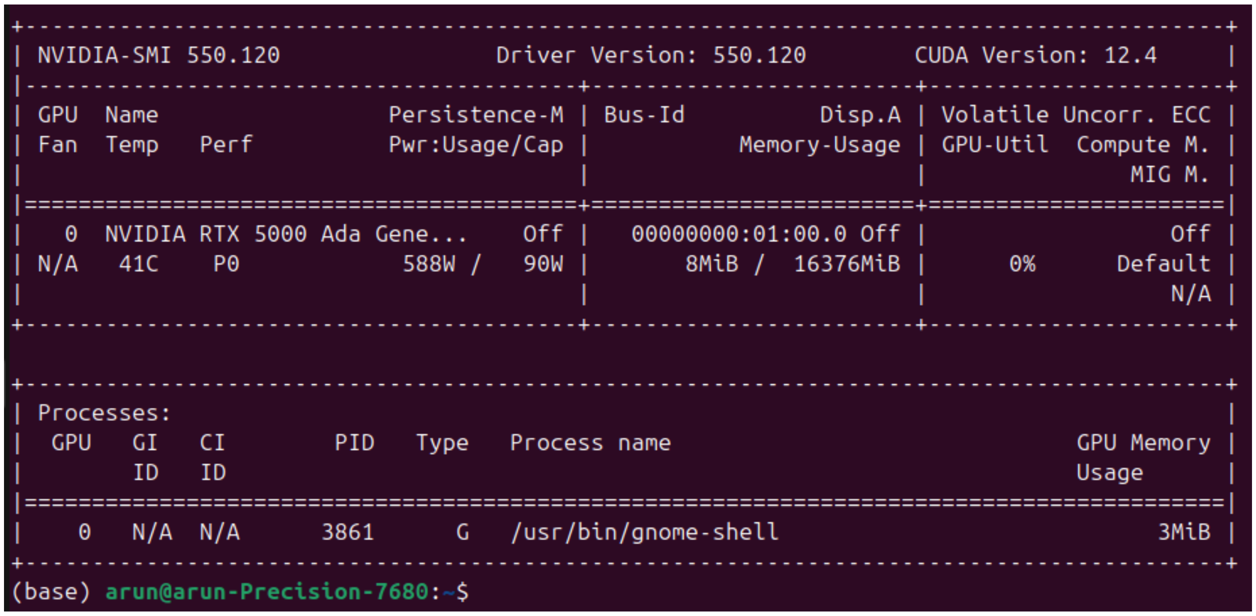
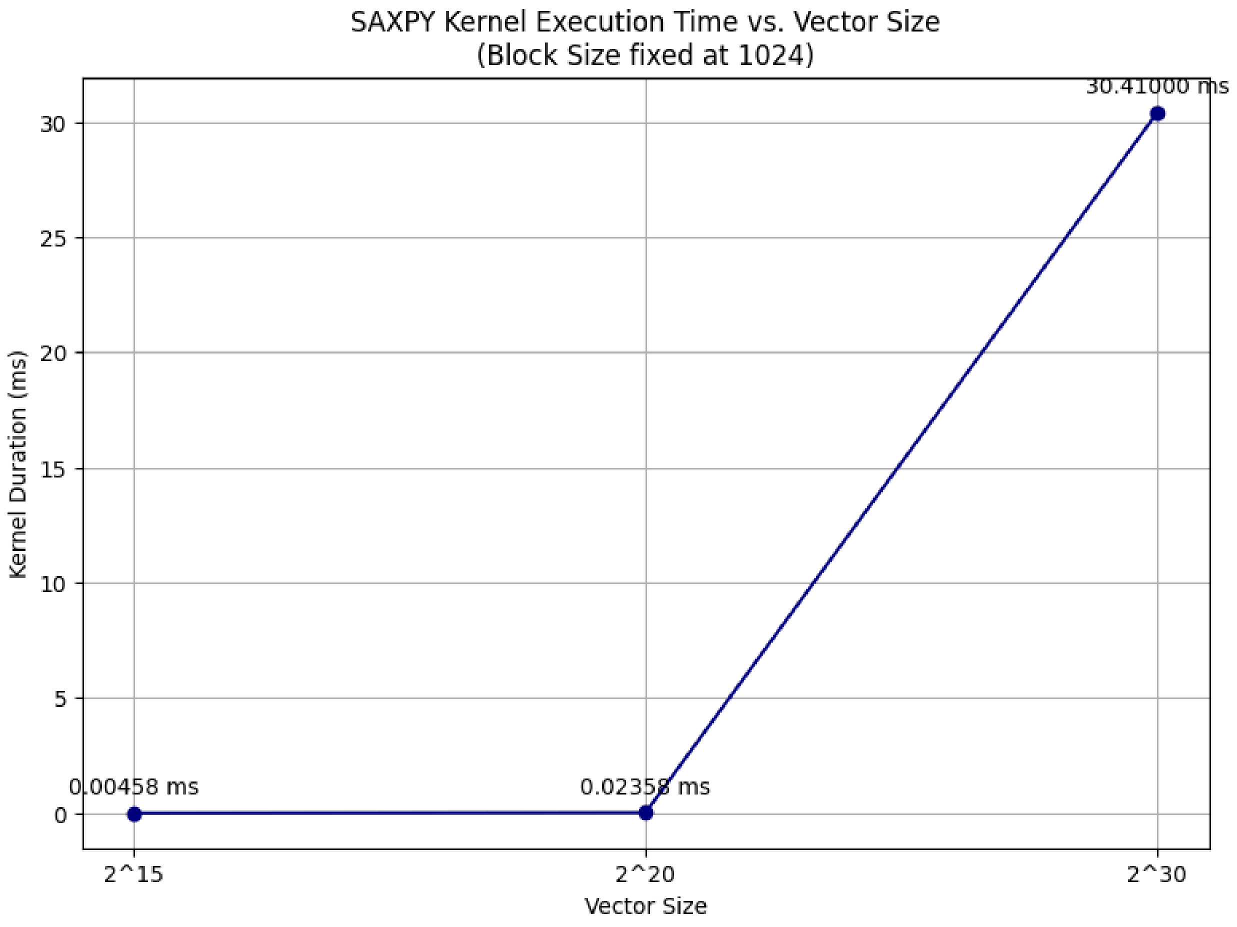
Current Setup →

