Job Scheduling Multi-Node Execution

NVIDIA Professional Services & Quantiphi

1. Environment Preparation

- mkdir demo
- cd demo
- enroot –help

- enroot import docker://quantiphinvidiapractice/pytorch:23.05-py3

- enroot create --name pytorch quantiphinvidiapractice+pytorch+23.05-py3.sqsh

- enroot list

2. Implementation - Multi-Node Training

- wget https://storage.googleapis.com/cdac-data/minGPT-ddp.zip
- unzip minGPT-ddp.zip
- Create a multi-node job file multi_node.sh

```
#!/bin/bash

#SBATCH --nodes=2
#SBATCH --job-name=multinode-example
#SBATCH --partition=dgxnp
#SBATCH --ntasks=2
#SBATCH --gres=gpu:A100-SXM4:8


nodes=( $( scontrol show hostnames $SLURM_JOB_NODELIST ) )
nodes_array=($nodes)
head_node=${nodes_array[0]}
echo $head_node
head_node_ip=$(srun --nodes=1 --ntasks=1 -w "$head_node" hostname --ip-address)
echo $head_node_ip
echo Node IP: $head_node_ip
export LOGLEVEL=INFO
head_node_array=($head_node_ip)
head_n=${head_node_array[1]}

srun --no-container-entrypoint  --container-image
$(pwd)/quantiphinvidiapractice+pytorch+23.05-py3.sqsh --container-mounts
$(pwd)/minGPT-ddp/:/workspace/ \
torchrun \
--nnodes 2 \
--nproc_per_node 8 \
--rdzv_id $RANDOM \
--rdzv_backend c10d \
--rdzv_endpoint $head_n:29500 \
/workspace/mingpt/main.py
```

In the above script we can make the following changes :

1. Change the number of nodes by changing
    a. #SBATCH –nodes - *line 4*
    b. #SBATCH --ntasks - line 5
    c. –nnodes - *line 24*
2. Change the number of GPUs by changing
    a. #SBATCH –gres=gpu:A100-SXM4:(No. Of gpus) - line 6
    b. –nproc_per_node - line 25

- Run the script : **sbatch sbatch_run.sh**
    - Submitted batch job job-id

- Check the job with squeue
    - squeue
      job-id dgxnp multinod abhishek R 0:29 2
      node-1,node-2

