CS542200 Parallel Programming

Homework 4: Blocked All-Pairs Shortest Path

Implementation

- How do you divide your data?
  - 把每一格交給一個 thread 去做
  - 同一個 block 內的，使用 share memory 共享計算需用到的 block

Phase	Grid dim	Block dim	Share mem	Return case
One	(1, 1)	(B, B)	B x B	
Two	(round, 2)	(B, B)	B x B x 2	blockIdx.x==Round
Three	(round, round)	(B, B)	B x B x 2	blockIdx.x or blockIdx.y==Round

  - 把需要計算的表格依照 row 對切

Phase	Grid dim	Block dim	Share mem	Return case
One	(1, 1)	(B, B)	B x B	
Two	(round, 2)	(B, B)	B x B x 2	blockIdx.x==Round
Three	(round/2, round) (round, round/2)	(B, B)	B x B x 2	blockIdx.x or blockIdx.y==Round

- How do you implement the communication?
  - 各自算完所有值，不須溝通

Single-GPU		No need to communication
Multi-GPU / Single node	Omp	No need to communication
Multi-GPU / Two node	Mpi	No need to communication

  - 各自算完一半值，把計算完的結果傳給對方

Single-GPU		No need to communication
Multi-GPU / Single node	Omp	cudaMemcpyPeer
Multi-GPU / Two node	Mpi	cudaMemcpy(with MPI_Send/ MPI_Recv)

- What's your configuration?
  - pitch_n = B x Round ，以此來取記憶體，在計算邊界的時候就不須先判斷是否超出記憶體空間
  - 使用 cudaMallocPitch 時，寬度增加，故推測資源使用量上升，所以能

用的 Block Factor 值下降

- Block Factor 越大會越快，越大越快，但不能超過 32
- 對調 kernel 計算的 x 跟 y，大概可以讓速度快一半，跟記憶體讀取有關

GPU / node	Dev mem	Dev size	Block Factor eg. 16x16	
Single / Single	cudaMallocPitch cudaMemcpy2D	pitch x pitch_n	32，n>32 n/3，else	Exchange x and y When kernel computing
Multi / Single	cudaMallocPitch cudaMemcpy2D	pitch x pitch_n	16，n>16 n/3，else	Exchange x and y When kernel computing
Multi / Two	cudaMalloc	Pitch_n x Pitch_n	32，n>32 16，else	Exchange x and y When kernel computing

- 當 Multi-gpu 需要溝通時，為了降低傳輸資料量，選擇使用 cudaMalloc
- 但是由於為了讓對調 kernel 計算的 x 跟 y，大概可以讓速度快一半，跟記憶體讀取有關，但是為了配合傳輸資料時的連續性，對調時，改成(round/2, round)-> (round, round/2)可保證資料正確性，但推測因溝通佔據大部分時間，無法增加效率

GPU / node	Dev mem	Dev size	Block Factor eg. 16x16	
Single / Single	cudaMallocPitch cudaMemcpy2D	pitch x pitch_n	32，n>32 n/3，else	Exchange x and y When kernel computing
Multi / Single	cudaMalloc	Pitch_n x Pitch_n	32，n>32 16，else	Exchange x and y When kernel computing
Multi / Two	cudaMalloc	Pitch_n x Pitch_n	32，n>32 16，else	Exchange x and y When kernel computing

- Briefly describe your implementation

Phase	Share mem	__syncthreads	If … break	computing
One	自己		threadIdx.y or threadIdx.x > n	
Two	Row or Col 自己		blockIdx.x==Round	
Three	Row and Col		blockIdx.x or blockIdx.y==Round	

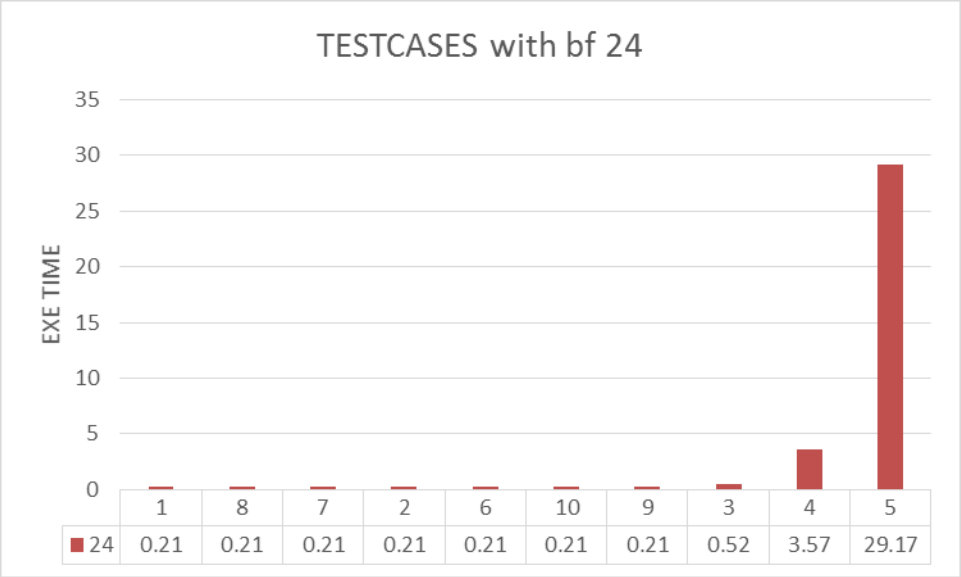
Experiment & Analysis

- Performance Metrics:
  - cpu time(初始化表格)/ IO time/ Communication time(MPI SEND/RECV、