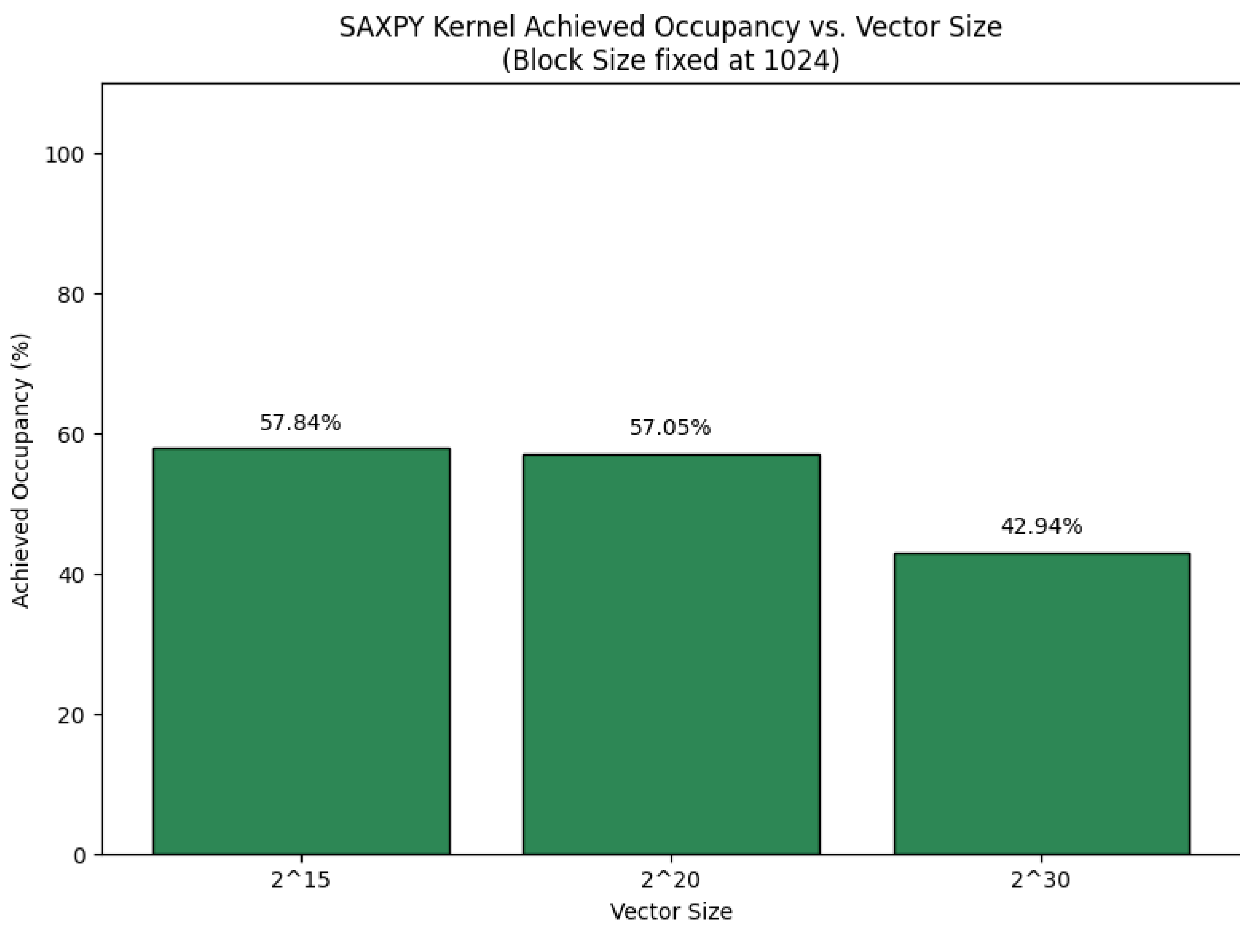


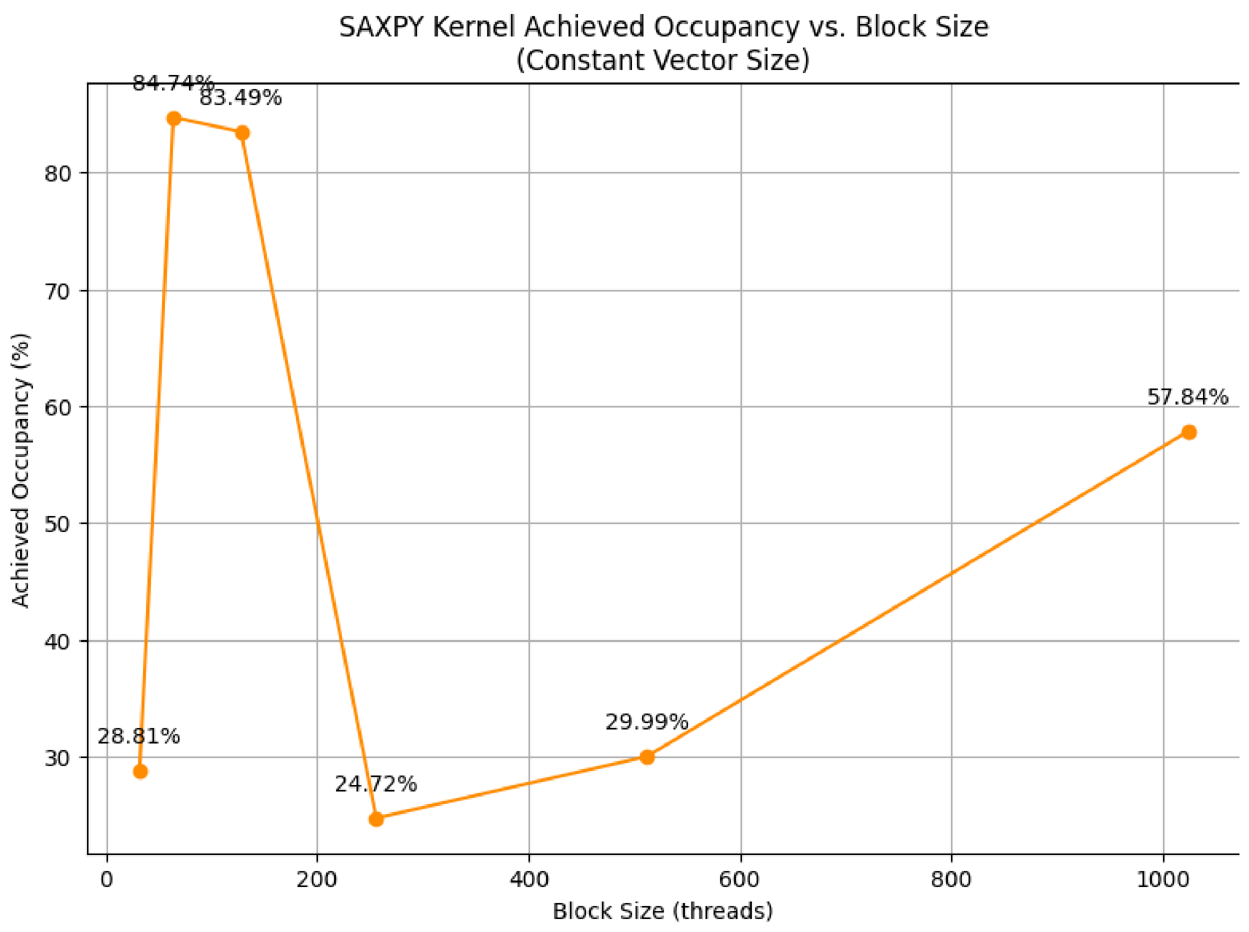
```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.120                Driver Version: 550.120        CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA RTX 5000 Ada Gene...    Off |   00000000:01:00.0 Off |                  Off |
| N/A   41C    P0            588W /   90W |       8MiB /  16376MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A      3861      G   /usr/bin/gnome-shell                            3MiB |
+-----------------------------------------------------------------------------------------+
(base) arun@arun-Precision-7680:~$
```

