Deep Learning on SHARCNET:

Best Practices

Fei Mao

Outlines

- What does SHARCNET have?
  - Hardware/software resources now and future
- How to run a job?
  - A torch7 example
- How to train in parallel:
  - A Theano-based MPI framework
- Why is Caffe running slow?
  - I/O considerations

What is SHARCNET?

Consortium of 18 Ontario institutions providing advanced computing resources and support...

compute • calcul CANADA

- member of Compute Canada
- > 3,000 Canadian and international users

Shared
Hierarchical
Academic
Research
Computing
NETwork

- 20,000 CPU cores
- 180 Tesla GPUs
- 10 Gb/s network

Essentials: Main GPU systems

system	GPU type	#GPU devices	target jobs
Monk	Tesla M2070	108 (2 per node)	*Light-weight Theano*
Copper (contributed)	Tesla K80	64 (8 per node)	*Single GPU, MultiGPU, Distributed*
Mosaic (contributed)	Tesla K20	20 (1 per node)	*Single GPU, Distributed*
GP3 (April 2017)	Tesla P100	>300	*Single GPU, Distributed*
IBM Minsky (very soon)	P100 with NV Link	4	*Testing*

- Non-contributors have 4 hour runtime limitation. (Do check pointing!)
- All have Infiniband high speed interconnection.
- More about software:
- https://www.sharcnet.ca/help/index.php/Machine_Learning_and_Data_Mining

Essentials: File system basics

location (path)	quota	expiry	access	purpose
`/home/$USER`	10 GB	none	unified	sources, small config files *backed up*
`/work(project)/$USER`	1 TB	none	unified*	active data files, automounted *convenient*
`/scratch/$USER`	none	2 months	per-cluster*	temporary files, checkpoints, *good performance*
`/tmp/$USER`	none	2 days	per-node*	*best random access, best performance if SSD*

Data transfer node: **dtn.sharcnet.ca**

- sees internet and most SHARCNET filesystems, no cputime limit

Storage limits and enforcement:

- scratch file systems have limited shared storage space
- Users exceeding quota or having >10^6 files in /work or /scratch will not be able to start new jobs!

Essentials: Managing jobs with SQ

- All significant work is submitted to the system as a *job*
- Jobs are run in *batch mode* via a job scheduling system
  - enforces policies to promote fair and efficient use
- Jobs are submitted using the **sqsub** command:

sqsub –r *run_time* -o log_file [*options*] *program* [*args*]

- required options are **-r** and **-o**
  - ***run_time*** - estimated run time for the job, eg.“**-r 1d”**
    - < 7 days, after ***run_time***, job is killed
  - For gpu jobs one has to specify the queue **"-q gpu**” and choose MPI or threaded “**-f mpi** or **threaded**” and number of cpus “**-n ...**”, gpus per node “**--gpp=**“, memory “**--mpp=**“
- ***sqjobs*** **:** list the status of submitted jobs
- ***sqkill*** **:** stop/dequeue a running/queued job

Running Torch7 Jobs

- A Neural Algorithm of Artistic Style

Running Torch7 Jobs

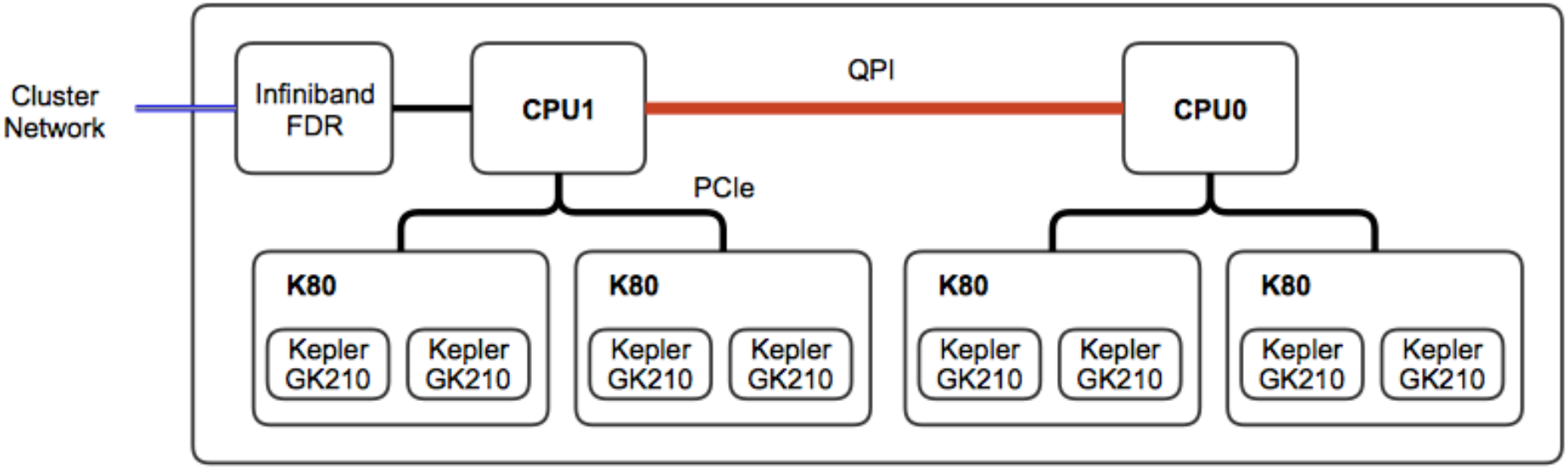
1. Get the code: git clone https://github.com/repo.git
2. Prepare the data
    - Only login nodes have access to internet
    - Use “dtn” server to download large data
3. Set the software environment
    - Easy bash script for DL tools on SHARCNET
4. Submit jobs
    - Choose the number CPU/GPU, mem size, runtime
    - Don’t ask more than needed!