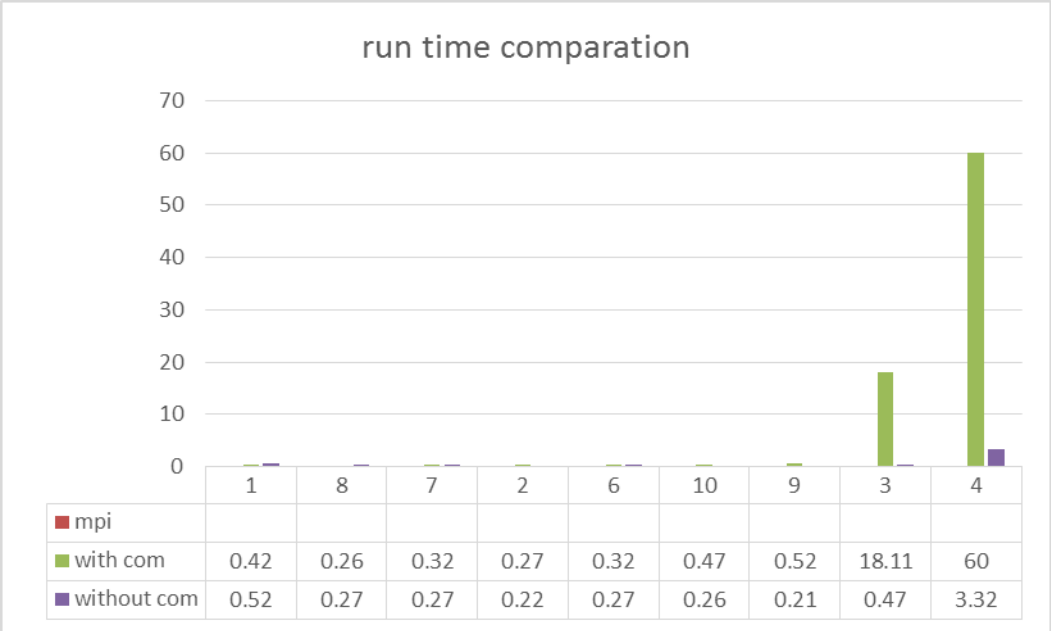
cudamemcpypeer…)：用 clock_gettime 來算，單位為 ns 會轉換成 s

- GPU time：用 cudaEventRecord 來算 GPU time
- nvprof

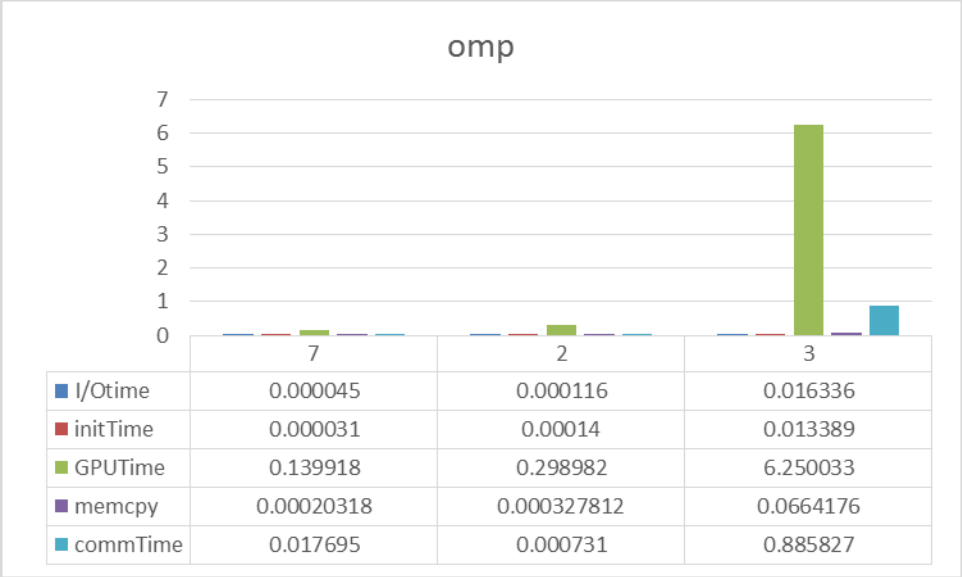
- Weak Scalability & Time Distribution
  - Total exe time：整體時間，隨資料量增加而增加
    - Single-GPU