- Verify the output file - slurm-<job-id>.out

```
WARNING:torch.distributed.run:
*****************************************
Setting OMP_NUM_THREADS environment variable for each process to be 1 in default, to avoid your system being overloaded, please further tune the variable for
optimal performance in your application as needed.
*****************************************
INFO:torch.distributed.launcher.api:Starting elastic_operator with launch configs:
  entrypoint       : /workspace/distributed/minGPT-ddp/mingpt/main.py
  min_nodes        : 2
  max_nodes        : 2
  nproc_per_node   : 8
  run_id           : 12222
  rdzv_backend     : c10d
  rdzv_endpoint    : 172.50.0.64:29500
  rdzv_configs     : {'timeout': 900}
  max_restarts     : 0
  monitor_interval : 5
  log_dir          : None
  metrics_cfg      : {}

INFO:torch.distributed.elastic.agent.server.local_elastic_agent:log directory set to: /tmp/torchelastic_gpqxakws/12222_hw_tu0ro
INFO:torch.distributed.elastic.agent.server.api:[default] starting workers for entrypoint: python
INFO:torch.distributed.elastic.agent.server.api:[default] Rendezvous'ing worker group
INFO:torch.distributed.elastic.agent.server.local_elastic_agent:log directory set to: /tmp/torchelastic_3xg2a6i1/12222_uvfjwx8o
INFO:torch.distributed.elastic.agent.server.api:[default] starting workers for entrypoint: python
INFO:torch.distributed.elastic.agent.server.api:[default] Rendezvous'ing worker group
INFO:torch.distributed.elastic.agent.server.api:[default] Rendezvous complete for workers. Result:
  restart_count=0
  master_addr=scn64-mn
  master_port=53295
  group_rank=0
  group_world_size=2
  local_ranks=[0, 1, 2, 3, 4, 5, 6, 7]
  role_ranks=[0, 1, 2, 3, 4, 5, 6, 7]
  global_ranks=[0, 1, 2, 3, 4, 5, 6, 7]
  role_world_sizes=[16, 16, 16, 16, 16, 16, 16, 16]
  global_world_sizes=[16, 16, 16, 16, 16, 16, 16, 16]

INFO:torch.distributed.elastic.agent.server.api:[default] Starting worker group
INFO:torch.distributed.elastic.agent.server.local_elastic_agent:Environment variable 'TORCHELASTIC_ENABLE_FILE_TIMER' not found. Do not start FileTimerServer.
INFO:torch.distributed.elastic.multiprocessing:Setting worker0 reply file to: /tmp/torchelastic_gpqxakws/12222_hw_tu0ro/attempt_0/0/error.json
INFO:torch.distributed.elastic.agent.server.api:[default] Rendezvous complete for workers. Result:
  restart_count=0
  master_addr=scn64-mn
  master_port=53295
  group_rank=1
  group_world_size=2
  local_ranks=[0, 1, 2, 3, 4, 5, 6, 7]
  role_ranks=[8, 9, 10, 11, 12, 13, 14, 15]
  global_ranks=[8, 9, 10, 11, 12, 13, 14, 15]
  role_world_sizes=[16, 16, 16, 16, 16, 16, 16, 16]
  global_world_sizes=[16, 16, 16, 16, 16, 16, 16, 16]
```

```
[GPU0] Epoch 1000 | Iter 0 | Eval Loss 0.00023
[GPU7] Epoch 1000 | Iter 0 | Eval Loss 0.00081
[GPU9] Epoch 1000 | Iter 0 | Eval Loss 0.00035
[GPU15] Epoch 1000 | Iter 0 | Eval Loss 0.00010
[GPU12] Epoch 1000 | Iter 0 | Eval Loss 0.00028
[GPU14] Epoch 1000 | Iter 0 | Eval Loss 0.00034
[GPU13] Epoch 1000 | Iter 0 | Eval Loss 0.00025
[GPU10] Epoch 1000 | Iter 0 | Eval Loss 0.00012
[GPU8] Epoch 1000 | Iter 0 | Eval Loss 0.00033
[GPU11] Epoch 1000 | Iter 0 | Eval Loss 0.00032
[GPU6] Epoch 1000 | Iter 0 | Eval Loss 0.00020
[GPU4] Epoch 1000 | Iter 0 | Eval Loss 0.00035
[GPU5] Epoch 1000 | Iter 0 | Eval Loss 0.00033
[GPU2] Epoch 1000 | Iter 0 | Eval Loss 0.00048
[GPU1] Epoch 1000 | Iter 0 | Eval Loss 0.00076
[GPU3] Epoch 1000 | Iter 0 | Eval Loss 0.00067
INFO:torch.distributed.elastic.agent.server.api:[default] worker group successfully finished. Waiting 300 seconds for other agents to finish.
INFO:torch.distributed.elastic.agent.server.api:Local worker group finished (SUCCEEDED). Waiting 300 seconds for other agents to finish
INFO:torch.distributed.elastic.agent.server.api:[default] worker group successfully finished. Waiting 300 seconds for other agents to finish.
INFO:torch.distributed.elastic.agent.server.api:Local worker group finished (SUCCEEDED). Waiting 300 seconds for other agents to finish
INFO:torch.distributed.elastic.agent.server.api:Done waiting for other agents. Elapsed: 0.0006580352783203125 seconds
INFO:torch.distributed.elastic.agent.server.api:Done waiting for other agents. Elapsed: 0.004417657852172851 6 seconds
```

3. Nsight Systems and Nsight Compute

Environment Preparation:

- enroot import docker://nvcr.io#nvidia/pytorch:22.04-py3
- enroot create --name profile nvidia+pytorch+22.04-py3.sqsh
- enroot start --rw profile bash

Nsight Systems Profiling:

- wget https://raw.githubusercontent.com/NVIDIA/nsight-training/master/cuda/2021_gtc/x-ac-03-v1/task1/task/nsys/application/main_baseline_nvtx.py
- nsys --help
- nsys profile --help
- Run NSight Systems profiling and store/print out detailed statistics, wait 30 seconds before beginning and only profile for 20 seconds
    - nsys profile --stats=true --delay 30 --duration 20 python3 main_baseline_nvtx.py

Multi-GPU Profiling

- Edit main_baseline_nvtx.py
    - line 179 change
    - [**model = Net().to(device)**] TO [**model = torch.nn.DataParallel(Net()).to(device)**]

If desired, download this report to your local machine and explore using the NSight Systems UI which can be downloaded here.
https://developer.nvidia.com/nsight-systems

Nsight Compute:

- wget https://gitlab.com/NERSC/roofline-on-nvidia-gpus/-/raw/master/example-codes/kernel_abc.cu

- ncu –help
- Simple example with three kernels called Kernel A, Kernel B, and Kernel C
    - cat kernel_abc.cu
- Compile kernel_abc.cu
    - nvcc –o kernel_abc kernel_abc.cu
- Run Nsight Compute on the executable with defaults
    - ncu ./kernel_abc
- Run NSight Compute on the executable but only profile kernel C
    - ncu -k kernel_C ./kernel_abc

- NSight Compute has different detail levels, let's run the full test suite with "--set full"" and save the output into a file called kernel_abc.ncu-rep
    - ncu -o kernel_abc.ncu-rep --set full ./kernel_abc

If desired, download this report to your local machine and explore using the NSight Compute UI which can be downloaded here.
https://developer.nvidia.com/nsight-compute