1- Prefix sum

In the prefix sum, I have divided the first part of for loop into n chunks and have run it with a scheduling policy. As can be seen from the graphs not much speedup can be gained in the parallel code because after the first part the end array element of the array is added to a temporary array. Then again the next part is made to run in parallel to add the missing part in the following array elements.

2- merge sort loop

For the merge sort loop, it is normal that we could achieve speedup as the number of threads increases. I used omp schedule static which was distributing the work equally among the threads as the number of threads increases the threads are less work to do and could finish a little faster their task, however, I used OpenMP single at the end to make sure only the thread that got freed first to call the merge sort function and do the remaining work. I think it could be faster if I used probably OpenMP dynamic it would help get better speedup if integrated into the right place.

3- merge sort task

We were able to get speedup a the number of threads increased. Splitting the tasks increases parallel efficiency which leads to better performance. The maximum speedup achieve was approximately 3. I will say the merge sort task performs a little better We can see at 16 threads merge sort task performs better than the merge sort loop.