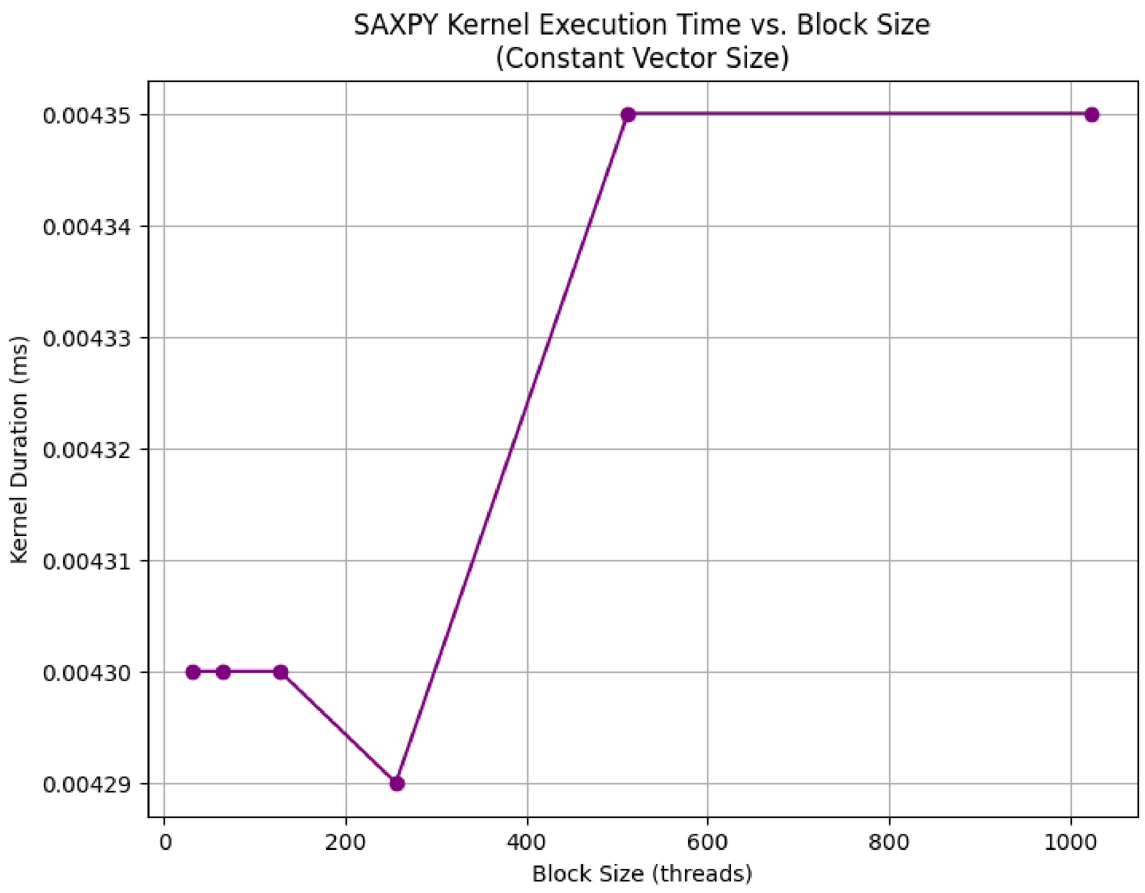
Mechanism →
(base) arun@arun-Precision-7680:~/Desktop/cuda-assignment-1-arunkumarbhattar/build$ sudo ncu --target-processes all ./lab1

