



# PCIe Switch Topologies for 8-GPU Servers: P2P, ACS, IOMMU

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 · pci switch gpu server topology · nvidia-smi topo -m pix pxb phb · acs disable gpu p2p nccl · pci p2p gpu performance · 8 gpu pci server architecture · pci fabric

Original article: <https://gpusmith.com/articles/en/pci-switch-topologies-p2p-ac-s-8-gpu-servers>



## In this article

1. [Why topology decides PCIe multi-GPU performance](#)
  2. [Building blocks](#)
  3. [Reading nvidia-smi topo -m](#)
  4. [How P2P is routed](#)
  5. [ACS: what it is and why it breaks P2P](#)
  6. [IOMMU and virtualization](#)
  7. [Disabling ACS safely](#)
  8. [NCCL on PCIe-only boxes](#)
  9. [Design patterns for 8 GPUs](#)
  10. [Validation runbook](#)
- 

## Why topology decides PCIe multi-GPU performance

In an 8-GPU server without NVLink across all eight cards, every collective, host-to-device copy and GPUDirect RDMA transfer travels over a PCIe tree fixed by the motherboard and switch configuration. Two servers with identical parts can behave very differently under NCCL because one keeps GPU-to-GPU traffic inside a PCIe switch while the other forces it through a CPU root complex or across sockets. The matrix printed by `nvidia-smi topo -m` should be read as a design document: it states where peer-to-peer (P2P) traffic physically goes.

NVIDIA lists the H100 SXM at **900 GB/s** of NVLink bandwidth and the H100 NVL at **600 GB/s** of NVLink, with **128 GB/s** of PCIe Gen5 on both ([NVIDIA H100 product page](#)). The L40S has no NVLink at all and lists "PCIe Gen4 x16: 64GB/s bidirectional" ([NVIDIA L40S product page](#)); the RTX PRO 6000 Blackwell Server Edition is a PCIe Gen5 card ([NVIDIA RTX PRO 6000 Blackwell Server Edition page](#)). On such systems PCIe is the GPU fabric, and its layout is the main performance variable left to the operator.

### What the PCIe numbers actually mean

PCI-SIG defines PCIe 5.0 at 32.0 Gb/s per lane per direction of raw bandwidth, and states that a 32-lane PCIe 5.0 link provides 128.0 GB/s "in each direction simultaneously" ([PCI-SIG PCIe 5.0 FAQ](#)). PCIe 3.0 and later use 128b/130b encoding instead of 8b/10b ([PCI-SIG PCIe 3.0 FAQ](#)). The derived per-direction figures for a Gen5 x16 GPU port are:

1. Raw rate: 32 Gb/s per lane x 16 lanes = 512 Gb/s per direction = **64 GB/s per direction**.
2. After 128b/130b encoding:  $64 \times 128/130 = \sim 63.0$  **GB/s per direction**.
3. Both directions together:  $\sim 126$  GB/s, which is the figure vendors round to "128 GB/s" when they quote raw bidirectional bandwidth.

Packet headers and flow control reduce delivered bandwidth further; the real figure depends on payload size and platform and must be measured.

## Building blocks

**Root complexes and host bridges.** Each CPU socket exposes PCIe root ports behind one or more host bridges. Two sockets are joined by an SMP interconnect (UPI on Intel, xGMI on AMD, per [Supermicro's GPU server root architecture brief](#)).

**PCIe switches.** A switch fans upstream links out to many downstream ports. Broadcom's PEX89000 Gen5 family is offered in 144-, 104-, 88-, 72-, 48-, 32- and 24-lane configurations, supports link widths from x1 to x16, quotes "full line rate on all ports" and a cut-through latency of "less than 115 ns (x16 to x16)" ([Broadcom PEX89000 product brief](#), PEX89000-PB102, October 2023). Its quoted 1024 Gb/s (128 GB/s) raw bandwidth per x16 port is the bidirectional raw figure derived above.

**Endpoints.** GPUs, NICs and NVMe drives. NIC placement matters as much as GPU placement, because GPUDirect RDMA moves data directly between GPU and NIC.

