CMSE 822 Homework 6

Mengzhi Chen and Tong Li

Problem 1

a)

Table 1

Scenario	#nodes	#edges	Bisection Bandwidth	Cost (thousand)	Grade	Rank
Bus	170	169	1	1019	853	3
Ring	170	170	2	1020	856	2
Fully Connected	40	780	400	980	1400	1
2D torus	144	288	24	1008	792	5
3D torus	125	375	50	1000	775	6
Hypercube	64	192	32	512	416	7
Fat tree	127	384	64	1019	827	4

The overall situations are listed in table 1. Having limited budget up to 102 million, for different scenarios, we can determine their number of nodes and edges, as well as their bisection bandwidths. According to customers' evaluation standard, their grades are also listed. Finally, they are ranked based on grades. We can easily see that "Fully connected" is the most suitable one.

b)

Scenario	#nodes	#edges	Bisection Bandwidth	Cost (thousand)	Grade	Rank
Bus	170	169	1	1019	854	4
Ring	170	170	2	1020	858	3
Fully Connected	40	780	400	980	1800	1
2D torus	144	288	24	1008	816	6
3D torus	125	375	50	1000	825	5
Hypercube	64	192	32	512	448	7
Fat tree	127	384	64	1019	891	2

The customers change their demands this time. So, we recalculate connection scenarios' grades and rank them. Again, "Fully connected" wins.

Problem 2

- Modules GNU/4.4.5 and CUDA/6.0 are used.
- All data are average of four repetitions.

During our performance tests, we vary the grid size (n) and block size but fixing the number of iterations as 4000(implemented in code). The results are shown in figure 1 to figure 3.

There are mainly three noticing points. Firstly, we see great improvement of speedup from v1 to v2. It is because, in v2, we largely reduce kernel initiation overheads by fusing everything into single kernel. However, the differences between v2, v3 and v4 are indistinctive. The optimizations of v3 and v4 are on registers and shared memories. The trend may indicate that the kernel is compute bound.

And then, we see every version benefits from increasing n from 200 to 1000 and 5000. Generally, the curves shift upward and keep their own trends from figure 1 to 3. It may due to the fact that with bigger n, computation takes up larger proportion and kernel initiation overheads become a small part.

Finally, we focus on performance with varying block size, we see 16*16 block is the best choice. Checking in CUDA calculator, we see 16*16 block has 100% device occupancy and it can exhaust several warps (multiple of 32). For other block sizes except 32*32 case, they do not meet all two features that 16*16 block owns. In situation of 32*32 block, although two conditions are fulfilled, it does not perform as well as 16*16 case. So, there must exist some unknown reasons beyond our consideration.

All in all, with the help of GPU and comprehensive consideration, we can achieve great accelerations of some tasks with supreme efficiencies.

