PART I · CHAPTER 3

GPU Architecture: From Silicon to FLOPS

How modern accelerators actually work — the SM hierarchy, Tensor Core GEMM engine, HBM subsystem, Transformer Engine FP8, and the architecture decisions that shape every performance choice you make.

Book: AI/ML Infrastructure from Silicon to Scale — V2 Edition | **Author:** Venkat Vinjam | **Cross-ref:** Ch 3A: Accelerator landscape | Ch 3B: GPU roadmap detail | Ch 4: HBM deep dive | Ch 5: Thermal/TCO

> *"The GPU is not a magic compute box. It is a collection of physical tradeoffs — transistor density, wire length, heat dissipation, memory bandwidth — that determine, with mathematical precision, exactly how fast your model can go and precisely why it cannot go faster."*
>
> — Principal Performance Architect's GPU Principle

This chapter builds the mental model of the GPU that every subsequent chapter depends on. Chapters 3A and 3B provide the accelerator landscape and detailed generation roadmap. This chapter covers the architectural concepts stable across generations — the compute hierarchy, Tensor Cores, memory subsystem, and Transformer Engine — so you can reason about any GPU, current or future.

3.1 The GPU Compute Hierarchy

```
H100 SXM5 COMPUTE HIERARCHY [SHIPPED]:

GPU (full chip):
  132 Streaming Multiprocessors (SMs)
  Peak BF16: 989 TFLOPS  |  Peak FP8: 1,979 TFLOPS
  Peak FP32: 67 TFLOPS (non-Tensor Core CUDA Cores only)
  TDP: 700W  |  HBM3: 80 GB at 3.35 TB/s

Streaming Multiprocessor (SM):
  4 warp schedulers — can issue 4 independent warps per cycle
  64 CUDA Cores (FP32 ALUs) per SM
  1 fourth-generation Tensor Core per SM
  228 KB L1 Cache / Shared Memory (SMEM) — 50%/50% configurable split
  65,536 × 32-bit Registers per SM

Warp (fundamental execution unit):
  32 threads executing in lockstep (SIMT)
  Thread divergence: branches serialize → avoid in GPU kernels
  Memory access: coalesced 128-byte HBM transactions per warp

OCCUPANCY FORMULA:
  Occupancy = active_warps / max_warps_per_SM
  H100 max warps per SM: 64
  At 64 regs/thread: 65536/64 = 1024 threads = 32 warps → 50% occupancy
  At 32 regs/thread: 65536/32 = 2048 threads = 64 warps → 100% occupancy
  Register spilling (overflow to L2/HBM): 200–800× latency penalty per access
```