More here:
https://www.sharcnet.ca/help/index.php/Torch

Multi-GPU training

1. Intra-node parallelism for nodes with multiple GPUs
2. Inter-node parallelism for nodes with high speed network

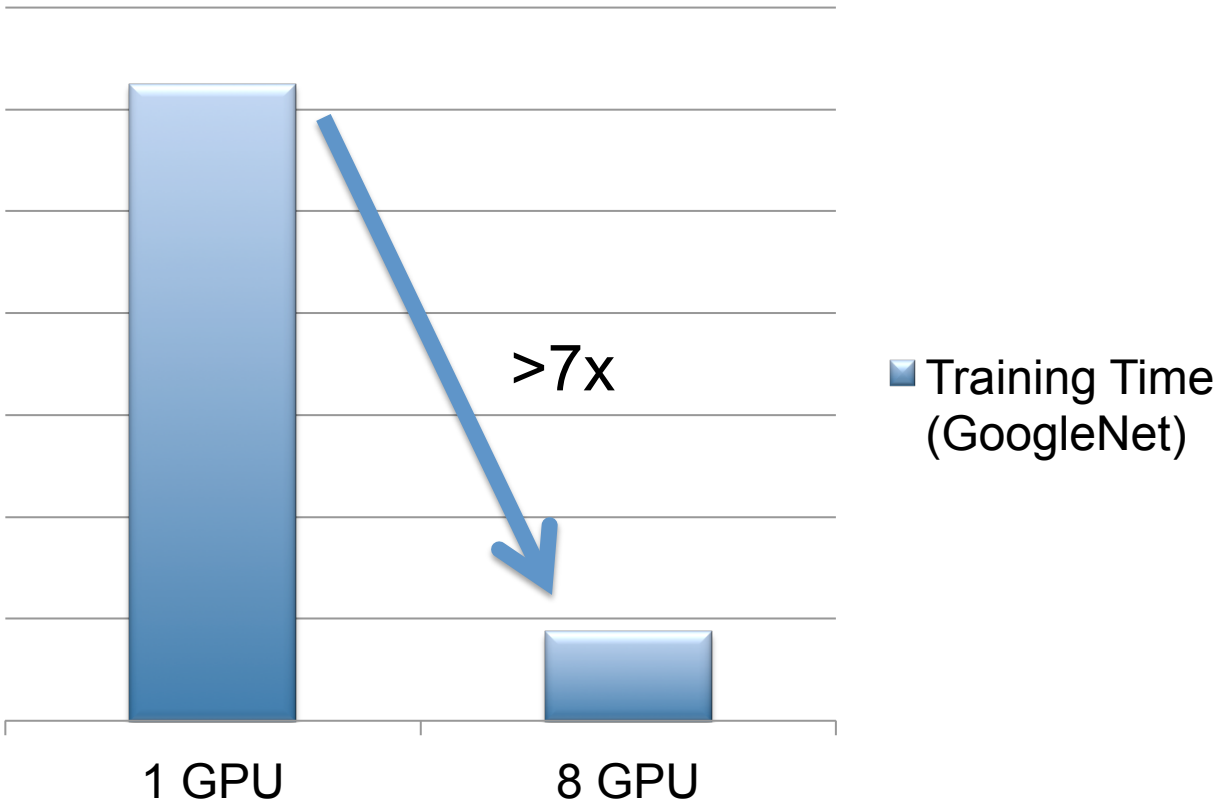
Hardware layout of a Copper's GPU node



Multi-GPU training

Theano-MPI: a Theano-based Distributed Training Framework
(He Ma, Fei Mao, Nikhil Sapru, Graham W. Taylor)

