Parallel Computing

Lab 2 Report

Name	Sec	BN
Sara Gamal	1	20
Eman Ibrahim	1	14

Test Cases

100 Elements			
	Kernal 1	**Kernal 2**	**Kernal 3**
10*10	3.3590us	5.1840us	6.8160us
100*1	3.2320us	3.5840us	15.200us
1*100	3.2320us	15.232us	3.5520us
10000 Elements			
	Kernal 1	**Kernal 2**	**Kernal 3**
100*100	3.5840us	53.407us	41.888us
10000*1	7.8080us	3.7440us	1.2028ms
1*10000	7.3600us	1.1937ms	3.6800us
1000000 Elements			
	Kernal 1	**Kernal 2**	**Kernal 3**
1000*1000	51.520us	1.7132ms	601.21us
1000000*1	546.49us	48.991us	132.01ms
1*1000000	526.94us	131.36ms	47.648us

Comments

1. For a matrix with 100 elements, there is no significant performance difference between the three kernels, as the small size limits the impact of memory access patterns and parallelism.

2. For larger matrix sizes, Kernel 3 outperforms Kernel 2 when the number of rows equals the number of columns. In Kernel 2 (one thread per row), threads within a warp access different rows, resulting in inefficient memory access. In contrast, Kernel 3 (one thread per column) benefits from consecutive memory accesses within the warp, leading to better memory coalescing and improved performance.

3. For larger matrix sizes, Kernel 2 outperforms Kernel 3 when the number of rows exceeds the number of columns. In Kernel 2 (one thread per row), each thread processes only a small number of columns, resulting in light computational workload per thread. However, Kernel 3 (one thread per column) struggles because the number of launched threads is limited by the smaller number of columns. With fewer threads available, parallelism is significantly reduced, leading to higher latency and lower overall efficiency.

4. For larger matrix sizes, Kernel 3 outperforms Kernel 2 when the number of columns exceeds the number of rows. In Kernel 3 (one thread per column), each thread processes only a small number of rows, resulting in light computational workload per thread.. However, Kernel 2 (one thread per row) struggles because the number of launched threads is limited by the smaller number of rows. With fewer threads available, parallelism is significantly reduced, leading to higher latency and lower overall efficiency.

5. For larger matrix sizes, Kernel 1 is the most efficient when the number of rows equals the number of columns. Since Kernel 1 assigns one thread per element, it ensures optimal workload distribution across all threads in a square matrix. This leads to efficient parallel execution with high memory coalescing, as adjacent threads access adjacent memory locations. Additionally, since the number of threads matches the number of elements, GPU resources are fully utilized, minimizing idle threads and maximizing performance.

.