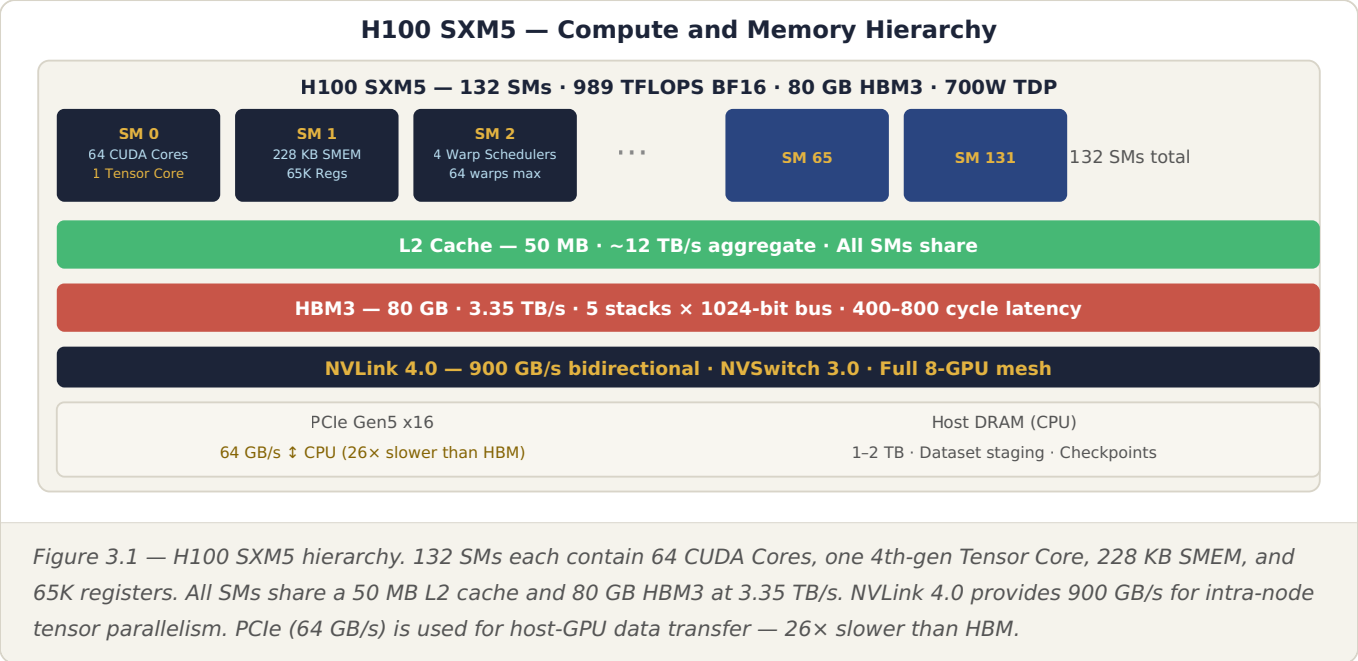


Figure 3.1 — H100 SXM5 hierarchy. 132 SMs each contain 64 CUDA Cores, one 4th-gen Tensor Core, 228 KB SMEM, and 65K registers. All SMs share a 50 MB L2 cache and 80 GB HBM3 at 3.35 TB/s. NVLink 4.0 provides 900 GB/s for intra-node tensor parallelism. PCIe (64 GB/s) is used for host-GPU data transfer — 26× slower than HBM.

3.2 Tensor Cores — The GEMM Engine

Generation	GPU	Min Precision	Peak BF16	Peak FP8	Tile Op
1st gen (Volta)	V100	FP16	125 TFLOPS	N/A	16×16×16 FP16
3rd gen (Ampere)	A100	BF16	312 TFLOPS	N/A	16×16×16 BF16
4th gen (Hopper)	**H100**	**FP8**	**989 TFLOPS**	**1,979 TFLOPS**	**WGMMA (warp-group)**
5th gen (Blackwell)	B200	NVFP4	~2,250 TFLOPS	~4,500 TFLOPS	WGMMA + larger tiles

```
TENSOR CORE ALIGNMENT REQUIREMENTS — the silent performance tax:

BF16/FP8 GEMM: Matrix dimensions must be multiples of 16.
  M=4096: aligned → full Tensor Core throughput ✓
  M=4097: padded to 4112 → 0.37% waste (negligible)
  M=13:   padded to 16   → 18.75% compute waste ✗ (avoid this)

PRODUCTION CONSEQUENCE:
  d_model, d_head, d_ffn must all be ≡ 0 mod 16.
  LLaMA-3 70B: d_model=8192 ✓, d_head=128 ✓, d_ffn=28,672 ✓
  A custom model with d_head=130 wastes 1.5% Tensor Core throughput.
  At 1,000 GPUs, $3/GPU-hr, 24×7: 1.5% = $39,000/year from misalignment.
```

3.3 The Memory Subsystem

TIER 1 Registers

256 KB/SM · ~33 MB total · 1-cycle latency · thread-private

65,536 × 32-bit registers per SM. At 128 regs/thread → max 512 threads → 25% occupancy. Register spilling maps overflow to HBM L2 at 200–800× latency. Profile: Nsight Compute "Registers per thread" and "Local memory spill stores."

TIER 2 L1/SMEM

228 KB/SM · ~80–100 TB/s aggregate · 20–30 cycle latency · thread block shared

Explicitly programmer-managed. The key mechanism for GEMM tiling: tile A [TILE_M × TILE_K] and B [TILE_K × TILE_N] into SMEM, compute all inner products from SMEM. Bank conflicts (32 banks, 4-byte words) reduce SMEM BW to peak/32. Fix: pad inner dimension — `smem[32][33]` instead of `smem[32][32]`.

TIER 3 L2 Cache

50 MB · ~12 TB/s aggregate · 200–400 cycle latency · all SMs share

Hardware-managed. Limited benefit for decode (weights far exceed 50 MB). Useful for KV cache hot blocks, embedding table lookups, LayerNorm running stats. Check L2 hit rate in Nsight Compute to classify streaming vs. reuse workloads.

