Hands-on: part IV

Connect to Dardel - Kerberos

Before connecting to Dardel, the local environment of your laptop must be configured.

Detailed instructions can be found on https://www.pdc.kth.se/support/documents/login/configuration.html.

•If you are using **Mac,** you need to follow a special set of instructions here: https://www.pdc.kth.se/support/documents/login/mac_login.html.

•If you are using **Windows**, we recommend using the WSL2 to use a fully-fledged Ubuntu: https://www.pdc.kth.se/support/documents/login/windows_login.html#wsl-approach.

To configure Kerberos on your local machine, you need to do the following steps:

1) Create .ssh folder if it does not already exist in your home folder.

2) Create a file called krb5.conf in .ssh with the following content

Connect to Dardel - SSH

1. See https://www.pdc.kth.se/support/documents/login/configuration.html
2. Create a file in .ssh called config with the following content

```
# Hosts we want to authenticate to with Kerberos
Host *.kth.se *.kth.se.
# User authentication based on GSSAPI is allowed
GSSAPIAuthentication yes
# Key exchange based on GSSAPI may be used for server authentication
GSSAPIKeyExchange yes
# Hosts to which we want to delegate credentials. Try to limit this to
# hosts you trust, and were you really have use for forwarded tickets.
Host *.csc.kth.se *.csc.kth.se. *.nada.kth.se *.nada.kth.se. *.pdc.kth.se *.pdc.kth.se.
# Forward (delegate) credentials (tickets) to the server.
GSSAPIDelegateCredentials yes
# Prefer GSSAPI key exchange
PreferredAuthentications gssapi-keyex,gssapi-with-mic
# All other hosts
Host *
```

2. Set the correct permission on the file

```
$ chmod 644 ~/.ssh/config
```

3. Connect

```
$ ssh YourUsername@dardel.pdc.kth.se
```

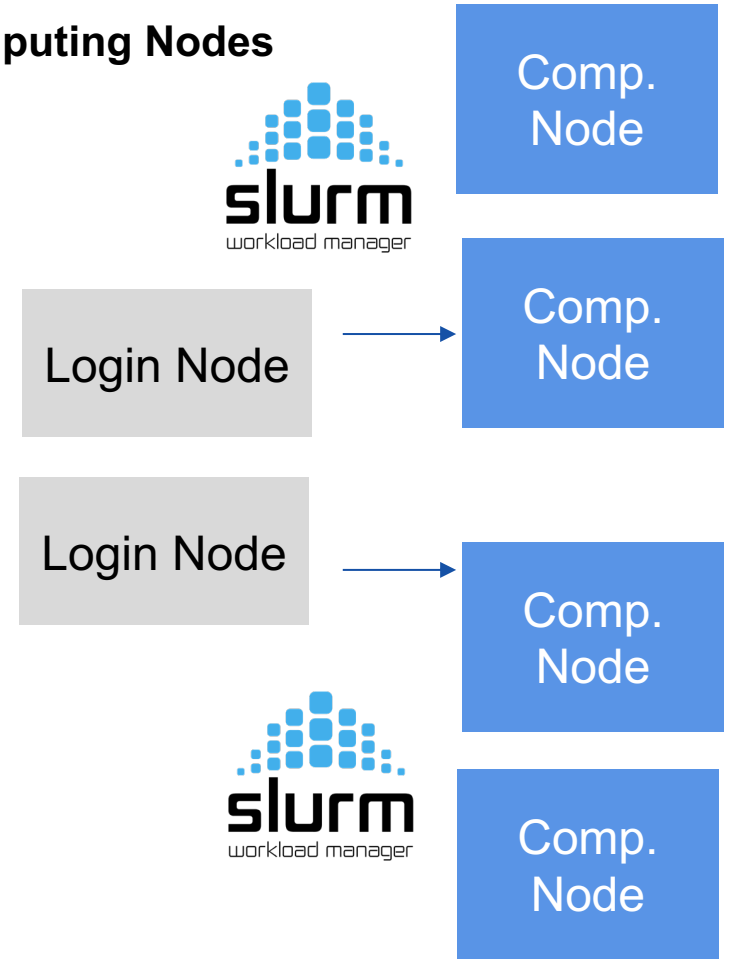
Use Slurm to Launch Your Simulation on Computing Nodes

- Slurm allows us to launch our simulation on the computing nodes
- Two ways:
  - **Interactive mode:** ask for a resource, and as soon it becomes available, we can launch your simulations interactively

```
$ salloc --nodes=1 -t 00:30:00 -A edu23.summer -p gpu --ntasks-per-node=1 --cpus-per-task=1
```

```
$ srun -n 1 ./hello.out
```

  - **Batch mode:** put your simulation into a queue



Profiling with rocprof

Profiling Reduction with rocprof

Run rocprof with –-stats for a summary in *results.stats.csv*

```
4_reduction> srun -n 1 rocprof --stats ./reduction
RPL: on '230811_145940' from '/cfs/klemming/root/rocm/opt/rocm-5.3.3' in './amd_part2/4_reduction'
RPL: profiling '"./reduction"'
RPL: input file ''
RPL: output dir '/tmp/rpl_data_230811_145940_117170'
RPL: result dir '/tmp/rpl_data_230811_145940_117170/input_results_230811_145940'
Usage: ./reduction num_of_elems
using default value: 52428800
ARRAYSIZE: 52428800
Array size: 200 MB
ROCProfiler: input from "/tmp/rpl_data_230811_145940_117170/input.xml"
  0 metrics
The average performance of reduction is 396.195 GBytes/sec
VERIFICATION: result is CORRECT

ROCPRofiler: 20 contexts collected, output directory /tmp/rpl_data_230811_145940_117170/input_results_230811_145940
File '/amd_part2/4_reduction/results.csv' is generating
File '/amd_part2/4_reduction/results.stats.csv' is generating
```