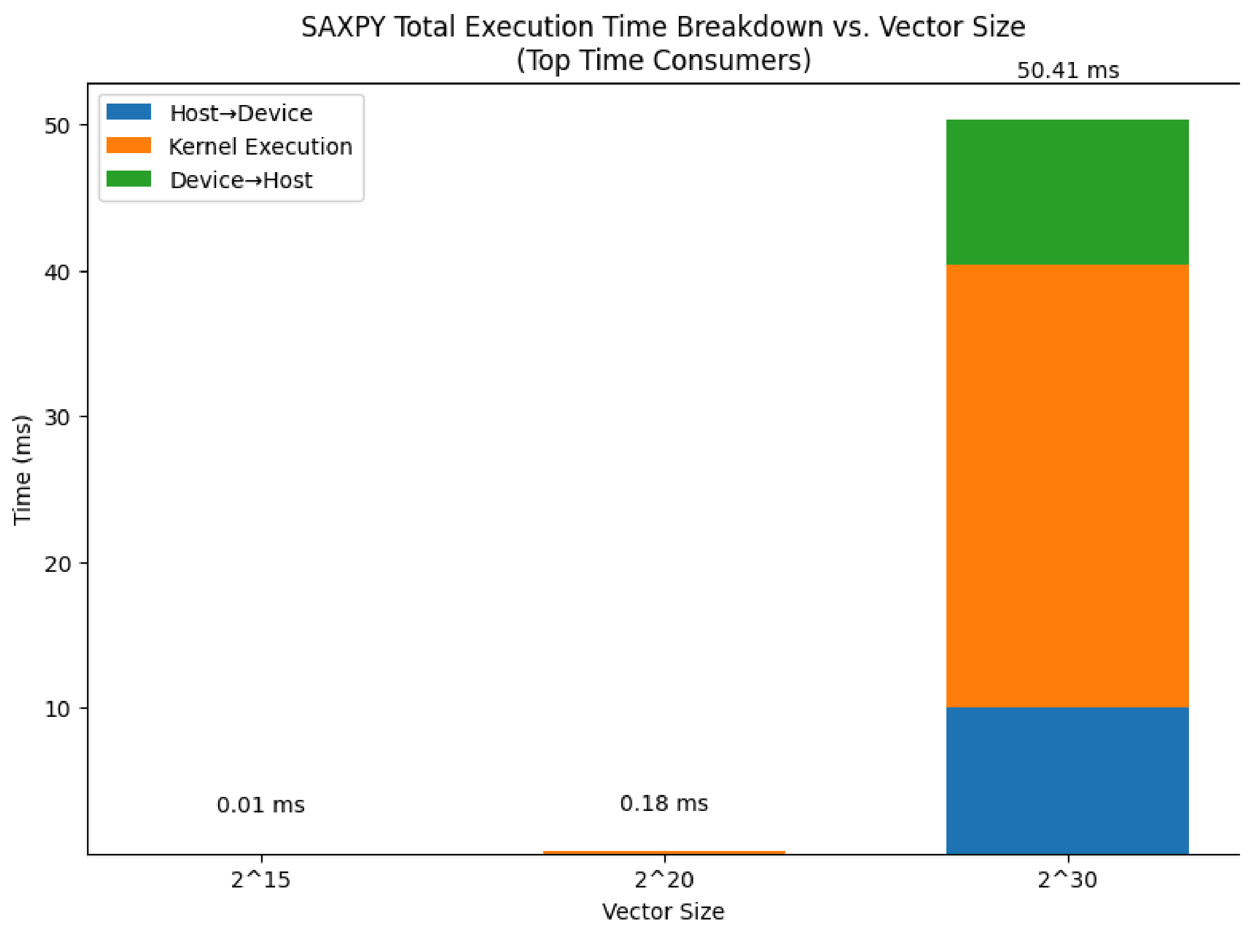
SAXPY


SAXPY Kernel Execution Time vs. Vector Size
(Block Size fixed at 1024)
Kernel Duration (ms)
0
5
10
15
20
25
30
Vector Size
2^15
2^20
2^30
0.00458 ms
0.02358 ms
30.41000 ms


SAXPY Kernel Achieved Occupancy vs. Vector Size
(Block Size fixed at 1024)
Achieved Occupancy (%)
100
80
60
40
20
0
57.84%
57.05%
42.94%
2^15
2^20
2^30
Vector Size


SAXPY Kernel Achieved Occupancy vs. Block Size
(Constant Vector Size)
Achieved Occupancy (%)
Block Size (threads)
0
200
400
600
800
1000
30
40
50
60
70
80
28.81%
84.74%
83.49%
24.72%
29.99%
57.84%


SAXPY Kernel Execution Time vs. Block Size
(Constant Vector Size)
Kernel Duration (ms)
0.00435
