Lab5: CUDA - Basic

助教: 鄭禎<zzchman@gmail.com>

2017.07.25

Outline

1. Platform Guide
2. Tools
3. Coding Assignment

The GPU Server

1. ssh to 166.111.68.163
   $ ssh nthu@166.111.68.163 **–p 2222**
   Password: nthu123

2. ssh to 140.114.91.187
   Username and password is the same as previous labs.

Compile & run

Compile

- `nvcc [options] input_file`
- Example: `nvcc -o executable code.cu`
- For more options, please see `nvcc –help`

Run

- `./executable [args]`

Examples in /home/share/

```
$ cp   /home/share/NVIDIA_CUDA-8.0_Samples.tar  ~
$ tar  –xf  NVIDIA_CUDA-8.0_Samples.tar
```

Outline

1. Platform Guide
2. Tools
3. Coding Assignment

nvidia-smi

Purpose: Query and modify GPU's state

You can query details about

- device type
- clock rate
- temperature
- power
- memory
- …

nvidia-smi: example

```
[root@pp31 ~]# nvidia-smi
Sun Dec  4 15:49:05 2016
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.48                 Driver Version: 367.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla C2070         Off  | 0000:03:00.0     Off | 6066MiB          Off |
| 30%   52C    P0    N/A /  N/A |      0MiB /  |              0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla M2090         Off  | 0000:06:00.0     Off | 6066MiB          Off |
| N/A   N/A    P0    75W /  N/A |      0MiB /  |              0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

nvidia-smi: example

```
[root@pp31 ~]# nvidia-smi -q -d CLOCK

==============NVSMI LOG==============

Timestamp Driver                    : Sun Dec  4 15:53:09 2016
Version Attached                    : 367.48
GPUs GPU                            : 2
0000:03:00.0
    Clocks
        Graphics                    : 573 MHz
        SM                          : 1147 MHz
        Memory                      : 1494 MHz
        Video                       : 540 MHz
    Applications Clocks
        Graphics Memory             : N/A
    Default Applications Clocks     : N/A
        Graphics
        Memory                      : N/A
    Max Clocks                      : N/A
        Graphics
        SM                          : 573 MHz
        Memory                      : 1147 MHz
        Video                       : 1494 MHz
                                    : 540 MHz
```

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 misusage.

cuda-memcheck

Some erroneous code

```
        cudaFree (d_data); cudaFree
        (d_data); // error
    return 0;
}
```

Error summary

```
[user0@gpucluster0 shared]$ cuda-memcheck sobel candy.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:/lib64/libcuda.so.1 [0x2e4263]
=========     Host Frame:sobel [0x3dcb6]
=========     Host Frame:sobel [0x27b1]
=========     Host Frame:/lib64/libc.so.6 (_ libc_start_main + 0xf5) [0x21af5]
=========     Host Frame:sobel [0x287d]
=========
========= ERROR SUMMARY: 1 error
```

cuda-memcheck error types

Name	Description	Location	Precision
Memory access error	Errors due to out of bounds or misaligned accesses to memory by a global, local, shared or global atomic access.	Device	Precise
Hardware exception	Errors that are reported by the hardware error reporting mechanism.	Device	Imprecise
Malloc/Free errors	Errors that occur due to incorrect use ofmalloc()/free() in CUDA kernels.	Device	Precise
CUDA API errors	Reported when a CUDA API call in the application returns a failure.	Host	Precise
cudaMalloc memory leaks	Allocations of device memory using cudaMalloc()that have not been freed by the application.	Host	Precise
Device Heap Memory Leaks	Allocations of device memory using malloc() in device code that have not been freed by the application.	Device	Imprecise

cuda-gdb

Similar to GDB

A tool provides developers with a mechanism for debugging CUDA application running on actual hardware.

For more details, please refer to
cuda-debugging-tools.pdf

cuda-gdb: print/set variables

Print variable

```
(cuda-gdb) print total
$1 = 11.1110363
```

Reassign value to variable

```
(cuda-gdb) print total = 31.1095
$2 = 31.109499
```

cuda-gdb: breakpoint

by kernel name

```
(cuda-gdb) break sobel_Kernel
```

by file & line number

```
(cuda-gdb) break test.cu:149
```

by address

```
(cuda-gdb) break 0x4e15f73
```

cuda-gdb: execution control

Launch application (with arguments)

