

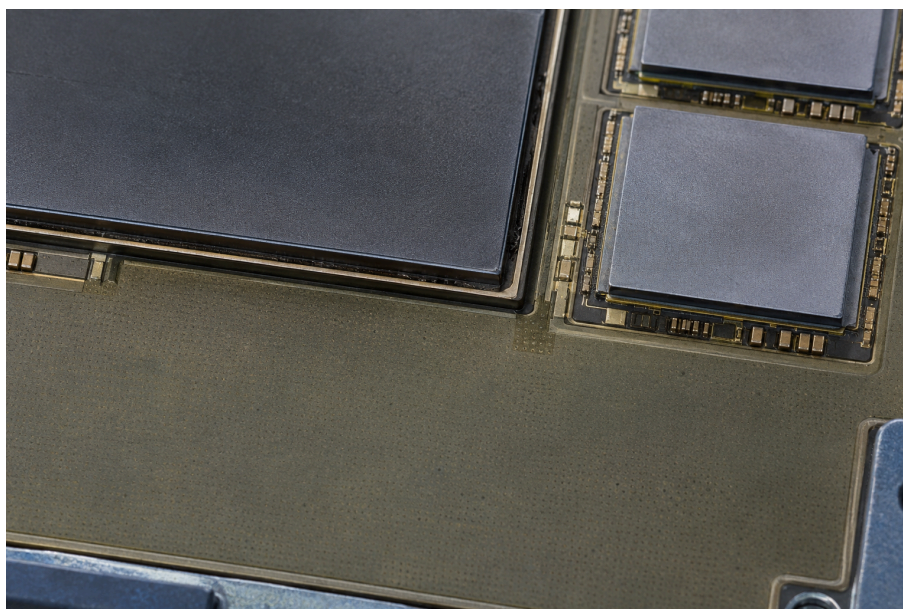


LLM Inference Roofline Model: FLOPS vs HBM Worksheet

Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-10-31 · llm inference roofline model · llm inference · arithmetic intensity · gpu memory bandwidth · hbm · prefill · decode · gpu procurement · roofline calculator

Original article: <https://gpusmith.com/articles/en/llm-inference-roofline-worksheet>



In this article

1. Executive Summary
 2. Introduction and Background
 3. Key Changes in How to Read an Upgrade Proposal
 4. The Buyer's Roofline Worksheet
 5. Calculate the Roofline Envelope
 6. Implementation Considerations and Process Changes
 7. Limits That Can Break the Simplified Estimate
 8. Hypothetical Shape Scenarios
 9. Data Analysis and Evidence
 10. Implications and Future Directions
 11. Conclusion
-

Executive Summary

An **LLM inference roofline model** screens whether a graphics processing unit (GPU) upgrade has enough arithmetic capacity and high-bandwidth memory (HBM) bandwidth to merit a workload-specific pilot. It places an upper bound on attainable arithmetic throughput, or equivalently a lower bound on execution time. It does not predict delivered tokens per second. The original framework counts operations against traffic reaching external memory after cache effects; an HBM worksheet must preserve that boundary.

(www2.eecs.berkeley.edu) (docs.nersc.gov)

The calculation needs **work F**, **HBM traffic Q**, compatible peak compute **C**, and peak bandwidth **BW**. Arithmetic intensity is F/Q in floating point operations per byte; the device ridge point is C/BW in the same units. The optimistic device time is $\max(F/C, Q/BW)$. Below the ridge, additional compute alone does not improve this bound. Above it, additional bandwidth alone does not improve it. These statements describe the simplified envelope, with the workload and byte count held fixed. (www2.eecs.berkeley.edu) (docs.nvidia.com) Low-batch decode and prompt prefill therefore require separate worksheets: decode processes a single token per active request per iteration, while prefill processes the prompt in parallel. (www.usenix.org)

Two **hypothetical examples**, calculated below rather than measured, illustrate the difference. A fictional device with **800 TFLOP/s** compatible dense compute and **3 TB/s** HBM bandwidth has a **266.67 FLOP/byte** ridge. A decode step doing **16 GFLOP** and moving **16.5 GB** has a **5.5 ms** device lower bound. A prefill doing **262.144 TFLOP** and moving **20 GB** has a **327.68 ms** lower bound. These are arithmetic consequences of explicitly assumed inputs, not hardware performance claims. Doubling the limiting resource halves the corresponding bound in these examples; real speedup still requires measurement. (www2.eecs.berkeley.edu)