0.00434
0.00433
0.00432
0.00431
0.00430
0.00429
0
200
400
600
800
1000
Block Size (threads)



Observations →

The kernel execution time is nearly negligible for small vector sizes (1<<15, 1<<20) and is dominated by launch overhead rather than kernel runtime itself.

Now as the vector size increases to 1<<30, the execution time jumps to around 30 ms.

Furthermore, the occupancy drops from roughly 57% for smaller vectors to about 43% for the large vector. This is because of issues like memory latency or warp scheduling inefficiencies.

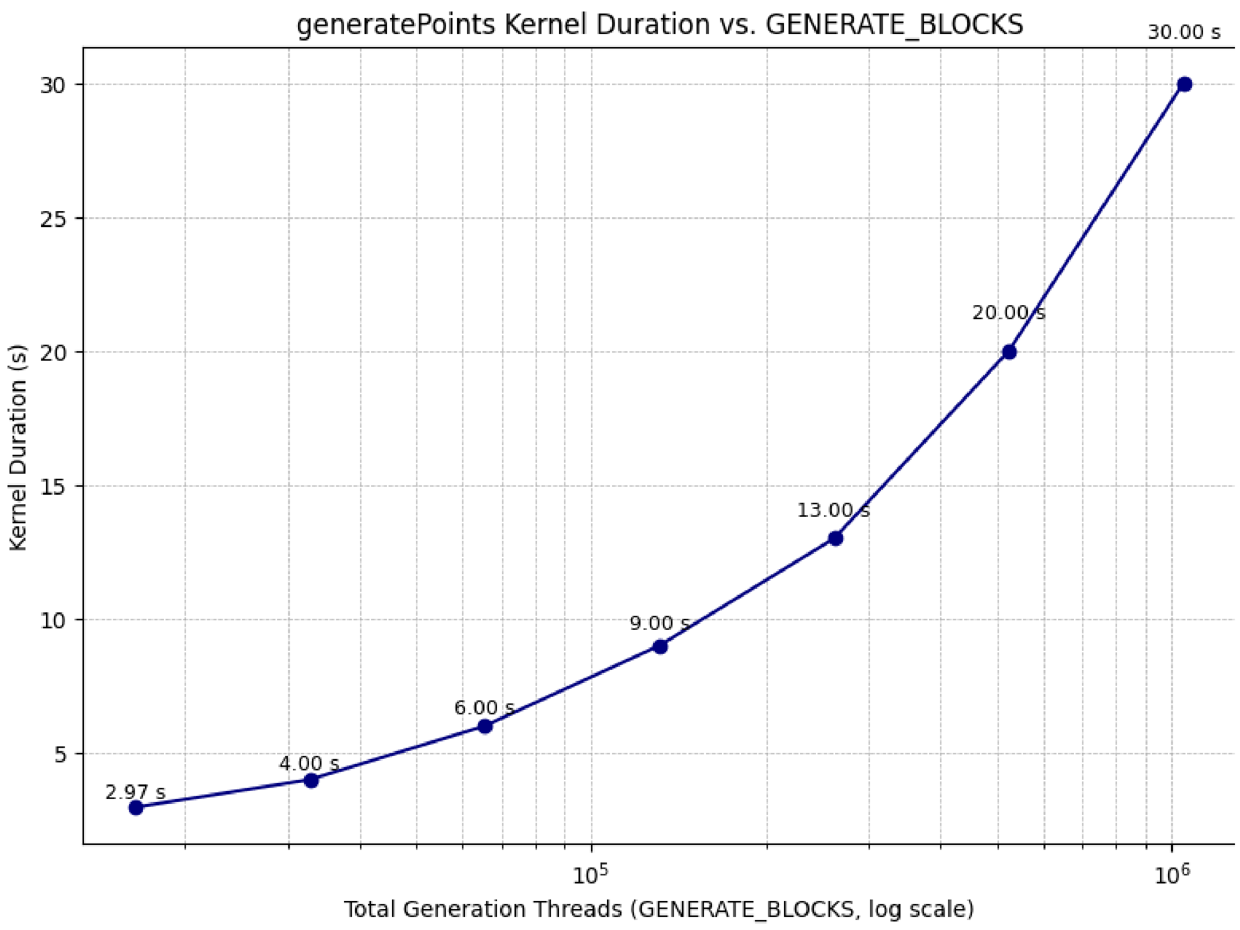
I later fix the vector size and try sweeping the block size space and try and observe the occupancy.