    - Multi-GPU implementation in the single

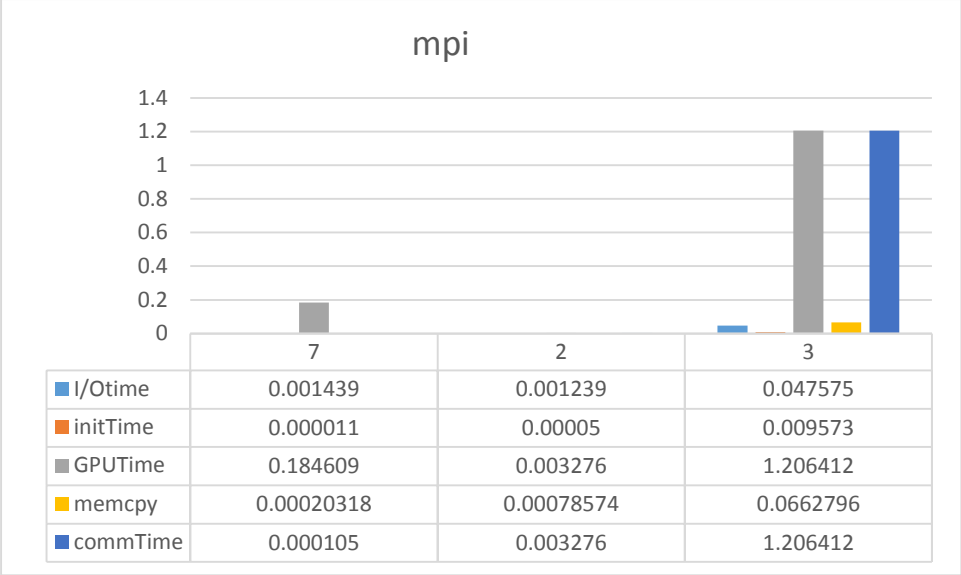


	1	8	7	2	6	10	9	3	4
mpi									
with com	0.42	0.26	0.32	0.27	0.32	0.47	0.52	18.11	60
without com	0.52	0.27	0.27	0.22	0.27	0.26	0.21	0.47	3.32

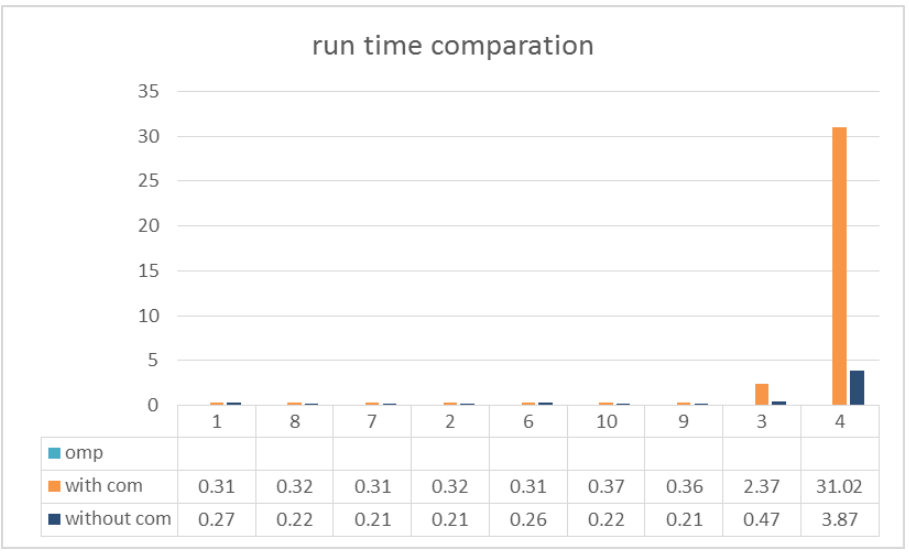


	7	2	3
I/Otime	0.000045	0.000116	0.016336
initTime	0.000031	0.00014	0.013389
GPUTime	0.139918	0.298982	6.250033
memcpy	0.00020318	0.000327812	0.0664176
commTime	0.017695	0.000731	0.885827

- Multi-GPU implementation with MPI

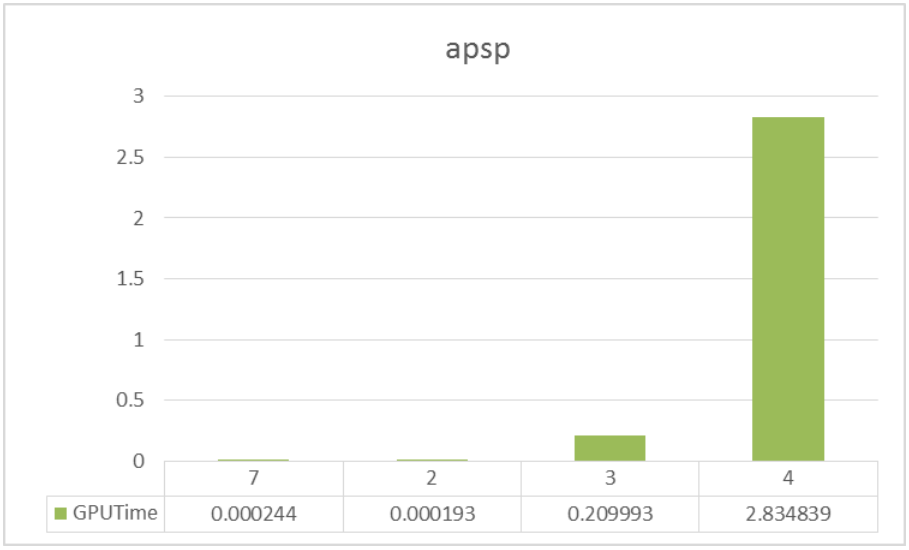


	7	2	3
I/Otime	0.001439	0.001239	0.047575
initTime	0.000011	0.00005	0.009573
GPUTime	0.184609	0.003276	1.206412
memcpy	0.00020318	0.00078574	0.0662796
commTime	0.000105	0.003276	1.206412