Official specification labels matter as much as the arithmetic. **AMD MI350X** lists **2.3 PFLOPS** BF16 matrix performance separately from **4.6 PFLOPS** with structured sparsity; dense work must use a compatible ceiling. (www.amd.com) Kernel selection, key-value (KV) cache traffic, quantization metadata, host dispatch and inter-GPU communication can change the applicable model. (docs.pytorch.org) (docs.ray.io) The procurement outcome should be **benchmark this candidate**, **reject the stated claim**, or **need more data**. Require

workload-matched traces, an appropriate profiler roofline, and unprofiled throughput at the chosen service-level objective (SLO). MLCommons distinguishes aggregate output throughput from per-user speed, reinforcing why a device bound cannot establish an interactive service result. (docs.mlcommons.org)

Introduction and Background

Peak floating point operations per second (FLOPS) answer a narrow question: how much compatible arithmetic an accelerator can theoretically execute. A buyer needs a different answer: whether that resource constrains the specified inference workload. Roofline analysis connects the questions through arithmetic intensity, the amount of work performed for each byte moved at a defined memory boundary. Its performance envelope joins a bandwidth-dependent region to a compute ceiling. (docs.nersc.gov)

This report uses that connection as a **procurement screen**, not a product leaderboard or a new measured benchmark. GPU Smith's perspective here is that of an adjacent, independent engineering advisor. The reader's deliverable is a reproducible worksheet and a bounded decision about a pilot, rather than a recommendation based on advertised FLOPS alone.

Separate the workload into prompt processing and token generation before comparing devices. Sarathi-Serve describes prefill as parallel prompt processing and decode as successive single-token-per-request iterations. Its scheduling analysis also shows why mixing those phases matters to serving behavior. (www.usenix.org) A proposal that averages them into one tokens-per-second claim can obscure which phase the buyer actually needs to improve.

The analysis is anchored to specifications and documentation accessed on **October 31, 2026**; retrieval dates are recorded in the input ledger. These are source snapshots, not a claim that every document was issued on that date. No client results, commercial conversion rates or measured GPU speedups are asserted.

The scope is deliberately bounded to a device-level HBM roofline. Modal's explanation explicitly excludes system latency from the diagram. (modal.com) A research implementation of this style of optimistic bound, Quail, likewise acknowledges that implementations cannot satisfy all its assumptions. (fsdatalab.github.io) Accordingly, a promising lower bound opens an evidence request; it cannot close a purchasing decision.

Key Changes in How to Read an Upgrade Proposal

Replace a headline FLOPS comparison with a compatible ceiling

The numerator must describe the instructions the implementation actually uses. **BF16**, or bfloat16, **FP16**, or 16-bit floating point, and lower-precision formats require distinct entries. NVIDIA's profiling guidance specifically requires selecting the roofline section that matches the instruction data path. A tensor-core ceiling should not be assigned indiscriminately to every kernel in a model. (docs.nvidia.com)

Precision and sparsity are separate choices. AMD's MI350X page lists BF16 matrix throughput without structured sparsity separately from its structured-sparsity throughput. (www.amd.com) The worksheet should require evidence that the deployed representation and kernels qualify for the selected peak. A weight file advertised by bitwidth is insufficient evidence about the execution path.

The source ledger should preserve the exact wording around each compute figure:

- **Device identity:** SKU, form factor and number of accelerators represented.
- **Compute mode:** input datatype, matrix or vector path, and accumulation mode where documented.
- **Sparsity qualification:** dense value or the precise condition attached to a sparse value.
- **Memory scope:** HBM bandwidth per device rather than a node aggregate.
- **Operating configuration:** power policy, clocks and software versions used in the pilot.
- **Evidence state:** official specification, analytical assumption or measurement.

