



GPU Server BIOS and OS Baseline: IOMMU, ACS, Persistence

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 · 2026-10-29 · gpu server bios settings · iommu pass-through gpu peer to peer · acs disable gpu p2p nccl · nvidia persistence daemon vs persistence mode · above 4g decoding resizable bar gpu server · gpudirect rdma iommu disable

Original article: <https://gpusmith.com/articles/en/gpu-server-bios-and-os-tuning-baseline>



In this article

1. Why a baseline matters
 2. The baseline at a glance
 3. IOMMU: off, pass-through or translated
 4. ACS and PCIe peer-to-peer
 5. Above 4G decoding and BAR size
 6. C-states, power profile and NUMA
 7. Persistence daemon, not persistence mode
 8. Fabric Manager, driver ordering and the software stack
 9. Kernel boot parameters
 10. Verification script
 11. Drift control
-

Why a baseline matters

Many GPU server incidents are settings nobody wrote down: an IOMMU in translated mode, PCIe Access Control Services (ACS) redirecting peer traffic through the root complex, a missing large-BAR window, or a driver tearing down GPU state between jobs. None stop a node from booting. They surface later as slow all-reduce, flaky GPUDirect RDMA, or a first CUDA call that takes seconds.

NVIDIA's NCCL troubleshooting guide puts it bluntly: when P2P access fails or is slow, the causes include "missing components, a bad configuration of a virtual machine or a container, or some BIOS settings" ([NCCL 2.32.3 GPU troubleshooting](#)). It also warns that symptoms "can look like application or framework bugs rather than a clean NCCL error." Hence a baseline: a versioned list of settings, each with a reason and a verification command, applied before a node enters a scheduler pool and audited afterwards. This one is assembled from vendor documentation; where NVIDIA and AMD differ, or only an OEM can answer, it says so. For reactive diagnosis of a link that is already misbehaving, see [GPU PCIe link troubleshooting](#).

The baseline at a glance

IOMMU and ACS answers depend on deployment shape, so the table separates bare-metal hosts (PCIe P2P, GPUDirect RDMA) from hosts passing GPUs to VMs.

Setting	Bare-metal baseline	VM / passthrough host	Symptom if wrong	Verify with
IOMMU (VT-d / AMD-Vi) in BIOS	Enabled	Enabled	>255 logical CPUs hidden (if disabled on large hosts); see below	<code>dmesg grep -i iommu</code>
Kernel IOMMU mode	<code>iommu=pt</code> (plus <code>intel_iommu=on</code> on Intel)	Translated (default), VFIO	Bare-metal P2P corruption or slow/failed RDMA	<code>/sys/kernel/iommu_groups/*/type</code>

Setting	Bare-metal baseline	VM / passthrough host	Symptom if wrong	Verify with
PCIe ACS	Disabled on bridges in the GPU/NIC path	Enabled (required)	Low P2P bandwidth or hangs	<code>lspci -vvv grep ACSctl</code>
Above 4G decoding	Enabled	Enabled	GPUs missing or BAR allocation failure	<code>nvidia-smi -q BAR1 total, dmesg</code>
CPU C-states	Shallow only (C2 disabled)	Shallow only	Latency spikes on networking paths	<code>cpupower idle-info</code>
PCIe ASPM	Disabled	Disabled	Link power-state latency	<code>/proc/cmdline, lspci LnkCtl</code>
NUMA per socket	NPS1 default (AMD), test others	NPS1	Remote-memory traffic	<code>numactl -H</code>
NUMA auto-balancing	Disabled	Disabled	Page-migration overhead	<code>/proc/sys/kernel/numa_balancing</code>
Driver persistence	<code>nvidia-persistenced</code> running	Same inside guest	Seconds-long CUDA start per job	<code>nvidia-smi -q Persistence Mode</code>
Fabric Manager (NVSwitch HGX)	Enabled, version-compatible	Per vendor	<code>cudaErrorSystemNotReady</code>	<code>systemctl status nvidia-fabricmanager</code>

The C-state, ASPM, NUMA and kernel-parameter rows are drawn from AMD's Instinct system-acceptance documentation ([BIOS settings](#), [kernel parameters](#), [OS tuning](#)). AMD notes that settings and locations "may vary by hardware and BIOS vendor" and that on Intel hosts "some settings might not apply or may be unavailable." NVIDIA publishes no equivalent platform-neutral table: its NVIDIA-Certified Systems configuration guide (updated 2026-09-30) covers UEFI, TPM 2.0 and memory population but no BIOS tuning values ([NVIDIA](#)). Take menu names and exact values from your OEM.

