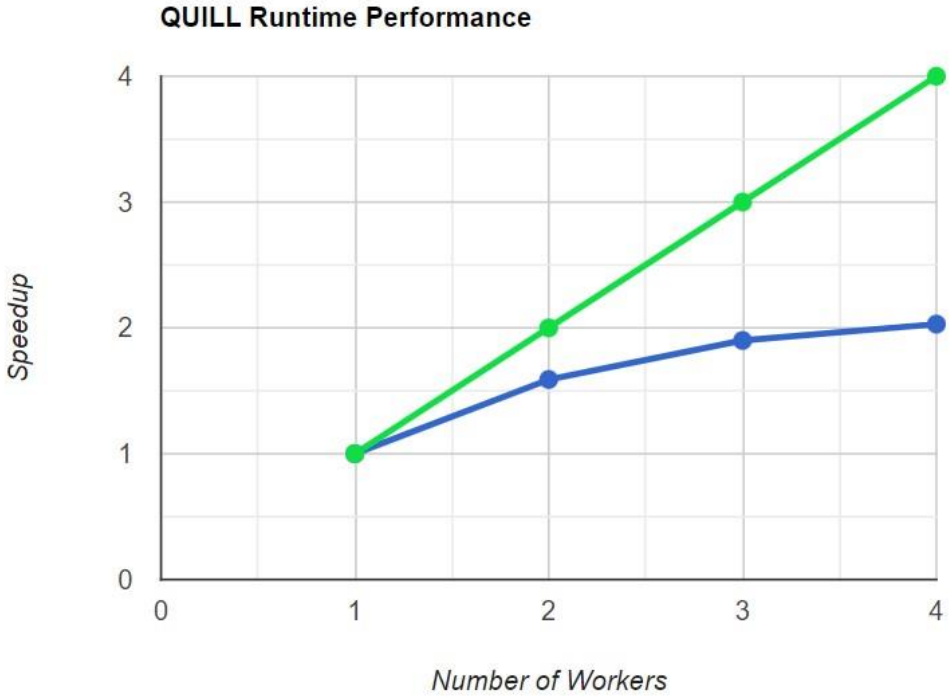


```
QUILL_WORKERS=1 ./nqueens 12
OK
NQueens(12) Time = 1.784883sec
QUILL_WORKERS=2 ./nqueens 12
OK
NQueens(12) Time = 1.125127sec
QUILL_WORKERS=3 ./nqueens 12
OK
NQueens(12) Time = 0.939995sec
QUILL_WORKERS=4 ./nqueens 12
OK
NQueens(12) Time = 0.877472sec
```

QUILL Runtime Speedup Statistics

QUILL_WORKERS= 2 Speedup = 1.59

QUILL_WORKERS= 3 Speedup = 1.90

QUILL_WORKERS= 4 Speedup = 2.03

The green line represents the linear speedup but the blue line represents the actual speedup which I am getting from the QUILL runtime which is a help first work stealing approach. The speedup lies in the “Sublinear” region as it doesn’t rise linearly with the number of workers but falls. However, the execution time decreases with an increase in the number of QUILL workers proving the power of parallelism as due to the work stealing approach, most of the time workers are busy with their own job(improving locality) and spending less time for stealing purposes) giving both high performance and productivity. It’s performance can be further improved by reducing the number of tasks created, and other optimization regarding the mutex locks for pop operation instead of using swap & compare instructions as in HCLib runtime.