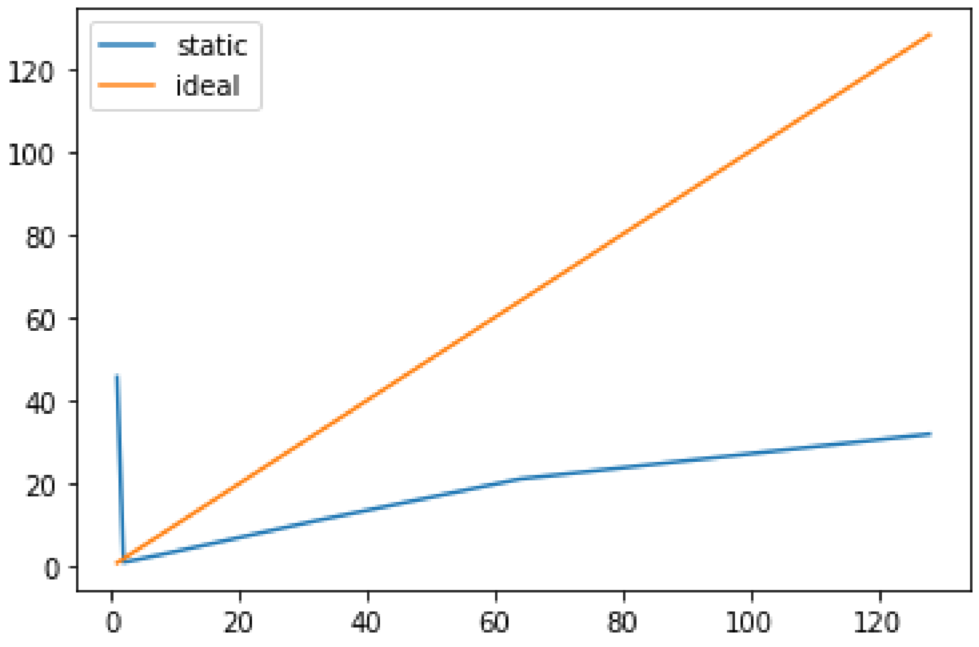
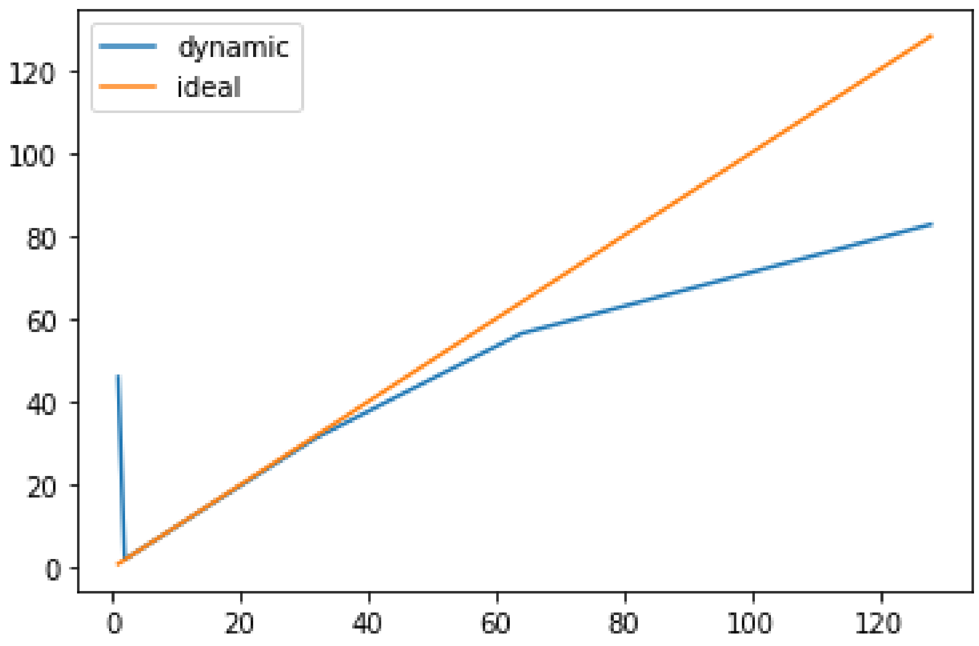
CMDA 3634 PRO04

Judson Murray

November 2022

1 Performance Write-up





2 Conceptual Write-up

In order to keep the read-write shared variables (optimal c, optimal cost, and tuples checked) thread safe, I declared new thread versions of them inside the parallel pragma to keep track of computation that each thread is performing, and at the end implemented a critical region in order to efficiently run the program in parallel. Inside the critical region, each thread would enter once per number of threads, and the computations found inside the parallel loop are then compared inside the critical region.

In the blog post, dynamic scheduling is described as each thread going back and ”requesting” a new chunk of iterations to perform after the first, and thus the size of that chunk and number of iterations to perform is determined after every successive iteration. It is mentioned that dynamic scheduling for loops is better when each iteration of the loop has a different computational cost, which is the case in our example, as the cost computed for each data point in set will be different. As a result, there is not a work load imbalance between threads, as shown in figure 5, with the first 60-70 threads taking on most of the work, therefore you do not have to wait for the more heavily loaded threads to finish their work to arrive at the output.