Omeiza Olumoye

CMSE 822 **Parallel Computing** Project

October 29, 2021

Parallel Quicksort

Table of Contents

1 Abstract **2**

2 Methodology **2**

3 Tools **2**

4 Benchmarking **2**

5 References **3**

1 Abstract

Quicksort algorithm is a divide and conquer sorting approach where by an element from an unsorted array is chosen as the pivot and the other elements around the pivot are reorganized. This is done in such a manner that when scanning the array from the right and left, elements that are smaller and larger than the pivot are swapped respectively. Ideally, we want the element at the pivot to be in the middle so the array can be divided about the pivot and a recursive quicksort algorithm be performed on each of the subarrays. The best runtime for the quicksort is $\mathcal{O}(n \log n)$ and the worst time is $\mathcal{O}(n^2)$ for an array of size n [1]. I want to implement the quicksort algorithm in a shared memory environment and document the speedup that can be achieved.

2 Methodology

I would run the parallelized version of the quicksort algorithm my local computer and the HPCC to document the time saved as opposed to running quicksort in a serial environment The different strategies I would implement for picking the pivot as as follows:

(a) Pick the first element

(b) Pick the last element

(c) Pick the median element

(d) Use a parallel prefix operation

3 Tools

The program will be written in C++ and the parallel portion of the code implemented with OpenMP.

4 Benchmarking

I want to know if the runtime can be reduced to $\mathcal{O}(\log n)$ or better still how much improvement can be achieved. Tables and figures of the performance of different number of threads will be compared; it will be interesting to know the memory usage since a recursive approach will be implemented. Also, the implementation will be and load balancing as the number of threads being used is increased.

5 References

[1] Victor Eijkhout, Edmond Chow, and Robert Van de Geijn. Introduction to high performance scientific computing. pages 347–351, 2021.