### Three reference layouts

Supermicro's brief describes three PCIe GPU server architectures ([Supermicro, GPU Servers Root Explained, July 2023](#)):

Table 1. PCIe GPU server layouts

| Layout          | Physical arrangement (per Supermicro brief)                                                                                                   | GPU-to-GPU path                                                                 | Main risk                                |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|------------------------------------------|
| Single root     | One CPU reaches all GPUs through PLX switches; each switch connects to the CPU "via 2 PCIe x16" links and serves up to five double-width GPUs | Switch-local (PIX) within a switch; through one root complex between switches   | Host-bound bandwidth shared by many GPUs |
| Dual root       | Each CPU has its own switch(es); the example shows 4 GPUs, 2 add-on cards and 4 NVMe per CPU/switch                                           | Switch-local within a socket; across the SMP interconnect between sockets (SYS) | Cross-socket P2P path                    |
| Direct attached | Each CPU attaches up to four GPUs directly, eight per system, no switches                                                                     | Always through a root complex (PHB/NODE) or socket link (SYS)                   | No switch-local P2P at all               |

For collective-heavy work on PCIe cards, the deciding question is which GPU pairs, and which GPU-NIC pairs, share a switch.

## Reading nvidia-smi topo -m

The nvidia-smi documentation (fetched 2026-10-28; v610 is the newest changelog branch) defines the legend ([NVIDIA nvidia-smi documentation, Topology section](#)):

Table 2. nvidia-smi topo -m legend

| Code | Documented meaning | What it implies for P2P |
|------|--------------------|-------------------------|
| X    | Self               | n/a                     |

| Code | Documented meaning                                                                                  | What it implies for P2P                 |
|------|-----------------------------------------------------------------------------------------------------|-----------------------------------------|
| NV#  | Connection traversing a bonded set of # NVLinks                                                     | NVLink path, not PCIe                   |
| PIX  | Connection traversing a single PCIe switch                                                          | Best PCIe case; stays in the switch     |
| PXB  | Connection traversing multiple PCIe switches (without traversing the PCIe Host Bridge)              | Cascaded switches; still avoids the CPU |
| PHB  | Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)                        | CPU root complex forwards the traffic   |
| NODE | Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node | Crosses host bridges inside one socket  |
| SYS  | Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)       | Crosses sockets                         |

Two details are easy to miss. The newer `topo -gpu` and `topo -nvme` legends word PIX as "at most a single PCIe bridge" and PXB as "multiple PCIe bridges", so a PIX entry should be checked against the vendor block diagram rather than assumed to be one switch chip. And the matrix describes the tree statically; it does not show whether ACS redirects traffic upward. For that, the documentation provides `nvidia-smi topo -p2p p` (P2P PCIe capability per GPU pair) and `nvidia-smi topo -p -i <id1>,<id2>` (most direct PCIe path between two GPUs). The v610 changelog adds `topo -gpu`, `topo -nic` and related subcommands, plus a "DR - Disabled by regkey" item for `topo -p2p`.

## How P2P is routed

NVIDIA's GPUDirect RDMA documentation (CUDA 13.4) states the hard requirement: the two devices "must share the same upstream PCI Express root complex." It then ranks paths: where "there are only PCIe switches on the path," the result "is optimal and yields the best performance"; a path through a single CPU/IOH "works, but yields worse performance"; and a path across a QPI/HT link "may be extremely performance-limited or even not work reliably" ([NVIDIA GPUDirect RDMA documentation, Supported Systems](#)). The text addresses GPU-NIC transfers, but the physical ordering applies equally to GPU-GPU P2P.

The Linux P2PDMA documentation (kernel 7.3.0) explains why the root complex is the weak point. Behind a switch, "based on the ACS settings the transaction can route entirely within the PCIe hierarchy and never reach the root port," and the kernel "will evaluate the PCIe topology and always permit P2P in these well-defined cases." Beyond that, because "the PCIe specification doesn't define the forwarding of transactions between hierarchy domains," the kernel "defaults to blocking such routing," with an allow list for known-good host bridges ([Linux kernel docs: PCI Peer-to-Peer DMA Support](#)). CUDA P2P enablement is decided separately by the NVIDIA driver, so `nvidia-smi topo -p2p p` is the authoritative answer for a given box.

In matrix terms, PIX and PXB avoid the CPU; PHB and NODE depend on its forwarding; SYS adds the socket link that NVIDIA flags as potentially unreliable.

## ACS: what it is and why it breaks P2P

Access Control Services (ACS) are per-port PCIe controls for policing peer-to-peer traffic. The kernel's `config_acs` parameter lists seven: Source Validation, Translation Blocking, P2P Request Redirect, P2P Completion Redirect, Upstream Forwarding, P2P Egress Control and Direct Translated P2P ([Linux kernel parameters documentation](#)). With P2P Request Redirect enabled on a switch downstream port, a GPU-to-GPU request under the same switch is sent upstream to the root complex, where the IOMMU can inspect it, instead of port-to-port.

