Homework 2 - ID1217

Youssef Taoudi, yousseft@kth.se

February 12, 2019

Abstract

Performance report for implementation of Matrix and Palindromes using OpenMP.

1 Matrix

I made two matrix programs, one that used the openMP critical section functionality and one that used reduction compare with customized min and max functions.

Figure 1: Speedup times for matrix using reduction

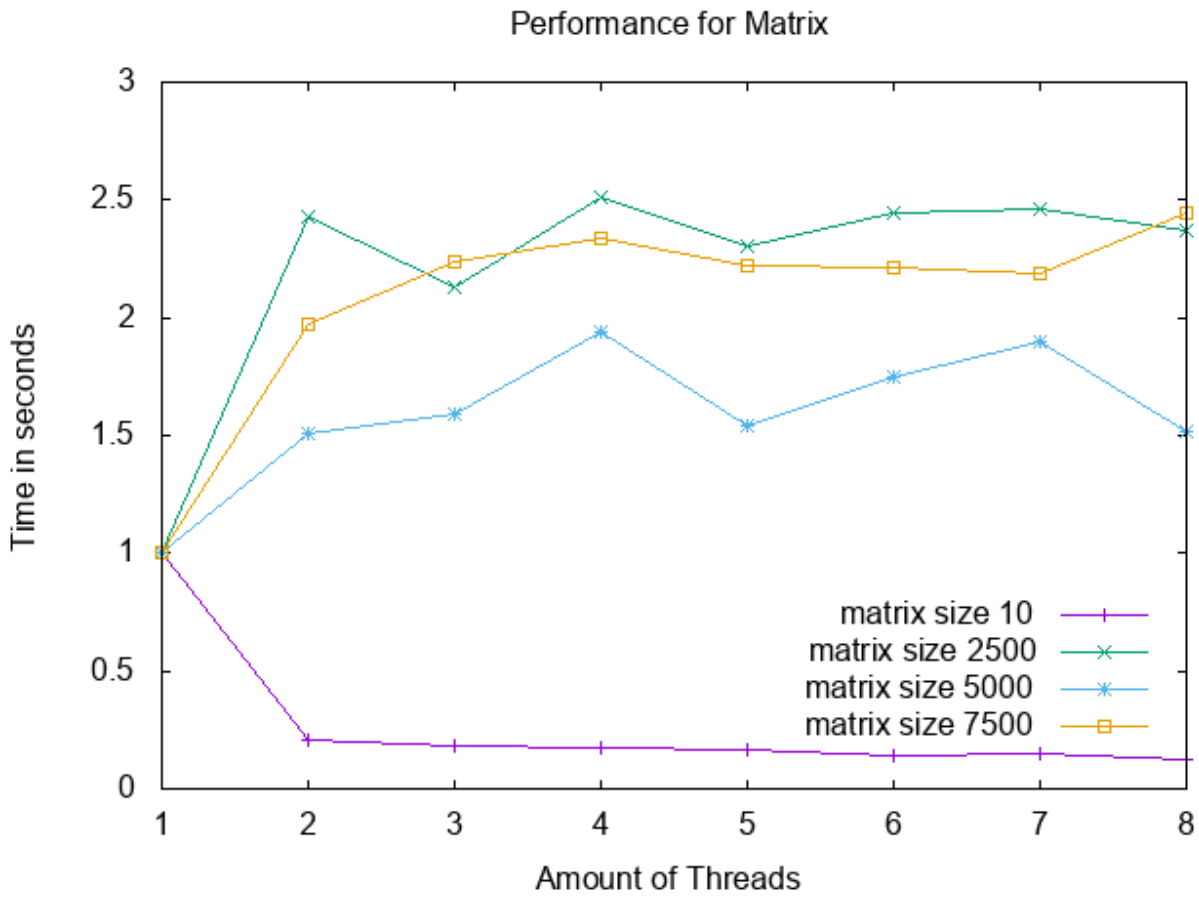


Figure 1 showcase the speedup when running the matrix using OpenMP reduction on different sizes.

