



FP16 vs FP8 vs INT4 Inference Cost: Quality and GPU Fit

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 · 2026-10-15 · fp16 vs fp8 vs int4 inference cost · fp8 inference · int4 quantization · bf16 · gpu memory · llm inference cost · quality adjusted cost · vllm · tensorrt llm

Original article: <https://gpusmith.com/articles/en/fp16-fp8-int4-gpu-inference-costs>



In this article

1. Executive Summary
 2. Introduction and Background
 3. FP16 and BF16: The Reference Configuration
 4. FP8: Separate Native W8A8 From Weight-Only Paths
 5. INT4 and FP4: Approve an Exact Recipe
 6. Feature Comparison
 7. Performance and Benchmarks
 8. Data Analysis and Evidence
 9. Implications and Future Directions
 10. Conclusion
-

Executive Summary

FP16 vs FP8 vs INT4 inference cost has no universal winner. The procurement choice is the lowest total cost among configurations that satisfy the workload's declared quality floor, latency limits, and capacity requirement. **FP8 W8A8** changes both weights and activations; **INT4 W4A16** changes weights while retaining higher precision activations. Weight precision, activation precision, and key-value cache precision must therefore be recorded separately. TensorRT-LLM and vLLM document materially different execution paths behind these labels. (nvidia.github.io) (docs.vllm.ai)

A useful initial screen is weight payload. For the explicitly hypothetical **70-billion-parameter** example developed below, the calculated payload is **130.39 GiB at 16 bits, 65.19 GiB at 8 bits, and 32.60 GiB at 4 bits**, before scales, retained tensors, cache, and runtime allocations. The formula is parameters multiplied by bits divided by eight; quantization metadata adds memory because groups have their own parameters. Actual allocator reservation is a separate measurement. (huggingface.co) (docs.pytorch.org) With the example's stated reserves and supported shard choices, the memory screen selects **4, 2, and 1 GPUs**, respectively. These are illustrative fit results, not measurements or certified model configurations.

Published results also resist a universal speed ranking. An ACL study covering the Llama-3.1 family and more than **500,000 evaluations** found strong FP8 quality preservation within its tests. Its asynchronous code-completion experiment on **four H100s** reached **3.5 queries/second with BF16, 6.9 with FP8, and 5.9 with INT4**, under its specific engine and workload. That evidence supports testing FP8 against INT4 rather than assuming fewer bits always means higher throughput. (aclanthology.org)

The report's **quality-adjusted cost** divides the complete period cost by output tokens belonging to responses that pass both the agreed evaluation and service criteria. Its hypothetical worksheet yields **\$8.50, \$4.77, and \$3.86 per million accepted output tokens**, but the last option fails the example's quality gate. These are calculated scenarios, not advertised rates. As an external price reference, Lambda's retrieved single-H100 listing was **\$4.29 per GPU-hour**; its billing documentation also makes idle running time chargeable. (lambda.ai) (docs.lambda.ai) The practical recommendation is to retain FP16/BF16 as the reference, test supported FP8 next, and approve a specific INT4 or FP4 recipe only after quality, cache behavior, actual memory, and service-level objectives have passed. (nvidia.github.io) (docs.cloud.google.com)

Introduction and Background

Private inference teams buy complete serving capacity. They must provide enough graphics processing units (**GPUs**), host resources, and operational headroom to handle a workload at an acceptable cost. Choosing precision becomes an engineering decision about which complete configuration meets that obligation. A compressed checkpoint is only one input: framework memory management and serving queues remain relevant after the weights have loaded. (docs.pytorch.org) (docs.ray.io)

The comparison must begin with exact recipes. **W8A8** denotes eight-bit weights and eight-bit activations; **W4A16** denotes four-bit weights and sixteen-bit activations. The numerical datatype still matters. Model Optimizer, for example, defines a W4A8 combination with integer weights and FP8 activations, while bitsandbytes documents a separate NormalFloat datatype. Neither should be relabeled as a generic INT4 system. (github.com) (huggingface.co)

This report takes the perspective of GPU Smith as an **adjacent independent engineering advisor**. Its decision framework concerns procurement, workload sizing, and serving acceptance; the publisher is not a precision format or a competing GPU option. The comparison uses independent documentation and original research, with no publisher pages used as evidence.

