CUDA gpuSwapCol Kernel Execution Time Prediction Report

This report applies the CUDA kernel performance prediction template to the kernel:
gpuSwapCol, launched with Grid = (5,5) and Block = (32,32).

We estimate the average per-launch execution time on five NVIDIA GPUs:
- GeForce GTX TITAN Black (6 GB)
- GeForce GTX TITAN X (12 GB)
- NVIDIA TITAN V (12 GB)
- GeForce RTX 2080 Ti (11 GB)
- GeForce RTX 4070 (12 GB)

```
__global__ void gpuSwapCol(int rows, float * dArray, int coli, int * dColj, int * dPivot)
{
    int rowIndex = blockIdx.x * blockDim.x + threadIdx.x;

    if(rowIndex >= rows)
        return;

    int colj = coli + (*dColj);
    float fholder;

    fholder = dArray[rowIndex+coli*rows];
    dArray[rowIndex+coli*rows] = dArray[rowIndex+colj*rows];
    dArray[rowIndex+colj*rows] = fholder;

    if((blockIdx.x == 0) && (threadIdx.x == 0)) {
        int iholder = dPivot[coli];
        dPivot[coli] = dPivot[colj];
        dPivot[colj] = iholder;
    }
}
```

1. Kernel Work Analysis

Launch configuration in main():
- Grid = (5, 5) → 25 blocks
- Block = (32, 32) → 1024 threads per block
- rows = 5 × 32 × 5 × 32 = 25,600
- cols = 100, coli = 0, *dColj = 1

Total threads:
T_total = 25 blocks × 1024 threads/block = 25,600 threads

The kernel uses:
rowIndex = blockIdx.x * blockDim.x + threadIdx.x;
gridDim.x = 5, blockDim.x = 32 → rowIndex ∈ [0, 159].
Since rows = 25,600, the condition (rowIndex >= rows) is never true, so all 25,600 threads execute the body.

For each thread (ignoring the final pivot swap):
- Load colj = coli + (*dColj) → one int load from dColj
- Load dArray[rowIndex + coli*rows] (float)
- Load dArray[rowIndex + colj*rows] (float)

• Store to dArray[rowIndex + coli*rows] (float)
• Store to dArray[rowIndex + colj*rows] (float)

Approximate per-thread global memory:
• 3 loads (1 int, 2 floats) → 12 bytes
• 2 stores (2 floats) → 8 bytes
→ 20 bytes per thread (base swap work).

Total base bytes:
Bytes_base = 25,600 threads × 20 bytes/thread = 512,000 bytes

Pivot swap region:
Executed only when blockIdx.x == 0 and threadIdx.x == 0.
For each such block there are 32 threads (threadIdx.y = 0..31), and gridDim.y = 5, so:
Threads_pivot = 5 blocks × 32 threads/block = 160 threads

Each pivot thread does:
int iholder = dPivot[coli];
dPivot[coli] = dPivot[colj];
dPivot[colj] = iholder;

That is 2 int loads + 2 int stores = 16 bytes per pivot thread.

Pivot bytes:
Bytes_pivot = 160 threads × 16 bytes = 2,560 bytes

Total global memory traffic:
Bytes_total ≈ 512,000 + 2,560 = 514,560 bytes ≈ 5.15 × 10^5 bytes (~0.49 MiB)

Floating-point arithmetic is minimal (no heavy FLOPs). The kernel is effectively memory + launch dominated.

2. GPU Specifications Used

Approximate published FP32 peak performance and memory bandwidth:

GPU	Peak FP32 (FLOPs/s)	Bandwidth (bytes/s)
GTX TITAN Black	5.12e12	3.36e11
GTX TITAN X	6.14e12	3.365e11
TITAN V	1.49e13	6.528e11
RTX 2080 Ti	1.345e13	6.16e11
RTX 4070	2.9e13	5.04e11

3. Time Estimates

We compute the memory-bound time for each GPU as:
t_mem = Bytes_total / Bandwidth

Using Bytes_total ≈ 5.1456 × 10^5 bytes, we obtain:

GPU	t_mem (µs)	t_body (µs)	t_total ≈ (µs)
GTX TITAN Black	1.53	1.53	≈ 6.53
GTX TITAN X	1.53	1.53	≈ 6.53
TITAN V	0.79	0.79	≈ 5.79
RTX 2080 Ti	0.84	0.84	≈ 5.84
RTX 4070	1.02	1.02	≈ 6.02

4. Conclusion

The gpuSwapCol kernel performs a constant amount of work per thread and moves approximately 0.5 MiB of data per launch. Given the high peak FLOPs of all five GPUs, the kernel is dominated by memory bandwidth and kernel launch latency (~5 µs).

Resulting per-launch times are in the 5.8–6.5 µs range, with only small variation between older and newer GPUs because the fixed launch overhead is a significant fraction of the total runtime.

These predictions are approximate but consistent with the CUDA kernel performance template and typical behavior of small, memory-bound kernels.