lab 4

1 message

Satyam Nalkar <satyamnalkar20@gmail.com> Fri, 19 Sep, 2025 at 5:21 pm
To: Satyam Nalkar <satyamnalkar95@gmail.com>

```
#include <iostream>
#include <mpi.h>
#include <vector>
#include <algorithm>
using namespace std;

int partition(vector<int>& A, int start, int end) {
    int pivot = A[end];
    int i = start - 1;
    for (int j = start; j < end; j++) {
        if (A[j] <= pivot) {
            i++;
            swap(A[i], A[j]);
        }
    }
    swap(A[i + 1], A[end]);
    return i + 1;
}

void quickSort(vector<int>& A, int start, int end) {
    if (start < end) {
        int index = partition(A, start, end);
        quickSort(A, start, index - 1);
        quickSort(A, index + 1, end);
    }
}


int main(int argc, char *argv[]) {
    MPI_Init(&argc, &argv);
    MPI_Status status;
    int rank, size;

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    vector<int> big_data;
    int total_size = 20;
    int part;

    if (rank == 0) {
        big_data = {2, 9, 20, 35, 40, 1, 5, 3, 8, 31, 21, 22, 19, 12, 6, 7, 15, 32, 10, 12};
        part = total_size / size;


        for (int i = 1; i < size; i++) {
            MPI_Send(&big_data[i * part], part, MPI_INT, i, 1, MPI_COMM_WORLD);
        }



        vector<int> local_data(big_data.begin(), big_data.begin() + part);
        quickSort(local_data, 0, part - 1);
```

```
   MPI_Bcast(&part, 1, MPI_INT, 0, MPI_COMM_WORLD);
   //gathered_data.resize(total_size);
   vector<int>gathered_data(total_size);

   MPI_Allgather(local_data.data(),part,MPI_INT,gathered_data.data(),part,MPI_INT,MPI_COMM_WORLD);

   sort(gathered_data.begin(),gathered_data.end());

  //cout << "Rank " << rank << " sorted part: ";
  //  for (int num : local_data) cout << num << " ";
  //  cout << endl;

     /*for(int i = 0; i<local_data.size(); i++){
       cout << local_data[i] << " ";
       }
       cout << endl;*/


    /*
     cout << "sorted data:";
     for(int i = 0; i < gathered_data.size(); i++){
       cout << gathered_data[i] << " ";
       }
       cout << endl;
      */


   cout << "Rank " << rank << "final sorted array:"<< endl;;
   for(int i = 0; i<gathered_data.size(); i++){
        cout << gathered_data[i] << " ";
     }
      cout << endl;




}




else {
    part = total_size / size;
    vector<int> local_data(part);
    MPI_Recv(local_data.data(), part, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);


    quickSort(local_data, 0, part - 1);

    //cout << "Rank " << rank << " sorted part: ";
    //for (int num : local_data) cout << num << " ";
    //cout << endl;
    /* for(int i = 0; i<local_data.size(); i++){
        cout << local_data[i] << " ";
     }
      cout << endl;*/


   vector<int>gathered_data(total_size);

   MPI_Allgather(local_data.data(),part,MPI_INT,gathered_data.data(),part,MPI_INT,MPI_COMM_WORLD);

   sort(gathered_data.begin(),gathered_data.end());

   cout << "Rank " << rank << "final sorted array:"<< endl;;
```

```
    for(int i = 0; i<gathered_data.size(); i++){
        cout << gathered_data[i] << " ";
     }
      cout << endl;

  }

  MPI_Finalize();
  return 0;
}
```