Training Neural Networks with hundreds of GPUs on Graham and Cedar

Fei Mao, SHARCNET
Outlines

• What is new?
  - Hardware/software

• Where to run a job?
  - Node types and scheduling policies

• Different methods for managing variables
  - Parameter server or replicated?
  - CPU memory or GPU memory?

• Interconnection and I/O bottleneck
  - TCP/IP or IB, Lustre or local

• Examples
## New systems

<table>
<thead>
<tr>
<th>system</th>
<th>GPU type</th>
<th>#GPU devices</th>
<th>target jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graham</td>
<td>P100-12g</td>
<td>320 (2 per node)</td>
<td>Single GPU, MultiGPU, Distributed</td>
</tr>
<tr>
<td>Cedar</td>
<td>P100-12g/16g</td>
<td>584 (4 per node)</td>
<td>Single GPU, MultiGPU, Distributed</td>
</tr>
<tr>
<td>Minsky</td>
<td>P100-NVLINK</td>
<td>4</td>
<td>Experiment</td>
</tr>
</tbody>
</table>

## Softwares

- Caffe2, TensorFlow, Theano, Torch
- More about software:
  - [https://docs.computecanada.ca/wiki/AI_and_Machine_Learning](https://docs.computecanada.ca/wiki/AI_and_Machine_Learning)
Node Types

Graham GPU nodes:
- 2 x E5-2683v4, 32 CPU cores, 128GB memory
- 2 x P100-12GB-PCIE, one each CPU socket
- IB FDR 56Gb/s, 1.6T NVMe SSD
- Accept single GPU jobs and whole node(s) jobs
- SLRUM 2 GPUs request: `#SBATCH --gres=gpu:2`
Node Types

Cedar Base GPU nodes:
• 2 x E5-2650v4, 24 CPU cores, 128GB memory
• 4 x P100-12GB-PCIE, two each CPU socket
• Intel OPA 100Gb/s (LP slot), 800GB SATA SSD
• Accept single GPU jobs and whole node(s) jobs
• SLRUM 4 GPUs request: #SBATCH --gres=gpu:4
Cedar Large GPU nodes: (best for ML/DL)
- 2 x E5-2650v4, 24 CPU cores, 256GB memory
- 4 x P100-16GB-PCIE, all under single CPU socket
- Intel OPA 100Gb/s (LP slot), 800GB SATA SSD
- Accept whole node(s) jobs, **single GPU jobs less than 24 hours**
- SLRUM 4 GPUs request: #SBATCH --gres=gpu:lgpu:4
Different methods for managing variables

Parameter Server(s):

Data Parallelism

\[ p'' = p' + \Delta p \]

- Parameter Servers
- Model Replicas
- Data

\( \Delta p' \)

\( p' \)
Different methods for managing variables

Parameter Server(s) in CPU or GPU?

- **G**
  - 2 CPU / 4 GPU
  - 2 Virtual Switches
  - 2 GPU per CPU
  - Bottleneck is between GPUs from different sockets
  - Good for **CPU** as server (Server operations can be overlapped with GPU training)

- **B**
  - 2 CPU
  - 4:1 Switched
  - Bottleneck is between CPU and GPU
  - Good for **GPU** as server
Different methods for managing variables

Replicated parameters on all GPUs:

- Updating parameters = `all_reduce` operation
- Total bandwidth between GPUs matters
- Cedar Large GPU node has the highest total PCIe bandwidth
- NVIDIA NCCL can be used, but not always the best, do benchmarking
Interconnection

Topology: Fat-Tree for both Graham and Cedar

Blocking factor:
- non-blocking for 32 servers under same “Edge” switch
- 2:1 blocking for Cedar when crossing switches
- 8:1 blocking for Graham when crossing switches
Network Protocols

TCP or IB (Infiniband)?
• IPoIB (IP over Infiniband) performance is ~30-40Gb/s

• Tensorflow:
  • gRPC (runs on top of TCP) by default
  • IB is supported as “third party contribution”

• Caffe2:
  • TCP only
  • IB will be supported in the future

• pyTorch:
  • initially MPI and raw TCP sockets, later RDMA

• Theano:
  • IB is supported via third party projects (e.g. Theano-MPI)

Choosing the correct interface when using TCP:
• Multiple network interfaces when run command “ifconfig”
• Should always use “ib0” interface’s IP address
• Other interfaces are very slow (some with only 1Gb/s)
Understanding I/O bottleneck

I/O will easily become bottleneck when training with many GPUs
- 1 P100 GPU can train Resnet-50 with a speed of 200 images/s (40MB/s)

- Lustre file system (/project and /scratch):
  - Able to achieve 30GB/s using hundreds of clients to load a huge file
  - Loading imagenet LMDB file sometime can only reach 300MB/s
  - Random accessing is even slower, can be less than 100MB/s

- Local file system (/localscratch):
  - SSD based, 800GB on Cedar, 1.6TB on Graham
  - Cedar has SATA SSD (~500MB/s), Graham has NVMe SSD (>1GB/s)
  - Need to copy the data every time before training starts
  - Local storage will be cleaned after job is finished/killed

- Highly suggest to copy the data to local SSD for faster random access
  - Original data can be stored on multiple OSTs on /project or /scratch to achieve higher throughput
  - Details on next page…
Lustre parallel file system

- By default, single file is stored in a single OST
- Commands to stripe the file across OSTs:
  - lfs setstripe -s 1m -c 8 <file> (stripe size 1MB, count 8 OSTs)
  - cp <target> <file> (above command only creates empty file)
  - lfs getstripe <file> (to check)
Examples

Common problems:

• What ML/DL tools are available?
• How to require hardware resources via SLRUM?
• How to launch the program on all nodes?
• How to use local storage?
• How to get correct IP address?
THANK YOU!

Q&A