GPU Programming Basics

Northeastern University *for all* Julian Gutierrez
NUCAR Laboratory David Kaeli

Hands-on Lab #3

Objective

- Optimize an implementation of a simple algorithm through pinned memory.
- Study the impact of Dynamic Parallelism on an algorithm.

NOTE: Unless stated otherwise, work in pairs.

Part 1: Pinned Memory

Copy the following folder to your home directory (or folder of your choosing):

```
/scratch/gutierrez.jul/GPUClass/HOL3/DELIVERABLE/
```

The code inside pinned folder is the same one from the previous hands on lab. The idea is to modify this code to use pinned memory instead of regular C memory allocation. The following are the templates for the commands you need:

```
cudaError_t cudaMallocHost ( void** ptr, size_t size )
```

```
cudaError_t cudaFreeHost ( void* ptr )
```

To compile the code:

```
nvcc -arch=sm_35 -O3 Nvprof.cu -o Nvprof
```

To execute the code:

```
./Nvprof 100000000
```

To profile the code:

```
nvprof ./Nvprof 100000000
```

Compare the performance of the code with and without pinned memory. Run nvprof to measure the actual changes between them and answer the following questions.

- What is the most impacted operation?
- Is this beneficial?
- Test it for different sized vectors (as in the following table) (NOTE: Test 1B only if professor allows you to):

Vector Size	Block Size	CPU Time	GPU Time (basic)	GPU Time (improved)
100	512			
100 000	512			
100 000 000	512			
1 000 000 000	512			

- What can happen if the RAM memory on the CPU is full with pinned memory?

- What happens if you don't perform the proper freeing of memory?
- Run nvprof to recollect the metrics and compare the bandwidth metrics. Any changes?

Part 2: Dynamic Parallelism

In this part we will explore how to implement a program using Dynamic Parallelism. In order to do so, look at the baseline code found on the dp folder.

1. Notice the size of all the data. When passing the number of matrices and matrix size to the program, we have to be careful of how much data can actually fit into the GPU.
2. Notice certain parts of the code that have omp on them.
    a. What is OMP?
    b. What do these code snippets do?
3. Compile the code as it is:
   ```
   nvcc -arch=sm_35 -O3 Baseline.cu -o Baseline
   ```
4. and run it using this command (100 matrices, with size 3000x3000):
   ```
   ./Baseline 100 3000
   ```
5. Look at the time it takes to run each stage.
6. Compile with omp flags.
   ```
   nvcc -arch=sm_35 -O3 Baseline.cu -o Baseline -Xcompiler "-fopenmp"
   ```
    a. What does -Xcompiler do?
    b. What does -fopenmp do?
7. Rerun the code. Did it Help? If so, where, why?
8. Why do we want to mix OMP and CUDA together?
9. Copy file and name it Modified.cu
   ```
   cp Baseline.cu Modified.cu
   ```
10. Write a kernel function to implement the second loop.
    a. Use same block_size as the others.
    b. Replace the for loop with the new kernel invocation in the calculation kernel.
    c. Think of what you need to pass to the kernel function.
    d. Compile the code.
       ```
       nvcc -arch=sm_35 -O3 Modified.cu -o Modified -Xcompiler "-fopenmp"
       ```
    e. It should give a compilation error. Why could this be?
    f. If you use the flag -rdc=true, does that fix it? What does -rdc do?
11. Compare results to the original implementation (Use the table below as guideline).
    a. Think of the advantages of using DP.
    b. Are there any scenarios where it performs better than the sequential (compare kernel time)?
    c. **IMPORTANT:** What is the impact of the number of matrices with DP. Compare the results when changing from 100 to 1000, and from 2000 to 5000. Is there a sweet spot?
    d. **Find the reason for all the results.**
    e. How does the block size affect? Try changing it to find a better one.

f. What are your conclusions for Dynamic Parallelism.
g. How would it compare if all the loops were assigned as one single kernel call, and the if statements inside it?
    i. Optional. Implement it.

12. Optional: Write a kernel function that implements the 3rd loops (2 functions).
    a. Does this achieve better performance.
13. Optional: How could we handle data arrays bigger than what can fit in the GPU.

Compare results guideline.

- Run these tests taking in mind the memory size of the GPU (it could cause the program to crash)

Number of Matrices	Matrix Size	Sequential (Approx.)	Baseline		DP with 1 loop	
			All GPU	Kernel	All GPU	Kernel
100	100					
100	1000					
100	3000					
1000	1000					
2000	750					
5000	500					