hp394_hw1

1.Code Description

The hp394_mm_rbyc do the matrix-matrix multiplication by row-by-column method, it take one parameters from the command line which is the dimension. After the multiplication, it can write the program running time and the dimension information into the rbyc_elapsed_time.csv.

The hp394_mm_tile do the matrix-matrix multiplication by blocked method, it take two parameters from the command line which is the dimension and the tile size. After the multiplication, it can write the program running time and the dimension information into the tile_elapsed_time.csv.

The hp394_mm_pt do the matrix-matrix multiplication by pthread method, it take two parameters from the command line which is the dimension and the thread number. After the multiplication, it can write the program running time and the dimension information into the pt_elapsed_time.csv.

2. Benchmark:

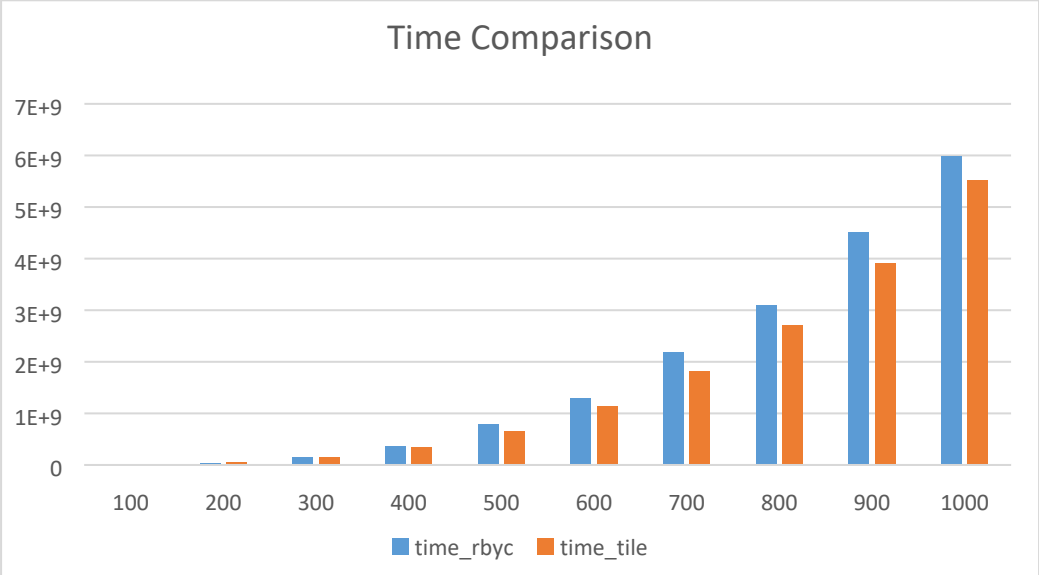

Fig-1 time comparison between the row-by-column method and blocked method

dim_n	time_rbyc(ns)	time_tile(ns)
100	4752647	5124557
200	38095182	41036151
300	149784613	141688028
400	351868490	332720713
500	781900981	653085209
600	1292352865	1136432343
700	2177289814	1803955539
800	3084791408	2694609687
900	4497876996	3904776951
1000	5988781383	5514193502

Tab-1 running time of the row-by-column method and blocked method

From the Fig-1, we can see that the block method has a better performance than the row-by-column method because the blocked method improve the operation rate of cache.

For the performance comparison among different dimension of tiles, we first set the dimension of the matrix is 1000, then we change different tile

size, then we get a table.

tile	elapsed time(ns)
5	5840655147
10	5340180875
20	5250138696
40	5031012396
50	4957490839
100	5113064947
200	5259529765
500	5935605064

