1. **How many warps can an SM hold (hardware/occupancy limit)?**
2. **How many active warps do you need to hide memory/latency (performance/throughput requirement)?**

- **Warp size** (almost all NVIDIA GPUs): 32 threads/warp.

- **Active warps per SM (hardware max)** depends on the GPU architecture. You don't guess this — the GPU datasheet gives it. Typical modern values: SM can support dozens of warps (e.g., 64 warps = 2048 threads if warp size = 32).

- **Occupancy** = active threads / max threads per SM. You can compute active warps from resource limits (threads, registers, shared memory).

To compute active warps per SM:

1. Find the max blocks per SM limited by each resource:
   - by threads: `blocks_threads_limit = floor(max_threads_per_SM / threads_per_block)`
   - by registers: `blocks_regs_limit = floor(total_regs_per_SM / (regs_per_thread * threads_per_block))`
   - by shared mem: `blocks_shmem_limit = floor(shared_mem_per_SM / shared_mem_per_block)`
2. `blocks_per_SM = min(blocks_threads_limit, blocks_regs_limit, blocks_shmem_limit, hardware_block_limit)`
3. Active warps per SM = `blocks_per_SM * (threads_per_block / warp_size)`
4. Occupancy = `(active warps per SM * warp_size) / max_threads_per_SM`.

Example:

GPU limits: `max_threads_per_SM = 2048`, `total_regs_per_SM = 65536`, `shared_mem_per_SM = 64 KB`.
Kernel: `threads_per_block = 256`, `regs_per_thread = 32`, `shared_mem_per_block = 16 KB`.

- threads limit: `2048 / 256 = 8` blocks
- regs per block = `256 * 32 = 8192` → regs limit: `65536 / 8192 = 8` blocks
- shmem limit: `64KB / 16KB = 4` blocks

So `blocks_per_SM = min(8,8,4) = 4` blocks.

Active warps per SM = `4 * (256 / 32) = 4 * 8 = 32 warps`
Active threads = `32 * 32 = 1024` → occupancy = `1024 / 2048 = 50%`.

2) Warps needed to *hide latency* (performance / throughput)

This is a modeling estimate: how many *independent* warps you need so the SM can keep doing useful work while some warps are waiting (e.g., for memory).

A simple useful formula:

- Let `L` = memory latency in cycles (e.g., 300 cycles).
- Let `M` = average number of **useful** cycles a single warp executes between its memory requests (i.e., instructions between memory ops, in cycles). If a warp issues a memory access every `M` cycles, that warp will be stalled for `L` cycles on that memory access.
- Then **additional** warps required to cover that stall ≈ `ceil(L / M)`. Total warps needed ≈ `1 + ceil(L / M)` (the +1 is the original warp that issued the mem op).

So:

```
required_warps_to_hide ≈ 1 + ceil( L / M )
```

Worked numeric example:

- Memory latency `L = 300` cycles.
- A warp issues a memory access every `M = 50` cycles (on average).

`ceil(300 / 50) = 6` → `required_warps ≈ 1 + 6 = 7 warps` active to hide that memory stall.

Note: That's *modeling*. Real systems need more warps because:

- Different warps may hit different latencies (L varies).
- Warps may contend for other resources (execution units).
- Instruction throughput per warp may be less than one useful instruction per cycle.
- Branch divergence reduces effective parallelism.

So in practice, GPU architects target dozens of warps per SM

Tricks :

- Identify hardware limits: `max_threads_per_SM`, `total_regs_per_SM`, `shared_mem_per_SM`, `warp_size` (usually 32), `max_blocks_per_SM`.

- Compute blocks-per-SM limited by each resource and take minimum.

- Convert blocks → warps: `warps = blocks * (threads_per_block / warp_size)`.

- For latency-hiding questions, compute `required_warps ≈ 1 + ceil(L / M)` and compare to the active warps

Q1 . For a vector addition, assume that the vector length is 8000, each thread calculates one output element, and the thread block size is 1024 threads. The programmer configures the kernel launch to have a minimal number of thread blocks to cover all output elements. How many threads will be in the grid?

