GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

NVFP4 vs MXFP4 vs FP8 Inference Compatibility Guide

> 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 · 2026-10-11 · nvfp4 · mxfp4 · fp8 · inference compatibility · vllm · sglang · tensorrt-llm · gpu inference · quantization

Original article: https://gpusmith.com/articles/en/nvfp4-vs-mxfp4-vs-fp8-runtime-compatibility

In this article

1. Executive Summary
2. Introduction and Background
3. NVFP4
4. MXFP4
5. FP8 and the BF16/FP16 Baselines
6. Feature Comparison
7. Performance and Benchmarks
8. Data Analysis and Evidence
9. Implications and Future Directions
10. Conclusion

Executive Summary

NVFP4, MXFP4, and FP8 are separate deployment contracts. Choosing an artifact requires agreement between its numeric representation, scaling recipe, serialized tensors, and the runtime kernels available on the intended graphics processing unit (GPU). **Open Compute Project (OCP) MXFP4** uses E2M1 elements with **32-value blocks and E8M0 scales**; **NVIDIA NVFP4** uses E2M1 elements with **16-value blocks and E4M3 scales**, plus a tensor-level scale. Sharing a four-bit element type does not make their scale tensors interchangeable. (www.opencompute.org) (developer.nvidia.com) (developer.nvidia.com)

As of **October 11, 2026**, the documentation reviewed supports a conditional answer. **Blackwell** has native four-bit floating-point (FP4) execution paths, but kernel availability still depends on GPU variant, model, and engine. **vLLM** documents native-kernel selection with a **Marlin W4A16 fallback** for NVFP4. **SGLang** also documents pre-Blackwell weight-only fallback and Blackwell backend choices. On AMD, **AMD Compute DNA (CDNA) 4 MI350/MI355** supports native MXFP4, while Quark describes earlier-generation MXFP4 and NVFP4 emulation as research paths. These statements establish documented capabilities, not a universal throughput or accuracy ranking. (docs.vllm.ai) (docs.sglang.io) (rocm.docs.amd.com) (quark.docs.amd.com)

Scope matters especially for **TensorRT-LLM** and **NVIDIA Inference Microservices (NIM)**. NIM's TensorRT-LLM NVFP4 profiles remain Blackwell-only; separately, a TensorRT-LLM Nemotron guide documents NVFP4 weight-only fallback on Hopper. Its GPT-OSS guide distinguishes Blackwell execution with MXFP8 activations from H200 execution with BF16 activations and MXFP4 expert weights. Treat these as different documented deployment paths rather than contradictory global support claims. (docs.nvidia.com) (nvidia.github.io) (nvidia.github.io)

For procurement, approve a **named artifact, pinned engine, exact GPU, and tested kernel path** together. FP8 often offers a broader native deployment envelope, but E4M3, E5M2, and FN/FNUZ variants still require explicit checks; TensorRT documents E4M3FN support rather than every FP8 subtype. Retain BF16 or FP16 as the evaluation reference, classify native, emulated, converted, and unsupported outcomes, and measure quality before comparing latency or throughput. MLPerf's pairing of a dataset with a quality target provides a useful methodological precedent. (docs.vllm.ai) (onnx.ai) (docs.nvidia.com) (mlcommons.org)

Introduction and Background

The question “NVFP4 vs MXFP4 vs FP8 inference compatibility” is a deployment decision before it is a precision comparison. A model registry entry can identify a quantized format without establishing the operations that will execute it. PyTorch explicitly describes some low-precision types as shell dtypes with limited operation and backend support. A dtype appearing in a framework therefore provides weaker evidence than a documented, selected inference kernel. (docs.pytorch.org)

Here, **general matrix multiplication (GEMM)** is the linear-algebra operation whose kernel support must be verified. **Open Neural Network Exchange (ONNX)** is the graph representation discussed below; representing a type in a graph does not establish execution on a GPU.

This report separates **four layers**. The **numeric format** defines representable values. The **quantization scheme** defines grouping, scales, rounding, and the tensors being quantized. The **artifact encoding** defines packed data, scale arrays, and metadata. The **execution path** identifies the loader, kernels, hardware instructions, and any fallback. The distinction follows from OCP leaving physical memory layout unspecified and from Safetensors exposing tensor metadata without defining a universal quantization recipe. (www.opencompute.org) (huggingface.co)

The intended audience is private-inference architects, model evaluators, platform teams, and procurement engineers. GPU Smith's published scope includes specifying and validating private AI infrastructure; its relevance here is the engineering acceptance decision, rather than participation as a format or runtime vendor. The compatibility tables compare formats and software paths, with advisory considerations kept in prose. (gpusmith.com)

The evidence is a documentation review accessed on the publication date. The verified OCP document is **Microscaling Formats version 1.0, September 2023**; this report does not claim that a search proves the absence of later revisions. The documentation versions and source commits reviewed for the runtime ledger are listed beside Table 2; they identify documentation provenance, not tested engine installations. Reconcile each documented capability with the actual release installed for deployment. (www.opencompute.org)

Native means the relevant operation uses hardware acceleration for the stated low-precision representation. **Emulated** means a documented software path reconstructs or simulates that representation using another compute precision. **Converted** means deployment changes the quantization representation, offline or during loading. **Unsupported** means the requested combination lacks a documented accepted path in the stated scope. A separate **not established** label records insufficient evidence instead of asserting universal impossibility.

