cs6550

Brett Bonar

September 21, 2018

1 Implementation

The master process with rank 0 splits the input array into equally sized arrays to be consumed by every process.

MPI Scatter is used to send the sublists to each process and an MPI Gather is used to collect the sorted sublists into the master process.

The master process then merges the sorted lists and outputs the final result.

2 Code

```
#include <iostream>
#include <mpi.h>
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include <vector>
#include <limits>
#define MCW MPI_COMM_WORLD

using namespace std;

int main(int argc, char **argv){
  int rank, size;
  int data;

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

  int length = 64;
  int sortSize = length / size;

  std::vector<int> sortedList(length);
  std::vector<int> aggregateList(length);
  std::vector<int> sortList(sortSize);
  if (rank == 0)
  {
    for (int i = 0; i < length; i++)
    {
      aggregateList[i] = std::rand() % length;
    }
  }

  MPI_Scatter(aggregateList.data(), sortSize, MPI_INT,
    sortList.data(), sortSize, MPI_INT, 0, MCW);

  std::sort(sortList.begin(), sortList.end());

  MPI_Gather(sortList.data(), sortSize, MPI_INT,
    aggregateList.data(), sortSize, MPI_INT, 0, MCW);

  if (rank == 0)
  {
    std::fill(sortedList.begin(), sortedList.end(),
      std::numeric_limits<int>::max());

    // Aggregate list will contain size sublists of length sortSize
    for (int i = 0; i < length; i += sortSize)
    {
      // Iterate over each item in sublist
      for (int j = 0; j < sortSize; j++)
      {
        sortedList.insert(
          std::upper_bound(
            sortedList.begin(),
            sortedList.end(),
            aggregateList[i + j]),
        aggregateList[i + j]);
      }
    }

    for (int i = 0; i < length; i++)
    {
      std::cout << sortedList[i] << std::endl;
    }
  }

  MPI_Finalize();

  return 0;
}
```

3 Compile and Run Commands

```
mpic++ Assignment3/Assignment3.cpp -o Assignment3/run.out
mpirun -np 8 Assignment3/run.out
```

4 Output

```
1
2
2
5
6
```

```
6
6
7
9
10
13
13
13
13
14
17
17
20
20
20
23
24
24
26
26
26
27
27
28
29
30
31
33
35
35
35
37
38
39
39
40
41
41
41
43
43
44
46
49
50
50
50
51
51
52
52
54
55
56
58
59
60
```

```
61
63
```