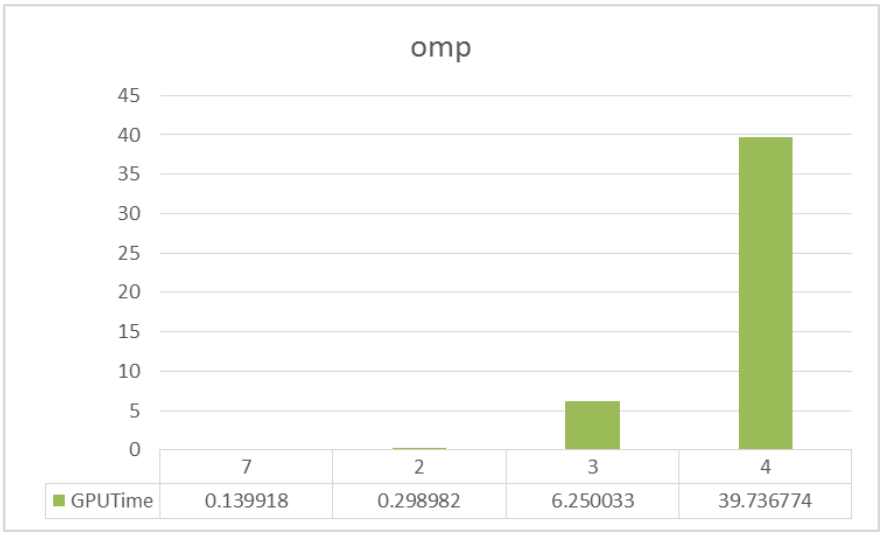


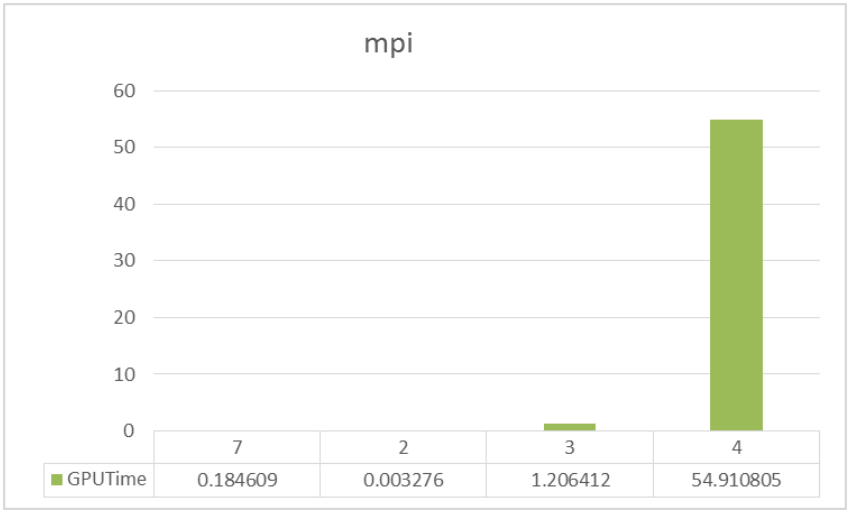
	1	8	7	2	6	10	9	3	4
omp									
with com	0.31	0.32	0.31	0.32	0.31	0.37	0.36	2.37	31.02
without com	0.27	0.22	0.21	0.21	0.26	0.22	0.21	0.47	3.87

