Test - Sep 30

Due 1 Oct at 23:55 **Points** 50 **Questions** 13
Available 30 Sep at 8:00 - 2 Oct at 2:00 2 days **Time limit** 150 Minutes

Instructions

This test consists of 13 questions (MCQs + MAQs + 1 programming question). The time limit is 150 minutes. It will be open from 8:00 am, Sep 30 until 23:55pm, Oct 1. (Note that for MAQs, negative marks will be given to incorrect choices.)

This quiz was locked 2 Oct at 2:00.

Attempt history

	Attempt	Time	Score
LATEST	Attempt 1	96 minutes	37.5 out of 50

Score for this quiz: **37.5** out of 50
Submitted 1 Oct at 21:56
This attempt took 96 minutes.

Question 1 — 2 / 2 pts

Choose the correct statement from the following.

- In a 5-dimensional hypercube, a node, T, is labelled 00001. Then the labels of all immediate neighbours of node T are 10001, 00000, 10010, 00101, 01001, respectively.

Correct!

- In a 5-dimensional hypercube, a node, T, is labelled 00100. Then the labels of all immediate neighbours of node T are 10100, 00000, 00110, 00101, 01100, respectively.

In a 5-dimensional hypercube, a node, T, is labelled 10001. Then the labels of all immediate neighbours of node T are 00001, 00000, 10011, 10101, 11001, respectively.

In a 5-dimensional hypercube, a node, T, is labelled 10010. Then the labels of all immediate neighbours of node T are 00000, 11010, 10011, 10000, 10110, respectively.

Question 2

2 / 3 pts

Which of the following interconnection networks can be used for building a memory modular machine (MMM)?

Correct!
- [x] Multistage network

Correct!
- [x] Bus
- [ ] Ring
- [ ] Fully connected network

rrect answer
- [ ] Fully connected crossbar network
- [ ] 3D torus
- [ ] Fat tree

Question 3

1.5 / 3 pts

Which of the following statements is (or are) correct?

rrect answer

- [ ] The bisection widths of 64-node hypercube and $4 \times 4 \times 4$ 3D torus are the same.

- [ ] Both ring and fat tree interconnection networks have a bisection width of 1.

- [ ] Both bus and fully connected network are scalable in terms of performance.

Correct!

- [x] Both 2D torus and 3D torus interconnection networks are regular networks.

Question 4

3 / 3 pts

Consider a memory system with a DRAM of 512MB and L1 cache of 32KB with the CPU operating at 1GHz. The $l_{DRAM} = 100$ ns and $l_{L1} = 1$ ns (l represents the latency). In each memory cycle, the processor fetches 4 words. What is the peak achievable performance of a two-loop dot product based matrix-vector product using this system?

```
        /*matrix-vector product loop*/
        for (i=0; i<dim; i++)
            c[i] = 0;
            for (j=0; j<dim; j++)
                c[i] += A[i][j] * b[j];
```

- [ ] 32MFLOPS

Correct!

- [x] 80MFLOPS

- [ ] 16MFLOPS

40MFLOPS

Question 5

1 / 3 pts

Choose the correct statment(s) from the following.

orrect answer

It is possible to design a broadcast algorithm for ring interconnect network with time complexity $O(\log p)$, where p is the number of nodes in the ring.

orrect answer

The time complexity of broadcast in bus interconnect network is $O(1)$.

Correct!

It is possible to design a broadcast algorithm for hypercube interconnect network with time complexity $O(\log p)$, where p is the number of nodes in the ring.

The time complexity of broadcast in ring interconnect network is $O(1)$.

Question 6

2 / 2 pts

Given the code segment in the following,

```
int a = 1;
int main(int argc, char **argv){
    int b = 2, c = 3;
    #pragma omp parallel num_threads (2) shared (b)
    {
        #pragma omp parallel num_threads (2) private (a)
        {
            int d = 4;
```

```
                #pragma omp single
                #pragma omp task
                {
                    int e, f;
                    e = b + c;
                    f = a + d;
                }
            }
        }
    }
```

which of the following is TRUE regarding the values of variables `e` and `f` within the task code region?