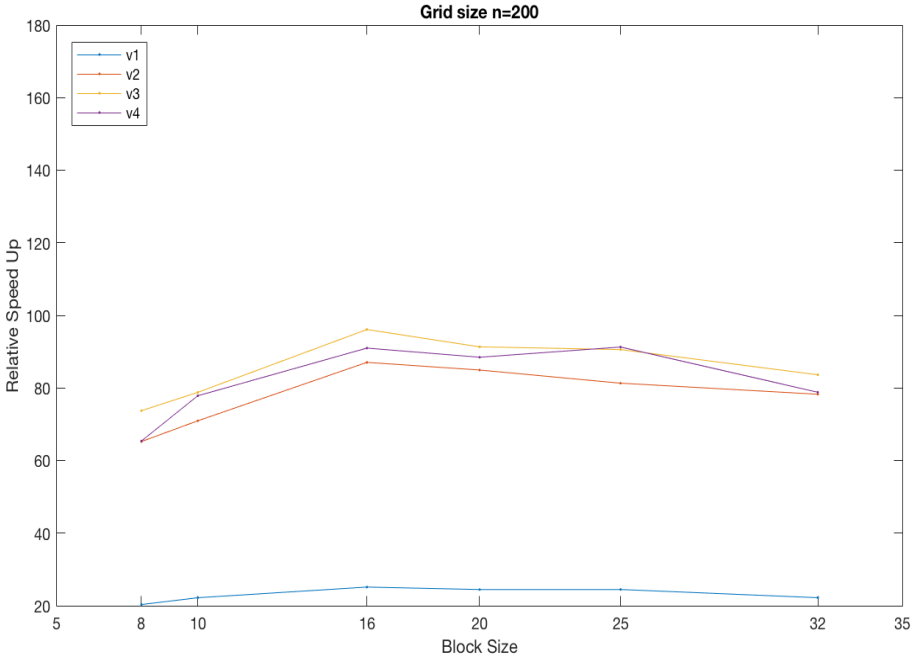

Fig 1

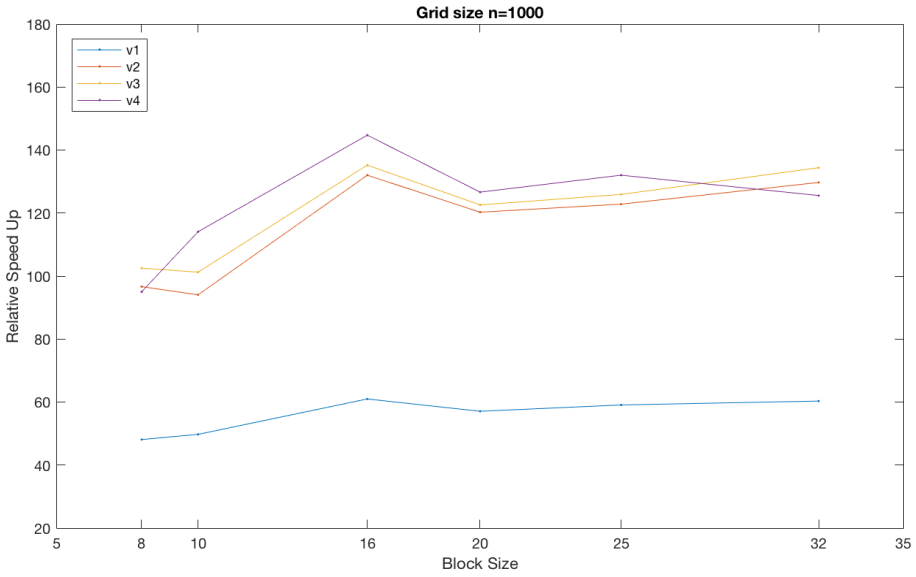
Grid size n=1000
v1
v2
v3
v4
Relative Speed Up
Block Size
180
160
140
120
100
80
60
40
20
5
8
10
16
20
25
32
35

Fig 2

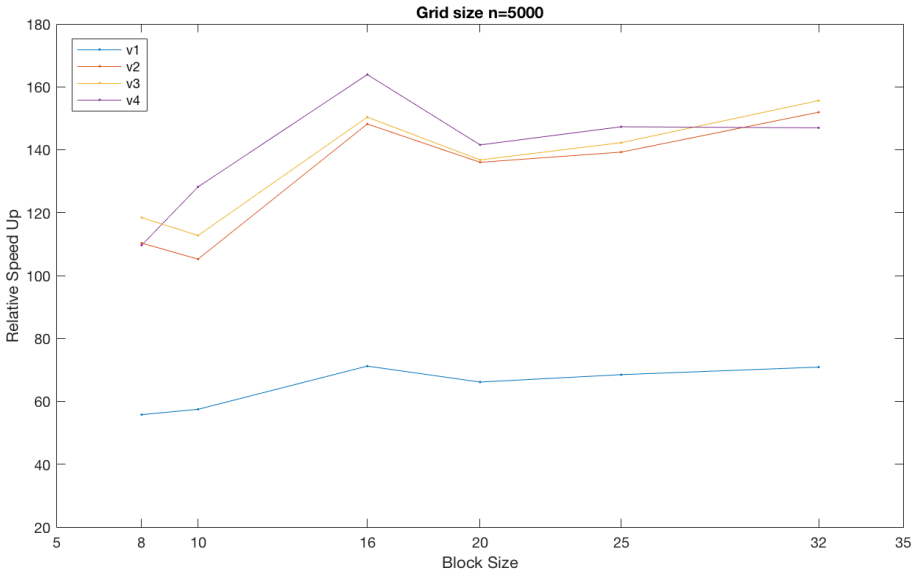
Grid size n=5000
v1
v2
v3
v4
Relative Speed Up
Block Size
180
160
140
120
100
80
60
40
20
5
8
10
16
20
25
32
35

Fig 3