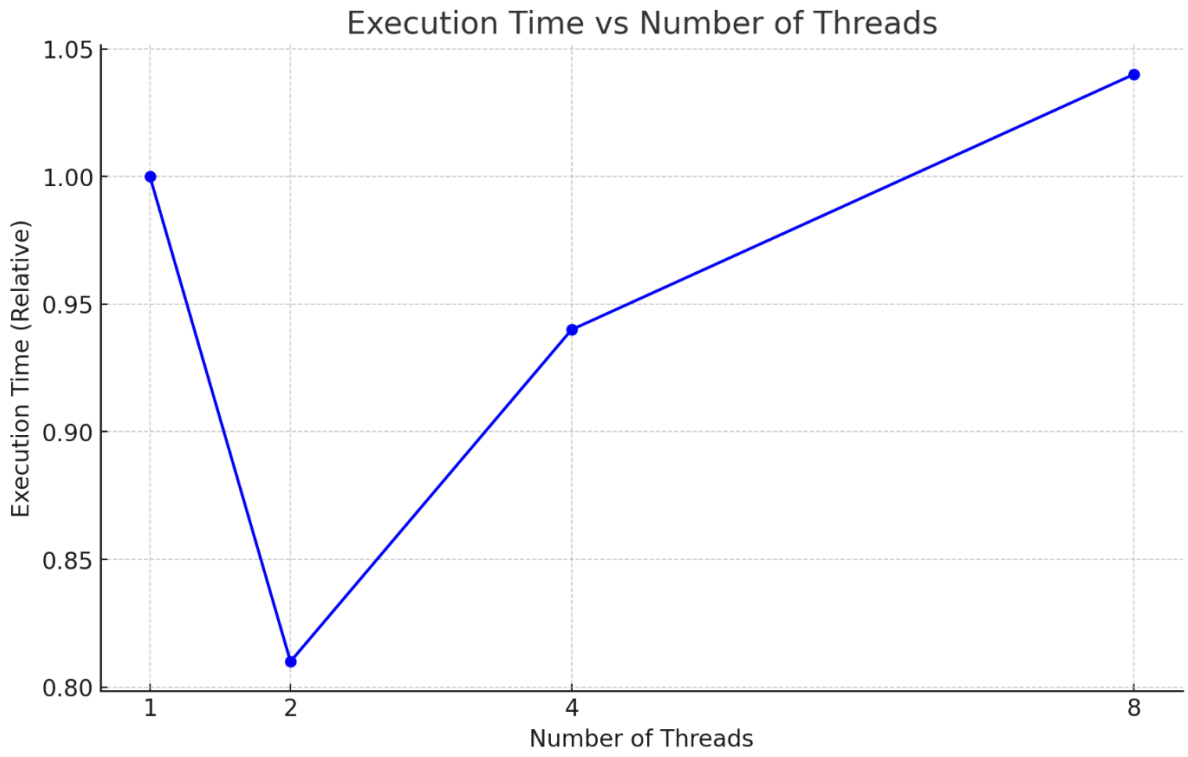
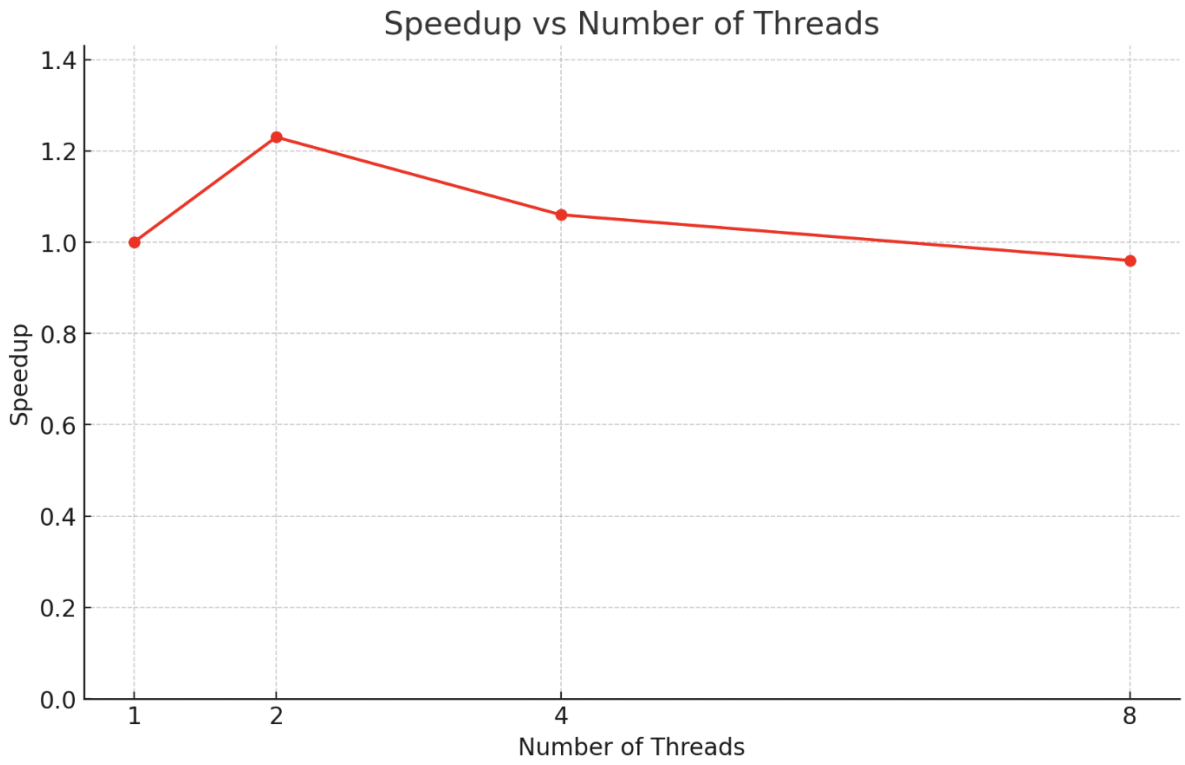
(1)

