Parallelization of Insertion Sort Using OpenMP

By- Hrithik Rai Saxena

Goal : Write a parallel program, which reads and sorts file sort.txt increasingly according to the 1st column (and 2nd if the values in 1st column are equal) to output file sorted.txt. Number of items (lines) is determined by reading (until EOF or error)

Given : Serial version of Insertion sort (with Index array and without)

Approach : Read the data by a single thread and determine the lowest (low) and highest (high) values. Split this range equally to all threads and let each thread sort one of the disjunct intervals – the resulting list is then given by the ordered sequence of sorted blocks

Timings : These are the timings for the task performed on a single machine on the "100k.txt" file with 1,4,8 and 12 threads respectively.

	------ Wall Times -----	
Num of Threads	**Calculation**	**Calculation+ Reading**
1	**7.545308**	**7.744148**
4	**0.910266**	**1.215617**
8	**0.228014**	**0.620118**
12	**0.114528**	**0.523619**

Observations :

1. Speed is directly proprtional to the number of threads.

2. Instead of sorting the large shared array, each thread has got its own private array to sort according to their range.

3. There was a sudden rise in speed as soon we introduced parallelism from 1 thread to 4.

4. Calculation time decreases as we introduce more threads but there is a certain delay in total time due to the sequential ordering of individual arrays by each thread by **pragma omp ordered.**

5. Its safe to assume that there is a limit till where we can go on with parallelising since introducing more and more threads wont necessarily guarantee speed due to arising communication bottlenecks and

ordered writing into the file.

6. Even better results are possible if we consider some more time efficient sorting techniques.