NVFP4

Capabilities

NVFP4's E2M1 element has a sign bit, exponent bits, and a mantissa bit, but the practical identity also includes its scaling hierarchy. NVIDIA describes a micro-block scale and a tensor-level 32-bit floating-point (FP32) scale. Record the complete recipe rather than reduce it to the element dtype. (developer.nvidia.com) (developer.nvidia.com)

TensorRT's NVFP4 implementation specifies a block size of **16** and restrictions on the blocking dimension. Its dynamic double-quantization workflow uses a calibrated tensor scale and consecutive dequantization operations. An exporter must reproduce the expected structure, not merely produce values that fit into an FP4 field. (docs.nvidia.com) (docs.nvidia.com)

W4A4 denotes four-bit weights and four-bit activations; **W4A16** denotes four-bit weights with sixteen-bit activations. These labels describe different execution workloads even when both load the same NVFP4 checkpoint. vLLM's ModelOpt documentation explicitly selects an available NVFP4 kernel or falls back to Marlin weight-only execution. (docs.vllm.ai)

Adoption

A concrete artifact interface is **ModelOpt's hf_quant_config.json**, which vLLM uses to detect supported quantization algorithms. The configuration is part of the loader contract. Retaining it alongside tensor files makes the intended recipe inspectable; renaming the checkpoint does not replace this metadata. (docs.vllm.ai)

SGLang documents **modelopt_fp4**, Blackwell kernel choices, and Marlin fallback on earlier NVIDIA architectures. It also names an AMD **petit_nvfp4** path. The latter establishes a documented way to consume an NVFP4 artifact, but does not establish native AMD FP4 hardware execution for that checkpoint. (docs.sglang.io) (docs.sglang.io)

NIM adds packaged-profile policy. Enabling **NIM_ALLOW_NVFP4_EMULATION=1** permits selected vLLM/SGLang profiles to reconstruct weights into BF16 in the Marlin kernel on eligible pre-Blackwell hardware. The documented relaxation applies per host, which matters when validating nodes with heterogeneous GPUs. (docs.nvidia.com) (docs.nvidia.com)

Strengths and Limitations

NVFP4 is a candidate when the target stack has the required native path and the model passes evaluation. **Lower stored weight precision** is a storage opportunity, not sufficient evidence for lower latency. NIM explicitly warns that its emulation can be slower than supported native precisions on the same GPU. (docs.nvidia.com)

The evaluation must identify **dense versus mixture-of-experts (MoE)** coverage. NIM notes that FP4 MoE kernels may be unavailable in some backend versions. A dense-layer smoke test should not be accepted as evidence that all expert kernels are supported. (docs.nvidia.com)

Avoid labeling all Hopper deployments unsupported: the general TensorRT-LLM Nemotron guide documents a W4A16 fallback without checkpoint conversion. Conversely, that model-specific evidence does not broaden NIM's packaged-profile policy. Record the model guide and package policy separately in the registry. (nvidia.github.io)

MXFP4

Capabilities

Open Compute Project (OCP) MXFP4 defines a microscaling representation with a shared power-of-two scale. Its published format structure differs from NVFP4's finer-grained scale hierarchy. That distinction affects reconstruction and quantization error even though the element representation is E2M1 in both cases. (www.opencompute.org)

The standard does not fix the physical memory layout. Implementations can pack or reorder weights and scales to match a kernel's access pattern. Triton's block-scaled matrix-multiplication tutorial explicitly discusses packed scale layouts. Standardized numeric semantics are useful for portability, but they do not create an engine-independent binary checkpoint. (www.opencompute.org) (triton-lang.org)

OCP leaves internal dot-product precision and operation order implementation-defined. Obtain the selected kernel's accumulation behavior separately. Neither “MXFP4” nor a nominal tensor-core precision alone specifies the complete reduction path. (www.opencompute.org)

Adoption

GPT-OSS provides a published model-family example: OpenAI distributes downloadable weights described as natively quantized in MXFP4, and describes the models as MoE Transformers. This establishes a real artifact family, not general support for every MXFP4 checkpoint in every engine. (openai.com) (openai.com)

TensorRT-LLM's GPT-OSS guide specifies different activation and backend choices on Blackwell and H200. vLLM documents checkpoint-specific activation overrides and notes that some MXFP4 linear backends use BF16 activations. Record the actual activation precision rather than infer W4A4 from the weight label. (nvidia.github.io) (docs.vllm.ai) (docs.vllm.ai)

On AMD, CDNA4 documentation lists hardware support for MXFP4. Quark supplies an export workflow into Safetensors and vLLM. Those references cover hardware capability and an artifact interface respectively; successful deployment still requires matching the tutorial's supported model and software configuration. (rocm.docs.amd.com) (rocm.docs.amd.com)

