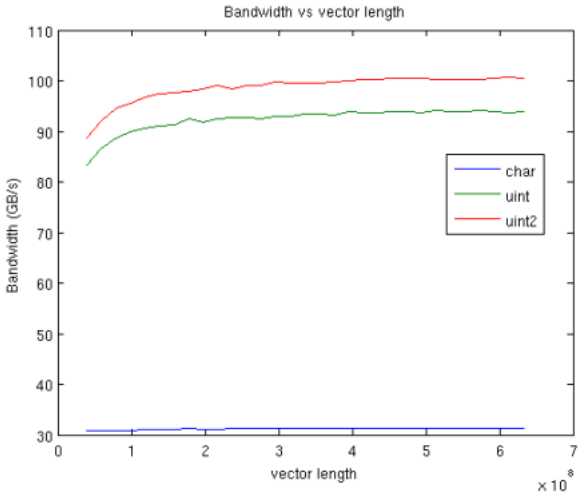
Programming Assignment 1
CME 213

Dong-Bang Tsai

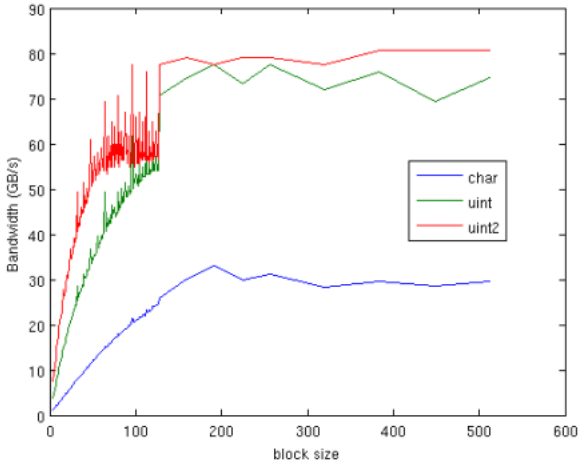
1) Caesar Cipher

a) Generate plots with char, uint, and uint2 on the same plot. Create a plot for

i) Bandwidth vs. vector length



ii) Bandwidth vs. block size



b) Explain the shape of your plots. You only need to plot enough data points to show the general trend.

i) For “Bandwidth vs. vector length”, the bandwidth is constrained by the memory speed instead of computational speed in GPU. Therefore, the performance differences between char, uint, uin2 are mainly from the memory access speed.

For char, each thread only accesses to one byte; therefore, in a warp, 32 threads only need 32 bytes data. However, Memory transactions are always 128 bytes, so it slows down the bandwidth even we only need 32bytes data in the warp, we get extra unused 96bytes data. The bandwidth is around 30GB/s.

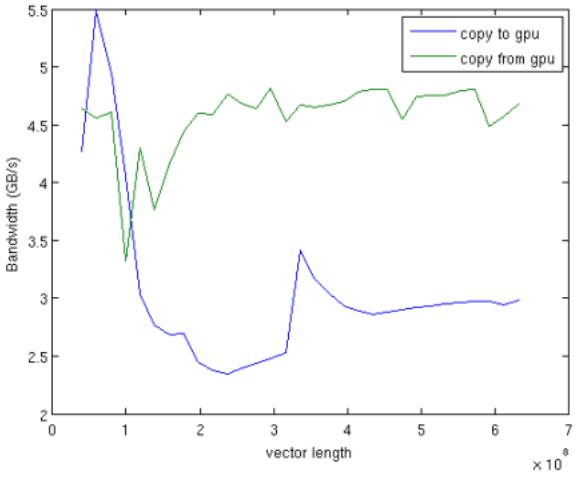
For uint, it will perform 4bytes memory transactions in each thread; therefore, it will perform 128bytes in a warp. In this case, the warp does one transaction and use all of the data which mean there is no unused data in that transaction which provides better performance. The bandwidth is around 90GB/s.

For uint2, it will perform 8bytes memory transactions in each thread; therefore, half warp issues a transaction. All the data will be used for those two transactions; therefore, the performance is good. It has slightly better performance compared with uint, the reason maybe two transactions avoid the problem that the GPU may wait for the data. The bandwidth is around 100GB/s.

As we increase the vector length, the bandwidth does't change too much, and it actually makes sense since at each time, we can only deal with the same amount of data.

For ii) We found that the bandwidth will increase as we increase the block size (# of threads in a block), and the increasing will stop when the # of threads reach around 128. When we increase the # of threads in a block, we can group them into several wraps in each block, and in each wrap, if the # of threads are near optimal value, 32, they can share the memory data which can speed up the bandwidth. Also, when we have more wraps in a block, we can also increase the concurrency performance.

c) What is the bandwidth that is achieved on the copies from the host to the device, and from the device to the host? How do they compare?



As we can see, when the vector length size is very large, the bandwidth of copying to GPU is around 4.8GB/s, and the bandwidth of copying from GPU to host is around 3.0GB/s. The difference may be the asymmetric in PCI-e bandwidth, and the motherboard chipset.

2) PageRank

a) Calculate the bandwidth when the average number of edges is 8.

The bandwidth is around 4.89GB/s when the average number of edges is 8.

i) How does this compare to the bandwidth in the Caesar Cipher? Explain any differences you observe.

It's extremely show compared with the Caesar Cipher. The reason is not in computational speed but in the memory transaction speed. When we perform

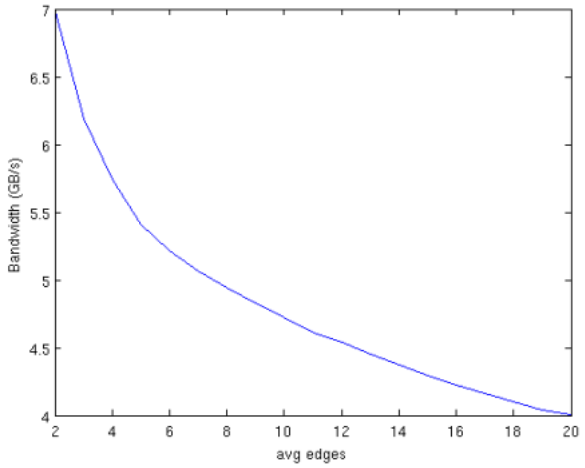
graph_nodes_in[graph_edges[j]]*inv_edges_per_node[graph_edges[j]];

the memory access pattern is not continuous, and in each memory transaction, the threads in a warp will obtain 128bytes. Therefore, lots of data are unused, and lots of time spends on copying the data.

ii)

The bandwidth will not change the bandwidth by changing the block size. I tried from # of threads in a block from 2 to 128, and all give similar result. As we already pointed out in i), the memory coalescing is not good in this memory access pattern; therefore, changing # of threads in a block or warp will do nothing for the bandwidth. The GPU still needs several requests for a thread to finish the jobs, and for the threads in a warp, there is almost no overlapping in turn of required data.

b) Plot bandwidth vs avg_edges ranging from 2 to 20 edges.



i)

When the average number of edges are lower, we found that the bandwidth is higher. The reason is that when the edges are lower, when the GPU compute

graph_nodes_in[graph_edges[j]]*inv_edges_per_node[graph_edges[j]];

the data will trend to be more closer, which means the memory coalescing is better when the thread try to access the data from global memory. Therefore, less memory transactions will be performed in the threads and wrap.

ii) Yes, the size of the vector will have a significant impact on the bandwidth. When the length of the vector is smaller, there is higher probability that GPU accesses to more useful data in one memory transaction.