Qiyuan Qiu 02/06/2014 CSC458

1. Speed up Curve
The macine I am using for this experiment is node2x12x1a.

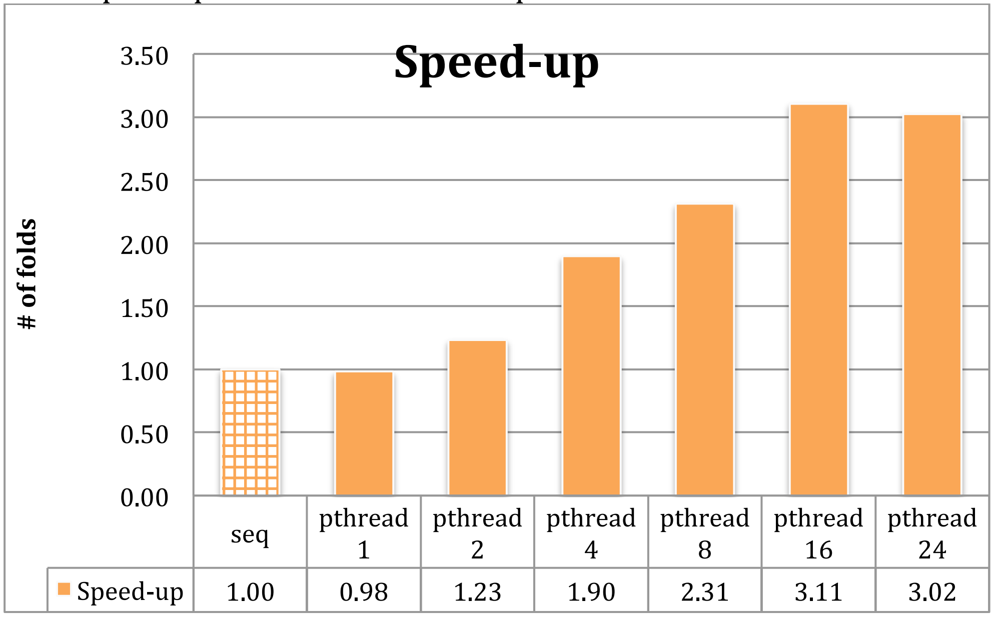
Experiments are performed on 2/5/2014.

The "seq" column is the experiment result for SOR sequential program. The sequential program has been run for ten iterations. The value in the column has unit second. The average run time for the sequential is 1.02 seconds. The chart below was constructed using this data.

Similarly, column three to eight are results for pthread version SOR. The number after "pthread" indicates how many threads were used when running.

	seq	pthread 1	pthread 2	pthread 4	pthread 8	pthread 16	pthread 24
iter 1	0.98	1.03	0.74	0.42	0.49	0.31	0.35
iter 2	1.03	1	0.82	0.43	0.47	0.31	0.34
Iter 3	0.97	1	0.89	0.69	0.46	0.31	0.34
iter 4	1.03	0.99	0.74	0.67	0.49	0.31	0.33
iter 5	1.03	1.06	0.78	0.52	0.47	0.33	0.34
iter 6	1.03	1.06	0.89	0.68	0.49	0.31	0.33
iter 7	1.04	1.06	0.9	0.44	0.29	0.3	0.35
iter 8	1.03	1.06	0.88	0.68	0.39	0.47	0.34
iter 9	1.04	1.06	0.74	0.43	0.39	0.31	0.33
iter 0	1.04	1.07	0.9	0.43	0.48	0.33	0.33
average	1.02	1.04	0.83	0.54	0.44	0.33	0.34
variance	0.00064	0.000966	0.0051	0.015	0.0043	0.0025	6.22222E-05
speed-up	1.00	0.98	1.23	1.90	2.31	3.11	3.02

The following is the speed up chart for the above experiments.

