SambaNova SYSTEMS

Best Practices, Q&A

May 1, 2024

Safe Harbor Statement

The following is intended to outline our general product direction at this time. There is no obligation to update this presentation and the Company's products and direction are always subject to change. This presentation is intended for information purposes only and may not be relied upon for any purchasing, partnership, or other decisions.

Table of Contents

- Do's
  - General
  - Modelbox
- Bringup
  - Compiling
  - Running
- Basic checks
- slurm_feeder
- Discussion

General Do's

- Do's:
  - Use an appropriate virtual environment (next slide)
  - Use **top** and **squeue** to verify node's idleness
  - Use **Slurm**
  - Use logging tools and flags for already established tools
    - use specialized logging tools only available on ANL for establishing models(RCW_toolkits)
    - Use `/opt/sambaflow/bin/sn*stat` tools for various stats (tile, ddr, pcie)
  - Use practices that optimize performance on the RDU
    - Use tools like **Sambatune** to examine performance

SambaNova SYSTEMS

Setup

- Always source the appropriate venv

```
...
$USER@sn30-r1-h1:/opt/sambaflow/apps# find . -name activate -print
./private/anl/venv/bin/activate
./starters/lenet/venv/bin/activate
./starters/logreg/venv/bin/activate
./micros/venv/bin/activate
./nlp/transformers_on_rdu/genslm/venv/bin/activate
./nlp/transformers_on_rdu/gpt13b/venv/bin/activate
./nlp/transformers_on_rdu/venv/bin/activate
./image/deepvit/venv/bin/activate
...
```

- For example:
  - `source /opt/sambaflow/apps/private/anl/venv/bin/activate`

Modelbox Do's

- Do's:
  - Use **screen** or **tmux** to setup workspace that is easy to return to in between development
  - Use `/data/scratch/$USER` for more granted Memory storage
  - Use `/data/ANL/results` to store compile artifacts
  - Verify Model architecture and input requirements
  - Verify that Data is properly formatted for the model you are testing working with

Bringup (Compile)

- Use BF16 in your code, when possible
  - Maximum operators are supported in BF16 and performant in BF16 (see documentation)
- Checkpoint should be using BF16
  - NLP checkpoint support for FP32 should be available in Sambaflow 1.18+
- Preprocessed datasets should be saved in BF16 (if applicable)
  - Most NLP datasets are integers.
- Compilation:
  - Compile using --log-level error
  - Use SN_NUM_THREADS=<INT> to parallelize compilation
  - Model compile and run phase should use same definition of the model
    - Otherwise, tracing phase will complain for missing/mismatched tensors
  - Recompilation
    - Only model definition change or input shape change requires recompilation

Bringup (Compile)

- Unsupported operators
  - Can be re-implemented using supported operators
    - e.g. torch.repeat() can be replaced by tensor copy and then expand()
  - Unsupported loss functions
    - Can be done on CPU
    - Slows down performance of model due to tensor transfers
  - SN NLP models use **AdamW**, not Adam as the optimizer
- Lookout for faster implementations
  - In some cases, torch.pow can be replaced by input.scale(1.0)*input
- Sambaloader:
  - The loader should set `drop_last=True` to avoid partial batches
  - `--num-workers` is a key argument to pass to the loader as it enables parallel loading

Bringup (Running)

- Always use slurm
- Always pass `--gres=rdu:nrdus` when using slurm
- Always pass `--cpus-per-task=<ncpus>` when using slurm
- Always set **OMP_NUM_THREADS**
  - `export OMP_NUM_THREADS=1`
- If you do not know how many RDUs you need, use slurm_feeder
  - slurm_feeder reads the pef to determine the RDU count
  - Note: RDU count is always 1 unless the model was compiled for Data Parallel, Model Parallel or is using spatial mapping
- Useful Python tools
  - Debugger: **pdb**
  - Profiler: **py-spy**
- When compiling for Data Parallel, set `-ws 2` (world size)
  - Once set, any number of RDUs (2 or more) can be used for running

Bringup (Running)

