PART I · CHAPTER 1

The AI/ML Performance Architecture Mindset

Why systems thinking scales — and what changes when you move to AI infrastructure. The roofline model, three performance regimes, and the seven analytical layers every principal architect applies before opening a profiler.

Book: AI/ML Infrastructure from Silicon to Scale — V2 Edition | **Author:** Venkat Vinjam | **Cross-ref:** Ch 4: Roofline deep dive | Ch 12: MFU benchmarking | Ch 17: Profiling tools

> *"The engineer who can walk into an architecture review and say: 'this design is memory-bandwidth-bound at these shapes — here is the roofline, here is the gap, here are two approaches to close it with their TCO implications' — that is the principal. This chapter builds that engineer."*
>
> — Book Mission Statement

Chapter 1 is the analytical skeleton on which every other chapter hangs. Every optimization in this book — FlashAttention tiling, KV cache paging, tensor parallelism topology, quantization bit-width selection — reduces to one of three quantities or a tradeoff among them: **arithmetic intensity, memory bandwidth, and communication volume**.

1.1 What Actually Changed — and What Did Not

The rise of large-scale AI feels like a rupture. But the underlying engineering discipline has not changed: *find the bottleneck, quantify the gap, close it at the lowest cost*. What changed is where the bottlenecks are and how far they cascade.

Traditional Systems Mindset	AI/ML Principal Architect Mindset
"Let me profile and see what's slow."	"Let me classify the regime first — compute, memory, or communication bound — then profile to confirm."
"This kernel is at 82% GPU utilization."	"82% SM utilization means nothing without arithmetic intensity. What is the FLOP/byte ratio at this batch size?"
"We need more bandwidth."	"How many bytes per FLOP does this operation require, and what bandwidth keeps the ALUs fed?"
"The AllReduce is slow."	"What is the message size, topology, algorithm (ring vs. tree), and effective bandwidth as a fraction of NVLink theoretical?"
"Add another node."	"What is the communication-to-computation ratio at this scale, and does adding a node help or hurt?"

> **The Scale Consequence**
>
> A 0.5% throughput loss that is a rounding error on a single GPU costs $2M/year at a 10,000-GPU cluster running at $3/GPU-hr. Communication is now a first-class citizen. Memory is the primary constraint. The system boundary expanded from a single server to a fleet of racks.

1.2 The Seven-Layer Performance Stack

Every AI/ML system is a stack of seven layers. Fixes applied to the wrong layer achieve nothing even when perfectly implemented. The principal's first question is always: *which layer is the active bottleneck?*

```
SEVEN-LAYER PERFORMANCE STACK:

Layer 7 — CLUSTER ORCHESTRATION
  Tools:   SLURM, Kubernetes, topology-aware placement
  Signal:  low GPU utilization across the fleet, gang-scheduling delays

Layer 6 — NETWORK / COMMUNICATION
  Tools:   NVLink, InfiniBand NDR, NCCL, AllReduce algorithms
  Signal:  comms:compute ratio >15%, NCCL bus BW <50% of theoretical

Layer 5 — DISTRIBUTED PARALLELISM
  Tools:   TP, PP, DP, ZeRO, FSDP, sequence parallelism
  Signal:  pipeline bubble >10%, communication not overlapped with compute

Layer 4 — FRAMEWORK + COMPILER
  Tools:   torch.compile, TensorRT, CUDA Graphs, XLA
  Signal:  graph breaks, kernel launch overhead dominating

Layer 3 — MEMORY SYSTEM
  Tools:   HBM access patterns, KV cache, fragmentation analysis
  Signal:  achieved BW <60% of HBM peak, high fragmentation, OOM

Layer 2 — GPU KERNEL
  Tools:   GEMM tiling, Tensor Core efficiency, SMEM occupancy
  Signal:  SM occupancy <50%, throughput below roofline ridge point

Layer 1 — SILICON PHYSICS
  Tools:   Clock, voltage, thermal throttle, ECC errors
  Signal:  SM clock below TDP target, GPU temp >83°C

RULE: Fix the lowest active bottleneck first.
The most expensive mistake: tuning Layer 2 (kernels) while a Layer 6
(AllReduce) bottleneck serializes with compute — GEMM gets 3% faster
while overall throughput does not move.
```