IOMMU: off, pass-through or translated

What NVIDIA documents. The GPUDirect RDMA guide explains that RDMA between a GPU and a third-party device relies on physical addresses being the same from every device's point of view, so IOMMUs "must be disabled or configured for pass-through translation for GPUDirect RDMA to work" ([CUDA GPUDirect RDMA, v13.4, updated 2026-10-12](#)). The NCCL guide is narrower and stronger for P2P: "On Linux bare-metal systems, however, CUDA and the NVIDIA driver stack do not support IOMMU-enabled PCIe peer-to-peer memory transfer," and the fix is to "avoid translated IOMMU mode for GPU PCIe P2P" ([NCCL troubleshooting](#)). It cites the CUDA Programming Guide's statement that the IOMMU must be disabled on bare-metal Linux "to prevent silent device memory corruption."

Disabled versus pass-through. These are not interchangeable in every NVIDIA document. One NVIDIA Maxine NIM page requires `iommu=pt` for its unpinned multi-GPU mode and states that "an IOMMU that is disabled outright is not equivalent to passthrough," because "cross-device shared-memory reads can still

return corrupted data" ([NVIDIA NIM multi-GPU systems](#)). That is application-specific, but points the same way as AMD.

What AMD documents. AMD lists `iommu=pt` as a required kernel parameter, describing it as allowing the adapter "to bypass DMA translation to memory, thereby improving performance," with `intel_iommu=on` needed on Intel hosts ([AMD kernel parameters](#)). Its MI300X health check fails a node unless `/proc/cmdline` "Contains `pci=realloc=off, amd_iommu=on` or `intel_iommu=on`, and `iommu=pt`" ([AMD Instinct MI300X](#)).

A second reason not to disable the IOMMU in firmware: AMD notes that on systems with 256 or more logical cores, disabling the IOMMU in BIOS "can limit the available logical cores to 255," because the kernel disables x2APIC and falls back to APIC ([AMD OS tuning](#)).

Worked example: a dual-socket host with two 96-core CPUs and SMT enabled has $2 \times 96 \times 2 = 384$ logical CPUs. With the IOMMU disabled in firmware, the AMD-documented fallback exposes at most 255, so $384 - 255 = 129$ logical CPUs (about 34%) would disappear from the OS. With SMT off the host has 192 and the limit does not bite, so the problem can appear only after an SMT policy change.

Baseline. For bare-metal GPU nodes: IOMMU enabled in BIOS, `iommu=pt` on the kernel command line (plus `intel_iommu=on` on Intel). The Linux kernel documents `iommu=pt` as passthrough mode by default, "(Equivalent to `iommu.passthrough=1`)" ([Linux kernel parameters](#)). This satisfies NVIDIA's pass-through option and AMD's requirement, and keeps x2APIC.

Verify the mode, not just the cmdline. At boot the kernel logs `iommu: Default domain type: followed by Passthrough or Translated`, with "(set via kernel command line)" when forced ([linux/drivers/iommu/iommu.c](#)). Per group, `/sys/kernel/iommu_groups/<grp_id>/type` reads identity when DMA "are not translated by the iommu," and DMA or DMA-FQ when they are ([Linux ABI testing files](#)). NCCL lists `iommu: Default domain type: Translated and DMAR: IOMMU enabled` as outputs that are "concerning for Linux bare-metal PCIe P2P."

VM hosts are the exception. VFIO hosts need translated domains for assigned devices; keep them in a separate host class.

ACS and PCIe peer-to-peer

ACS lets a switch or root port force peer-to-peer requests upstream instead of routing them directly, turning a switch-local GPU path into a root-complex round trip. NCCL describes the result as "a significant performance reduction or even a hang" ([NCCL troubleshooting](#)).

The GPUDirect RDMA guide ranks the paths: when only PCIe switches are on the path it "is optimal and yields the best performance"; a path through a single CPU/IOH "works, but yields worse performance"; a path across a QPI/HT socket link "may be extremely performance-limited or even not work reliably" ([GPUDirect RDMA](#)). ACS is how a switch-only path silently becomes a CPU path.

