GPUHackShef Documentation

Release

Mozhgan K. Chimeh

Jun 30, 2020

Contents

1 The JADE Facility **3**
- 1.1 Getting an account . . . 3

2 The Ascent Facility **5**
- 2.1 Getting an account . . . 5
- 2.2 Storage Areas . . . 5
- 2.3 Getting Started . . . 6

3 Useful Training Material **9**
- 3.1 Ascent Material . . . 9
- 3.2 General Training Material . . . 9

4 CUDA Profiling for HPC **11**
- 4.1 Compiler settings for profiling . . . 11
- 4.2 Nsight Systems and Nsight Compute . . . 11
- 4.3 Visual Profiler (nvprof and nvvp) . . . 12

The University Of Sheffield.

NVIDIA®

OpenACC

More Science, Less Programming

This is the documentation for the Sheffield GPU Hackathon.

You may find it useful to read the helpful GPU Hackathon Attendee Guide prepared by Oak Ridge.

The site encourages user contributions via GitHub and has useful information for how to access the GPU systems used for the Hackathon event.

CHAPTER 1

The JADE Facility

The JADE facility consists of 2 head nodes and 22 NVIDIA DGX-1 servers, each with 8 GPUs and 40 CPU cores. For the Hackathon event 2 nodes have been reserved for exclusive access.

Official Documentation Site

The official JADE documentation site is located at jade.ac.uk

1.1 Getting an account

If not already registered, you need to sign up with the Hartree Centre SAFE system providing an Institutional email address, which will be used as the SAFE login ID.

Note: emails such as gmail or hotmail will not be accepted

At this stage, in order to access the machine, you also need to enter an SSH key. Instructions on how to create this key and add it to your account details are at: SAFE User Guide

An email will be sent to you containing a password for SAFE which needs to be changed on first logging in. Once you have signed up, log on to SAFE and click on 'Request Join Project'. From the dropdown list choose JAD044 and enter the Project signup password which, for this Project, is jadetraining44, and click 'Request'.

Accounts on the system for your project will then be created for you, please refer to the official documentation on how to use the system.

If you encounter any problems do let me know. Alternatively, raise an issue on GitHub or contact one of the hackathon organisers.

CHAPTER 2

The Ascent Facility

The Ascent training system, is an 18-node, stand-alone compute system with the same node architecture as Summit. As stated in the hackathon attendee guide, we recommend that all participants obtain access to the system before the event.

2.1 Getting an account

In order to obtain an ascent account please follow instructions on the link provided and use the project ID GEN132.

You should use Thomas Papatheodore as the name for the project PI.

Accessing Ascent

2.2 Storage Areas

The Ascent has the following storage areas available.

2.2.1 NFS Directories

This is where you might want to keep source code and build your application.

NOTE: These directories are read-only from the compute nodes!

```
/ccsopen/home/userid
```

Your personal home directory

```
/ccsopen/proj/gen132
```

Can be accessed by all participants of this event

You can create a directory here with your application name to collaborate with others (source code, scripts, etc.)

2.2.2 GPFS Directories (parallel file system)

This is where you should write data when running on Ascent's compute nodes.

```
/gpfs/wolf/gen132/scratch/userid
```

Your personal GPFS scratch directory

```
/gpfs/wolf/gen132/proj-shared
```

Can be accessed by all participants of the event

You can create a directory here with your application name to collaborate with others (data written from compute nodes)

2.3 Getting Started

Once you have access to Ascent, log in

```
ssh USERNAME@login1.ascent.olcf.ornl.gov
```

From your `/ccsopen/home/userid` directory:

```
git clone https://code.ornl.gov/t4p/Hello_jsrun.git
cd Hello_jsrun
```

Load CUDA Toolkit

```
module load cuda
```

Compile the code

```
make
```

Grab a Single Node in an Interactive Job Using LSF and load the CUDA Toolkit

```
bsub -P GEN132 -nnodes 1 -W 30 -alloc_flags "gpumps" -Is /bin/bash
```

GEN132 is the project ID provided for the Hackathon.

The `-alloc_flags "gpumps"` enables the MPS server - essentially allowing multiple processes (e.g., MPI ranks) to access the same GPU concurrently. Whether or not to include it depends on your application, but it is included here since (when testing different layouts) you might attempt to target the same GPU with multiple MPI ranks. If you do so without MPS, this program will hang.

