GPU SMITH
COMPUTE ENGINEERING

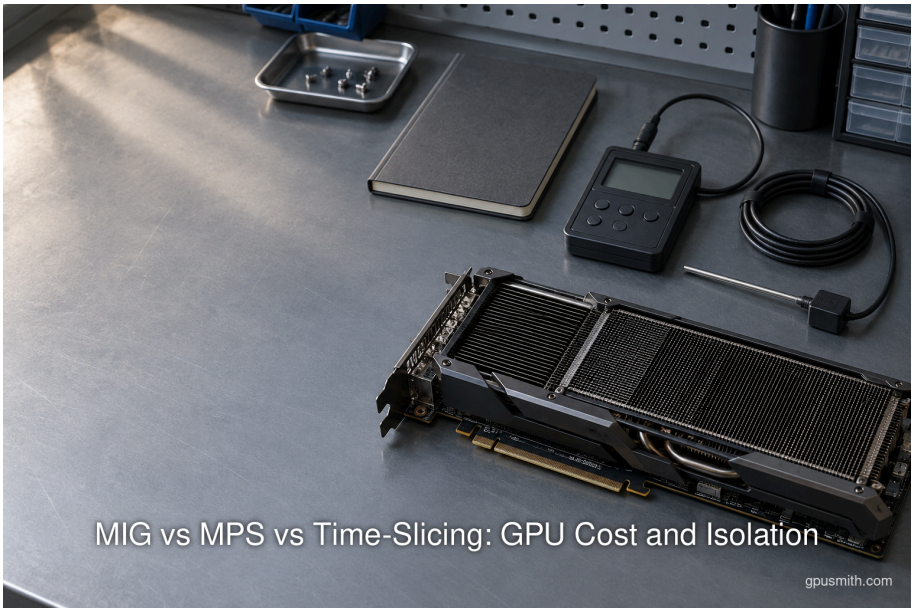
GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

MIG vs MPS vs Time-Slicing: GPU Cost and Isolation

> 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 · 2026-09-26 · mig vs mps vs time slicing · gpu sharing · gpu inference · nvidia mig · cuda mps · gpu time-slicing · kubernetes gpu · multi-tenant inference · gpu cost

Original article: https://gpusmith.com/articles/en/mig-vs-mps-vs-time-slicing



In this article

Executive Summary

For a private Kubernetes cluster, **exclusive GPU allocation** is the simplest allocation and the clearest tenant boundary. **Multi-Instance GPU (MIG)** is the option when supported hardware must offer physically separated compute and memory paths, bounded framebuffer capacity, memory-bandwidth quality of service, and an independently documented fault domain. **Multi-Process Service (MPS)** improves concurrency for processes that leave GPU capacity idle and can enforce compute and memory shares, but its standard scheduling and bandwidth resources remain shared. **CUDA time-slicing** creates additional schedulable accesses to the same GPU; those replicas do not create proportional compute capacity or memory isolation. (docs.nvidia.com) (docs.nvidia.com) (developer.nvidia.com) (github.com) (github.com)

The answer depends on two gates before utilization: the tenant trust boundary and whether every model plus its peak key-value (KV) cache fits its assigned memory. For illustration, an **8-billion-parameter** model stored in 16-bit values needs about **16 billion bytes for weights alone**. That arithmetic excludes framework allocations, activations, workspaces, fragmentation, and KV cache. An H100 **1g.10gb** instance cannot fit even that weight estimate; an H200 **1g.18gb** instance has little nominal room; a B200 **1g.23gb** instance offers more, but none is a fit guarantee without a measured peak. Profile counts vary by GPU: the cited H100, H200 and B200 tables list up to **seven** of their smallest stated instances, while another supported product has fewer. (github.com) (docs.pytorch.org) (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com)

As of **September 26, 2026**, NVIDIA GPU Operator documentation names **v26.7.1** and its bundled device plugin **v0.20.1**. The plugin still labels its Kubernetes MPS sharing mode experimental and excludes MIG-enabled devices from that plugin mode, although CUDA MPS itself can run above MIG outside the plugin's sharing configuration. The operator's time-slicing guide warns that container attribution in DCGM Exporter is unavailable, while newer DCGM documentation describes opt-in time-sharing or MPS attribution. Treat the actual exporter output as a release-specific acceptance test. (docs.nvidia.com) (docs.nvidia.com) (github.com) (github.com) (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com)

The economic decision in this node-level worksheet is **monthly node cost per SLO-compliant tenant**, not replicas per GPU. Count only tenants whose **95th-percentile latency** and error budget pass simultaneously under representative steady and bursty traffic. Then divide amortized node cost, measured energy, licensing, and measured operator labor by that count. For a full deployment-cost comparison, also allocate storage, networking, facility, and support costs where they differ between candidates, and state the allocation basis. No public result establishes the qualified density of a mixed embedding, reranker, and 8B chat portfolio on a particular cluster. A reproducible solo baseline, neighbor burst, memory-pressure test, and reconfiguration drill must supply the denominator. (sre.google) (grafana.com) (sre.google) (docs.vllm.ai) (grafana.com) (www.spec.org)