NumOfThreads	GNU time(Sec)	Speedup
1	1	1
2	0.81	1.16
4	0.94,	1.06
8	1.04	0.99

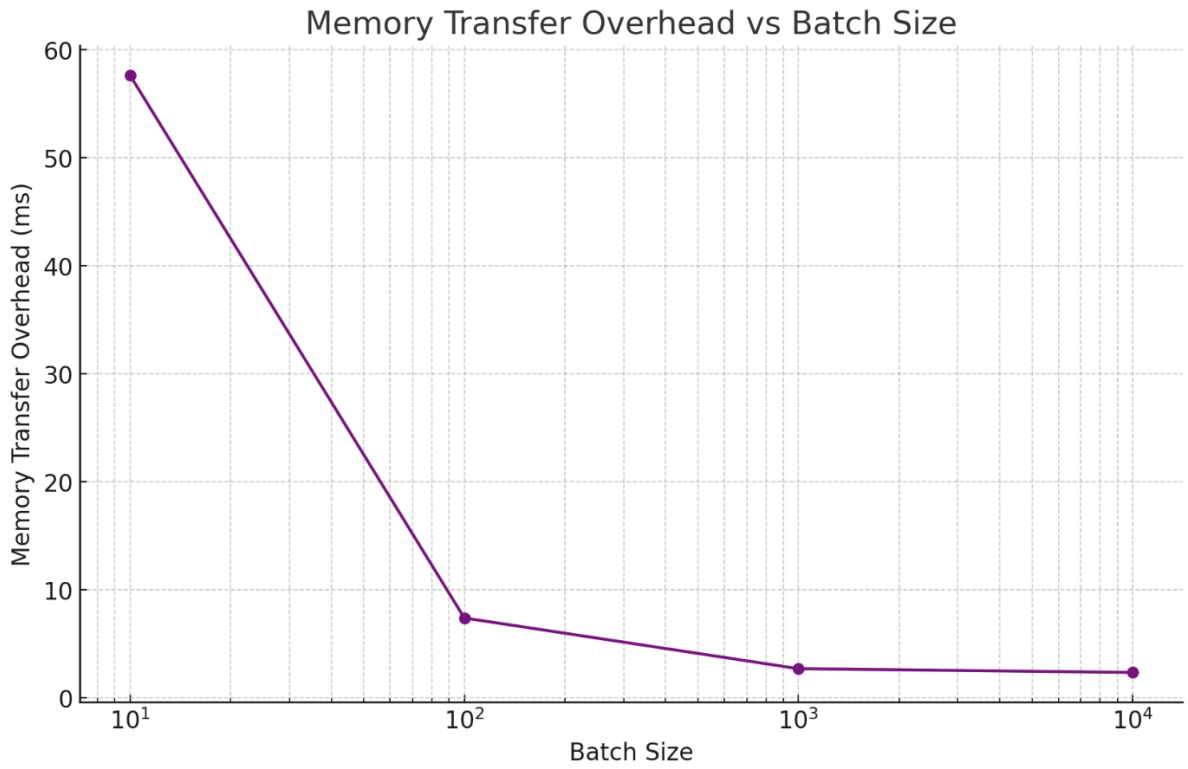




Explanation:
The speedup plot relative to the number of threads for the provided parallel pipeline code using Intel Threading Building Blocks (TBB) shows an initial increase in speedup when going from 1 to 2 threads, indicating that parallel execution effectively reduces execution time here. However, as the number of threads increases to 4 and 8, the speedup plateaus and then slightly declines, suggesting that overheads associated with managing additional threads, synchronization, or limited parallelism in the workload begin to outweigh the benefits of adding more threads. Few reasons could be Serial and Parallel Filters and Amdahl's Law.

(2)

Batch Size	total time for memory transfers(ms)
10	57.633
100	7.401
1000	2.7034
10000	2.358



Explanation:

Overhead of Small Transfers: Smaller data transfers between the host (CPU) and device (GPU) are less efficient because the overhead of initiating a transfer is relatively high compared to the actual time spent moving data. For very small batches, this overhead dominates the total time spent in data transfer, leading to higher per-element overhead.

GPU Memory Access Patterns: GPUs are designed to handle large blocks of data more efficiently than small ones. Larger batches allow for more coalesced memory accesses, where multiple data elements are loaded in a single memory transaction. This reduces the number of memory accesses required and maximizes memory bandwidth utilization, making transfers more efficient.

(3)

Batchsize of 10000 gave the highest performance. Fixing it and varying the block size and grid size to see performance numbers.

gridsize	blocksize	Read mapper Kernel Execution time
16384	**256**	**131.23ms**

16384	512	238.55ms
8912	256	132.78ms
8912	512	241.77ms