Introduction to GPU Computing on Raad2

Research Computing

Dec 2019

Qatar Foundation

TEXAS A&M UNIVERSITY at QATAR

Overview

- GPU Cluster at TAMUQ
- Software stack
- Hands on exercises
- Resources

The “Introduction to Linux” and “Introduction to computing on raad2”short courses offered by Research Computing should be treated as a prerequisite for this course.

TEXAS A&M UNIVERSITY at QATAR

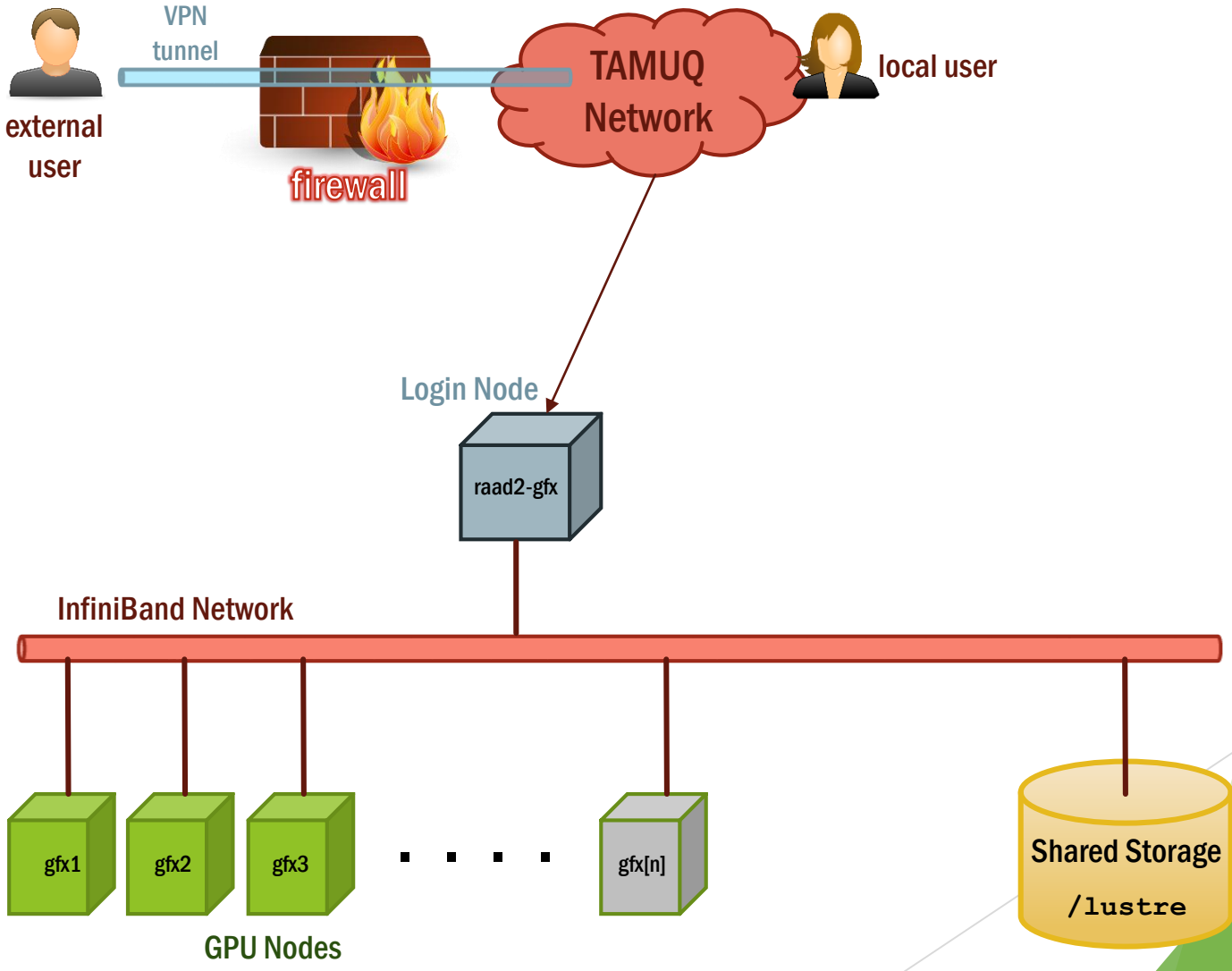
GPU Cluster at TAMUQ

raad2-gfx.qatar.tamu.edu

Introduction

- 03 node GPU cluster with 06 NVIDIA Volta V100 GPUs was acquired in early 2019.
- GPU cluster at TAMUQ is part of raad2 supercomputer but it is logically a different cluster.
  - Shares same storage system i.e. Lustre
  - Shares same authentication mechanism and passwords are synced.
  - Different nodes configuration and high speed network.
  - Different software stack i.e. Applications and Compilers etc.
  - Different instance of resource manager i.e. SLURM.
- Supports multiple AI/ML and HPC applications.
- Containers support is available.