The consequence is that a PIX pair in `nvidia-smi topo -m` can behave like a PHB pair. NCCL's troubleshooting guide (NCCL 2.32.3) says IO virtualization can interfere with GPU Direct by "redirecting all PCI point-to-point traffic to the CPU root complex," causing "a significant performance reduction or even a hang" (NCCL 2.32.3 GPU troubleshooting, PCI Access Control Services). Redirected traffic from every pair under a switch then shares the switch uplink.

NCCL's bare-metal check is `sudo lspci -vvv | grep ACSCtl;SrcValid+` indicates ACS might be enabled (NCCL 2.32.3 GPU troubleshooting). The flags that force traffic upstream are `ReqRedir` and `CmpltRedir` on the downstream ports above each GPU.

## IOMMU and virtualization

ACS exists largely to make the IOMMU meaningful: if devices could talk directly, an IOMMU at the root complex could not isolate them, so the two settings are usually changed together.

On bare-metal Linux, NCCL 2.32.3 states that "CUDA and the NVIDIA driver stack do not support IOMMU-enabled PCIe peer-to-peer memory transfer" and that "the IOMMU must be disabled on Linux bare-metal systems to prevent silent device memory corruption." It treats `dmesg` messages such as `iommu: Default domain type: Translated or DMAR: IOMMU enabled` as signs of translated mode, to be avoided for GPU PCIe P2P (NCCL 2.32.3 GPU troubleshooting, Linux bare-metal IOMMU and PCIe peer-to-peer). The CUDA 13.4 GPUDirect RDMA documentation is less strict for GPU-NIC traffic: IOMMUs "must be disabled or configured for pass-through translation for GPUDirect RDMA to work" (NVIDIA GPUDirect RDMA documentation). The relevant Linux options are `iommu=pt` (equivalent to `iommu.passthrough=1`), `intel_iommu=off` and `iommu=off` ([Linux kernel parameters documentation](#)). The NCCL page never mentions `passthrough`: its fix is to "avoid translated IOMMU mode" ("Disable IOMMU translation for the GPU path"), which an identity (pass-through) domain meets in principle, but the CUDA guide it cites says disabled. Validate `passthrough` for GPU-GPU P2P per platform with `simpleP2P` and `nvbandwidth` rather than assume it.

Virtual machines invert the requirement. NCCL's guide states that "virtual machines require ACS to function, hence disabling ACS is not an option," and that for maximum performance in VMs, "ATS needs to be enabled in network adapters" (NCCL 2.32.3 GPU troubleshooting). With Address Translation Services, a device issues requests that are already translated, and ACS has a separate Direct Translated P2P control for those requests, so a virtualized host can keep isolation for untranslated traffic while allowing an efficient NIC path. Guests also need a faithful PCIe tree: NCCL "relies on `/sys` to discover the PCI topology."

## Disabling ACS safely

Three methods turn ACS redirection off on a single-tenant bare-metal host; the first two come from the [NCCL 2.32.3 troubleshooting guide](#), the third from the [kernel parameters documentation](#):

Table 3. ACS disable methods

| Method                         | How                                                                                                              | Persistence                                     |
|--------------------------------|------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| BIOS                           | Disable IO virtualization or VT-d                                                                                | Persistent; also disables the IOMMU             |
| setpci on Broadcom PLX bridges | <code>sudo lspci   grep PLX</code> , then <code>sudo setpci -s &lt;BDF&gt; ECAP_ACS+0x6.w=0000</code> per bridge | Lost at every reboot; must run from a boot unit |
| Kernel parameter               | <code>pci=disable_acs_redir=&lt;pci_dev&gt;[;...]</code> , or <code>pci=config_acs=</code> with per-bit flags    | Persistent via boot command line                |

The cost is explicit: `disable_acs_redir` "removes isolation between devices and may put more devices in an IOMMU group" ([Linux kernel parameters documentation](#)). On a multi-tenant host that matters: IOMMU groups are "the unit of ownership used by VFIO", and the VFIO documentation cites a bridge "allowing redirection without reaching the IOMMU" as a cause of reduced isolation ([Linux kernel docs: VFIO](#)); once ACS is off, GPUs under one switch typically can no longer go to different guests, and each can reach its peers unseen by the IOMMU. The working rule: disable ACS only on single-tenant bare-metal hosts; keep it on, with ATS on NICs, for hosts that serve VMs.

A `setpci` fix that is not reapplied after reboot produces nodes that pass acceptance and later regress, so the persistence mechanism belongs in the node image and the check in the boot-time health gate, next to the checks in [PCIe link health troubleshooting](#).

