(Reduce)
What is your speedup for 16 threads?
The speedup for threads is approximately two. Within tasking construct I have stored result in an partial_sum array based on thread ID of currently executing thread. This avoids data race as each thread writes in its own variable in the array. In the end the final result is calculated sequentially by reducing the partial_sum array.

(Merge sort)
What is your speedup for 16 threads?
The speedup for 16 threads is one. In this code, I have used tasking constructs on merge sort recursively. A random cut off value has been used to avoid a very small array size for merge. On reaching this cut off, merge sort is calculated for the entire array of the given cut off size.

(LCS)
What is your speedup for 16 threads?
The speedup for 16 threads is around 2. In LCS, the rows and columns can be calculated parallely except for the diagonal value. Hence used tasking construct to achieve the same.

(Bubble sort)

What is your speedup for 16 threads?
The speed up for 16 threads is 16.On extracting the dependencies, it can be seen that parallelism can be achieved by calculating the alternate values based on the iteration’s even or odd value.