Introduction and Background

A Kubernetes scheduler sees integer device resources advertised by a plugin. It does not infer how many tokens per second a serving process can deliver, whether two pods share a memory controller, or whether a neighbor can exhaust framebuffer memory. Kubernetes explicitly treats extended resources as integer quantities that it cannot itself overcommit; the GPU plugin implements sharing by advertising additional resource references. The distinction matters most for low-duty-cycle inference endpoints whose peak requests can coincide. (kubernetes.io) (kubernetes.io) (kubernetes.io) (github.com)

This report compares four deployment modes for enterprise platform engineers and security architects: one pod or trusted service group on an exclusive device, MIG instances, plugin-managed MPS, and plugin-managed CUDA time-slicing. The target decision is a stated tenant count with a latency service-level objective (SLO) and a trust boundary. This is a multi-tenant inference allocation decision. An embedding service, a reranker, and a chat endpoint may all look idle in average GPU utilization while holding different amounts of memory and producing different burst patterns. The relevant measurement is how many independent tenants pass their own SLO together, not how many pods start. Google SRE guidance calls for the measurement method and valid conditions to be explicit, and warns that average traffic can hide bursts. (sre.google) (grafana.com) (sre.google) (sre.google) (grafana.com)

GPU Smith, an independent private AI infrastructure engineering consultancy, describes workload characterization and total-cost comparison as parts of its assessment method. That is an appropriate framing for this choice: models, context lengths, peak concurrency, electrical measurement, and labor assumptions must be recorded before a sharing mode is priced. Its site describes infrastructure specification and validation against written acceptance criteria; the mode comparison below treats those criteria as inputs, not as vendor performance claims. (gpusmith.com) (gpusmith.com)

This is a private-cluster analysis. Virtual GPUs assigned to virtual machines introduce a distinct license and allocation layer. NVIDIA documents vGPU for Compute as licensed through AI Enterprise and describes per-vGPU licensing for assigned VMs; those charges belong in the worksheet only when that virtualization path is used. No general VM license price is inferred for ordinary bare-metal Kubernetes sharing. (docs.nvidia.com) (docs.nvidia.com)

Exclusive GPU Allocation

Capabilities

An ordinary GPU device request provides **exclusive access** according to NVIDIA's GPU Operator guidance. Kubernetes schedules the advertised device as an integer resource; with no sharing configuration, one workload receives the whole assigned GPU resource. This keeps the scheduling model legible and allows a single service to use available framebuffer and compute without a neighboring tenant on the same device. It does not make a node immune to other host or facility failures, and it does not replace ordinary container or network security controls. (kubernetes.io) (kubernetes.io)

The practical baseline is not necessarily one model per physical server. Multiple models from the same trust domain may be served inside one process or serving system with its own batching and admission control. The comparison unit here is a separately scheduled GPU allocation, so an exclusive device can still host a consolidated application if the platform owner controls it. Triton's optimization guidance starts with a baseline measured by Perf Analyzer before changing batching or model instance counts; use the same method before choosing any sharing mode. (docs.nvidia.com)

Adoption, Strengths and Limitations

Exclusive allocation is the reference configuration for the proposed experiment: record latency, error rate, power, memory high-water mark, queue depth, and achieved throughput for each endpoint alone. OpenTelemetry defines HTTP server request duration and active requests as observable metrics; Ray Serve documents request throughput, latency, error rate, tokens, and cache measures for large language model (LLM) serving. Those application metrics matter because device utilization by itself cannot establish whether the service passed its SLO. (opentelemetry.io) (prometheus.io) (opentelemetry.io) (docs.ray.io) (opentelemetry.io) (docs.ray.io) (opentelemetry.io)

Its economic weakness is stranded capacity when a service reserves an entire GPU but uses only a small portion of it over the relevant window. The actual penalty is site-specific: idle time, burst overlap, device purchase price, energy, and operator labor differ. Its strength is an uncomplicated rollback target. If a sharing candidate misses the SLO or isolation gate, move the tenant back to the measured exclusive placement and compare the cost using the same demand trace. KServe documents canary deployment support that can be used for progressive rollout, but the allocation rollback itself must be rehearsed in the local serving stack. (kserve.github.io) (kubernetes.io)

Multi-Instance GPU

Capabilities

MIG divides supported NVIDIA GPUs into predefined GPU instances (GIs) with dedicated compute and memory resources. NVIDIA documents separate paths through the memory system, including the assigned memory capacity and bandwidth. Its concurrency matrix attributes physical partitioning, memory-bandwidth quality of service, and error isolation to MIG. These are architectural guarantees about the supported partition

boundary, not claims about a particular model's p95 latency. A GI can be divided into compute instances, but those compute instances share that GI's memory and engines, so the unit chosen for an isolation policy must be stated precisely. (www.nvidia.com) (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com)