TIER 4 HBM3

80 GB (H100) / 141 GB (H200) / 192 GB (B200) · 3.35 / 4.8 / 8.0 TB/s · 400–800 cycle latency

Per-GPU global memory. Holds model weights, activations, KV cache. Primary bottleneck for memory-bound operations. Achieving >80% utilization requires coalesced access, tiling, and quantization. Uncoalesced warp access: 3–12% of peak bandwidth (32× penalty).

TIER 5 Host DRAM

1–2 TB · 64 GB/s (PCIe Gen5) one-way · 10,000–50,000 cycle GPU-view latency

CPU memory. Dataset staging, checkpoint buffers, model parameter spill. PCIe is 26× slower than HBM — frequent GPU↔CPU transfers are catastrophic for performance. Minimize at all costs in the inference hot path.

3.4 The Transformer Engine — H100's Key Innovation

```
TRANSFORMER ENGINE (H100 specific):

Problem: FP8 delivers 2× GEMM throughput vs BF16 but requires per-tensor
         scaling to prevent numerical overflow — complex to manage manually.

Solution: Hardware-managed Transformer Engine:
  1. Monitors activation magnitudes per tensor automatically
  2. Computes per-tensor scale factors (no user code required)
  3. Casts to FP8 E4M3 (weights/activations) for GEMM
  4. Accumulates in FP32 (prevents numerical degradation)
  5. Outputs in BF16 for next layer

  FP8 E4M3: 1 sign + 4 exponent + 3 mantissa → range ±448 (precision-optimized)
  FP8 E5M2: 1 sign + 5 exponent + 2 mantissa → range ±57344 (range-optimized, gradients)

RESULT:
  FP8 GEMM: 1,979 TFLOPS (2× BF16 989 TFLOPS) [SHIPPED]
  Automatic quality preservation — no calibration step required
  This is the primary reason H100 delivers ~3× training throughput over A100
  (FP8 throughput × NVLink BW improvement × architectural improvements)

NVFP4 (B200, 2024):
  4-bit float with 2 exponent + 1 mantissa bits
  Block-level scaling for quality preservation
  9,000 TFLOPS FP4 on B200 [SHIPPED]
```

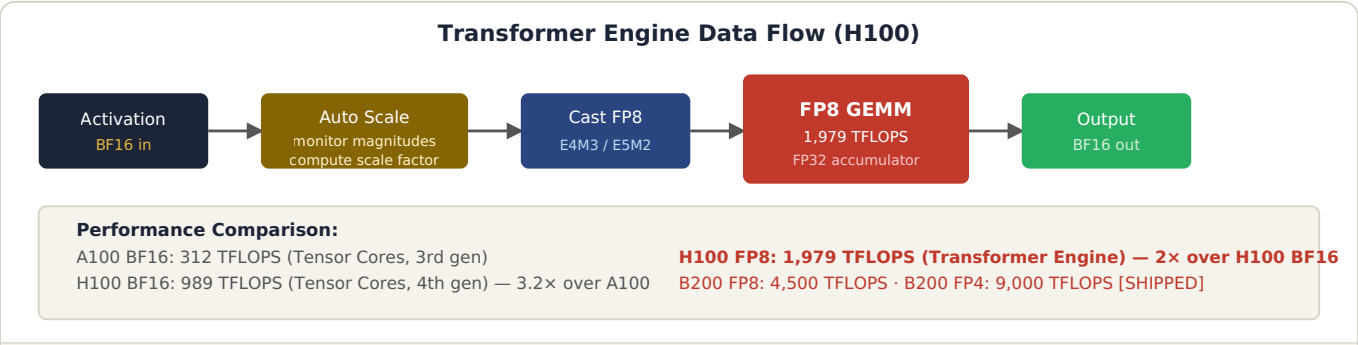


Figure 3.2 — Transformer Engine data flow on H100. The hardware automatically monitors activation magnitudes, computes per-tensor FP8 scale factors, executes GEMM in FP8 with FP32 accumulation, and outputs in BF16. No user code or calibration required. This delivers 1,979 TFLOPS FP8 — 2× over H100 BF16 and 6.3× over A100 BF16.