Check. NCCL's procedure is `sudo lspci -vvv | grep ACSctl`; "If lines show 'SrcValid+', then ACS might be enabled." `lspci | grep PLX` identifies Broadcom PLX switches.

Disable. NCCL says some systems disable ACS in BIOS by disabling IO virtualization or VT-d. That conflicts with the IOMMU baseline, so prefer a BIOS with a separate ACS control (AMD lists ACS: Disabled separately for non-virtualized hosts). For PLX switches NCCL documents an OS method that "needs to be done again after each reboot":

```
sudo setpci -s 03:00.0 ECAP_ACS+0x6.w=0000 # repeat per bridge BDF
```

Run it from a boot-time unit and re-check it in the drift audit.

VMs. NCCL is explicit: "Virtual machines require ACS to function, hence disabling ACS is not an option." For maximum performance inside VMs it says ATS needs to be enabled in network adapters. AMD's virtualization table likewise sets ACS and IOMMU to Enabled.

Do not trust a single matrix. A positive `nvidia-smi topo -p2p p` result "is useful, but it is not a complete system sanity check" for the IOMMU issue, per NCCL. Measure with [nvbandwidth](#), which NCCL recommends over the CUDA sample, and run `simpleP2P`, which validates copied data.

Above 4G decoding and BAR size

Each NVIDIA data-center GPU exposes a BAR1 aperture that "is used to map the FB (device memory) so that it can be directly accessed by the CPU or by 3rd party devices (peer-to-peer on the PCIe bus)" ([nvidia-smi documentation](#)). The GPUDirect RDMA guide notes that on some GPUs "BAR1 size is set to 16GB or larger," and that "Large BARs can pose a problem for the BIOS, especially on older motherboards," where boot "can stop during the early POST phase" ([GPUDirect RDMA](#)).

The arithmetic shows why 64-bit MMIO is mandatory. The legacy 32-bit address space is 2^{32} bytes = 4 GiB in total, shared with everything else below 4 GiB. Eight GPUs at the documented 16 GB lower bound already need $8 \times 16 = 128$ GB of BAR1 space, plus NIC and NVMe BARs, and plus VF BARs if SR-IOV is enabled. That cannot fit below 4 GiB, so "Above 4G decoding" (64-bit MMIO) must be on. AMD lists it as required for MI3xx systems: "GPU large BAR support" ([AMD BIOS settings](#)).

Resizable BAR is less uniform. AMD enables it for virtualization and otherwise says to "defer to the system manufacturer for suggested setting." No NVIDIA data-center document we found gives a general recommendation; take it from the OEM. With SR-IOV disabled, AMD requires `pci=realloc=off`.

Verify. Compare `nvidia-smi -q "BAR1 Memory Usage → Total"` against the value documented for the GPU and OEM platform. Check `dmesg` for BAR assignment failures.

C-states, power profile and NUMA

Vendor-neutral guidance is thin. The most complete published table is AMD's, for MI3xx on EPYC hosts ([AMD BIOS settings](#)):

AMD setting	Recommended value	AMD's stated reason
Power Management	Maximum Performance	"Optimizes power delivery for GPU workloads."
Global C-state control	Enabled	"do not disable this menu item"
DF C-states	Disabled	Disable "to reduce latency unless a power reduction is needed"
ASPM	Disable	(none given)
SMT control	Disable	Set Auto if the workload "is not compute-bound" or benefits from SMT
NUMA nodes per socket	NPS1	"Some workloads may have performance improvements with values greater than 1"; NPS0 not allowed
PCIe ten bit tag, ARI	Enabled	Required

In the OS, AMD recommends disabling the C2 state with `cpupower idle-set -d 2`, noting it is "particularly important for high-performance networking," or using `processor.max_cstate=0`, which it calls the preferred method because `cpupower` settings must be reapplied after every reboot ([AMD OS tuning](#)). It also lists `pcie_aspm=off` as "highly recommended when prioritizing performance" and `numa_balancing=disable` as optional, warning that kernel updates can re-enable NUMA balancing ([AMD kernel parameters](#)).

On Intel or Grace hosts, start from the OEM performance profile and record deviations. Leave AMD's riskier optional parameters (`mitigations=off`, `nowatchdog`, `norandmaps`) out of a default baseline: AMD itself warns they reduce security or remove "a key safeguard against system lockups."