Strengths and Limitations

MXFP4 offers a standards-defined numeric vocabulary across implementations. Its value for multi-vendor planning is that the element and shared-scale semantics can be stated independently of a single checkpoint container. The portability limit remains the standard's deliberate lack of a physical-layout prescription. (www.opencompute.org)

Earlier AMD hardware is a counterexample to a simplistic compatibility label. ATOM states that MI300 does not accelerate MXFP4, while Quark documents an emulation path for MI300 and MI325 intended for research. Both can be true: accepting an artifact and accelerating its format are separate properties. (rocm.docs.amd.com) (quark.docs.amd.com)

For NVIDIA, do not group every Blackwell device into an identical kernel target. vLLM documents optional **b12x kernels for SM120 and SM121**, NVIDIA streaming-multiprocessor (SM) architecture targets, including separate MoE parallelism constraints. A data-center kernel validated on one target does not establish availability of that kernel on another Blackwell variant. (docs.vllm.ai)

FP8 and the BF16/FP16 Baselines

Capabilities

Eight-bit floating point (FP8) is a family. The original formats paper defines E4M3 with exponent/mantissa allocation **4/3** and E5M2 with **5/2**. It distinguishes their special-value conventions, including E4M3's absence of infinities. Those semantics matter for conversion and numerical validation. (arxiv.org) (arxiv.org)

ONNX documents **E4M3FN, E4M3FNUZ, E5M2, and E5M2FNUZ** as separate types. FN denotes finite values; FNUZ denotes finite values with unsigned zero. The variants differ in infinity and signed-zero behavior. Its documentation warns against treating E4M3FN and E4M3FNUZ as a simple interchange because their exponent biases differ. (onnx.ai) (onnx.ai)

Bfloat16 (BF16) and **half precision (FP16)** are both sixteen-bit formats, with different exponent and mantissa allocation. They serve as reference execution and evaluation choices rather than guarantees of bit-identical results across kernels. (docs.pytorch.org)

Adoption

vLLM documents hardware-accelerated FP8 weight-and-activation execution on Ada, Hopper, Blackwell, and AMD MI300X, while separating older NVIDIA weight-only execution through Marlin. Thus FP8 weights loading on Ampere should not be reported as native FP8 matrix multiplication. (docs.vllm.ai)

SGLang lists **w8a8_fp8** support on NVIDIA and AMD, with Aiter or Triton FP8 on AMD. The documented scheme still needs a compatible hardware-specific backend. (docs.sglang.io)

AMD HIP, the Heterogeneous-computing Interface for Portability, documents the FNUZ default on gfx94x and distinguishes native FP8 type support by architecture. Record the exact subtype when moving artifacts between hardware families. (rocm.docs.amd.com) (rocm.docs.amd.com)

FP8 is also a collection of scaling recipes. TensorRT-LLM separates per-tensor, blockwise, rowwise, and key/value (KV) cache quantization. Its blockwise recipe uses different scale representations between specified Blackwell targets and Hopper. A comparison labeled only “FP8” omits information necessary to reproduce the run. (nvidia.github.io) (nvidia.github.io)

Strengths and Limitations

FP8 may be the practical candidate when native four-bit kernels are unavailable. This is a compatibility judgment, not a quality or performance promise. ONNX Runtime states that quantization performance depends on the model and hardware, including the effects of quantization/dequantization overhead. (onnxruntime.ai)

KV-cache precision should be recorded independently from weight and activation precision. TensorRT-LLM documents a specific NVFP4 KV-cache workflow requiring offline ModelOpt quantization and FP8 weight/activation quantization. That is a distinct combination from an NVFP4 weight checkpoint, despite sharing the label. (nvidia.github.io)

A framework's packed FP4 dtype does not solve the deployment contract. PyTorch's **float4_e2m1fn_x2** packs two values into a byte, while its shell-type operations remain selectively supported. Serialization capability and arithmetic coverage must be validated separately. (docs.pytorch.org) (docs.pytorch.org)

Feature Comparison

Format anatomy and execution vocabulary

Table 1 compares storage semantics with the information a deployment record still needs. Exponent/mantissa labels describe element encoding; they do not describe activation coverage or accumulator precision. The baseline layouts come from PyTorch, and the FP8 layouts from the original formats paper. (docs.pytorch.org) (arxiv.org)

Format	Element and scaling	Artifact implication	Execution question
NVFP4	E2M1; E4M3 micro-block scales; tensor-level scale. (docs.nvidia.com)	Retain packed weights and both scale levels.	Native FP4 GEMM, weight-only reconstruction, or conversion?
OCP MXFP4	E2M1; E8M0 shared scale per standard block. (www.opencompute.org)	Verify packing, scale layout, and grouping axis.	Which activation dtype and accumulator does the kernel use?
FP8	E4M3 or E5M2, with distinct FN/FNUZ variants. (onnx.ai)	Preserve subtype and quantization scale recipe.	Native FP8 weight/activation execution or weight-only fallback?
BF16 / FP16	Sixteen-bit floating-point baselines. (docs.pytorch.org)	Retain original tensors for evaluation and conversion.	Which operators and reductions use which precision?

