Homework 2

Student ID - 5296193

OpenMP implementation

Parallelizing using OpenMP

1. Data set is divided among all threads; each thread always gets the same data set every loop in each r-bit iteration as the OpenMP scheduling is done statically.
2. Each thread counts the number of input which have the same r-bit value. Prefix scan of the counts is done across threads. All the input is put back into a temporary data array in sorted order using counting sort. All these can be done in parallel and without any synchronization issues as each thread as its own data set.
3. Finally, the actual data and temporary data set is swapped.

Results for OpenMP:

OpenMP	**Number of threads**				
Time (in sec)	**1**	**2**	**4**	**8**	**16**
1M.txt	0.0786	0.0448	0.0289	0.0266	0.0358
10M.txt	0.7483	0.4190	0.3148	0.1878	0.1500
100M.txt	8.1973	7.0378	2.5226	1.4884	1.0774

MPI Implementation

Parallelizing using MPI

1. Data is divided among all the process; each thread does an enumeration sort for a r-bit value and puts into a local sorted array.
2. Each of the sorted local data from each process is then gathered into the ROOT using MPI_Gatherv().
3. Everything in each process is independent as there is no dependency or serialization, the only serial part is the communication which is inevitable in this case.

Results for MPI:

MPI	**Number of threads**				
Time (in sec)	**1**	**2**	**4**	**8**	**16**
1M.txt	0.0516	0.0307	0.0223	0.0200	0.0234
10M.txt	0.5649	0.3303	0.2087	0.1575	0.1371
100M.txt	6.2087	3.4805	2.0892	1.5231	1.0180