OLCF Training

AI on Frontier

Junqi Yin
Analytics and AI Methods at Scale

Aug 24, 2023
Outline

- Frontier DL Environment
- Distributed Training
- Examples
  - PyTorch
  - TensorFlow
- Training Large Language Models (LLMs)
Most DL codes work on Frontier as is
Distributed Training Strategies

• Data Parallel
  – PyTorch DDP, TensorFlow Multiworker Mirrored strategy
  – Horovod

• Model Parallel
  – PyTorch FSDP
  – DeepSpeed-Megatron 3D parallelism
Distributed Training Strategies: FSDP

• FSDP – fully sharded data parallel, similar to ZeRO

1.5X communication size of DDP
Distributed Training Strategies: DeepSpeed+Megatron

- **DeepSpeed + Megatron**
  - Sharded data parallel – ZeRO, shared optimizer states, gradients, and parameters
  - Pipeline parallel – layer placement + micro-batch (Gpipe)
  - Tensor parallel – sharded tensor
Performance baselines: data-parallel

- **ResNet50**
  - Mixed
  - ~1.0x per GCD
  - 98% at 1024

![Graph showing ResNet50 performance on ImageNet with Summit, Frontier, and Ideal lines.](image-url)
Performance baselines: data-parallel

- **STEMDL**
  - Tiramisu network
  - 220M parameters
  - 97% at 8192 GPUs

Accelerating Collective Communication in Data Parallel Training across Deep Learning Frameworks. USENIX NSDI'22, 2022
Distributed Training Examples: STEMDL Benchmark

- One of MLCommon Science benchmarks
- Code on Frontier
  - [https://code.ornl.gov/jqyin/stemdl-benchmark/-/tree/frontier](https://code.ornl.gov/jqyin/stemdl-benchmark/-/tree/frontier)
  - Classification – PyTorch
  - Reconstruction – TensorFlow
  - Built/job scripts
  - Conda environment

---

Electron Microscope

- E-Beam
- 2D Scan
- Material
- Diffracted Beam

CBED Scan

Material Properties
Distributed Training Examples: STEMDL Benchmark

Software Environment

A pre-built environment is provided at `~/lustre/orion/world-shared/stf218/junqi/miniconda`.

```
module load rccm/5.1.0
BUILD=/lustre/orion/world-shared/stf218/junqi/bin:SPATH
source $BUILD/etc/profile.d/conda.sh
```

To activate PyTorch env:
```
conda activate $BUILD/envs/pyt_env
```

To activate TensorFlow env:
```
conda activate $BUILD/envs/tf_env
```

To build from source, please follow the scripts in `utils`

Output

The classification task outputs the miperf and tensorboard logs to `logs` folder (default)
```
#stemdl_classification.log
:::MLLOG {"namespace": "", "time_ms": 1692186386826, "event_type": "POINT_IN_TIME", "key": "eval_accuracy", "value": 0.8189140737030217}
:::MLLOG {"namespace": "", "time_ms": 1692186386826, "event_type": "POINT_IN_TIME", "key": "f1_score", "value": 2.25048123106490135}
:::MLLOG {"namespace": "", "time_ms": 1692186386826, "event_type": "INTERVAL_END", "key": "eval_stop", "value": null, "metadata": []}
:::MLLOG {"namespace": "", "time_ms": 1692186387223, "event_type": "INTERVAL_START", "key": "epoch_stop", "value": null, "metadata": []}
```

The reconstruction task outputs the training performance in FLOPS and loss values to `output_logs` folder (default)
```
58.0, step= 18, epoch= 9.55e-03, loss= 0.52, lr= 1.38e-06, step_time= 15.93 sec, ranks= 8, examples/sec= 4.0, flops = 4.40e+13, ave
5.1, step= 20, epoch= 1.91e-02, loss= 0.46, lr= 1.00e-06, step_time= 1.71 sec, ranks= 8, examples/sec= 37.5, flops = 4.19e+14, ave
23.3, step= 36, epoch= 2.87e-02, loss= 0.53, lr= 1.00e-06, step_time= 1.71 sec, ranks= 8, examples/sec= 37.4, flops = 4.09e+14, ave
9.4, step= 40, epoch= 3.82e-02, loss= 0.52, lr= 1.00e-06, step_time= 1.71 sec, ranks= 8, examples/sec= 37.4, flops = 4.09e+14, ave
```

Data

Sample data (pre-processed) are located at `~/lustre/orion/world-shared/stf218/junqi/stemdl-data`
Training LLMs on Frontier

• Ideal case
  – Peak performance
  – Linear scaling
  – Chinchilla scaling

Evaluation of Pre-Training Large Language Models on Leadership Platforms, Journal of Supercomputing, 2023
Training LLMs on Frontier

- Actual measurement
  - 2 frameworks: FSDP, DeepSpeed-Megatron
  - 3D parallelisms: DP, TP, PP
Training LLMs on Frontier

- Scaling
  - DP is preferable
  - Large batch
  - bf16 more stable
Training LLMs on Frontier

- Computation and energy needed to pretrain LLMs

<table>
<thead>
<tr>
<th>Model</th>
<th>Summit</th>
<th>Frontier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (node hours)</td>
<td>Energy (MWh)</td>
</tr>
<tr>
<td>GPT-13B</td>
<td>23K</td>
<td>30</td>
</tr>
<tr>
<td>GPT-175B</td>
<td>[0.6M, 12M]</td>
<td>[309, 6176]</td>
</tr>
<tr>
<td>GPT-1T</td>
<td>[1B, 21B]</td>
<td>[3.7e5, 7.4e6]</td>
</tr>
</tbody>
</table>

[ # tokens = # params, # tokens = 20*# params ]

- Takeaway: pre-train O(13B) ~ O(DD), pre-train O(175B) ~ O(INCITE)

_Evaluation of Pre-Training Large Language Models on Leadership Platforms, Journal of Supercomputing, 2023_
## Simulation-ML Integration

<table>
<thead>
<tr>
<th>Method</th>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework C++ API (TensorFlow/PyTorch C++)</td>
<td>• Portable&lt;br&gt;• Better latency&lt;br&gt;• Easy to deploy</td>
<td>• Not flexible</td>
</tr>
<tr>
<td>Framework Server (TensorFlow Serving/TorchServe)</td>
<td>• Flexible&lt;br&gt;• Better throughput&lt;br&gt;• Options to deploy</td>
<td>• High maintenance</td>
</tr>
<tr>
<td>Third-party API (SmartRedis/RedisAI)</td>
<td>• Easy integration&lt;br&gt;• More functionality&lt;br&gt;• Model support</td>
<td>• Portability</td>
</tr>
</tbody>
</table>

Strategies for Integrating Deep Learning Surrogate Models with HPC Simulation Applications, IPDPSW 2022
Questions?

yinj@ornl.gov