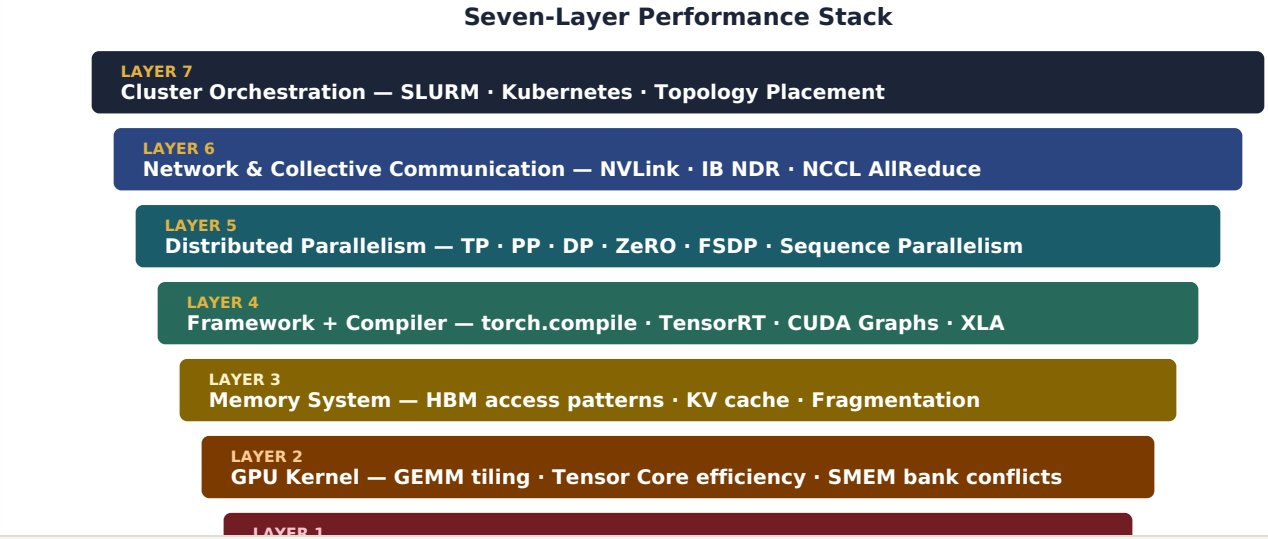


Figure 1.1 — The Seven-Layer Performance Stack. Fix the lowest active bottleneck first. Optimizing Layer 2 kernels while a Layer 6 AllReduce bottleneck exists is the most common performance engineering mistake in AI/ML infrastructure.

1.3 The Three Performance Regimes

Every GPU operation falls into exactly one of three regimes at any given (hardware, operation, batch size) tuple. Identifying the regime is step zero — before profiling, before code review, before any optimization decision.

```
REGIME CLASSIFICATION:

REGIME 1 — COMPUTE BOUND  (Arithmetic Intensity > Ridge Point)
  H100 ridge point: 989 TFLOPS ÷ 3.35 TB/s = 295 FLOP/byte
  Symptom: SM utilization >90%, HBM bandwidth well below peak
  Examples: Large GEMM prefill (AI ~1,365 at S=4096)
  Fix lever: FP8, NVFP4, better Tensor Core tile sizing
  Wrong fix: Adding HBM bandwidth — more BW won't help

REGIME 2 — MEMORY BANDWIDTH BOUND  (AI < Ridge Point)
  Symptom: HBM bandwidth near peak, SM compute utilization below peak
  Examples: GEMV decode batch=1 (AI ≈ 1), LayerNorm, RMSNorm
  Fix lever: Operator fusion, INT4/FP8 quantization, increase batch size
  Wrong fix: Faster compute — more TFLOPS won't help

REGIME 3 — COMMUNICATION BOUND
  Symptom: GPU idle >20% during collective ops (visible in Nsight trace)
  Examples: TP AllReduce at large cluster, PP pipeline bubble fill
  Fix lever: Overlap comm with compute, reduce volume, topology-aware placement
  Wrong fix: Faster single-GPU kernel — bottleneck is not on-GPU
```

> **The Principal's First Question**
>
> When given any performance number, first ask: *which regime?* A kernel running at 2% of peak compute on a memory-bound operation is running correctly at the memory-bandwidth ceiling. A kernel at 20% of peak on a compute-bound operation has 80% headroom and is a real problem. These require completely different interventions.