The documentation and price observations were retrieved for the **October 15, 2026** publication snapshot. “Latest,” “main,” and “master” documentation can change; an implementation must record the actual release and image digest it uses. Docker documents digest-based pulls as a way to select an exact image version. (docs.docker.com) No local GPU benchmark was performed for this report. Published measurements retain their original scope, and all worksheet inputs are explicitly hypothetical. No checkpoint hash, conversion time, measured power, or customer result is invented.

The central question is consequently precise: does a candidate recipe remove enough physical capacity or deliver enough additional acceptable work to reduce the complete bill? A memory estimate screens candidates. A workload-specific acceptance test decides which candidates qualify.

FP16 and BF16: The Reference Configuration

Capabilities

FP16 is half-precision floating point; **BF16**, or bfloat16, is another sixteen-bit floating-point format. They should be recorded as distinct datatypes even when their weight payload has the same size. TensorRT-LLM documents support for both where corresponding checkpoints are available. This report uses FP16/BF16 as a category for the reference experiment, not as a claim that the two formats produce interchangeable outputs. (nvidia.github.io)

The reference establishes task quality and serving behavior before additional compression. Its actual checkpoint, tokenizer, generation settings, and output requirements define what each candidate must preserve. This is an acceptance design choice rather than an assertion of exact token equivalence: deterministic checks can evaluate structured fields, while human assessments can evaluate less mechanical requirements. MLflow documents both programmatic and human feedback mechanisms. (www.mlflow.org) (mlflow.org)

Adoption

For an enterprise approval process, begin by freezing the reference configuration:

- **Identity:** Record the original checkpoint revision, file hashes, tokenizer revision, and applicable license.
- **Execution:** Pin the container digest, engine build, driver, GPU SKU, and parallelism settings. (docs.docker.com)
- **Behavior:** Freeze prompts, retrieval inputs, sampling parameters, output limits, and stop conditions.
- **Accounting:** Record allocated and reserved memory separately before comparing compression candidates. (docs.pytorch.org)

These are recommended evidence fields, not details of a benchmark conducted here. If the reference itself misses the workload's service objective, it remains a quality comparator but cannot be treated as deployable capacity.

Strengths and Limitations

The reference avoids the need to select an additional weight quantizer and its calibration procedure. It also supplies the paired outputs against which lower precision can be evaluated. Post-training quantization (**PTQ**) is an additional transformation performed after model training; quantization-aware training (**QAT**) instead introduces simulated quantization during training. A QAT checkpoint is therefore a different preparation path and should be identified explicitly. (huggingface.co)

The economic limitation is weight payload. Retaining a higher precision may require more shards or leave less room for active requests. Yet a larger payload does not by itself establish higher cost per accepted token: that conclusion still requires measured service performance and the complete capacity bill. Ray's benchmark guidance emphasizes that performance depends on the workload and hardware stack. (docs.ray.io)

FP8: Separate Native W8A8 From Weight-Only Paths

Capabilities

FP8 inference can describe several combinations of scaling and execution. The current vLLM guide distinguishes native W8A8 computation on NVIDIA devices with **compute capability at least 8.9** from weight-only FP8 Marlin execution on older supported devices. Its documented weight-only floor is **7.5**. Loading FP8 weights on such a path does not establish FP8 activation computation. (docs.vllm.ai)

Granularity also changes the recipe. The guide's dynamic example uses static per-channel weight quantization and dynamic per-token activation quantization. TensorRT-LLM separately documents block-scaling kernels with different scale representations across GPU architectures. A result for one of these schemes cannot establish the performance or accuracy of another. (docs.vllm.ai) (nvidia.github.io)

Adoption

Treat FP8 as a candidate with a complete execution specification:

- **Recipe:** State W8A8 or W8A16, format, scale granularity, and static or dynamic activation scaling. (github.com)
- **Preparation:** Record whether representative calibration was used; the documented dynamic round-to-nearest path does not require a calibration dataset. (docs.vllm.ai)
- **Exclusions:** Export the layers retained at higher precision; the guide's example excludes the output head. (docs.vllm.ai)
- **Cache:** Choose the key-value (**KV**) cache datatype independently and evaluate that choice separately. (arxiv.org)

The KV cache stores attention state across generation. Changing weight precision does not define its representation. Original cache research identifies longer contexts and larger batches as separate memory pressures. (arxiv.org)

