2. Aggregate measurements
Analysis:
(a) Run xhpl as described above with 1, 2, 4, and 8 threads.
Record the "real" time as well as the GFLOPs values in your writeup.
OMP_NUM_THREADS=1, real: 2m26.781s, Gflops: 4.444e+01.
OMP_NUM_THREADS=2, real: 1m29.240s, Gflops: 8.530e+01.
OMP_NUM_THREADS=4, real: 1m1.898s, Gflops: 1.523e+02.
OMP_NUM_THREADS=8, real: 0m48.721s, Gflos: 2.458e+02.

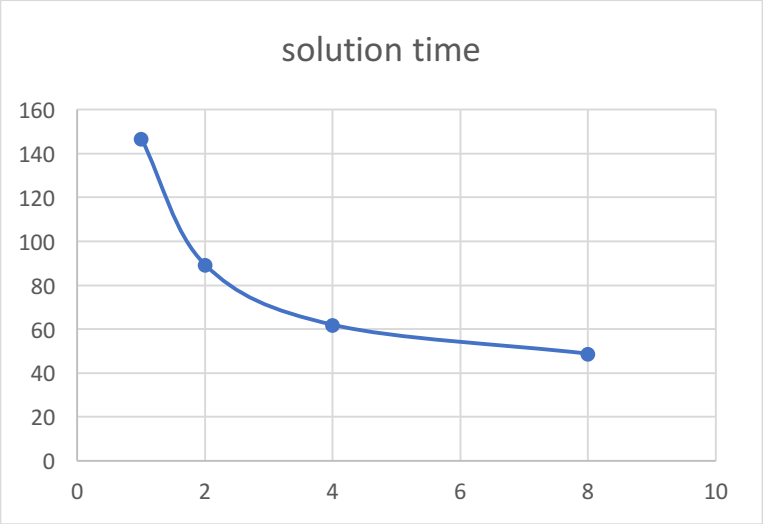
(b) List the speedup of 2, 4, and 8 threads (versus 1 thread).

	1	2	4	8
Speedup: S=told/tnew	1	1.64	2.37	3.01

(c) List the parallel efficiency of 2, 4 and 8 threads.

	2	4	8
Parallel efficiency: Ep=Sp/p	0.82	0.59	0.38

(d) Does this benchmark show strong scaling? Why or why not?
Yes this benchmark shows the solution time has decreased as the processors increased.



(e) Do the results gathered contain enough info to say if the benchmark exhibits weak scaling?
No, the results cannot gather enough info to prove that the benchmark exhibits weak scaling. The problem size is fixed for the whole system but not fixed for per processor and even if we manually calculate, the assumption that the job is evenly divided into each of the processor is not proven. So, it is hard to say that the benchmark exhibits weak scaling.

3. Profiling with perf
Questions:

(a) What function accounts for most of the run time?

67.73% xhpl [.] dgemm_kernel

(b) What instruction is reported as taking the most time?

1.37 | vmovup (%r15),%ymm3

vector move up

(c) What effect discussed in class might cause the answer to the previous question to not be the exact instruction causing the slowdown?

In class we discuss about the skid that cause the slowdown. When doing the instruction, a sudden call from the kernel may cause the kernel to ski for a while and making the perf measure not accurate. In general, memory access is always the instruction that is taking the most time, but the skid might happen and cause the perf to measure so accurate that the instruction a few lines before that would be the actual one.