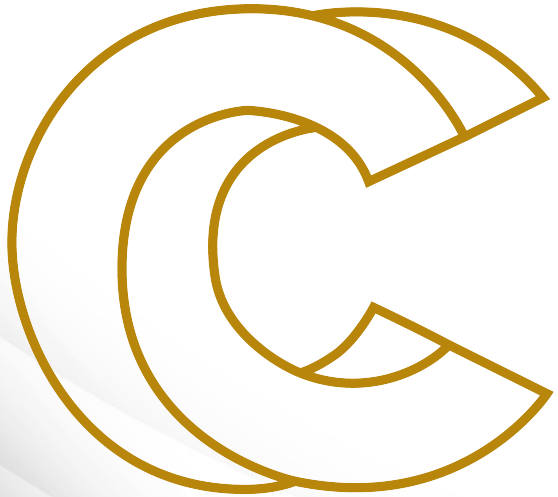


EURO 4SEE

Dr. İsmail Güzel

TÜRKİYE

Lightning

- From PyTorch to Lightning
- ddp fsdp deepseed types
- Hands on!

OpenMPI and DDP

```python
from datetime import datetime
import argparse
import os
import torch
import torch.nn as nn
import torch.distributed as dist
import torchvision.transforms as transforms
from torchvision.datasets import MNIST
from torch.utils.data.distributed import DistributedSampler
from torch.nn.parallel import DistributedDataParallel
from torch.utils.data import DataLoader


class ConvNet(nn.Module):…


def train(num_epochs, batch_size):…

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--epochs', default=2, type=int, metavar='N',
                        help='number of total epochs to run')
    parser.add_argument('--batch_size', default=32, type=int, metavar='N',
                        help='number of batch size to run')
    args = parser.parse_args()

    train(args.epochs, args.batch_size)


if __name__ == '__main__':
    main()
```

```python
torch.manual_seed(0)
local_rank = int(os.environ['OMPI_COMM_WORLD_LOCAL_RANK'])
world_size = int(os.environ['OMPI_COMM_WORLD_SIZE'])
rank = int(os.environ['OMPI_COMM_WORLD_RANK'])

# For multi-node jobs, use the TCP init method instead.
master_addr = os.environ["MASTER_ADDR"]
master_port = os.environ["MASTER_PORT"]
dist.init_process_group(backend='nccl',
                        init_method=f"tcp://{master_addr}:{master_port}",
                        rank=rank, world_size=world_size)

torch.cuda.set_device(local_rank)
verbose = dist.get_rank() == 0  # print only on global_rank==0
model = ConvNet().cuda()

criterion = nn.CrossEntropyLoss().cuda()
optimizer = torch.optim.SGD(model.parameters(), 1e-4)
```

```python
model = DistributedDataParallel(model, device_ids=[local_rank])

train_dataset = MNIST(root='./data', train=True,
                      transform=transforms.ToTensor(), download=True)
train_sampler = DistributedSampler(train_dataset)
train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size,
                          shuffle=False, num_workers=0, pin_memory=True,
                          sampler=train_sampler)

start = datetime.now()
for epoch in range(num_epochs):…
if verbose:
    print("Training completed in: " + str(datetime.now() - start))
```

8 tasks:
1 server
8 GPU
8*16 CPU

```
#!/bin/bash
#SBATCH --account=<kullanici-adiniz>
#SBATCH --partition=palamut-cuda
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=16
#SBATCH --time=30:00
#SBATCH --gres=gpu:8

#SBATCH --output=./out/%j.out
#SBATCH --error=./out/%j.err

export MASTER_ADDR=$(scontrol show hostname ${SLURM_NODELIST} | head -n 1)
export MASTER_PORT=$(expr 10000 + $(echo -n $SLURM_JOBID | tail -c 4))


module purge
eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash hook)"
conda activate pytorch_cuda_12

nvidia-smi --query-
gpu=timestamp,name,pci.bus_id,power.draw,temperature.gpu,utilization.gpu,ut
ilization.memory --format=csv -l >> ./out/${SLURM_JOBID}.nout  &

module load centos7.9/comp/gcc/12.2 centos7.9/lib/openmpi/5.0.0-gcc-12.2
mpirun -np 8 python mnist_ddp_mpirun.py --epochs=100 --batch_size=256
```

