## NO Threading

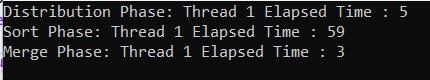
Distribution Phase: Elapsed Time: 2

# Sort Phase: Elapsed Time: 11

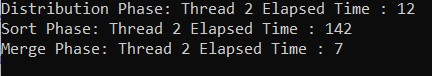
Merge Phase: Elapsed Time: 1

**Threading With a single spinlock.**

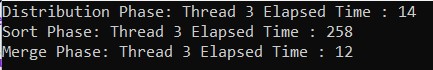
## For Thread count: 1



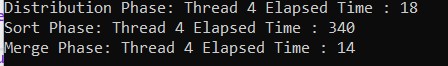
## For Thread Count: 2



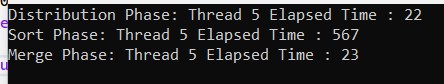
## For Thread Count: 3



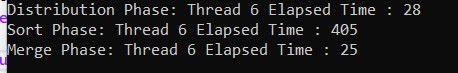
# For Thread Count: 4



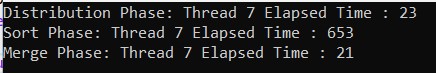
## For Thread Count: 5



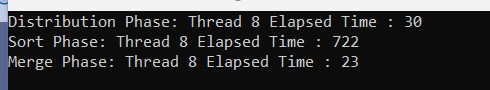
## For Thread Count: 6



# For Thread Count: 7

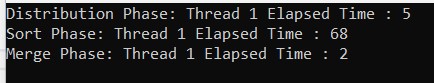


## For Thread Count: 8

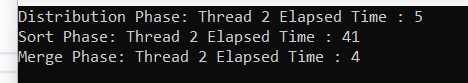


**Threading With a fine-grain lock.**

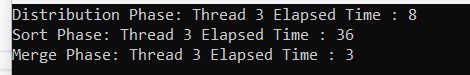
## For Thread count: 1



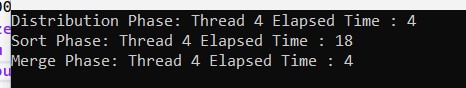
## For Thread Count: 2



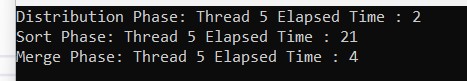
## For Thread Count: 3



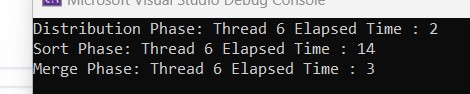
## For Thread Count: 4



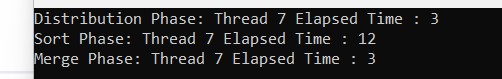
# For Thread Count: 5



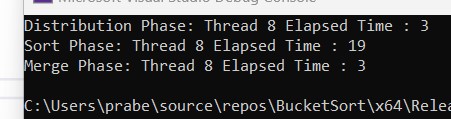
## For Thread Count: 6



## For Thread Count: 7



## For Thread Count: 8



From the above observation we can say that fine grain locking is a more efficient way of locking than single spin locking. In single spin locking, all thread shares same lock so one variable control all the thread so, it took more time to process data, whereas in fine-grain locking there is list of locker and lock is based on index so there is little chance to lock other threads by one that lock the data update.