MIG support and geometry depend on GPU and driver. NVIDIA's support table says the feature begins with Ampere; it lists a maximum of seven instances for specific H100, H200, and B200 products, but the RTX PRO 6000 Blackwell Server Edition is listed at four. A profile's name expresses nominal framebuffer capacity, not free memory available to the model after runtime reservation. Every profile-size recommendation therefore starts with a measured model fit and the exact GPU SKU. (docs.nvidia.com) (docs.nvidia.com) (docs.pytorch.org)

Adoption, Strengths and Limitations

MIG is suited to tenants whose policy requires a hardware-defined boundary or whose latency is sensitive to memory-bandwidth contention. It also makes fragmentation visible: a collection of small instances can leave unusable capacity for a larger model, and NVIDIA documents placement fragmentation as instances are created and destroyed. A model that needs a profile larger than the remaining contiguous geometry can force workload migration even when aggregate memory appears sufficient. This is why unused high-bandwidth memory (HBM) must be tracked per partition, not only at node level. (arxiv.org) (arxiv.org) (developer.nvidia.com) (docs.nvidia.com)

Reconfiguration has an availability cost. The GPU Operator's MIG Manager requires user workloads to vacate the GPUs being configured and stops GPU-related pods during changes. On Ampere, enabling MIG mode attempts a reset; on Hopper and newer, enabling the mode does not require a GPU reset, but the no-workload requirement still applies to operator reconfiguration. The operator can change geometry dynamically, yet no official fixed downtime applies to every cluster. Record cordon, drain, configuration, restart, readiness, and rollback minutes in a local drill. (docs.nvidia.com) (canonical.com) (docs.nvidia.com) (docs.nvidia.com)

Multi-Process Service

Capabilities

CUDA MPS lets kernels from different processes use otherwise idle capacity concurrently. Standard MPS provides isolated GPU address spaces for clients and can constrain memory allocation; the Kubernetes device plugin's MPS mode uses a daemon to enforce equal-fraction memory and compute limits per advertised replica. That limit is materially different from a time-slicing replica, which is just a shared access. It is also different from MIG's physical memory paths and bandwidth quality of service. (docs.redhat.com) (docs.nvidia.com) (docs.nvidia.com) (github.com) (github.com) (www.nvidia.com)

The NVIDIA MIG concurrency table describes standard MPS as logical partitioning with shared scheduling hardware, caches, and memory bandwidth. Newer MPS documentation adds optional **static streaming-multiprocessor (SM) partitions** and **MPS v3 cgroup memory partitioning**. Static SM partitioning reserves assigned SMs, so idle capacity in one partition is not automatically borrowed by another client. Driver r610 adds partial isolation for faults triggered by an assigned SM; NVIDIA explicitly says this is not isolation from every GPU or system-level failure. MPS v3 memory partitioning requires Linux cgroup v2, CUDA 13.4 or later,

and a non-MIG device. These features must be configured and tested; they should not be credited to a default plugin deployment by assumption. (docs.nvidia.com) (developer.nvidia.com) (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com)

Adoption, Strengths and Limitations

As of September 2026, the NVIDIA device-plugin README still calls Kubernetes MPS sharing **experimental**. Its sharing mode cannot currently be configured on MIG-enabled devices, although NVIDIA separately documents CUDA MPS on top of MIG. The plugin also says MPS and time-slicing sharing modes are mutually exclusive and that the chosen sharing method applies across GPUs on a node. Cluster design should therefore reserve a node pool for each mode when their policies differ, and test whether the chosen operator version changes those constraints. (kubernetes.io) (github.com) (github.com) (github.com) (github.com)

MPS is a candidate when tenants are within a compatible trust domain, each service's memory needs fit its imposed quota, and concurrency improves qualified density. It is not automatically the cheapest mode: a large chat KV cache may require a bigger equal share than a small embedding endpoint, wasting the rest; static SM partitions may also strand idle compute. Client telemetry needs care because common process accounting can attribute activity to the MPS server. Use per-service latency, errors, queue length, and memory counters alongside DCGM data. (github.com) (developer.nvidia.com) (docs.nvidia.com) (docs.ray.io) (opentelemetry.io)

CUDA Time-Slicing

Capabilities

The NVIDIA device plugin's **time-slicing** configuration creates a specified number of references to each GPU and advertises them as shared accesses. This is Kubernetes capacity bookkeeping, not a proportional partition of the hardware. NVIDIA states that a pod requesting more than one shared GPU does not receive guaranteed proportional compute, and recommends `failRequestsGreaterThanOne=true` to prevent that interpretation. The default of that setting is false, so admission behavior should be configured explicitly. (github.com) (github.com) (github.com)

