Chapter 3

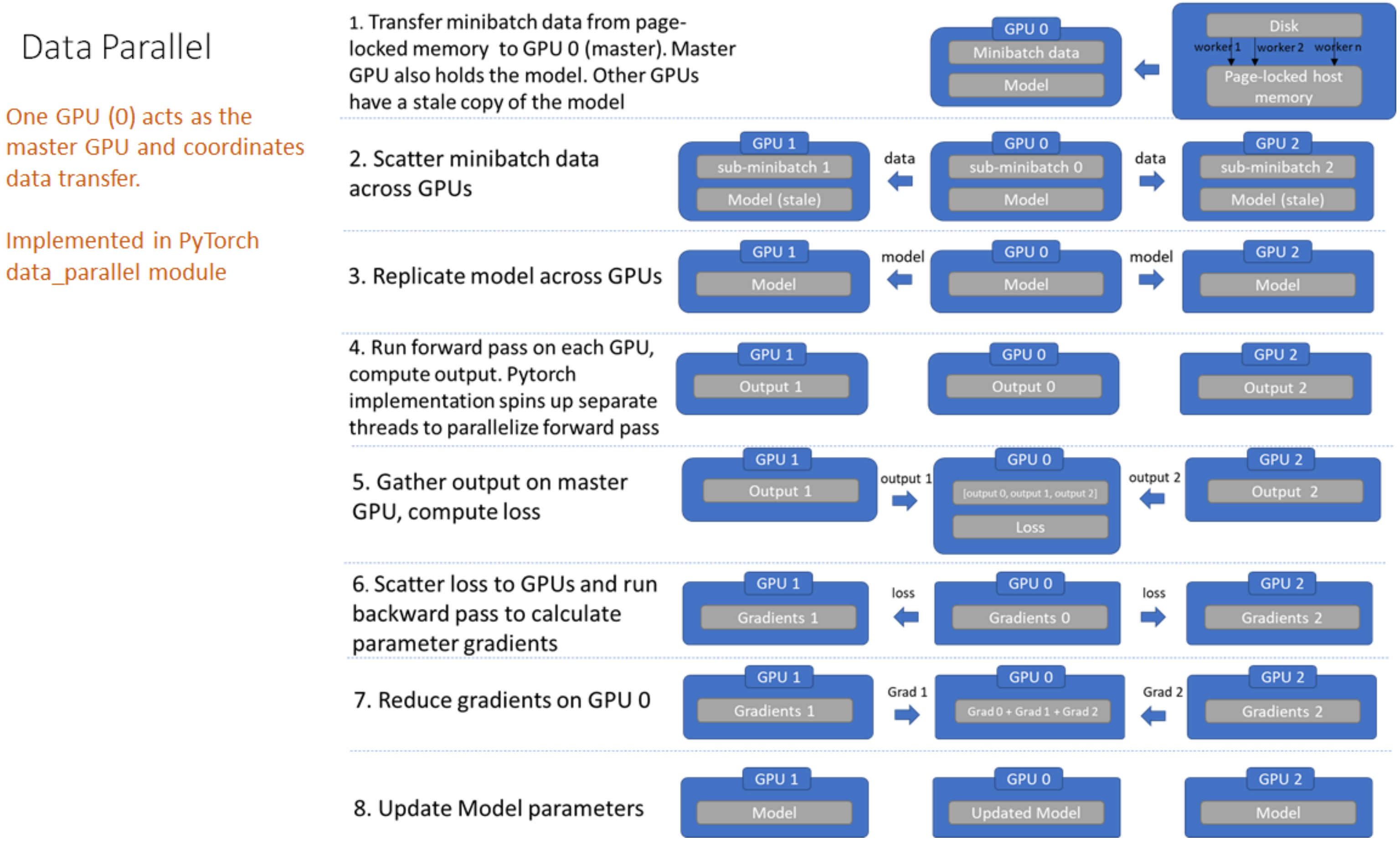
Building a Data Parallel Training and Serving Pipeline

DP와 DDP 중심으로 - Alan

Data Parallel

1. Workflow

- Main Thread에서 GPU 상의 하위 Thread로 분산 처리 하는 방법
- 모델 업데이트는 Master에서만



Data Parallel

One GPU (0) acts as the master GPU and coordinates data transfer.

Implemented in PyTorch data_parallel module

1. Transfer minibatch data from page-locked memory to GPU 0 (master). Master GPU also holds the model. Other GPUs have a stale copy of the model

2. Scatter minibatch data across GPUs

3. Replicate model across GPUs

4. Run forward pass on each GPU, compute output. Pytorch implementation spins up separate threads to parallelize forward pass

5. Gather output on master GPU, compute loss

6. Scatter loss to GPUs and run backward pass to calculate parameter gradients

7. Reduce gradients on GPU 0

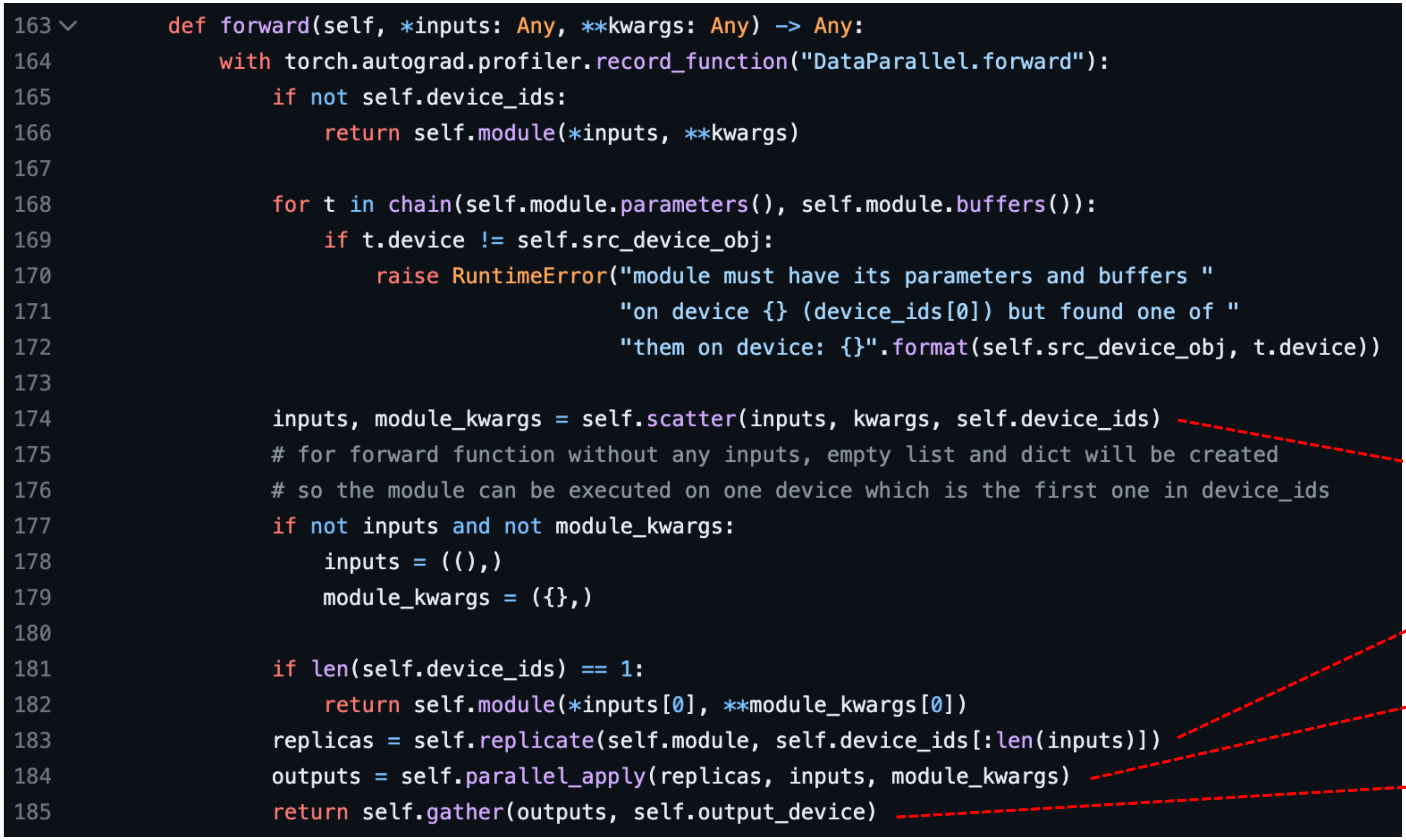
8. Update Model parameters

출처 : https://www.telesens.co/2019/04/04/distributed-data-parallel-training-using-pytorch-on-aws/
참고 : https://medium.com/huggingface/training-larger-batches-practical-tips-on-1-gpu-multi-gpu-distributed-setups-ec88c3e51255

Data Parallel

2. nn.parallel.data_parallel

- ○ Forward pass는 코드내 step 2~5
- ○ Backward pass는 C++ 코드로 구현되어 있음



```python
    def forward(self, *inputs: Any, **kwargs: Any) -> Any:
        with torch.autograd.profiler.record_function("DataParallel.forward"):
            if not self.device_ids:
                return self.module(*inputs, **kwargs)

            for t in chain(self.module.parameters(), self.module.buffers()):
                if t.device != self.src_device_obj:
                    raise RuntimeError("module must have its parameters and buffers "
                                       "on device {} (device_ids[0]) but found one of "
                                       "them on device: {}".format(self.src_device_obj, t.device))

            inputs, module_kwargs = self.scatter(inputs, kwargs, self.device_ids)
            # for forward function without any inputs, empty list and dict will be created
            # so the module can be executed on one device which is the first one in device_ids
            if not inputs and not module_kwargs:
                inputs = ((),)
                module_kwargs = ({},)

            if len(self.device_ids) == 1:
                return self.module(*inputs[0], **module_kwargs[0])
            replicas = self.replicate(self.module, self.device_ids[:len(inputs)])
            outputs = self.parallel_apply(replicas, inputs, module_kwargs)
            return self.gather(outputs, self.output_device)
```

