Solution Brief

Cross-Industry Architecture
Intel® Cloud Optimization Modules

intel software

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

Authors: Ankur Singh, Benjamin Consolvo

Date: August 9, 2023

Fine-tune GPT2-Small on the OpenWebText dataset in a distributed fashion on AWS on 3rd or 4th Generation Intel® Xeon® Scalable Processors.

The **Intel Cloud Optimization Modules for AWS: GPT2-Small Distributed Training** is designed to illustrate the process of fine-tuning a LLM on 3rd or 4th Gen. Xeon CPUs on AWS. Specifically, we show the process of training the GPT2-Small (124M parameter) model on the OpenWebText dataset in a distributed fashion. The project builds upon the initial codebase of nanoGPT, by Andrej Karpathy. The objective here is not necessarily to arrive at a chatGPT-like AI model, but rather to understand how to set up distributed training so that you can fine-tune to your specific objective. The result of training here will result in a base LLM that can generate words (or tokens), but it will be suitable for your use-case when you train it on your specific task and dataset.

Use it as a reference solution for:

- Setting up an AWS cluster for distributed training.
- Fine-tuning a LLM on a single machine.
- Fine-tuning LLM in a distributed setup, taking advantage of Intel optimizations.

Who needs it?

- Developers aiming to fine-tune their LLMs on multiple Intel Xeon CPUs, leveraging Intel's accelerated deep learning software libraries, including Intel Extension for PyTorch and oneCCL.
- Developers interested in learning the process of setting up AWS Clusters for distributed training.

What it does

This module demonstrates how to transform a standard single-node PyTorch training scenario into a high-performance, distributed training scenario on multiple CPUs. To fully capitalize on Intel hardware and further optimize the fine-tuning process, this module integrates the Intel® Extension for PyTorch*, and Intel® oneAPI Collective Communications Library (oneCCL). The module serves as a guide to setting up an AWS cluster for distributed training while showcasing a complete project for fine-tuning LLMs.

- It provides step-by-step instructions for configuring an AWS Cluster, simplifying the process of establishing a distributed training environment.
- A guide through the entire lifecycle of fine-tuning LLMs, starting from data preprocessing to model fine-tuning.
- The module capitalizes on Intel® Extension for PyTorch, harnessing the power of Intel Advanced Vector Extensions 512 (AVX-512) and Intel Advanced Matrix Extension (AMX) instruction sets. This enables significant acceleration of the fine-tuning process, boosting overall training performance. The use of Intel's optimized communications library, oneCCL, ensures that distributed workflows are streamlined, enhancing efficiency in a multi-node training setup.

In summary, this module empowers users to harness the full potential of Intel hardware for distributed training an LLM.

Cloud Solution Architecture

This cloud solution utilizes a basic AWS cluster setup, with the central components being the EC2 instances forming the cluster. To enable seamless communication between these instances, we connected each of them to the same VPC network and established a permissive security group that allows all traffic from other nodes within the cluster. The raw dataset is taken from Hugging Face* Hub, and after the model has been trained, the weights are saved to the EC2 instances and optionally to an S3 bucket.