## NCCL on PCIe-only boxes

NCCL selects the P2P transport when CUDA reports that GPUs can access each other's memory ([NCCL 2.32.3 GPU troubleshooting](#)). Two environment variables in NCCL 2.32.3 override that decision ([NCCL 2.32.3 environment variables](#)):

- `NCCL_P2P_DISABLE=1` disables "direct GPU-to-GPU (P2P) communication" entirely.
- `NCCL_P2P_LEVEL` sets "the maximum distance between GPUs where NCCL will use the P2P transport," using the same vocabulary as the topology matrix: LOC (never), NVL, PIX, PXB, PHB and SYS. If unset, NCCL selects a value from the architecture; legacy integers (LOC=0 to SYS=4) are discouraged.

The GPU-NIC equivalent, `NCCL_NET_GDR_LEVEL`, caps GPU-NIC distance for GPUDirect RDMA with the same LOC/PIX/PXB/PHB/SYS values. `NCCL_NET_GDR_READ` defaults to 1 on NVLink-based platforms and 0 otherwise, and the documentation notes that reading directly from GPU memory "is known to be slightly slower" on some platforms, such as PCI-E ([NCCL 2.32.3 environment variables](#)).

## When to force P2P off

The NCCL guide uses these variables diagnostically. Three situations justify a persistent override:

1. **Hangs or corruption on PHB/SYS pairs** that persist after ACS and IOMMU are corrected: cap with `NCCL_P2P_LEVEL=PIX` so P2P is used only where no CPU forwarding is involved.
2. **VM guests where ACS must stay on:** P2P through the redirected path may be slower than NCCL's non-P2P path through host memory; measure both.
3. **Mixed designs** where some GPU pairs are PIX and others SYS: a level cap keeps the fast pairs on P2P without forcing the slow ones.

Overrides should be recorded per node type, together with the XML written by `NCCL_TOPO_DUMP_FILE` ([NCCL 2.32.3 environment variables](#)).

## Design patterns for 8 GPUs

The design question is which of three flows share which links: GPU-GPU collectives, GPU-NIC traffic, and host-to-GPU loading.

### Lane budget (illustrative)

Taking the 144-lane PEX89000 configuration as a reference and x16 for every GPU, NIC and uplink:

- **Eight GPUs on one switch:**  $8 \times 16 = 128$  lanes for GPUs, plus one x16 uplink = 144 lanes. Every GPU pair is PIX, but no lanes remain for NICs and host traffic is oversubscribed 8:1.
- **Four GPUs per switch, two switches:**  $4 \times 16 = 64$  GPU lanes, plus  $2 \times 16 = 32$  NIC lanes, plus  $2 \times 16 = 32$  uplink lanes = 128 lanes per switch. Host oversubscription is  $64:32 = 2:1$ , and every GPU has a PIX NIC shared with one other GPU.

These are derived counts, not a specific product; real boards also spend lanes on NVMe and management.

### Uplink oversubscription arithmetic (illustrative)

For Supermicro's single-root description (each switch on 2 x16 uplinks, up to five GPUs), with ~63.0 GB/s per direction per x16 port after encoding:

1. Uplink capacity per direction:  $2 \times 63.0 = 126$  GB/s.
2. Demand if five GPUs read from host memory at once:  $5 \times 63.0 = 315$  GB/s.
3. Ratio:  $315 / 126 = 2.5:1$ , so each GPU gets at most ~25.2 GB/s from the host ( $126 / 5$ ), before CPU memory and root-port limits.

An illustrative 400 Gb/s NIC port carries  $400 / 8 = 50$  GB/s per direction. Under the same switch as its GPUs (PIX), GPUDirect RDMA traffic never touches the uplink; under the other socket (SYS), those 50 GB/s cross both uplinks and the socket interconnect, competing with host traffic.

## Ring placement

An 8-GPU ring spanning switches on two sockets crosses the inter-switch cut once in each direction. With P2P across SYS, ring bandwidth is bounded by the slowest of the GPU port, the uplink and cross-socket P2P forwarding, which NVIDIA warns may be "extremely performance-limited." With P2P capped at PXB, the crossing goes through host memory instead. Neither outcome is predictable from the matrix alone. Rank-to-NIC pairing is covered in [CPU/NIC affinity and rank binding](#).