1.4 Roofline Analysis from First Principles

The Roofline Model provides a visual and mathematical framework for locating any operation relative to hardware limits. It requires two hardware numbers and one workload number.

```
ROOFLINE FORMULAS:

Arithmetic Intensity (AI) = FLOPs executed / Bytes read+written from HBM
                                            [units: FLOP/byte]

Memory ceiling:   P_memory(AI) = AI × HBM_bandwidth
Compute ceiling:  P_compute    = Peak_TFLOPS
Achievable perf:  P_max(AI)    = min(AI × HBM_BW,  Peak_TFLOPS)

Ridge Point (regime boundary):
  AI_ridge = Peak_TFLOPS / HBM_bandwidth

GPU RIDGE POINTS (BF16 dense) [SHIPPED]:
  V100 SXM2:  112 TFLOPS / 0.90 TB/s  = 124 FLOP/byte
  A100 SXM4:  312 TFLOPS / 2.00 TB/s  = 156 FLOP/byte
  H100 SXM5:  989 TFLOPS / 3.35 TB/s  = 295 FLOP/byte  ← current baseline
  H200 SXM5:  989 TFLOPS / 4.80 TB/s  = 206 FLOP/byte  (lower — BW-rich)
  B200 SXM:  2250 TFLOPS / 8.00 TB/s  = 281 FLOP/byte
  MI300X:    1307 TFLOPS / 5.30 TB/s  = 247 FLOP/byte

KEY INSIGHT: The ridge point doubled from V100 (124) to H100 (295).
Operations that were compute-bound on V100 may now be memory-bound on H100.
Re-evaluate every kernel's regime classification at each GPU generation.
```

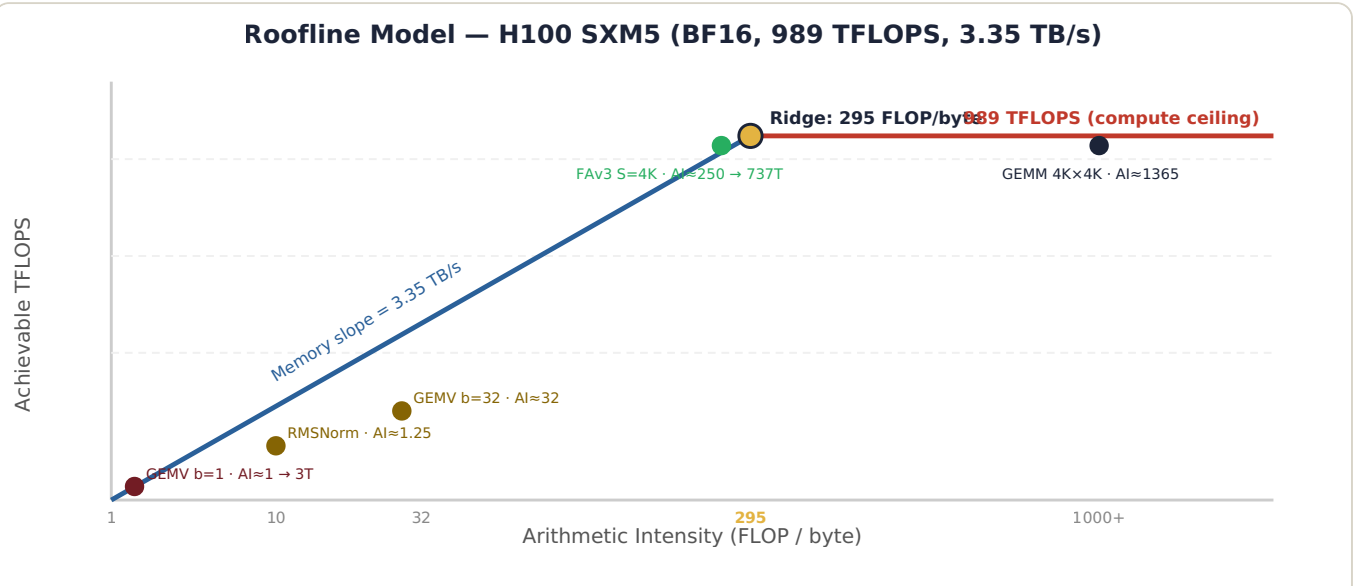


Figure 1.2 — Roofline Model for H100 SXM5 (BF16). Ridge point = 295 FLOP/byte. GEMV batch=1 sits at AI≈1 (3 TFLOPS effective — correct behavior, not a bug). FlashAttention v3 at S=4096 approaches the ridge (AI≈250, 737 TFLOPS, 74% MFU). Large GEMM prefill at AI≈1365 is compute-bound (≈800 TFLOPS, 80% MFU).