- Batch size:
  - Always sweep on batch sizes to determine the sweet spot
  - Models with `--grad-accummulation-steps` argument support ability to change batch size in multiples of a base batch size (`-b` argument)
- Host CPU overhead for dataloaders:
  - sweep on `--num-workers` to determine the sweet spot
- When doing data-parallel on one host
  - export `SAMBA_CCL_PCIE_TRANSPORT=1` (use local pcie transport network within host)
- If you have a significant host component you should sweep on **OMP_NUM_THREADS** to determine the sweet spot
- Use `SF_RNT_TILE_AFFINITY` to debug issue related to jitter on your runs
  - It asks the Runtime to assign your job to specific TILES/RDUS
  - Details on next slide

Bringup (Running)

SN30: Use the 64-bit SF_RNT_TILE_AFFINITY environment variable to hint scheduler to choose the physical RDU/TILE where your graph should be running

- Note: This variable maps to RDUs within a single node
- Each byte (8-bit set) maps to an RDU:
  - `Bit[0-7]` maps to RDU0, `Bit[8-15]` maps to RDU1, and so on until `Bit[56-63]` maps to RDU7
  - Each bit within the byte is a Tile ID that maps to a physical tile on the RDU, starting from LSB (0) to MSB (7).
    - rdu0 = 11111111 = 255 = 0xff
    - rdu1 = 11111111 00000000 = 0xff00
    - rdu2 = 11111111 00000000 00000000 = 0xff0000
    - rdu3 = 0xff000000
    - rdu4 = 0xff00000000
    - rdu5 = 0xff0000000000
    - rdu6 = 0xff000000000000
    - rdu7 = 0xff00000000000000

Bringup (Running)

- If you have noticeable transfer time between host and RDU try
  - `--tensormem ddr`
- Always check to see if the CPUs are in performance mode
  - `cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor`
- Check /etc/environment for the following work arounds
  - `GLIBC_TUNABLES=glibc.tune.hwcaps=-AVX_Usable,-AVX2_Usable,-Prefer_ERMS,-Prefer_FSRM,Prefer_No_AVX512,Prefer_No_VZEROUPPER,-AVX_Fast_Unaligned_Load,-ERMS`
  - Set `IBV_FORK_SAFE=1` for python multiprocess safety
- Check your hardware environment

Basic Checks

- `snfadm -l inventory | less`

```
Platform: DataScale SN30-8

Physical Inventory:
Component Name                    Serial Number      Inventory State Functional State
/NODE/XRDU_0/RDU_0                | 205057B469B35895   | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_0/DIMM_A0   | 1F310A0          | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_1/DIMM_B0   | 1F31085          | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_2/DIMM_E0   | 1F3123D          | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_3/DIMM_F0   | 1F310E6          | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_4/DIMM_G0   | 1F6F66D          | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_5/DIMM_H0   | 1F6F8AE          | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_6/DIMM_C0   | 1F31058          | Present       | Online
/NODE/XRDU_0/RDU_0/DDRCH_7/DIMM_D0   | 1F31185          | Present       | Online
/NODE/XRDU_0/RDU_0/PCIE_0
...
```

Basic Checks

- Check if there are faults in hardware
  - Contact support if faults are present, errors are okay!
- `snfadm -l fault`

```
Timestamp              | Fault UUID                          | Fault Type          | Component Name                 |Status
-------------------------------------------------------------------------------------------------------------------------------------------
2024-03-25 12:32:59    | 7cc4780b-eade-11ee-88b7-5db0411a8cf0 | FTYPE_PCI_LINK_HEALTH | /NODE/XRDU_1/RDU_0/PCIE_1    | Active
2024-03-25 12:32:59    | 7cc4780d-eade-11ee-88b7-5db0411a8cf0 | FTYPE_PCI_LINK_HEALTH | /NODE/XRDU_1/RDU_0/PCIE_4    | Active
```