These are proposed purchasing controls. They turn an ambiguous comparison into an auditable input set. NERSC notes that practical execution conditions, including power constraints and programming models, affect the relevance of nominal peaks. (docs.nersc.gov)

Separate memory capacity from memory bandwidth

Capacity determines whether the required representation and working state fit; bandwidth constrains how quickly bytes can move. Google's TPU v5p documentation distinguishes nominal physical HBM from user-allocatable memory reported by tools. This is a useful warning about interpreting capacity labels even when the purchase under consideration is a GPU. (docs.cloud.google.com) A capacity increase may enable a different batching or placement policy, but that improvement must be modeled as a workload change.

HBM traffic is also different from allocated bytes. A retained weight or cache block can be allocated once but read repeatedly, while cache reuse can prevent some reads from reaching HBM. Triton's matrix multiplication tutorial explicitly changes block order to promote reuse. (triton-lang.org) Therefore, use an allocation worksheet for fit and a traffic worksheet for roofline analysis; reconcile them without treating them as interchangeable.

Replace one inference number with phase-specific objectives

Use **time to first token (TTFT)** for initial responsiveness and an explicitly defined subsequent-token latency measure for generation. MLCommons describes system throughput as total output tokens across concurrent users and separately considers per-user speed. (docs.mlcommons.org) Specify the buyer's percentile, request distribution and arrival process before interpreting either metric. These are pilot requirements chosen by the buyer, not parameters supplied by a GPU datasheet.

The Buyer's Roofline Worksheet

Establish units and the source ledger

Use decimal bandwidth consistently: **1 TB/s = 10^{12} bytes/s** and **1 TFLOP/s = 10^{12} FLOP/s**. (www.nist.gov) Binary memory units differ: **1 GiB = 2^{30} bytes**. (physics.nist.gov) These are unit definitions used in the calculations, not performance observations. Google's explicit use of GiB for TPU memory illustrates why retaining the source unit matters. (docs.cloud.google.com)

Table 1 defines the minimum input ledger. Its sensitivity entries are proposed checks rather than measured confidence intervals. Every row should be completed separately for the incumbent and candidate, and separately for prefill and decode.

Variable	Unit and meaning	Primary source or evidence	Retrieved / assumption flag	Sensitivity to evaluate
C	Compatible peak FLOP/s per device	Exact vendor datatype and sparsity row; examples in Table 3	2026-10-31 / specification	Eligible dense versus sparse ceiling
BW	Peak HBM bytes/s per device	Vendor memory-bandwidth row	2026-10-31 / specification	Peak bound versus separately labeled sustained measurements
P, b_w	Active weight count; stored bytes per weight	Pinned model artifact and packed representation; metadata matters (docs.pytorch.org)	Record artifact date / assumed until checked	Full resident weights versus active subset
B, S, O	Active batch; input and output tokens	Buyer request trace; distinguish prefill and decode (www.usenix.org)	Record trace interval / workload input	Representative, long-context and latency-critical shapes
F	Executed or analytically counted FLOP per step	Operation count at the chosen data path (docs.nvidia.com)	Record method / estimate until profiled	Weight-only work versus attention and other operations
Q_w	Weight bytes crossing HBM per step	HBM counters or explicit reuse assumption (docs.nersc.gov)	Record run / assumed until measured	One full read versus extra reads or cache filtering
Q_KV	KV bytes crossing HBM per step	Attention implementation and context lengths (arxiv.org)	Record run / assumed until measured	Context distribution and KV sharing
Q_other	Other HBM reads and writes	Activations, temporaries and output traffic accounting (docs.nersc.gov)	Record run / estimate	Fused versus materialized intermediates
L, H_KV, d_h, b_KV	Layers, KV heads, head dimension, bytes per cache element	Pinned model configuration; use KV rather than query-head count (huggingface.co)	Record revision / architecture input	Cache datatype, attention windows and partitioning
I, I_ridge	FLOP/byte	Derived as F/Q and C/BW (docs.nersc.gov)	Calculation / derived	Recompute whenever traffic or compute mode changes
t_bound	Seconds per modeled step	$\max(F/C, Q/BW)$ (docs.nersc.gov)	Calculation / theoretical lower bound	Phase and kernel scope; omitted work
TTFT, token latency	Seconds and declared percentiles	Unprofiled service test; throughput alone is insufficient (docs.mlcommons.org)	Record run / measured	Arrival rate, concurrency and output lengths

The ledger separates supplied facts from assumptions. A large unresolved range in Q should produce a range of lower bounds, not an artificially precise latency forecast. Preserve the original unit and a normalized byte value side by side in the working spreadsheet.