- Computing

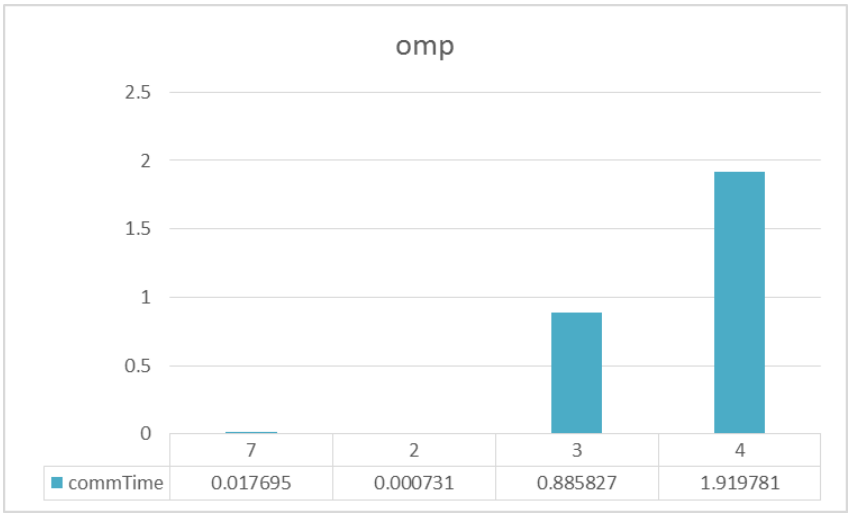


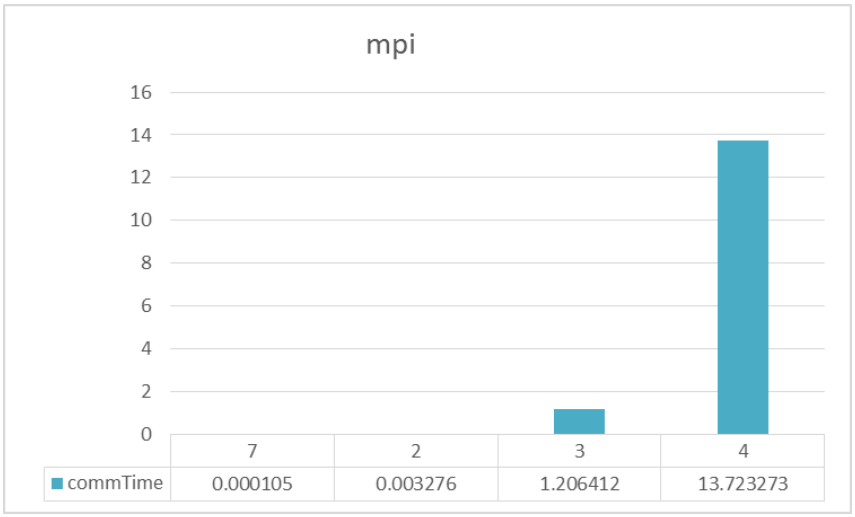
	7	2	3	4
GPUTime	0.000244	0.000193	0.209993	2.834839