Strengths and Limitations

FP8 is a reasonable early compression candidate when the intended hardware has a supported native path. NVIDIA documents FP8 Tensor Core capability on H100, and AMD documents FP8 capability on MI300X. Those capabilities establish an opportunity for suitable software, not cross-platform equivalence or a measured application speedup. (www.nvidia.com) (www.amd.com)

The strongest quality evidence must still be scoped. The ACL comparison characterized FP8 as lossless across its tested model scales and evaluation suite; this is an empirical finding within those tests. It does not certify private domain data or a different cache configuration. (aclanthology.org)

Current cache documentation adds another qualification: with Flash Attention 3, vLLM's FP8-cache path can quantize queries as well as keys and values. The cache switch can therefore alter attention computation, not just storage. Evaluate it as a separate candidate rather than attributing its results to weight quantization. (docs.vllm.ai)

INT4 and FP4: Approve an Exact Recipe

Capabilities

INT4 W4A16 is weight-only integer quantization with higher precision activations. TensorRT-LLM describes dequantization inside the linear-layer path; its GPTQ and activation-aware weight quantization (**AWQ**) options use grouped parameters. These recipes should not be described as uniformly four-bit arithmetic. (nvidia.github.io)

GPTQ uses approximate second-order information for weight quantization. **AWQ** uses activation information to identify salient weight channels and does not require backpropagation or reconstruction. Those different preparation procedures help explain why a bit-width label cannot stand in for a recipe or a calibration record. (arxiv.org)

W4A8 adds another activation choice. Model Optimizer's documented variant combines block-wise INT4 weights with per-tensor FP8 activations and AWQ. **NVFP4 W4A4**, meanwhile, uses floating-point weights and activations with a different scaling scheme. Neither is interchangeable with INT4 W4A16. (github.com)

(docs.vllm.ai)

Adoption

Approval should capture preparation and runtime evidence:

- **Calibration:** Archive dataset identifiers, sampling procedure, sequence lengths, and quantizer settings.
- **Parameters:** Record group size, scale dtype, zero points, clipping, and excluded modules. (docs.vllm.ai)
- **Conversion cost:** Measure quantization wall time and compute expense, including unsuccessful attempts.
- **Executed kernels:** Verify that the intended activation precision actually ran. The documented NVFP4 path falls back to weight-only operation below **SM100**. (docs.vllm.ai)

Quantizer settings are not universal recommendations. The vLLM INT4 guide recommends deployment-representative calibration and gives a starting recipe; the original ACL experiment instead used its own GPTQ configuration and calibration corpus. (docs.vllm.ai) (aclanthology.org)

Strengths and Limitations

INT4's payload reduction can create a valuable physical fit boundary, especially where the reference requires multiple devices. Its acceptance risk must be measured against the actual task set. A long-context benchmark or constrained-output task may expose changes absent from an average score.

FP4 requires additional care with layer sensitivity. An original study of MXFP4 and NVFP4 on Qwen2.5 models found multilayer perceptron projection components especially sensitive in its component-isolation tests. That finding motivates selective higher precision and retesting; it does not establish an end-to-end throughput result. (arxiv.org)

Finally, **NF4** and bitsandbytes Float4 are distinct from integer INT4 and hardware-specific NVFP4. The bitsandbytes compute dtype is configurable independently of weight storage. Its documentation is evidence for the library's mechanisms, not proof of zero degradation on a private inference task. (huggingface.co)

Feature Comparison

Table 1 is a precision compatibility crosswalk. It combines current documentation rather than promising that every listed model, backend, and architecture combination works. The engine's specific method matrix and a startup test remain the approval boundary. (nvidia.github.io)

Exact candidate	Weights / activations / KV cache	Documented execution boundary	Preparation and approval requirement
FP16 or BF16 reference	16 / 16 / separately selected	Corresponding checkpoint and engine support required. (nvidia.github.io)	Freeze the exact reference and output acceptance rules.
FP8 W8A8, dynamic example	FP8 / FP8 / separately selected	Native NVIDIA computation at capability ≥ 8.9 in the cited vLLM guide. (docs.vllm.ai)	Dynamic per-token activation recipe; record excluded layers. (docs.vllm.ai)

