| Parallel and Distributed Computing -- (7A)<br>Quiz 03 (Fall 2022). Instructor: Dr. Syed M. Irteza | | Name: Nofeeza Masood |
|---|---|---|
| Date: 2022-10-11 | | |
| Total Marks: 15 | Time Allowed: 10 mins | Roll Number: 19L-2201 |

1. Assuming recursive doubling, what would the cost estimate of 1-to-all broadcast be (m = message size, $t_s$ = startup time, $t_w$ = transfer time, p = number of processes)? [2m]
   a. $T = (t_s + mt_w) * \log(p)$
   b. $T = (t_s + pt_w) * \log(m)$
   c. $T = t_s + (pt_w * p)$
   d. $T = (t_s + pt_w) * p$

2. For a 2*16 mesh, using a naïve solution, how many messages would you expect the sending process to send, for a 1-to-all broadcast? [2m]
   a. Less than 22
   b. More than 30
   c. Less than 10
   d. Less than 16

   (p-1) * P
   2(15) ×

3. With all-to-1 reduction, we are essentially? [2m]
   a. Requiring all (p-1) processes to send messages to all p process
   b. Requiring 1 process to send messages to (p-1) processes
   c. Requiring all (p-1) processes to send messages to 1 process
   d. Requiring all p processes to send messages to 1 process

4. Can you describe the use of the following terms or clauses in OpenMP: [3*3m]

   a. critical

Critical section is the part of thread currently performing the code. It can be performing any set of instruction.

   b. lastprivate(var)

The use of the variable updated in the last process when used in the next process is named lastprivate(var). It is basically using an updated variable in a thread

   c. schedule(static, 4)

static scheduling where the number of processes (threads used) is predefined. In this case it is 4.

4 is chunksize, not # of threads.

7

1

⇒ critical clause has the critical section that has shared memory or variables. It is used to avoid the race condition and allow several variables to share a particular var or memory.

⇒ lastprivate (var)
assign last value of master thread to the variable given
This is a private var which is declared in OpenMP part and carries last value of master thread.

⇒ schedule (static, 4)
4 is chunk size.

| Parallel and Distributed Computing -- (7A)<br>Quiz 04 (Fall 2022). Instructor: Dr. Syed M. Irteza | | Name: Moteeza Masood |
|---|---|---|
| Date: 2022-10-25 | | Roll Number: 19L-2201 |
| Total Marks: 15 (5*2 + 5) | Time Allowed: 10 mins | |

1. Linked-List iteration was not something that could easily be parallelized in OpenMP due to:
   a. We are unsure of the number of iterations involved
   b. There is no parallel while loop construct in OpenMP
   c. The data structure storing each node is very large
   d. Both (a) and (b)

11

2. When the chunk size is not specified, and we use static scheduling in OpenMP, the chunk size:
   a. Equals 1
   b. Equals total_threads
   c. Equals n/num_of_procs (where n = total iterations)
   d. Equals n/total_threads (where n= total iterations)

3. With guided scheduling, the chunk size used:
   a. Keeps increasing, till the size C (parameter) is reached
   b. Keeps decreasing, till the size C (parameter) is reached
   c. Stays constant
   d. Decreases and increases within the range of size C (parameter)

4. When `OMP_DYNAMIC` is set to `true`, then OpenMP will use:
   a. The same number of threads as mentioned in the `num_threads()` clause
   b. The same number of threads as mentioned in the `omp_set_num_threads()` function
   c. The same number of threads as `[2 * omp_get_num_procs()]`
   d. A dynamic adjustment algorithm to create threads that optimize system performance

5. When we computed the value of π using the Monte Carlo method, we did not use the omp for construct. What feature of the problem enabled us to solve without needing that?

The #pragma omp parallel feature allowed us to compute the value of π.

6. What could be the output of this program?

```
#include <iostream>
#include <omp.h>
using namespace std;
int main() {
   int nums[12] = { 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2 };

#pragma omp parallel for num_threads(4) schedule(static, 3)
   for (int j = 0; j < 12; j++) {
      nums[j] += j;
      int x = omp_get_thread_num();
      cout << "At thread: " << x << " iteration: " << j << endl;
   }

   for (int i = 0; i < 12; i++) {
      cout << nums[i] << "\t";
   }
   return 0;
}
```

R.W

nums[j] = nums[j] + j

0 = 13 + 0
1 = 12 + 1
2 = 11 + 2
⋮
11 = 2 + 11

num[j]=13
j=0
x=0

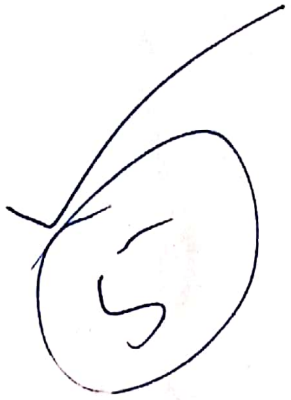
At thread: 0 iteration: 1

0, 1, 2 | 3, 4, 5 | 6, 7, 8 | 9, 10, 11

Answers at back → (TPO)

Answer:

| | | | | |
|---|---|---|---|---|
| At | thread: | 0 | iteration: | 0 |
| At | thread: | 0 | iteration: | 1 |
| At | thread: | 0 | iteration: | 2 |
| " | " | 1 | " | 3 |
| " | " | 1 | " | 4 |
| " | " | 1 | " | 5 |
| " | " | 2 | " | 6 |
| " | " | 2 | " | 7 |
| " | " | 2 | " | 8 |
| " | " | 3 | " | 9 |
| " | " | 3 | " | 10 |
| " | " | 3 | " | 11 |



13 13 13 13 13 13 13 13 13 13 13 13