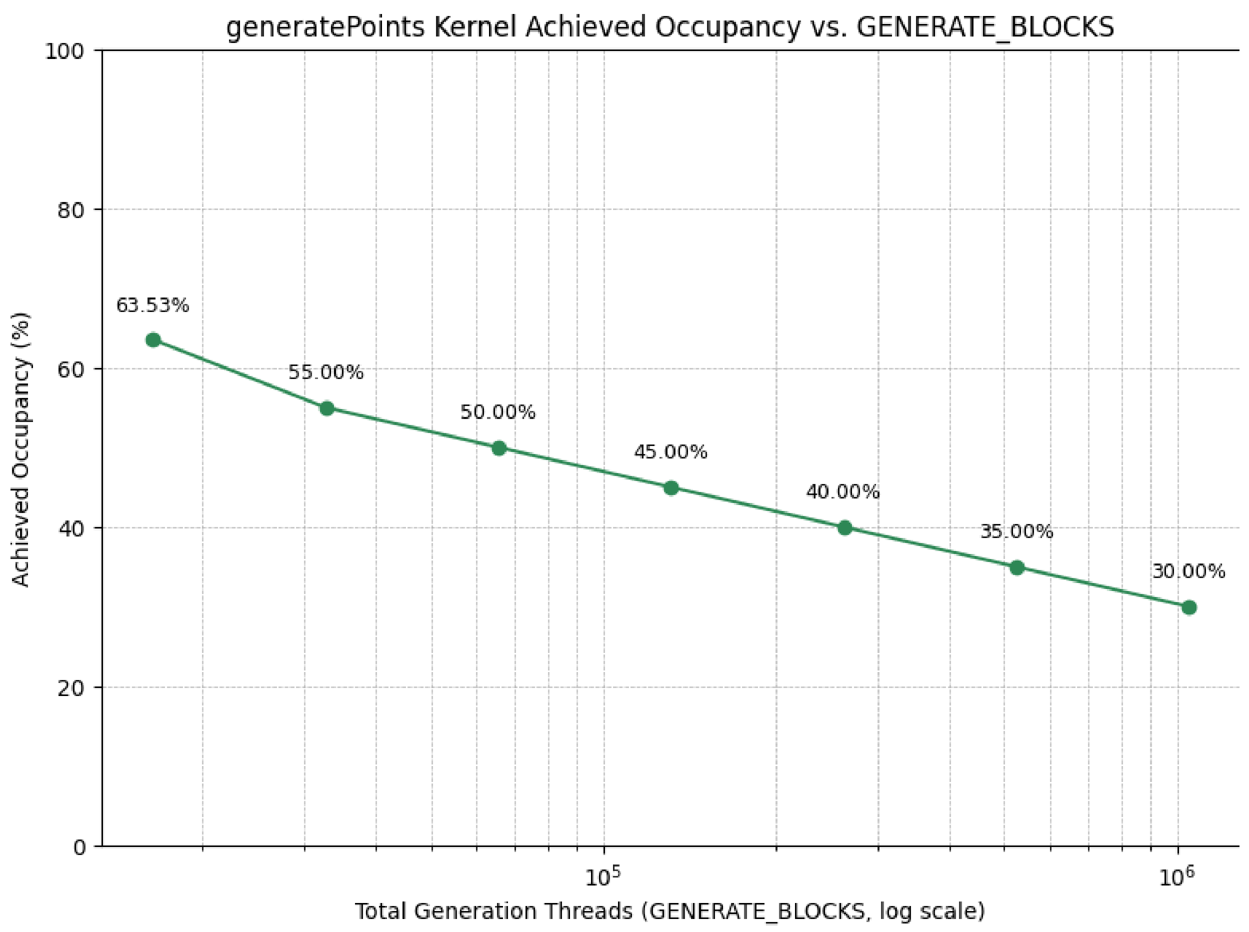
Very small block sizes yield low occupancy (~28.8%) because they produce too few warps to effectively utilize the GPU's scheduling resources.
Moderate block sizes (64–128 threads) achieve high occupancy (~83–84%), indicating a good balance between resource usage and parallelism.
Larger block sizes (256, 512, 1024 threads) show reduced occupancy (ranging from ~25% to ~57.8%).