Exact candidate	Weights / activations / KV cache	Documented execution boundary	Preparation and approval requirement
FP8 W8A16 Marlin	FP8 / 16-bit / separately selected	Documented weight-only path at capability ≥ 7.5 ; this is a different execution recipe. (docs.vllm.ai)	Compare independently with native W8A8.
INT4 W4A16 AWQ or GPTQ	INT4 / FP16 or BF16 / separately selected	Support differs by quantizer and backend. ModelOpt's Linux AWQ entry names Ampere and later. (nvidia.github.io) (github.com)	Representative calibration and exact grouped-weight settings.
INT4-FP8 W4A8 AWQ	INT4 / FP8 / separately selected	ModelOpt lists Ada and later; some model-format entries are experimental. (github.com)	Check the downstream engine matrix for the actual GPU.
NVFP4 W4A4	FP4 / FP4 / separately selected	The cited vLLM recipe changes to weight-only execution below SM100. (docs.vllm.ai)	Validate scale calibration, sensitive layers, and actual kernels.
FP8 KV-cache variant	Keep parent weight/activation recipe; change cache independently	Attention backend can change the executed math. (docs.vllm.ai)	Calibrate scales where required and rerun long-context evaluation.

The crosswalk exposes a documentation discrepancy worth resolving during implementation. The vLLM INT4 tutorial literally states **compute capability greater than 8.0** while also naming Ampere; its broader compatibility table lists implementations separately. This report does not silently rewrite that inequality. Use the exact model, kernel, and release test to settle support rather than applying the tutorial's sentence to every INT4 implementation. (docs.vllm.ai)

Weight Payload, Cache, and Runtime Reservation

The **weight-memory preflight** is:

```
payload_bytes = quantized_parameter_count * weight_bits / 8
packed_weights = payload_bytes + scales_bytes + zero_points_bytes
                + padding_bytes + retained_tensor_bytes
```

For this mixed-precision estimate, define `quantized_parameter_count` to include only parameters stored at `weight_bits`, excluding all retained tensors. Define `retained_tensor_bytes` as the actual storage bytes of those excluded tensors, counted once. Express every metadata term in bytes. Such exclusions matter: the [LLM Compressor FAQ](#) says linear layers are typically quantized except for `lm_head`. The later hypothetical worksheet separately assumes a uniform weight bit width across all 70 billion parameters.

Per-group or per-channel quantization adds parameters beyond the packed payload. Mixed paths also retain other dtypes; bitsandbytes, for example, processes outliers separately through sixteen-bit matrix multiplication. (huggingface.co)

For a conventional decoder with equally sized key and value caches, a useful dimensional estimate is:

```
KV_bytes = 2 * layers * KV_heads * head_dimension
           * total_cached_tokens * bytes_per_cache_element
```

This is an author-specified shape calculation, not an engine allocation guarantee. It assumes those tensors are stored in full with the stated shape. Use KV heads rather than query heads, count simultaneously resident tokens across requests, and account separately for prefix sharing, cache blocks, metadata, and replication. Architectures with different state representations require a different formula. Cache research supports treating this memory separately from weight payload. (arxiv.org)

Actual high-bandwidth memory (**HBM**) usage also includes workspaces, graph pools, communication buffers, and framework reservation. PyTorch distinguishes tensor allocations from memory managed by its caching allocator; its counters do not establish complete device usage. Record engine/cache reservation and device-level occupancy as well as framework peaks. (docs.pytorch.org)

Performance and Benchmarks

What Published Evidence Establishes

The ACL experiment provides a useful counterexample to a single bit-width ranking. For asynchronous Llama-3.1-Instruct-70B code completion, with **256 prefill tokens and 1,024 decode tokens**, vLLM **0.6.4.post1** on **four H100s** delivered **3.5 queries/second in BF16, 6.9 in FP8, and 5.9 in INT4**. FP8 led that particular throughput test. (aclanthology.org)

For synchronous Llama-3.1-Instruct-8B code completion on **one A6000**, the same reported engine and token lengths produced **24.5 seconds with BF16 and 9.7 seconds with INT4**. These experiments differ in model size, GPU, and load mode. They cannot be combined into a hardware-independent speedup or a current per-token price. (aclanthology.org)

The paper's RULER long-context results also differed by precision: for its **8B** model, BF16 and FP8 scored **82.8**, while INT4 scored **81.1**. Those scores support a separate long-context acceptance check; they do not specify the performance of an enterprise document workflow. (aclanthology.org)