Calculate the Roofline Envelope

Let $Q = Q_w + Q_{KV} + Q_{other}$, counting both reads and writes that reach HBM. Count work and traffic over the same interval and device boundary. NERSC requires runtime, total operations and total moved bytes for workload characterization. (docs.nersc.gov) The following expressions are a spreadsheet-ready calculator; seconds result when the inputs use operations, bytes and their per-second rates. (docs.nersc.gov)

```
intensity_flop_per_byte = work_flop / hbm_bytes
ridge_flop_per_byte = peak_flop_per_second / peak_hbm_bytes_per_second
compute_lower_seconds = work_flop / peak_flop_per_second
memory_lower_seconds = hbm_bytes / peak_hbm_bytes_per_second
device_lower_seconds = MAX(compute_lower_seconds, memory_lower_seconds)
roof_flop_per_second = MIN(peak_flop_per_second,
                           intensity_flop_per_byte * peak_hbm_bytes_per_second)
```

The maximum expresses the optimistic envelope: execution cannot take less time than either resource requirement permits. It assumes their overlap when translating the roof into time. Do not add the two terms and call that the original roofline lower bound; equally, do not assume the maximum includes launch, queuing or communication latency. (www2.eecs.berkeley.edu) (modal.com)

For a dense weight-matrix approximation, let P be the weights participating in the modeled multiplication and T the tokens processed together. Counting a multiply plus an add as two operations gives $F_{weight} = 2PT$. If those weights cross HBM once, $Q_w = P b_w$, so $I_{weight} = 2T/b_w$. These are analytical assumptions for the examples below, not an exact count of a complete Transformer. They implement the operation-to-traffic ratio in the original model. (www2.eecs.berkeley.edu)

Preserve unrounded input cells and make the classification a formula, so reviewers can change assumptions directly.

For conventional single-token decode, T equals active batch B . For a simple prefill shape, T equals B times prompt length S . This explains the effect of weight reuse across tokens, but does not include attention's context-dependent work. The phase distinction is consistent with Sarathi-Serve's analysis. (www.usenix.org) If the weight-only estimate puts a workload below the ridge, include omitted traffic before drawing conclusions: additional bytes lower intensity further, whereas additional arithmetic can change the classification.

Implementation Considerations and Process Changes

Specify traffic before classifying prefill and decode

"Decode is memory bound" is a useful starting hypothesis for low active batches, not a universal label for every kernel or every serving configuration. The GQA research identifies the loading of weights and cached keys and values at each generation step as a bandwidth burden. (arxiv.org) Larger batches can amortize weight traffic,

but they also change cache requirements and the service's scheduling conditions.

Concurrency is not active batch size. The buyer should record how many requests actually contribute tokens to each modeled iteration. A request waiting in a queue does not supply useful matrix reuse. vLLM's chunked prefill can combine smaller prompt chunks with decode requests, so the observed execution shape may differ from the submitted prompt shape. (docs.vllm.ai)

Apply the worksheet in this order:

- **Choose the interval:** one decode iteration or one explicitly defined prefill chunk.
- **Count active work:** tokens actually scheduled, rather than requests merely connected.
- **Account for weights:** include packed data and associated scale information. (docs.pytorch.org)
- **Account for attention:** use current context lengths and KV-head structure. (arxiv.org)
- **Account for reuse:** distinguish HBM reads avoided by cache behavior from bytes still resident. (triton-lang.org)
- **Recalculate by shape:** keep the distribution of bounds rather than averaging away the latency-critical requests.

The sequence is an analytical procedure, not a claim that a framework exposes all those inputs automatically. Unobserved traffic remains a labeled assumption.