Table 2 the elapsed time vary from different tile

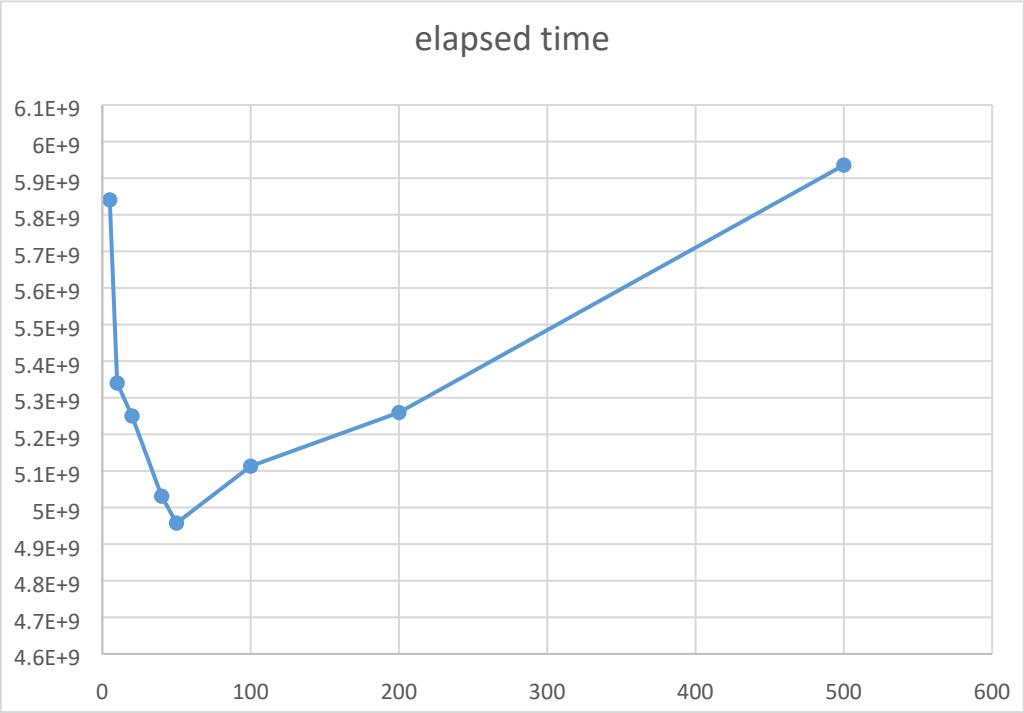

Fig-2 the elapsed time varying from different tile

From the Fig-2, we find that, with the increasing of tile size, the elapsed

time first decrease and then it increase. The best performance appeared when tile size is appropriate to the square root of the matrix dimension.

For the performance comparison among different thread number, we first set the dimension of the matrix is 1000, then we change different thread number, then we get a table.

thread_number	elapsed time(ns)
2	5415536361
3	3677972696
4	2677625699
5	2117904489
6	3403857856
7	3860767797
8	3665988286
9	3694940270
10	3608024249

Table 3 the elapsed time vary from different thread number

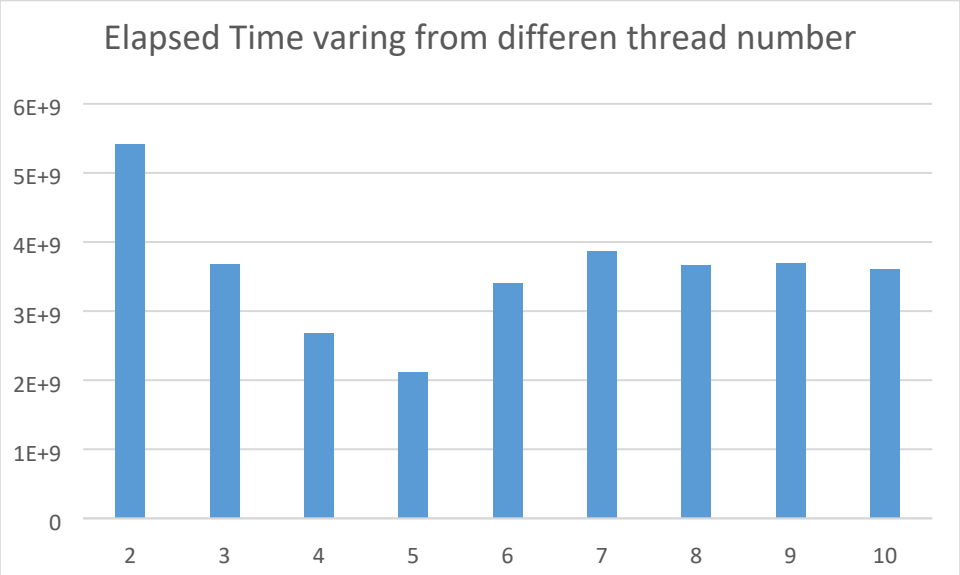

Fig-3 the elapsed time varying from different thread number

From the Fig-3, we find that, with the increasing of thread number, the elapsed time first decrease and then it increase. When the thread number is greater than a value, the extra time consuming of switching context of the program become an import reason to impact on the performance of the calculation. In this case, the best thread number is 5.