Table 4. Pattern checklist

| Check                | Preferred outcome                                                            |
|----------------------|------------------------------------------------------------------------------|
| GPU pairs per switch | Collective-heavy groups (e.g. tensor parallel) inside one switch (PIX)       |
| GPU-NIC pairing      | Every GPU has a PIX or PXB NIC; no GPU depends on a SYS NIC                  |
| Uplink ratio         | Known and recorded (e.g. 2:1, 2.5:1); host-heavy workloads sized accordingly |
| ACS/IOMMU policy     | Matches tenancy: off on single-tenant bare metal, on with ATS for VMs        |
| NCCL overrides       | Documented per node type with the topology dump                              |

## Validation runbook

Table 5. Validation steps

| Step              | Command                                                                                                   | Expected result                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 1. Topology       | <code>nvidia-smi topo -m</code> (or <code>topo -gpu</code> / <code>topo -nic</code> on v610+ drivers)     | Matches the vendor block diagram: PIX within switches; each GPU's NIC at PIX or PXB   |
| 2. P2P capability | <code>nvidia-smi topo -p2p p</code>                                                                       | OK for all intended pairs; no unexpected "not supported" or "DR" entries              |
| 3. ACS state      | <code>sudo lspci -vvv   grep ACSCt1</code>                                                                | No ReqRedir+/CmplRedir+ on switch downstream ports for bare-metal single-tenant hosts |
| 4. IOMMU mode     | <code>cat /proc/cmdline; dmesg   grep -i -E "iommu dmar default domain"</code>                            | No "Default domain type: Translated" on bare-metal P2P hosts                          |
| 5. Functional P2P | <code>cuda-samples simpleP2P</code>                                                                       | Ends with "Test passed"                                                               |
| 6. Bandwidth      | <code>./nvbandwidth -t device_to_device_memcpy_write_ce</code> (list tests with -l)                       | PIX pairs near the per-direction port figure; PHB/SYS pairs recorded separately       |
| 7. NCCL           | all-reduce test with and without <code>NCCL_P2P_LEVEL=PIXB</code> ; save <code>NCCL_TOPO_DUMP_FILE</code> | Override choice justified by measured difference                                      |
| 8. Persistence    | Reboot, repeat steps 3-6                                                                                  | Same results; no regression from lost <code>setpci</code> settings                    |

Steps 2 to 5 follow NCCL's troubleshooting guide, which recommends `nvbandwidth` over the CUDA sample for bandwidth ([NCCL 2.32.3 GPU troubleshooting](#); [NVIDIA nvbandwidth](#)). GPU-NIC paths need their own test, described in [GPUDirect RDMA qualification](#), and the results belong in the node record defined by the [GPU cluster acceptance testing runbook](#).

The NCCL guidance quoted here is from the NCCL 2.32.3 documentation (v2.32.3-1, released 2026-09-17, still the latest release on 2026-10-28); it is version-specific and should be re-read for the deployed release. Bandwidth figures here are derived from line rates; only measured results belong in a capacity plan.

GPU Smith can review a PCIe server's block diagram and topology output against the intended workload before purchase or acceptance.

## Source links

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

1. <https://www.nvidia.com/en-us/data-center/h100>
2. <https://www.nvidia.com/en-us/data-center/l40s>
3. <https://www.nvidia.com/en-us/data-center/rtx-pro-6000-blackwell-server-edition>
4. [https://pcisig.com/faq?field\\_category\\_value%5B%5D=pci\\_express\\_5.0&keys=](https://pcisig.com/faq?field_category_value%5B%5D=pci_express_5.0&keys=)
5. [https://pcisig.com/faq?field\\_category\\_value%5B%5D=pci\\_express\\_3.0&keys=](https://pcisig.com/faq?field_category_value%5B%5D=pci_express_3.0&keys=)
6. <https://supermicro.com/products/brief/product-brief-GPU-Servers-Root-Explained.pdf>
7. [https://www.mouser.com/datasheet/2/678/PEX89000\\_PB102-3394994.pdf](https://www.mouser.com/datasheet/2/678/PEX89000_PB102-3394994.pdf)
8. <https://docs.nvidia.com/deploy/nvidia-smi/index.html>
9. <https://docs.nvidia.com/cuda/gpudirect-rdma>
10. <https://docs.kernel.org/driver-api/pci/p2pdma.html>
11. <https://docs.kernel.org/admin-guide/kernel-parameters.html>
12. [https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/troubleshooting/gpu\\_troubleshooting.html](https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/troubleshooting/gpu_troubleshooting.html)
13. <https://docs.kernel.org/driver-api/vfio.html>
14. <https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html>
15. <https://github.com/NVIDIA/nvbandwidth>

## 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.