1.5 BERT FLOP Breakdown and the Bandwidth Paradox

A worked example grounds the roofline framework in a specific model.

```
BERT-LARGE FORWARD PASS ANALYSIS:
Config: L=24, H=1024, A=16 heads, d_head=64, S=512, BF16

Per-layer FLOPs:
  QKV projections:  3 × 2 × S × H²    = 3.22 GFLOP
  Attention QK^T:   2 × S² × H        = 0.54 GFLOP
  Attention × V:    2 × S² × H        = 0.54 GFLOP
  Output proj:      2 × S × H²        = 1.07 GFLOP
  FFN (4H hidden):  2 × 2 × S × H × 4H = 4.29 GFLOP
  ──────────────────────────────────────────────
  Per layer total:  ~9.66 GFLOP
  Full model (24 layers): 232 GFLOP

ARITHMETIC INTENSITY:
  Weights: ~340 MB BF16
  At batch=512, S=512: AI = 232 GFLOP / 340 MB ≈ 682 FLOP/byte → COMPUTE-BOUND ✓
  At batch=1, S=1 (decode step): AI = 0.45 GFLOP / 340 MB ≈ 1.3 FLOP/byte → MEMORY-BOUND ✗

THE BANDWIDTH PARADOX:
  Same model. Same hardware. Same weights.
  Compute-bound at batch=512. Memory-bound at batch=1.
  Batch size is the primary lever for controlling which regime you're in.
  This is why decode is expensive and prefill is cheap per token.
```

1.6 Hardware Efficiency: MFU and HFU

```
MODEL FLOP UTILIZATION (MFU):

  MFU = (FLOPs_per_sample × samples_per_second) / Peak_TFLOPS

  For LLaMA-3 70B (N=70B), H100 (989 TFLOPS), seq=4096, throughput=0.8 samples/s:
    FLOPs/sample ≈ 2 × 70×10⁹ × 4096 = 5.73 × 10¹¹
    MFU = (5.73×10¹¹ × 0.8) / 989×10¹² = 0.463 = 46.3%

MFU INTERPRETATION:
  MFU > 55%   Excellent — world-class cluster efficiency
  MFU 40–55%  Good — some optimization headroom
  MFU 25–40%  Mediocre — active investigation warranted
  MFU 10–25%  Poor — systematic bottleneck (comms? scheduling?)
  MFU < 10%   Critical — something is fundamentally broken

HFU (Hardware FLOP Utilization) for training with gradient checkpointing:
  HFU = MFU × (1 + recompute_fraction)
  If 33% of activations are recomputed: HFU = MFU × 1.33

PRODUCTION REFERENCE VALUES [REPRESENTATIVE]:
  LLaMA-3 405B training (H100, 4D parallel, Megatron-LM): 45–52% MFU
  LLaMA-3 70B prefill (TRT-LLM, FP8, batch=32):           50–65% MFU
  LLaMA-3 70B decode  (batch=1):                            3–8%  MFU  ← bandwidth floor
  LLaMA-3 70B decode  (batch=64):                          30–45% MFU
```

1.7 The Optimization Leverage Hierarchy

Not all optimizations have equal impact. A principal engineer knows which level to try first — and which to skip when a higher-leverage fix is available.

Level	Category	Typical Speedup	Example	When to Try
1	Algorithmic change	**10–1000×**	FlashAttention vs standard attention	Always first — zero quality loss available
2	Numerical format	**2–4×**	BF16 → FP8 (2× throughput on H100)	After algorithmic; quality budget allows
3	Operator fusion	1.5–3×	Fuse RMSNorm + GEMM + SiLU → single kernel	Memory-bandwidth-bound pipeline
4	Tiling and parallelism	1.1–2×	GEMM tile size for shape + GPU generation	Compute-bound but below roofline
5	Hardware configuration	1.05–1.3×	NCCL env vars, SM clock locking	After all above applied

Cost consequence of getting the hierarchy right:

A team spends 3 engineer-months on GEMM tile tuning and achieves +8% throughput. A one-week switch from standard attention to FlashAttention would have delivered +320% throughput. Opportunity cost: $720K in engineer time for 8% vs $30K for 320%. Know the hierarchy before opening a profiler.