Persistence daemon, not persistence mode

When no client has the GPU open, the NVIDIA kernel driver tears down GPU state, and repeated initialization "leads to long load times for each CUDA job, on the order of seconds" ([NVIDIA driver persistence: background](#)).

NVIDIA offers two fixes. Legacy persistence mode (`nvidia-smi -pm 1`) is a driver flag; `nvidia-smi` documents that it "does not persist across reboots" and defaults to Disabled after each reboot ([nvidia-smi documentation](#)). The NVIDIA Persistence Daemon is a user-space process that "sleeps with the device files open." NVIDIA's position: "The daemon approach provides a more elegant and robust solution to this problem than persistence mode," and "Eventually, the legacy persistence mode will be deprecated and removed in favor of the NVIDIA Persistence Daemon," while "NVIDIA will support both solutions for the near future" ([Persistence Daemon, updated 2026-09-09](#)). As of 2026-10-28 that page still describes removal as future.

Baseline. Run `nvidia-persistenced` as a service under an unprivileged user (`nvidia-persistenced --user <user>`); the driver ships sample init scripts but does not install them. `nvidia-smi` uses the daemon when it is running and falls back to legacy mode otherwise. Verify that `nvidia-smi -q` shows "Persistence Mode: Enabled" after a cold boot, not just after someone ran a command.

Fabric Manager, driver ordering and the software stack

On NVSwitch-based HGX and DGX systems, NVLink is not usable until Fabric Manager has configured the switches. The Fabric Manager guide (version 2.6, 2026-09-21) states that if an application starts before FM finishes, or FM fails, "CUDA initialization will fail with the `cudaErrorSystemNotReady` error"; on H100-generation and later systems, GPUs must also register with the NVLink fabric before CUDA initialization proceeds ([Fabric Manager user guide](#)). Fourth-generation NVSwitch systems (HGX B200/B300/B100) "require an additional NVLSM service." From driver branch 620, FM has its own semantic version (from 2.0.x) and is installed independently of the driver (no R620 driver was tagged on NVIDIA's `open-gpu-kernel-modules` repository as of 2026-10-28; newest 615.78.08, 2026-10-07), so follow the guide revision matching your branch; on older NVSwitch generations FM checks that the driver-provided `libnvidia-fmdrv` library version matches.

Enable it with `sudo systemctl enable nvidia-fabricmanager` and gate workload start on `nvidia-smi -q` reporting fabric state `Completed` (other documented values: `In Progress`, `Not Started`, `Not supported`).

Pin driver, FM, toolkit and container versions together; the [CUDA driver/toolkit compatibility matrix](#) covers that ordering.

For GPUDirect RDMA, `nvidia-peermem` ships with the driver but "must be loaded on each node boot," unless NCCL's automatic DMA-BUF path makes it unnecessary (NCCL troubleshooting). Pick one path per fleet. Full RDMA qualification is covered in [GPUDirect RDMA qualification](#).

Kernel boot parameters

A defensible bare-metal starting point is `iommu=pt` everywhere, `intel_iommu=on` on Intel CPUs, and, on AMD Instinct nodes, AMD's required `pci=realloc=off` and `pci=bfsort` ([AMD kernel parameters](#)). Add `numa_balancing=disable`, `pcie_aspm=off` and `processor.max_cstate=0` only after an A/B test on your workload.

One nuance: the kernel documents `pcie_aspm=off` as "Don't touch ASPM configuration at all. Leave any configuration done by firmware unchanged" ([Linux kernel parameters](#)). It does not turn off ASPM that firmware enabled, so the BIOS ASPM setting belongs in the baseline too.

Apply with `grubby --update-kernel=ALL --args=...` on RHEL-family systems or `GRUB_CMDLINE_LINUX` plus `update-grub` on Ubuntu, then confirm with `cat /proc/cmdline` after reboot.