The plugin and GPU Operator state that time-sliced replicas have no memory or fault isolation from each other. A neighbor can compete for framebuffer and execution time, and increasing the advertised replica count can admit more contenders without increasing physical capacity. Red Hat's accelerator guidance also warns that raising replica counts can increase out-of-memory (OOM) risk. No setting in the replica count alone creates a memory-bandwidth quota. GPU Operator 26.7.1 separately documents optional CUDA-memory limits with an R615 or later driver; that release-specific feature is not an intrinsic time-slicing guarantee and does not imply MIG-like bandwidth or fault isolation. (github.com) (docs.nvidia.com) (docs.redhat.com) (docs.redhat.com) (docs.nvidia.com)

Adoption, Strengths and Limitations

Time-slicing is operationally attractive for trusted, low-duty-cycle jobs that need occasional GPU access and tolerate shared-device contention. (docs.redhat.com) It can be useful for development or batch-like inference, or for production services only after co-tenancy testing shows the requested tail-latency and error budgets still

pass. A scheduler-visible replica is an admission slot; it should be priced by measured qualified tenants, not by configured replicas. Kubernetes' own integer extended-resource semantics do not change that physics. (kubernetes.io) (github.com)

The GPU Operator's time-slicing guide says its DCGM Exporter path cannot associate metrics with containers. Newer DCGM Exporter documentation describes opt-in attribution for supported time-sharing or MPS assignments, while release notes add per-process metrics. The two statements reflect documentation and version differences. A platform team should record exporter, DCGM, driver, plugin and operator versions, enable the intended mapping, inspect `/metrics`, and rely on application instrumentation if tenant attribution is absent. The operator also does not automatically watch ConfigMap edits; its guide says the device-plugin pods must be restarted to apply a changed time-slicing map. (docs.nvidia.com) (docs.nvidia.com) (opentelemetry.io) (prometheus.io)

Feature Comparison

Table 1 separates scheduler-visible allocation from enforcement and the operational consequence. "Documented" refers to the cited vendor behavior; actual fault propagation must still be observed in the proposed test.

Mode	Scheduler advertises	Memory and SM limit	Bandwidth and fault boundary	Main operational cost
Exclusive	One ordinary GPU resource per assigned device. (kubernetes.io)	Whole allocated device; service admission controls its own demand.	No co-tenant on the assigned GPU under this allocation.	Potential idle capacity; simplest rollback baseline.
MIG	Separate GI or profile resources exposed by the device plugin. (docs.redhat.com)	Profile-defined compute and HBM; CI subdivision shares the parent GI's memory. (developer.nvidia.com)	Physical paths, memory QoS, and documented error isolation. (docs.redhat.com)	Fixed geometry, profile fragmentation, drain and reconfiguration. (developer.nvidia.com) (canonical.com)
Plugin-managed MPS	Equal shared accesses to an underlying GPU. (github.com)	Daemon enforces equal-fraction memory and compute; newer standalone MPS controls are configuration dependent. (github.com) (developer.nvidia.com)	Standard mode shares bandwidth and scheduling; optional static SM mode has only partial fault isolation. (developer.nvidia.com) (developer.nvidia.com)	Experimental plugin support, node-wide mode, quota mismatch. (github.com) (github.com)
Time-slicing	Configured replica references per physical GPU. (github.com)	No limit created by replicas; separate release-specific memory control may be available. (docs.nvidia.com)	No inherent memory or fault isolation and no proportional compute guarantee. (docs.redhat.com) (github.com)	Neighbor testing and per-tenant attribution are essential.

The matrix makes the security decision early. (learn.microsoft.com) If the policy requires MIG's documented physical boundary, a cheaper measured MPS or time-slicing result does not satisfy the policy. Conversely, a trusted internal batch service may accept time-slicing if the measured service objective passes. This is a policy filter followed by an empirical density comparison, not one composite isolation score. NVIDIA's concurrency matrix supports the architectural distinction; Google SRE guidance supports stating the SLO conditions explicitly. (learn.microsoft.com) (sre.google) (grafana.com) (sre.google)

Table 2 is a model-fit crosswalk. Its memory arithmetic is deliberately conservative only for raw 16-bit weights: an 8B parameter count times two bytes is about 16 billion bytes. (github.com) (docs.pytorch.org) The table does not certify an engine, quantization format, context length, or KV cache fit.