A Workload-Specific Quality Acceptance Set

Approve the evaluation contract before examining the candidate winner. MLflow supports a mixture of code checks, automated judges, and human feedback. Google's judge guidance requires comparison with human-rated ground truth. These mechanisms make the following proposed rubric implementable, but they do not choose the business threshold. (www.mlflow.org) (docs.cloud.google.com)

- **Population:** Stratify real prompts by task, language, input length, output length, and business importance.
- **Separation:** Keep calibration examples separate from held-out acceptance prompts.
- **Correctness:** Score required facts, calculations, citations, and tool decisions against task-specific references.
- **Structure:** Use deterministic checks for schemas, required fields, and allowed outputs. (mlflow.org)

- **Equivalence:** Declare whether exact strings, equivalent structured values, or human-rated meaning is required.
- **Review:** Blind reviewers to precision labels, randomize paired outputs, and adjudicate disagreements.
- **Judge alignment:** Compare automated judgments with human decisions and inspect disagreements. (docs.cloud.google.com)
- **Uncertainty:** Publish sample sizes and confidence intervals, including results for critical subgroups.

An observed pass rate is not its lower confidence bound. NIST documents exact binomial confidence methods, including situations where simple symmetric approximations are inadequate. Use those methods only when their independence assumptions fit; repeated users or clustered tasks require an appropriate paired or clustered analysis. (www.itl.nist.gov)

Serving, Latency, Errors, and Energy Protocol

Measure **time to first token (TTFT)** from request arrival, including queuing. Measure **time per output token (TPOT)** during decode, and separately record individual **inter-token latency (ITL)** gaps. Ray defines TPOT as an average; a percentile of those averages is mathematically different from a percentile of individual gaps. (docs.ray.io)

Latency limits must be workload-specific. MLPerf Inference v5.0's Llama-3.1-405B server benchmark sets **99th-percentile TTFT at 6 seconds** and **TPOT at 175 milliseconds**. Its separate Llama-2-70B Interactive benchmark uses **450 milliseconds** and **40 milliseconds**, respectively. Those are benchmark-specific constraints, not default enterprise targets. (mlcommons.org)

A reproducible load test should capture:

- **Parity:** Keep checkpoint revision, prompts, engine release, hardware, and generation settings fixed where supported.
- **Load:** Sweep concurrency and offered request rate using the intended arrival distribution.
- **Phases:** Report cold start, warm steady state, and overload recovery separately.
- **Latency:** Report median and tail TTFT, TPOT, and the declared ITL aggregation.
- **Throughput:** Report input tokens, generated output tokens, completed requests, and acceptable work separately.
- **Errors:** Count timeouts, cancellations, truncation, invalid output, and memory failures in the request ledger.
- **Cache:** Record dtype, scaling, prefix state, occupancy, and the actual context distribution. (arxiv.org)
- **Power:** Measure whole-server electrical input over the same period and include idle capacity. SPEC documents analyzer-based power and Active Idle measurement. (www.spec.org)
- **Repetition:** Repeat runs until the declared uncertainty criterion is satisfied, preserving per-run aggregates.
- **Publication:** Release hashes, calibration procedure, evaluation artifacts, commands, and raw aggregates subject to data-access constraints. (docs.docker.com)

Data Analysis and Evidence

Hardware and Price Inputs Need Their Own Units

NVIDIA's reference architecture lists **80 GB** of HBM for H100 SXM, **141 GB** for H200 SXM, and **180 GB** for B200 SXM. AMD's MI300X announcement specifies **192 GB**. These are vendor capacity specifications, not a promise that all advertised bytes are available to model tensors. The worksheet below instead uses an explicitly hypothetical usable budget in GiB. (docs.nvidia.com) (ir.amd.com)

Power ceilings must also stay separate from measured energy. NVIDIA lists H100 SXM at up to **700 W** configurable thermal design power and describes B200 configurations up to **1 kW per GPU**. Those specifications cannot substitute for measured whole-server power or joules per accepted token. (www.nvidia.com) (docs.nvidia.com) (www.spec.org)

As retrieved on the publication date, Lambda listed a single-H100 SXM instance at **\$4.29 per GPU-hour**, versus **\$3.99 per GPU-hour** for its eight-GPU listing, before applicable taxes. AWS updates Capacity Blocks reservation prices regularly based on supply and demand; obtain a dated reservation quote before using AWS pricing in procurement. These are different rental products. Neither is a measured private-server ownership cost. (lambda.ai) (aws.amazon.com) Lambda also documents minute-increment on-demand billing and charges for running idle instances, reinforcing the need to price calendar capacity rather than only busy generation time. (docs.lambda.ai)