```
(cuda-gdb) run arg1 arg2
```

Resume execution

```
(cuda-gdb) continue
```

Kill the program

```
(cuda-gdb) kill
```

cuda-gdb: execution control

Interrupt the program

- Ctrl + C

Single stepping

	At source level	At assembly level
Over function calls	next	nexti
Into function calls	step	stepi

nvprof

A CUDA profiler

Provides feedback to optimize CUDA programs

--metrics <METRIC_NAME> to measure specific metrics

--events <EVENT_NAME> to record specific events

-o <FILE> to save result to a file

-i <FILE> to read result from a file

nvvp

nvprof’s GUI counterpart

easier to use

Outline

1. Platform Guide
2. Tools
3. Coding Assignment

Coding Assignments

Given 3 arrays: A, B, D = {1, 2, …, 10}

Task#1: C = A^A + B^B

Task#2: C = A + B, F = D + C

- ✓ Requirement:
  - Implement task#1 with single stream
  - Implement task#2 with 2 streams to overlap the data transfer data of D
  - C and F must be copied back to host for correctness check
  - Report the kernel time & total execution time
  - `Pass` cuda-memcheck
  - If you can't complete today, make another appointment with TA

Steps to follow

1. Initialize CUDA device
2. Allocate memory in device & put sequential code into kernel function
3. Relabel index variables with combinations of `threadIdx, blockIdx, blockDim, gridDim`
4. Optimizations (requires great deal of effort!)

Vector Add

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
     int i = threadIdx.x;
     C[i] = A[i] + B[i];
}

int main()
{
     // Device memory allocation
     // Host to Device memory copy
     // Kernel invocation with N threads
     // Device to Host memory copy
}
```

Device memory operations

Three functions:

`cudaMalloc(), cudaFree(), cudaMemcpy()`

Similar to the C's **`malloc(), free(), memcpy()`**

1. `cudaMalloc(void **devPtr, size_t size)`

   `devPtr`: return the address of the allocated device memory

   `size`: the allocated memory size (**bytes**)

2. `cudaFree (void *devPtr)`

3. `cudaMemcpy( void *dst, const void *src, size_t count, enum cudaMemcpyKind kind)`

   `count`: size in **bytes** to copy

cudaMemcpyKind

one of the following four values

cudaMemcpyKind	Meaning	dst	src
cudaMemcpyHostToHost	Host → Host	host	host
cudaMemcpyHostToDevice	Host → Device	device	host
cudaMemcpyDeviceToHost	Device → Host	host	device
cudaMemcpyDeviceToDevice	Device → Device	device	device

host to host has the same effect as memcpy()

How to measure kernel execution time?

cudaEventCreate(): Init timer

cudaEventDestroy(): Destroy timer

cudaEventRecord(): Set timer

cudaEventSynchronize(): Sync timer after each kernel call

cudaEventElapsedTime(): Returns the elapsed time in milliseconds

How to measure kernel execution time?

```
cudaEvent_t start, stop;
float time;

cudaEventCreate (&start);
cudaEventCreate (&stop);

cudaEventRecord (start, 0);
kernel <<< grid, threads >>> (d_in, d_out);
cudaEventRecord (stop, 0);
cudaEventSynchronize (stop);

cudaEventElapsedTime (&time, start, stop);
fprintf (stderr, "%lf\n", time);

cudaEventDestroy (start);
cudaEventDestroy (stop);
```

Multiple Streams

Different streams may execute their commands **out of order** with respect to one another or concurrently

```
cudaStream_t stream[2];
cudaStreamCreate(&stream[0]);
cudaStreamCreate(&stream[1]);
cudaMallocHost(&hostPtr, 2 * size); // pined(page locked mem)
for (int i = 0; i < 2; ++i) {
      cudaMemcpyAsync(/*…*/,        // async memcpy
            cudaMemcpyHostToDevice, stream[i]);
      kernel<<<100,512,0,stream[i]>>>(/*…*/);
      cudaMemcpyAsync(/*…*/,
            cudaMemcpyDeviceToHost, stream[i]);
}
cudaStreamDestroy(stream[0]);
cudaStreamDestroy(stream[1]);
```

References

CUDA C Programming Guide

- http://docs.nvidia.com/cuda/cuda-c-programming-guide/

CUDA Toolkit Documentation

- http://docs.nvidia.com/cuda/