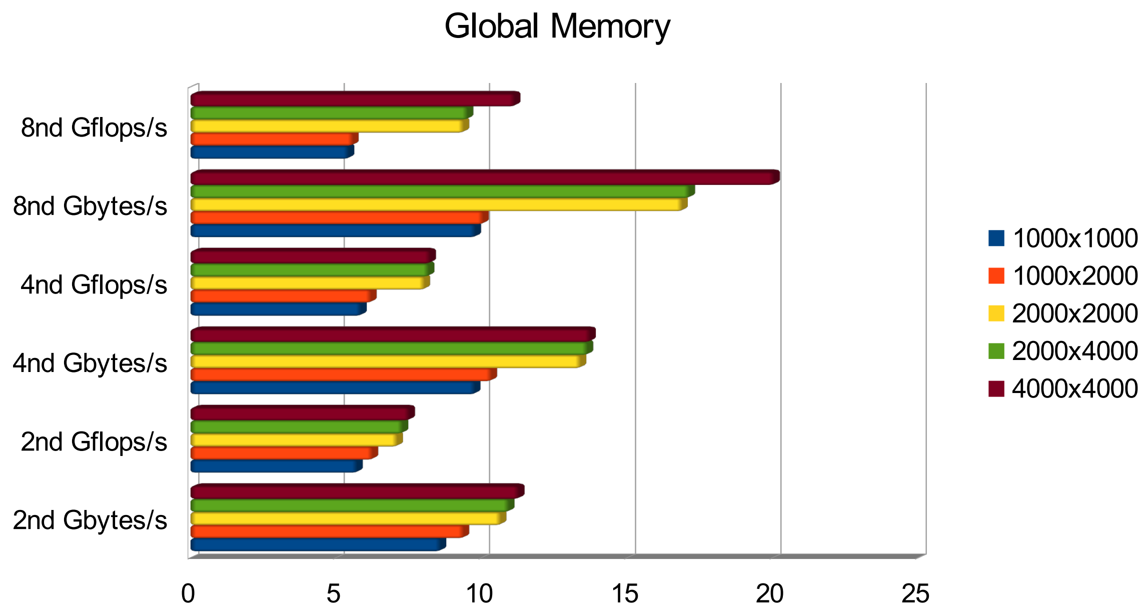
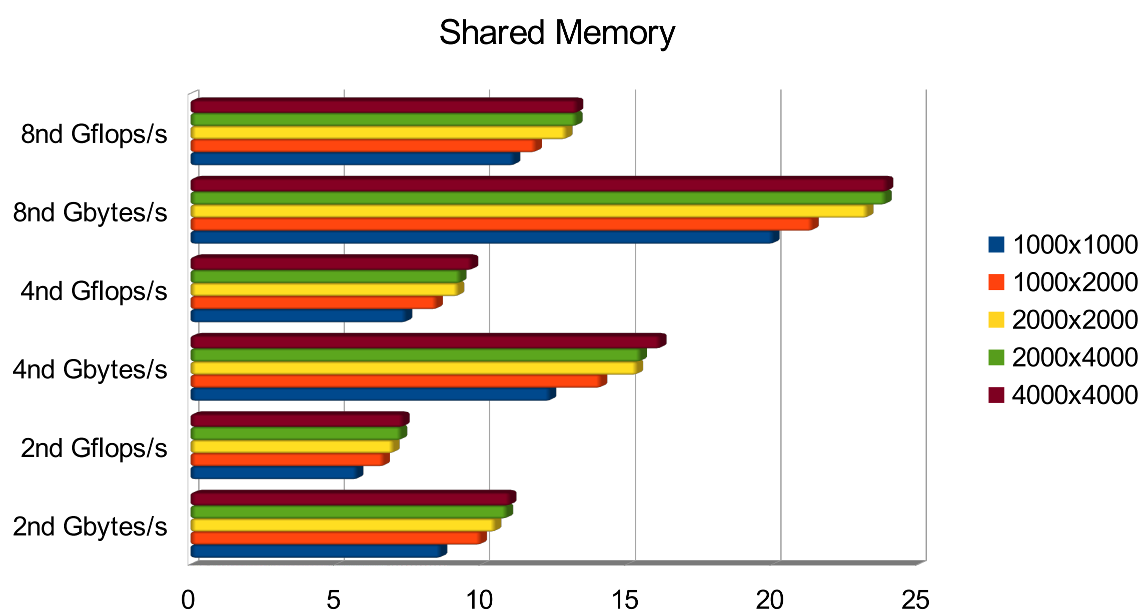
Programming Assignment 2
CME 213

Dong-Bang Tsai

1. Plot bandwidth vs. size of the grid, and flops vs. size of the grid for each of order. Do this for shared memory on one plot and global memory on a separate plot.