Blocks needed = `ceil(8000 / 1024)`.

Compute:

- `1024 × 7 = 7168`
- `1024 × 8 = 8192` → so `ceil(8000/1024) = 8` blocks.

Threads in the grid = `8 blocks × 1024 threads/block = 8192` threads.

(So 8192 threads are launched; 8192 − 8000 = 192 threads will be idle / do no useful work.)

Q2. If we want to copy 3000 bytes of data from host array h_A (h_A is a pointer to element 0 of the source array) to device array d_A (d_A is a pointer to element 0 of the destination array), what would be an appropriate API call for this data copy in CUDA?

cudaMemcpy(d_A, h_A, 3000, cudaMemcpyHostToDevice);

1. **Destination pointer** → `d_A` (device memory).
2. **Source pointer** → `h_A` (host memory).
3. **Size in bytes** → `3000`.
4. **Transfer direction** → `cudaMemcpyHostToDevice`.

If `cudaMemcpy` returns `cudaSuccess`, the copy succeeded.

Q3. If the SM of a CUDA device can take up to 1536 threads and up to 4 thread blocks. Which of the following block configuration would result in the largest number of threads in the SM?

Rule:
Active threads per SM = `min(threads_per_block × blocks_per_SM, max_threads_per_SM)`
where `blocks_per_SM` ≤ `max_blocks_per_SM`.

Threads per block	Max blocks possible (limited by 4 blocks/SM & 1536 threads)	Total threads
256	min(4, floor(1536 / 256) = 6) → 4 blocks	256 × 4 = **1024**
384	min(4, floor(1536 / 384) = 4) → 4 blocks	384 × 4 = **1536** ✓
512	min(4, floor(1536 / 512) = 3) → 3 blocks	512 × 3 = **1536** ✓
768	min(4, floor(1536 / 768) = 2) → 2 blocks	768 × 2 = **1536** ✓
1024	min(4, floor(1536 / 1024) = 1) → 1 block	**1024**

Possible

- 384 threads/block (4 blocks/SM),
- **or** 512 threads/block (3 blocks/SM),
- **or** 768 threads/block (2 blocks/SM).

Q4. For a vector addition, assume that the vector length is 2000, each thread calculates one output element, and the thread block size is 512 threads. How many threads will be in the grid?

Find number of blocks needed

$$\text{blocks} = \lceil \frac{\text{vector length}}{\text{threads per block}} \rceil$$

$$\text{blocks} = \lceil \frac{2000}{512} \rceil$$

- 512×3=1536
- 512×4=2048

So ceil =4 block

threads in grid=blocks×threads per block

so 4×512=2048 threads

Q5. each block executed 32T warps if we assign 3B to SM and each block has 256T. how many warps are there in sm ? also solved utlized and unutilized

1) Warps per block

Warp size = 32 threads.

$$\text{warps per block} = \frac{256 \text{ threads}}{32 \text{ threads/warp}} = 8 \text{ warps}$$

2) Total warps in the SM

With 3 blocks per SM:

$$\text{total warps} = 3 \text{ blocks} \times 8 \text{ warps/block} = 24 \text{ warps}$$

Total threads = $3 \times 256 = 768$ threads.

---

3) Utilized vs. unutilized (assuming SM max = 1536 threads)

- **Max warps per SM:**

$$\frac{1536 \text{ threads}}{32 \text{ threads/warp}} = 48 \text{ warps}$$

- **Utilized warps:** 24
- **Unutilized warps:** $48 - 24 = 24$

Or in threads:

- Utilized threads = 768
- Unutilized threads = $1536 - 768 = 768$

- **Active warps** in SM = **24**
- **Active threads** = **768**
- **Utilized** = 24/48=50%
- **Unutilized** = **50%** (24 warps / 768 threads idle)