https://github.com/uoguelph-mlrg/Theano-MPI.git



Why is Caffe running slow?

```
123874 solver.cpp:228] Iteration 560, loss = 6.69755
123874 solver.cpp:244]     Train net output #0: loss = 6.69755 (* 1 = 6.69755 loss)
123874 sgd_solver.cpp:106] Iteration 560, lr = 0.01
123884 blocking_queue.cpp:50] Waiting for data
123884 blocking_queue.cpp:50] Waiting for data
123884 blocking_queue.cpp:50] Waiting for data
123884 blocking_queue.cpp:50] Waiting for data
123874 solver.cpp:228] Iteration 580, loss = 6.62282
123874 solver.cpp:244]     Train net output #0: loss = 6.62282 (* 1 = 6.62282 loss)
123874 sgd_solver.cpp:106] Iteration 580, lr = 0.01
123884 blocking_queue.cpp:50] Waiting for data
123884 blocking_queue.cpp:50] Waiting for data
123884 blocking_queue.cpp:50] Waiting for data
123884 blocking_queue.cpp:50] Waiting for data
123884 blocking_queue.cpp:50] Waiting for data
123874 solver.cpp:228] Iteration 600, loss = 6.50693
123874 solver.cpp:244]     Train net output #0: loss = 6.50693 (* 1 = 6.50693 loss)
123874 sgd_solver.cpp:106] Iteration 600, lr = 0.01
```

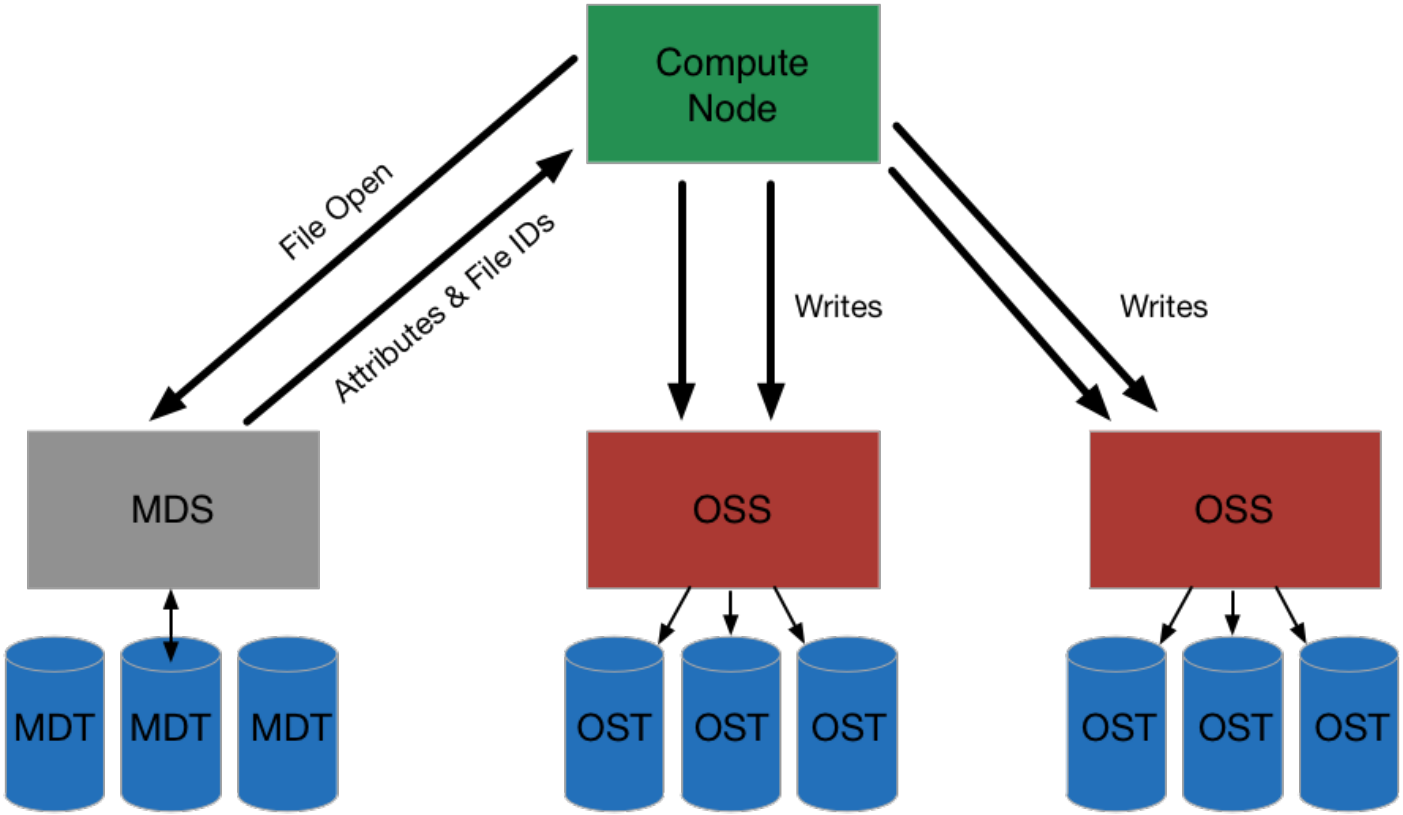
- We all know that faster training needs faster I/O
- Which is faster? /work or /tmp?