Step2. Scatter minibatch data

Step3. Replcatte model

Step4. Run forward pass

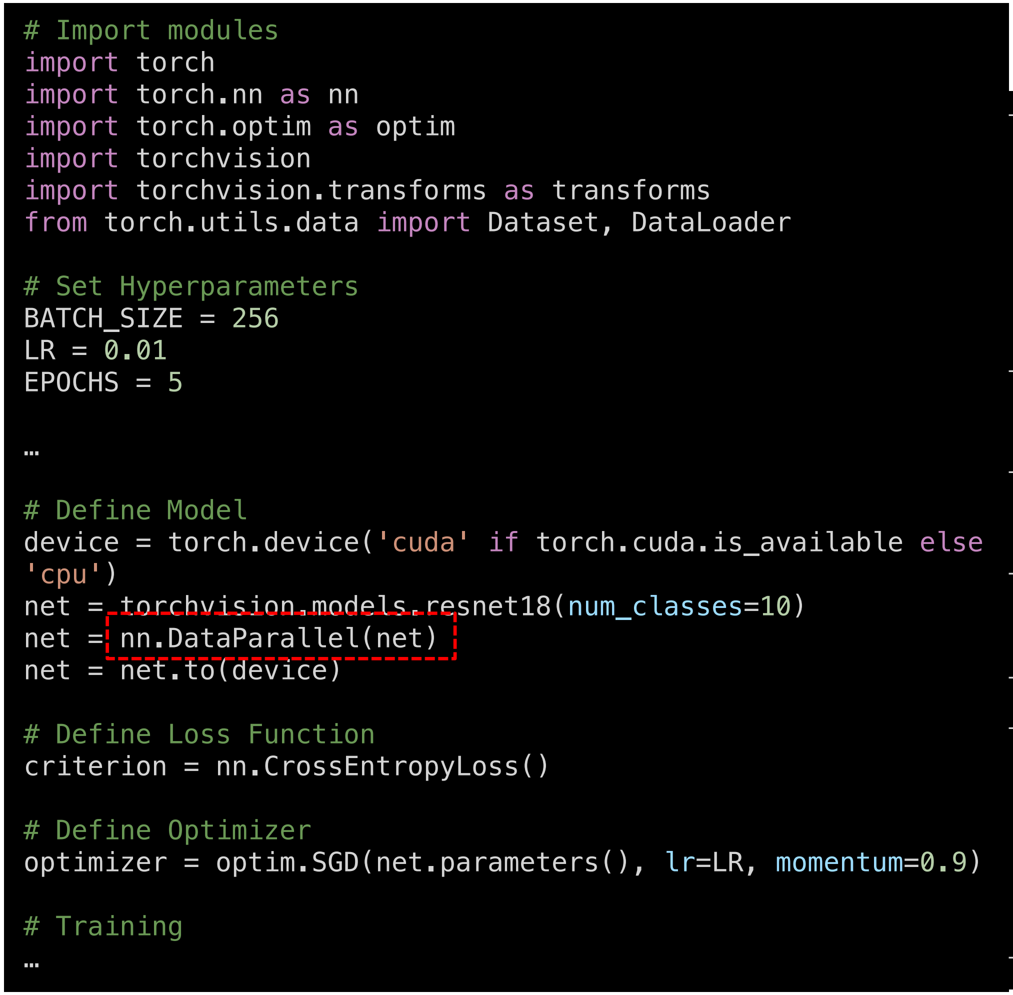
Step5. Gather output

출처https://github.com/pytorch/pytorch/blob/main/torch/nn/parallel/distributed.py

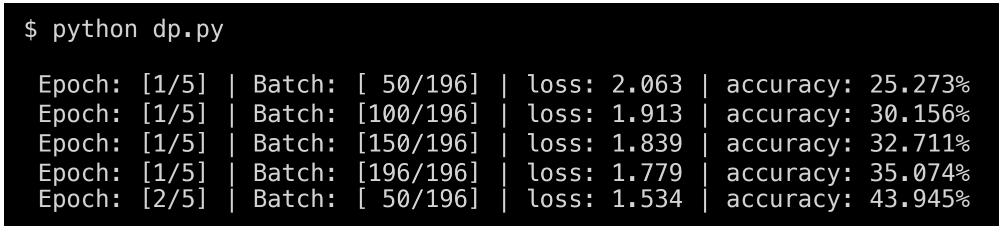
Data Parallel

3. Code 구현

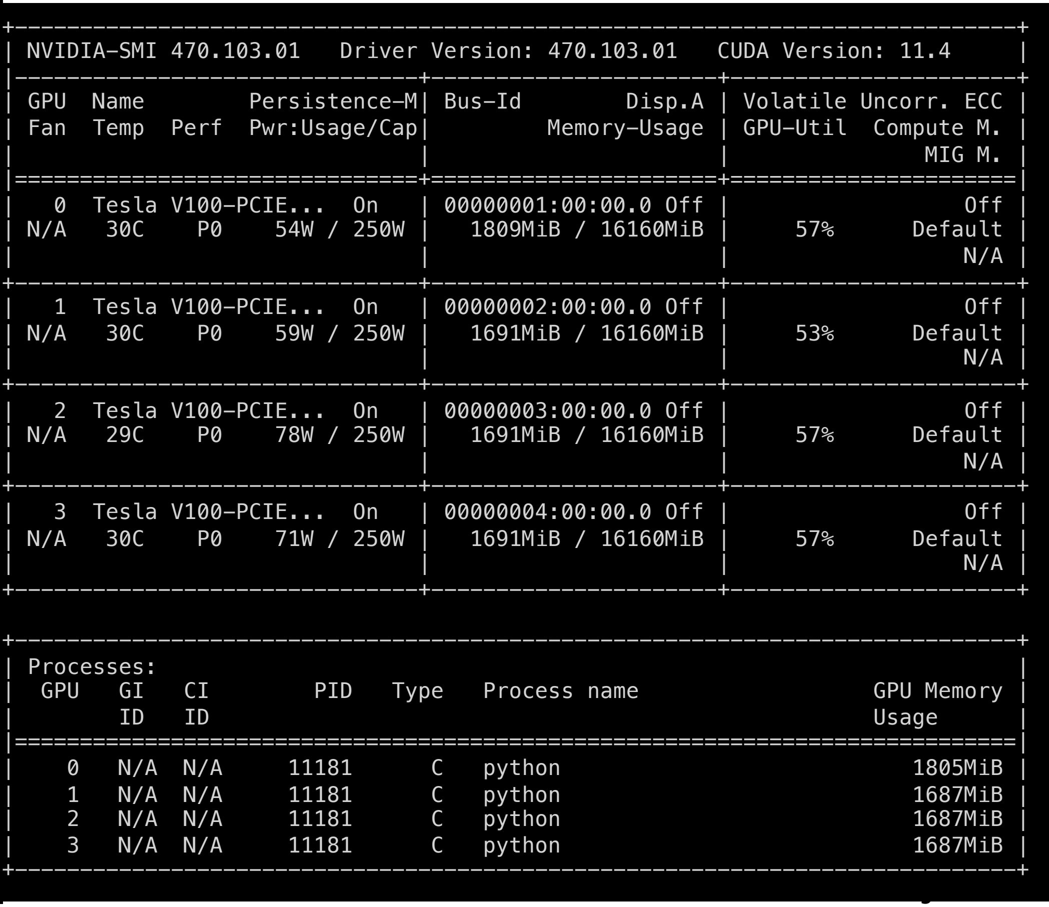
- nn.DataParallel()로 감싸면 끝!



```python
# Import modules
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import Dataset, DataLoader

# Set Hyperparameters
BATCH_SIZE = 256
LR = 0.01
EPOCHS = 5

…

# Define Model
device = torch.device('cuda' if torch.cuda.is_available else
'cpu')
net = torchvision.models.resnet18(num_classes=10)
net = nn.DataParallel(net)
net = net.to(device)

# Define Loss Function
criterion = nn.CrossEntropyLoss()

# Define Optimizer
optimizer = optim.SGD(net.parameters(), lr=LR, momentum=0.9)

# Training
…
```



```
$ python dp.py

 Epoch: [1/5] | Batch: [ 50/196] | loss: 2.063 | accuracy: 25.273%
 Epoch: [1/5] | Batch: [100/196] | loss: 1.913 | accuracy: 30.156%
 Epoch: [1/5] | Batch: [150/196] | loss: 1.839 | accuracy: 32.711%
 Epoch: [1/5] | Batch: [196/196] | loss: 1.779 | accuracy: 35.074%
 Epoch: [2/5] | Batch: [ 50/196] | loss: 1.534 | accuracy: 43.945%
```



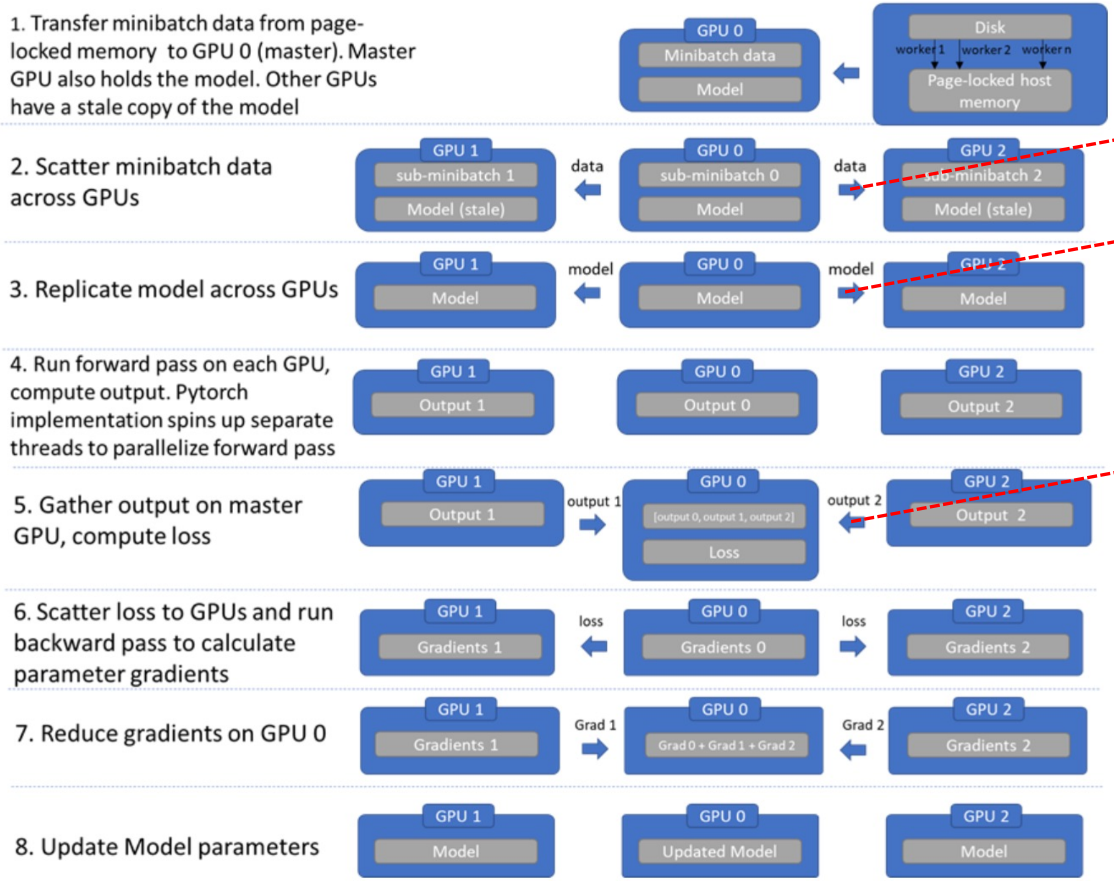
```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.103.01   Driver Version: 470.103.01   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-PCIE...  On   | 00000001:00:00.0 Off |                  Off |
| N/A   30C    P0    54W / 250W |   1809MiB / 16160MiB |     57%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-PCIE...  On   | 00000002:00:00.0 Off |                  Off |
| N/A   30C    P0    59W / 250W |   1691MiB / 16160MiB |     53%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-PCIE...  On   | 00000003:00:00.0 Off |                  Off |
| N/A   29C    P0    78W / 250W |   1691MiB / 16160MiB |     57%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-PCIE...  On   | 00000004:00:00.0 Off |                  Off |
| N/A   30C    P0    71W / 250W |   1691MiB / 16160MiB |     57%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     11181      C   python                           1805MiB |
|    1   N/A  N/A     11181      C   python                           1687MiB |
|    2   N/A  N/A     11181      C   python                           1687MiB |
|    3   N/A  N/A     11181      C   python                           1687MiB |
+-----------------------------------------------------------------------------+
```

출처http://10.40.21.246:32800/tutorials/pytorch4

Data Parallel

4. Inefficiencies

- ○ Dataset과 Model을 모두 복제
- ○ 자원 사용의 불균형



1. Transfer minibatch data from page-locked memory to GPU 0 (master). Master GPU also holds the model. Other GPUs have a stale copy of the model

2. Scatter minibatch data across GPUs

3. Replicate model across GPUs

4. Run forward pass on each GPU, compute output. Pytorch implementation spins up separate threads to parallelize forward pass

5. Gather output on master GPU, compute loss

6. Scatter loss to GPUs and run backward pass to calculate parameter gradients

7. Reduce gradients on GPU 0

8. Update Model parameters

중복 데이터 사본

Forward pass 전 GPU간 모델 복제

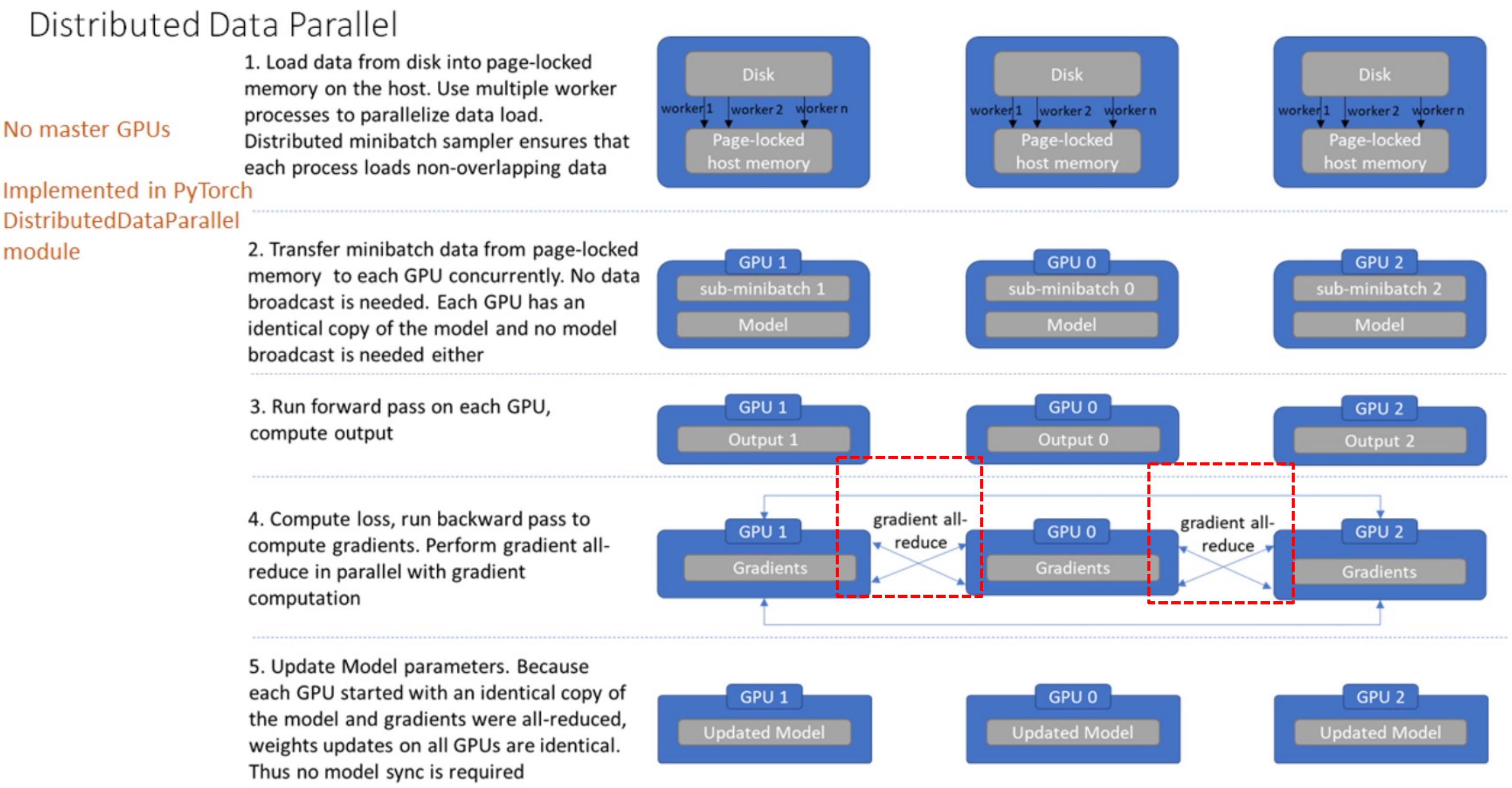
Logit을 master에 모은 후 loss 계산
Master 와 worker간의 메모리 불균형
Master에서만 loss 계산, 자원사용 불균형