Whole-GPU Fit (Hypothetical Example)

Table 2 shows author calculations, not observed deployments. Assume a generic **70-billion-parameter** model, hypothetical metadata equal to **2% of payload**, **12 GiB** of aggregate KV allocation, **8 GiB of runtime reserve per GPU**, and **72 GiB usable per GPU**. Only shard counts **1, 2, 4, and 8** are permitted in this example. Scales and reserves must be replaced with measurements before procurement; granularity is a documented source of additional metadata. (huggingface.co)

Hypothetical candidate	Payload GiB	Payload plus assumed metadata GiB	Minimum permitted GPUs at 12 GiB cache	Per-GPU modeled footprint GiB	GPUs at 64 GiB cache
16-bit weights	130.39	132.99	4	44.25	4
8-bit weights	65.19	66.50	2	47.25	4
4-bit weights	32.60	33.25	1	53.25	2

The arithmetic divides packed weights plus aggregate cache evenly across shards, then adds the per-device reserve. That simplification may fail for replicated state, uneven layers, or unsupported parallelism. The table shows why integer counts matter: a continuous byte reduction becomes an economic opportunity only after a deployable topology is found. Increasing cache to the stated **64 GiB** changes the lower-precision counts even though weight payload is unchanged. Assume the initial cache uses **BF16**. An idealized **FP8** payload at the same token count halves the example's cache to **6 GiB**, before scale and allocation overhead, and leaves its GPU counts unchanged. These are direct sensitivity calculations under the same assumptions, not observed FP8-cache savings. The actual attention backend and calibration still require testing. (docs.vllm.ai)

Memory fit alone is insufficient. Determine the replicas needed for offered load from measured per-replica performance under the service objective. Multiply by GPUs per replica, add the chosen redundancy, and round to the actual purchasable node size. A deployment that fits on fewer devices may still need additional replicas at peak demand.

Accepted-Token Cost (Hypothetical Example)

Define the cost metric over one declared measurement period:

```
accepted_output_tokens = sum(
    output_tokens_for_response
    for response in period
    if task_evaluation_passes(response) and serving_SLO_passes(response)
)
quality_adjusted_cost_per_million = (
    total_period_cost / accepted_output_tokens * 1_000_000
)
```

A **service-level objective (SLO)** is the declared latency and reliability target. The configuration must first pass the aggregate quality floor and percentile SLOs. A per-response acceptance rule then determines the denominator. These are distinct gates. Do not multiply separately reported quality and latency percentages unless their joint relationship is known; calculate the intersection from response records. Also report cost per accepted task and output-length distribution so additional verbosity cannot make the token metric look better.

Table 3 uses the preceding hypothetical GPU counts, an assumed **\$3 per GPU-hour**, and a **96% task-quality floor**. All rows are assumed to pass the aggregate serving SLO. Task-quality scores and accepted-token rates are separate invented inputs; the latter include the joint response rules.

Hypothetical recipe	GP Us	Task score	Generated output tokens/s	Joint accepted output tokens/s	Cost per million accepted output tokens	Eligible at 96% floor?
FP16/BF16 reference	4	98%	400	392.0	\$8.50	Yes
FP8 W8A8	2	97%	360	349.2	\$4.77	Yes
INT4 W4A16	1	94%	240	216.0	\$3.86	No

FP8 is the least expensive eligible candidate in this constructed scenario. Relaxing the floor to **93%** admits INT4 and changes the winner. Raising it above the FP8 score leaves the reference eligible until its own score is exceeded. These statements are arithmetic consequences, not empirical recommendations. For real acceptance, substitute the agreed uncertainty-aware quality rule rather than comparing point estimates alone. (www.itl.nist.gov)

The numerator should include all provisioned capacity, idle time, host and network charges, storage, software, amortized conversion and evaluation work, and any applicable operations allocation. For ownership, use the declared capital and operating-cost model; for rental, use the actual bill. Avoid adding electricity again when it

is already included in a rental charge. Report joules per accepted token separately, using the same denominator.

