Assignment 3 Report

Cory Christensen

January 31, 2020

1 Implementation

To implement this code I used process zero to create a list of 4 numbers for each process excepting zero. The code will generate random numbers and send four to each process. These processes will sort their individual sets using bubble sort. Process zero then gathers in the first number from each then will pull the minimum from the list and request the next value from the process that minimum value belonged to. This processes continues until each process has run out of numbers. When a process runs out of numbers it sends a -1 and finalizes.

The expected output is a list of unsorted values, followed by the sorted list of each process. While process zero is pulling the minimum values it says what the index of that value is in the sorted list and what the actual value is. After the process finishes it displays the sorted list.

2 Code

```
#include <iostream>
#include <mpi.h>
#include <unistd.h>
#include <stdlib.h>

//#include "/usr/local/include/mpi.h"
#define MCW MPI_COMM_WORLD

using namespace std;

void bubbleSort(int *arr, int size){
    bool sorted = false;
    int temp;
    while(!sorted){
        sorted = true;
        for(int i = 0; i< size- 1; i++){
            if(arr[i+1] < arr[i]){
                sorted = false;
                //cout << "swap " << arr[i] << " and " << arr[i+1] << endl;
                temp = arr[i];
                arr[i] = arr[i+1];
                arr[i+1] = temp;
            }
            //arr[i] = arr[0];
        }
    }
    //return arr;
}

int main(int argc, char **argv){

    int rank, size;
    int data;

    MPI_Init(&argc, &argv);
    MPI_Status mystatus;
    MPI_Comm_rank(MCW, &rank);
    MPI_Comm_size(MCW, &size);

    //number of elements for each process to take on
    const int numSplit = 4;
    int split[numSplit];

    const int numData = size*numSplit - numSplit;
    int bigData[numData];

    //int sortOpt[size];
    if(!rank){
        cout << "The unsorted list: " << endl;
        for(int i=0;i<numData/numSplit;++i){
            for(int j = 0; j < numSplit; j++) {
                split[j] = rand()%100;
                cout << split[j] << " ";
            }
            MPI_Send(split, numSplit, MPI_INT, i + 1, 0, MCW);
        }
        cout << endl;
        //request data
        sleep(1);
        //MPI_Barrier(MCW);
```

```
//get initial list to merge
int tempArr[size-1];
for(int i = 0; i < size-1; i++) {
    data = -1;

    MPI_Send(&data, 1, MPI_INT, i+1, 0, MCW);
    MPI_Recv(&data,1,MPI_INT,MPI_ANY_SOURCE,0,MCW,&mystatus);
    tempArr[i]= data;

}
int min = tempArr[0];
int minProcess;
int minIndex;
bool assigned = false;
for(int i = 0; i < numData; i++) {

    minProcess = 1;
    assigned = false;
    for (int j = 0; j < size - 1; j++) {
        if(tempArr[j] != -1 && (tempArr[j] < min || !assigned)){
            min = tempArr[j];
            minProcess = j+1;
            minIndex = j;
            if(!assigned){
                assigned = true;
            }
        }
    }
    bigData[i] = tempArr[minIndex];
    cout << "The " << i << " element of " << bigData[i] <<" was added

    data = -1;
    MPI_Send(&data, 1, MPI_INT, minProcess, 0, MCW);
    MPI_Recv(&data,1,MPI_INT,MPI_ANY_SOURCE,0,MCW,&mystatus);
    tempArr[minIndex] = data;
    //sleep(1);
}

sleep(1);
cout << "The sorted list:" << endl;
for(int i = 0; i < numData; i++){
    cout << bigData[i] << " ";
}
cout << endl;
```

```
    }
    else {
        MPI_Recv(split,numSplit,MPI_INT,MPI_ANY_SOURCE,0,MCW,&mystatus);
        sleep(1);
        cout << "rank " << rank;
        for(int i = 0; i < numSplit; i++){
            cout << " " << split[i];
        }
        cout << endl;


        //sort group and send back
        bubbleSort(split,numSplit);
        //qsort(split,split+numSplit);
        //MPI_Barrier(MCW);

        for(int i = 0; i < numSplit; i++){
            //anything can be recieved this is just meant to wait for process
            MPI_Recv(&data,numSplit,MPI_INT,MPI_ANY_SOURCE,0,MCW,&mystatus);
            data = split[i];
            MPI_Send(&data, 1, MPI_INT, 0, 0, MCW);
        }
        MPI_Recv(&data,numSplit,MPI_INT,MPI_ANY_SOURCE,0,MCW,&mystatus);
        data = -1;
        MPI_Send(&data, 1, MPI_INT, 0, 0, MCW);
    }




    MPI_Finalize();

    return 0;
}
```

3 Run Commands

```
mpic++ Assn3.cpp
mpirun -np 8 -oversubscribe a.out
```

4 Output

```
The unsorted list:
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23
```

```
rank 2 93 35 86 92
rank 4 90 59 63 26
rank 5 40 26 72 36
rank 6 11 68 67 29
rank 3 49 21 62 27
rank 7 82 30 62 23
rank 1 83 86 77 15
The 0 element of 11 was added to the sorted list
The 1 element of 15 was added to the sorted list
The 2 element of 21 was added to the sorted list
The 3 element of 23 was added to the sorted list
The 4 element of 26 was added to the sorted list
The 5 element of 26 was added to the sorted list
The 6 element of 27 was added to the sorted list
The 7 element of 29 was added to the sorted list
The 8 element of 30 was added to the sorted list
The 9 element of 35 was added to the sorted list
The 10 element of 36 was added to the sorted list
The 11 element of 40 was added to the sorted list
The 12 element of 49 was added to the sorted list
The 13 element of 59 was added to the sorted list
The 14 element of 62 was added to the sorted list
The 15 element of 62 was added to the sorted list
The 16 element of 63 was added to the sorted list
The 17 element of 67 was added to the sorted list
The 18 element of 68 was added to the sorted list
The 19 element of 72 was added to the sorted list
The 20 element of 77 was added to the sorted list
The 21 element of 82 was added to the sorted list
The 22 element of 83 was added to the sorted list
The 23 element of 86 was added to the sorted list
The 24 element of 86 was added to the sorted list
The 25 element of 90 was added to the sorted list
The 26 element of 92 was added to the sorted list
The 27 element of 93 was added to the sorted list
The sorted list:
11 15 21 23 26 26 27 29 30 35 36 40 49 59 62 62 63 67 68 72 77 82 83 86 86 90 92 93
```