MARS LAB, Dept. CDS, IISc

Links for project: ML system for training large scale DNNs on Hybrid CPU GPU Platforms
Funded By Shell India

Project Under Dr.Sathish Vadhiyar
Curated By: Dheemanth R Joshi

#————————————————————Literature Survey Ends Here ——————————————————————————

GPU parallelisation Strategies

I) Distributed Data Parallel (Meta)

1. Publication: https://arxiv.org/pdf/2006.15704
2. Documentation: https://pytorch.org/tutorials/intermediate/ddp_tutorial.html
3. GitHub Repo: https://github.com/pytorch/pytorch/blob/main/torch/nn/parallel/distributed.py
4. Description: The paper implements ring all reduce strategy to synchronise gradients across multiple workers. Gradient Bucketing strategy is introduced to send large chunk of tensors for observed improved efficiency.

II) Gpipe (Google Brain ++)

1. Publication: https://arxiv.org/pdf/1811.06965
2. Documentation: https://torchgpipe.readthedocs.io/en/stable/guide.html
3. GitHub Repo: https://github.com/kakaobrain/torchgpipe
4. Description: A improvised implementation of GPipe with activation checkpointing and pipelined micro batch training.

CPU+GPU executions

III) ZeRO-Offload (Microsoft->DeepSpeed)

1. Publication: https://arxiv.org/pdf/2101.06840
2. Documentation: https://www.microsoft.com/en-us/research/project/deepspeed/
3. GitHub Repo: https://github.com/microsoft/DeepSpeed
4. Description: Storing Entire model on CPU and periodically transfer it to GPU for training. Parameter update is performed by the CPU.

(IV) CoTrain: A scheduler for improvised CPU GPU executions.

1. Publication: https://dl.acm.org/doi/fullHtml/10.1145/3605573.3605647
2. Description: Last stage schedule improvisation on DeepSpeed

(V) Hyscale GNN: a hybrid CPU GPU/FPGA trainer for large scale GNNs

1. Publication: https://arxiv.org/pdf/2303.00158
2. Description: CPU and GPU collaboratively train the model. 2 stage data prefetching and dynamic resource management make sure that both CPU and GPU don't become overloaded with tasks.

Generic Links:

1. Hippie Paper(Last Stage Schedule in Pipeline Parallelism) : https://dl.acm.org/doi/abs/10.1145/3472456.3472497
2. Find the Latest Machine Learning Systems Papers Here: https://github.com/byungsoo-oh/ml-systems-paper
3. Attention is all new need (Transformer paper for reference): https://arxiv.org/pdf/1706.03762

#———————————————————————————————— Literature Survey Ends Here ————————————————————————————

#——————————————…——————————PyTorch   Tools For Implementation ——————————————————————————

1. PyTorch Documentation: https://pytorch.org/docs/stable/index.html

I) (Optional For Now) torch autograd function:

1. https://pytorch.org/docs/stable/autograd.html
2. Description: torch's autograd implementation and docs. Reading on custom autograd functions is suggested.

II) PyTorch Distributed Library (for distributed communication protocols)

1. https://pytorch.org/tutorials/beginner/dist_overview.html
2. Communication Package: https://pytorch.org/docs/stable/distributed.html
3. Remote Procedure Call (RPC) framework: https://pytorch.org/docs/stable/rpc.html
4. Distributed library is used for inter process communication. Dist allows you to initialise process groups and establish communication

protocols between them.

III) PyTorch Multiprocessing Library(Helpful for custom parallelisation strategies.)

1. Package: https://pytorch.org/docs/stable/multiprocessing.html
2. Best Practices and IPC: https://pytorch.org/docs/stable/notes/multiprocessing.html
3. Note: Can be used to implement shared memory system approach and parameter server approach.

IV) PyTorch Hooks
1. Note: Each hook functionality has its own attributes kindly thoroughly understand the functionality of each of the hooks before using them.
2. Description: Hooks are handles to functions which fire on a particular event completion trigger. These events include forward pass and backward pass.

Different Hooks: (All the hooks are either methods of nn.Module/Tensor class)

1. Torch Tensor class: https://pytorch.org/docs/stable/tensors.html
2. Torch nn.Module Class: https://pytorch.org/docs/stable/generated/torch.nn.Module.html

Note: Find the hook methods by typing "register_hook" in tensor document and "register_{forward/backward}_hook" in Module document.

Note: Use torchvision/ Hugging Face/ Models,Datasets used in Papers ONLY to test all the parallelisation strategies.

#———————————————————————— Tools End Here ——————————————————————————————————

# ————————————————————————AI Models for training ————————————————————————————

1. For simple learning: VGG19 + CIFAR10 Dataset. Import from torch vision library.
2. Transformer Based Models: Encoder+Decoder / BERT(encoder only)/ GPT (decoder only). Import from hugging face
3. Colossal AI: https://github.com/hpcaitech/ColossalAI
4. Nvidia Nemo: https://github.com/NVIDIA/NeMo

#——————————…——————————-

#————————————————————————- Video Links

——————————————————————————

I) Core Tools:

1. OpenMP: https://www.youtube.com/watch?v=cMWGeJyrc9w&list=PLLbPZJxtMs4ZHSamRRYCtvowRS0qIwC-I
2. CUDA: https://www.youtube.com/watch?v=GOam2jFb700&list=PLbRMhDVUMngfj_NXI7jqMYLnhcRhRKAGq

II) PyTorch DDP:

1. https://www.youtube.com/watch?v=-K3bZYHYHEA&list=PL_lsbAsL_o2CSuhUhJIiW0IkdT5C2wGWj

III) AIAUN: (Transformer from scratch implementation on PyTorch):

1. Code From Scratch: https://www.youtube.com/watch?v=ISNdQcPhsts&t=9651s
2. Lecture from scratch: https://www.youtube.com/watch?v=kCc8FmEb1nY&t=3828s
3. Essence and Concept building (3 blue 1 brown): https://www.youtube.com/watch?v=wjZofJX0v4M&t=1225s
4. Essence and Concept building (StatQuest): https://www.youtube.com/watch?v=zxQyTK8quyY&t=3s