Distinguish KV capacity from KV traffic

For an uncompressed cache in a conventional full-attention layout, an allocation estimate is $2 \times L \times B \times S \times H_{KV} \times d_h \times b_{KV}$ bytes. The leading factor represents keys and values. This is bookkeeping for the stated layout, not a universal traffic formula. Grouped-query attention shares a key and value head across query groups, so query-head count is not a substitute for KV-head count. (arxiv.org)

A concrete architecture check comes from the official **Qwen2.5-7B-Instruct** card: it lists **28 query heads and 4 KV heads**. (huggingface.co) Substituting the query count for the KV count would inflate that factor by **7**, a derived ratio. The worksheet still needs the pinned layer count, head dimension, cache datatype, current sequence lengths and actual layout before calculating bytes. Do not infer them solely from the model's name.

Allocation does not establish reads per iteration. Hugging Face documents caches that grow with generation, attention variants that cap growth, and offloading that moves cache storage to CPU memory. (huggingface.co) Each changes the applicability of a simple full-context, HBM-only estimate. PagedAttention also allows noncontiguous KV storage, so implementation layout is part of the evidence request. (arxiv.org)

Handle caching and quantization as changed inputs

Prefix reuse and kernel-local reuse are different mechanisms. vLLM states that automatic prefix caching reduces repeated prefill work but does not reduce the time spent generating new tokens. (docs.vllm.ai) Measure the prefix-hit distribution and count only the prompt work executed under the tested policy.

Weight quantization changes the representation but does not establish a matching arithmetic ceiling. TorchAO describes low-precision tensors with extra metadata, including scales and packing formats. (docs.pytorch.org) Request the storage datatype, runtime operand datatype and selected kernel separately. FlashInfer's attention

API exposes a tensor-core option, further illustrating that an attention workload's arithmetic path is implementation-dependent. (docs.flashinfer.ai)

Limits That Can Break the Simplified Estimate

Cache hierarchy, kernel mix and overlap

An HBM roofline uses traffic crossing HBM. An L2-cache roofline uses a different boundary and byte count. NERSC describes hierarchical rooflines with different intensities at different cache levels. (docs.nersc.gov) Do not use an L2 traffic counter as Q beneath an HBM bandwidth roof.

Attention implementations can change Q substantially without changing the model's mathematical output. FlashAttention uses tiling to reduce memory reads and writes. (arxiv.org) This is why an optimization or software upgrade can move the workload on the diagram rather than simply raise a hardware ceiling. Triton likewise demonstrates reuse-sensitive block ordering. (triton-lang.org)

A whole-model point can hide alternating kernel bottlenecks. Request phase-separated and major-kernel evidence in addition to the aggregate. A compatible tensor roofline remains appropriate only for the data path it describes. (docs.nvidia.com) Treat overlapping transfers, fused operations and kernel boundaries consistently; adding lower bounds for operations that actually overlap can overstate even the analytical floor.

Host dispatch, interconnect and runtime configuration

An HBM-only model has no term for central processing unit (CPU) launch or scheduling overhead. PyTorch documents how Compute Unified Device Architecture (CUDA) graph replay removes layers of argument setup and dispatch, with graph-safety restrictions that commonly involve static shapes and control flow. (docs.pytorch.org) FlashInfer documents a corresponding implementation tradeoff: graph-compatible execution can constrain kernel dispatch choices across shapes. (docs.flashinfer.ai)

Communication belongs outside a single-device roof. Ray describes tensor parallelism as splitting weights across GPUs and pipeline parallelism as splitting layers. (docs.ray.io) Ray Serve also uses locality-aware routing to reduce network latency. (docs.ray.io) A multi-device buyer must therefore request topology and communication evidence, rather than summing HBM bandwidths and treating the result as one universally accessible memory system.

Check these exclusions before treating a bound as decision-useful:

- **Memory fit:** include runtime reservations and working state, not nominal weights alone. (docs.cloud.google.com)
- **Launch and scheduling:** establish whether the workload is device busy or host limited. (docs.pytorch.org)
- **Communication:** capture transfers and synchronization on the critical path. (docs.ray.io)
- **Power configuration:** preserve the tested operating conditions. (docs.nersc.gov)
- **Kernel eligibility:** verify datatype, sparsity and actual instruction path. (docs.nvidia.com)
- **Tail latency:** evaluate the declared service objective alongside aggregate throughput. (docs.mlcommons.org)

Profiling is evidence, with collection conditions attached

Nsight Compute can require more replay passes when more metrics are selected; its default cache flushing can also change the cache state seen by repeated kernels. Host timing during collection includes profiling overhead. These are documented collection effects, so a profiler run should explain its replay and cache-control settings. (docs.nvidia.com)

Keep **unprofiled service timing** separate from the diagnostic trace. JAX provides a clear example of another timing pitfall: asynchronous dispatch can return before device work completes. (docs.jax.dev) Its profiling documentation also distinguishes initial compilation and transfer costs. (docs.jax.dev) A purchasing comparison should define warm-up, completion synchronization and whether cold-start costs fall inside the service target.