Reproducible Calculator

The Python calculator below reproduces the hypothetical tables. It is self-contained, uses only the standard library, and accepts explicit topology and reservation assumptions. It prices output tokens only; input processing still consumes capacity and must remain in the workload and cost ledger.

```
from math import ceil

GIB = 2**30

def fit_gpus(parameters, bits, metadata_fraction, kv_gib,
             runtime_gib_each, usable_gib_each,
             allowed_shards=(1, 2, 4, 8)):
    payload = parameters * bits / 8 / GIB
    packed = payload * (1 + metadata_fraction)
    for shards in sorted(set(allowed_shards)):
        if shards < 1:
            raise ValueError("Shard counts must be positive")
        footprint = (packed + kv_gib) / shards + runtime_gib_each
        if footprint <= usable_gib_each:
            return shards, payload, packed, footprint
    raise ValueError("No permitted memory fit")

def accepted_cost(period_cost, accepted_output_tokens,
                  quality_score, quality_floor, aggregate_slo_pass):
    eligible = quality_score >= quality_floor and aggregate_slo_pass
    cost = (period_cost * 1_000_000 / accepted_output_tokens
           if accepted_output_tokens > 0 else float("inf"))
    return eligible, cost

# Every input below is hypothetical.
rows = [
    ("FP16/BF16", 16, 0.98, 392.0),
    ("FP8 W8A8", 8, 0.97, 349.2),
    ("INT4 W4A16", 4, 0.94, 216.0),
]

for name, bits, score, accepted_per_second in rows:
    n, payload, packed, footprint = fit_gpus(
        70e9, bits, 0.02, 12, 8, 72)
    # Choose replicas from a measured SLO-compliant capacity test.
    replicas = 1
    redundancy_gpus = 0
    node_size = 1
    provisioned = ceil(
        (n * replicas + redundancy_gpus) / node_size) * node_size
    hours = 1
    assumed_rate = 3
    extra_period_cost = 0
```

```

period_cost = provisioned * assumed_rate * hours + extra_period_cost
accepted_tokens = accepted_per_second * hours * 3600
eligible, cost = accepted_cost(
    period_cost, accepted_tokens, score, 0.96, True)
print(name, n, round(payload, 2), round(packed, 2),
      round(footprint, 2), eligible, round(cost, 2))

```

This is a preflight and accounting calculator, not a scheduler simulator. A real run should supply total measured period cost and accepted token counts from its logs. It must reject a topology that the engine cannot execute, even when the arithmetic fits. No cheaper numerical result overrides an unsuccessful quality or service gate.

Implications and Future Directions

The recommended selection bands depend on verified acceptance results:

- **Retain FP16/BF16** when compressed candidates fail the task floor or when preparing and validating compression costs more than the capacity it releases.
- **Prefer a tested FP8 recipe** when native support exists and its accepted-work economics beat the reference.
- **Prefer tested INT4 W4A16** when its physical fit or measured serving advantage removes capacity while meeting the same acceptance rules.
- **Evaluate W4A8 separately** because it changes activation precision and documented backend requirements. (github.com)
- **Evaluate FP4 separately** because its scaling and fallback behavior differ from INT4. (github.com)
- **Treat KV quantization as another experiment** with its own quality and attention-backend consequences. (docs.vllm.ai)
- **Retest after material changes** to model, tokenizer, calibration, engine, driver, GPU, context distribution, or evaluation contract.

Several failure modes deserve explicit inspection. Activation outliers can require a quantization technique designed for that distribution; SmoothQuant's offline transformation moves quantization difficulty from activations to weights. That is a particular mechanism, not a reason to assume any low-bit activation path is suitable. (proceedings.mlr.press)

For mixture-of-experts (**MoE**) deployments, inspect the actual checkpoint and runtime allocation instead of substituting active parameters into the weight-storage formula. If all experts remain resident, storage must include those resident tensors. Test routing, shared experts, excluded projections, and cross-device traffic under representative prompts. The costing framework remains usable, but its uniform-sharding assumption must be replaced where inappropriate.

Unsupported kernels and fallback layers are equally important. The cache backend and the checkpoint label can accompany changes in executed precision; current documentation already describes such alternate paths. Preserve startup logs, kernel profiles, retained-layer maps, and actual dtype records as acceptance evidence. (nvidia.github.io) (huggingface.co)

GPU Smith's adjacent advisory perspective is to turn those records into a reviewable capacity decision. The deliverable should state which configuration passed, which constraints excluded alternatives, and which inputs could reverse the decision. It should make no claim of vendor partnership or field results that the evidence does not establish.