The table's central implication is that **a format name is necessary but incomplete**. The MX standard leaves accumulation details implementation-defined, and PyTorch restricts operations on shell dtypes. A valid registry entry needs both representation details and execution evidence. (www.opencompute.org) (docs.pytorch.org)

Source-linked runtime compatibility ledger

Table 2 summarizes documentation-supported paths reviewed on **October 11, 2026**. **N** means native execution is documented in the stated scope; **E** means emulation or weight-only fallback; **C** means conversion; **U** means unsupported in that documented scope; **NE** means not established by the reviewed evidence. A row is a starting point for validation, not an assurance that every architecture or operator is covered. No execution logs or deployment test results are reported here; the labels below describe documentation support, not author-tested deployments.

Documentation provenance for Table 2. These are the documentation editions or source revisions checked for this review, not a jointly tested software stack. Versioned and commit-pinned links below replace changing runtime citations where verified.

Engine or evidence layer	Reviewed documentation identity	Scope and reproducibility limit
vLLM	**v0.31.0**: ModelOpt, online quantization, FP8, and hardware compatibility.	Versioned documentation for loader, activation and fallback behavior; no installed vLLM build was tested for this article.
SGLang	Documentation source commit **bdf8886ad386685b4d089c8e66725b9280533591**: quantization guide.	Source-file revision checked for platform, backend and conversion descriptions; it does not identify a tested serving release or prove the build identity of the live documentation site.
TensorRT-LLM	The live quantization page identifies generating commit **c76f4a8**: page provenance. Reviewed quantization source, hardware source, Nemotron guide, and GPT-OSS guide are pinned to that commit.	Documentation commit, not a tested engine binary; retain model-specific and packaged-profile distinctions.
NIM	**2.0.13**: advanced configuration.	Packaged-profile policy is separate from general TensorRT-LLM behavior; validate the actual container and profile.
Core TensorRT	**11.4.0**: quantization workflows and explicit quantization.	Graph and type support in this documentation edition is separate from serving-engine model coverage.
AMD Quark evidence	**0.13**: PyTorch serving guide; **release-0.12**: the ONNX microscaling tutorial already linked in the ledger.	The 0.13 guide remains a live `/latest/` source; a release-0.13 URL could not be retrieved. Its software-build provenance is unresolved, so its emulation guidance remains documentation-only evidence requiring deployment validation.
ONNX representation	The ledger's QuantizeLinear link is an operator-specification reference, not an inference-engine release.	It does not establish a tested execution-provider or hardware path.

Artifact and target	vLLM	SGLang	TensorRT-LLM / NIM	ONNX / core TensorRT and limits
NVFP4, NVIDIA Blackwell	**N**, supported native GEMM selection; ModelOpt loader. (docs.vllm.ai)	**N**, backend depends on GPU target. (docs.sglang.io)	**N**, model matrix required; NIM native profiles. (docs.nvidia.com)	**N** for documented NVFP4 scheme and valid graph, not arbitrary FP4 files. (docs.nvidia.com)
NVFP4, NVIDIA Ampere/Ada/Hopper	**E**, Marlin W4A16 when native kernel unavailable. (docs.vllm.ai)	**E**, documented pre-Blackwell Marlin path. (docs.sglang.io)	**E** documented for Nemotron on Hopper; **U** for NIM's TensorRT-LLM NVFP4 profiles. (nvidia.github.io) (docs.nvidia.com)	**NE** for universal NVFP4 fallback; validate target-specific graph and kernels.
NVFP4, AMD Instinct	**E**, Quark documents MI3xx research emulation. (quark.docs.amd.com)	Documented AMD petit path; **C** to MXFP4 available on FP4-capable AMD targets. (docs.sglang.io)	**NE**; the NVIDIA hardware-support page does not establish AMD deployment. (nvidia.github.io)	**NE** for a generic NVFP4-to-AMD contract.

Artifact and target	vLLM	SGLang	TensorRT-LLM / NIM	ONNX / core TensorRT and limits
MXFP4, Blackwell	Backend-specific path; inspect activation precision. (docs.vllm.ai)	Broad format support is insufficient to establish every NVIDIA model/backend combination: **NE** pending that check.	**N** for documented GPT-OSS Blackwell expert path, with MXFP8 activations. (nvidia.github.io)	**NE** for direct MXFP4 in core TensorRT; its documented MX workflow is MXFP8. (docs.nvidia.com)
MXFP4, Hopper or earlier NVIDIA	**E** where a compatible weight-only backend exists; **U** for Marlin MXFP4 on Turing. (docs.vllm.ai)	**NE** for a universal model/backend path.	**E** for GPT-OSS H200 expert weights with BF16 activations, not native FP4. (nvidia.github.io)	**NE**; ONNX representation is not hardware support.
MXFP4, AMD CDNA4 / CDNA3	**N** on documented CDNA4 export path; **E** research-only on MI300/MI325. (quark.docs.amd.com)	**C**, quark_mxfp4 can convert NVFP4/BF16 into MXFP4 on supported CDNA4 hardware. (docs.sglang.io)	**NE** for AMD.	AMD Quark documents a custom microscaling operator; validate the execution provider. (quark.docs.amd.com)
FP8, NVIDIA and AMD	**N** on documented FP8-capable targets; **E** weight-only on older NVIDIA targets. (docs.vllm.ai)	Documented w8a8_fp8 support on NVIDIA/AMD; verify the selected hardware-specific backend. (docs.sglang.io)	Distinct FP8 scaling and cache recipes; model-specific coverage. (nvidia.github.io)	TensorRT supports E4M3FN; ONNX supports additional variants. (docs.nvidia.com)
BF16 / FP16 reference	Starting tensors can be **C** to online low-precision recipes during loading. (docs.vllm.ai)	Starting tensors can be **C** in documented AMD quark_mxfp4 path. (docs.sglang.io)	Hardware breadth is not per-model/per-format proof.	ONNX quantization granularity follows scale shape; graph validation remains required. (onnx.ai)

