Digital Research Alliance of Canada

Accelerated DataFrame with Dask-cuDF on multiple GPUs

Jinhui Qin

SHARCNET | Compute Ontario | Digital Research Alliance of Canada

help@sharcnet.ca

SHARCNET™

Compute • Calcul Ontario

Overview

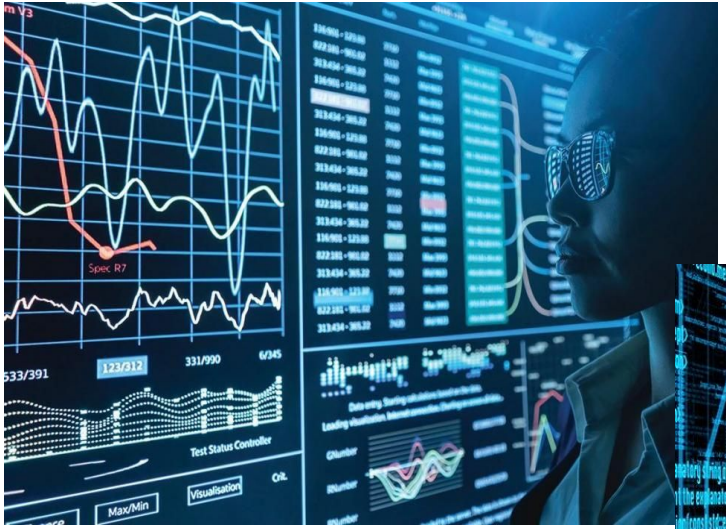
python™

NumPy

pandas

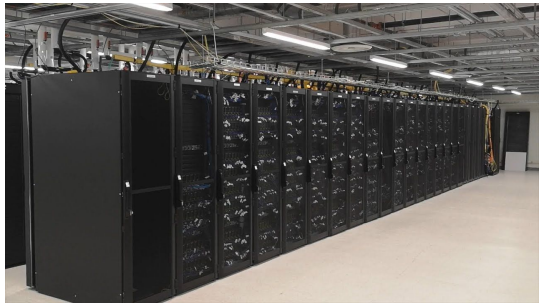
SciPy

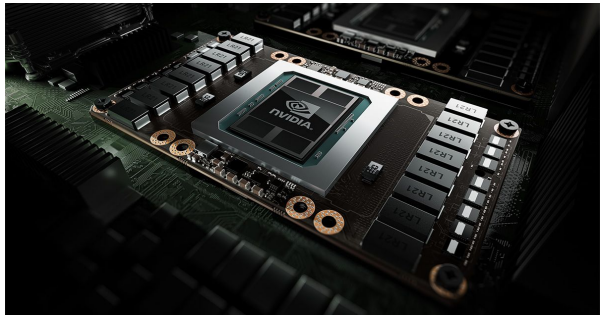
scikit learn



Processing Large Datasets

- Distributed data analysis
- Using GPU accelerators





DASK

APACHE Spark

RAPIDS

PyTorch

TensorFlow

RAPIDS

- A suite of open source software libraries from NVIDIA
- Executing data science and analytics pipelines on GPUs
- Built on NVIDIA CUDA primitives
- Scalable with Dask integration
- Provided with user-friendly Python interface
  - E.g. cuDF, cuML, cuGraph, cuSignal, cuSpacial, …
- Helpful links
  - Docs: https://docs.rapids.ai/
  - Examples: https://github.com/rapidsai/notebooks
  - Use cases and blogs: https://medium.com/rapids-ai

Dask-cuDF

- **Pandas**, a CPU based DataFrame library
- **cuDF**, a GPU based DataFrame library with a pandas-like API
- **Dask**, providing a framework for scalable computing
  - One centralized dask scheduler
  - One or multiple dask workers
- **Dask-cuDF**, scaling out cuDF DataFrame partitions across multiple GPUs
  - A dask-cuda cluster
    - one dask scheduler
    - one or multiple dask-cuda workers
  - A client connects to the dask-cuda cluster