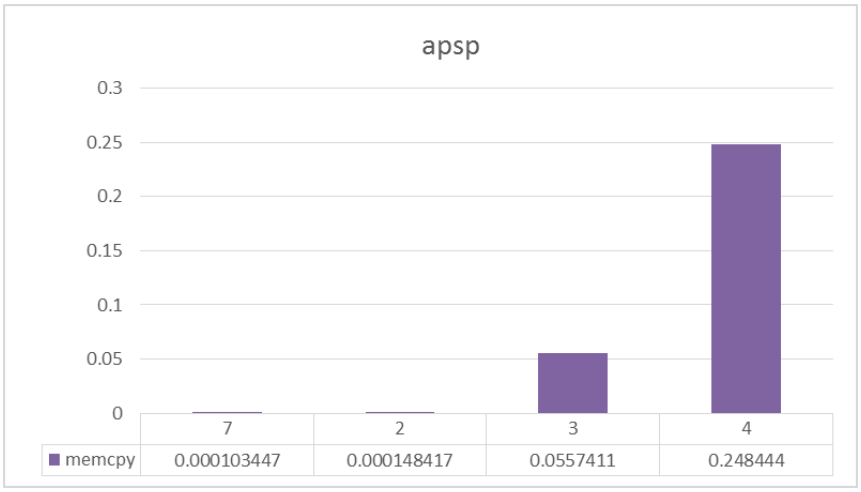


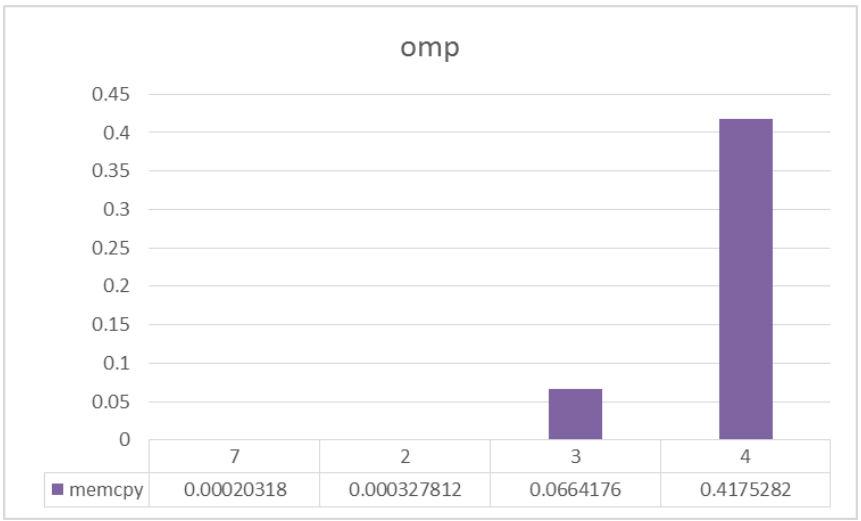
- Communication

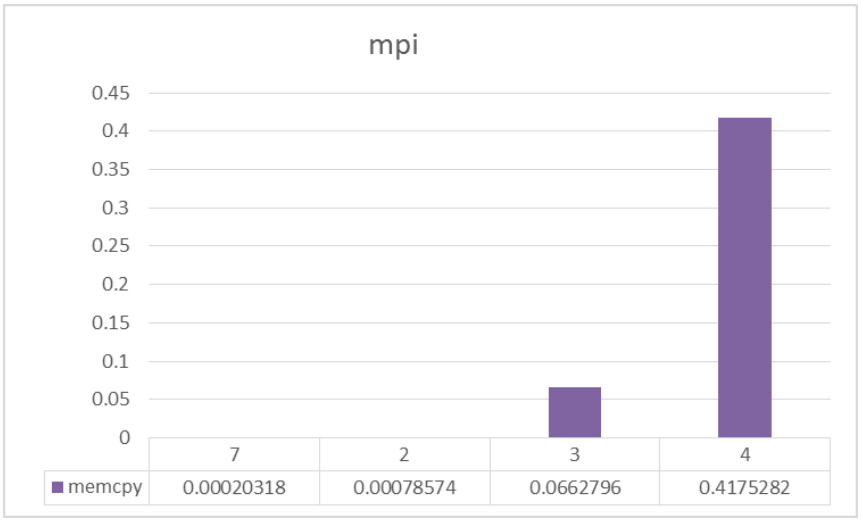



mpi
16
14
12
10
8
6
4
2
0
7
2
3
4
commTime
0.000105
0.003276
1.206412
13.723273

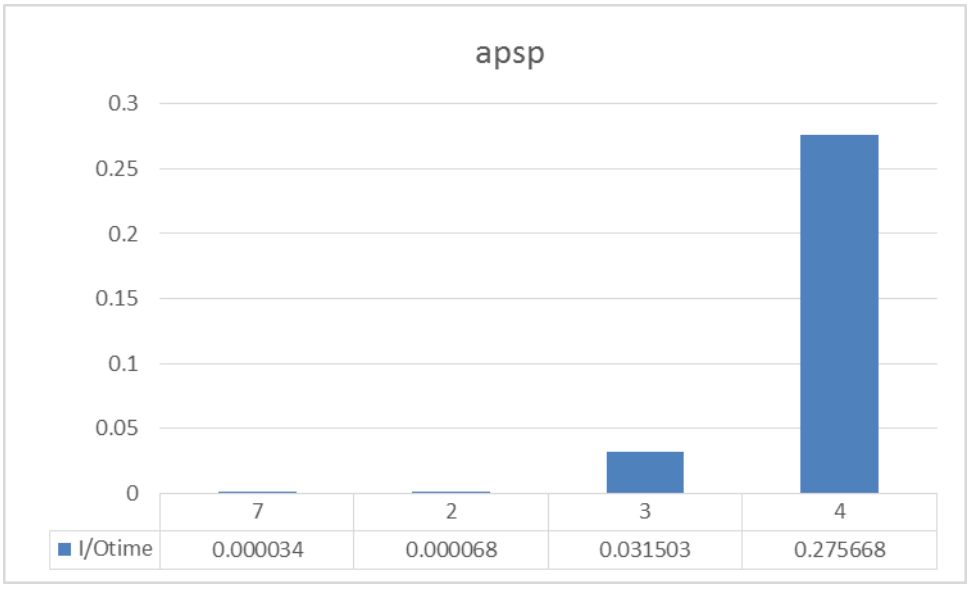
- memory copy

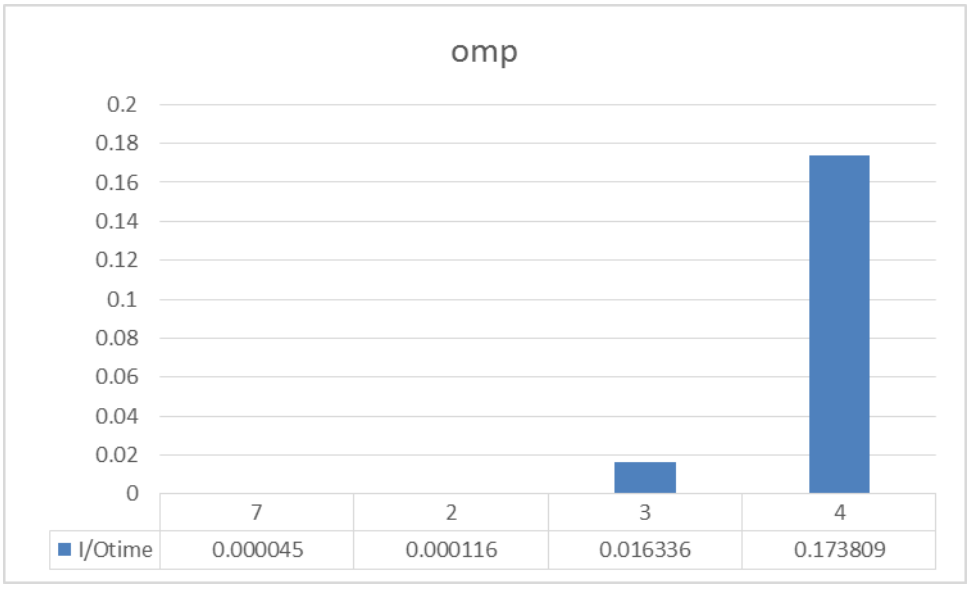

apsp
0.3
0.25
0.2
0.15
0.1
0.05
0
7
2
3
4
memcpy
0.000103447
0.000148417
0.0557411
0.248444