- `snfadm -l error`

```
Original Timestamp     | Error UUID                          | Error Type              | Component Name
-------------------------------------------------------------------------------------------------------------------------------------------
2024-03-25 12:07:31    | ee28a286-eada-11ee-b2d7-db955cbc340d | ETYPE_PCIE_LINK_DEGRADED  | /NODE/XRDU_0/RDU_0/PCIE_0
2024-03-25 12:07:31    | ee28a288-eada-11ee-b2d7-db955cbc340d | ETYPE_PCIE_LINK_FAULTED   | /NODE/XRDU_0/RDU_1/PCIE_0
2024-03-25 12:07:31    | ee28a28a-eada-11ee-b2d7-db955cbc340d | ETYPE_PCIE_LINK_FAULTED   | /NODE/XRDU_0/RDU_1/PCIE_5
2024-03-25 12:07:31    | ee28a28c-eada-11ee-b2d7-db955cbc340d | ETYPE_PCIE_LINK_FAULTED   | /NODE/XRDU_1/RDU_0/PCIE_0
2024-03-25 12:32:59    | 7cc4780a-eade-11ee-88b7-5db0411a8cf0 | ETYPE_PCIE_LINK_FAULTED   | /NODE/XRDU_1/RDU_0/PCIE_1
2024-03-25 12:32:59    | 7cc4780c-eade-11ee-88b7-5db0411a8cf0 | ETYPE_PCIE_LINK_DEGRADED  | /NODE/XRDU_1/RDU_0/PCIE_4
```

Basic Checks

- `snconfig show Node all`

```
XRDU Name: XRDU_2
        Number of RDUS: 2
        RDU name: RDU_0
                RDU Mem Size: 383.5 GB
                RDU Online Tiles: 4
                RDU is perfect: True
                RDU PCI BDF 0: d1:00.0
                RDU PCIE Name: PCIE_0
                        Bandwidth: 32 GB/s
                        Speed: 16 GT/s
                        Width: 16
```

slurm_feeder

- Computation graphs executed on RDUs are stored in PEF files which specify the required resources.
- SambaNova provides a Python library to read these resource requirement
- slurm_feeder is an example of Slurm job submission code in Python
  - It illustrates how to use this library to automate the composition of correct GRES requests
- slurm_feeder constructs a batch script that specifies the necessary resources and submits the job
  - Viewing that script will show you the required resources
  - A note on resources
    - If there are insufficient resources to meet requirements, the job will be in a **PENDING** state until it can be scheduled
    - If the resource requirement exceeds the resource spec of all nodes, the job will fail

slurm_feeder arguments

```
(venv):~$ /opt/sambaflow/slurm/python/slurm_feeder -h

usage: slurm_feeder [-h] [-a PYTHON_APP] -c {srun,sbatch} [--mpirun]
                    [-n NJOBS] -p PEF_FILE [--python-arg PYTHON_ARG]
                    [-s SCRIPT] [--single-tile] [-t TIME] [-w NODE]

optional arguments:
  -h, --help            show this help message and exit
  -a PYTHON_APP, --python-app PYTHON_APP
                        Path to python app to run
  -c {srun,sbatch}, --command {srun,sbatch}
                        Slurm command srun or sbatch
  --mpirun              Use mpirun to run
  -n NJOBS, --njobs NJOBS
                        Instances of job to run
  -p PEF_FILE, --pef-file PEF_FILE
                        Path to pef file to execute
  --python-arg PYTHON_ARG
                        Python app args
  -s SCRIPT, --script SCRIPT
                        Path to script to run
  --single-tile         Specify single-tile GRES requirement
  -t TIME, --time TIME  Minutes to run before job times out
  -w NODE, --node NODE  Which node to dispatch the job
```

slurm_feeder usage

Basic Structure:

```
slurm_feeder -c {slurm_command} -s {/path/to/script} -p {/path/to/pef}
```

Example:

```
(venv):~/app-test$ /opt/sambaflow/slurm/python/slurm_feeder -c sbatch -s ./logreq.sh -p out/logreg/logreg.pef

(venv):~/app-test$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               297      PROD 20719_lo  …  R       0:05      1 ..

(venv):~/app-test$ ls
batch.txt  job297-00.out  logreq.sh  mnist_data  out
```

Discussion

SambaNova® SYSTEMS