The figure above shows the speedup for different amount of threads on four sizes using reduction. The purple line shows the speedup when running in parallel on a matrix of size 10. In this case, there was a slowdown in performance most likely due to the fact that the overhead of the context switches are greater than the actual speed of running the program sequentially. In the other cases the speedup is higher. For some reason, a matrix of size 2500 is faster than that of 5000 and 7500. The peak for speedup is when running on 4 threads, my guess is because my computer ran the program on 4 cores.

Figure 2: Speedup times for matrix using locks

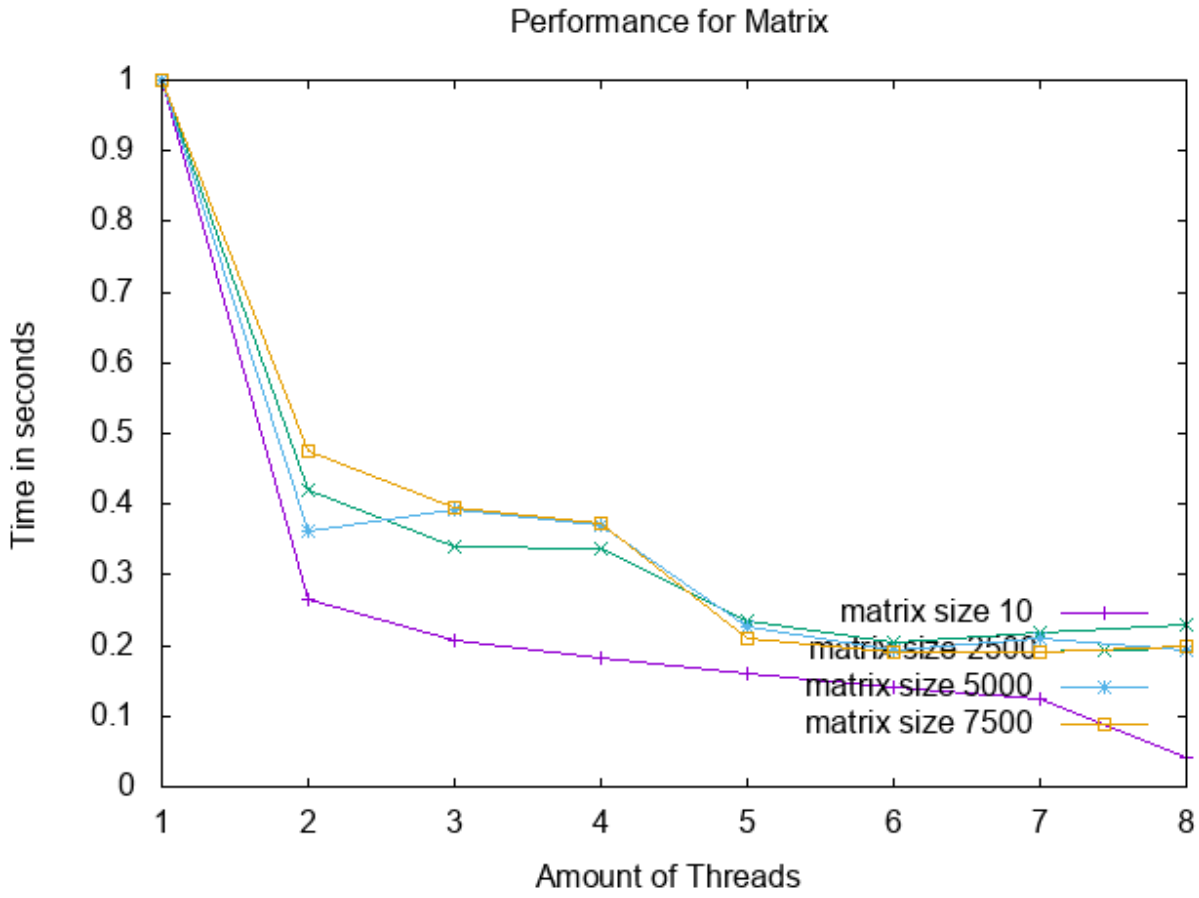
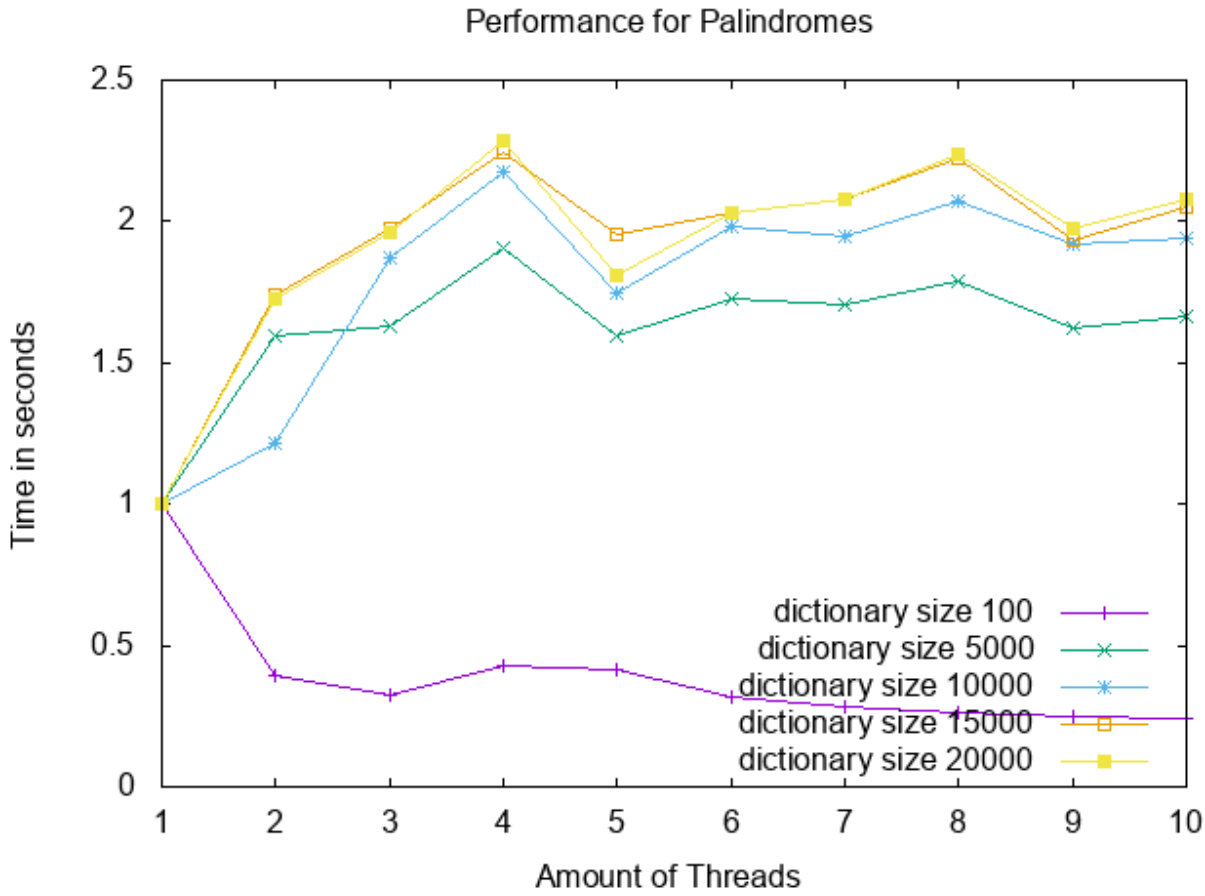


Figure 2 shows the performance of running the matrix with critical sections for different sizes. As you can see, the overhead of the locks that are used internally in the critical section functionality slow down the program for parallel execution and no matter the size, the speedup is always lower than sequential execution.

2 Palindromes

Similarly to Matrix, I executed the program for five different sizes of dictionaries for 10 different number of threads as shown in figure 3.

Figure 3: Speedup times for dictionary of different sizes



In this case, the speedup gets better and better for greater sizes except for the case between 15000 and 20000. For a dictionary of size 100, running the program in parallel is inefficient meaning there is a certain threshold where running the program sequentially is better than in parallel (on my machine). Also worth noting is that palindromes do not use locks or reduction, there are no shared variables or race conditions. Once again, four threads is the most efficient.