Hypothetical Shape Scenarios

Low-batch decode (Hypothetical Example)

Both examples use a fictional accelerator and a deliberately simplified dense model. Assume **P = 8 billion participating weights**, **b_w = 2 bytes**, **C = 800 TFLOP/s** and **BW = 3 TB/s**. Their derived ridge is **266.67 FLOP/byte**. None of these figures identifies or measures a commercial GPU. The operations and traffic are explicit hypothetical inputs to the roofline equation. (docs.nersc.gov)

For decode, assume **B = 1**, one full weight read, and **0.5 GB** of additional HBM traffic. Then $F = 2PB = 16$ **GFLOP**, and $Q = 16.5$ **GB**. The resulting intensity is **0.970 FLOP/byte**. Compute requires at least **0.020 ms**, while memory movement requires at least **5.5 ms**. The larger term is the device lower bound. All values in this paragraph are derived from the stated assumptions. (docs.nersc.gov)

Doubling only compute leaves that **5.5 ms** bound unchanged. Doubling only bandwidth changes it to **2.75 ms**, assuming identical work and traffic. These are conditional model results, not expected speedups. The result is useful because it identifies the vendor evidence required: does the real execution move approximately the assumed bytes, and is device execution a material component of the service's elapsed time?

High-batch prefill (Hypothetical Example)

Now assume **B = 8** prompts of **2,048 tokens** processed together, giving **16,384 token positions**. Hold the same fictional weights and device constant. For the weight-matrix approximation, $F = 2PBS = 262.144$ **TFLOP**. Assume **20 GB** total HBM traffic, consisting of the weight read plus a hypothetical allowance for other traffic. This is a chosen traffic budget, not a measured attention implementation. (docs.nersc.gov)

Table 2 compares the derived quantities. The prefill excludes an explicit context-dependent attention operation count and is therefore a simplified weight-work envelope rather than a complete end-to-end prefill estimate.

Derived quantity	Low-batch decode (Hypothetical Example)	High-batch prefill (Hypothetical Example)
Work F	16 GFLOP	262.144 TFLOP
HBM traffic Q	16.5 GB	20 GB

Derived quantity	Low-batch decode (Hypothetical Example)	High-batch prefill (Hypothetical Example)
Arithmetic intensity	0.970 FLOP/byte	13,107.2 FLOP/byte
Compute lower time	0.020 ms	327.68 ms
Memory lower time	5.5 ms	6.667 ms
Device lower time	5.5 ms	327.68 ms
Double compute only	5.5 ms	163.84 ms
Double bandwidth only	2.75 ms	327.68 ms

Every cell is an analytical result from the hypothetical inputs and the same lower-bound formula.

(docs.nersc.gov) The examples show why a claim about FLOPS can be plausible for one shape and irrelevant to another. They do not prove that batching achieves the assumed reuse, that attention is negligible, or that a deployed system can attain peak rates.

For reproducibility, enter the raw counts into the calculator rather than rounded intensities. For sensitivity analysis, replace Q with the optimistic and pessimistic traffic assumptions and rerun both device terms. If a conclusion reverses across that interval, classify the purchase as **need more data**. This is a decision rule proposed by this report, not a statistical confidence claim.

Data Analysis and Evidence

Official specifications are inputs, not benchmark outcomes

Table 3 demonstrates source interpretation across selected accelerator specifications. It is not a ranking.

