Assignment 4: OpenMP - loops

The purpose of this assignment is for you to learn more about

- the parallel loop construct of OpenMP,
- how loop scheduling works in OpenMP,
- how easy (or not so easy) is writting parallel codes with OpenMP loop construct.

As usual all time measurements are to be performed on the cluster.

Activate OpenMP in GCC by passing `-fopenmp` to the compiler and linker. (Note that if you omit this parameter, the code will probably still compile. But its execution will be sequential.)

You can control the number of threads in OpenMP in two ways: the environment variable `OMP_NUM_THREADS` or by calling the `omp_set_num_threads` function.

Loop scheduling in OpenMP can be controlled either by specifying `schedule(runtime)` in the parallel for construct and specifying a `OMP_SCHEDULE` environment variable, or by using the `omp_set_schedule` function.

The first two problems are there to kickstart your OpenMP programming. The third is to show why people love OpenMP (and only use pthreads when necessary). Problems 4 and 5 are the harder bits of the assignment. Problem 6 is hard.

1 Transform

Question: Write a program that will generate a random array of integers. Then it will square the value of each entry of the array in parallel using the OpenMP for loop construct. Measure the time squaring takes. Write the program so that you can control the length of the array, the number of thread and the scheduling policy.

Question: Plot a speedup chart for different numbers of thread (1, 2, 4, 8, 16), for arrays of sizes 10^8, with different scheduling policies (static, dynamic,1, dynamic,1000, dynamic,100000).

2 Reduce

Question: Write a program that will generate a random array of integers. Then it will compute the minimum value of the array in parallel using the OpenMP for loop construct. Measure the time computing the minimum takes. Write the program so that you can control the length of the array, the number of thread and the scheduling policy.

Question: Plot a speedup chart for different numbers of thread (1, 2, 4, 8, 16), for arrays of sizes 10^8, with different scheduling policies (static, dynamic,1, dynamic,1000, dynamic,100000).

3 Numerical Integration

Question: Take your sequential code to compute numerical integration from the previous assignment. Modify it to do the numerical integration in parallel with the OpenMP loop construct.

Question: Plot a speedup chart for different numbers of thread (1, 2, 4, 8, 16), for different scheduling policies (dynamic,1 and dynamic,1000), for different number of points (N=10^3 and N=10^9), and intensity (10, and 1000).

4 Find First

Question: Implement a parallel function using OpenMP parallel loop constructs to find the first occurrence *pos* of an integer in an array using $\theta(n)$ work.
Question: Implement a parallel function using OpenMP parallel loop constructs to find the first occurrence *pos* of an integer in an array using $\theta(pos)$ work.
Question: Plot a speedup chart for different numbers of thread (1, 2, 4, 8, 16), for arrays of sizes 10^8. Pick the scheduling policy you feel is appropriate.

5 Prefix Sum

Question: Implement a parallel function using OpenMP parallel loop constructs to compute the prefix sum of an array.
Question: Plot a speedup chart for different numbers of thread (1, 2, 4, 8, 16), for arrays of sizes 10^9. Pick the scheduling policy you feel is appropriate.

6 Merge Sort

Question: Implement a parallel function using OpenMP parallel loop constructs to perform merge sort on an array of integer.
Question: Plot a speedup chart for different numbers of thread (1, 2, 4, 8, 16), for arrays of sizes 10^9.