Concurrent Programming

Lab 1

Vignesh Iyer

The code has two parts in the file mysort.cpp. The merge sort algorithm is a divide and conquer algorithm. The algorithm divides the given array input into two parts which is the left and right part. Then the parts are sorted individually and finally both of them sorted together. The code allocates memory to both the left and the right part of the array and sorts them individually using the merge sort algorithm and then again it combines the individual left and right part of the algorithm. Then when the argument passed indicates merge sort. It proceeds to call a structure which stores the number of threads, the lowest and the highest index of the given array i.e. in each of the particular thread. Then each of the thread run parallel with the array being divided according to number of given threads. This function can take any threads and the size of the array as long as it does interfere with the heap memory of the given system. Then in the final step all the threads are merged again combining and again using merge sort to sort the final array. And the memory is freed in order to avoid memory buffer overflow.

The bucket sort algorithm, divides the given array in the buckets which are equal to the number of threads. Then each of the bucket is allocated a memory according to the given structure which gives the thread number and is implemented as a list. The bucket size is determined based on the modulus operation of the total elements to the number of thread. If the remainder is zero then the bucket size is equal to the ratio of the total elements to the number of threads/buckets. If the modulus is a non-zero number, then the bucket size is equal the number of elements to (no of threads-1). Thus each bucket is then sorted using the insertion sort algorithm due to the fact that it is more stable and occupies less memory as compared to quick sort as is faster. As well as due to fact that insertion sort is better when

the elements are added in a sequential manner. The elements are sorted in each buckets using the total buckets divided by the element and the ratio value in integer determines the bucket number. The insertion sort will first insert the first element and proceed to compare the incoming element with the first and will put it if the incoming element is less than the first one. Again it recursively checks for the next element and compares with the elements inside the bucket. Then a lock is put in each thread to ensure no memory is accessed in the process which could cause data race. Thus then each of threads are joined and the final array is given as an output.

Steps to run:

1. Run make
2. The execution is: ./mysort inputfile –o outputfile –t Num_Threads –alg=bk → For bucket sort.
3. The execution is: ./mysort inputfile –o outputfile –t Num_Threads –alg=ms → For merge sort.

Difficulties Faced:

Some of the difficulties where the dynamic memory allocation. It had to ensure the size of the bucket for bucket sort is dynamic rather than being fixed. Also for merge sort the elements which were not in the extra i.e. each of the thread having equal amount of elements was to be ensured it had to allotted a thread in order to avoid segmentation faults or stack memory stash memory smashing which is when the program tries to access more memory than it is being allocated.

Also another issue for not using parallel quick sort is due to the fact that the reference element in a quick sort is a random one which when enters in the particular part of the given part of array tends to be difficult to make it fast. Also since the quick sort is a tree algorithm, traversing back has additional overhead. This makes parallel implementation on quick sort to be difficult.

Thus these were the difficulties faced.

References:

[1] https://www.gribblelab.org/CBootCamp/7_Memory_Stack_vs_Heap.html

[2] https://stackoverflow.com/questions/736920/is-there-ever-a-good-reason-to-use-insertion-sort

[3] https://www.geeksforgeeks.org/merge-sort/

[4] https://www.geeksforgeeks.org/insertion-sort/

[5] https://stackoverflow.com/questions/1345670/stack-smashing-detected

[6]https://forum.learncodethehardway.com/t/stack-smashing-detected-unknown-terminated-aborted-core-dumped-error/2535

[7]https://cboard.cprogramming.com/c-programming/92701-help-***-stack-smashing-detected-***.html