600086 Lab Book

# Week 3 – Lab C

Date: 22nd Feb 2022

## E1. Vector dot-product

### Question:

### Text Description automatically generated with medium confidenceSolution:

Text

Description automatically generated

### Test data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | 1 | 2 | 3 | 4 | 5 |
| **B** | 10 | 20 | 30 | 40 | 50 |

### Sample output:



### Question:

### Graphical user interface, text, application Description automatically generatedSolution:

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

### Test data:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **B** | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 | 120 |

### Sample output:

Text

Description automatically generated

### Reflection:

This is a very quick technique for calculating the dot product when increasing the lengths of the arrays used in the C++ method the completion time increases greatly whereas with the GPU solution the lengths of the test data array can be increased with almost no difference to the execution times.

### Metadata:

“Dot-Product”,”thread”,”c++”,”GPU”

### Further information:

## E2. Vector dot-product using unified memory

### Question:

Exercise 2.1 Vector dot-product using managed memory

### Solution:

Text

Description automatically generated

Text

Description automatically generated

### Test data:



### Sample output:

Text

Description automatically generated

### Question:

Exercise 2.3 Vector dot-product using GPU-declared \_\_managed\_\_ memory

### Solution:

Text

Description automatically generated

Text

Description automatically generated

### Test data:



### Sample output:

Text

Description automatically generated

### Reflection:

Declaring a shared variable makes the code much neater and easier to read I struggled getting cudaMallocManaaged() to work as I had expected it however the declared shared memory worked immediately and felt much more intuitive. Added the for loops on the print statements sin each section as I had built them with extension in mind so the size characteristic can be adjusted to suit larger vectors and the code will run without fault limited purely by the thread limit.

### Metadata:

“Dot-Product”,”thread”,”c++”,”GPU”,”declared”,”managed”

### Further information:

E3. Vector dot-product using shared Memory

### Question:

Task 1. Threads synchronization.

Analyse the above process and identify areas where thread execution needs to be synchronized by

calling CUDA function: \_\_syncthreads();

### Solution:

Text

Description automatically generated

Text

Description automatically generated

### Test data:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **B** | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 |

### Sample output:

Text

Description automatically generated

### Reflection:

The \_\_sync thread function has been added into the kernel prior to the subtotal calculation this is to ensure all active threads reach this point before collapsing the values this is because a thread may still require access to c[0] and c[1] which are overwritten as a result of this subtotal calculation.

### Question:

Task 2. Consider different thread configurations, for example, <<<1, 8>>>, <<<2, 4>>>, <<<4, 2>>>

and observe if the above program can calculate the vector dot-product correctly. If not, analyse the

issues and consider how to fix them.

### Solution:

Text

Description automatically generated

Text

Description automatically generated

### Test data:

### Sample output:

### Reflection:

In its initial state the program cannot compile as it only handles up to 2 return blocks to change the code so that it can handle any combination of threads and blocks then adjustments need to be made to the shared variable this is done by adding a set of variables to define these at the head of the solution adjusting these allows for any size of array the only restriction is that the blocks\*threadsPerBlock must be greater than the ArrayLength variable

### Metadata:

“Dot-Product”,”thread”,”c++”,”GPU”,”declared”,”managed”

### Further information:

N/A