Illustrative device/profile	Nominal HBM	Weight-only 8B/16-bit comparison	Required acceptance measurement
H100 `1g.10gb`	10 GB; up to seven such instances. (docs.nvidia.com)	Below the approximate 16 billion-byte weight floor. (github.com) (docs.pytorch.org)	Reject this unquantized placement before load testing; test another profile or representation.
H200 `1g.18gb`	18 GB; up to seven such instances. (docs.nvidia.com)	About 2 GB nominal headroom before all non-weight allocations.	Measure peak allocated and reserved memory, KV cache, and error-free concurrency. (docs.pytorch.org) (docs.vllm.ai) (docs.pytorch.org)
B200 `1g.23gb`	23 GB; up to seven such instances. (docs.nvidia.com)	About 7 GB nominal headroom before runtime and cache.	Repeat the same test with actual profile definitions and engine version. (docs.vllm.ai)
H100 `3g.40gb`	40 GB; up to two such instances. (docs.nvidia.com)	More room for runtime and KV cache, with fewer schedulable partitions.	Compare qualified tenant density and unused HBM, not profile count alone. (sre.google)

A successful model load is only the first fit check. (docs.pytorch.org) PyTorch notes that its caching allocator can make unused reserved memory appear occupied in `nvidia-smi`; it exposes separate peak allocated and peak reserved counters. Record both, then compare with device-level telemetry. For the chat endpoint, vary context length, output length, and concurrent sequences because KV cache changes with the request mix. Meta's reference implementation states that cache is preallocated according to maximum sequence length and batch size. (docs.pytorch.org) (docs.pytorch.org) (docs.vllm.ai) (docs.pytorch.org) (github.com)

Performance and Benchmarks

There is no universal inference throughput multiplier for MIG, MPS, or time-slicing. A published research evaluation of a hybrid MIG and MPS approach used multiple A100 GPUs and 11 deep neural network workloads with different SLOs; that is useful evidence that profile fragmentation and workload slack matter, but it is not a benchmark for this proposed embedding, reranker, and 8B chat mix. A newer research paper likewise reports behavior that changes with memory contention. Treat such work as test design input, not as a density number to transfer into a procurement model. (arxiv.org) (arxiv.org) (developer.nvidia.com) (arxiv.org)

Step 1: freeze the inventory. For each tenant, record model hash, weights and precision, serving engine and container image, driver, CUDA version, GPU SKU, profile definition, peak context and output lengths, maximum simultaneous sequences, expected duty cycle, trust boundary, and target p95 and error budget. Record a target request trace rather than a single average requests-per-second value. Google SRE notes that averages can hide instantaneous peaks; OpenTelemetry's active-request metric and Ray's serving metrics provide separate views of arrival pressure and in-flight work. (sre.google) (grafana.com) (opentelemetry.io) (docs.ray.io) (docs.ray.io)

Step 2: establish solo baselines. Run each endpoint alone on the selected full GPU and on each candidate MIG profile that passes the fit check. Capture p50, p95, and p99 end-to-end latency, errors, request queue, tokens per second where relevant, peak allocated and reserved GPU memory, wall power, and achieved arrival rate. For token streaming, report time to first token and time per output token separately. MLPerf's server methodology illustrates why an arrival process and latency constraints must both be specified, (docs.mlcommons.org) although its published Llama 2 70B limits are not this report's SLO. (mlcommons.org) (mlcommons.org) (docs.pytorch.org) (docs.vllm.ai) (www.spec.org)

Step 3: add realistic co-tenancy. Replay steady traffic and then a bursty neighbor with the same synchronized traces for each mode. vLLM's benchmarking CLI documents request-rate, burstiness and concurrency controls, as well as separate probe requests for observing interference. Triton Perf Analyzer documents timed request replay. If an HTTP load generator is used, Grafana k6's arrival-rate executors can hold or ramp request rates, while its `dropped_iterations` metric reveals a generator that could not supply the requested load. These are measurement tools, not evidence that one sharing mode wins. (docs.vllm.ai) (grafana.com) (docs.vllm.ai) (grafana.com) (docs.nvidia.com) (grafana.com) (grafana.com)

Step 4: inject bounded faults and reconfigure. In a controlled test cluster, drive one tenant toward its configured memory limit, exercise application crashes and CUDA allocation failures, and observe neighboring request errors and recovery time. For MPS, test unsynchronized client termination on a disposable test server; record whether other clients fail or hang and rehearse restarting the affected server and all its clients. (NVIDIA MPS guide) The observed behavior should be recorded per GPU, driver, runtime and sharing configuration; do not generalize a single trial into an architecture guarantee. Before production placement, rehearse `terminate_client <server PID> <client PID>` in Legacy MPS v2 or `client terminate <client PID>` in MPS v3 and verify its effect on the server and other clients. (NVIDIA MPS guide) Distinguish host cgroup OOM counters from GPU-memory allocation errors because Linux `memory.events` describes cgroup OOM kills. Then drain and change MIG geometry or time-slicing settings, timing each step until all endpoints pass readiness and their SLO again. (docs.kernel.org) (www.nvidia.com)

