CS546 HW 3

Design Report

Algorithm of Prefix Sum –

Step 1 – Initialize all the variables needed for program

Step 2 - Take the number of processors and number of elements of array via command line arguments

Step 3 - Generate the random numbers and store them in array and display them if rank of processor is 0.

Step 4 - Initialize the MPI code with MPI_Init () function

Step 5 - record the time with MPI_Wtime () function

Step 6 - Define the size and rank of processor with MPI_Comm_size () and MPI_Comm_rank () function respectively.

Step 7 – Use the MPI_Barrier () function for processor synchronization.

Step 8 - If rank of processor is zero,

A – Processor 0 will receive the local sum from other processors using MPI_Recv () function.

B – Processor 0 will add the local sum received from other processors to its own local sum.

Step 9 – If rank of processor is other than zero,

A - Current processor will calculate the local sum

B - Send the local sum to the processor 0 using MPI_Send () function.

Step 10 – record the time using MPI_Wtime () function.

Step 11 – If rank of the processor is 0,

A - Display the prefix sum.

B – Calculate & display the total time by taking difference between stop and start time

Step 12 – Use MPI_Finalize () function to end the MPI code

Explanation -

After initializing rank and size of processors, I checked whether the total size of array is multiple of total number of processors or not. For this, I calculated starting index of array for every processor and, also calculated the size of buffer i.e. total elements for each processor to calculate the local sum. By using the buffer size assigned to each processor, every processor calculates the sum of its assigned elements and send it to the processor 0.

For example, if array size of random numbers is 16 and total processors are 4, then each processor will compute sum of 4 elements and send it to the processor 0. At processor 0, it will calculate its own local sum. Then it will add the local sum received from other processors to its own local sum.

In case of array elements with, for example, 17, I assigned 12 elements to first 3 processors (4 elements to each processor) and last 5 elements to last processor by calculating mod of total array elements and total number of processors.

Hence, here I can **guarantee the correctness of my code** by calculating the prefix sum of elements whether they are multiple to number of processors or not.