- `f = 5`

Correct!

- `e = 5`
- `e = 3`
- The value of `e` is undeterministic

Question 7

2 / 2 pts

Continued from previous question (Q6), how many tasks will be created in the parallel region (assume nested parallelism is enabled)?

- 3
- 1

Correct!

- 2
- 4

Question 8

3 / 3 pts

Which of the following `for` loop parallelization is (or are) correct?

```
#pragma omp parallel for
for(int i=0; i<n; i++){
    a[i] = do_work(i);
    if(a[i] < b[i])
        break;
}
```

Correct!

```
#pragma omp parallel for reduction(+:dotp)
for(int i = 0; i < n; i++) {
    dotp += a[i] * b[i];
}
```

```
#pragma omp parallel for
for(int i=1; i<100; i++)
    a[i] = i*a[i-1];
```

```
#pragma omp parallel for
for(int i=k; i<n; i++) {
    a[i] = a[i] + a[i-k];
}
```

Question 9

3 / 3 pts

The code in the following listing gives a function that will sum all of the elements in an n-element array `d`.

```
double sum_array(double *d, int n) {
    int i;
    double sum = 0.0;
    for(i=0; i<n; i++){
        double val = d[i];
        sum += val;
    }
```

```
        return sum;
    }
```

M is trying to structure the code in parallel using OpenMP. The OpenMP implementation is as follows:

```
    double sum_array_omp(double *d, int n){
        int i;
        double sum = 0.0;
        #pragma omp parallel for schedule(static)
        for(i=0; i<n; i++){
            double val = d[i];
            sum += val;
        }
    return sum;
    }
```

After testing the code, it reveals that function `sum_array_omp` returns erroneous results.

i) What is the cause of the incorrect results of function `sum_array_omp`?

Correct!

- (●) It is due to race condition.
- ( ) It is because the scope of variable `sum` is undeterministic.
- ( ) It is due to false sharing.
- ( ) It is because wrong loop schedule kind `static` is used

Question 10

4 / 4 pts

Continued from the previous question (Q9), ii) which of the following implementation(s) will give correct result?

```
double sum_array_omp(double *d, int n){
    int i;
    double sum = 0.0;
    #pragma omp parallel reduction(+:sum)
    for(i=0; i<n; i++){
        double val = d[i];
        sum += val;
    }
    return sum;
}
```

Correct!

```
double sum_array_omp(double *d, int n){
    int i;
    double sum = 0.0;
    #pragma omp parallel for reduction(+:sum)
    for(i=0; i<n; i++){
        double val = d[i];
        sum += val;
    }
    return sum;
}
```

Correct!

```
double sum_array_omp(double *d, int n){
    int i;
    double sum = 0.0;
    #pragma omp parallel for
    for(i=0; i<n; i++){
        double val = d[i];
        #pragma omp critical
        sum += val;
    }
    return sum;
}
```

```
double sum_array_omp(double *d, int n){
    int i;
    double sum = 0.0;
    #pragma omp parallel for private(i)
firstprivate(sum)
    for(i=0; i<n; i++){
        double val = d[i];
        sum += val;
    }
    return sum;
}
```

