Assignment-Report Submission

Assignment-4

Submitted by: - Parth Parashar

I have completed all the given programs and all of them are working.

I had a very good learning experience with this assignment. I got to learn a lot about the MPI functions as well as bucket sort.

Bucket sort is a very efficient algorithm when it comes to sorting integer elements. This is evident from the run time of the algorithm itself. But when combined with parallel programming capabilities, this algorithm becomes even better saving a lot of precious memory as well as run-time is reduced a lot.

In this exercise, when I started with the bsort-file.c version, I got to learn about the file handling capabilities of C.

I got to learn a lot about the use of fread and fwrite operations which in all honesty were quite new to me. Being a java developer from the past 5 years now, transitioning back to the use of fread and write in c with extensive use of pointers wasn't an easy task but I managed to make it work.

The use of fseek and ftell to find out the number of elements present in the file was also really new to me but I was able to implement it properly.

The second program bsort-mpi.c was something that I was not really confident about. But following the steps given made it a little easier to implement.

I got to learn a lot about the use of MPI_Send and MPI_Receive functions along with the use of MPI File operations. It gave me a whole new insight on the use of MPI file operations

I ran into a couple of problems where when I used to run the program, it would throw an MPI not found exception along with segmentation faults.

All this was due to the use of faulty offsets. I rectified the faulty offset calculations and the program started running fine.

This version of bucket sort was even faster than the bsort-file.c version. This can be proved by looking at the run times of the programs.

Message Passing makes it incredibly easy to collect the sorted buckets which when combined with the merging of these buckets and then using bubble sort to sort it made it incredibly convenient. The use of MPI_Send and MPI_Receive was confusing at first as well but with the correct usage of offset, I was able to made it work.

As for the bsort-mpi2.c , the only changes that I had to made was the use of scatterv and gatherv functions.

The use of this made the program even more efficient.

The programs are running and showing the outputs on the screen as well so that it is easy to see the MPI in action.