High Performance Computing: Project Preproposal

Evaluating the Performance of Various Sorting Algorithm Implementations

Fabian Weiland; CSC746; FALL 2024

Sorting algorithms are fundamental in computer science, with naive approaches exhibiting time complexities of O(n^2) and O(n*log n). Non-optimized implementations often consist of repetitive, independent operations executed serially, presenting opportunities for enhancement through vectorization and parallelization techniques. Vectorization performs the same operation on multiple data points simultaneously, which is still a serial optimization strategy. In contrast, true parallelization leverages multiple computational cores.

This study compares several sorting algorithms, including bubble sort, merge sort, insertion sort, and potentially quick sort if time permits. The focus will be on automatically vectorized implementations and those utilizing varying degrees of concurrency via the Open Multi-Processing API (OpenMP), benchmarking them against their basic counterparts.

The experiments measure MFLOP/s rates and analyze parallelization speedup across concurrency levels (1, 4, 16, and 64 threads).

Each implementation is structured as a function that takes the problem size n as a parameter, representing the size of the input array. Within the function, the array is then sorted in ascending order.

Conducted on NERSC-9 Perlmutter, with problem sizes ranging from 2048, 4096, 8192, 16384, 32768 both vectorization and parallelization are expected to outperform the basic implementations. As the speed up depends on problem size and concurrency level, I expect greater parallel speedup for larger problem sizes as more work becomes available for parallelization.