Step 5: make telemetry auditable. DCGM can expose GPU and MIG-device metrics and labels MIG instances with profile and instance identifiers, but individual profiling fields may be unsupported on a given device. DCGM's own guidance says profiling metrics are interval averages, not request traces. Use an application latency histogram and compute a pooled p95 from histogram buckets; Prometheus explicitly warns that averaging per-replica quantiles is statistically invalid. (prometheus.io) For shared modes, inspect whether process or container attribution is actually present before using telemetry for tenant chargeback. (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com) (prometheus.io) (opentelemetry.io) (prometheus.io)

Data Analysis and Evidence

Define **qualified tenant density**, `Q`, as the number of tenants on a shared GPU that meet their own p95 latency and error-budget thresholds **simultaneously** over the specified trace. The **consolidation ratio** is `Q_shared / Q_exclusive` under the same trace and node accounting. Monthly **node-level cost per qualified tenant** is `(amortized node + measured node energy + licensing + measured operator labor) / Q`. Record the energy-meter boundary; add allocated storage, networking, facility, and support costs when comparing full deployment cost, with the allocation basis stated. A failed or unmeasured tenant contributes zero to `Q`; a configured replica contributes nothing by itself. Service-level thresholds should be pass/fail criteria, as Grafana k6 documents, with validity conditions stated as Google SRE recommends. (grafana.com) (sre.google) (sre.google) (grafana.com) (sre.google)

Table 3 is the worksheet to publish with any private-cluster result. It deliberately leaves unmeasured fields blank. The nominal profile counts are sourced facts; the cost and SLO outputs are local observations.

Field	Exclusive	MIG	MPS	Time-slicing
GPU, driver, operator, plugin, engine, model hashes	Record exact versions. (docs.nvidia.com)	Record exact versions and GI geometry. (www.nvidia.com)	Record daemon mode, quotas and whether static SM or v3 controls are enabled. (developer.nvidia.com)	Record replica count, `failRequestsGreaterThanOne`, and memory-control configuration. (github.com)
Advertised accesses / partition slots	Physical device count.	Profile count, for example up to seven H100 `1g.10gb` instances.	Configured shared accesses; equal fractions in plugin mode. (github.com)	Configured replicas, not compute shares. (github.com)
SLO-qualified tenants `Q`	Measure.	Measure.	Measure.	Measure.
Unused HBM per partition	Measure peak headroom.	Sum profile capacity minus peak used capacity, per GI. (docs.pytorch.org)	Measure quota headroom and whole-device idle memory.	Measure full-device peak and contention; replicas impose no intrinsic partition. (learn.microsoft.com)
Reconfiguration and recovery minutes	Measure redeploy.	Measure drain, geometry change and readiness. (canonical.com)	Measure daemon/config change and readiness.	Measure ConfigMap update, plugin restart and readiness.
Monthly node-level cost per qualified tenant	Node + measured node energy + license + labor, divided by `Q`.	Same scoped formula; measure reconfiguration labor.	Same scoped formula; measure daemon and quota labor.	Same scoped formula; measure plugin and fault-response labor.

This worksheet prevents a common false result: dividing the monthly GPU bill by ten time-slicing replicas when only three tenants meet latency and error targets together. Ten replicas may be an admission configuration, but the NVIDIA plugin explicitly says they are references to the same GPU and do not guarantee proportional compute. The numerical example is a counting illustration, not a measured cluster result. The denominator must come from the experiment. (github.com) (github.com)

Energy should be measured at a stated electrical boundary. SPEC's server-power methodology measures at AC input alongside performance; a data-center allocation may additionally use facility power usage effectiveness, defined by the US Department of Energy as a facility-to-IT-energy ratio. If only node power is measured, say so and do not silently call it facility energy. A GPU thermal design power rating is a hardware limit, not the energy a serving node consumed over the trace. (www.spec.org) (www.energy.gov) (www.nvidia.com)

Licensing is conditional. For a bare-metal Kubernetes comparison, enter only licenses actually required by the chosen software and support stack. If virtual machines use NVIDIA vGPU for Compute, its AI Enterprise licensing and per-vGPU assignment rules become separate cost lines. NVIDIA also documents SKU-specific included subscriptions, so procurement should verify the exact SKU and contract rather than apply one universal rate. Operator labor belongs in the same numerator: measure minutes spent on geometry changes, telemetry repair, quota tuning, and incident rehearsal, then multiply by the organization's loaded labor rate. (docs.nvidia.com)

Implications and Future Directions

A defensible deployment policy begins with **isolation**. Use an exclusive GPU when a whole device is required for model fit, maximum burst headroom, or a straightforward dedicated allocation. Use MIG when supported hardware's documented physical boundary and bandwidth quality of service match the tenant policy, and the model plus peak cache fits an available profile. Consider MPS when sharing inside a compatible trust domain benefits from enforced compute and memory shares and the experimental plugin status is acceptable to the platform's release process. Use time-slicing for compatible, lower-duty-cycle tenants only after burst and fault tests show that their SLOs survive co-tenancy. (docs.redhat.com) (github.com) (github.com) (docs.redhat.com)