3.5 GPU Generation Comparison

GPU	Year	BF16 TFLOPS	FP8 TFLOPS	HBM	HBM BW	NVLink BW	TDP	Key Innovation
V100 SXM2	2018	125	N/A	32 GB HBM2	900 GB/s	300 GB/s	300W	First Tensor Cores (FP16)
A100 SXM4	2020	312	N/A	80 GB HBM2e	2.0 TB/s	600 GB/s	400W	3rd gen TC, BF16 native, MIG
H100 SXM5	2022	**989**	**1,979**	80 GB HBM3	**3.35 TB/s**	900 GB/s	700W	TE FP8, NVLink 4.0, 4th gen TC
H200 SXM5	2024	989	1,979	**141 GB HBM3e**	**4.8 TB/s**	900 GB/s	700W	HBM3e upgrade (memory only)
B200 SXM6	2024	**~2,250**	**~4,500**	**192 GB HBM3e**	**8.0 TB/s**	**1,800 GB/s**	1,000W	Dual-die MCM, NVFP4, NVLink 5.0

H200 vs H100 insight:

H200 has identical TFLOPS to H100 — same compute die, same Transformer Engine. The only changes are HBM3e (141 vs 80 GB, 4.8 vs 3.35 TB/s). ROI is positive ONLY for bandwidth-bound workloads (decode at small batch, long-context inference). For compute-bound prefill workloads at large batch: zero throughput improvement.

3.6 SXM vs PCIe Form Factor

```
SXM (Server eXchange Module):
  Die-to-die connection to NVSwitch via copper traces on substrate (no PCIe)
  H100 SXM: 900 GB/s NVLink · 700W TDP · requires HGX/DGX/OVX servers

  PCIe form factor:
  Standard PCIe x16 card
  H100 PCIe: 600 GB/s NVLink (PCIe replaces partial NVLink budget) · 350W
  Fits in standard 1U/2U server without specialized power delivery

WHEN SXM MATTERS (TP over NVLink):
  TP=2, 70B model, 80 layers, one AllReduce per layer, ~0.5 GB message:
    SXM (900 GB/s): 0.5/900 =  0.56ms × 80 = 44ms overhead per inference step
    PCIe (600 GB/s): 0.5/600 = 0.83ms × 80 = 67ms overhead per inference step
    Difference at TPOT=42ms: 44ms vs 67ms = SXM 52% faster TP communication

WHEN PCIe IS ACCEPTABLE:
  Data Parallelism only (no TP): no inter-GPU communication at inference time
  Single-GPU inference (model fits with quantization): no NVLink needed
  Budget-constrained deployments where TP latency overhead is acceptable
```

3.7 AMD MI300X — The Memory Architecture

> **Architectural approach:**
>
> MI300X uses a chiplet design — 6 compute XCDs (CDNA 3 architecture) + 4 memory controller MCDs stacked on a passive silicon interposer — breaking the reticle-size limit that forces NVIDIA to MCM on Blackwell. The result: 192 GB HBM3 in a single package with 5.3 TB/s bandwidth.

Property	H100 SXM5	MI300X	H200 SXM5	Winner
Peak BF16	989 TFLOPS	**1,307 TFLOPS**	989 TFLOPS	MI300X (+32%)
HBM Capacity	80 GB HBM3	**192 GB HBM3**	141 GB HBM3e	MI300X (2.4×)
HBM Bandwidth	3.35 TB/s	**5.3 TB/s**	4.8 TB/s	MI300X (+22% vs H200)
TPOT (70B BF16, b=1)	41.8 ms	**26.4 ms**	29.2 ms	MI300X (37% faster)
70B BF16 fits single GPU?	**No (140 GB)**	**Yes (192 GB)**	**No (141 GB)**	MI300X only
Ridge Point (BF16)	295 FLOP/byte	247 FLOP/byte	206 FLOP/byte	H200 (most BW-rich)

The Memory Moat — cost consequence:

MI300X serves LLaMA-3 70B (140 GB BF16) on one GPU. H100 requires TP=2 (two GPUs, AllReduce every layer). At 100 serving instances: MI300X saves 100 GPUs. At $25K/GPU = $2.5M capex, $3/GPU-hr × 100 GPUs × 8,760 hrs/yr = $2.6M/year OpEx. Over 3 years: $10.3M savings purely from the memory moat.

