh \
32      python train_cifar_multi_node.py --no_snapshot --epoch 4 --batchsize 32 --comm
33      > ${LOGDIR}/multi_node_log_$(date +%s).txt 2>&1
```

ChainerMNはmpirunコマンドを使って起動

スクリプトの変更箇所(答え; 1/2)

```
68      # Setup multi-node communicator.
69      comm = chainermn.create_communicator(args.communicator)
70      device = """PUT YOUR CODE"""
71      assert device >= 0, 'invalid de
72
73      if comm.mpi_comm.rank == 0:
74          print('=========================================')
75          print('Num process (COMM_WORLD): {}'.format(MPI.COMM_WORLD.Get_size()))
76          print('Using GPUs')
77          print('Using {} communicator'.format(args.communicator))
78          print('Num Minibatch-size: {}'.format(args.batchsize))
                n epoch: {}'.format(ar
                ==================
82      # t and distribute the datase
83      # Datasets of worker 0 are evenly
84      if """PUT YOUR CODE""":
85          print('Using CIFAR10 dataset:
86          train, test = dataset_cifar10
87          datasize = len(train) * args.e
88      else:
89          train, test = None, None
90      train = chainermn."""PUT YOUR CODE"""(train, comm, shuffle=True, max_buf_len=1024 * 1024 * 1024)
91      test = chainermn."""PUT YOUR CODE"""(test, comm, shuffle=True, max_buf_len=1024 * 1024 * 1024)
```

device = **comm.intra_rank**

if **comm.rank** == 0:

train=**chainermn.scatter_dataset**
(train, comm, shuffle=True, ...)
test=**chainermn.scatter_dataset**
(test, comm, shuffle=True, ...)

NVIDIA DEEP LEARNING INSTITUTE

スクリプトの変更箇所(答え; 2/2)

```
92
93      train_iter = chainer.iterators.SerialIterator(train, args.batchsize)
94      test_iter = chainer.iterators.SerialIterator(test, args.batchsize,
95                                                   repeat=False, shuffle=False)
96
97      class_labels = 10
```

optimizer=**chainermn.create_multi_node_optimizer(**
chainer.optimizers.MomentumSGD(0.05), comm)

```
103     # Create a multi node optimizer from a standard Chainer optimizer.
104     optimizer = chainermn."""PUT YOUR CODE"""(
105         chainer.optimizers.MomentumSGD(0.05), comm)
106     optimizer.setup(model)
107     optimizer.add_hook(chainer.optimizer.WeightDecay(5e-4))
```

evaluator=**chainermn.create_multi_node_evaluator**
(evaluator, comm)

```
112     # Create a multi node evaluator from
113     evaluator = extensions.Evaluator(
114     evaluator = """PUT YOUR CODE"""
```


NVIDIA DEEP LEARNING INSTITUTE

マルチノードでの学習実行結果