The current software stack complicates simple labels. Newer MPS static SM and cgroup memory controls can narrow some gaps, but their prerequisites and partial fault semantics must be verified; they do not turn default MPS into MIG. Newer DCGM capabilities may improve process attribution for shared devices, but the GPU Operator time-slicing guide still warns about container mapping. NVIDIA's 26.7.1 operator also adds optional CUDA-memory controls requiring an R615 or newer driver. Decision records should therefore name the actual feature and version, not just "MPS" or "time-slicing." (developer.nvidia.com) (developer.nvidia.com)

The rollout checklist is practical:

- **Freeze inputs:** preserve request traces, SLO windows, model and container hashes, and exact GPU/profile definitions. (sre.google) (grafana.com) (docs.vllm.ai) (grafana.com)
- **Admit only fits:** verify peak allocated and reserved HBM at maximum intended sequence length and concurrency. (docs.pytorch.org) (docs.vllm.ai) (docs.pytorch.org)
- **Test interference:** measure solo, steady co-tenant, bursty neighbor, and bounded fault phases with the same arrival process. (docs.vllm.ai) (grafana.com) (grafana.com)
- **Verify observability:** inspect actual DCGM entity and workload labels, supported fields, and application latency histograms. (prometheus.io) (opentelemetry.io)

- **Rehearse rollback:** time draining and MIG geometry changes or plugin restarts, then canary traffic back to the previous placement. KServe documents progressive canary rollouts and rollback support. (www.nvidia.com) (kserve.github.io)
- **Reprice the result:** divide the stated-scope monthly cost by qualified tenants and keep unused HBM and reconfiguration minutes visible beside the ratio. (www.spec.org) (www.energy.gov)

Future comparisons should publish the trace, model mix, container and driver versions, profile definitions, SLO thresholds, observed fault behavior, and raw result tables. That would make a density number portable only where hardware and workload match. The cited research on mixed MIG and MPS highlights fragmentation and workload-specific slack, while benchmark tooling already supports reproducible request pacing and probes. Neither substitutes for a site-specific acceptance test. (arxiv.org) (docs.vllm.ai) (grafana.com)

Frequently Asked Questions (FAQs)

Is MIG always faster than MPS or time-slicing?

No universal ordering follows from the documents. MIG offers documented partition boundaries and bandwidth quality of service; MPS can use idle capacity across processes; time-slicing grants shared access. Throughput and p95 depend on model size, memory pressure, batch behavior, and request timing. Measure qualified density under the same trace. (learn.microsoft.com) (developer.nvidia.com) (docs.vllm.ai) (grafana.com)

Can time-slicing replicas be treated as fractional GPUs?

They are scheduler-visible shared accesses. NVIDIA explicitly says multiple requested replicas do not guarantee proportional compute, and the replicas do not isolate GPU memory or faults. Use the replica count to control admission, then measure p95 and errors to establish actual capacity. (github.com) (docs.redhat.com) (grafana.com)

Does MPS guarantee tenant fault isolation?

Standard MPS is logical sharing. NVIDIA's newer static SM partitioning documentation reports partial isolation for some SM-triggered faults from driver r610, while stating that not every GPU or system failure is isolated. The exact enabled mode and observed behavior should be part of acceptance testing. (developer.nvidia.com) (developer.nvidia.com)

How should a team compare costs when demand is bursty?

Run synchronized traces with steady and burst phases, count only tenants meeting both latency and error budgets, and divide measured monthly node, energy, licensing, and labor costs by that count. State that this is a node-level comparison unless storage, networking, facility, and support allocations are also included on a documented basis. Report the arrival trace and any dropped load-generator iterations so a lower observed rate cannot masquerade as efficiency. (sre.google) (grafana.com) (grafana.com) (www.spec.org)

Conclusion

MIG, MPS, and time-slicing solve different allocation problems. MIG supplies the documented physical partition and memory-bandwidth boundary on supported GPUs, with fixed profile sizes and reconfiguration work. MPS offers concurrent execution and enforceable shares in the plugin's documented mode, with important version-specific options and an experimental Kubernetes status. Time-slicing exposes additional scheduling slots without creating proportional compute or inherent memory and fault isolation. An exclusive device remains the clearest baseline and the fallback for a tenant whose model or policy does not fit a sharing mode.

For enterprise inference, the decision should be made in this order: enforce the trust boundary, reject profiles that cannot hold the measured model and peak cache, reproduce co-tenant latency and fault behavior, and then calculate cost per qualified tenant. Publish the denominator, not just the configured replica count. A candidate with lower nominal GPU spend but more SLO failures, stranded HBM, or operator intervention may be more expensive per usable tenant. The recommended mode is therefore conditional on the actual GPU, software versions, model portfolio, trace, and acceptance criteria, all of which can be recorded and retested when the deployment changes.

