Q2

a. We do the experiment on the COE machine with 24 cores and 1,000,000 total darts.

# of threads	Pthreads RT(ms)	Omp RT(ms)	Pthreads Pi	Omp Pi
1	101.19	59.28	3.142368	3.140828
2	541.10	682.69	3.144876	3.140532
4	559.97	477.13	3.142280	3.142772
8	588.00	496.21	3.142396	3.137960
16	697.13	545.17	3.139588	3.143468

Something weird just happens here. I do not find any speed up in my programs either pthreads or openmp. On the contrary, when I increase the number of threads from 1 to 2, I found that the running time is slowed by nearly 400%. First, I thought that was caused by the mutex on the update of the global variable. However, after I remove the mutex code, the running time is still increasing.

Moreover, the running time of pthreads is increasing while I increase the number of threads from 2 to 16. So I guess that is caused by some overhead of threads creating or the joining part of the main thread.

b. The running time also increases after we change the number of threads from 1 to 2. However, the difference between openmp and pthreads is that the running time of openmp does not increase straightforwardly from threads number 2 to 16. It decreases when we change the number of threads from 2 to 4. And the running time of 4 threads is the fastest between thread number 2, 4, 8, 16. Note that in the calculation of Pi, I use not only the “omp parallel” for parallel calculation but also the “omp parallel for” for the parallel loop.

c. For **strong scaling**: we calculate Pi using the fixed number of darts, which is 1,000,000 in the above experiment. As we discussed above, the speedup is even negative as we increase the number of threads. So we guess that the speedup is limited by the overhead of thread creation. As for Amdahl's law, there is even no part that could not be paralleled, so ideally, the speedup should be linear.

# of threads	# of darts	Pthreads RT(ms)	Omp RT(ms)
1	1,000,000	99.83	61.30
2	2,000,000	1087.69	1820.88
4	4,000,000	2222.95	1844.49
8	8,000,000	4640.39	3943.56
16	16,000,000	10257.89	8627.12

For **weak scaling**: we increase the number of darts based on the number of threads. Except for the change from thread number 1 to 2, all the other speedup is nearly linear, which is a reflection of Gustafson’s law. Moreover, there is also a strange situation similar to strong scaling. That is, the running time nearly does not increase on the omp program even if we increase both the number of threads and the number of darts. So, I guess the 4 threads may be the optimal number of threads on the omp program due to some reason.