Read the ledger by **artifact, target, engine, then kernel**. Native hardware capability is only one input. Triton documents accelerated block-scaled kernels on NVIDIA compute-capability 10 hardware and AMD CDNA4, while vLLM exposes separate SM120/SM121 kernel choices. None independently certifies a particular serving model. (triton-lang.org) (triton-lang.org) (docs.vllm.ai)

Decision tree and conversion risks

Use the following acceptance sequence:

- **Starting artifact:** Inspect tensor dtypes, shapes, offsets, and scale tensors before choosing a loader. Safetensors metadata supports that inspection. (huggingface.co)
- **Loader contract:** Match exporter metadata to the engine's schema, including ModelOpt's configuration where applicable. (docs.vllm.ai)
- **Target kernel:** Inspect selected linear and expert backends; distinguish production kernels from testing emulation. (docs.vllm.ai)
- **Conversion allowance:** If representation changes during loading, record the resulting scheme as a new deployment variant. vLLM documents load-time conversion. (docs.vllm.ai)

- **Quality tolerance:** Define the task's acceptance threshold before comparing speed, following the principle of a benchmark dataset paired with a quality target. (mlcommons.org)
- **Deployment disposition:** Accept the native path, accept a measured emulated path, authorize a measured conversion, or retain the artifact as unsupported for that pinned stack.

Conversion is not relabeling. OCP MXFP4 and NVFP4 have different scale granularities. Reconstructing approximate source values and requantizing into a target representation can create a different artifact; it cannot restore information already discarded. Preserve the original checkpoint and treat conversion as an independently evaluated candidate. (www.opencompute.org)

The conversion review should cover:

- **Calibration:** Preserve dataset identity, preprocessing, and selected samples. Quark's NVFP4 recipe needs calibration for global activation scales. (quark.docs.amd.com)
- **Scale reconstruction:** Preserve both NVFP4 scale levels and the documented dequantization sequence. (docs.nvidia.com)
- **Packing:** Confirm nibble order, tensor strides, and repacking. TensorRT stores packed FP4 with a specific element order. (docs.nvidia.com)
- **Graph operators:** Match quantize/dequantize (Q/DQ) graphs to parser support; TensorRT does not accept every pre-quantized ONNX operator. (docs.nvidia.com)
- **Custom operators:** Treat an exporter-specific microscaling operator as an execution-provider dependency. (quark.docs.amd.com)
- **Fallback precision:** Record each reconstructed weight and activation dtype. vLLM MXFP4 backends need not use FP4 activations. (docs.vllm.ai)
- **Metadata retention:** Carry scale format and grouping metadata forward. AMD's published checkpoint example records this contract in configuration. (rocm.blogs.amd.com)

Performance and Benchmarks

No controlled cross-format experiment was performed for this report. The evidence supports a **validation protocol**, not a universal NVFP4, MXFP4, or FP8 accuracy ranking. Native low-precision kernels, weight-only fallback, and converted artifacts should be reported separately. ONNX Runtime's documented dependence on model and hardware is a reason to avoid predicting speed from bit width alone. (onnxruntime.ai)

Begin with a BF16 or FP16 reference and the same application evaluation set. Compare **task quality**, numerical outputs, and runtime behavior independently. A quantized candidate can differ in output without necessarily failing the task, while a numerically close candidate can miss an application requirement. Declare acceptance thresholds before seeing the results. MLPerf couples performance work to a dataset and quality target. (mlcommons.org)

A useful evaluation sequence is:

- **Provenance:** Hash source and candidate artifacts, tokenizer, configuration, and conversion outputs.

