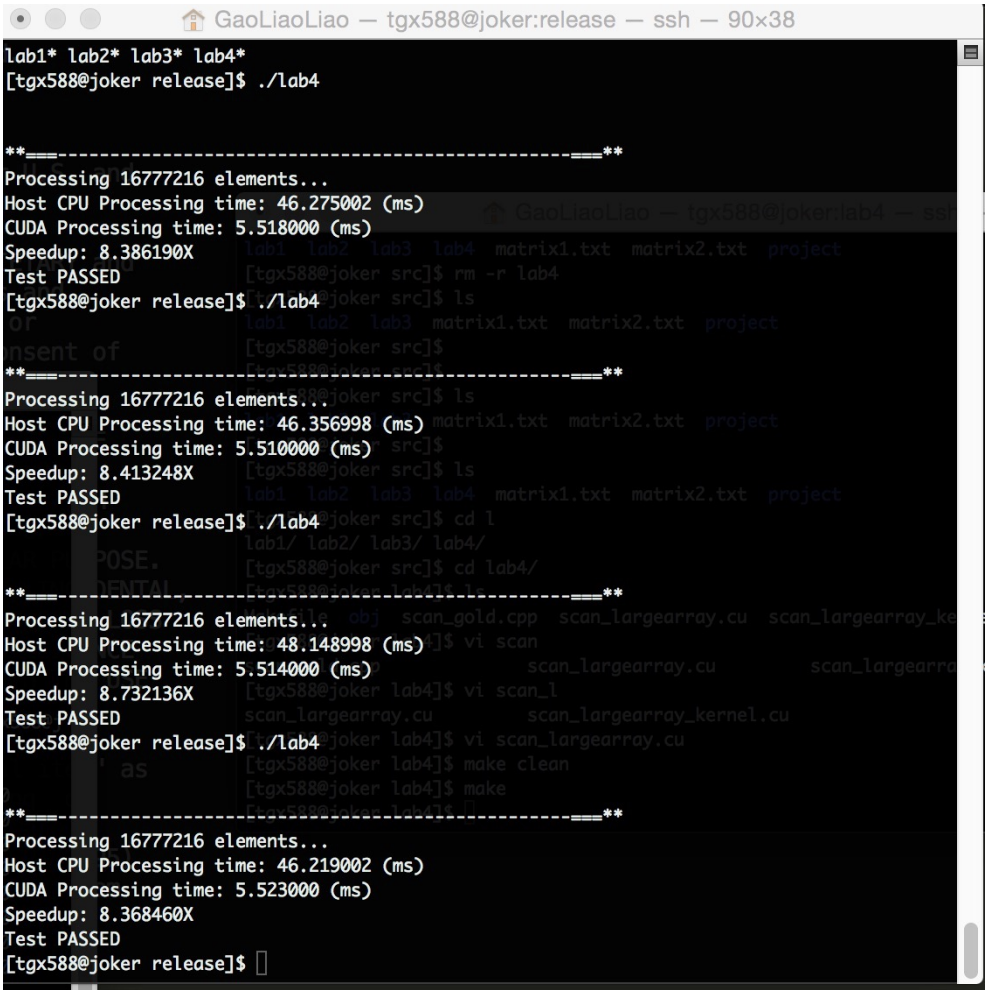
Lab 4 report (Tianqi Gao, Weihan Chu)

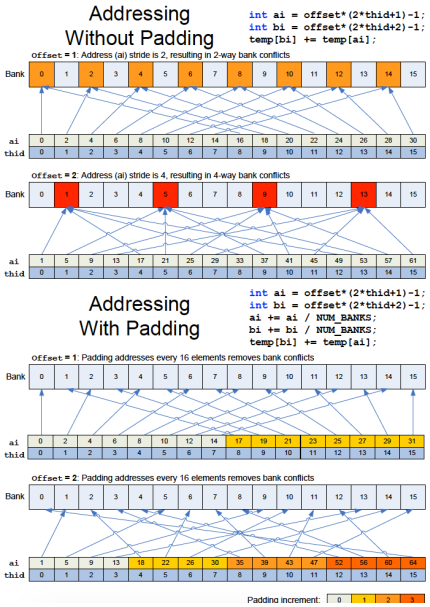


For the result, when we run our program without arguments and we processing 16777216 elements. The CPU processing time is 49ms, but the CuDA processing time is 5.543 ms. tH SPEEDUP IS 8.854772x. So the CUDA is more better.

For the handle of arrays not a power of two in size, we process the remain with a scan kernel modified to handle non power of 2 arrays. This kernel pads the shared memory array used out to the next higher power of two and initializes this extra memory to zero while loading in the data from device memory.

For the handle of the shared memory bank conflicts, we just need to pad the 2D array to a width that is not evenly divisible by number of shared memory banks. For this problem, we can achieve this by adding a variable amount of padding to each shared memory array index we compute. We just add to the index the value of the index divided by the number of shared memory banks

Below is a picture I found on the website which has the same idea of handle bank conflict.



In this lab, we also use the shared memory to speed up. What's more, in the beginning, we just implement the Hillis and Steele's algorithm which is not very good. Then, we use the Blelloch's algorithm,which can speed up the computation.

For the FLOPS rate in CPU, because the complexity of this algorithm is O(n), so the FLOPS=16777216/(49/1000)=342392163. For the FLOPS rate in CUDA, because the complexity of this algorithm is also 2n, so the FLOPS=2*16777216/(5.543/1000)=6053478621.685. for the CPU, the theoretical performance limits is 20G, for the CUDA it is 1t. For the bottlenecks, The CPU is write/read. For the CUDA, it is the number of threads.