System Architecture



GPU node architecture

- Each node in GPU cluster has
  - 02 Nvidia V100 GPUs connected via PCIe link.
    - cgroups are used to isolate GPUs in multi tenant environment.
  - 02 Intel Xeon Gold 6140 Processor.
    - 18 cores each with 2.30 GHz base and 3.70 GHz turbo freq.
  - 192 GB memory.
  - 800GB local SSD.

Technical specifications

Login Node	raad2-gfx.qatar.tamu.edu
GPU	02 NVIDIA Tesla V100 Per Node
GPU Nodes	gfx[1-3]
Memory	192 GB per Node
CPU	Intel Xeon 6140 2.30 GHz Base and 3.70 GHz Turbo Freq.
Sockets	02 Per Server
Cores Per Socket	18 cores
Node Local Storage	800GB per node (SSD)

Nvidia Tesla V100 GPU

- Nvidia Tesla V100 is a powerful datacenter GPU which is designed for AI, HPC and graphics.

GPU Architecture	NVIDIA Volta
Tensor Cores	640
CUDA Cores	5,120
Double precision performance	7 TFLOPS
Single precision performance	14 TFLOPS
GPU Memory	16GB
Memory bandwidth	900 GB/sec
Compute APIs	CUDA, DirectCompute, OpenCL, OpenACC

User resource limits

Workload Manager	Slurm
Queue	gpu
Local SSD Storage	400GB
Per Job GPU Limit	02
Per User GPU Limit	02
Max Wall time	03:00:00

A user requesting 01 GPU will be allocated 18 cores on same socket, 400GB SSD local storage and 96GB of memory.

Software Stack

Packages and Compilers

- What is available on the system?
  - CUDA toolkit (9.0, 9.1, 9.2, 10.0)
  - cuDNN 7.3.1
  - Multiple versions of Python (2.7 & 3.6)
    - Python for ML/DL applications, we recommend use of Anaconda distribution.
  - HPC applications such as Gromacs, Lammps, MATLAB and NAMD.
- Can I bring my own application?
  - Any Python/R code can be easily ported to the system. Users can install any python packages at their own with Conda virtual environments.
  - Users can bring their containerized applications. Singularity containers are available on the system.

Development mode

- Interactive session to GPU nodes allows users to develop and test their code before production runs.
- To submit an interactive job to the system, issue;
  - `sinteractive`
  - Allocates 1 GPU with 18 cores for 1 hour with ssh access.
- Submit an interactive job with customized paramteres
  - `srun --ntasks=18 --gres=gpu:v100:1 --walltime=04:00:00 --pty /bin/bash`
- Load required programming environment.
  - `module load cuda10.0`
- Run your application.
  - `nvcc mycode.cu`
- Terminate the interactive session.
  - `exit`

Production mode

- Once you have developed and tested your code in development mode, you can submit batch jobs for longer runs.
- A sample SLURM job file should look like below;

```
#!/bin/bash
#SBATCH -J batch
#SBATCH --qos=normal
#SBATCH --time=24:00:00
#SBATCH --ntasks=18
#SBATCH --gres=gpu:v100:1
srun --ntasks=1 python myapp.py
```

```
sbatch gpu.job
```

Anaconda

- The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X.
- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

Conda Python Environment

- In development mode, create conda Python virtual environment
  - `conda create -n dlproject python=3.6`
- Activate the newly created environment
  - `conda activate dlproject`
- Install required packages
  - `conda install tensorflow-gpu`
  - `conda install keras`
- Run python application
  - `python myapp.py`
- Deactivate the environment
  - `conda deactivate`
- List available environments
  - `conda env list`

Conda Cheat Sheet: https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

Jupyter Lab and Notebooks

- JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning.
- Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
- Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
- https://jupyter.org/

Jupyter Notebook on raad2-gfx

Step 01: On Local machine, login to raad2-gfx with port forwarding

```
ssh -L PORT:localhost:PORT username@raad2-gfx.qatar.tamu.edu
```

Step 02: Start development session on raad2-gfx by submitting interactive job

```
srun --ntasks=18 --gres=gpu:v100:1 --tunnel LPORT:PORT --pty
/bin/bash
```

Step 3: Setup conda environment

```
source /cm/shared/apps/anaconda3/etc/profile.d/conda.sh
```

Step 4: Create and activate Conda Environment

```
conda create -n myproject python=3.6
conda activate myproject
conda install jupyter
conda install jupyterlab
```

Step 5: Unset XDG_RUNTIME_DIR (Fix for known bug)

```
unset XDG_RUNTIME_DIR
```

Step 6: Launch Jupyter Notebook

```
jupyter lab --port PORT
```

*You may choose any random PORT b/w [8100 – 9999]

Hands-on Session

Resources

- Research Computing Wiki Page
  - https://rc-docs.qatar.tamu.edu/index.php/Raad2-gpu
- TensorFlow GPU Guide
  - https://www.tensorflow.org/guide/gpu
- Matlab GPU Guide
  - https://www.mathworks.com/help/parallel-computing/run-matlab-functions-on-a-gpu.html
- Nvidia GPU Cloud
  - https://ngc.nvidia.com/