- **Task specification:** Preserve prompts, few-shot configuration, generation arguments, metrics, and evaluation-code revision. The evaluation harness documents sharing task configuration with a commit hash. (github.com)
- **Layer checks:** Inspect representative reconstructed tensors, output drift, saturation, and invalid values before application testing.
- **Task checks:** Measure application scores, including subgroup or long-context behavior where required; record uncertainty and repeats. Harness configuration makes repeats explicit. (github.com)
- **Kernel checks:** Inspect dispatch and fallback; a testing emulation backend is not production-performance evidence. (docs.vllm.ai)
- **Warmup:** Separate load, compilation, graph capture, and steady-state service. PyTorch recommends warmup before benchmarking. (docs.pytorch.org)
- **Timing:** Synchronize GPU work appropriately when measuring kernels. PyTorch warns that unsynchronized measurements can capture launch time instead of completed work. (docs.pytorch.org)
- **Memory:** Record checkpoint bytes, post-load allocation, peak working allocation, cache allocation, and reserved memory separately.
- **Service metrics:** Record time to first token (TTFT), inter-token latency (ITL), time per output token (TPOT), and throughput under declared load.
- **Power:** Record measurement boundary and workload. MLPerf power measurement uses whole-system alternating-current (AC) power at the wall. (mlcommons.org)

TTFT covers request arrival through first-token availability under the chosen measurement boundary. **ITL** is the interval between successive emitted tokens; retain its distribution. **TPOT** summarizes output-token time and should not silently replace that distribution. MLPerf rules distinguish first-token latency from average intervals between generated tokens. State whether client transport, queueing, and tokenization are included. (github.com)

Warmup must not hide operating conditions. Report cold-start behavior separately when a private deployment scales workers down or frequently replaces models. During steady-state testing, keep request distribution, concurrency, context length, and stopping criteria consistent. Kernel timing and end-to-end serving latency answer different questions; synchronized GPU timing is necessary for the former but excludes other service costs. (docs.pytorch.org)

The **do-not-compare checklist** is short:

- **Different models:** Do not interpret a model-family change as a precision effect.
- **Different execution:** Do not compare native W4A4 with weight-only fallback without labeling the compute difference.
- **Different service load:** Do not combine single-stream latency with saturated aggregate throughput.
- **Different context/cache:** Do not change sequence lengths, cache precision, or prefix reuse without reporting the effect.
- **Different preparation:** Do not compare an uncalibrated conversion with a tuned artifact as though format alone explains the result.
- **Different measurement scope:** Do not compare GPU telemetry power with whole-system wall power; benchmark power remains tied to its accompanying workload. (mlcommons.org)

Report **quality first, deployment feasibility second, performance third**. If the artifact fails the quality target, throughput is not a procurement justification. If quality passes but the kernel is emulated, the deployment may still be useful, provided its performance and operational requirements meet the acceptance criteria.

Data Analysis and Evidence

Storage calculations with explicit assumptions

A theoretical weight-storage estimate is useful for planning, provided it is separated from measured video random-access memory (VRAM). For **P stored values**, **b payload bits per value**, **k values per scale block**, **s scale bits**, and **M additional bytes**, use:

B = P × b / 8 + ceil(P / k) × s / 8 + M.

For real tensors, apply the ceiling separately per tensor and include alignment, padding, unquantized layers, indexes, and global scales in M. This is an analyst calculation based on format structure. OCP MXFP4 defines four-bit elements and an eight-bit scale per block of thirty-two, while NVFP4 uses sixteen-value micro-block scaling. (www.opencompute.org) (docs.nvidia.com)

Worked example (Hypothetical Example), storage only: assume exactly **70,000,000,000 values**, every value quantized, divisible block counts, no padding, and no other metadata except stated scales. Use decimal gigabytes, meaning **1 GB = 1,000,000,000 bytes**. BF16/FP16 payload is **140 GB**; FP8 payload is **70 GB**, excluding its recipe-dependent scales. These totals are arithmetic from sixteen-bit and eight-bit element widths, not observed device allocations. (docs.pytorch.org) (arxiv.org)

Under those assumptions, MXFP4 payload is **35 GB**, shared scales add **2.1875 GB**, and the sum is **37.1875 GB**, or **4.25 bits per value**. NVFP4 payload is **35 GB**, micro-block scales add **4.375 GB**, and the subtotal is **39.375 GB**, or **4.5 bits per value**, before tensor-level FP32 scales. These are derived quantities, not benchmark measurements. (www.opencompute.org) (developer.nvidia.com)

The estimate excludes **KV cache, activations, workspaces, graph capture, replicas, and engine allocations**. It assumes complete quantization coverage, which should be checked against the real artifact rather than inferred from a model name. OpenAI's GPT-OSS model card distinguishes total parameters from active parameters; storage accounting should count stored tensors rather than substitute active-per-token counts. (deploymentsafety.openai.com) (huggingface.co)

Artifact evidence and benchmark-report template

AMD's Kimi MXFP4 example exposes the deployment contract: packed FP4 weights, separate E8M0 scales, and configuration fields for per-group quantization, group size, and scale format. Its runtime dynamically quantizes activations. The example illustrates what to inspect; it is not a universal Safetensors schema. (rocm.blogs.amd.com) (rocm.blogs.amd.com) (rocm.blogs.amd.com)

Table 3 is a **recommended benchmark-report template**, not a results table. It translates compatibility questions into reviewable measurements. Reproducibility fields follow the evaluation harness's configuration-and-revision approach; quality and power fields follow MLCommons methodology. (github.com) (mlcommons.org)

