ITCS 4145 Parallel Computing
Test 2 5:00 pm - 6:15 pm, Wednesday April 13th, 2016

Name: .....

Answer questions in space provided below questions. Use additional paper if necessary but make sure your name is on additional sheets. *Clearly show how you obtained your answers. (No points for simply writing a numeric answer without showing how you got the answer, even if correct.)* This is a closed book test. Do not refer to any materials except those supplied for the test. Supplied: *"Brief Summary of CUDA"*

Total /40
6 pages

Qu. 1 Answer each of the following briefly:

(a) In Assignment 4 (Monte Carlo π calculation using MPI with a workpool), how do the slaves know when to terminate?

2

The suggestion in the assignment write up is for the master to send each slave a message with a tag with a particular value (say 999) that is interpreted as a terminator.

(b) What does the Suzaku routine **SZ_Scatter(A,A1) do**? How is the size of data transfer determined?

4

To scatter an array from the master to all processes, that is, to send consecutive blocks of data in an array to consecutive destinations. The size of the block sent to each process is determined by the size of the destination array, **A1**.

(c) In the command mpicc, what does the -c option specify (as used in Assignment 5 on Suzaku)?

2

Creates an object file, e.g. suzaku.o

(d) In the Seeds framework, there is a variable called “segment” that is an input parameter in the DiffuseData method. What is the name of the corresponding variable in the Suzaku framework and what does it specify?

2

taskID, an integer identifying the task.

(e) What is the sequential time complexity of the algorithm used in the gravitational *N*-body problem given *N* bodies and *t* iterations? When parallelized using a master-slave pattern, what is the parallel time complexity (not given in the slides but can be inferred if each process handles one body at the same time)?

4

$O(N^2t)$
When parallelized: $O(Nt)$

(f) Very briefly describe how to solve a general system of linear equations ***by iteration***.

2

Rearrange the *i*th equation to give the *i*th unknown in terms of the other unknowns. Make an initial guess for each unknown. Then compute each unknown from the equations repeatedly until converged sufficiently.

(g) In a pipeline executing more than one instance of a complete problem, the speedup is given by:

$$S(m) = p \times m/(p + m - 1)$$

where there are m instances of the problem and p pipeline stages. Briefly explain how this equation derived? 2

p x m is the sequential time
p + m – 1 is the parallel time. Takes p steps to complete the first instance and m -1 for the remaining m - 1 instances.

(h) What is the best possible sequential time complexity (lower bound) for a compare and exchange sorting algorithm that does not use any special properties of the numbers if there are N numbers? (Sufficient to state without proof.)

2

O(N log N)

Hence what is best possible parallel time complexity (lower bound) for a compare and exchange sorting algorithm that does not use any special properties of the numbers if there are N numbers and N parallel processes?

2

O(log N)

(i) Show the steps to sort the following four numbers using odd-even transposition sort. It will take four steps for four numbers generally and show all four steps even if sorted before that. 4

7 4 8 3

7↔4 8↔3
4 7↔3 8
4↔3 7↔8
3 4↔7 8
3 4 7 8

(j) Is the following a Bitonic sequence? Explain your answer. 2

3 4 5 3 4 5

No. It does not have an increasing sequence followed by a decreasing sequence even if rotated

(k) In CUDA, what does **threadIdx.y** indicate? 2

Indicates the thread index in the y dimension in a CUDA block.

Qu. 2 Write a CUDA program that copies the values from a 1-dimensional array of integers, A[N], into a second array B[N] in reverse order, e.g. if the array A[N] has the values:

A[0]	A[1]	A[2]	…	A[N-3]	A[N-2]	A[N-1]
4	6	5	…	3	1	9

afterwards the array B[N] has the values:

B[0]	B[1]	B[2]	…	B[N-3]	B[N-2]	B[N-1]
9	1	3	…	5	6	4

Use one CUDA thread to copy one element in the array. Organize the kernel structure as a 1-D grid of 1-D blocks with 16 threads in each block and the minimum number of blocks necessary given N as a defined constant. For example if N = 30, you would need two blocks. Your code must take into account any value for N. You may assume that the program has code to store initial values in the array A[N]. Declare all variables and arrays needed. Provide comments in your code to help the grader! ***If I do not understand the code, I will assume it is incorrect.***

Note: The C library function **double ceil(double x)** returns the smallest integer value greater than or equal to **x**. Use this to round a number up.

10

```
#define N 2000     // size of vectors  can be any value

__global__ void reverse(int *a, int *b) {
        int i = blockIdx.x*blockDim.x + threadIdx.x;
        if (i < N) b[N - i - 1] = a[i];
}

int main (int argc, char **argv ) {
        int  A[N], B[N], *devA, *devB;

        int size = N *sizeof( int);
        int grid =  (int) ceil((double) N/16);
                                                // put values into array A
        cudaMalloc( (void**)&devA, size) );
        cudaMalloc( (void**)&devB, size );
        cudaMemcpy( devA, A, size, cudaMemcpyHostToDevice);
        reverse<<<grid, N>>>(devA, devB);
        cudaMemcpy( B, devB size, cudaMemcpyDeviceToHost);

        cudaFree( devA);
        cudaFree( devB);
        return (0);
}
```

Intentionally blank to continue you answer for question 2