Profiling Reduction with rocprof

Run rocprof with a selected list of hardware counters specified in *my_counters.txt*

```
4_reduction> cat my_counters.txt
pmc: Wavefronts VALUInsts VFetchInsts VWriteInsts VALUUtilization VALUBusy WriteSize


4_reduction> srun -n 1 rocprof -i my_counter.txt ./reduction
…

ROCProfiler: input from "/tmp/rpl_data_230810_164433_16749/input0.xml"
  gpu_index =
  kernel =
  range =
  7 metrics
    Wavefronts, VALUInsts, VFetchInsts, VWriteInsts, VALUUtilization, VALUBusy, WriteSize

The average performance of reduction is 91.2052 GBytes/sec
VERIFICATION: result is CORRECT

ROCPRofiler: 20 contexts collected, output directory /tmp/rpl_data_230810_164433_16749/input0_results_230810_164433
File '4_reduction/my_counter.csv' is generating
```

Profiling Reduction with rocprof

Inspect the output *my_counters.csv*

```
4_reduction> cat my_counters.csv

Index,KernelName,gpu-id,queue-id,queue-
index,pid,tid,grd,wgr,lds,scr,arch_vgpr,accum_vgpr,sgpr,wave_size,sig,obj,Wavefronts,VALUInsts,VFetchInsts,VWriteInsts,VALUUtiliza
tion,VALUBusy,WriteSize
……
..
```

Question: post your output to our Slack channel

Matrix Multiplication in HIP

Matrix Multiplication in HIP

1. Compile and run the example matrix multiplication code on the Dardel supercomputer

```
5_MatrixMultiplication> make
hipcc --offload-arch=gfx90a -I./include   -c -o MatrixMultiplication.o MatrixMultiplication.cpp
hipcc  MatrixMultiplication.o -o MatrixMultiplication
```

Matrix Multiplication in HIP

2. Compile and run the example matrix multiplication code on the Dardel supercomputer

```
5_MatrixMultiplication> srun -n 1 ./MatrixMultiplication -h
Usage
-h, --help                                  Display this information
-q, --quiet                                 Quiet mode. Suppress all text output.
-e, --verify                                Verify results against reference implementation.
-t, --timing                                Print timing.
-v, --version                               AMD APP SDK version string.
-d, --deviceId                              Select deviceId to be used[0 to N-1 where N is number devices available].
-x, --height0                               height of matrix A
-y, --width0                                width of matrix A and Height of matrix B
-z, --width1                                width of matrix B
-b, --blockSize                             Use local memory of dimensions blockSize x blockSize
-i, --iterations                            Number of iterations for kernel execution
    --eAppGflops                            Prints GFLOPS calculated from transfer + kernel time
```

Matrix Multiplication in HIP

2. Run the program with different input size for A and B

```
5_MatrixMultiplication> srun -n 1 ./MatrixMultiplication -q -t -x 1024 -y 1024 -z 1024 -i 10 --timing  --eAppGflops
```

Question 1: what is the maximum GFlops achieved?

Use rocBLAS library on AMD GPU

Use rocBLAS library on AMD GPU

1. compile and run the BLAS level 3 matrix-matrix multiplication example. Remember to load rocm module on the Dardel supercomputer

```
6_rocBLAS_gemm> module load rocm/5.3.3

6_rocBLAS_gemm> make
hipcc -std=c++14 -I./common -isystem/cfs/klemming/root/rocm/opt/rocm-5.3.3/include -
isystem/cfs/klemming/root/rocm/opt/rocm-5.3.3/include  -g -Ofast -march=native -Wall  -c
gemm.cpp -o gemm.o
hipcc -std=c++14 -I./common -isystem/cfs/klemming/root/rocm/opt/rocm-5.3.3/include -
isystem/cfs/klemming/root/rocm/opt/rocm-5.3.3/include  -g -Ofast -march=native -Wall  -c
common/ArgParser.cpp -o common/ArgParser.o
hipcc  gemm.o  ./common/ArgParser.o  -L/cfs/klemming/root/rocm/opt/rocm-5.3.3/lib -
L/cfs/klemming/root/rocm/opt/rocm-5.3.3/lib -lrocblas -Wl,-
rpath=/cfs/klemming/root/rocm/opt/rocm-5.3.3/lib -Wl,-rpath=/cfs/klemming/root/rocm/opt/rocm-
5.3.3/lib -lm -lpthread -lstdc++ -o 6_rocBLAS_gemm
```

Use rocBLAS library on AMD GPU

1. Try with different input and check the output (you might want to uncomment the CPU part to check the correctness of the results)
2.

```
6_rocBLAS_gemm
  --K <value>              Matrix/vector dimension
  --M <value>              Matrix/vector dimension
  --N <value>              Matrix/vector dimension
  --alpha <value>          Alpha scalar
  --beta <value>           Beta scalar

_rocBLAS_gemm> srun -n 1 ./6_rocBLAS_gemm --K 128 --M 128 --N 128 --alpha 0.25 --beta 0.75
```

Use rocBLAS library on AMD GPU

Exercise 1: convert your code in 5_MatrixMultiplication to use rocBLAS sgemm in this example.

Exercise 2: rerun the matrix multiplication using rocBLAS and compare your achieved Gflops with the original matrix multiplication implementation.

Q & A