16 tasks:
2 server
16 GPU
2*8*16 CPU

```
#!/bin/bash
#SBATCH --account=iguzel
#SBATCH --partition=palamut-cuda
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=16
#SBATCH --time=30:00
#SBATCH --gres=gpu:8

#SBATCH --output=./out/%j.out
#SBATCH --error=./out/%j.err

module purge
eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash hook)"
conda activate pytorch_cuda_12

export MASTER_ADDR=$(scontrol show hostname ${SLURM_NODELIST} | head -n 1)
export MASTER_PORT=$(expr 10000 + $(echo -n $SLURM_JOBID | tail -c 4))


module load centos7.9/comp/gcc/12.2 centos7.9/lib/openmpi/5.0.0-gcc-12.2
mpirun -np 16 python mnist_ddp_mpirun.py --epochs=100 --batch_size=256

exit
```

```python
import argparse
import os
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torch.nn.functional as F
from torchvision.datasets import MNIST
from torch.utils.data import DataLoader
#import lightning as L
import pytorch_lightning as L
import mlflow
```

```python
class LitConvNet(L.LightningModule):
    def __init__(self, num_classes=10):
        super().__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=5, stride=1, padding=2),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=5, stride=1, padding=2),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.fc = nn.Linear(7*7*32, num_classes)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.reshape(out.size(0), -1)
        out = self.fc(out)
        return out

    def training_step(self, batch, batch_idx):
        images, labels = batch
        outputs = self(images)
        loss = F.cross_entropy(outputs, labels)
        self.log("train_loss", loss)
        return loss

    def configure_optimizers(self):
        optimizer = torch.optim.SGD(self.parameters(), 1e-4)
        return optimizer
```

```python
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--gpus', default=1, type=int, metavar='N',
                        help='number of GPUs per node')
    parser.add_argument('--nodes', default=1, type=int, metavar='N',
                        help='number of nodes')
    parser.add_argument('--epochs', default=2, type=int, metavar='N',
                        help='maximum number of epochs to run')
    args = parser.parse_args()
    mlflow.autolog()
    batch_size = 100
    dataset = MNIST(os.getcwd(), download=True,
                    transform=transforms.ToTensor())
    train_loader = DataLoader(dataset, batch_size=batch_size,
                              num_workers=10, pin_memory=True)
    convnet = LitConvNet()

    trainer = L.Trainer(devices=args.gpus,
                        num_nodes=args.nodes,
                        max_epochs=args.epochs,
                        accelerator='gpu',
                        strategy='ddp')

    from datetime import datetime
    t0 = datetime.now()
    trainer.fit(convnet, train_loader)
    dt = datetime.now() - t0
    print('Training took {}'.format(dt))

    trainer.save_checkpoint("./out/lightning_model.ckpt")

if __name__ == '__main__':
    main()
```

```
#!/bin/bash
#SBATCH --account=<kullanici-adiniz>
#SBATCH --partition=palamut-cuda
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=16
#SBATCH --time=30:00
#SBATCH --gres=gpu:8

#SBATCH --output=./out/%j.out
#SBATCH --error=./out/%j.err

module purge
eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash hook)"
conda activate lightning

python3 mnist_lightning_ddp.py --gpus=8 --epochs=100

exit
```

8 tasks:
1 server
8 GPU
8*16 CPU

sbatch job.slurm

job.slurm

```
#!/bin/bash
#SBATCH --account=<kullanici-adiniz>
#SBATCH --partition=palamut-cuda
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=16
#SBATCH --time=30:00
#SBATCH --gres=gpu:8

#SBATCH --output=./out/%j.out
#SBATCH --error=./out/%j.err

module purge
eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash hook)"
conda activate lightning

python3 mnist_lightning_ddp.py --gpus=8 --nodes=2 --epochs=300

exit
```

16 tasks:
2 server
16 GPU
2*8*16 CPU

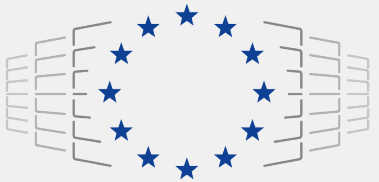
Hands On !

https://github.com/ismailguzel/lightning_ncc_Turkiye

EURO 4SEE

Thanks!

Co-funded by
the European Union



EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101191697. The JU receives support from the Digital Europe Programme and Germany, Türkiye, Republic of North Macedonia, Montenegro, Serbia, Bosnia and Herzegovina.