```
[1527482343.161277] [a080:11782:0]         sys.c:744  MXM  WARN  Conflicting CPU frequencies detected,
using: 2600.39
(中略)
Start a training script using multiple nodes.
==========================================
Num process (COMM_WORLD): 4
Using GPUs
Using hierarchical communicator
Num Minibatch-size: 32
Num epoch: 4
==========================================
Using CIFAR10 dataset:
/lustre/gt00/share/lecture/20180531_dl_intro/dataset//cifar10
epoch       main/loss   validation/main/loss  main/accuracy
1           2.60504     142.057               0.115625
2           2.7311      2.31962               0.129688
3           2.80991     2.37229               0.115132
4           2.8726      2.36062               0.126563
Throughput: 173.3976045844656 [images/sec.] (10000 / 57.670923563011456)
```

2ノードx2GPUで173.4image/sec.
(対1GPUで3.35倍)

(OPTION) 追加で試す

早く終わった方のために

- プロセス間通信のアルゴリズムを変更するとどうなるか比較してみる
  - デフォルトはhierarchicalだが、ほかにも存在
  - 詳細は http://chainermn.readthedocs.io/en/latest/reference/index.html#chainermn.create_communicator を参照
- Chainer単体で2GPUを使った結果と、ChainerMNで1ノード2GPUを使った結果とを比較してみる
  - 違いがあるのかどうか検証

NVIDIA DEEP LEARNING INSTITUTE

(FYI) 性能が思うように出ないときに

プロファイラによる解析

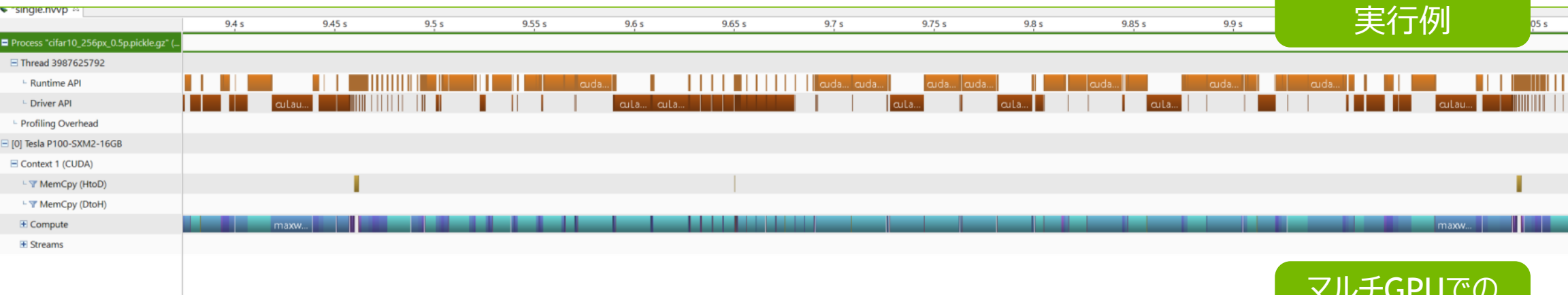
- 十分にGPUを使い切れているかどうかを確認する際、プロファイラが有効なことがある
  - コマンドはnvprof
  - 結果の可視化にはNVIDIA Visual Profilerを使う
- 実行するには対象コマンドcmdに対し、`nvprof -o ${resultfile} cmd`とすればOK
  - オプション`--profile-child-processes`を追加すると子プロセスも追跡できる
  - 子プロセスを追うときは、出力ファイル名に%pを含める（ファイル名にプロセスIDを含めるため）

NVIDIA DEEP LEARNING INSTITUTE

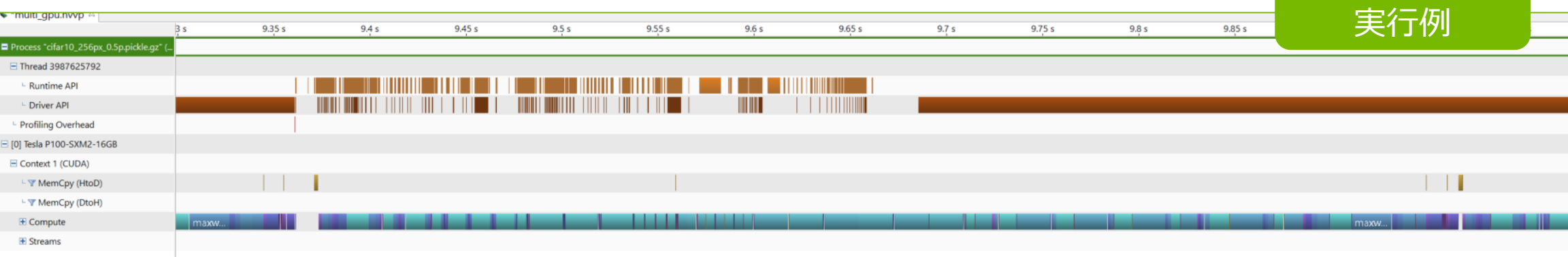
(FYI) 性能が思うように出ないときに

プロファイラによる解析

シングルGPUでの実行例



マルチGPUでの実行例

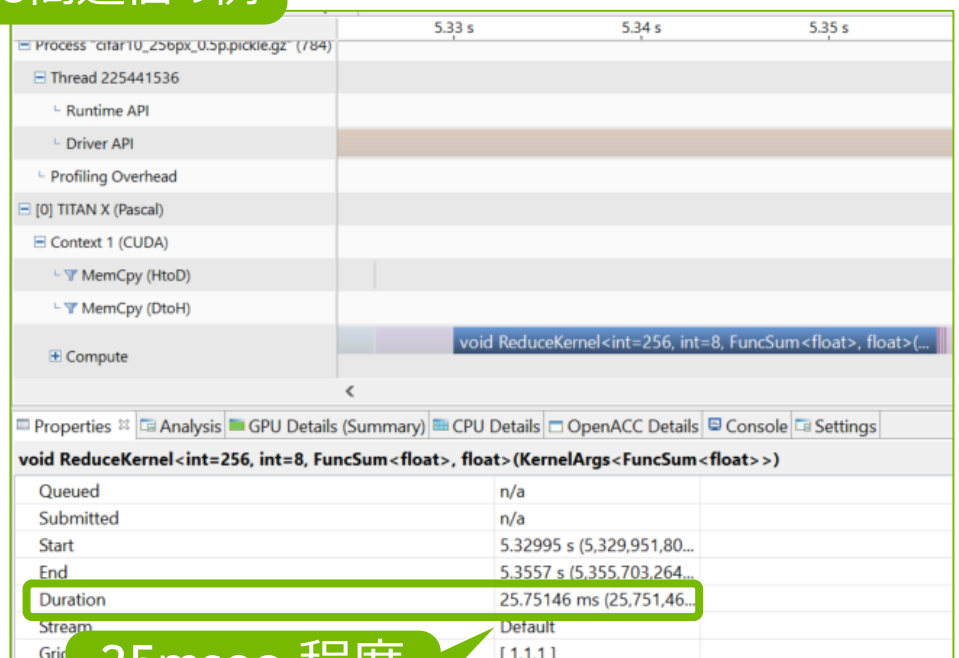


NVIDIA DEEP LEARNING INSTITUTE

(FYI) 性能が思うように出ないときに

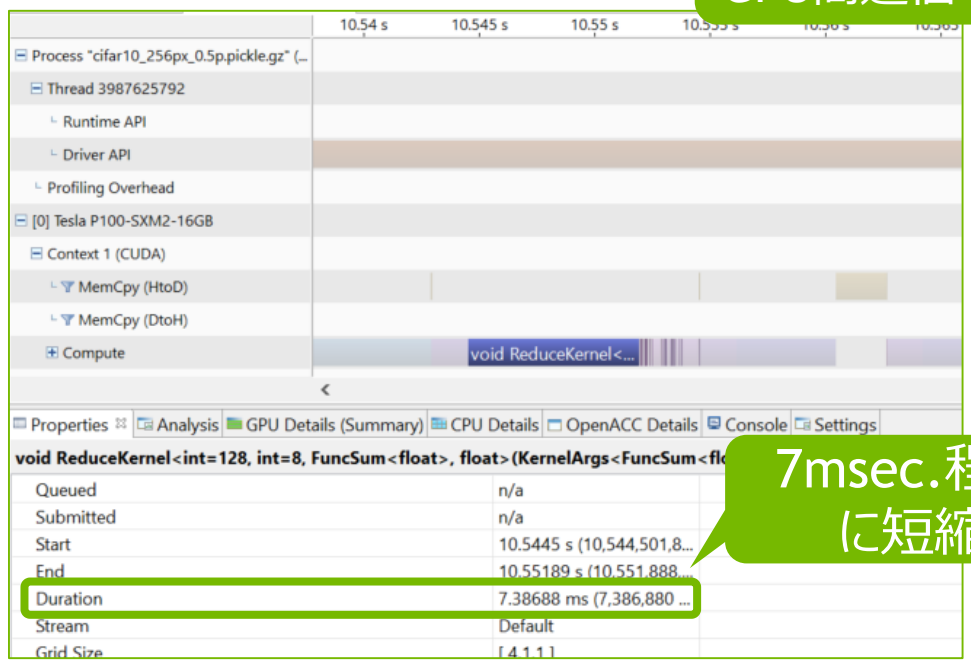
プロファイラによる解析

PCIe接続での
GPU間通信の例



25msec.程度
かかっている

NVLinkでの
GPU間通信の例



7msec.程度
に短縮

NVIDIA DEEP LEARNING INSTITUTE

NVIDIA | DEEP LEARNING INSTITUTE

www.nvidia.com/dli