출처 : https://www.telesens.co/2019/04/04/distributed-data-parallel-training-using-pytorch-on-aws/
참고 : https://medium.com/huggingface/training-larger-batches-practical-tips-on-1-gpu-multi-gpu-distributed-setups-ec88c3e51255

Distributed Data Parallel

1. Workflow

- ○ Master가 없으며 모든 프로세스가 동일한 작업을 수행
- ○ Node간 전송 작업은 Gradients를 교환할 때 뿐 <- DP에 비해 노드간 전송량 획기적으로 줄음
- ○ Gradients 교환은 All-reduce 기법을 통해 모든 노드들이 참여



Distributed Data Parallel

No master GPUs

Implemented in PyTorch DistributedDataParallel module

1. Load data from disk into page-locked memory on the host. Use multiple worker processes to parallelize data load. Distributed minibatch sampler ensures that each process loads non-overlapping data

2. Transfer minibatch data from page-locked memory to each GPU concurrently. No data broadcast is needed. Each GPU has an identical copy of the model and no model broadcast is needed either

3. Run forward pass on each GPU, compute output

4. Compute loss, run backward pass to compute gradients. Perform gradient all-reduce in parallel with gradient computation

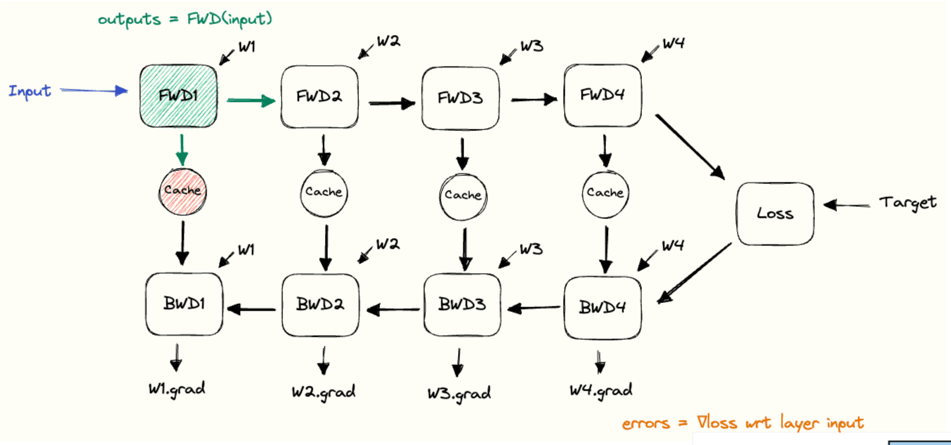
5. Update Model parameters. Because each GPU started with an identical copy of the model and gradients were all-reduced, weights updates on all GPUs are identical. Thus no model sync is required

출처 : https://www.telesens.co/2019/04/04/distributed-data-parallel-training-using-pytorch-on-aws/
참고 : https://medium.com/huggingface/training-larger-batches-practical-tips-on-1-gpu-multi-gpu-distributed-setups-ec88c3e51255

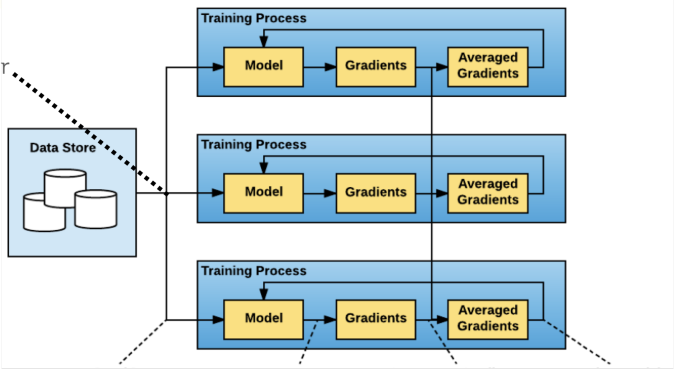
Distributed Data Parallel

1. Workflow

- ○ Gradient 교환은 backward pass 가 종료되고 진행됨
- ○ 그럼 모든 노드에서 backward가 끝나기를 기다려야 하나?



1. 데이터 Scatter



2. Forward 수행 3. Backward 수행 4. Gradient All-reduce 5. 파라미터 업데이트

```
import torch
import torch.nn as nn
import torch.optim as optim

net = nn.Linear(10, 10)
opt = optim.SGD(net.parameters())

input, target = ...
out = net(input)
loss = nn.MSELoss(out, target)

loss.backward()

opt.step()
```

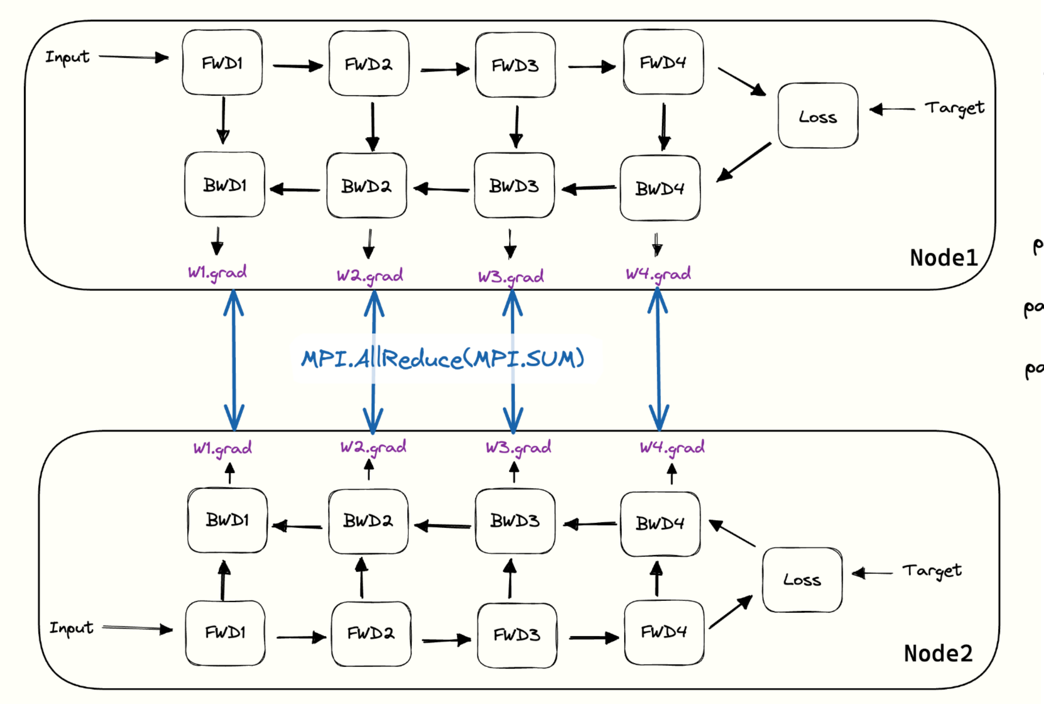
Which one is better?

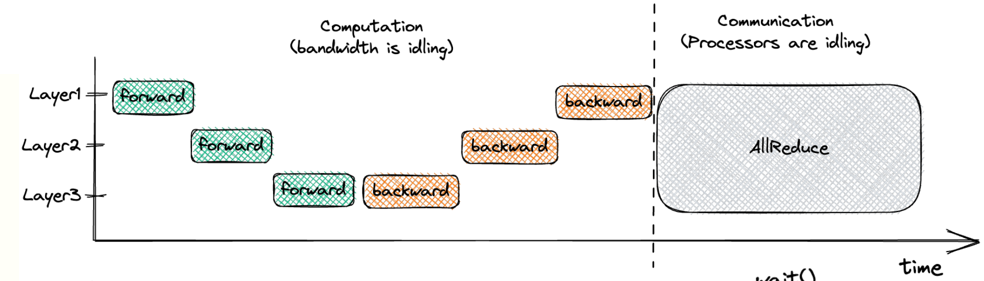
출처 : https://siboehm.com/articles/22/data-parallel-training

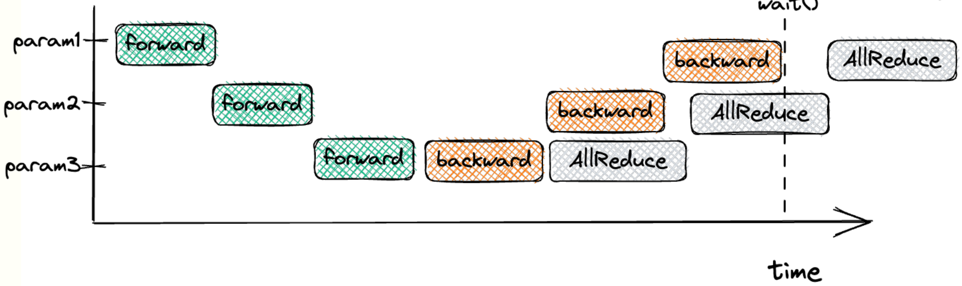
Distributed Data Parallel

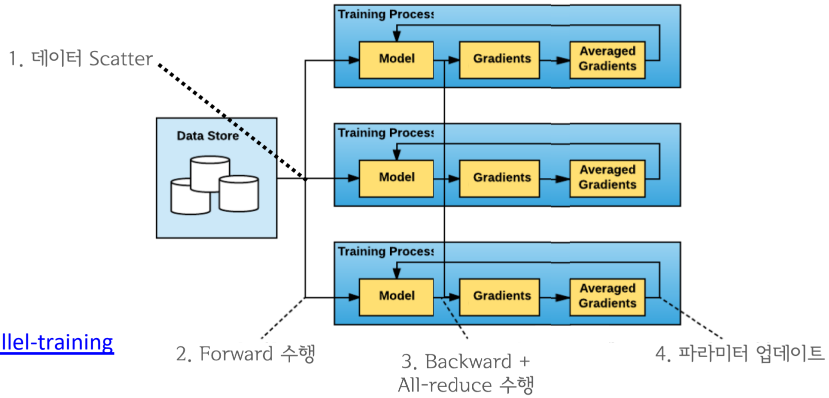
1. Workflow

- ○ Backward 연산은 Network의 뒤부터 진행됨 <- 그럼 먼저 끝난 레이어들 부터 동기화 하면 안됨?
- ○ Backward 연산이 전체적으로 가장 무거운 연산이므로 어느 정도 모아서(Gradient Bucketing) 동기화하면 기다렸다 한번에 전송하는것보다 전체적인 동기화 시간을 감소 할 수 있음