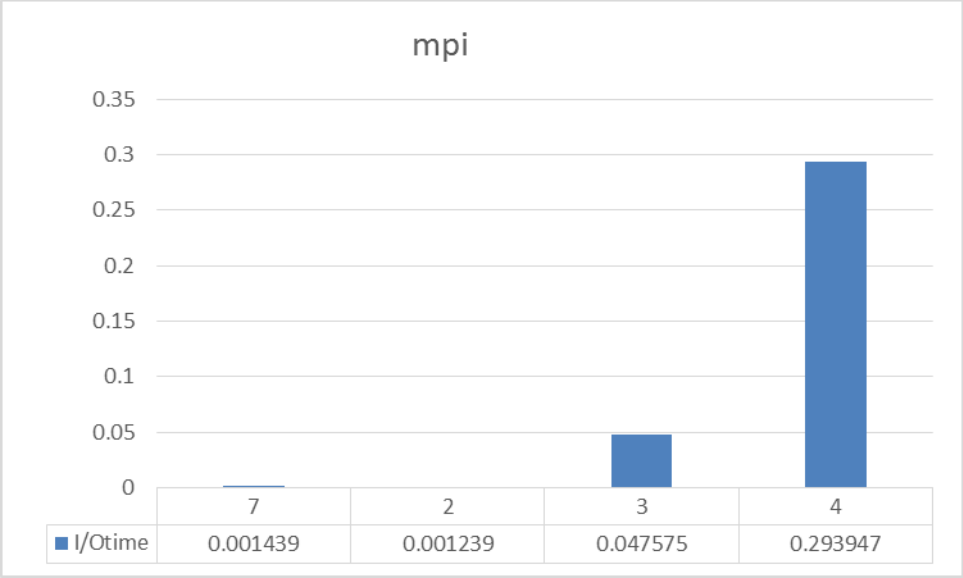

omp
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
7
2
3
4
memcpy
0.00020318
0.000327812
0.0664176
0.4175282



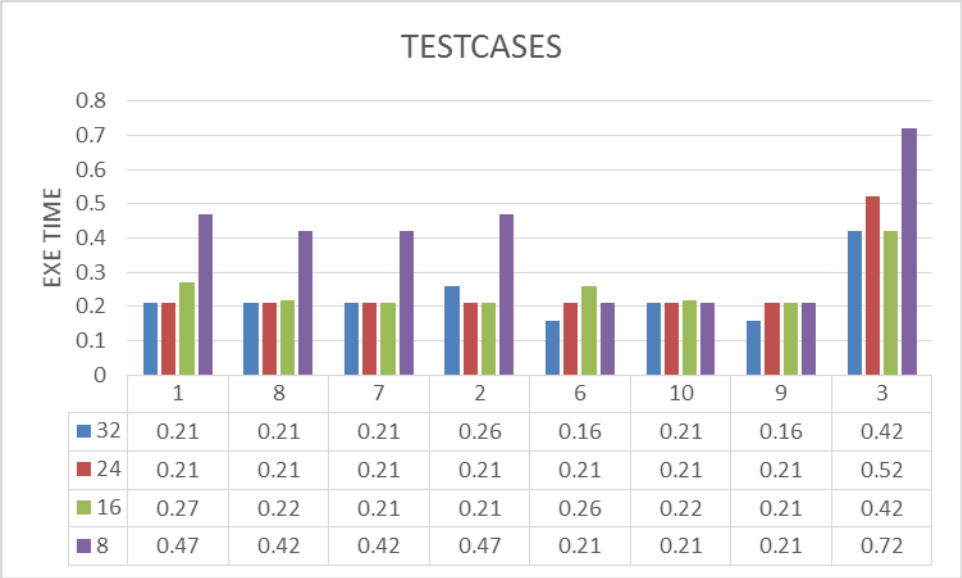
- I/O of your program w.r.t. input size.