Create & Connect to a Dask-CUDA cluster

- On a multi-gpu node

```
from dask_cuda import LocalCUDACluster
from distributed import Client

cluster = LocalCUDACluster()
client = Client(cluster)
```

- More options with UCX integration via UCX-py

```
cluster = LocalCUDACluster(protocol="ucx", enable_tcp_over_ucx=True)
```

```
cluster = LocalCUDACluster(protocol="ucx", enable_tcp_over_ucx=True,
         enable_nvlink=True, enable_infiniband=True)
```

Using RAPIDS on clusters

- RAPIDS Installation
  - Conda, Build from source, Docker containers
  - Build a Singularity/Apptainer container for RAPIDS

- Working with a Singularity/Apptainer container
  - Docs wiki: https://docs.alliancecan.ca/wiki/Apptainer
  - Docs wiki: https://docs.alliancecan.ca/wiki/RAPIDS
  - Ask help: help@sharcnet.ca or support@tech.alliancecan.ca

Demo example

Data transformations with a User Defined Function (UDF)

- Case 1: using Pandas and Numpy on CPU
- Case 2: using cuDF and Cupy on a single GPU
- Case 3: using Dask-cuDF on multiple GPUs
  - Data can't fit on a single GPU
  - Creating a LocalCUDACluster with multiple GPUs

RAPIDS container

- rapidsai-22.12-cuda11.4-runtime-ubuntu20.04-py3.9

GPU resources on general purpose clusters

Cluster	GPU nodes	GPUs / node	GPU type	GPU mem	Total GPUs
Narval	159	4	A100	**40GB**	636
Beluga	172	4	V100	**16GB**	688
Cedar	146	4	P100	**12GB** or **16GB**	1352
	192	4	V100	**32GB**	
Graham	160	2	P100	**12GB**	490
	7	8	V100	**16GB or 32GB**	
	30	4	T4	**16GB**	

https://docs.alliancecan.ca/wiki/Using_GPUs_with_Slurm

Work interactively on a GPU node

- Request an interactive session on a GPU node, i.e. using salloc with a GPU request

```
[user@gra-login]$ salloc --time=1:0:0 --account=def-someuser --cpus-per-task=4 --mem=16G --gres=gpu:t4:1
[user@gra####]
```

- Activate RAPIDS via the container on a GPU node

```
[user@gra####] module load apptainer/1.0
[user@gra####] apptainer shell --nv -B /home -B /project -B /scratch rapids.sif
Apptainer> souce /opt/conda/etc/profile.d/conda.sh
Apptainer> conda activate rapids
(rapids) Apptainer>
```

- Check GPU status

```
(rapids) Apptainer> nvidia-smi
```

Work interactively on a GPU node

- Launch Jupyter server in container shell

```
(rapids) Apptainer> jupyter-lab --no-browser --ip $(hostname -f)
... ...
    http://<node-hostname>:8888/lab?token=109c369980ab346bcd129eb...
```

- Setup a SSH tunnel on a second terminal from your local computer

```
[you@local]$ ssh -L 9999:<node-hostname>:8888 <user>@<cluster>.computecanada.ca
[user@cluster-login]$
```

- The URL to connect on a local web browser would be:

http://localhost:9999/lab?token=109c369980ab346bcd129eb..

Submit a batch job

- Prepare a job submission script, e.g. *submit_job.sh*

```
#! /bin/bash
#SBATCH --time=<dd:hh:mm>
#SBATCH --account=def-<someuser>
#SBATCH --cpus-per-task=<##>
#SBATCH --mem=<####>
#SBATCH --gres=gpu:<type>:<#>

module load apptainer
apptainer exec --nv -B <host fs> rapids.sif /path/to/run_job.sh
```

- Submit job with *sbatch*

```
[user@gra-login]$ sbatch submit_job.sh
```

- Prepare a job execution script, e.g. *run_job.sh*

```
#! /bin/bash

source /opt/conda/etc/profile.d/conda.sh
conda activate rapids
nvidia-smi

python /path/to/my_rapids_code.py
```

Note: make *run_job.sh* an executable file

```
[user@gra-login]$ chmod +x run_job.sh
```