Why is Caffe running slow?

```
130075 solver.cpp:228] Iteration 336660, loss = 1.22738
130075 solver.cpp:244]     Train net output #0: loss = 1.22738 (* 1 = 1.22738 loss)
130075 sgd_solver.cpp:106] Iteration 336660, lr = 1e-05
130075 solver.cpp:228] Iteration 336680, loss = 1.30872
130075 solver.cpp:244]     Train net output #0: loss = 1.30872 (* 1 = 1.30872 loss)
130075 sgd_solver.cpp:106] Iteration 336680, lr = 1e-05
130075 solver.cpp:228] Iteration 336700, loss = 1.6427
130075 solver.cpp:244]     Train net output #0: loss = 1.6427 (* 1 = 1.6427 loss)
130075 sgd_solver.cpp:106] Iteration 336700, lr = 1e-05
130075 solver.cpp:228] Iteration 336720, loss = 1.27567
130075 solver.cpp:244]     Train net output #0: loss = 1.27567 (* 1 = 1.27567 loss)
130075 sgd_solver.cpp:106] Iteration 336720, lr = 1e-05
130075 solver.cpp:228] Iteration 336740, loss = 1.66995
130075 solver.cpp:244]     Train net output #0: loss = 1.66995 (* 1 = 1.66995 loss)
130075 sgd_solver.cpp:106] Iteration 336740, lr = 1e-05
130075 solver.cpp:228] Iteration 336760, loss = 1.36151
130075 solver.cpp:244]     Train net output #0: loss = 1.36151 (* 1 = 1.36151 loss)
130075 sgd_solver.cpp:106] Iteration 336760, lr = 1e-05
130075 solver.cpp:228] Iteration 336780, loss = 1.3062
130075 solver.cpp:244]     Train net output #0: loss = 1.3062 (* 1 = 1.3062 loss)
130075 sgd_solver.cpp:106] Iteration 336780, lr = 1e-05
```

- /work (~1GB/s) is fast, but why is Caffe waiting for data?
- What happens if data on /tmp (~100MB/s)?
- Random access speed is important for LMDB, which is low for a remote file system like /work and /scratch
- What is the best way to copy the data from /work to /tmp?

Parallel file system (Lustre)



- By default, single file is stored in a single OST
- 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)

Performance with stripe size/count

"rsync" ~9.4GB from /work to /tmp (imagenet val lmdb):

- 1m-1c: 73.56MB/s, 60.71MB/s, 62.82MB/s
- 1m-2c: 136.60MB/s, 104.86MB/s, 130.03MB/s
- 1m-4c: 180.46MB/s, 165.92MB/s, 141.17MB/s
- 1m-8c: 181.27MB/s, 175.14MB/s, 176.73MB/s
- 4m-8c: 132.11MB/s, 139.70MB/s, 111.86MB/s

Tips:

- Keep stripe size to be 1m (default),stripe count to be 4 to 8
- /work is for large file sequential read/write, /tmp is for small files or random access to a large file
- Small files should be moved to /tmp in a tar format
- Multiple copies of common dataset (e.g. Imagenet) on every nodes' /tmp without expiry

THANK YOU!

Q&A