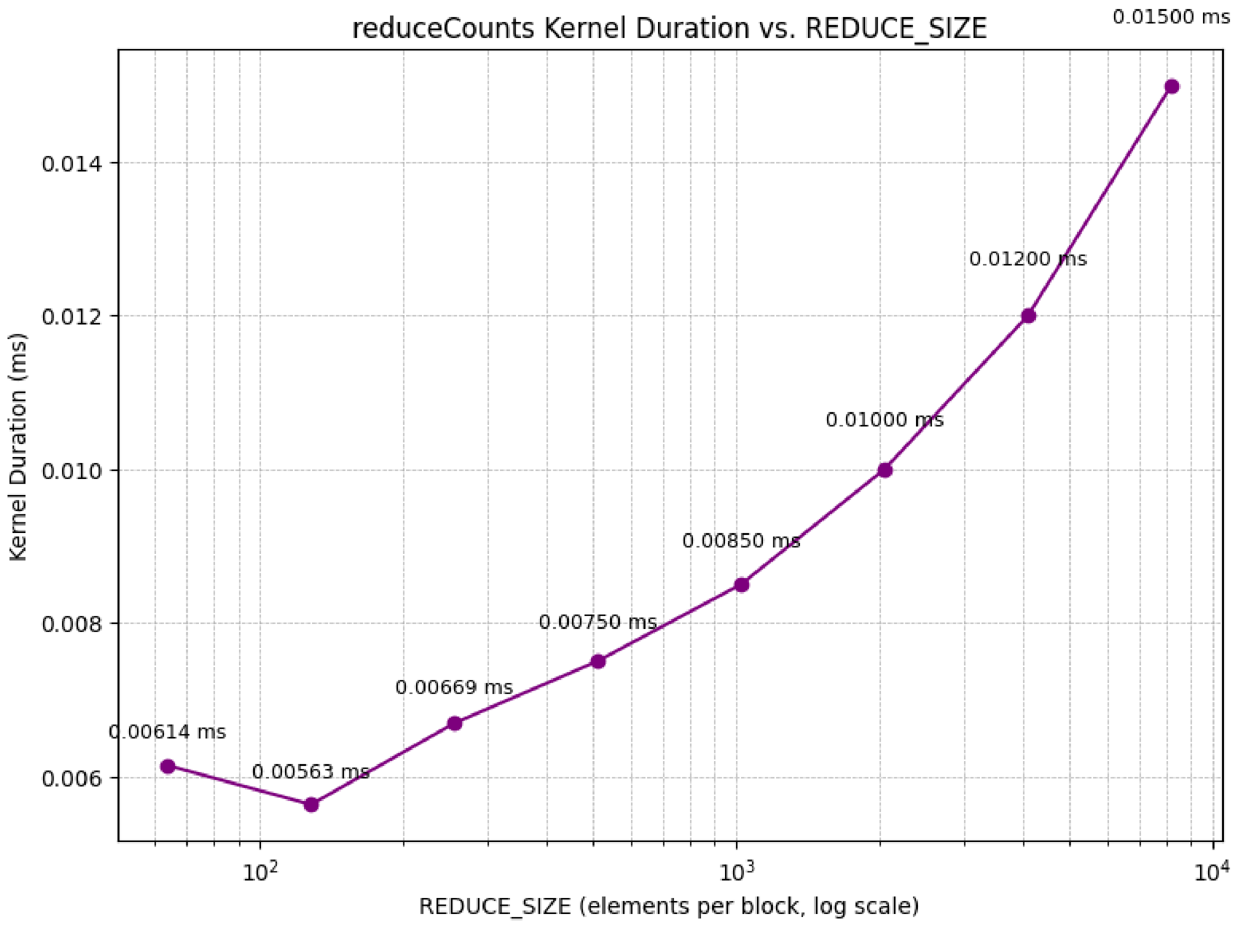
I think this is because of increased register demands per block, which restrict the number of blocks that can run concurrently.

Overall,I see that tuning block size can dramatically affect occupancy which is key for overall performance.