Launch a Simple Job Configuration with jsrun by setting the number of OpenMP threads to `1` (for simplicity)

```
export OMP_NUM_THREADS=1
```

Launch the job

```
jsrun -n6 -c1 -g1 ./hello_jsrun | sort
###########################################################################
*** MPI Ranks: 6, OpenMP Threads: 1, GPUs per Resource Set: 1 ***
===========================================================================
MPI Rank 000, OMP_thread 00 on HWThread 000 of Node h49n16 - RT_GPU_id 0 : GPU_id 0
```

```
MPI Rank 001, OMP_thread 00 on HWThread 005 of Node h49n16 - RT_GPU_id 0 : GPU_id 1
MPI Rank 002, OMP_thread 00 on HWThread 009 of Node h49n16 - RT_GPU_id 0 : GPU_id 2
MPI Rank 003, OMP_thread 00 on HWThread 089 of Node h49n16 - RT_GPU_id 0 : GPU_id 3
MPI Rank 004, OMP_thread 00 on HWThread 093 of Node h49n16 - RT_GPU_id 0 : GPU_id 4
MPI Rank 005, OMP_thread 00 on HWThread 097 of Node h49n16 - RT_GPU_id 0 : GPU_id 5
```

2.3.1 Summit Node Diagram

Use the Summit node diagram to understand how your MPI ranks are being assigned to the hardware.

Also, see the Hardware Threads section of the Summit User Guide to understand the difference between the 42 physical coresper node versus the 4 hardware threads per physical core.

CHAPTER 3

Useful Training Material

The following list is a useful catalogue of training material.

3.1 Ascent Material

- Summit User Guide
- Ascent User Guide – This is actually just a small addition to the Summit User Guide that outlines the main differences between the two systems.

Tools to learn how to use the *jsrun* job launcher

- Hello_jsrun – A "Hello, World!" type program to help understand resource layouts on Summit/Ascent nodes.
- jsrunVisualizer – A web-based tool to learn the basics of *jsrun*.
- Jsrun Quick Start Guide

3.2 General Training Material

- OpenACC Online Course (NVIDIA)
- OLCF Training Archive (from past OLCF training events)
- OLCF Tutorials
- Sheffield One Day CUDA Course
- Sheffield Two Day CUDA Course

CHAPTER 4

CUDA Profiling for HPC

When using HPC systems it can be difficult to attach profiling GUIs for interactive profiling. Instead it is best to profile remotely capturing as much data as possible and viewing profile information locally.

CUDA profiling tools offer timeline-style tracing of GPU activity, plus detailed kernel level information.

Newer CUDA versions and GPU architectures can use Nsight Systems and Nsight Compute. All current CUDA versions and GPU architectures support Nvprof and the Visual Profiler.

4.1 Compiler settings for profiling

Compile with `-lineinfo` (or `--generate-line-info`) to include source-level profile information. Do not profile debug builds, it will not be representative.

4.2 Nsight Systems and Nsight Compute

Note: Requires CUDA >= 10.0 and Compute Capability > 6.0

Trace application timeline using remote Nsight Systems CLI (`nsys`):

```
nsys profile -o timeline ./myapplication
```

Import into local Nsight Systems GUI (`nsight-sys`):

- `File > Open`
- Select `timeline.dqrep`

Capture Kernel level info using remote Nsight CUDA CLI (`nv-nsight-cu-cli`):

```
nv-nsight-cu-cli -o profile ./bin/Release/atomicIncTest
```

Import into local Nsight CUDA GUI `nv-nsight-cu` via:

```
nv-nsight-cu profile.nsight-cuprof-report
```

Or Drag `profile.nsight-cuprof-report` into the `nv-nsight-cu` window.

4.2.1 Documentation

- Nsight Systems
- Nsight Compute

4.3 Visual Profiler (nvprof and nvvp)

Note: Suitable for all current CUDA versions and GPU architectures

Trace application timeline remotely using `nvprof`:

```
nvprof -o timeline.nvprof ./myapplication
```

Capture kernel-level metrics remotely using `nvprof` (this will take a while):

```
nvprof --analysis-metrics -o analysis.nvprof ./myapplication
```

Import into the Visual Profiler GUI (`nvvp`)

- `File > Import`
- Select `Nvprof`
- Select `Single process`
- Select `timeline.nvvp` for `Timeline data file`
- Add `analysis.nvprof` to `Event/Metric data files`

4.3.1 Documentation

- Nvprof Documentation