Field	Record for every candidate	Interpretation
Artifact	Source revision, hashes, tokenizer, quantization configuration, tensor/scales inventory.	Establishes what was tested.
Software	Engine release/commit, container digest, driver, CUDA or ROCm, kernel libraries.	Makes a support claim reproducible.
Execution	GPU model, selected linear/MoE kernels, native/emulated/converted status, activation/cache/accumulator dtypes.	Prevents a storage label from becoming a compute claim.
Quality	Dataset revision, prompts, metrics, thresholds, repeats, reference results.	Determines whether the candidate is acceptable.
Latency and load	TTFT, ITL distribution, TPOT definition, throughput, concurrency, input/output lengths.	Makes service comparisons interpretable.
Memory and preparation	Load time, warmup policy, peak/resident/reserved memory, conversion-time allocation.	Separates capacity planning from file size.
Power	Sensor boundary, whole-system or GPU-only measurement, workload, sample interval, energy.	Keeps energy comparisons tied to the same task.

A filled report should support replay by another engineer. Missing kernel identity is a compatibility gap; missing quality thresholds is an acceptance gap. For energy comparisons, retain the measured workload because MLCommons states that a power result is valid for its accompanying benchmark. (mlcommons.org)

The evidence does not justify a single percentage advantage for any format. A useful procurement comparison instead reports which candidates passed quality, which executed natively, and which met the service objective within the available memory and power envelope.

Implications and Future Directions

For procurement, acquire evidence for a **specific deployment tuple**: artifact revision, exporter recipe, engine build, GPU model, kernel backend, and workload envelope. The engine's broad hardware page and its quantization/model matrix answer different questions. TensorRT-LLM publishes both, illustrating why hardware support cannot stand in for per-format or per-model support. (nvidia.github.io) (nvidia.github.io)

A private model registry should retain an acceptance record with the following fields:

- **Source identity:** Original repository, revision, tensor hashes, and license record.
- **Numeric semantics:** Exact FP4 or FP8 subtype, including special-value conventions.
- **Quantization scheme:** Weight, activation, and cache precision; block axes, scales, rounding, calibration, and excluded modules.
- **Artifact encoding:** Packed layout, byte/nibble order, scale tensor names, and exporter schema.
- **Conversion lineage:** Source artifact, converter version, settings, and resulting artifact hashes.
- **GPU identity:** Exact product and architecture target, rather than only a generation name.
- **Kernel evidence:** Selected linear and expert backends, fallback messages, and representative profiler traces.

- **Software provenance:** Engine commit/release, container digest, dependency versions, and build options.
- **Quality evidence:** Reference artifact, datasets, evaluation configuration, results, and acceptance thresholds.
- **Service evidence:** Request distribution, concurrency, cache policy, latency distributions, throughput, and memory.
- **Power evidence:** Instrumentation boundary, workload, and energy calculation when measured.
- **Acceptance owner:** Decision, permitted operating envelope, unresolved limitations, and revalidation trigger.

These are recommended controls rather than claims that a registry implements them. Their purpose is to preserve the distinction between representation and deployed behavior. Safetensors metadata inspection can support the tensor inventory; it does not supply every field in this record. (huggingface.co)

For disconnected environments, collect the exact software and model dependencies needed to replay the accepted tuple before moving it into the private boundary. GPU Smith's published validation method describes acceptance testing against written criteria; format compatibility is one such criterion that should be concrete and reproducible. (gpusmith.com)

Future interoperability should be judged by **documented importer/exporter agreement and executable tests**. ONNX QuantizeLinear represents per-tensor, per-axis, and blocked quantization, while version **23** includes FP4E2M1. Those are representation capabilities. Core TensorRT's documented schemes and AMD Quark's custom microscaling operator demonstrate that execution-provider coverage requires separate validation. (onnx.ai) (onnx.ai) (quark.docs.amd.com)

Standardization can reduce ambiguity without eliminating deployment work. The next useful compatibility update is a versioned, source-linked ledger with tested artifacts and known execution paths, rather than a blanket statement that all four-bit formats are portable.

Conclusion

NVFP4 vs MXFP4 vs FP8 inference compatibility is decided by the complete deployment path. The element format, scale recipe, checkpoint encoding, and selected kernels must agree. A model that loads through emulation is usable only under the measured conditions that justify accepting that path; successful loading alone does not establish native execution or a performance advantage.

NVFP4 and MXFP4 should remain separate registry entries even when both use FP4 elements. FP8 should include its subtype and scaling recipe. BF16 or FP16 provides the reference artifact for the evaluation protocol, while weight, activation, and cache precision remain independent fields.

The practical sequence is to inspect the artifact, match the loader contract, establish the target kernel, classify conversion or fallback, and evaluate quality before measuring steady-state service behavior. Where documentation is incomplete, preserve the uncertainty and require deployment evidence. Where documentation is model-specific or package-specific, preserve that scope.

For a hardware or model commitment, the reviewable deliverable is an accepted deployment tuple and a benchmark report another engineer can replay. That approach makes a low-precision artifact an engineering decision supported by evidence, rather than a purchasing decision inferred from its bit-width label.

