|  |  |  |
| --- | --- | --- |
| Parallel and Distributed Computing ( 6E / 6F )  Quiz 03 (Spring 2022). Instructor: Dr. Syed M. Irteza | | Name: |
| Date: 2022/04/25 | | Roll Number: |
| Total Marks: 15 (5\*2m + 5m) | Time Allowed: 10 mins |

1. If we use the schedule(dynamic, 3) clause within the #pragma omp parallel for, we are enabling:
   1. Each thread is assigned 1/3rd of the total iterations of the *for* loop in round-robin manner
   2. Each idle thread is dynamically assigned 1/3rd of the remaining iterations of the *for* loop
   3. Each thread is assigned 3 contiguous iterations of the *for* loop in round-robin manner
   4. Each idle thread is dynamically assigned the 3 leftmost contiguous remaining iterations of the for loop
2. If we use the schedule(static, 4) clause within the #pragma omp parallel for, we are enabling:
   1. Each thread is assigned 1/4th of the total iterations of the *for* loop in round-robin manner
   2. Each idle thread is dynamically assigned 1/4th of the remaining iterations of the *for* loop
   3. Each thread is assigned 4 contiguous iterations of the for loop in round-robin manner
   4. Each idle thread is dynamically assigned the 4 leftmost contiguous remaining iterations of the *for* loop
3. With the schedule(guided, 2) clause within the #pragma omp parallel for, we are enabling:
   1. A scheduling mechanism where chunk sizes vary, but will not exceed 2 iterations
   2. A scheduling mechanism where chunk sizes decrease, but don’t go below 2 iterations
   3. A scheduling mechanism where the fixed chunk size is 2 iterations
   4. A scheduling mechanism where the maximum chunk size is 2 iterations
4. When the OMP\_NESTED environment variable is set to TRUE, this indicates that OpenMP will:
   1. Create a new team of threads with each layer of nested parallel pragma code
   2. Not create a new team of threads with each layer of nested parallel pragma code,
   3. Consider each new nested *for* loop to be an OpenMP *for* loop construct
   4. Not parallelize any for loops mentioned in the code
5. The OMP\_DYNAMIC environment variable when set to FALSE, indicates that OpenMP will:
   1. Always use static scheduling
   2. Always use guided scheduling
   3. Generate the same number of threads as requested by the num\_threads() clause
   4. Create threads according to its dynamic adjustment algorithm
6. When parallelizing linked list traversal, why was the #pragma omp single clause used in one of the solutions? Can you explain the purpose of this clause in that context? [5m]