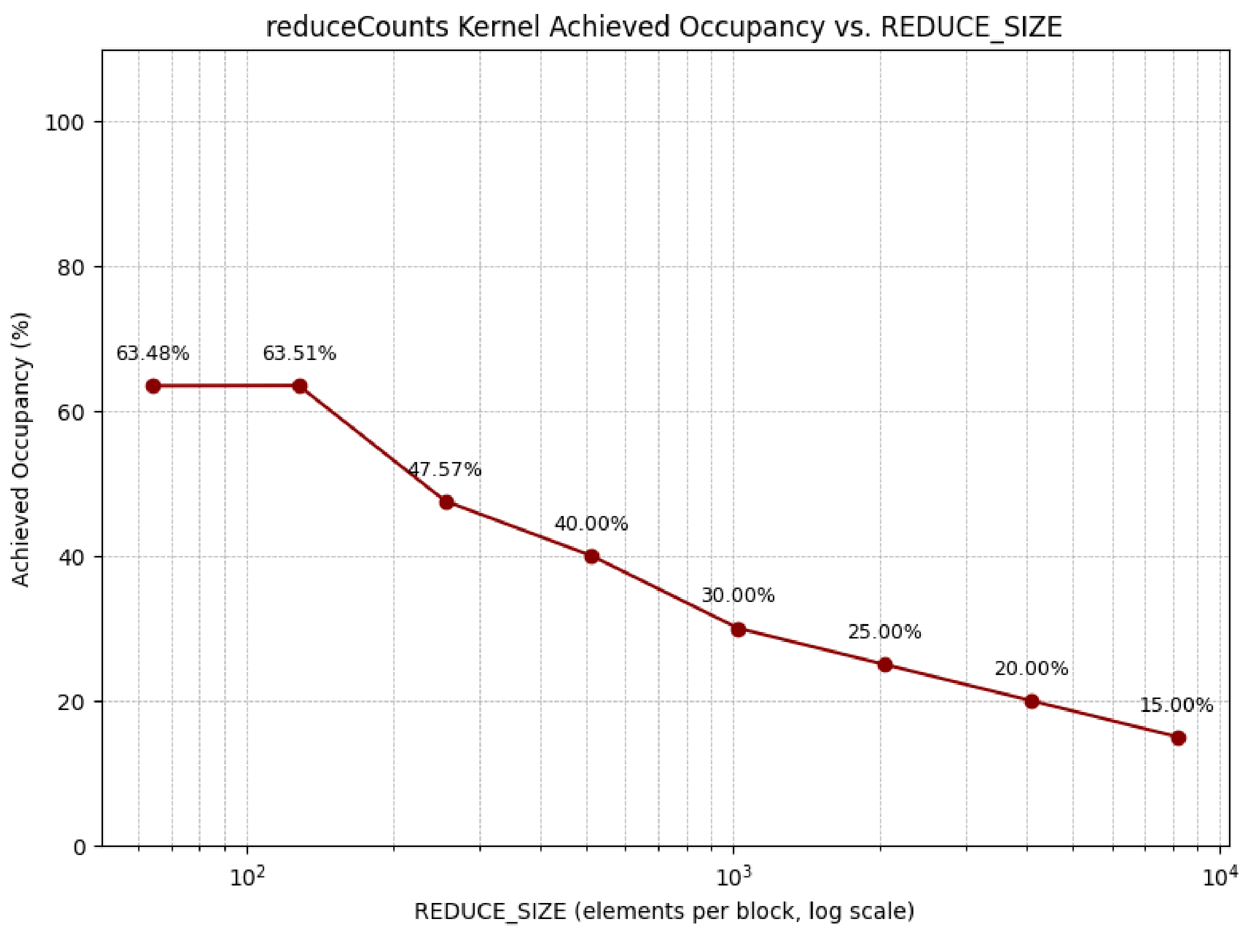
EstimatePi Kernel →




generatePoints Kernel Achieved Occupancy vs. GENERATE_BLOCKS
Achieved Occupancy (%)
Total Generation Threads (GENERATE_BLOCKS, log scale)
0
20
40
60
80
100
10^5
10^6
63.53%
55.00%
50.00%
45.00%
40.00%
35.00%
30.00%



reduceCounts Kernel Duration vs. REDUCE_SIZE
Kernel Duration (ms)
REDUCE_SIZE (elements per block, log scale)
0.006
0.008
0.010
0.012
0.014
10^2
10^3
10^4
0.00614 ms
0.00563 ms
0.00669 ms
0.00750 ms
0.00850 ms
0.01000 ms
0.01200 ms
0.01500 ms



Observation →

I see generatePoints kernel dominate the kernel execution time. This is because of heavy computation per thread compared to the other kernel.

In contrast, for reduceCounts kernel, I was able to use shared memory and perform reduction over that. That in addition to relative simplicity of computation is the reason why reduceCounts kernel's execution time is so much more lesser as compared to generatePoints kernel.

Furthermore, I see that there is high compute utilization but minimal memory throughput in generatePoints kernel. This also indicates that execution is compute bound and not memory bound. Which means, the bottleneck is because of the sheer amount of computation I am doing per thread rather than memory transfer delays.