Tim McMullen

159.735 Assignment 2, Parallel Bucket Sort

The Parallel bucket sort, sorts a list of numbers by putting numbers into a bucket based on the numbers value, and the range of the bucket, once placed in a bucket a quick sort is performed on each bucket sorting them from lowest to highest. After the sort each set of numbers is joined in a way that once connected they flow from lowest to highest.

To implement this in parallel we make it that each processor is itself a bucket. to start the master node will send each node a equal portion of the list of numbers, we achieve this by using the scatter command, once received the node will then perform a bucket sort, sorting each number into its corresponding bucket which are stored as an array, we then find the displacement between each bucket within the array to find how many numbers each bucket currently has. Once the displacement is found an All to all is performed to find the required buffer size, to store the incoming buckets then each said buffer is created. Once created a All to allv is called passing the buckets to each node, the all to allv is required as each bucket passed is of a variable size and so needs to be able to change as required. As each node now has all the numbers assigned to it we are able to do a quick sort using the systems Qsort command to order them from lowest to highest. Finally a gather is performed to determine the amount of incoming data from each node, this is then used with a gatherv to collect the data and store them into the final bucket, witch is ordered from highest to lowest within the set range of values

Time taken

amount of numbers	1 processor	2 processor	3 processor	4 processor
10,000,000	9.344049	5.073806	3.776180	3.058293
100,000,000	111.281794	56.844244	41.758511	33.896611

These results show that by parallizing the bucket sort we are able to greatly reduce the time taken to perform the bucket sort and as such also the total time taken to sort all the numbers into the required order. The time reductions are as expected as by adding an additional node we are able to split the required processing, but as the number of processors and numbers to sort are increased so is the communication time between each of them