3.8 Custom Accelerators

Accelerator	Key Strength	When to Use	When NOT to Use
Google TPU v5p	Gemini training at trillion-token scale on GCP	Large-scale training, JAX ecosystem, GCP-committed	Non-GCP infrastructure; irregular kernels
AWS Trainium2	60% cost reduction vs H100 for LLM training on AWS	AWS-native training at scale with Neuron SDK	Non-AWS; CUDA-dependent model code
Intel Gaudi 3	Open SW stack, OAM form factor, IBFabric	Open ecosystem, heterogeneous clusters	CUDA kernel dependency; NVIDIA-only stack
Groq LPU	Sub-1ms TTFT for small models (<13B), SRAM-only	Ultra-low latency real-time inference	Large models; high-throughput batching

Chapter 3 Quick Reference

Component	H100 Spec	Why It Matters
SM count	132 SMs	Total parallelism. Max concurrent warps = 132 × 64 = 8,448.
SMEM per SM	228 KB	Tiling budget. FlashAttention Q+K+V tiles must fit here (<228 KB).
Registers per SM	65,536 × 32-bit	At 128 regs/thread → 25% occupancy → register spill risk.
HBM3 bandwidth	3.35 TB/s	The memory roof. TPOT_min = 140 GB / 3.35 TB/s = 41.8 ms for 70B BF16.
L2 cache	50 MB	~35% of embedding table for 7B model. Hit rate visible in NCU.
NVLink 4.0	900 GB/s	TP AllReduce constraint. TP must stay within NVSwitch domain.
Transformer Engine	FP8 = 2× BF16	The reason H100 is 3× faster than A100 for transformers. Auto-scaling — no user code.
Ridge point (BF16)	295 FLOP/byte	The regime boundary. Below = memory-bound; above = compute-bound.
MI300X memory moat	192 GB HBM3	Eliminates TP=2 for 70B serving. $10M+ 3-year TCO savings at 100 instances.

Chapter 3 Review Questions

0. **An SM has 65,536 registers. A kernel uses 128 registers per thread. What is the maximum occupancy? If you restructure the kernel to use 64 registers per thread (without changing FLOP count), what happens to occupancy and why might throughput increase even if the kernel now has more L2 accesses?**
   What is the Nsight Compute metric that tells you whether register pressure is hurting occupancy, and what threshold indicates a problem?

0. **Explain what the Transformer Engine does automatically that A100 required manual management of. Why does automatic per-tensor FP8 scaling preserve quality while naïve FP8 casting does not?**
   What is the difference between FP8 E4M3 and FP8 E5M2, and why is E4M3 preferred for weights/activations while E5M2 is preferred for gradients?

0. **H100 TP=2 serving LLaMA-3 70B: 80 layers, one AllReduce per layer, 0.5 GB message per AllReduce. Compare total AllReduce overhead over SXM NVLink (900 GB/s) vs PCIe NVLink (600 GB/s). At TPOT_min = 41.8ms, what fraction of TPOT is consumed by AllReduce in each case?**

0. **A company is choosing between 100 H100 SXM (TP=2 for 70B serving) and 50 MI300X (TP=1). Calculate TPOT_min for each configuration in BF16. Then compute the 3-year TCO at $25K/GPU (H100), $30K/GPU (MI300X), and $3/GPU-hr OpEx. Which is cheaper, and under what assumption does the answer flip?**

0. **Tensor Core alignment requires matrix dimensions to be multiples of 16. A custom attention module uses d_head=96 (not a multiple of 16 — needs padding to 112). What fraction of Tensor Core throughput is wasted? At 1,000 H100 GPUs running 24/7 at $3/GPU-hr, what is the annual cost of this misalignment?**

0. **Design the GPU infrastructure for a production deployment serving LLaMA-3 70B at 10K concurrent users, 4096-token average context, P99 TPOT < 25ms. Evaluate H100 (TP=2), H200 (TP=1, requires INT4 to fit), and MI300X (TP=1, BF16). Show GPU count, total capex, and 3-year TCO for each. Which do you recommend?**

Chapter Cross-References

Ch 3A Accelerator landscape — TPU, Trainium, Gaudi, Groq, MTIA, and META MTIA. When custom silicon outcompetes GPUs and what the evaluation framework looks like.

Ch 3B GPU generation roadmap — H100 → H200 → B200 → GB300, MI300X → MI355X → MI400. Architecture deltas, performance per dollar progression, and TCO comparisons across generations.

Ch 4 HBM deep dive — the five-tier memory hierarchy, HBM3 physical architecture (TSV stacks, 1024-bit bus), and FlashAttention's impact on HBM utilization patterns. Ch 4 is where the HBM numbers in §3.3 are derived from silicon physics.

Ch 5 Thermal and TCO — the TDP numbers in §3.5 (700W H100, 1000W B200) feed directly into cooling architecture (liquid cooling required at >50 kW/rack), power cost per GPU-hour, and $/token/day TCO analysis.