Intel® Cloud Optimization Modules for AWS*:
GPT2-Small Distributed Training

This module illustrates the process of fine-tuning a Large Language model (LLM) on a 3rd or 4th Generation Intel® Xeon® Scalable Processors on AWS in a distributed fashion. We will employ the nanoGPT implementation of GPT2-Small model and use the OpenWebText dataset from Hugging Face Hub.

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

AWS Cluster Setup

For distributed training, our example deploys a cluster consisting of three 3rd Gen. Xeon CPUs (Amazon Elastic Compute Cloud* (EC2) m6i.4xlarge instances) with an Ubuntu 22.04 AMI and 250 GB of storage. For maximum performance, we recommend using **bfloat16** precision on the 4th Gen. Xeon® CPUs (R7iz) on AWS with the deep learning acceleration engine called Intel® Advanced Matrix Extensions (AMX).

If you are using a 4th Gen. Xeon CPU, you can verify AMX is present 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==1.13.0 --extra-index-url
https://developer.intel.com/ipex-whl-stable-cpu
```

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-1.13.0%2Bcpu-cp310-cp310-
linux_x86_64.whl
pip install oneccl_bind_pt-1.13.0+cpu-cp310-cp310-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 -genv LD_PRELOAD="/usr/lib/x86_64-linux-
gnu/libtcmalloc.so" accelerate launch --config_file ./multi_config.yaml main.py
```

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

Links:
All Cloud Modules | GitHub Repo | DevMesh Discord