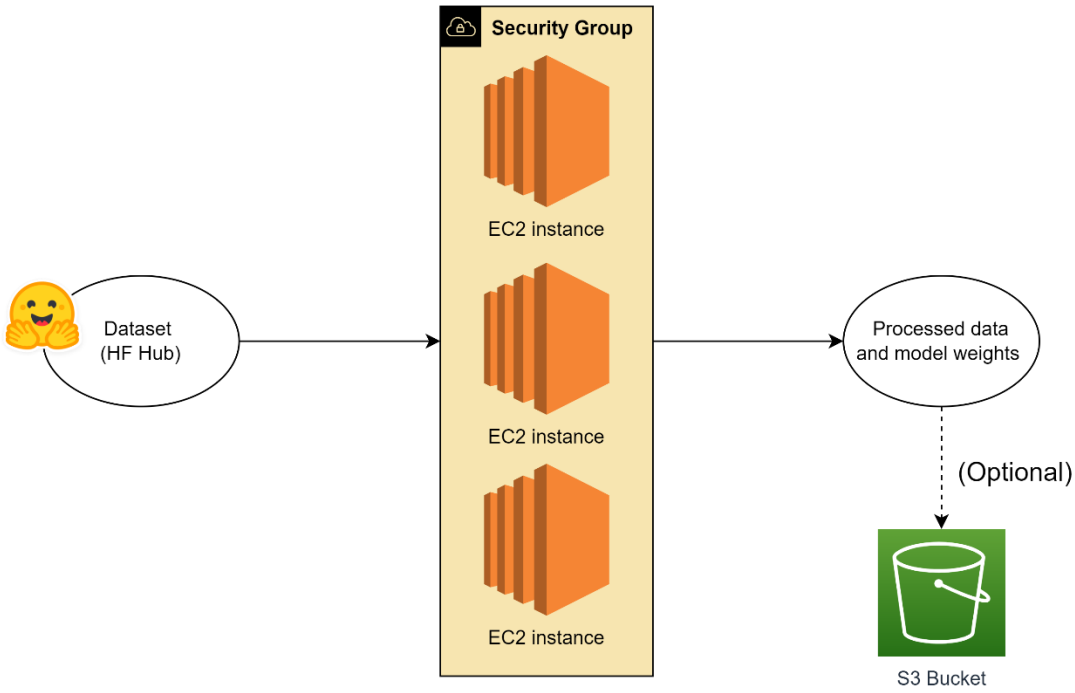

Figure 1: AWS Architecture Diagram for Distributed Training on Multiple EC2 instances. Image from author.

Solution Component Overview

This solution is derived from the nanoGPT implementation by Andrej Karpathy. The code has been enhanced through refactoring to achieve better modularity and suitability for distributed fine-tuning on 3rd or 4th Gen Xeon CPUs. For data, we used the OpenWebText dataset obtained from Hugging Face. To fully leverage Intel hardware capabilities and enable distributed training, we incorporated the Intel Extension for PyTorch and oneCCL.

Code Highlights

Enable the Intel Extension for PyTorch

The Intel Extension for PyTorch elevates PyTorch performance on Intel hardware with the integration of cutting-edge features and optimizations. This extension efficiently utilizes Intel hardware capabilities, such as Intel AVX-512 and Intel AMX on Intel CPUs. Unleashing this power is straightforward – just wrap your model and optimizer objects with **`ipex.optimize`**.

```
# Setup up CPU autocast and bfloat16 dtype
dtype = torch.bfloat16
self.autocast_ctx_manager = torch.cpu.amp.autocast(
        cache_enabled=True, dtype=dtype
)

# Wrap both Pytorch model and Optimizer
self.model, self.optimizer = ipex.optimize(
        self.model, optimizer=self.optimizer,
        dtype=dtype, inplace=True, level="O1",
)
```

Gradient Accumulation with Hugging Face Accelerate

To streamline the gradient accumulation process, the Accelerate library by Hugging Face is implemented. This package helps to abstract away the complexity of supporting multi-CPUs/GPUs and provides an intuitive interface, user-friendly API, making gradient accumulation and clipping hassle-free during the training process.

```
# Initializing Accelerator object
self.accelerator = Accelerator(
        gradient_accumulation_steps=gradient_accumulation_steps,
        cpu=True,
)

# Gradient Accumulation
with self.accelerator.accumulate(self.model):
       with self.autocast_ctx_manager:
             _, loss = self.model(X, Y)
       self.accelerator.backward(loss)
       loss = loss.detach() / gradient_accumulation_steps

# Gradient Clipping
self.accelerator.clip_grad_norm_(
       self.model.parameters(), self.trainer_config.grad_clip
)
```

Distributed Training

For distributed training, we utilized oneCCL. With optimized communication patterns, oneCCL enables developers and researchers to train newer and deeper models more quickly across multiple nodes. It offers a tool called **`mpirun`**, which allows one to seamlessly launch distributed training workloads.

```
# Generating Multi-CPU config
accelerate config --config_file ./multi_config.yaml

# Launching Distributed Training job
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
```

Next Steps

Download the module from GitHub ›

Register for office hours to get help on your implementation ›

Check out the full suite of Intel Cloud Optimization Modules ›

Come chat with us on our DevHub Discord server to keep interacting with other developers ›

Intel® technologies may require enabled hardware, software, or service activation. Learn more at intel.com or from the OEM or retailer. Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Optimization notice: Intel® compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel® microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel® microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product user and reference guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804. https://software.intel.com/en-us/articles/optimization-notice

Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. See backup for configuration details. For more complete information about performance and benchmark results, visit intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and noninfringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.

*Other names and brands may be claimed as the property of others.

1121/SS/CMD/PDF