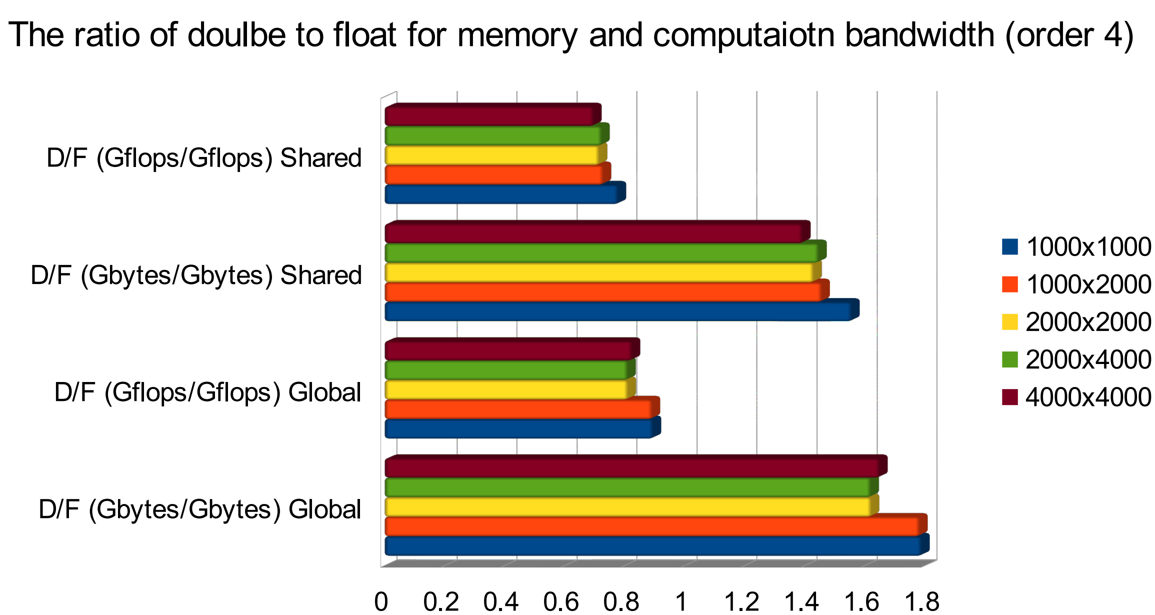


2. How does the shared memory performance compared with the cache only performance? Explain why one outperforms the other. Do larger orders favor the cache or shared memory implementation? Try to explain all your findings.

   When the grid size is small, the shared memory version dramatically outperforms cache only version. As we can see in the figure, the performance differences in share memory version are not obvious between different grid size, but in cache only version, the performance will be increased when the grid size is increased. However, when the grid size is increased, the performance difference is not huge between two version. The reason may be that when the grid size is larger, the parallel computing in cache only version in different blocks can hide the memory latency; therefore, the speedup is not significant. However, in the smaller grid size, the share memory version can provide quick access to the data when the computation requires the data which is reused, so the speedup is significant.

   In the result I obtained, the larger order method favors the shared memory implementation; especially when the grid size is small, the performance difference between two implementations is large. The reason is when higher order method is used, in single computation, more neighborhood data is required, which means that the same data is used more than once from different threads in the same block when we use share memory. Since reads are faster when we read from shared memory, higher order implementation can take advantage of that. However, when the order is lower, the data reused probability is lower between different threads in a block, and we still need time to read them from global memory but we only used it a few time. Therefore, sometimes, it'll slow down the computation speed.

3. Create a plot of the ratio of double to float for bandwidth and flops against the size of the input. Do this only for order 4.



4. How do the flops and bandwidth compare between float and double?

   The computational speed (flops) is decreased since it takes longer for one instruction to finish arithmetic operation. However, the memory bandwidth is actually increased since accessing pattern of double is coalesced, and the GPU loads more data in one clock cycle.

Raw Data:

Data of Float Precision

Global Memory	1000x1000	1000x2000	2000x2000	2000x4000	4000x4000
2nd Gbytes/s	8.60215	9.39335	10.6667	10.9464	11.2908
2nd Gflops/s	5.73477	6.26223	7.11111	7.29761	7.52720
4nd Gbytes/s	9.77995	10.3493	13.4116	13.6519	13.7428
4nd Gflops/s	5.86797	6.20957	8.04694	8.19113	8.24565
8nd Gbytes/s	9.79592	10.0629	16.9014	17.1531	20.0662
8nd Gflops/s	5.44218	5.59050	9.38967	9.52948	11.1479
Global Memory	1000x1000	1000x2000	2000x2000	2000x4000	4000x4000
2nd Gbytes/s	8.63309	10	10.4918	10.8967	11.0123
2nd Gflops/s	5.75540	6.66667	6.99454	7.26447	7.34155
4nd Gbytes/s	12.3839	14.1343	15.3698	15.4964	16.1698
4nd Gflops/s	7.43034	8.48057	9.22190	9.29782	9.70187
8nd Gbytes/s	20.0557	21.3967	23.2821	23.9103	23.9651
8nd Gflops/s	11.1421	11.8871	12.9345	13.2835	13.3139

Data of Double Precision

	1000x1000	1000x2000	2000x2000	2000x4000	4000x4000
G 4nd Gbytes	17.5055	18.4544	21.7539	22.1224	22.6749
G 4nd Gflops	5.25164	5.53633	6.52617	6.63671	6.80248
S 4nd Gbytes	19.2771	20.5920	21.9780	22.4168	22.5273
S 4nd Gflops	5.78313	6.17761	6.59341	6.72504	6.75818