1.8 Mental Models for Scale

```
SCALE-INVARIANT QUESTIONS (ask at every level):
  Single GPU:  "What fraction of peak TFLOPS am I achieving? Why not more?"
  Single node: "What fraction of NVLink bandwidth is consumed during comms?"
  Cluster:     "What is mean GPU utilization over a 24-hour rolling window?"

NEW BOTTLENECKS THAT EMERGE AT EACH SCALE THRESHOLD:
  4   GPUs:   TP AllReduce latency starts mattering (>10% wall time)
  8   GPUs:   NVSwitch topology determines TP group efficiency
  64  GPUs:   Pipeline bubble (1/PP_degree) becomes measurable overhead
  256 GPUs:   NCCL ring AllReduce latency limits gradient sync frequency
  1K  GPUs:   Straggler detection becomes a fleet health problem
  10K GPUs:   Silent data corruption (SDC) requires active monitoring
              (~1 SDC event per 2 weeks on a 10K H100 cluster)

INTERCONNECT NUMBERS TO KNOW COLD [SHIPPED]:
  NVLink 4.0 (H100):   900 GB/s bidirectional per-GPU
  InfiniBand NDR:       50 GB/s per port (400 Gbps)
  NVLink-C2C (GB200): 10,000 GB/s CPU-GPU (same superchip)
  PCIe Gen5 x16:        64 GB/s one-way GPU↔CPU
  Cross-node vs intra:  18× bandwidth difference — defines TP scope
```

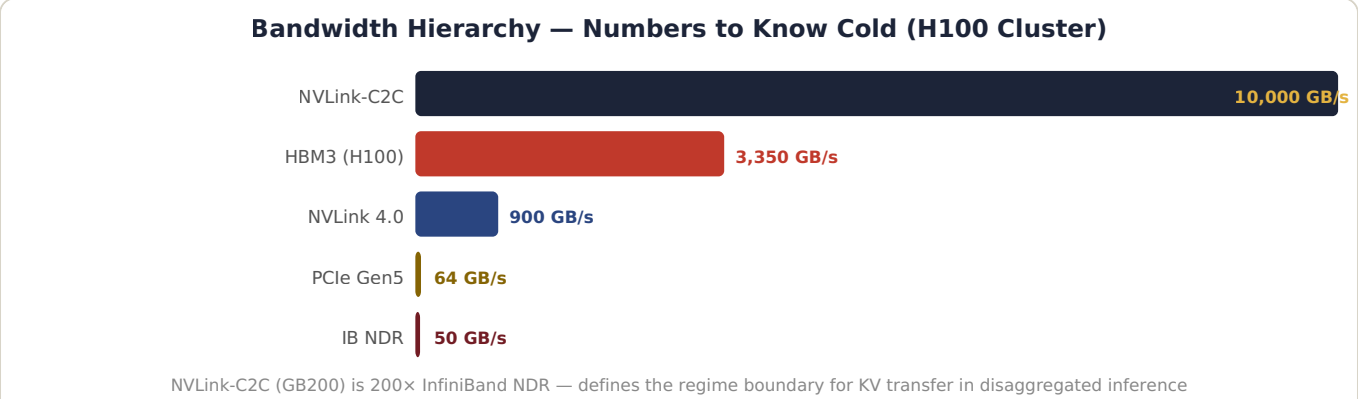


Figure 1.3 — Bandwidth hierarchy for H100 cluster. HBM3 at 3,350 GB/s is the primary compute bottleneck. NVLink (900 GB/s) enables intra-node tensor parallelism. InfiniBand NDR (50 GB/s) is suitable for pipeline and data parallelism but cannot support tensor parallelism (18× gap).

