

Lab3 CUDA Basic

Nov, 2021 Parallel Programming

Overview

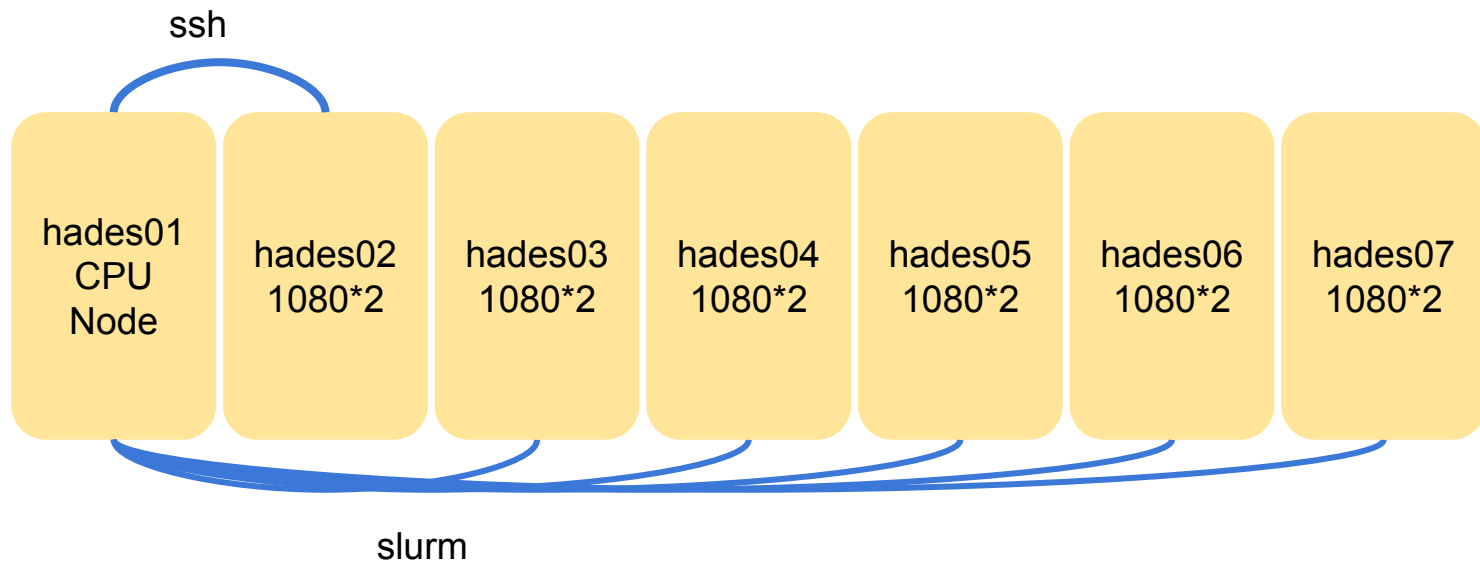
- ❖ Platform guide
- ❖ Tools
- ❖ Assignment

Platform Guide

The GPU Cluster

- ❖ Host: `hades.cs.nthu.edu.tw`
- ❖ Account: same as `apollo`
- ❖ Password: same as `apollo`

The GPU Cluster



Job Scheduler

- ❖ SLURM

- ❖ Course partition: pp21

```
[[pp21t00@hades01 ~]$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
pp21*	up	5:00	1	drain	hades04
pp21*	up	5:00	4	idle	hades[03,05-07]

- ❖ Limitation

- 2 gpus
- 5 minutes

Access Resource

❖ hades02

- `ssh hades02`
- If you want to specify which GPU to use.
- `export CUDA_VISIBLE_DEVICES=<gpu id>`
- eg. `export CUDA_VISIBLE_DEVICES=1`
- eg. `export CUDA_VISIBLE_DEVICES=0,1`

❖ hades[03-07]

- Slurm
- Access gpus with flag `--gres=gpu:<number of gpu>`
- eg. `srun -n 1 --gres=gpu:1 ./executable`
- eg. `srun -n 1 --gres=gpu:2 ./executable`
- Two GPUs will be executed on the same node.

Compile & run

❖ Compiler nvcc

- `nvcc [options] <inputfile>`
- `eg. nvcc cuda_code.cu -o cuda_executable`

❖ Run

- Refer to the previous slide

Practice

- ❖ In this practice, you can try to run the **deviceQuery** on
 - hades02
 - scheduler
- ❖ Compile
 - `cp -r /home/pp21/share/lab3/deviceQuery $HOME`
 - `cd $HOME/deviceQuery`
 - `nvcc deviceQuery.cpp -o deviceQuery`
- ❖ Run it on
 - hades02
 - scheduler
- ❖ How many CUDA cores on GTX1080 ?

Tools

nvidia-smi

- ❖ NVIDIA System Management Interface program
- ❖ You can query details about
 - gpu type
 - gpu utilization
 - memory usage
 - temperature
 - clock rate
 - ...

nvidia-smi example

