HW04 - Bitonic Sorting

CS5500

Chelsi Gupta

September 28, 2018

1 Steps for Implementation

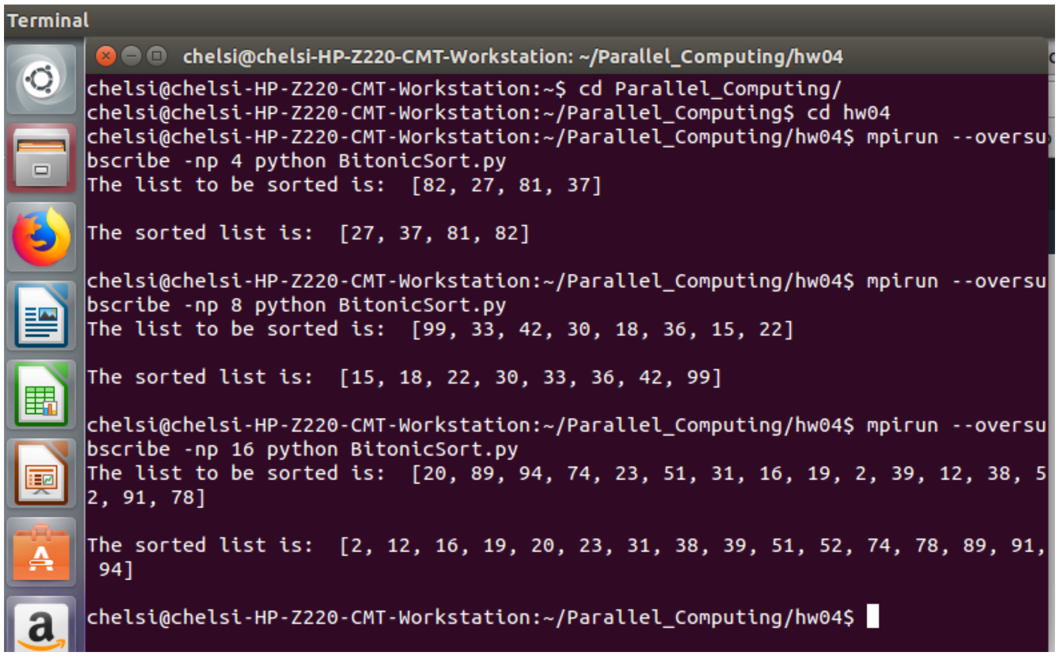
- I have made 3 functions in my program:
  - **cube:**
    Input- i(dimension of cube), sendData(actual data of process)
    Output- recvData(data process receives from cube exchange), dest(rank of process with which cube exchange is happening)
    Purpose- This function implements the cube network and determines the rank of the process with which the current process is exchanging the data(dest). The current process sends its data to 'dest' and also receives dest's data.
  - **compare_ascending:**
    Input- rank(current process),dst(cube ex-changer),rdata(data received),rbuf(actual data)
    Output- rbuf(swapped data)
    Purpose- This function arranges the data in ascending order i.e. if the rank of the process is less than its cube ex-changer(dst) then the data it should have must be smaller than dst's data.
  - **compare_descending:**
    Input- rank(current process),dst(cube ex-changer),rdata(data received),rbuf(actual data)
    Output- rbuf(swapped data)
    Purpose- This function arranges the data in descending order i.e. if the rank of the process is less than its cube ex-changer(dst) then the data it should have must be greater than dst's data.
- First created an empty list 'ListtoSort' which is filled by process 0 with a random sample of values ranging from 0 to 100 and size of sample is equal to number of processes.
- Process 0 then scatters the 'ListtoSort' giving 1 element to each process which is stored in their 'rbuf'.

- Calculated 'k', k-1 is the highest dimension of cube we need to achieve in order to accomplish the sort.
- Iterated i from 0 to k-1 and nested a for loop for j inside that, applied cube operation passing j and rbuf(of current process).
- Calculated a number 'x', if 'rank/x' is an even number then the process compares and swaps in ascending order otherwise in descending order.
- The I applied barrier so that all the processes apply a particular cube operation on the same level.
- At last process 0 gathered the sorted data from all processes and displayed the sorted list.
- I compiled and ran the program using "mpirun –oversubscribe -np 8 python BitonicSort.py".

2 Code

```
from mpi4py import MPI
import math
import random
comm = MPI.COMM_WORLD


def cube(i, sendData):
        size = comm.Get_size()
        rank = comm.Get_rank()
        mask = 1
        mask = mask << i
        dest = rank ^ mask
        #dest is the process we want to exchange the data with
        comm.send(sendData, dest=dest)
        recvData = comm.recv(source=dest)
        return recvData, dest


def compare_ascending(rank,dst,rdata,rbuf):
        if(rank<dst and rdata<rbuf):
        #rdata is the data process received and rbuf is the data it has right now
                rbuf = rdata
        if(rank>dst and rdata>rbuf):
                rbuf = rdata
        return rbuf


def compare_descending(rank,dst,rdata,rbuf):
        if(rank<dst and rdata>rbuf):
        #rdata is the data process received and rbuf is the data it has at this stage
                rbuf = rdata
        if(rank>dst and rdata<rbuf):
                rbuf = rdata
        return rbuf


size = comm.Get_size()
rank = comm.Get_rank()
ListtoSort = []
if rank == 0:
        ListtoSort = random.sample(range(100), size)
        print ("The list to be sorted is: ",ListtoSort,"\n")
rbuf = comm.scatter(ListtoSort,root=0)

k = int(math.log2(size))
for i in range(0,k,1):
        for j in range(i,-1,-1):
                rdata,dst = cube(j,rbuf)
                x = 2**(i+1)
                if (rank//x)%2 == 0:
                        rbuf = compare_ascending(rank,dst,rdata,rbuf)
                else:
                        rbuf = compare_descending(rank,dst,rdata,rbuf)
                comm.Barrier()

sortedList = comm.gather(rbuf,root=0)
if rank == 0:
        print ("The sorted list is: ",sortedList,'\n')
```

3 Output



Terminal
chelsi@chelsi-HP-Z220-CMT-Workstation: ~/Parallel_Computing/hw04

```
chelsi@chelsi-HP-Z220-CMT-Workstation:~$ cd Parallel_Computing/
chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing$ cd hw04
chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw04$ mpirun --oversu
bscribe -np 4 python BitonicSort.py
The list to be sorted is:  [82, 27, 81, 37]

The sorted list is:  [27, 37, 81, 82]

chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw04$ mpirun --oversu
bscribe -np 8 python BitonicSort.py
The list to be sorted is:  [99, 33, 42, 30, 18, 36, 15, 22]

The sorted list is:  [15, 18, 22, 30, 33, 36, 42, 99]

chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw04$ mpirun --oversu
bscribe -np 16 python BitonicSort.py
The list to be sorted is:  [20, 89, 94, 74, 23, 51, 31, 16, 19, 2, 39, 12, 38, 5
2, 91, 78]

The sorted list is:  [2, 12, 16, 19, 20, 23, 31, 38, 39, 51, 52, 74, 78, 89, 91,
 94]

chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw04$ 
```

Figure 1: My Output