Figures were retrieved on **October 31, 2026**; an undated live page is not a guaranteed specification for every shipping configuration. The NVIDIA page retains a preliminary-specification qualification. (www.nvidia.com)

Example device	Official compute statement	Official HBM statement	Worksheet treatment
NVIDIA H200 SXM	1,979 TFLOPS BF16 tensor figure is sparsity-qualified (www.nvidia.com)	141 GB, 4.8 TB/s (www.nvidia.com)	Retain the sparse qualification; obtain a compatible dense ceiling before calculating a dense-work ridge.
AMD MI350X	2.3 PFLOPS BF16 matrix; 4.6 PFLOPS with structured sparsity (www.amd.com)	288 GB, 8 TB/s (www.amd.com)	Dense specification ratio: 2,300 TFLOP/s divided by 8 TB/s gives 287.5 FLOP/byte , derived from these inputs.
Intel Gaudi 3	1,678 TFLOPS BF16 matrix (cdrdv2-public.intel.com)	128 GB, 3.7 TB/s in the technical paper (cdrdv2-public.intel.com)	The displayed specification ratio is 453.51 FLOP/byte , rounded from 1,678/3.7. This is not observed utilization. (cdrdv2-public.intel.com)

The rows expose incompatible labels rather than establish a winner. A higher ridge means more work per byte is needed to reach that device's stated compute roof. It does not establish that the device is faster or slower on a given model. Vendor peaks also do not supply measured effective bandwidth for the buyer's access pattern.

Google's TPU v5p provides a unit cross-check outside the GPU rows: the documentation lists **459 TFLOPS** BF16 per chip and **2,765 GB/s** HBM bandwidth per chip. (docs.cloud.google.com) Normalizing to decimal bytes gives a derived ridge of **166.00 FLOP/byte**. This checks dimensional arithmetic; it is not a cross-platform inference comparison or a reason to substitute TPU software behavior into a GPU pilot.

For procurement, these ratios identify the traffic intensity needed to use a stated arithmetic ceiling. Ask the supplier to preserve the original specification row alongside each normalized value. If the proposed implementation uses another datatype or operating mode, replace that input before comparing bounds. A completed row should explain eligibility, rather than merely reproduce a larger number.

Implications and Future Directions

Request an evidence packet that can test the proposal

A useful vendor packet permits another engineer to reconstruct the inputs and compare like workloads. MLPerf Inference defines benchmarks by datasets and quality targets, as well as specified load-generation patterns. (mlcommons.org) Those principles are valuable even when the buyer's application does not match a published benchmark. SPEC ACCEL similarly emphasizes comparability and reproducibility, although its rules do not govern this proposed inference pilot. (spec.org)

Request the following artifacts before authorizing a performance-based upgrade:

- **Workload record:** prompt and output distributions, arrival process, active batches, concurrency, prefix-hit behavior and chosen SLO.
- **Model identity:** exact artifact revision, weights, quantization scales, adapters and acceptance criteria for output quality.
- **Hardware identity:** per-device SKU, memory configuration, device count, power policy and interconnect topology.
- **Software identity:** driver, runtime, serving engine, kernel selection and graph-capture configuration.
- **Roofline worksheet:** units, input provenance, traffic assumptions, omitted work and incumbent/candidate bounds.
- **Diagnostic evidence:** representative phase traces and compatible kernel rooflines, with collection settings disclosed. (docs.nvidia.com)
- **Service evidence:** unprofiled TTFT and subsequent-token latency distributions, accepted-request throughput and failure accounting. Aggregate and per-user rates should be distinguished. (docs.mlcommons.org)
- **Reproducibility:** disclose sufficient configuration and artifact details for another engineer to repeat the comparison.
- **Economic evidence:** total proposed costs and capacity at the same quality and latency target, with the accounting boundary stated.

Choose the gate outcome explicitly

Use three outcomes rather than presenting a single estimated tokens-per-second number:

- **Benchmark this candidate:** the relevant bound improves, workload fit is credible, and the missing implementation evidence can be obtained in a controlled pilot.
- **Reject the stated claim:** the proposal substitutes an incompatible compute mode, mixes device and system units, or treats a device lower bound as demonstrated service performance.
- **Need more data:** traffic, active batch, cache policy, communication or latency targets are too uncertain to determine whether the proposed resource matters.

