CSC4005/7007 High Performance Computing (2012/13)
Assignment 2 – Parallel Searching using OpenMP

This assignment builds on Assignment 1, please re-read carefully the specification given in Assignment 1. The primary aim is to design, implement and experiment with OpenMP programs to perform the same search process. You will be provided with three new test case folders, `test0`, `test1` and `test2`, on which to perform your experiments.

TEST0

In this test case you may assume that there is only one occurrence of the pattern in the text.

1. Run the program `searching_sequential.c` on `test0` as follows,
   `$time ./searching_sequential`
   Note the elapsed time taken and the number of comparisons reported (which may be negative because of integer overflow). The elapsed time will be used as a reference point for comparison with your OpenMP implementation.
2. Re-engineer the `searching_sequential.c` into an OpenMP program, `searching_OMP_0.c`, based around a *parallel for loop* in function `hostMatch`. **Only two shared variables may be set in the *for loop*: one to compute the total number of comparisons made; and one to record the position of the matching point.** Experiment with your program using differing numbers of threads and scheduling strategies. Ensure that the number of comparisons reported and the position of the pattern match the sequential program.
3. Using 8 cores in the debug_eth queue on the Dell cluster, run `searching_OMP_0.c` with 1, 2, 4, 8, 16, 32 and 64 threads.
   a. Draw a graph of the parallel speedup (PS) vs the number of threads. PS is the elapsed time taken by the sequential program divided by the elapsed time taken using P threads.
   b. Draw a graph of the parallel efficiency (PE) vs the number of threads. PE is simply PS/P.
   c. Draw a graph of the total number of comparisons vs the number of threads.
   d. Compute the single thread overhead (STO), where
   STO = 100*((ElapsedTime(OpenMP$_{simgle_thread}$)/ElapsedTime(Sequential))-1.

TEST1

In this test case you may also assume that there is one occurrence of the pattern in the text.

1. Run the program `searching_sequential.c` on `test1` and note the elapsed time taken and the number of comparisons reported.
2. Run your OpenMP program, `searching_OMP_0.c`, on the same test case using two threads.
3. Experiment with the two programs, adding extra debug statements as you deem necessary. On the basis of your experiments explain the difference in the observed performance.
4. Create an OpenMP program `searching_OMP_1.c` in which you should attempt to reduce significantly the number of comparisons performed. Describe informally the parallel algorithm used.

TEST2

1. Experiment with `searching_sequential.c` and `searching_OMP_1.c` on the final test case, `test2`. Vary the number of threads and add extra debug statements as you deem necessary. Comment on your results and on the characteristics of `test2`.
2. Create `searching_OMP_2.c` which should attempt to minimize the number of comparisons performed and correctly detect the position of the pattern as required by the specification in Assignment 1.
3. Using 8 cores in the debug_eth queue on the Dell cluster, run `searching_OMP_2.c` with 1, 2, 4, 8, 16, 32 and 64 threads. Report your results in a table with columns for PS, PE and total number of comparisons made. Comment on your results.

You should submit a CD containin*g your 3 OpenMP programs and a report addressing the above issues.* **Please provide a hard-copy of the report**.