National University of Computer and Emerging Sciences
Karachi Campus

Parallel & Distributed Computing (CS2006)

Final Exam

Total Time (hrs)	3
Total Marks	50
Total Questions	7

Date: 23/5/2024

Course Instructor

Mr. S. Farooq Zaidi

Roll No ______ Section ______ ______ Student Signature

Do not write below this line.

- Attempt All questions.
- Write with a clear blue pen in neat handwriting.
- **Answer all parts of a question in order and together.**
- All questions are self-explanatory. In case of any confusion on state your assumptions and solve the question. You will be compensated if you are correct. I may or may not visit for queries.

CLO 1: Learn about parallel and distributed computers.

Q1. For a certain parallel program, Amdahl's law only gives a speed-up between 1.1 and 1.15 and with a higher number of CPUs the speed-up is negligible. However, Gustafson's law keeps on giving a decent speed-up even for large numbers of CPU Cores around 1.4.
Give reasons for each of the following scenarios using the above information whether the parallelized program should be used or not.

A. Input data is fixed at 4096 integers; a large number of CPU cores are available.
B. Input data is fixed at 4096 integers; max 16 CPU cores available.
C. No limit on input data, large number of CPU cores available.
D. No limit on input data, max 2 CPU cores available.

E. Write the definition of Amdahl and Gustafson laws along with formulas explaining each term.

[20 min,1 x 5 = 5 marks]

CLO 2: Write portable programs for parallel or distributed architectures using Message-Passing Interface (MPI) library.

Q2. An array of N elements was scattered on P processors using MPI_Scatter call. Later some extensive operations were performed on each element, changing its value. We now need to get product numbers among all these elements (scattered throughout P processors). We need this maximum on all P processors.

A. What is the most effective MPI call(s) for the task mentioned above?
B. What will be the correct arguments for the MPI call you have mentioned in part (a) for the master and slave processor. Briefly mention the purpose of each argument.
(Assume master is ranked 0.)

[25 min,2.5 + 2.5 = 5 marks]

National University of Computer and Emerging Sciences

Karachi Campus

CLO 3: Analyze complex problems with shared memory programming with openMP.

Q3.

A. The below OpenMP code is not giving any improvement in runtimes if run on 1 CPU, 16 CPUs or even 56 CPUs? 'ACME Solutions' team is stuck and needs your assistance. Point out the issue in the code below. Also, give the correct parallel code that should be used to accomplish the same task being performed in the code given below.

```
float acc = 0;
#pragma omp for
    for (int i = 0; i < n; i++) {
                float x = algorithm(i);
                acc += x;
    } //for
```

B. In the context of parallel and distributed computing, what factors would influence your decision to use MPI versus OpenMP versus GPU programming for solving a particular problem? Use scenarios for each case (i.e. MPI, OpenMP, and GPU programming) to build your arguments.

[15 min, 2.5 x 2 = 5 marks]

CLO 1: Learn about parallel and distributed computers.

Q4. For the vector addition kernel and the corresponding kernel launch code, answer each of the sub-questions below.

```
1.    __global__
2.    void Add(float* A_d, float* B_d, float* C_d, int n) {
3.      int i = threadIdx.x + blockDim.x * blockIdx.x;
4.      if(i<n) C_d[i] = A_d[i] + B_d[i];
5.    }

6.    int vectAdd() {
7.      //assume A, B and C have been declared
8.      int size = n * sizeof(float);
9.      cudaMallocManaged((void**) &A, size);
10.     cudaMallocManaged((void**) &B, size);
11.     cudaMallocManaged((void**) &C, size);
12.     //Assume that data is copied into A, B and C from somewhere.
13.     Add<<<ceil(n/256), 256>>>(A, B, C, n);
14.   }
```

A. Assume that the size of A, B, & C is 1000 elements. How many thread blocks will be generated?
B. Assume that the size of A, B, & C is 1000 elements. How many warps are there in each block?
C. Assume that the size of A, B, & C is 1000 elements. How many threads will be created in the grid?
D. Assume that the size of A, B, & C is 1000 elements. Will there be any warp with thread divergence and what will be its maximum slowdown?
E. Assume that the size of A, B, and C is 768 elements. Is there any control divergence during the execution of the kernel? If so, identify the line number of the statement that causes the control divergence. Explain why or why not.

[25 min, 2 x 5 = 10 marks]

National University of Computer and Emerging Sciences
Karachi Campus

CLO 1: Learn about parallel and distributed computers.

Q5. For the vector addition kernel and the corresponding kernel launch code given in Q4 above, answer each of the sub-questions below.

A. Is there any better kernel launch configuration that might show more speedup. Briefly explain the aspects of kernel launch configuration that will support your answer.
B. Update the kernel code to incorporate tiling for vector additions so that more 1000 elements (1 million or so) can be handled. Explain what you have done.
C. Is there any better kernel launch configuration that might show more speedup on this updated kernel. Briefly explain the aspects of kernel launch configuration that will support your answer.
D. Why don't we need halo elements in tiled vector addition?

[30 min, 2.5 x 4 = 10 marks]

CLO 1: Learn about parallel and distributed computers.

Q6.

A. What does __synchthreads() do? What might go wrong if __syncthreads were removed from the code below?

```
1.    __shared__ int our_data[1025];
2.    our_data[threadIdx.x] = my_element;
3.    __syncthreads();
4.    output_data[tid] = my_element + our_data[threadIdx.x + 1];
```

B. Imagine using tiling to sort a large number of elements (much much greater than the cores available on GPU). Discuss which of the two sorting algorithms odd-even sort or merge sort is best suited for CUDA Device?
C. Give a structure + pseudocode of merge sort on cuda and show where will __syncthreads() will be placed.
D. How is the prefix scan function used in radix sort. How does a CUDA parallel prefix scan function work? Explain briefly with help of a diagram and example.
E. Discuss the benefits of launching a GPU programming kernel with more than one block. How is it related to the nature of the problem being solved? How do threads per block affect the computation? Give coding examples of each of these three questions separately.

[30 min, 2 x 5 = 10 marks]

CLO 1: Learn about parallel and distributed computers.

Q7.

A. Give a tiled matrix multiplication kernel that uses only P number of threads where the tile size is P x P. Explain your code.
B. Given that matrices are of size 2048 x 1024. Give a few lines of code to where kernel launch parameters are declared and initialized using the dim3 structure for best optimality.
C. How does tiled matrix multiplication help in speedup if the same code is implemented without tiling in the GPU programming context?

[20 min, 2 + 1 + 2 = 5 marks]