A refresh should preserve the prior acceptance set and compare revisions against it. Pinning the runtime image and retaining the evaluation records helps distinguish a precision change from a software or workload change. MLflow documents preservation of original assessments when human experts override feedback, and Docker documents image version pinning by digest. (www.mlflow.org) (docs.docker.com)

Conclusion

The useful answer to FP16 vs FP8 vs INT4 inference cost is a qualified capacity decision: select the least expensive **eligible complete configuration**. Weight payload provides a transparent starting calculation. Exact recipes, cache allocation, runtime reservation, supported parallelism, and measured serving behavior determine whether that calculation translates into fewer physical GPUs.

Quality belongs inside that decision. Freeze the reference model and workload, define the acceptance rubric, preserve paired outputs, and publish uncertainty and subgroup results. Then enforce latency and reliability objectives at the intended offered load. Count accepted output tokens from the intersection of quality and serving decisions rather than treating an aggregate accuracy score as a token multiplier.

The hypothetical worksheet demonstrates how an attractive low-bit cost can remain ineligible, and how a different quality threshold or cache budget can change the result. Its figures illustrate the method; they are not procurement quotes, measurements, or a guarantee for a named checkpoint.

A defensible approval record contains the model and container identities, exact precision recipe, calibration procedure, actual memory and kernel evidence, evaluation results, service percentiles, integer fleet size, and dated cost ledger. With those records, FP16/BF16, FP8, INT4, and FP4 become testable alternatives. Without them, a precision label cannot establish quality-adjusted inference cost.

Source links

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

1. <https://nvidia.github.io/TensorRT-LLM/1.3.0rc3/legacy/reference/precision.html>
2. https://docs.vllm.ai/en/stable/features/quantization/llm_compressor/fp8/
3. https://huggingface.co/docs/transformers/main/quantization/concept_guide
4. <https://docs.pytorch.org/docs/stable/notes/cuda>
5. <https://aclanthology.org/2025.acl-long.1304/>
6. <https://lambda.ai/instances>
7. <https://docs.lambda.ai/public-cloud/billing/>
8. <https://nvidia.github.io/TensorRT-LLM/latest/features/quantization.html>
9. <https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/evaluate-judge-model>

10. <https://docs.ray.io/en/master/serve/llm/benchmarks.html>
11. https://github.com/NVIDIA/Model-Optimizer/blob/main/docs/source/guides/0_support_matrix.rst
12. <https://huggingface.co/docs/bitsandbytes/main/reference/nn/linear4bit>
13. <https://docs.docker.com/reference/cli/docker/image/pull/>
14. <https://www.mlflow.org/docs/latest/genai/assessments/feedback/>
15. <https://mlflow.org/docs/latest/genai/eval-monitor/scorers>
16. <https://arxiv.org/abs/2402.02750>
17. <https://www.nvidia.com/en-us/data-center/h100/>
18. <https://www.amd.com/en/products/accelerators/instinct/mi300/mi300x.html>
19. https://docs.vllm.ai/en/stable/features/quantization/quantized_kvcache/
20. <https://arxiv.org/abs/2210.17323>
21. https://docs.vllm.ai/projects/llm-compressor/en/latest/examples/quantization_w4a4_fp4/
22. https://docs.vllm.ai/en/stable/features/quantization/llm_compressor/int4/
23. <https://aclanthology.org/2025.acl-long.1304.pdf>
24. <https://arxiv.org/abs/2603.08747>
25. <https://docs.vllm.ai/projects/llm-compressor/en/stable/faq/faq/>
26. <https://www.itl.nist.gov/div898/handbook/prc/section2/prc241.htm>
27. <https://mlcommons.org/2025/04/llm-inference-v5/>
28. https://www.spec.org/power/docs/1.12/SPECpower_ssj2008-Result_File_Fields.html
29. <https://docs.nvidia.com/enterprise-reference-architectures/hgx-ai-factory-h100-h200-b200/latest/components.html>
30. <https://ir.amd.com/news-events/press-releases/detail/1173/amd-delivers-leadership-portfolio-of-data-center-ai-solutions-with-amd-instinct-mi300-series>
31. <https://aws.amazon.com/ec2/capacityblocks/pricing/>
32. <https://proceedings.mlr.press/v202/xiao23c.html>
33. <https://huggingface.co/docs/bitsandbytes/main/index>

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.