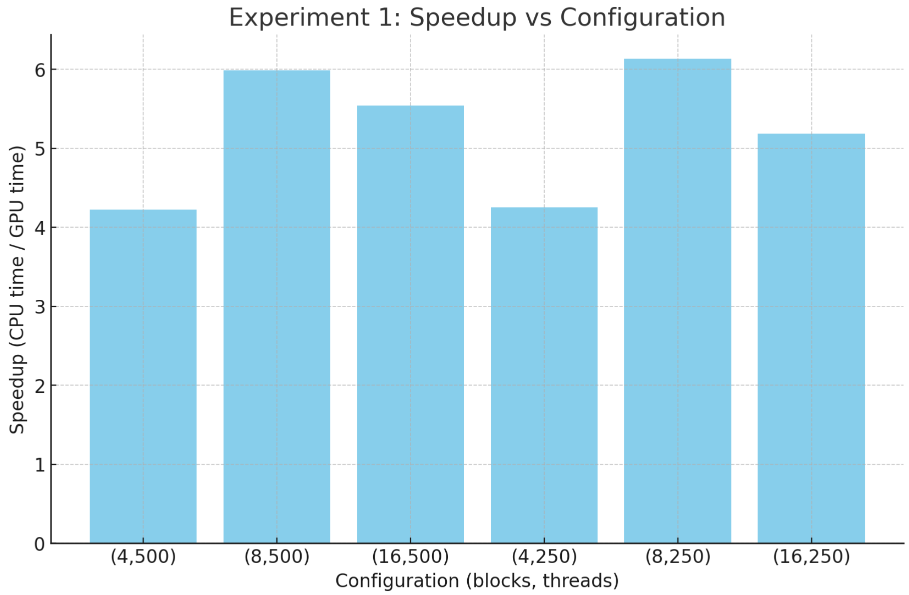
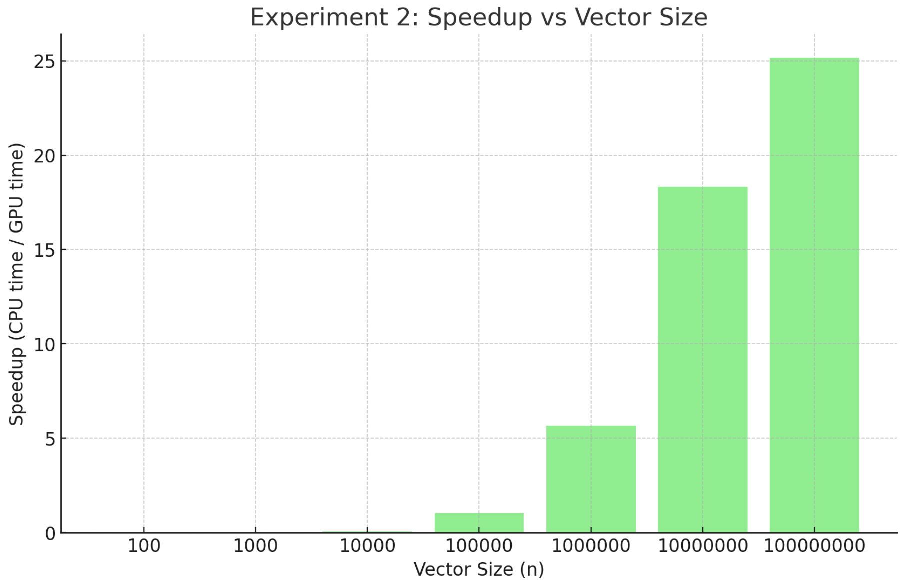
Experiment 1



In general there are improvements with the increase of the block size to 8 blocks. Which make sense, because the more blocks allocated means the lesser tasks for each thread. However, once to 16 there is a slight decrease, my assumption is that there are too many threads for n=1,000,000. To amortize the scheduling overhead for each threads and warps, and it increases the fragmentations and also becomes harder to synchronize each warps.

Experiment 2



The speed up graph is quite straight forward as the larger the vector size to more speed up. But you can see the speed up is not completely linear. My assumption is that when the vector size is small, we are bounded by the clock speed on GPU which is slower than CPU, therefore it is faster on CPU. As the vector size increases we also saturated the resources on GPU, so the speedup growth is slowing down. Although it is obvious faster on GPU.

```
---  Expriment 1 ---
Each vector will have 1000000 elements
Using 500 blocks per grid and 4 threads per block
Total time taken by the sequential part = 0.003285
Total time taken by the GPU part = 0.000778
Each vector will have 1000000 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.003318
Total time taken by the GPU part = 0.000554
Each vector will have 1000000 elements
Using 500 blocks per grid and 16 threads per block
Total time taken by the sequential part = 0.003282
Total time taken by the GPU part = 0.000592
Each vector will have 1000000 elements
Using 250 blocks per grid and 4 threads per block
Total time taken by the sequential part = 0.003348
Total time taken by the GPU part = 0.000787
Each vector will have 1000000 elements
Using 250 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.003324
Total time taken by the GPU part = 0.000542
Each vector will have 1000000 elements
Using 250 blocks per grid and 16 threads per block
Total time taken by the sequential part = 0.003287
Total time taken by the GPU part = 0.000634
--- Expriment 2 ---
Each vector will have 100 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.000002
Total time taken by the GPU part = 0.000352
Each vector will have 1000 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.000004
Total time taken by the GPU part = 0.000418
Each vector will have 10000 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.000034
Total time taken by the GPU part = 0.000549
Each vector will have 100000 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.000329
Total time taken by the GPU part = 0.000315
Each vector will have 1000000 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.003294
Total time taken by the GPU part = 0.000581
Each vector will have 10000000 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.033502
Total time taken by the GPU part = 0.001827
Each vector will have 100000000 elements
Using 500 blocks per grid and 8 threads per block
Total time taken by the sequential part = 0.335144
Total time taken by the GPU part = 0.013322
```