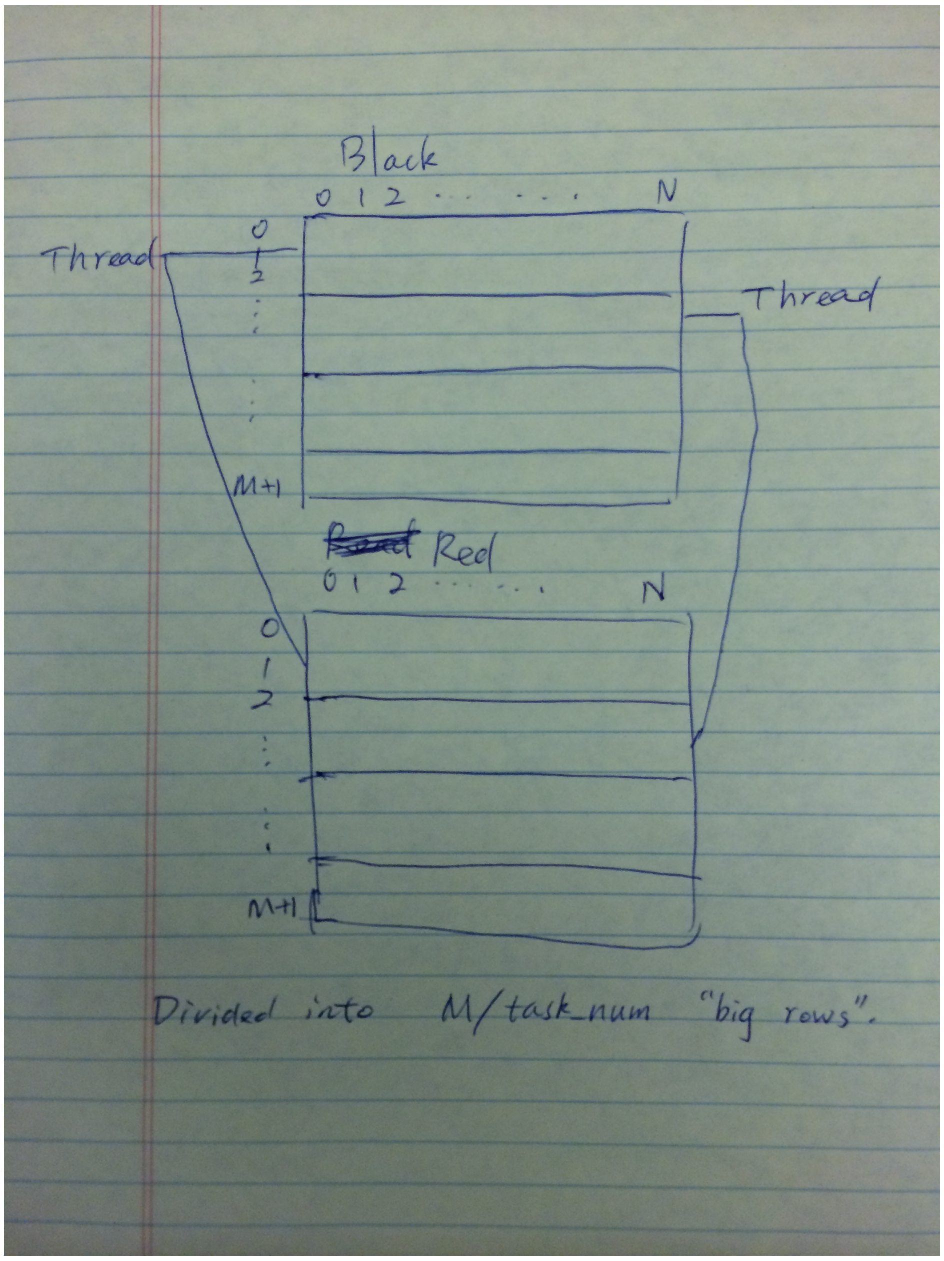


The multi-thread version does not have a speed up when using only one thread. Overall the more threads, the more speed up we can have. But the speed up we get after 16 threads stops to grow. The largest speed up "jump" comes from switching from eight threads to sixteen threads.

2. Data Partition
In the given program, the matrix is divided into two sub matrixes.
Two sub matrixes are of the same size (M+2) * (N+1). One is named Black and the other named Red.

Each thread actually does the update in both Black and Red.

Lets first focus on Black. In work_thread function, each thread will calculate its own begin and end. This effectively cut the Black in to M/task_num many rows.



Just like this picture, both matrixes are divided into "big rows" and each thread is in charge of updating one "big row".

This sort of division is based on the begin position. If the begin is an odd number, the thread will be in charge of odd “big row” update. Otherwise the thread will be in charge of even “big row” update.

Same thing happens for the Red matrix. Notice that if one thread is in charge of some particular “big row” it does the update for both the same “big row” in Black and Red matrixes.