Source links

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

1. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/introduction.html
2. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/concepts.html
3. https://developer.nvidia.com/blog/?p=49216
4. https://github.com/NVIDIA/k8s-device-plugin
5. https://github.com/meta-llama/llama3
6. https://docs.pytorch.org/docs/main/tensor_attributes.html
7. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/supported-mig-profiles.html
8. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/26.7/getting-started.html
9. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/release-notes.html
10. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/deployment-considerations.html
11. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/26.7/gpu-sharing.html
12. https://docs.nvidia.com/datacenter/dcgm/latest/reference/command-line-reference/dcgm-exporter.html
13. https://sre.google/sre-book/service-level-objectives/
14. https://grafana.com/docs/k6/latest/using-k6/thresholds/
15. https://docs.vllm.ai/en/latest/benchmarking/cli/
16. https://grafana.com/docs/k6/latest/using-k6/scenarios/concepts/arrival-rate-vu-allocation/
17. https://www.spec.org/power_ssj2008/
18. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/
19. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/

20. https://gpusmith.com/articles/en/mig-vs-mps-time-slicing-gpu-passthrough
21. https://grafana.com/docs/k6/latest/testing-guides/api-load-testing/
22. https://gpusmith.com/
23. https://docs.nvidia.com/ai-enterprise/release-8/latest/infra-software/vgpu/overview.html
24. https://docs.nvidia.com/ai-enterprise/release-7/latest/infra-software/vgpu/licensing.html
25. https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/optimization.html
26. https://opentelemetry.io/docs/specs/semconv/http/http-metrics/
27. https://prometheus.io/docs/practices/histograms/
28. https://docs.ray.io/en/latest/serve/llm/user-guides/observability.html
29. https://kserve.github.io/website/docs/reference/crd-api
30. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
31. https://www.nvidia.com/en-us/technologies/multi-instance-gpu/
32. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/supported-gpus.html
33. https://docs.pytorch.org/docs/2.14/notes/cuda.html
34. https://arxiv.org/abs/2409.14447
35. https://developer.nvidia.com/blog/?p=21816
36. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/26.7/gpu-operator-mig.html
37. https://canonical.com/microk8s/docs/addon-gpu
38. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/getting-started-with-mig.html
39. https://docs.redhat.com/en/documentation/red_hat_openshift_service_on_aws_classic_architecture/4/html/architecture/nvidia-gpu-architecture-overview
40. https://docs.nvidia.com/deploy/mps/when-to-use-mps.html
41. https://developer.nvidia.com/blog/nvidia-cuda-13-1-powers-next-gen-gpu-programming-with-nvidia-cuda-tile-and-performance-gains
42. https://docs.nvidia.com/deploy/mps/common-tasks.html
43. https://docs.nvidia.com/deploy/mps/mpsv3-memory-partitioning.html
44. https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/
45. https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/3.4/html/working_with_accelerators/about-gpu-time-slicing_accelerators
46. https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/2.25/html-single/working_with_accelerators/index
47. https://docs.nvidia.com/datacenter/dcgm/latest/release-notes/dcgm-exporter.html
48. https://developer.nvidia.com/blog/?p=50380
49. https://developer.nvidia.com/blog/?p=121255
50. https://learn.microsoft.com/en-us/azure/aks/concepts-gpu-partitioning
51. https://docs.pytorch.org/docs/2.14/generated/torch.cuda.memory.max_memory_allocated.html
52. https://docs.vllm.ai/en/latest/design/metrics/
53. https://docs.pytorch.org/docs/2.14/generated/torch.cuda.memory.max_memory_reserved.html

54. https://docs.pytorch.org/docs/2.14/torch_cuda_memory.html
55. https://arxiv.org/abs/2604.22430
56. https://docs.mlcommons.org/inference/submission/
57. https://mlcommons.org/2024/03/mlperf-llama2-70b/
58. https://docs.nvidia.com/deeplearning/triton-inference-server/archives/triton-inference-server-2670/user-guide/docs/perf_analyzer/docs/inference_load_modes.html
59. https://docs.kernel.org/7.1/admin-guide/cgroup-v2.html
60. https://docs.nvidia.com/datacenter/dcgm/latest/reference/dcgm-exporter-metrics.html
61. https://docs.nvidia.com/datacenter/dcgm/latest/learn/modules/profiling.html
62. https://www.energy.gov/cmei/femp/cooling-water-efficiency-opportunities-federal-data-centers
63. https://www.nvidia.com/en-gb/data-center/h200/?ncid=so-link-516563-vt11
64. https://docs.nvidia.com/ai-enterprise/planning-resource/licensing-guide/latest/licensing.html
65. https://kserve.github.io/website/blog/kserve-0.20-release

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