Question 11

4 / 6 pts

M is tasked to implement a parallel code for generating a histogram from the values in a large input array named `input`. For each element of the input array, the code uses the function `bin_func` to compute a 'bin' that the element belongs to (`bin_func` always returns an integer between `0` and `NUM_BINS - 1`), and increments a count of elements in that bin. M targets running the program on a small parallel machine with only two cores. The implementation is given below. Assume the cache line is 64-byte wide.

```
    #define NUM_BINS 8
    #define NUM_THREADS 2
    float input[N]; //assume input is initialized and N is very large
    int histogram_bins[NUM_BINS]; //output bins
    int partial_bins[NUM_THREADS][NUM_BINS]; //assume bins are initialized
to 0

    #pragma omp parallel num_threads(NUM_THREADS)
    {
        int k = omp_get_thread_num();
        #pragma omp for
            for (int i=0; i<N; i++)
                partial_bins[k][bin_func(input[i])]++;
    }
    for (int i=0; i<NUM_BINS; i++)
        histogram_bins[i] = partial_bins[0][i] + partial_bins[1][i];
```

M runs this code on an input of 1 million (`N = 1,000,000`) to create a histogram with 8 bins (`NUM_BINS=8`). He is very disappointed when his program obtains far less than a linear speedup. Choose the correct statment(s) from the following list regarding this implementation.

orrect answer

- [ ] Changing the value of `NUM_BINS` to greater than or equal to 16 could solve the issue, hence, improve the performance.

Correct!

- [x] The poor performance is caused by false sharing.

Correct!

The following code could solve the issue, hence, improve the performance.

```
        #define NUM_BINS 8
        float input[N]; //assume input is initialized and N is
very large
        int histogram_bins[NUM_BINS]; //output bins

        #pragma omp parallel num_threads(2)
        {
            int k = omp_get_thread_num();
            int partial_bins[NUM_BINS] = {0}; //assume bins are
initialized to 0
            #pragma omp for
            for (int i=0; i<N; i++)
                partial_bins[bin_func(input[i])]++;
            for (int i=0; i<NUM_BINS; i++)
                #pragma omp critical
                histogram_bins[i] = partial_bins[i];
        }
```

The poor performance is caused by race condition.

Question 12

4 / 6 pts

One way to get a numerical approximation of the number π is to use many terms in the formula

$$\pi = 4[1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots] = 4\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}.$$

A student implements this computation both in serial and parallel. The following listing gives the serial and parallel functions to compute number π, respectively. The serial code estimates the number π correctly. However, the OpenMP parallel code does not.

```
    double seq_pi(int n){
        double factor =1.0;
        double sum = 0.0;
```

```
        for(int k=0; k<n; k++) {
            sum += factor/(2*k + 1);
            factor = -factor;
        }
        return (4.0 * sum);
    }

    double par_pi(int n, int thread_count){
        double factor = 1.0;
        double sum = 0.0;
        #pragma omp parallel for num_threads(thread_count)
reduction(+:sum)
            for(int k=0; k<n; k++) {
                sum += factor/(2*k + 1);
                factor = -factor;
            }
        return (4.0 * sum);
    }
```

Which of the following statments is (or are) correct regarding the error(s) in the code?

Correct!

- [x] There is a loop-carried deppendency in the parallel for loop.

Correct!

- [x] The scope of variable `factor` in the parallel region is incorrect.

- [ ] The code line `sum += factor/(2*k + 1);` should be enclosed in a `#pragma omp critical` section.

orrect answer

- [ ] The scope of variable `factor` should be changed to `private` or `firstprivate`

Question 13

6 / 10 pts

We have a linked list composed of a collection of nodes, each of which is `struct` with two members: an integer and a pointer to the next node.

We would like to find out how many prime numbers are there in the list (i.e., count the total number of integer members that are prime numbers in the list). A simple baseline serial code is given in `ll_incomplete.cpp` (https://ulwazi.wits.ac.za/courses/18738/files/2113664/download?download_frd=1) . In this code, complete the function `par_count_prime`, which is a parallel version of serial function `seq_count_prime`. Test if your parallel result is the same as the serial result. Upload your program as `ll_complete.cpp`. (Note the goal of this parallelization is correctness, not necessarily a speedup greater than 1.)

ll_complete.cpp (https://ulwazi.wits.ac.za/files/2129517/download)

Code parallelized, but forced to execute sequentially

Quiz score: **37.5** out of 50