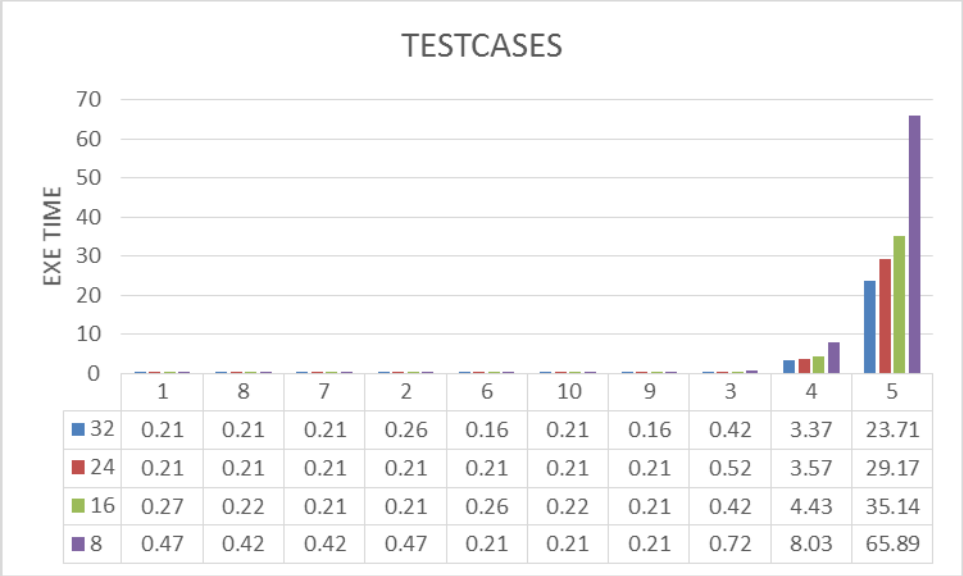






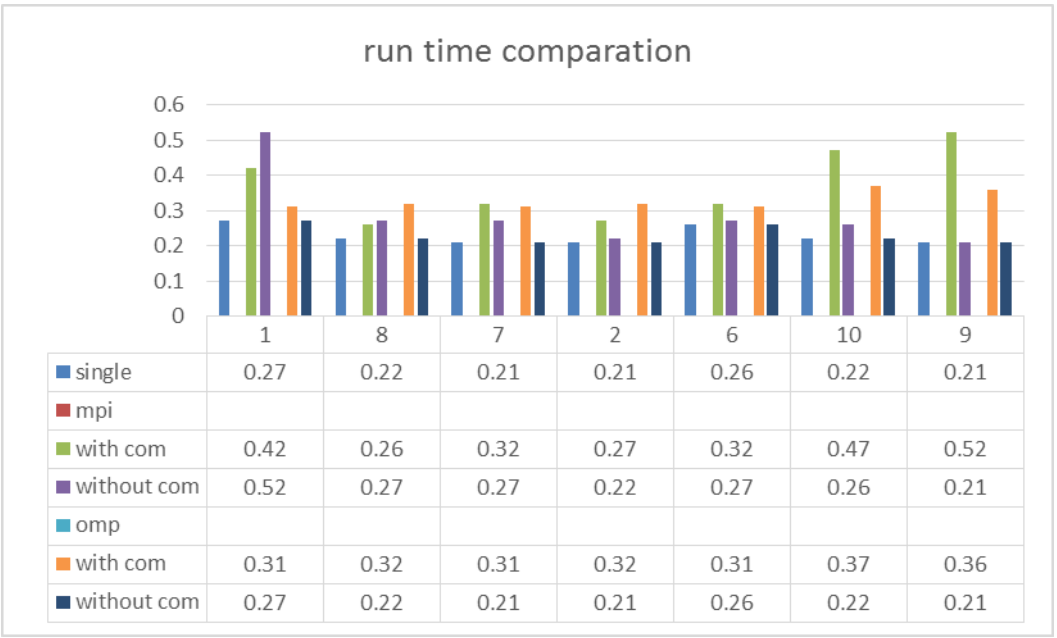
- Blocking Factor
  - 越大越快，最大到 32



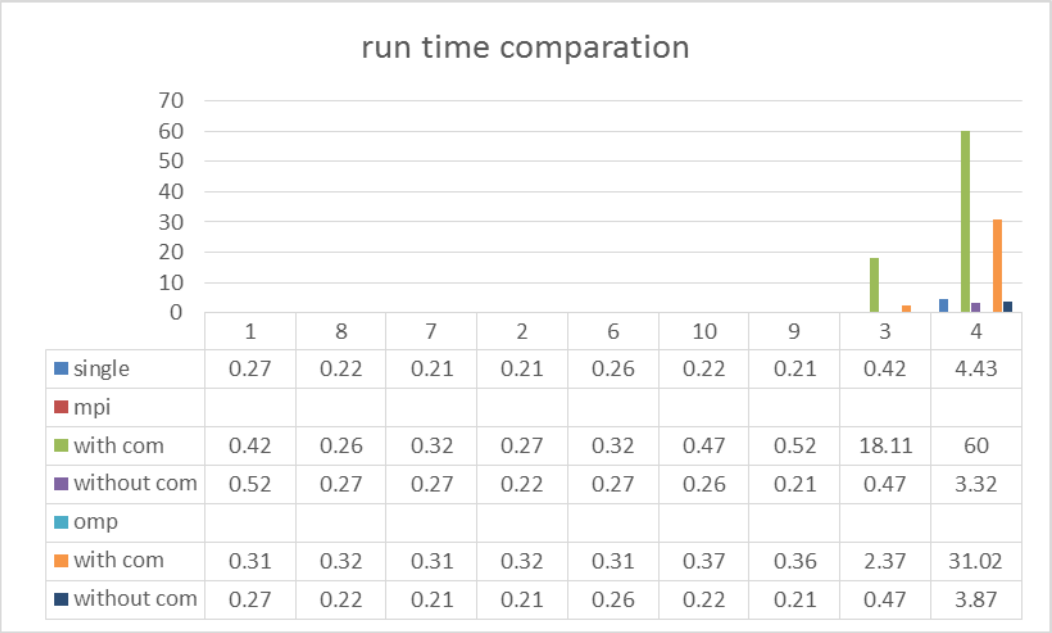


	1	8	7	2	6	10	9	3	4	5
32	0.21	0.21	0.21	0.26	0.16	0.21	0.16	0.42	3.37	23.71
24	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.52	3.57	29.17
16	0.27	0.22	0.21	0.21	0.26	0.22	0.21	0.42	4.43	35.14
8	0.47	0.42	0.42	0.47	0.21	0.21	0.21	0.72	8.03	65.89

- Reduce communication： 不溝通比溝通快



	1	8	7	2	6	10	9
single	0.27	0.22	0.21	0.21	0.26	0.22	0.21
mpi							
with com	0.42	0.26	0.32	0.27	0.32	0.47	0.52
without com	0.52	0.27	0.27	0.22	0.27	0.26	0.21
omp							
with com	0.31	0.32	0.31	0.32	0.31	0.37	0.36
without com	0.27	0.22	0.21	0.21	0.26	0.22	0.21



	1	8	7	2	6	10	9	3	4
single	0.27	0.22	0.21	0.21	0.26	0.22	0.21	0.42	4.43
mpi									
with com	0.42	0.26	0.32	0.27	0.32	0.47	0.52	18.11	60
without com	0.52	0.27	0.27	0.22	0.27	0.26	0.21	0.47	3.32
omp									
with com	0.31	0.32	0.31	0.32	0.31	0.37	0.36	2.37	31.02
without com	0.27	0.22	0.21	0.21	0.26	0.22	0.21	0.47	3.87