```
# michael1017 @ hades02 in ~ [15:08:34]
```

```
$ nvidia-smi
```

Thu Nov 12 15:08:36 2020

NVIDIA-SMI 450.57			Driver Version: 450.57			CUDA Version: 11.0		
GPU Name			Persistence-M		Bus-Id	Disp.A	Volatile	Uncorr. ECC
Fan	Temp	Perf	Pwr:Usage/Cap		Memory-Usage		GPU-Util	Compute M.
							MIG M.	
0	GeForce	GTX 1080	On		00000000:4B:00.0	Off		N/A
0%	37C	P8	7W / 200W		1MiB / 8119MiB		0%	Default
							N/A	
1	GeForce	GTX 1080	On		00000000:4D:00.0	Off		N/A
0%	44C	P8	14W / 200W		1MiB / 8117MiB		0%	Default
							N/A	
Processes:								
GPU	GI	CI	PID	Type	Process name			GPU Memory
	ID	ID						Usage
No running processes found								

cuda-memcheck

- ❖ This tool **checks memory errors** of your program, and it also reports hardware exceptions encountered by the GPU. These errors may not cause program to crash, but they could result in unexpected program behavior and memory misuse.
- ❖ Error types
 - [cuda-memcheck](#)

cuda-memcheck

```
cudaFree(device_t);  
cudaFree(device_t); // free an address twice, error
```

