SWE3021 Multicore Programming - Midterm Exam Fall 2019

Student ID	Name

For Instructor/TA only,

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total

Academic Honor Pledge

I affirm that I will not at any time be involved with cheating or plagiarism while enrolled as a student Programming Language class at SungKyunKwan University. I understand that violation of this code will result in penalties as severe as indefinite suspension from the university.

Your signature: ____________________

Your exam and project scores will be posted under anonymous alias. Please provide an alias that only you can recognize. Please do not use any offensive alias.

Your alias: ____________________

1. Provide a brief description of the following terms. [20 pts]

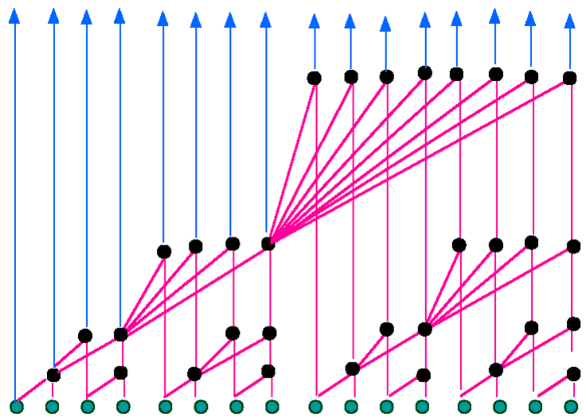
a. False Sharing:
Concurrent threads access different parts of the same cacheline, which invalidates each other's CPU cache.

b. SIMD:
Single Instruction Multiple Data: That is, with a single instruction, multiple data are concurrently processed.

c. NUMA:
Non Uniform Memory Access: A computer architecture where each CPU socket has its own memory banks. In a NUMA machine, accessing data from a remote memory bank is slower than accessing data from local memory bank.

d. Straggler Problem:
Some processors will likely run out of work to do before others are finished

2. The following figure shows the dependency of 16 processors that compute the prefix sum of an input array of size 16. [15 pts]



a. What is the critical path?

5

b. What is the maximum degree of concurrency?

16

c. What is the average degree of concurrency?

(16+8+8+8+8)/5 = 48/5

3. You have written a parallel program that shows the following execution times with varying the number of processors. [15pts]

Number of Processors	Execution Time
1	110 sec.
2	60 sec.
4	35 sec.
8	22.5 sec.

a. According to Amdahl's Law, how much speed-up can you achieve from this program? If you need assumptions for your answer, please describe them.

Sequential part: 10 seconds.

With infinite number of processors, the execution time will drop down to 10, i.e., 11x faster performance can be achieved with an infinite number of processors.

Speed-up = T1 / Tf = 110 / 10 = 11.

b. Is this program strong scaling? Define "strong scaling" and justify your answer.

Yes. This program seems to have a fixed problem size, which takes 100 seconds with a single processor. With N processors, the parallel execution time decreases by a factor of N. Therefore, this program is strong scaling.

c. According to Gustafson's Law, how much speed-up can you achieve from this program? If you need assumptions for your answer, please describe them.

Unless the problem size increases as the number of processors increases, Gustafson's law cannot be applied.

4. Does the following code have loop-carried dependences? If so, rewrite the code to eliminate them. [10 pts]

```
int A[N], B[N+1], C[N];

#pragma omp parallel for private(i)
for(i=1; i<N; i++){
    A[i] = A[i] + B[i];
    B[i+1] = C[i] + A[i+1];
}
```

Your answer:

```
int A[N+1], B[N+1], C[N];

A[1] = A[1] + B[1];
#pragma omp parallel for private(i)
for(i=1; i<N-1; i++){
    B[i+1] = C[i] + A[i+1];
    A[i+1] = A[i+1] + B[i+1];
}
B[N] = C[N-1] + A[N];
```

.

5. Rewrite the following code using SOR (Successive Over Relaxation) to eliminate loop-carried dependences. [10 pts]

```
int A[100][100];
for(k=0;l<100;k++){
   #pragma omp parallel for private(i,j)
   for(i=1; i<100; i++){
       for(j=1; j<100; j++){
            A[i][j] = 0.25*(A[i-1][j] + A[i][j-1] + A[i+1][j] + A[i][j+1]);
       }
   }
}
```

Your answer:

```
int A[101][101];
for(k=0;l<100;k++){
   #pragma omp parallel for private(i,j)
   for(i=1; i<100; i+=2){
       for(j=1; j<100; j+=2){
            A[i][j] = 0.25*(A[i-1][j] + A[i][j-1] + A[i+1][j] + A[i][j+1]);
       }
       for(j=2; j<100 && i<99; j+=2){
            A[i+1][j] = 0.25*(A[i][j] + A[i+1][j-1] + A[i+2][j] + A[i+1][j+1]);
       }
   }

   #pragma omp parallel for private(i,j)
   for(i=1; i<100; i+=2){
       for(j=2; j<100; j+=2){
            A[i][j] = 0.25*(A[i-1][j] + A[i][j-1] + A[i+1][j] + A[i][j+1]);
       }
       for(j=1; j<100 && i<99; j+=2){
            A[i+1][j] = 0.25*(A[i][j] + A[i+1][j-1] + A[i+2][j] + A[i+1][j+1]);
       }
   }
}
```

.

6. Given the following sequence of numbers:

```
9, 1, 13, 2, 0, 10, 8, 3, 14, 6, 15, 12, 4, 5, 11, 7
```

a. Show the steps of bitonic sorting. [10pts]

Answer:
9 1 13 2 0 10 8 3 14 6 15 12 4 5 11 7

1 9 13 2 0 10 8 3 6 14 15 12 4 5 11 7

1 2 9 13 10 8 3 0 6 12 14 15 11 7 5 4

0 1 2 3 8 9 10 13 15 14 12 11 7 6 5 4

—

0 1 2 3 7 6 5 4 15 14 12 11 8 9 10 13

0 1 2 3 7 6 5 4 8 9 10 11 15 14 12 13

0 1 2 3 5 4 7 6 8 9 10 11 12 13 15 14

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

b. What is the average degree of concurrency for parallel bitonic sorting when N processors sort N numbers? ($N = 2^k$). [10pts]

Answer:
$2 \times logN = 2 \times k$ steps.
All N processors are involved in every step.
Therefore, the average degree of concurrency is (N * k) /k = N

7. Write a parallel function `void counting_sort(int arr[], int size)` in pseudo code (e.g., parallel_for i in 1..N), Pthread, or OpenMP that counts the number of keys, builds a histogram, and compute the position in answer array of size MAX, i.e., elements in arr[] are guaranteed to be greater than or equal to 0 but smaller than MAX. [10 pts]

Your answer:

```
void counting_sort(int arr[], int size)
{
    int histogram[MAX];
    #pragma omp parallel for
    for (i=0; i<MAX; i++){
        histogram[i] = 0;
    }

    #pragma omp parallel for
    for (i=0; i<size; i++){
        #pragma omp atomic
        histogram[arr[i]]++;
    }

    for (i=0;i<MAX;i++){
        for (j=0;j<histogram[i];j++){
            cout << i << " ";
        }
    }
}
```

.

8. Consider a 2D partitioning of a matrix on 16 processors. Suppose you are running a parallel Gaussian elimination program using 16 processes. Each process has its own local memory that are not shared with other processes. How many one-to-one communications between processes will be be necessary? Justify your answer with illustrations of each step. (If one process broadcasts its sub-matrix to K processes, the number of communications is K.) [10 pts]

Your answer:

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(a) Step 1

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(b) Step 2

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(c) Step 3

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(d) Step 4