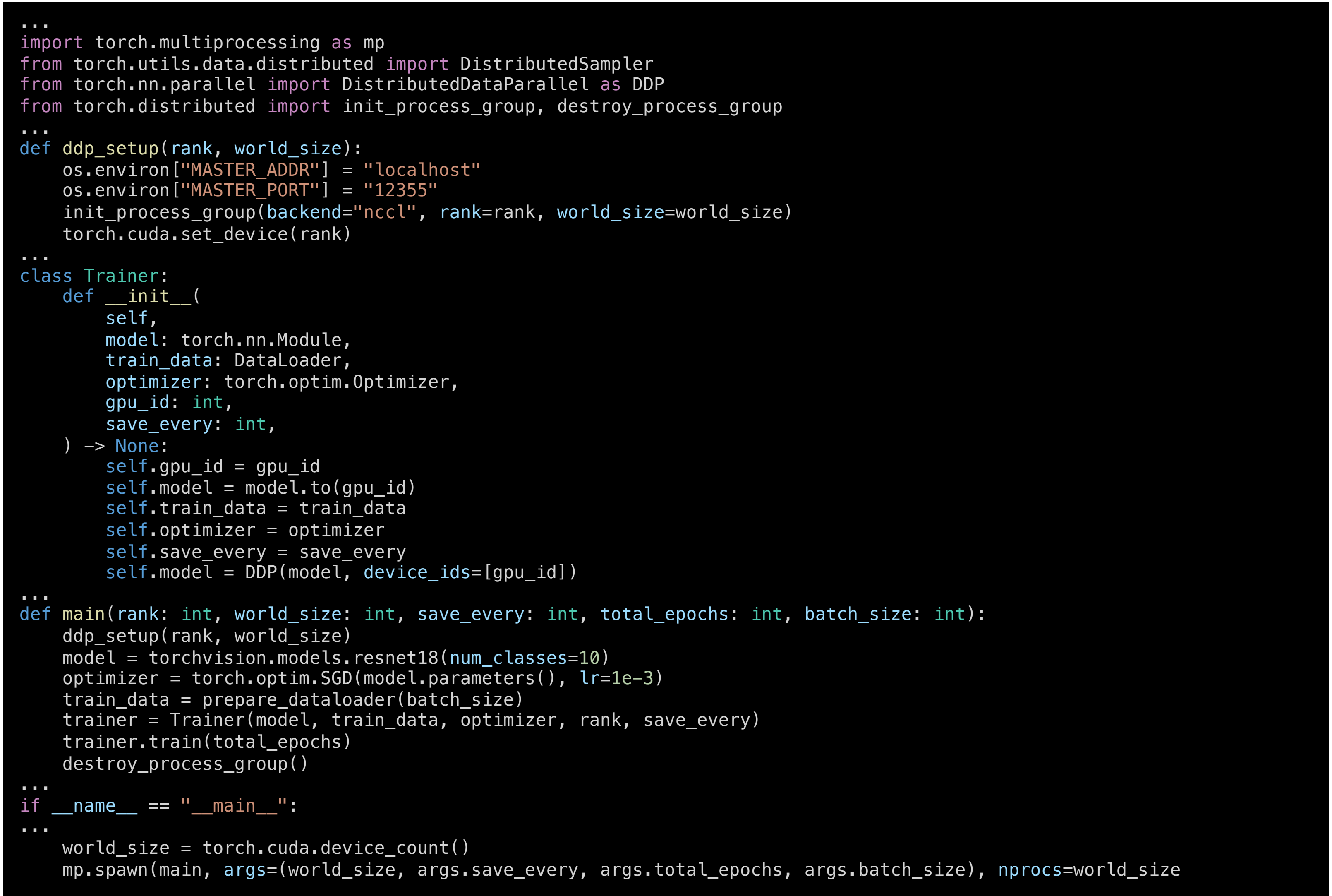
출처 : https://siboehm.com/articles/22/data-parallel-training

Distributed Data Parallel

출처 : https://github.com/pytorch/examples/tree/main/distributed/ddp-tutorial-series
참고 : https://confluence.tde.sktelecom.com/display/GLMMODEL/5.+Distributed+Training

2. torch.multiprocessing.spawn

- torch.multiprocessing.spawn을 통해 subprocess 분기
- DistributedDataParallel로 모델을 래핑

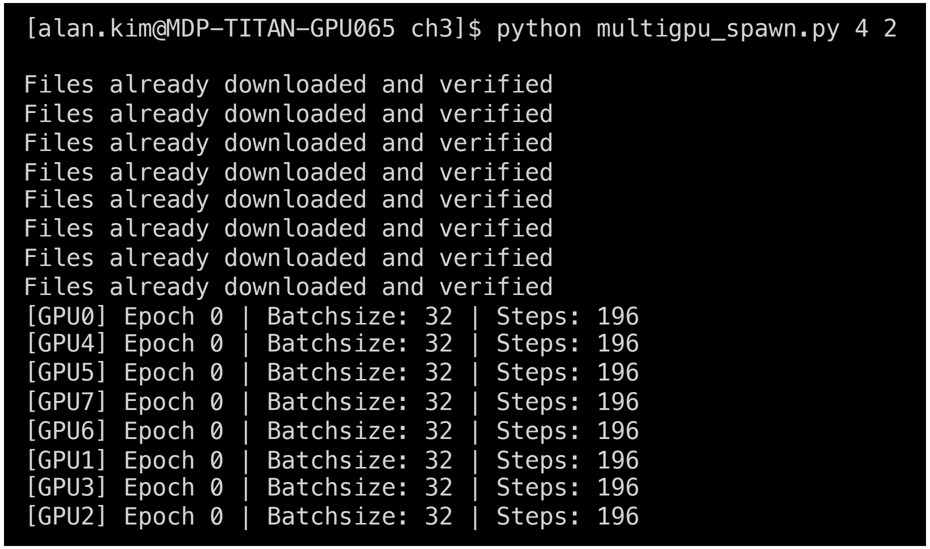


```python
...
import torch.multiprocessing as mp
from torch.utils.data.distributed import DistributedSampler
from torch.nn.parallel import DistributedDataParallel as DDP
from torch.distributed import init_process_group, destroy_process_group
...
def ddp_setup(rank, world_size):
    os.environ["MASTER_ADDR"] = "localhost"
    os.environ["MASTER_PORT"] = "12355"
    init_process_group(backend="nccl", rank=rank, world_size=world_size)
    torch.cuda.set_device(rank)
...
class Trainer:
    def __init__(
        self,
        model: torch.nn.Module,
        train_data: DataLoader,
        optimizer: torch.optim.Optimizer,
        gpu_id: int,
        save_every: int,
    ) -> None:
        self.gpu_id = gpu_id
        self.model = model.to(gpu_id)
        self.train_data = train_data
        self.optimizer = optimizer
        self.save_every = save_every
        self.model = DDP(model, device_ids=[gpu_id])
...
def main(rank: int, world_size: int, save_every: int, total_epochs: int, batch_size: int):
    ddp_setup(rank, world_size)
    model = torchvision.models.resnet18(num_classes=10)
    optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)
    train_data = prepare_dataloader(batch_size)
    trainer = Trainer(model, train_data, optimizer, rank, save_every)
    trainer.train(total_epochs)
    destroy_process_group()
...
if __name__ == "__main__":
...
    world_size = torch.cuda.device_count()
    mp.spawn(main, args=(world_size, args.save_every, args.total_epochs, args.batch_size), nprocs=world_size
```

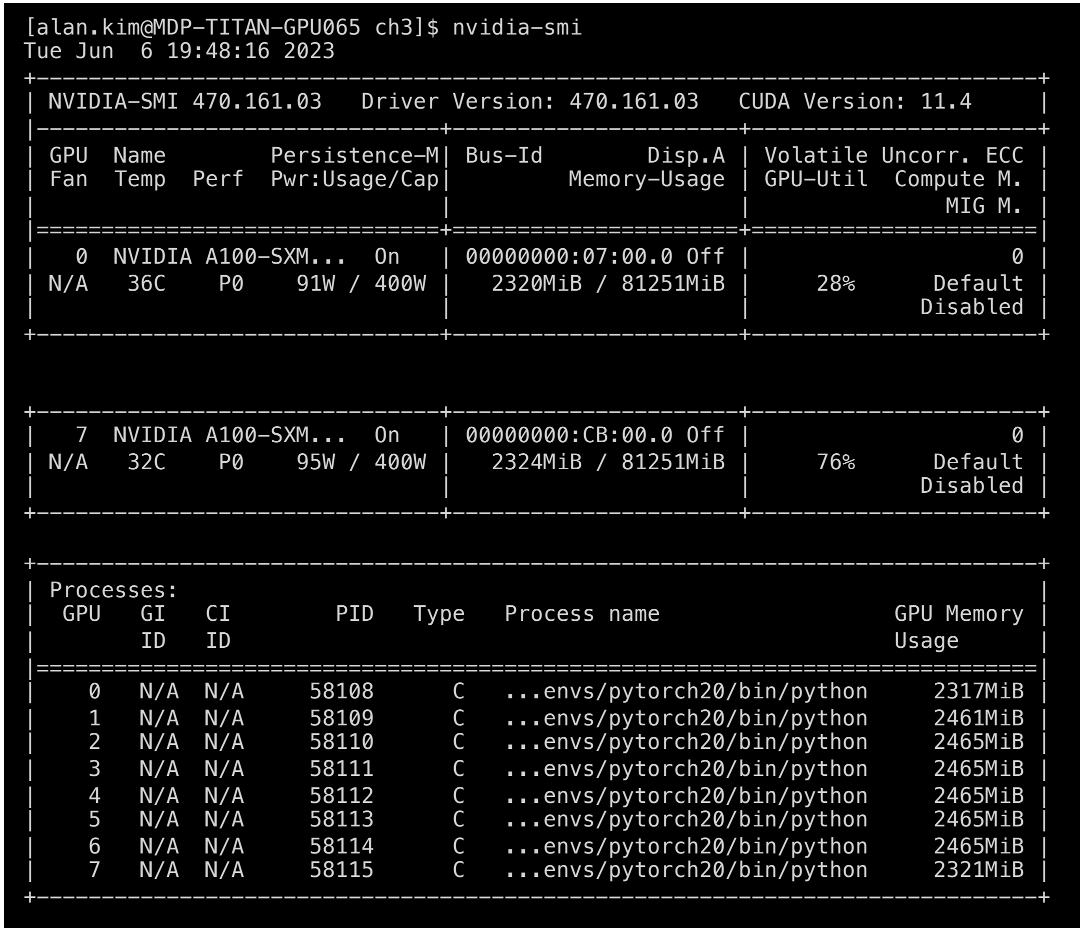
Distributed Data Parallel

2. torch.multiprocessing.spawn

- DP에 비해 GPU 메모리 사용율이 일정



```
[alan.kim@MDP-TITAN-GPU065 ch3]$ python multigpu_spawn.py 4 2

Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
[GPU0] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU4] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU5] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU7] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU6] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU1] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU3] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU2] Epoch 0 | Batchsize: 32 | Steps: 196
```



```
[alan.kim@MDP-TITAN-GPU065 ch3]$ nvidia-smi
Tue Jun  6 19:48:16 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.161.03   Driver Version: 470.161.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM...  On   | 00000000:07:00.0 Off |                    0 |
| N/A   36C    P0    91W / 400W |   2320MiB / 81251MiB |     28%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+


+-------------------------------+----------------------+----------------------+
|   7  NVIDIA A100-SXM...  On   | 00000000:CB:00.0 Off |                    0 |
| N/A   32C    P0    95W / 400W |   2324MiB / 81251MiB |     76%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     58108      C   ...envs/pytorch20/bin/python     2317MiB |
|    1   N/A  N/A     58109      C   ...envs/pytorch20/bin/python     2461MiB |
|    2   N/A  N/A     58110      C   ...envs/pytorch20/bin/python     2465MiB |
|    3   N/A  N/A     58111      C   ...envs/pytorch20/bin/python     2465MiB |
|    4   N/A  N/A     58112      C   ...envs/pytorch20/bin/python     2465MiB |
|    5   N/A  N/A     58113      C   ...envs/pytorch20/bin/python     2465MiB |
|    6   N/A  N/A     58114      C   ...envs/pytorch20/bin/python     2465MiB |
|    7   N/A  N/A     58115      C   ...envs/pytorch20/bin/python     2321MiB |
+-----------------------------------------------------------------------------+
```

출처 : https://github.com/pytorch/examples/tree/main/distributed/ddp-tutorial-series
참고 : https://confluence.tde.sktelecom.com/display/GLMMODEL/5.+Distributed+Training

Distributed Data Parallel

2. torchrun (Elastic launch)

- torch.distributed.launch의 확장 기능으로 torchruntorch.distributed.launch을 제공
  - Worker의 RANK가 자동 할당 (이를 통해 WORLD_SIZE 확인 가능 ← 별도로 안줘도 됨)
  - Failover 기능 추가
  - 최소 노드 및 최대 노드를 지정하여 노드수를 변경하며 분산 학습 가능
  - Heterogeneous한 자원 구성 사용 가능 (예: 8GPU 노드 + 4GPU 노드로 12 GPU 분산 학습 가능)

Torch.distributed.launch

torchrun

- Torchrun이 RANK 정보를 전달 하므로 LOCAL_RANK, WORLD_SIZE와 같은 정보를 사전에 정의할 필요가 없다

```
$ python -m torch.distributed.launch --use-env train_script.py
```

```
$ torchrun train_script.py
```

```
import argparse parser = argparse.ArgumentParser() parser.ad
d_argument("--local-rank", type=int) args = parser.parse_args
() local_rank = args.local_rank
```

```
import os
local_rank = int(os.environ["LOCAL_RANK"])
```