A lower bound alone cannot establish a candidate's advantage: equal-looking inputs do not establish equal utilization. Compare measured service capacity at matched quality and latency. Quail acknowledges that its ideal assumptions cannot all be satisfied. (fsdatalab.github.io)

Measure cost after qualifying performance

Evaluate cost per accepted output token or completed request only after defining the latency and quality gates. Include the buyer's relevant acquisition or rental costs, power and operating assumptions, and the useful capacity actually delivered. No universal price or amortization period is assumed here.

Report cold-start and warmed operation separately when both affect the application. JAX documents first-execution compilation and CPU-to-accelerator transfers as relevant timing costs. (docs.jax.dev) For distributed deployments, include routing and communication within the service boundary rather than attributing every improvement to GPU arithmetic. (docs.ray.io)

Conclusion

An LLM inference roofline worksheet turns a GPU upgrade proposal into a testable resource argument. Its value comes from matching arithmetic, bytes and precision to a specific phase and device boundary. A compute peak becomes meaningful only after the workload's intensity is compared with the corresponding ridge.

The buyer should keep three distinctions visible: capacity versus traffic, theoretical lower time versus observed service latency, and a resource improvement versus a validated speedup. A low-batch decode workload may provide little reuse of weights; a large prefill can provide much more. Neither label removes the need to account for attention, cache policy, scheduling and communication.

The practical deliverable is a source ledger, a range of phase-specific bounds and an evidence packet. Complete those inputs for the incumbent and candidate, label assumptions, and preserve the calculations so another engineer can reproduce them. Where a proposal changes quantization, batching or placement, recalculate the workload rather than carrying forward an old intensity.

The final purchasing gate remains empirical: compare qualified service capacity and cost under the same quality and latency requirements. Advance a candidate when the analytical opportunity and available evidence justify that test. Reject an unsupported performance claim when its inputs are incompatible. Request more data when the resource boundary is unresolved. That discipline makes the roofline a useful screen without asking it to promise a speedup it cannot establish.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-134.pdf>
2. <https://docs.nersc.gov/tools/performance/roofline>
3. <https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html>
4. <https://www.usenix.org/conference/osdi24/presentation/agrawal>
5. <https://www.amd.com/en/products/accelerators/instinct/mi350/mi350x.html>
6. https://docs.pytorch.org/ao/stable/contributing/quantization_overview.html
7. <https://docs.ray.io/en/latest/serve/llm/user-guides/cross-node-parallelism.html>
8. <https://docs.mlcommons.org/endpoints-submission-docs/understand/what-is-mlperf-endpoints>
9. <https://modal.com/gpu-glossary/perf/roofline-model>
10. <https://fsdatalab.github.io/blog/introducing-quail>
11. <https://docs.cloud.google.com/tpu/docs/v5p>
12. <https://triton-lang.org/main/getting-started/tutorials/03-matrix-multiplication.html>
13. <https://www.nist.gov/pml/owm/metric-si-prefixes>
14. <https://physics.nist.gov/cuu/Units/binary.html>
15. <https://arxiv.org/html/2305.13245v3>
16. <https://huggingface.co/Qwen/Qwen2.5-7B-Instruct>
17. <https://docs.vllm.ai/en/stable/configuration/optimization>
18. https://huggingface.co/docs/transformers/en/kv_cache
19. <https://arxiv.org/html/2309.06180v1>
20. https://docs.vllm.ai/en/stable/features/automatic_prefix_caching
21. <https://docs.flashinfer.ai/api/attention.html>
22. <https://arxiv.org/abs/2205.14135>
23. <https://docs.pytorch.org/docs/main/notes/cuda.html>
24. <https://docs.ray.io/en/latest/serve/advanced-guides/performance.html>
25. https://docs.jax.dev/en/latest/async_dispatch.html
26. <https://docs.jax.dev/en/latest/201/profiling.html>
27. <https://www.nvidia.com/en-us/data-center/h200>
28. <https://cdrdv2-public.intel.com/817486/gaudi-3-ai-accelerator-white-paper.pdf>
29. <https://mlcommons.org/benchmarks/inference-datacenter>
30. <https://spec.org/accel/docs/runrules.html>

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the [hardware library](#), [GPU references](#), [networking references](#) and [vendor research](#). [Contact GPU Smith](#) with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: <https://gpusmith.com>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.