Module 9 Lab

Reduction - Thrust

GPU Teaching Kit – Accelerated Computing

OBJECTIVE

Use the Thrust template library to perform a reduction of a 1D list of 32-bit floats. The reduction should give the sum of the list. Your code should be able to handle lists of arbitrary length, but for brevity assume that all data will fit within the GPU global memory.

PREREQUISITES

Before starting this lab, make sure that:

- You have completed week 4 lecture videos

INSTRUCTION

Edit the code in the code tab to perform the following:

Instructions about where to place each part of the code is demarcated by the `//@@` comment lines.

QUESTIONS

(1) Name 3 applications of reduction.

ANSWER: **Determining min, max, and average of a data series.**

(2) Are there places in your solution where there is an implicit memory copy between the host and device (a copy that is not from `thrust::copy`)?

ANSWER: **In the example solution, this happens when hostOutput is assigned the value returned by 'thrust::reduce()';**

(3) If the Thrust version of reduce were not performing as well as you expected, how might you go about investigating and solving the problem?

ANSWER: **This could be quite challenging due to thrust's relatively simple iterface. One option is to use 'nvprof' or CUDA's visual profiler in combination with looking at the thrust source code.**

CODE TEMPLATE

The following code is suggested as a starting point for students. The code handles the import and export as well as the checking of the solution. Students are expected to insert their code is the sections demarcated with `//@@`. Students expected the other code unchanged. The tutorial page describes the functionality of the `wb*` methods.

```
#include <thrust/device_vector.h>
#include <thrust/host_vector.h>
#include <wb.h>

int main(int argc, char *argv[]) {
  wbArg_t args;
  float *hostInput;
  float hostOutput;
  int inputLength;

  args = wbArg_read(argc, argv); /* parse the input arguments */

  // Import host input data
  wbTime_start(Generic, "Importing data to host");
  hostInput = (float *)wbImport(wbArg_getInputFile(args, 0), &inputLength);
  wbTime_stop(Generic, "Importing data to host");

  wbTime_start(GPU, "Doing GPU Computation (memory + compute)");
  // Declare and allocate thrust device input and output vectors
  wbTime_start(GPU, "Doing GPU memory allocation");
  //@@ Insert code here
  wbTime_stop(GPU, "Doing GPU memory allocation");

  // Copy to device
  wbTime_start(Copy, "Copying data to the GPU");
  //@@ Insert code here
  wbTime_stop(Copy, "Copying data to the GPU");

  // Execute vector addition
  wbTime_start(Compute, "Doing the computation on the GPU");
  //@@ Insert Code here
  wbTime_stop(Compute, "Doing the computation on the GPU");
  ////////////////////////////////////////////////////////

  wbTime_stop(GPU, "Doing GPU Computation (memory + compute)");

```

```
37    wbSolution(args, &hostOutput, 1);
38
39    free(hostInput);
40    return 0;
41  }
```

CODE SOLUTION

The following is a possible implementation of the lab. This solution is intended for use only by the teaching staff and should not be distributed to students.

```
1   #include <thrust/device_vector.h>
2   #include <thrust/host_vector.h>
3   #include <wb.h>
4
5   int main(int argc, char *argv[]) {
6     wbArg_t args;
7     float *hostInput;
8     float hostOutput;
9     int inputLength;
10
11    args = wbArg_read(argc, argv); /* parse the input arguments */
12
13    // Import host input data
14    wbTime_start(Generic, "Importing data to host");
15    hostInput = (float *)wbImport(wbArg_getInputFile(args, 0), &inputLength);
16    wbTime_stop(Generic, "Importing data to host");
17
18    // Declare and allocate host output
19    //@@ Insert code here
20
21    wbTime_start(GPU, "Doing GPU Computation (memory + compute)");
22
23    // Declare and allocate thrust device input and output vectors
24    wbTime_start(GPU, "Doing GPU memory allocation");
25    //@@ Insert code here
26    thrust::device_vector<float> deviceInput(inputLength);
27    wbTime_stop(GPU, "Doing GPU memory allocation");
28
29    // Copy to device
30    wbTime_start(Copy, "Copying data to the GPU");
31    //@@ Insert code here
32    thrust::copy(hostInput, hostInput + inputLength, deviceInput.begin());
33    wbTime_stop(Copy, "Copying data to the GPU");
34
35    // Execute vector addition
36    wbTime_start(Compute, "Doing the computation on the GPU");
37    //@@ Insert Code here
38    hostOutput = thrust::reduce(deviceInput.begin(), deviceInput.end());
39    wbTime_stop(Compute, "Doing the computation on the GPU");
40    ////////////////////////////////////////////////////////
```

```
41
42    wbTime_stop(GPU, "Doing GPU Computation (memory + compute)");
43
44    wbSolution(args, &hostOutput, 1);
45
46    free(hostInput);
47    return 0;
48  }
```