출처 : https://pytorch.org/docs/stable/elastic/run.html

Distributed Data Parallel

2. torchrun

- Single-node multi-worker

```
torchrun
    --standalone
    --nnodes=1
    --nproc-per-node=$NUM_TRAINERS
    YOUR_TRAINING_SCRIPT.py (--arg1 ... train script args...)
```

출처 : https://pytorch.org/docs/stable/elastic/run.html

Distributed Data Parallel

2. torchrun

- rendezvous backend
  - Multi process 학습을 진행할 경우 fail-over나 탄력적인 노드 구성을 위해 학습 그룹에 포함된 노드간 정보를 공유하기 위한 방법
    - --rdzv-id: 고유한 작업 ID (작업에 참여하는 모든 노드가 공유)
    - --rdzv-backend: 랑데부 구현체, 보통 c10d 권장
    - --rdzv-endpoint: 랑데부 백엔드가 실행 중인 엔드포인트. 보통 형태로 host:port

- Stacked single-node multi-worker
  - 하나의 노드 내에서 서로 다른 분산 학습이 실행 되는 경우

```
torchrun
    --rdzv-backend=c10d
    --rdzv-endpoint=localhost:0
    --nnodes=1
    --nproc-per-node=$NUM_TRAINERS
    YOUR_TRAINING_SCRIPT.py (--arg1 ... train script args...)
```

출처 : https://pytorch.org/docs/stable/elastic/run.html

Distributed Data Parallel

2. torchrun

- rendezvous backend failover
  - Worker를 추가, 삭제없이 고정하고 3번의 장애를 허용하는 예

```
torchrun
    --nnodes=$NUM_NODES
    --nproc-per-node=$NUM_TRAINERS
    --max-restarts=3
    --rdzv-id=$JOB_ID
    --rdzv-backend=c10d
    --rdzv-endpoint=$HOST_NODE_ADDR
    YOUR_TRAINING_SCRIPT.py (--arg1 ... train script args...)
```

- rendezvous backend Elastic
  - 분산 학습을 구성하는 최소 및 최대 노드 수를 지정
  - 학습중이라도 새로운 노드가 추가 되면 포함하여 학습 가능

```
torchrun
    --nnodes=1:4
    --nproc-per-node=$NUM_TRAINERS
    --max-restarts=3
    --rdzv-id=$JOB_ID
    --rdzv-backend=c10d
    --rdzv-endpoint=$HOST_NODE_ADDR
    YOUR_TRAINING_SCRIPT.py (--arg1 ... train script args...)
```

출처 : https://pytorch.org/docs/stable/elastic/run.html

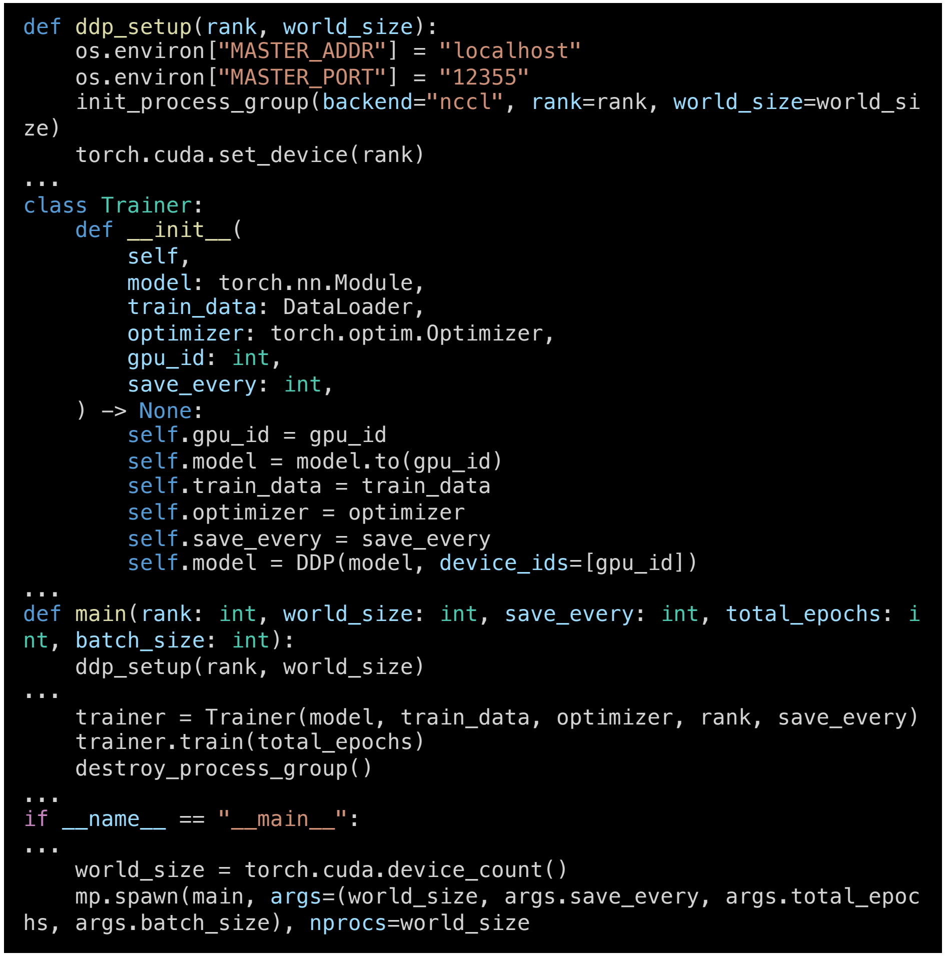
Distributed Data Parallel

출처 : https://github.com/pytorch/examples/tree/main/distributed/ddp-tutorial-series
참고 : https://confluence.tde.sktelecom.com/display/GLMMODEL/5.+Distributed+Training

2. torchrun

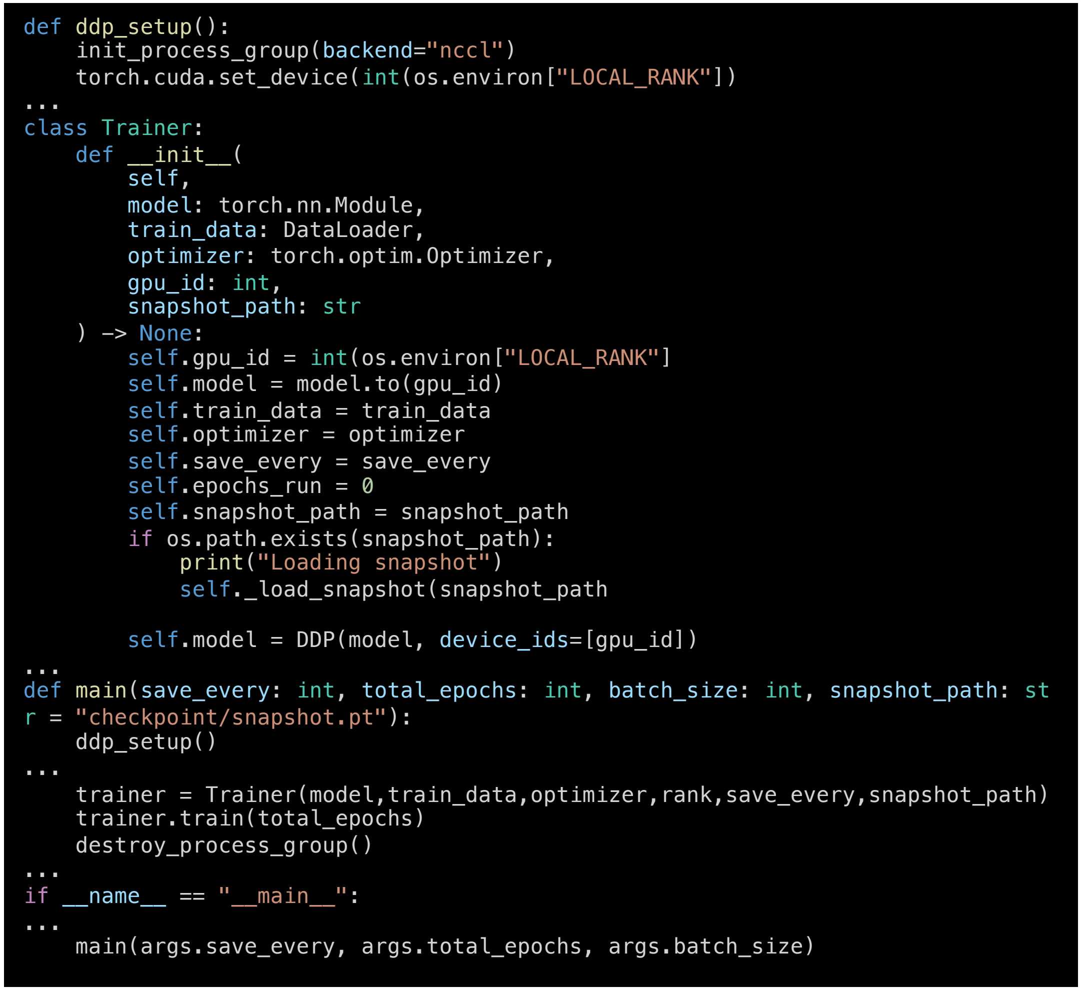
- Torch distributed laucher를 통해 subprocess 분기
- RANK 자동 할당

spawn



```
def ddp_setup(rank, world_size):
    os.environ["MASTER_ADDR"] = "localhost"
    os.environ["MASTER_PORT"] = "12355"
    init_process_group(backend="nccl", rank=rank, world_size=world_si
ze)
    torch.cuda.set_device(rank)
...
class Trainer:
    def __init__(
        self,
        model: torch.nn.Module,
        train_data: DataLoader,
        optimizer: torch.optim.Optimizer,
        gpu_id: int,
        save_every: int,
    ) -> None:
        self.gpu_id = gpu_id
        self.model = model.to(gpu_id)
        self.train_data = train_data
        self.optimizer = optimizer
        self.save_every = save_every
        self.model = DDP(model, device_ids=[gpu_id])
...
def main(rank: int, world_size: int, save_every: int, total_epochs: i
nt, batch_size: int):
    ddp_setup(rank, world_size)
...
    trainer = Trainer(model, train_data, optimizer, rank, save_every)
    trainer.train(total_epochs)
    destroy_process_group()
...
if __name__ == "__main__":
...
    world_size = torch.cuda.device_count()
    mp.spawn(main, args=(world_size, args.save_every, args.total_epoc
hs, args.batch_size), nprocs=world_size
```

torchrun



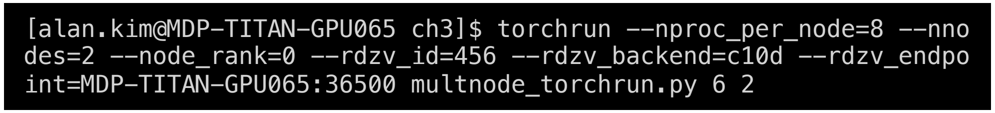
```
def ddp_setup():
    init_process_group(backend="nccl")
    torch.cuda.set_device(int(os.environ["LOCAL_RANK"])
...
class Trainer:
    def __init__(
        self,
        model: torch.nn.Module,
        train_data: DataLoader,
        optimizer: torch.optim.Optimizer,
        gpu_id: int,
        snapshot_path: str
    ) -> None:
        self.gpu_id = int(os.environ["LOCAL_RANK"]
        self.model = model.to(gpu_id)
        self.train_data = train_data
        self.optimizer = optimizer
        self.save_every = save_every
        self.epochs_run = 0
        self.snapshot_path = snapshot_path
        if os.path.exists(snapshot_path):
            print("Loading snapshot")
            self._load_snapshot(snapshot_path

        self.model = DDP(model, device_ids=[gpu_id])
...
def main(save_every: int, total_epochs: int, batch_size: int, snapshot_path: st
r = "checkpoint/snapshot.pt"):
    ddp_setup()
...
    trainer = Trainer(model,train_data,optimizer,rank,save_every,snapshot_path)
    trainer.train(total_epochs)
    destroy_process_group()
...
if __name__ == "__main__":
...
    main(args.save_every, args.total_epochs, args.batch_size)
```