Verification script

A baseline is only real if a script can fail a node. A minimal outline:

```
#!/usr/bin/env bash
set -u; fail=0; f(){ echo "FAIL: $*"; fail=1; }

# 1. Kernel command line
grep -qw 'iommu=pt' /proc/cmdline || f "iommu=pt missing"
```

```

# 2. Effective IOMMU mode (bare-metal class only)
for t in /sys/kernel/iommu_groups/*/type; do
  [ "$(cat "$t")" = identity ] || f "$t is $(cat "$t")"
done
dmesg | grep -q 'Default domain type: Translated' && f "translated IOMMU domain"

# 3. ACS on bridges
lspci -vvv 2>/dev/null | grep ACSctl | grep -q 'SrcValid+' && f "ACS SrcValid+ present"

# 4. NUMA balancing and C-states
[ "$(cat /proc/sys/kernel/numa_balancing)" = 0 ] || f "numa_balancing on"
cpupower idle-info | tee /tmp/cstates.txt >/dev/null

# 5. Persistence and fabric
nvidia-smi -q | grep -q 'Persistence Mode *: Enabled' || f "persistence off"
systemctl is-active -q nvidia-persistenced || f "persistenced inactive"
# NVSwitch systems only:
# systemctl is-active -q nvidia-fabricmanager || f "fabricmanager inactive"

# 6. Topology and bandwidth (compare to the golden node)
nvidia-smi topo -m > /tmp/topo.txt
nvidia-smi topo -p2p p >> /tmp/topo.txt
nvbandwidth -p device_to_device -F json > /tmp/nvbw.json
dmesg -T | grep -iE 'AER|NVRM: Xid|BAR' > /tmp/dmesg-flags.txt

exit $fail

```

The group loop assumes every IOMMU group on a bare-metal node should be identity; on mixed hosts, filter to GPU, NIC and switch BDFs. Compare bandwidth against a golden node of the same SKU and BIOS revision, not theoretical peaks. `nvbandwidth` reports per-pair matrices and supports a `--pair` option to isolate one GPU pair ([nvbandwidth README](#)).

Drift control

Settings drift through BIOS updates that reset menus, kernel updates (AMD warns they "can re-enable NUMA balancing"), and non-persistent runtime changes such as `setpci` ACS writes and `cpupower` settings.

- **Store BIOS state as data.** Where the OEM offers export/import tools (AMD notes some do), version the export per SKU and BIOS revision and diff it on every firmware update. AMD: "Always record existing production system settings before making changes."
- **Make runtime settings boot units**, not shell history.
- **Gate scheduling on the verifier.** Run it at boot and on a timer; drain failing nodes. It is one stage of [cluster acceptance testing](#), not a substitute for burn-in.
- **Keep bare-metal and VFIO host classes separate**; their IOMMU and ACS requirements are opposite.
- **Re-confirm AMD-sourced values** against the OEM guide before copying them to Intel or NVIDIA-GPU builds.

GPU Smith can review a baseline like this against a specific OEM platform during commissioning.

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/deeplearning/nccl/user-guide/docs/troubleshooting/gpu_troubleshooting.html
2. <https://instinct.docs.amd.com/projects/system-acceptance/en/latest/common/bios-settings.html>
3. <https://instinct.docs.amd.com/projects/system-acceptance/en/latest/common/kernel-parameters.html>
4. <https://instinct.docs.amd.com/projects/system-acceptance/en/latest/common/os-tuning.html>
5. <https://docs.nvidia.com/certification-programs/latest/nvidia-certified-configuration-guide.html>
6. <https://docs.nvidia.com/cuda/gpudirect-rdma/index.html>
7. <https://docs.nvidia.com/nim/maxine/synthetic-video-detector/latest/advanced-usage/multi-gpu.html>
8. <https://instinct.docs.amd.com/projects/amdgpu-docs/en/latest/system-optimization/mi300x.html>
9. <https://docs.kernel.org/admin-guide/kernel-parameters.html>
10. <https://github.com/torvalds/linux/blob/master/drivers/iommu/iommu.c>
11. <https://docs.kernel.org/admin-guide/abi-testing-files.html>
12. <https://github.com/NVIDIA/nvbandwidth>
13. <https://docs.nvidia.com/deploy/nvidia-smi/index.html>
14. <https://docs.nvidia.com/deploy/driver-persistence/background.html>
15. <https://docs.nvidia.com/deploy/driver-persistence/persistence-daemon.html>
16. <https://docs.nvidia.com/datacenter/tesla/fabric-manager-user-guide/index.html>

THE PUBLISHER

About GPU Smith

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

Start with the workload

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

Hardware and supplier research

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

Deployment and operations

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

Work with GPU Smith

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

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

Website: <https://gpusmith.com>

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