```
[mewtwo@hades02 HW4_cuda_sobel]$ cuda-memcheck ./sobel input/candy.bmp out.bmp  
===== CUDA-MEMCHECK  
===== Program hit cudaErrorInvalidDevicePointer (error 17) due to "invalid device pointer" on CUDA API call to cudaFree.  
===== Saved host backtrace up to driver entry point at error  
===== Host Frame:/usr/lib64/nvidia/libcuda.so.1 [0x32f6a3]  
===== Host Frame:./sobel [0x454c0]  
===== Host Frame:./sobel [0x356f]  
===== Host Frame:/usr/lib64/libc.so.6 (__libc_start_main + 0xf5) [0x21b35]  
===== Host Frame:./sobel [0x36ff]  
=====  
===== ERROR SUMMARY: 1 error
```

cuda-gdb

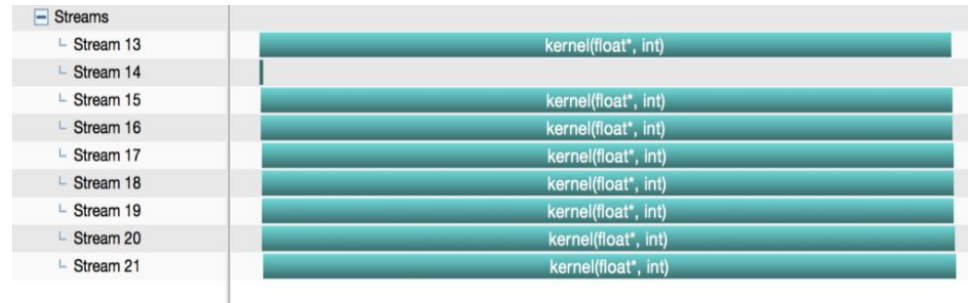
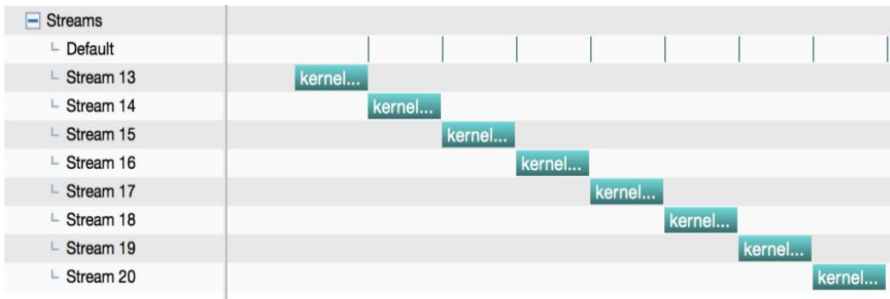
❖ [cuda-gdb tutorial](#)

nvprof

- ❖ A CUDA profiler provides feedback to optimize CUDA programs
 - `nvprof ./lab3 in.png out.png`
 - `-o <FILE>` to save result to a file
 - `-i <FILE>` to read result from a file

nvvp

- ❖ [nvvp-tutorial](#)
- ❖ GUI version of nvprof
- ❖ Useful for the stream optimization
 - Timeline

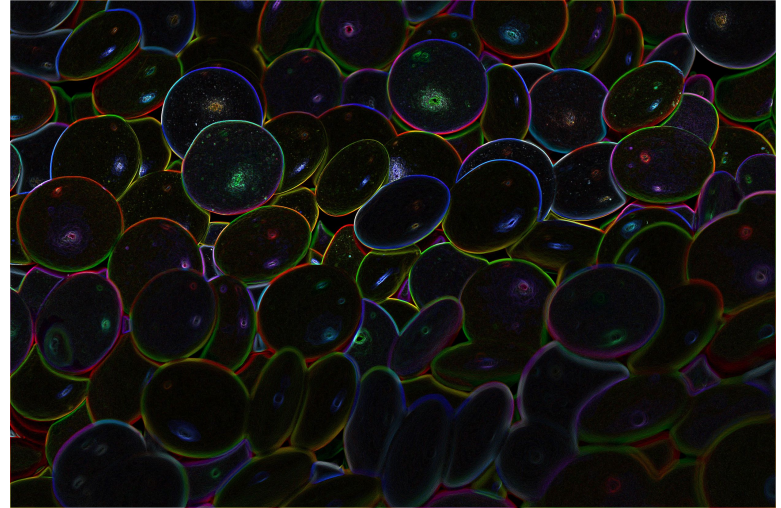


nvvp is useful for checking the concurrency of stream

Lab3 Assignment

Problem Description

- ❖ Edge Detection: Identifying points in a digital image at which the image brightness changes sharply



Sobel Operator

- ❖ Used in image processing and computer vision, particularly within edge detection algorithms.
- ❖ Uses two 3x3 kernels g_x , g_y which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical.

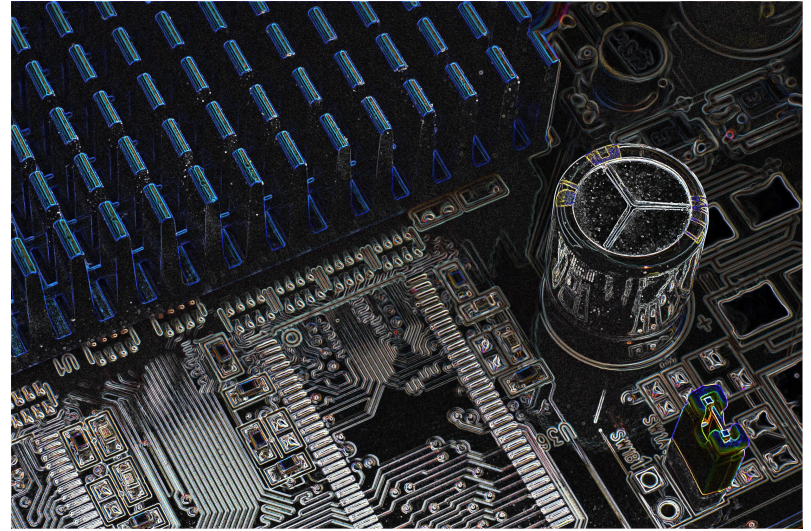
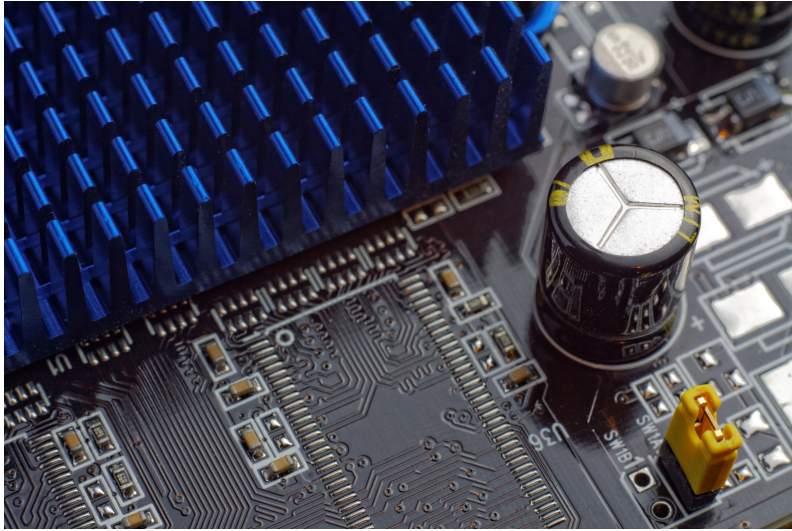
5x5 Variation

- ❖ We use this kernel instead of the 3x3 one in this lab

$$g_x = \begin{pmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 6 & 12 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{pmatrix},$$

$$g_y = \begin{pmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

Sample Result



Preparation

- ❖ TA provides CPU version, Makefile, and hint
- ❖ File located at `/home/pp21/share/lab3`
- ❖ Please do not copy the testcases.
- ❖ `lab3.cu` is cpu version (you need to rewrite it with cuda!)
- ❖ You can follow hints to write

How to run

❖ **hades02**

- `./lab3 <input> <output>`
- `CUDA_VISIBLE_DEVICES=0 ./lab3 <input> <output>`

❖ **hades[03-07]**

- `srun -n 1 ./lab3 <input> <output>`
- `srun -n 1 --gres=gpu:1 ./lab3 <input> <output>`

❖ **Compare your result with the answer**

- `png-diff <output image> <answer image>`

Hints

- ❖ Malloc memory on GPU
- ❖ Copy original image to GPU
- ❖ Put filter matrix on device memory (or declare it on device)
- ❖ Copy filter matrix to shared memory
(don't let only one thread do it)
- ❖ Parallel the sobel computing
- ❖ Copy the results from device to host
- ❖ Free unused address

Submission

- judge will execute your code with single process, single GPU
- submit your code and Makefile (optional) to eeclass before 11/18 23:59
- use lab3-judge