Distributed Data Parallel

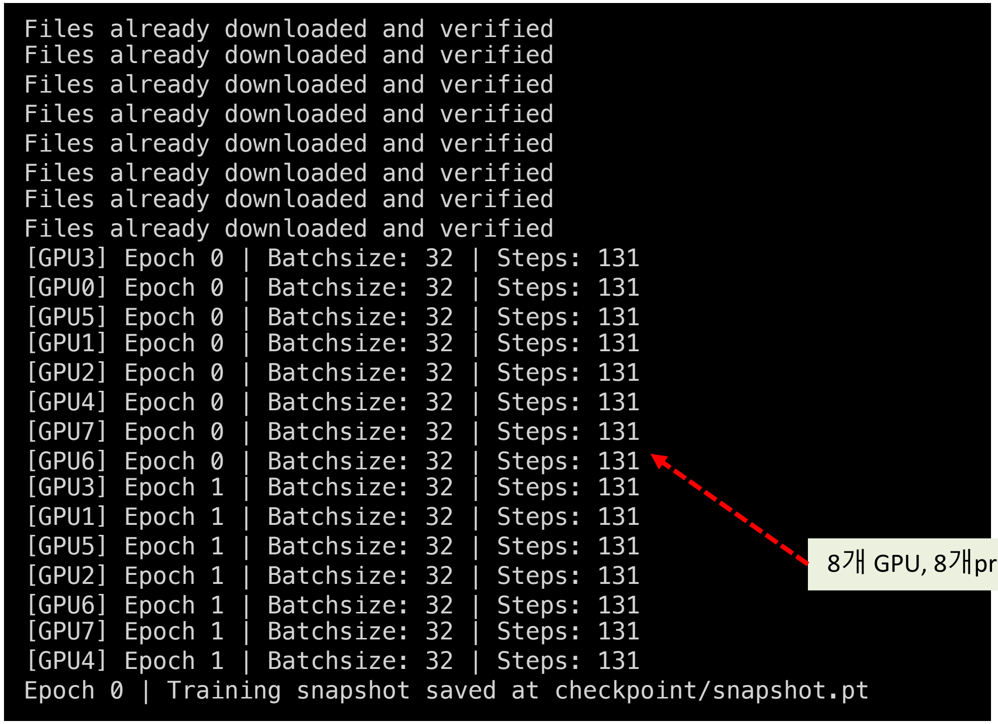
2. torchrun

- 2 node, Heterogeneous GPU 자원 학습

Master Node



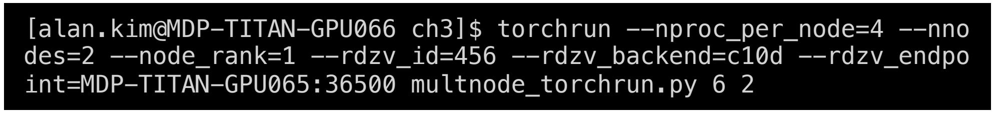
```
[alan.kim@MDP-TITAN-GPU065 ch3]$ torchrun --nproc_per_node=8 --nno
des=2 --node_rank=0 --rdzv_id=456 --rdzv_backend=c10d --rdzv_endpo
int=MDP-TITAN-GPU065:36500 multnode_torchrun.py 6 2
```



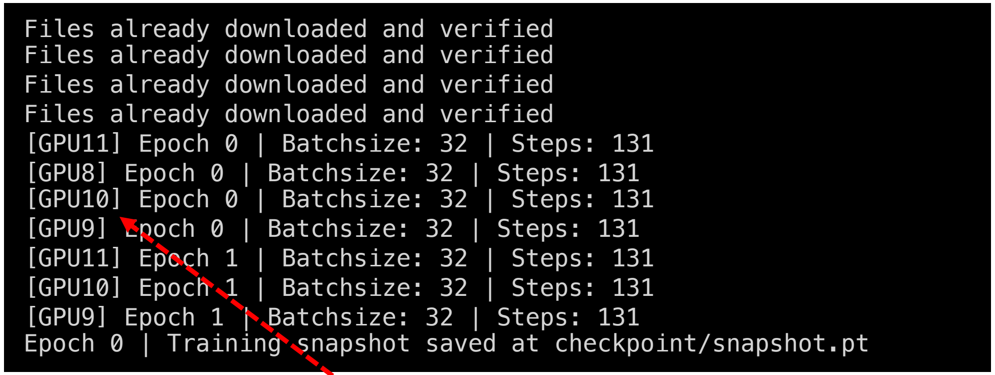
```
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
[GPU3] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU0] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU5] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU1] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU2] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU4] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU7] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU6] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU3] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU1] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU5] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU2] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU6] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU7] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU4] Epoch 1 | Batchsize: 32 | Steps: 131
Epoch 0 | Training snapshot saved at checkpoint/snapshot.pt
```

8개 GPU, 8개process로 분산 학습

Worker Node



```
[alan.kim@MDP-TITAN-GPU066 ch3]$ torchrun --nproc_per_node=4 --nno
des=2 --node_rank=1 --rdzv_id=456 --rdzv_backend=c10d --rdzv_endpo
int=MDP-TITAN-GPU065:36500 multnode_torchrun.py 6 2
```



```
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
[GPU11] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU8] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU10] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU9] Epoch 0 | Batchsize: 32 | Steps: 131
[GPU11] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU10] Epoch 1 | Batchsize: 32 | Steps: 131
[GPU9] Epoch 1 | Batchsize: 32 | Steps: 131
Epoch 0 | Training snapshot saved at checkpoint/snapshot.pt
```

4개 GPU, 4개process로 분산 학습

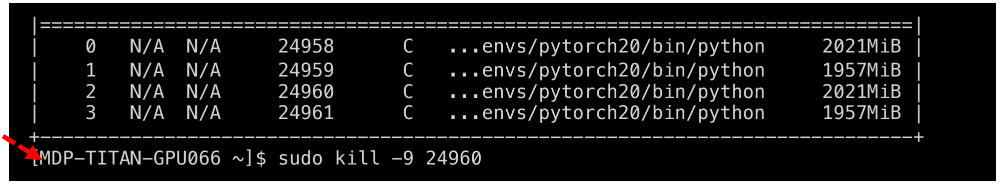
Distributed Data Parallel

2. torchrun

- Failover 구현

2번 노드에서 process 1개 kill

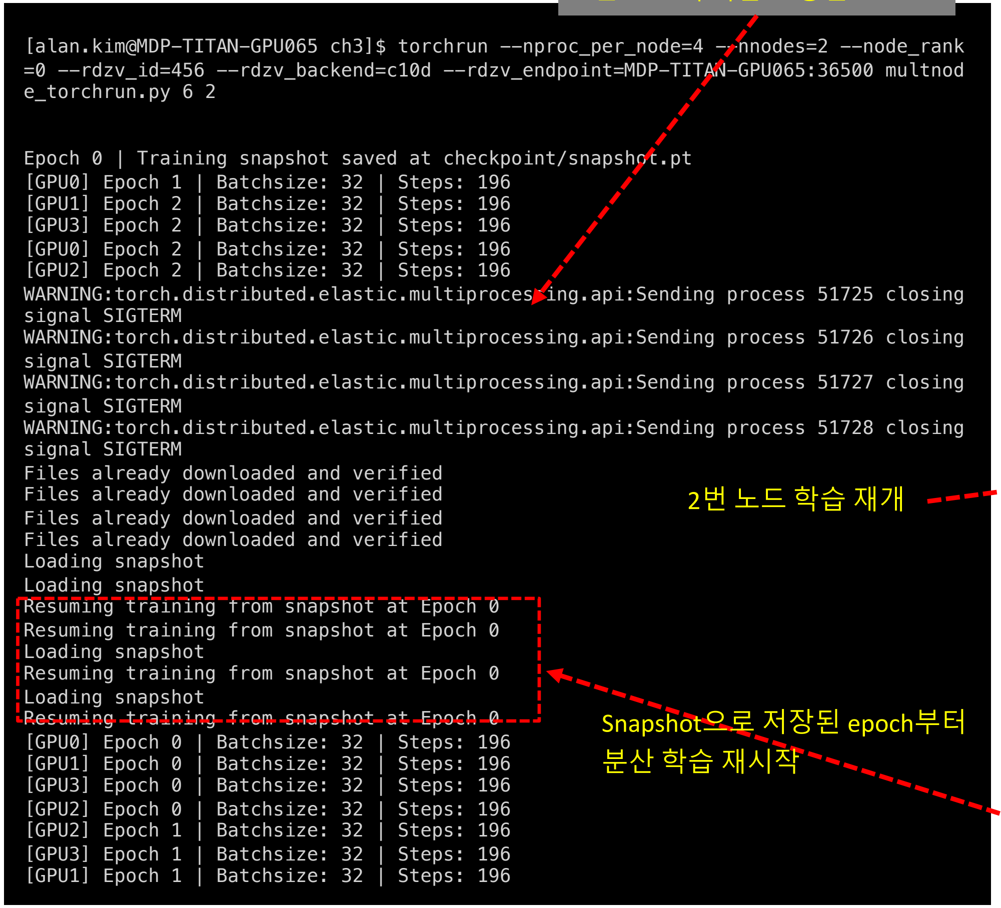
Worker Node



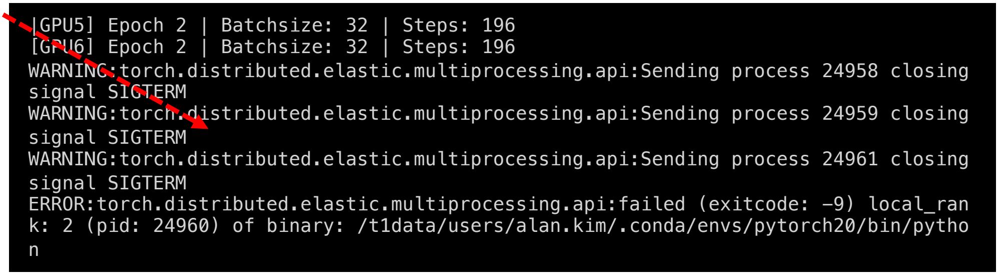
```
|===============================================================================|
|    0   N/A  N/A     24958      C   ...envs/pytorch20/bin/python     2021MiB |
|    1   N/A  N/A     24959      C   ...envs/pytorch20/bin/python     1957MiB |
|    2   N/A  N/A     24960      C   ...envs/pytorch20/bin/python     2021MiB |
|    3   N/A  N/A     24961      C   ...envs/pytorch20/bin/python     1957MiB |
+-------------------------------------------------------------------------------+
[MDP-TITAN-GPU066 ~]$ sudo kill -9 24960
```

2번 노드에서 전체 process stop
1번 노드의 학습도 중단

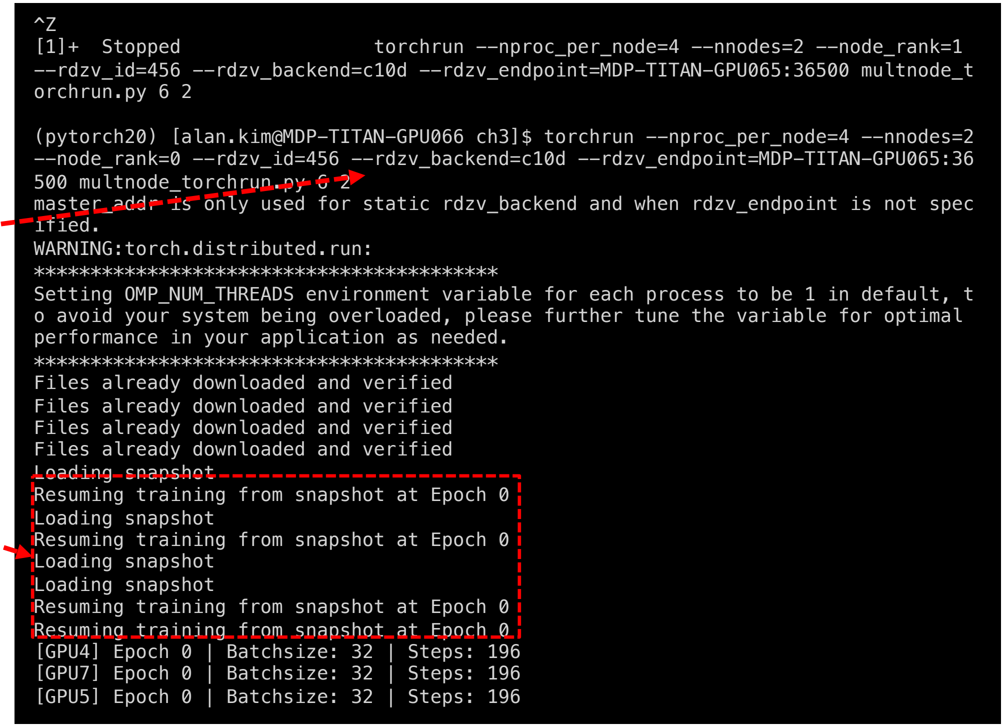
Master Node



```
[alan.kim@MDP-TITAN-GPU065 ch3]$ torchrun --nproc_per_node=4 --nnodes=2 --node_rank
=0 --rdzv_id=456 --rdzv_backend=c10d --rdzv_endpoint=MDP-TITAN-GPU065:36500 multnod
e_torchrun.py 6 2


Epoch 0 | Training snapshot saved at checkpoint/snapshot.pt
[GPU0] Epoch 1 | Batchsize: 32 | Steps: 196
[GPU1] Epoch 2 | Batchsize: 32 | Steps: 196
[GPU3] Epoch 2 | Batchsize: 32 | Steps: 196
[GPU0] Epoch 2 | Batchsize: 32 | Steps: 196
[GPU2] Epoch 2 | Batchsize: 32 | Steps: 196
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 51725 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 51726 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 51727 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 51728 closing
signal SIGTERM
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Loading snapshot
Loading snapshot
Resuming training from snapshot at Epoch 0
Resuming training from snapshot at Epoch 0
Loading snapshot
Resuming training from snapshot at Epoch 0
Loading snapshot
Resuming training from snapshot at Epoch 0
[GPU0] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU1] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU3] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU2] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU2] Epoch 1 | Batchsize: 32 | Steps: 196
[GPU3] Epoch 1 | Batchsize: 32 | Steps: 196
[GPU1] Epoch 1 | Batchsize: 32 | Steps: 196
```



```
[GPU5] Epoch 2 | Batchsize: 32 | Steps: 196
[GPU6] Epoch 2 | Batchsize: 32 | Steps: 196
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 24958 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 24959 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 24961 closing
signal SIGTERM
ERROR:torch.distributed.elastic.multiprocessing.api:failed (exitcode: -9) local_ran
k: 2 (pid: 24960) of binary: /t1data/users/alan.kim/.conda/envs/pytorch20/bin/pytho
n
```



```
^Z
[1]+  Stopped                 torchrun --nproc_per_node=4 --nnodes=2 --node_rank=1
--rdzv_id=456 --rdzv_backend=c10d --rdzv_endpoint=MDP-TITAN-GPU065:36500 multnode_t
orchrun.py 6 2

(pytorch20) [alan.kim@MDP-TITAN-GPU066 ch3]$ torchrun --nproc_per_node=4 --nnodes=2
--node_rank=0 --rdzv_id=456 --rdzv_backend=c10d --rdzv_endpoint=MDP-TITAN-GPU065:36
500 multnode_torchrun.py 6 2
master_addr is only used for static rdzv_backend and when rdzv_endpoint is not spec
ified.
WARNING:torch.distributed.run:
*****************************************
Setting OMP_NUM_THREADS environment variable for each process to be 1 in default, t
o avoid your system being overloaded, please further tune the variable for optimal
performance in your application as needed.
*****************************************
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Loading snapshot
Resuming training from snapshot at Epoch 0
Loading snapshot
Resuming training from snapshot at Epoch 0
Loading snapshot
Loading snapshot
Resuming training from snapshot at Epoch 0
Resuming training from snapshot at Epoch 0
[GPU4] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU7] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU5] Epoch 0 | Batchsize: 32 | Steps: 196
```

