CS336 : Parallel & Distributed processing

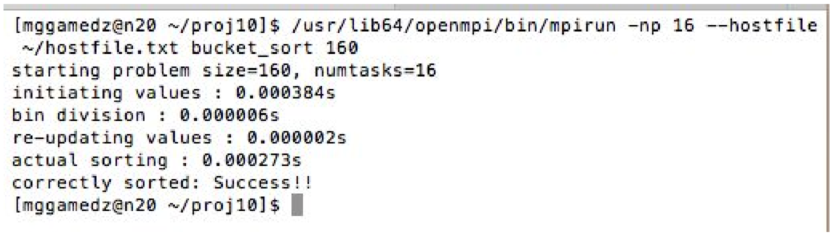
Project 10 Report

Summary of tasks

For this project the main aim was to write a program that implements the bucket sort algorithm with MPI onto a program to sort an array of integers using multiple, distributed processors.

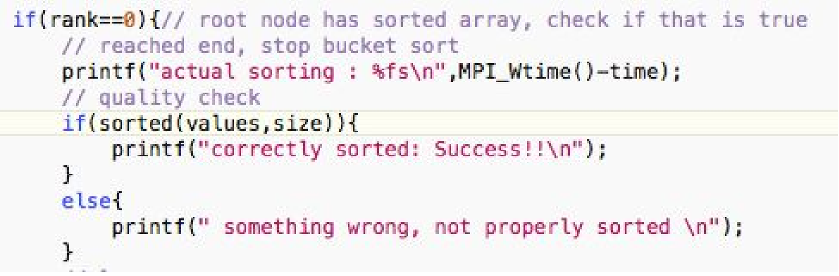
Tasks

4. Running my function, I got a result similar to this attached snippet for a problem size of 160:



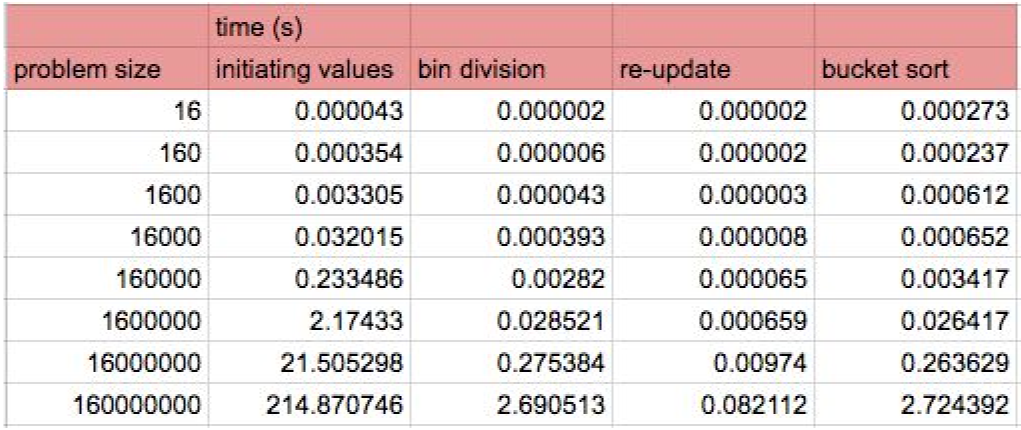
```
[mggamedz@n20 ~/proj10]$ /usr/lib64/openmpi/bin/mpirun -np 16 --hostfile
 ~/hostfile.txt bucket_sort 160
starting problem size=160, numtasks=16
initiating values : 0.000384s
bin division : 0.000006s
re-updating values : 0.000002s
actual sorting : 0.000273s
correctly sorted: Success!!
[mggamedz@n20 ~/proj10]$ 
```

My code also has a validator condition which checks that my sort algorithm worked well.



```
if(rank==0){// root node has sorted array, check if that is true
    // reached end, stop bucket sort
    printf("actual sorting : %fs\n",MPI_Wtime()-time);
    // quality check
    if(sorted(values,size)){
        printf("correctly sorted: Success!!\n");
    }
    else{
        printf(" something wrong, not properly sorted \n");
    }
```

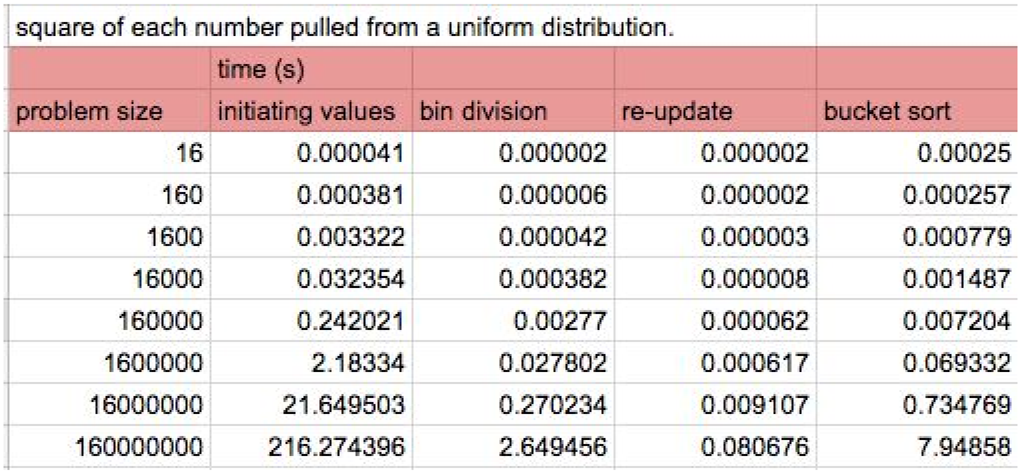
5. Timing the different section of the MPI bucket sort, I got this result:



	time (s)			
problem size	initiating values	bin division	re-update	bucket sort
16	0.000043	0.000002	0.000002	0.000273
160	0.000354	0.000006	0.000002	0.000237
1600	0.003305	0.000043	0.000003	0.000612
16000	0.032015	0.000393	0.000008	0.000652
160000	0.233486	0.00282	0.000065	0.003417
1600000	2.17433	0.028521	0.000659	0.026417
16000000	21.505298	0.275384	0.00974	0.263629
160000000	214.870746	2.690513	0.082112	2.724392

Initiating the values in the array is the most expensive time step for my sort program.

6. Squaring the randomly generated value increases the initiation time by a small frction, thus slowing the whole process. The rest of the other processes are unchanged by this, as can be seen below. However the actual sort is slowed down by this addition as can be seen from my data:



square of each number pulled from a uniform distribution.				
	time (s)			
problem size	initiating values	bin division	re-update	bucket sort
16	0.000041	0.000002	0.000002	0.00025
160	0.000381	0.000006	0.000002	0.000257
1600	0.003322	0.000042	0.000003	0.000779
16000	0.032354	0.000382	0.000008	0.001487
160000	0.242021	0.00277	0.000062	0.007204
1600000	2.18334	0.027802	0.000617	0.069332
16000000	21.649503	0.270234	0.009107	0.734769
160000000	216.274396	2.649456	0.080676	7.94858

This was an interesting discovery on my end. The prompt for this task assumed this was caused by a load imbalance which slows down the sort and this indeed caused a significant slow down in the program.

Extensions

None

Collaborators

I worked alone. Thanks to Martin for help debugging.

.