Chapter 1 Quick Reference

Concept	Formula / Value	Why It Matters
H100 ridge point (BF16)	**295 FLOP/byte**	Operations below this are memory-bound; above are compute-bound. The principal's first diagnostic number.
Arithmetic Intensity	AI = FLOPs / HBM_bytes	The single number that places an operation on the roofline and determines which fixes can help.
MFU	FLOPs_per_step × steps_s / Peak_TFLOPS	The cluster efficiency signal. Below 25% demands root-cause investigation. Above 55% is world-class.
TPOT minimum (BW-bound)	model_bytes / HBM_BW	70B BF16 on H100: 41.8ms/token — the physics floor no software can cross.
NVLink vs IB gap	900 GB/s vs 50 GB/s = 18×	Defines TP scope: TP must use NVLink. PP can cross IB. Any design violating this fails.
Optimization leverage order	Algorithm → Format → Fusion → Tiling → Config	Trying Level 4 before Level 1 is the most expensive performance engineering mistake.
SDC rate (10K H100s)	~1 event / 2 weeks	Silent data corruption is a real fleet operations concern at scale — requires active monitoring.

Chapter 1 Review Questions

0. **A kernel has arithmetic intensity of 50 FLOP/byte on H100 (ridge ≈ 295 FLOP/byte). Is it memory-bound or compute-bound? What is its theoretical peak performance in TFLOPS?**
   By how much would batch size need to increase (assuming AI scales linearly with batch) to push this operation above the ridge? At what batch size does the same kernel become compute-bound on H200 (ridge ≈ 206 FLOP/byte)?

0. **A training cluster at 1,024 H100 GPUs achieves 34% MFU on LLaMA-3 70B with TP=8, PP=8, DP=16. Theoretical maximum for this config is 54%. Walk through the seven-layer diagnostic process.**
   What metric at each layer indicates that layer is NOT the bottleneck? Which layers can you eliminate in under 30 seconds with a single Nsight Systems timeline screenshot?

0. **Explain the bandwidth paradox: why is the same model compute-bound at batch=512 and memory-bound at batch=1 on the same GPU?**
   What is the exact batch size crossover for a single linear layer with weight matrix W ∈ ℝ^(8192×8192) in BF16 on H100? Show the arithmetic intensity calculation for both batch sizes.

0. **Your inference team proposes switching from BF16 to FP8 to improve decode TPOT. The current TPOT is 41.8ms at batch=1 on H100 for a 70B model. Will FP8 help? By how much?**
   At what batch size would FP8 stop delivering proportional TPOT improvement? Derive the crossover using the ridge point formula.

0. **Design a diagnostic plan for a 10K-GPU H100 training cluster that has dropped from 50% to 32% MFU over the past week without any code changes.**
   List the five metrics you check first (with exact metric names from DCGM or Prometheus), the order you check them, and the root cause hypothesis each metric would confirm or eliminate.

0. **Rank the following optimizations by expected throughput improvement for LLaMA-3 70B decode at batch=1 on H100, and justify each ranking from first principles:**
   (a) Switch from standard attention to FlashAttention v3, (b) Switch from BF16 to INT4 weights, (c) Optimize GEMM tile size for H100 Tensor Cores, (d) Enable CUDA Graphs for the decode pipeline, (e) Switch from TP=2 to TP=1 (single GPU, model doesn't fit — use INT4 to fit).

Chapter Cross-References

Ch 2 Transformer architecture as a FLOP and memory problem — applying the roofline framework to the QKV projections, FFN, and attention operations that dominate every LLM workload.

Ch 3 / 3B GPU silicon architecture and generation roadmap — the hardware numbers (TFLOPS, HBM BW, ridge points) behind the roofline model derive from Ch 3's SM hierarchy, Tensor Core generations, and HBM specs.

Ch 4 Memory hierarchy and FlashAttention deep dive — Ch 4 extends the memory-bound regime analysis with the five-tier hierarchy (registers → SMEM → L2 → HBM → host DRAM) and derives the full FlashAttention memory traffic reduction.

Ch 12 Benchmarking and MFU measurement — the MFU formula introduced in §1.6 is operationalized in Ch 12 with exact measurement methodology, MLPerf context, and regression CI pipelines.

Ch 17 Observability and profiling — Nsight Systems, DCGM, and Prometheus metrics that surface each layer of the seven-layer stack are covered in Ch 17, with exact metric names and threshold values.