Tutorial 3
Memory Consistency and CUDA Programming

CS3210 – 2020/21 Semester 1

Learning Outcomes

1. Understand & apply the various memory consistency models
2. Understand & utilise programming models on GPGPU

1. **[Memory Consistency]** Examine the following shared memory program, running on three processors.

P1	P2	P3
(1) X = 1 (2) Y = 1 (3) while (T == 0); (4) print A	(5) while (Y == 0); (6) while (Z == 0); (7) T = X (8) A = X (9) B = X	(10) Z = 2 (11) X = 3 (12) while (T == 0); (13) print B

Code 1: Operations that will execute in each processor

The above code aims to ensures that `print A` and `print B` prints the same value. All variables are initially set to 0.

(a) What values can variables A, B, T, X have, under sequential consistency model, after all operations are completed?

(b) This code fails to achieve the aforementioned aim (i.e. ensuring that `print A` and `print B` prints the same value), even under sequential consistency. Give an execution scenario by specifying the instruction interleaving, showing that the code fails even under sequential consistency.

(c) Rewrite the code such that the code would not fail under sequential consistency.

(d) Would the new code fail under relaxed consistency? Justify your answer with examples.

2. **[Relaxed Consistency]** Examine the following shared memory program:

P1	P2	P3
(1) X = 5 (2) Y = 2 (3) print Y	(4) while (X == 0); (5) Y = 6 (6) print Z	(7) while (Y == 0); (8) Z = 5 (9) print X

Code 2: Shared memory program operations on all processors

The program was run systems that utilise different relaxed consistency models, namely, Total Store Ordering (TSO), Processor Consistency (PC) & Partial Store Ordering (PSO).

(a) In another run, the following output was observed:

```
X: 5
Y: 2
Z: 0
```

Under which relaxed consistency model(s) is this output valid and why?

(b) The following output was printed during one of the runs:

```
X: 0
Y: 2
Z: 5
```

Under which relaxed consistency model(s) is this output valid and why?

3. **[Parallel programming models on GPU - question adapted from Final assessment AY2018/19 S1]** Consider the problem of matrix-vector multiplication: $A \cdot b = c$, where A is a 2D array of size $N \times N$, and b and c are 1D arrays of size N. Each element of the result is computed as follows: $c_i = \sum_{j=0}^{N-1} a_{ij} \cdot b_j$

   (a) How would you classify the memory model for CUDA: shared memory, distributed memory, or distributed shared memory? Explain by mentioning different type of memories available in a CUDA program.

   (b) A kernel uses 64 registers and each block has 512 threads and requires very little shared and local memory. This kernel is run on a GPU device of compute capability 6.x with 65536 (1024x64) registers. How many blocks and warps can reside on the GPU at a time?

   (c) You are required to implement matrix-vector multiplication in CUDA. You should minimally show:
   - Type of memory that the kernel is using
   - Number of blocks in a grid, number of threads in a block, virtual arrangement of the threads (1D/2D/3D)
   - Kernel code
   - Code that calls the kernel Note: marks are not deducted for minor issues in syntax

   (d) A CUDA implementation of the matrix-vector multiplication (point c.) stores matrix A and vectors b and c in shared memory. However, the size of the data is greater than the amount of space available in shared memory. You should continue to use shared memory. Explain how to change your implementation to achieve this.