2번 노드 학습 재개

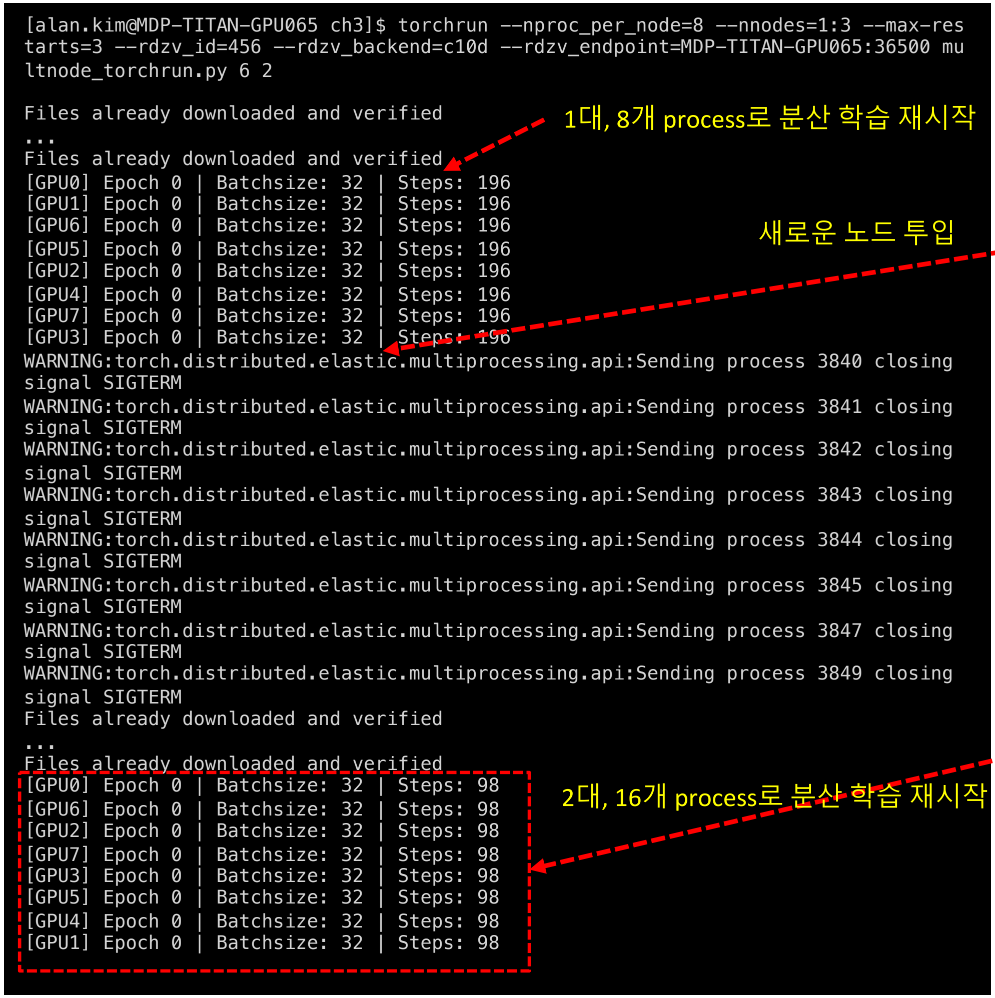
Snapshot으로 저장된 epoch부터
분산 학습 재시작

Distributed Data Parallel

2. torchrun

- Elastic 구현

Master Node



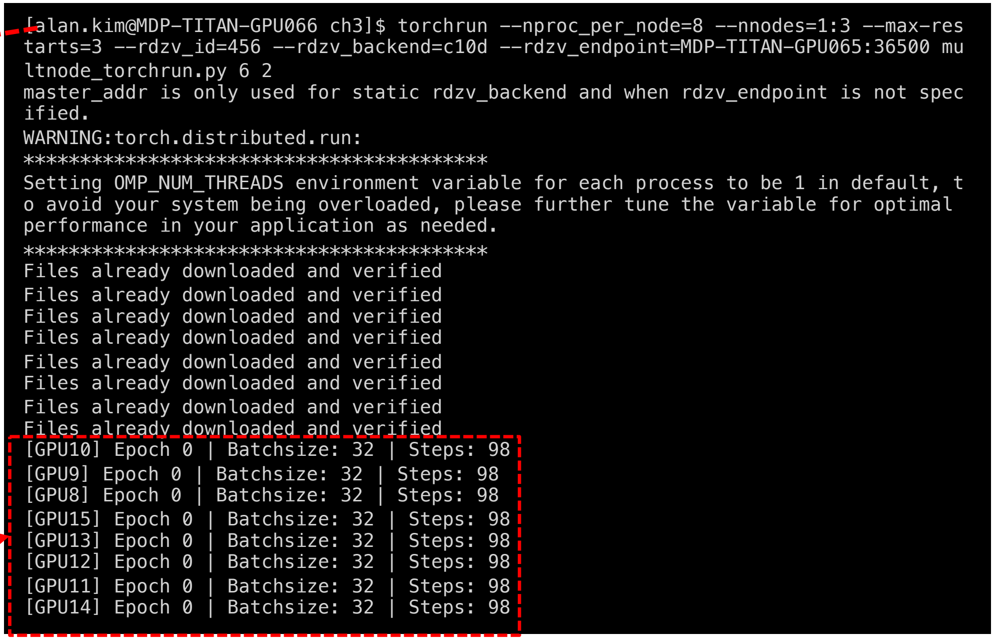
```
[alan.kim@MDP-TITAN-GPU065 ch3]$ torchrun --nproc_per_node=8 --nnodes=1:3 --max-res
tarts=3 --rdzv_id=456 --rdzv_backend=c10d --rdzv_endpoint=MDP-TITAN-GPU065:36500 mu
ltnode_torchrun.py 6 2

Files already downloaded and verified
...
Files already downloaded and verified
[GPU0] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU1] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU6] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU5] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU2] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU4] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU7] Epoch 0 | Batchsize: 32 | Steps: 196
[GPU3] Epoch 0 | Batchsize: 32 | Steps: 196
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3840 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3841 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3842 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3843 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3844 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3845 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3847 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 3849 closing
signal SIGTERM
Files already downloaded and verified
...
Files already downloaded and verified
[GPU0] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU6] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU2] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU7] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU3] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU5] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU4] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU1] Epoch 0 | Batchsize: 32 | Steps: 98
```

1대, 8개 process로 분산 학습 재시작

새로운 노드 투입

2대, 16개 process로 분산 학습 재시작

Worker Node 1



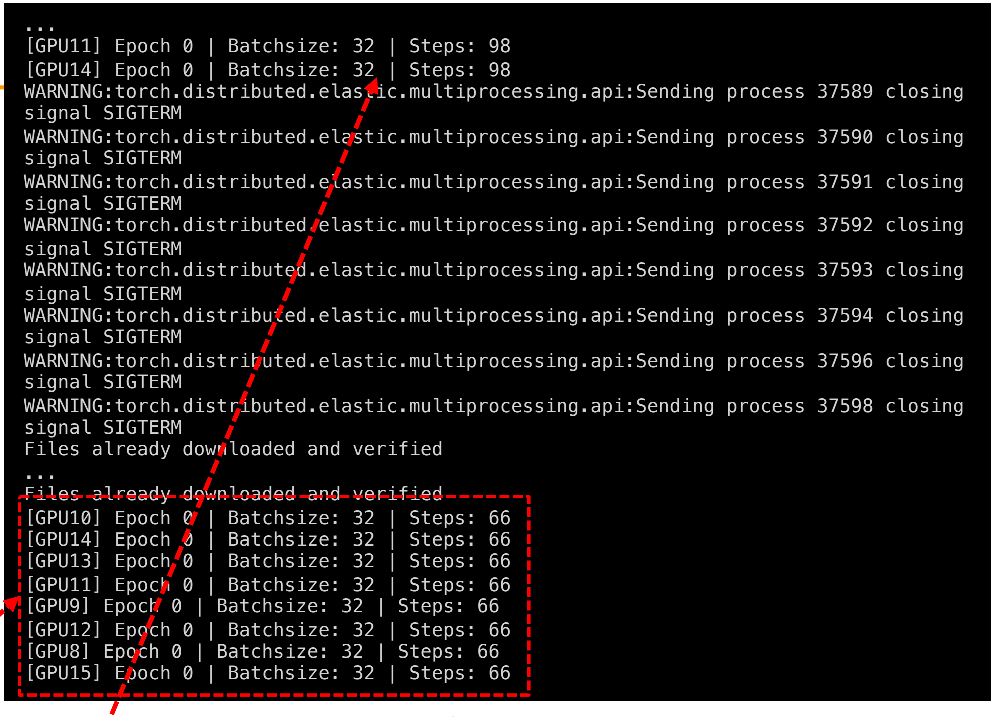
```
[alan.kim@MDP-TITAN-GPU066 ch3]$ torchrun --nproc_per_node=8 --nnodes=1:3 --max-res
tarts=3 --rdzv_id=456 --rdzv_backend=c10d --rdzv_endpoint=MDP-TITAN-GPU065:36500 mu
ltnode_torchrun.py 6 2
master_addr is only used for static rdzv_backend and when rdzv_endpoint is not spec
ified.
WARNING:torch.distributed.run:
*****************************************
Setting OMP_NUM_THREADS environment variable for each process to be 1 in default, t
o avoid your system being overloaded, please further tune the variable for optimal
performance in your application as needed.
*****************************************
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
[GPU10] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU9] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU8] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU15] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU13] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU12] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU11] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU14] Epoch 0 | Batchsize: 32 | Steps: 98
```