Source links

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

1. https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf
2. https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/
3. https://docs.vllm.ai/en/latest/features/quantization/modelopt/
4. https://docs.sglang.io/docs/advanced_features/quantization
5. https://rocm.docs.amd.com/projects/atom/en/latest/online_quantization_best_practices.html
6. https://quark.docs.amd.com/latest/pytorch/quantizing_large_models.html
7. https://docs.nvidia.com/nim/large-language-models/latest/reference/advanced-configuration.html
8. https://nvidia.github.io/TensorRT-LLM/latest/deployment-guide/deployment-guide-for-nemotron-3-on-trtllm.html
9. https://nvidia.github.io/TensorRT-LLM/deployment-guide/deployment-guide-for-gpt-oss-on-trtllm.html
10. https://docs.vllm.ai/en/latest/features/quantization/llm_compressor/fp8/
11. https://onnx.ai/onnx/technical/float8.html
12. https://docs.nvidia.com/deeplearning/tensorrt/latest/inference-library/quantized-types-explicit-quantization.html
13. https://mlcommons.org/benchmarks/inference-datacenter/
14. https://docs.pytorch.org/docs/main/tensor_attributes.html
15. https://huggingface.co/docs/safetensors/metadata_parsing
16. https://gpusmith.com/
17. https://docs.nvidia.com/deeplearning/tensorrt/latest/inference-library/quantized-types-schemes.html
18. https://docs.nvidia.com/deeplearning/tensorrt/latest/inference-library/quantized-types-workflows.html
19. https://triton-lang.org/main/getting-started/tutorials/10-block-scaled-matmul.html
20. https://openai.com/index/introducing-gpt-oss/
21. https://docs.vllm.ai/en/latest/features/quantization/online/
22. https://rocm.docs.amd.com/en/docs-10.1.0/reference/gpu-arch/mi350.html
23. https://rocm.docs.amd.com/projects/ai-developer-hub/en/v8.0/notebooks/gpu_dev_optimize/mxfp4_quantization_quark_vllm.html
24. https://gpusmith.com/articles/en/cuda-compute-capability-table-gpu
25. https://docs.vllm.ai/en/latest/features/quantization/b12x/
26. https://arxiv.org/abs/2209.05433
27. https://rocm.docs.amd.com/projects/HIP/en/latest/reference/low_fp_types.html
28. https://nvidia.github.io/TensorRT-LLM/features/quantization.html
29. https://onnxruntime.ai/docs/performance/model-optimizations/quantization.html
30. https://gpusmith.com/articles/en/vllm-kv-cache-capacity-planner

31. https://docs.vllm.ai/en/v0.31.0/features/quantization/modelopt/
32. https://docs.vllm.ai/en/v0.31.0/features/quantization/online/
33. https://docs.vllm.ai/en/v0.31.0/features/quantization/llm_compressor/fp8/
34. https://docs.vllm.ai/en/v0.31.0/features/quantization/
35. https://github.com/sgl-project/sglang/blob/bdf8886ad386685b4d089c8e66725b9280533591/docs/docs/advanced_features/quantization.mdx
36. https://github.com/NVIDIA/TensorRT-LLM/blob/c76f4a8/docs/source/features/quantization.md
37. https://github.com/NVIDIA/TensorRT-LLM/blob/c76f4a8/docs/source/supported-hardware.md
38. https://github.com/NVIDIA/TensorRT-LLM/blob/c76f4a8/docs/source/deployment-guide/deployment-guide-for-nemotron-3-on-trtllm.md
39. https://github.com/NVIDIA/TensorRT-LLM/blob/c76f4a8/docs/source/deployment-guide/deployment-guide-for-gpt-oss-on-trtllm.md
40. https://docs.nvidia.com/nim/large-language-models/_2.0.13/2.0.13/reference/advanced-configuration.html
41. https://docs.nvidia.com/deeplearning/tensorrt/11.4.0/inference-library/quantized-types-workflows.html
42. https://docs.nvidia.com/deeplearning/tensorrt/11.4.0/inference-library/quantized-types-explicit-quantization.html
43. https://quark.docs.amd.com/release-0.12/onnx/tutorial_microscaling_quantization.html
44. https://onnx.ai/onnx/operators/onnx__QuantizeLinear.html
45. https://docs.vllm.ai/en/latest/api/vllm/config/kernel/
46. https://rocm.blogs.amd.com/software-tools-optimization/kimi-k25-mxfp4-atom/README.html
47. https://github.com/EleutherAI/lm-evaluation-harness/blob/main/docs/task_guide.md
48. https://docs.pytorch.org/tutorials/recipes/recipes/benchmark.html
49. https://github.com/mlcommons/inference_policies/blob/master/inference_rules.adoc
50. https://gpusmith.com/articles/en/llm-inference-hardware-sizing-guide
51. https://deploymentsafety.openai.com/gpt-oss/tacit-knowledge-and-troubleshooting
52. https://nvidia.github.io/TensorRT-LLM/supported-hardware.html
53. https://gpusmith.com/articles/en/air-gapped-llm-deployment-guide

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.