CME213, Spring 2022
Parallel Computing
Julian Cooper

Homework 3
Due Sunday, May 1st via GradeScope

Problem 1: Recurrence Implement a simple CUDA program for a recurrence relation (inspired by the Mandelbrot Set) for many different starting points.

- **1.1 Allocate GPU memory** Idea: Use `cudaMalloc` to allocate memory on the GPU device for both the input and output arrays we will need for the recurrence implementation. Free memory with `cudaFree` at end of `main()`.

```
        // Allocate num_bytes of memory to the device arrays
        cudaMalloc(&device_input_array, num_bytes);
        cudaMalloc(&device_output_array, num_bytes);
        ...
        ...

        // Deallocate memory from both device arrays
        cudaFree(device_input_array);
        cudaFree(device_output_array);
```

- **1.2 Initialize array of random floats** Idea: Use in-built rand() functor from the standard cpp library and scale it between -1 and 1 as required.

```
        // Initialize an array of size arr_size in input_array with
        //random floats between -1 and 1
        void initialize_array(vec &input_array, size_t arr_size) {
          input_array.resize(arr_size);
          std::generate(input_array.begin(), input_array.end(), rand);

          for(int i=0; i<arr_size; i++){
            input_array[i] = static_cast<float>(input_array[i])/(RAND_MAX/2)-1;
          }
        }
```

- **1.3 Implement recurrence kernel** Idea: Recurrence operations themselves are not independent and therefore not parallelizable, however, we can parallelize doing many of these recurrence loops for different starting points (constants). So, we want to parallelize over our 1-dim input array of constants.

  Since our kernel needs to handle cases where the number of threads is less than the number of entries in our input array, we need to use a grid-stride loop.

```
    /**
    * Implement the kernel recurrence.
    * The CPU implementation is in host_recurrence() in main_q1.cu.
```

```
*/
__global__ void recurrence(const elem_type* input_array,
                           elem_type* output_array,
                           size_t num_iter,
                           size_t array_length) {

    for (int xid = blockIdx.x * blockDim.x + threadIdx.x;
        xid < array_length;
        xid += blockDim.x * gridDim.x) {

        elem_type z = 0;
        elem_type constant = input_array[xid];

        int it=0;
        while(it<num_iter) {
        z = z * z + constant;
        it++;
        }
        output_array[xid] = z;
    }
}
```

Console logs.

```
Starting at Fri Apr 29 00:28:45 UTC 2022

nvcc -O3 -std=c++11 -arch=compute_75 -code=sm_75 -o main_q1 main_q1.cu

Output from main_q1
----------------
Largest error found at pos: 15 error 7.81565e-08 expected 1.52526 and got 1.525
Largest error found at pos: 0 error 0 expected 0.680375 and got 0.680375
Largest error found at pos: 439038 error 1.19193e-07 expected 1.00014 and got 1
Largest error found at pos: 142710 error 2.38333e-07 expected 2.00072 and got 2
Largest error found at pos: 482709 error 5.61004e-07 expected 16.9994 and got 1
Largest error found at pos: 482709 error 1.15797e-06 expected 289.897 and got 2
Largest error found at pos: 482709 error 2.324e-06 expected 84041.4 and got 840
Largest error found at pos: 482709 error 4.63941e-06 expected 7.06296e+09 and g
Largest error found at pos: 482709 error 9.25711e-06 expected 4.98854e+19 and g
Largest error found at pos: 138972 error 1.79297e-05 expected 8.03779e+21 and g
Largest error found at pos: 905817 error 2.59306e-05 expected 1.66519e+35 and g

Questions 1.1-1.3: your code passed all the tests!

                Q1.4
------------------------------------------
Number of Threads   Performance TFlops/sec
               32                 0.843698
               64                  1.71319
               96                   2.5743
              128                  3.42082
              160                   4.4214
              192                  5.79964
              224                   6.8216
              256                  7.70162
              288                  8.27133
              320                  9.18469
              352                  10.0615
              384                  10.7552
              416                   10.529
              448                  11.2155
              480                  11.9355
              512                  12.3663
              544                  11.3325
              576                  11.5058
              608                  12.4919
              640                  13.0052
              672                  11.5092
              704                   12.075
              736                   12.695
              768                  12.7192
              800                  11.7815
              832                  12.4544
              864                  12.4641
              896                  13.1923
              928                  12.4277
              960                  12.2472
              992                  12.8563
             1024                  13.0942

                Q1.5
------------------------------------------
Number of Blocks   Performance TFlops/sec
              36                  1.68669
              72                  3.35125
             108                  4.43883
             144                  5.86105
             180                   6.8456
             216                   8.1191
             252                  8.12224
             288                  9.39581
             324                  7.71112
             360                  8.63641
             396                  8.93084
             432                  9.58119
             468                  9.36923
             504                   9.9853
             540                  9.35853
             576                  9.94791
             612                  9.56487
             648                  9.41304
             684                  9.93957
             720                  10.5008
             756                  10.0355
             792                  10.6237
             828                  10.1014
             864                  11.9466
             900                   12.516
             936                   12.909
             972                  12.8201
            1008                  12.9218
            1044                  12.5967
            1080                  12.8112
            1116                  12.4403
            1152                  12.7656

                Q1.6
------------------------------------------
Number of Iters   Performance TFlops/sec
             20                  1.54131
             40                  3.29381
             60                  4.80154
             80                  5.64972
            100                  6.20655
            120                  7.32422
            140                  5.63426
            160                  8.15661
            180                   8.3581
            200                  8.83392
            300                  6.81323
            400                  8.40619
            500                  8.00461
            600                  8.20031
            700                  8.46228
            800                  8.77809
            900                  9.48087
           1000                  9.32557
           1200                  9.32836
```

1400	9.49436
1600	9.68148
1800	9.66993
2000	9.91434
2200	9.97533
2400	10.0087
2600	10.1537
2800	10.083
3000	10.2325

- **1.4 Vary number of threads per block** ... Many threads concurrently trying to access cache (traffic)

  [Insert graph]

- **1.5 Vary number of blocks**

- **1.6 Vary number of iterations**

Problem 2: PageRank ...

- **2.1: ...** ...

Submission information logs.

```
jelc@cardinal1:~$ /afs/ir.stanford.edu/class/cme213/script/submit.py hw2 private/cme213-
Submission for assignment 'hw2' as user 'jelc'
Attempt 1/10
Time stamp: 2022-04-13 20:09
List of files being copied:
    private/cme213-jelc53/hw2/sum.h  [768 bytes]
    private/cme213-jelc53/hw2/parallel_radix_sort.h  [7625 bytes]

Your files were copied successfully.
Directory where files were copied: /afs/ir.stanford.edu/class/cme213/submissions/hw2/jel
List of files in this directory:
    sum.h  [768 bytes]
    parallel_radix_sort.h  [7625 bytes]
    metadata  [137 bytes]

This completes the submission process. Thank you!

jelc@cardinal1:~$ ls /afs/ir.stanford.edu/class/cme213/submissions/hw2/jelc/1
metadata  parallel_radix_sort.h  sum.h
```