Intel® Cloud Optimization Modules for Microsoft Azure*: Stable Diffusion Distributed Training

This module illustrates the process of fine-tuning a stable diffusion model on 3rd or 4th Generation Intel® Xeon® Scalable Processors on Microsoft Azure in a distributed architecture. We will employ the Stable Diffusion v1 implementation and use the Dicoo images from Hugging Face* Hub.

To capitalize on the capabilities of Intel hardware, we also integrate Intel AI software offerings, like the Intel® Extension for PyTorch* and the Intel® oneAPI Collective Communications Library (oneCCL).

Azure Cluster Setup

For distributed fine-tuning, our example deploys a cluster consisting of three 3rd Gen Intel® Xeon® CPUs from the Azure Dv5 series with an Ubuntu 22.04 image and 32 GiB of storage deployed on an Azure Trusted Virtual Machine. For maximum performance, we recommend using the precision data type `bfloat16` on 4th Gen Intel® Xeon® CPUs on Azure with the deep learning acceleration engine, Intel® Advanced Matrix Extensions (Intel® AMX).

If you are using a 4th Gen Xeon CPU, you can verify AMX support by running:

```
lscpu | grep amx
```

You should see the following flags:

```
amx_bf16 amx_tile amx_int8
```

Intel® Extension for PyTorch*

Intel upstreams as many optimizations as possible to PyTorch. These features, however, often debut in the Intel® Extension for PyTorch. Install PyTorch (guide), and then the Intel Extension for PyTorch:

```
pip install intel_extension_for_pytorch==2.0.0
```

To enable the optimizations, only add these **two lines** to your Python* code:

```
import intel_extension_for_pytorch as ipex

model.train()
model, optimizer = ipex.optimize(model, optimizer=optimizer,
                                 dtype="torch.bfloat16", level="O1", inplace=True)
```

Documentation | Cheat Sheet | Examples | Tuning Guide

Install oneCCL

Download the appropriate wheel file and install it using the following commands:

```
wget https://intel-extension-for-
pytorch.s3.amazonaws.com/torch_ccl/cpu/oneccl_bind_pt-2.0.0%2Bcpu-cp38-cp38-
linux_x86_64.whl
pip install oneccl_bind_pt-2.0.0+cpu-cp38-cp38-linux_x86_64.whl
```

To use `oneccl_bindings_for_pytorch`, source the environment by running the following command:

```
oneccl_path=$(python -c "from oneccl_bindings_for_pytorch import cwd; print(cwd)")
source $oneccl_path/env/setvars.sh
```

To launch distributed fine-tuning:

```
mpirun -f ~/hosts -n 3 -ppn 1 accelerate launch textual_inversion.py --
pretrained_model_name_or_path="runwayml/stable-diffusion-v1-5" --train_data_dir="dicoo" …
```

The `mpirun` command runs the fine-tuning process across 3 machines (`-n 3`) with one process per machine (`-ppn 1`). Additionally, environment variables can be set using `-genv` argument.

Next Steps:

All Cloud Modules | GitHub Repo | DevMesh Discord