Distributed Data Parallel

2. torchrun

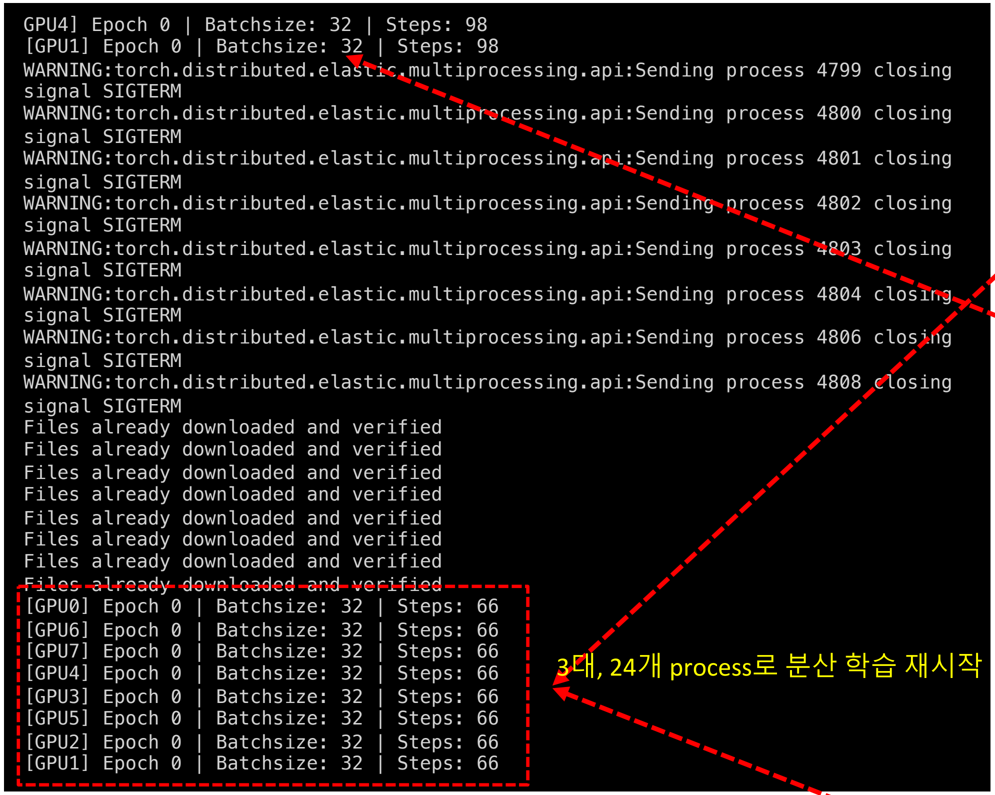
○ Elastic 구현

Worker Node 1



```
...
[GPU11] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU14] Epoch 0 | Batchsize: 32 | Steps: 98
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37589 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37590 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37591 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37592 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37593 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37594 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37596 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 37598 closing
signal SIGTERM
Files already downloaded and verified
...
Files already downloaded and verified
[GPU10] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU14] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU13] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU11] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU9] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU12] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU8] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU15] Epoch 0 | Batchsize: 32 | Steps: 66
```

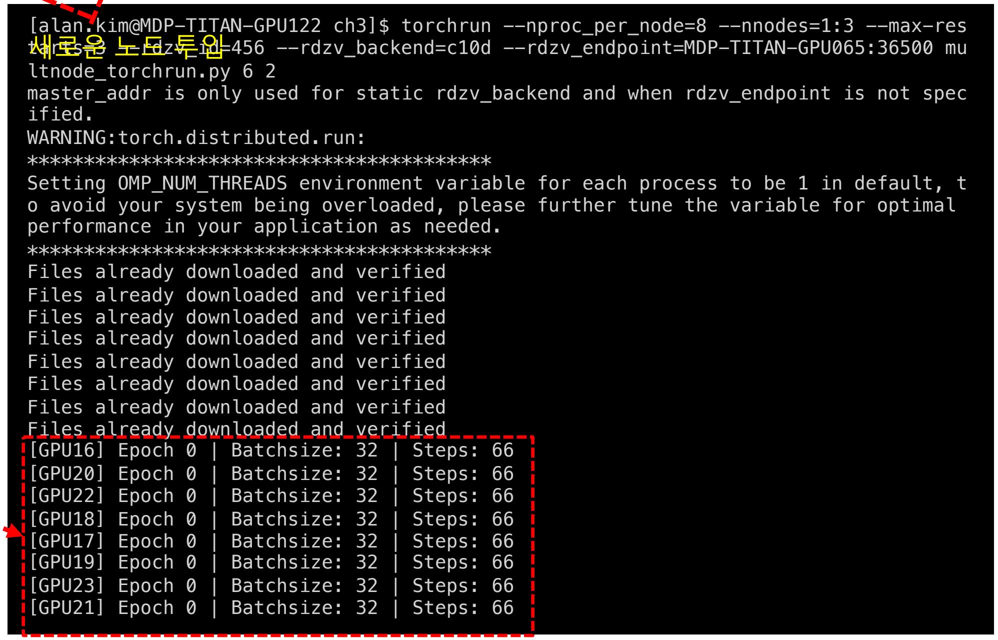
Master Node



```
GPU4] Epoch 0 | Batchsize: 32 | Steps: 98
[GPU1] Epoch 0 | Batchsize: 32 | Steps: 98
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4799 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4800 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4801 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4802 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4803 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4804 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4806 closing
signal SIGTERM
WARNING:torch.distributed.elastic.multiprocessing.api:Sending process 4808 closing
signal SIGTERM
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
[GPU0] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU6] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU7] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU4] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU3] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU5] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU2] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU1] Epoch 0 | Batchsize: 32 | Steps: 66
```

3대, 24개 process로 분산 학습 재시작

Worker Node 2



새로운 노드 투입

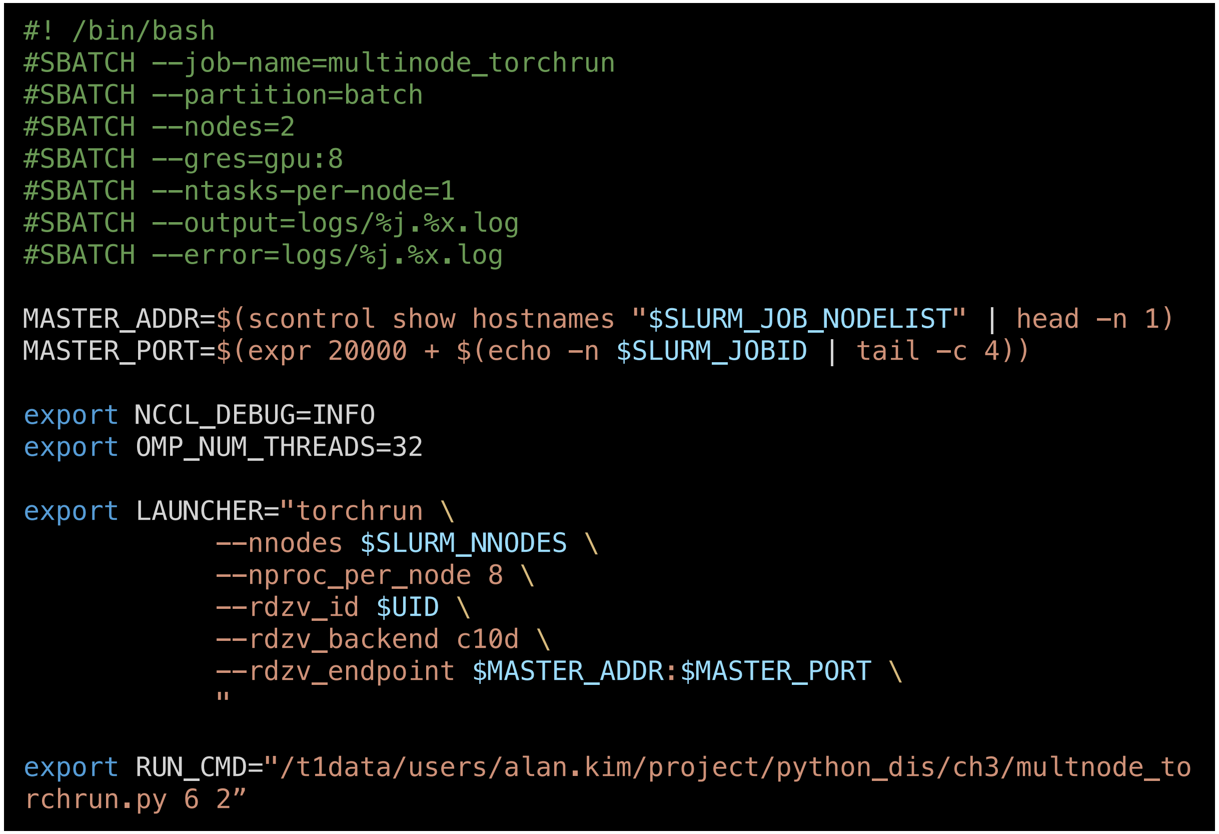
```
[alan.kim@MDP-TITAN-GPU122 ch3]$ torchrun --nproc_per_node=8 --nnodes=1:3 --max-res
tarts=3 --rdzv_id=456 --rdzv_backend=c10d --rdzv_endpoint=MDP-TITAN-GPU065:36500 mu
ltnode_torchrun.py 6 2
master_addr is only used for static rdzv_backend and when rdzv_endpoint is not spec
ified.
WARNING:torch.distributed.run:
*****************************************
Setting OMP_NUM_THREADS environment variable for each process to be 1 in default, t
o avoid your system being overloaded, please further tune the variable for optimal
performance in your application as needed.
*****************************************
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
[GPU16] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU20] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU22] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU18] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU17] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU19] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU23] Epoch 0 | Batchsize: 32 | Steps: 66
[GPU21] Epoch 0 | Batchsize: 32 | Steps: 66
```

Distributed Data Parallel

2. torchrun

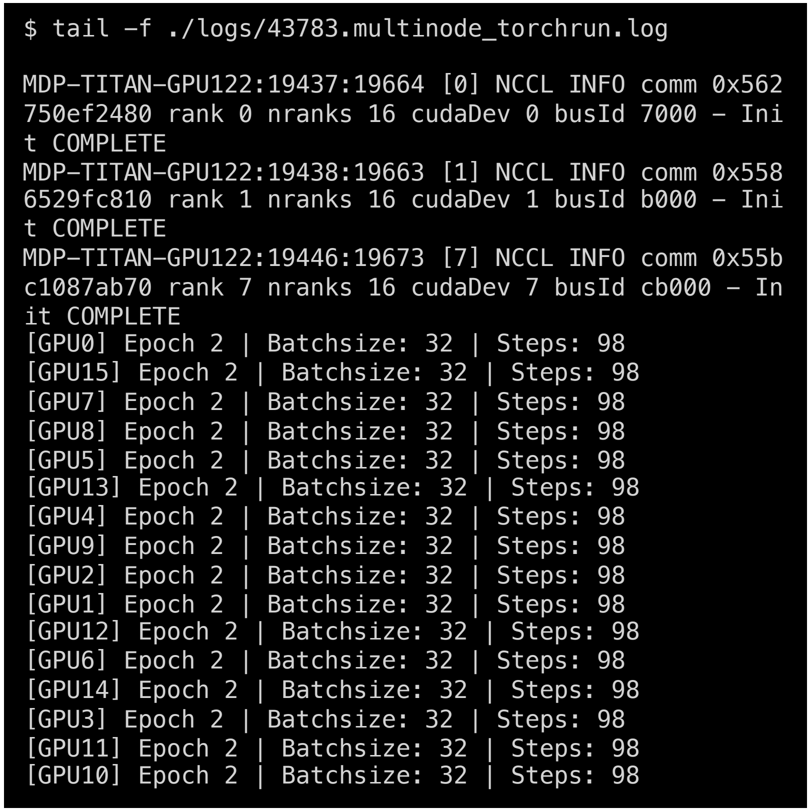
- Slurm batchjob

sbatch script



```
#! /bin/bash
#SBATCH --job-name=multinode_torchrun
#SBATCH --partition=batch
#SBATCH --nodes=2
#SBATCH --gres=gpu:8
#SBATCH --ntasks-per-node=1
#SBATCH --output=logs/%j.%x.log
#SBATCH --error=logs/%j.%x.log

MASTER_ADDR=$(scontrol show hostnames "$SLURM_JOB_NODELIST" | head -n 1)
MASTER_PORT=$(expr 20000 + $(echo -n $SLURM_JOBID | tail -c 4))

export NCCL_DEBUG=INFO
export OMP_NUM_THREADS=32

export LAUNCHER="torchrun \
            --nnodes $SLURM_NNODES \
            --nproc_per_node 8 \
            --rdzv_id $UID \
            --rdzv_backend c10d \
            --rdzv_endpoint $MASTER_ADDR:$MASTER_PORT \
            "

export RUN_CMD="/t1data/users/alan.kim/project/python_dis/ch3/multnode_to
rchrun.py 6 2”
```

logs



```
$ tail -f ./logs/43783.multinode_torchrun.log

MDP-TITAN-GPU122:19437:19664 [0] NCCL INFO comm 0x562
750ef2480 rank 0 nranks 16 cudaDev 0 busId 7000 - Ini
t COMPLETE
MDP-TITAN-GPU122:19438:19663 [1] NCCL INFO comm 0x558
6529fc810 rank 1 nranks 16 cudaDev 1 busId b000 - Ini
t COMPLETE
MDP-TITAN-GPU122:19446:19673 [7] NCCL INFO comm 0x55b
c1087ab70 rank 7 nranks 16 cudaDev 7 busId cb000 - In
it COMPLETE
[GPU0] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU15] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU7] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU8] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU5] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU13] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU4] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU9] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU2] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU1] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU12] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU6] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU14] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU3] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU11] Epoch 2 | Batchsize: 32 | Steps: 98
[GPU10] Epoch 2 | Batchsize: 32 | Steps: 98
```

End of Document