VDC: A Software-Emulated Virtual Datacenter for AI/HPC Research and Education

PacketFive Networks Limited
connect@packetfive.com

June 12, 2026

Abstract

The skills required to operate modern AI and high-performance computing (HPC) clusters are increasingly inaccessible: production-grade GPU servers, NVLink switches and InfiniBand fabrics are priced beyond the reach of universities, independent researchers, and most engineering teams. This paper introduces VDC, the PacketFive Virtual Datacenter — a 100% software-emulated AI/HPC datacenter built on QEMU. VDC presents unmodified Linux guests with real PCIe virtual GPUs, virtual RDMA-capable NICs, virtual top-of-rack InfiniBand/RoCEv2 switches and a virtual GPU-to-GPU peer fabric. The result is a complete multi-node GPU cluster that runs on a single laptop while preserving the hardware semantics — DCB, ECN, IB Local Identifiers, GPU peer DMA — that matter for production debugging and skill transfer.

1 Introduction

The cost gap between the hardware required to deploy an AI/HPC system and the hardware available for learning has widened sharply since 2023. A single eight-GPU server with the matching NVLink switch and InfiniBand HCAs now exceeds the entire annual capital budget of most academic research groups. Existing virtualisation approaches available in Linux — bridges, `macvlan`, Soft-RoCE — do not preserve the hardware semantics that make production AI/HPC operations difficult: priority flow control, explicit congestion notification, IB Local Identifier routing, and GPU peer-to-peer DMA.

VDC fills this gap. Rather than hiding hardware behind host-side shortcuts, VDC builds custom QEMU PCIe device models and the corresponding Linux kernel drivers, so that the guest operating system sees real RDMA host channel adapters and real GPUs on its PCIe bus. Standard, unmodified userspace — `libibverbs`, `perftest`, NCCL's IB transport, and a CUDA-equivalent runtime — runs against these devices exactly as it would on physical Mellanox or NVIDIA hardware.

2 Design Philosophy

VDC is built on three commitments:

- **Preserve hardware semantics.** A learner who masters VDC must be able to transfer that knowledge to production NVIDIA, AMD or Intel hardware without re-learning fundamentals. Frame headers, DMA queues, doorbells and interrupts behave the way real silicon behaves.
- **Use unmodified guest stacks.** The guest kernel runs upstream Linux drivers (with a small additional module per device), and userspace runs the standard distribution-provided RDMA and GPU stacks. No patched libraries, no special compilers.
- **Open and modifiable.** Every line of VDC is open source. Researchers can inspect, modify and extend the emulation to study new protocols, scheduling algorithms or fabric topologies.

3 System Architecture

VDC is composed of four interlocking open-source components, each released as a stand-alone repository and integrated through the VDC orchestration project.

3.1 HiSwitch — Virtual TOR Fabric

`hicain-vswitchd` is a standalone C daemon implemented around an `epoll` event loop on a set of host UNIX-domain sockets. Each socket represents a physical port; raw byte streams arriving on a port are classified into one of three pipelines:

- An **InfiniBand pipeline** parses the IB Local Routing Header (LRH) and forwards based on

the Local Identifier (LID) using an IB forwarding table.

- A **RoCEv2/Ethernet pipeline** parses the Ethernet MAC header and forwards using a standard L2 forwarding table with VLAN tagging.
- A **control plane** listens on a dedicated JSON-over-UDS management socket, accepting commands to bring ports up and down, configure per-port mode (`IB`, `ETH` or `AUTO`), manipulate forwarding tables, and stream telemetry to the management dashboard.

DCB primitives — priority flow control, enhanced transmission selection, explicit congestion notification — are modelled end-to-end across the fabric, allowing realistic experiments with congestion control and lossless transports.

3.2 HiNIC — Virtual RoCE-IB NIC

HiNIC is a custom QEMU PCIe device exposing Base Address Registers (BARs) for memory-mapped I/O and a doorbell page. When the guest rings a doorbell to submit a work request, the device model reads the descriptor and the associated buffer directly from guest memory, frames the bytes into the appropriate protocol (RoCEv2 over UDP/4791 or native InfiniBand), and writes them to the UNIX-domain socket that represents the "cable" to HiSwitch.

Two Linux kernel modules accompany the device. `hicain_net.ko` probes the device by Vendor/Device ID and registers a `netdev`; `hicain_ib.ko` registers as an RDMA-capable device with the kernel's `ib_core` subsystem. Together they expose the standard Verbs API to userspace, so `libibverbs`, `perftest`, and NCCL's IB transport operate without modification.

3.3 HiGPU — Virtual GPU

HiGPU is a QEMU PCIe device that implements a Single-Instruction Multiple-Thread (SIMT) execution model with configurable Streaming Multiprocessors, tensor units, register files, shared memory, an L2 cache and HBM-equivalent device memory. The accompanying userspace mirrors the NVIDIA stack one-to-one, with cleanly-named HiCAIN equivalents: HCC (CUDA Runtime), `hcc` (the LLVM-backed compiler analogous to `nvcc`), `hi-smi` (analogous to `nvidia-smi`), HiCCL (analogous to NCCL), and HiSHMEM. This naming preserves the conceptual mapping — a learner who knows CUDA recognises every API — without using vendor trademarks.

3.4 HiLink — GPU Peer Fabric

HiLink is an independent fabric carrying only GPU-to-GPU peer traffic, equivalent in role to NVIDIA NVLink and NVSwitch. `higpu-link-switchd` is a second host daemon, and the inter-VM fast path uses `ivshmem` to provide cross-VM shared memory at memory-copy speed rather than network speed. A VM may have a HiGPU but no HiNIC — in which case it still participates in GPU peer DMA over HiLink — or vice versa; the two fabrics are functionally orthogonal, exactly as in a real GPU server.

4 Reference Deployment Topology

The reference VDC rack is modelled on the Open Compute Project Open Rack v3 (ORv3), the 2023-vintage rack design adopted across the hyperscaler AI build-out. Each rack carries:

- One 1OU **HiSwitch** TOR (RoCEv2 + InfiniBand, 10 ports, DCB enabled).
- One 1OU **HiLink** spine (GPU peer fabric, 8 ports).
- One 1OU management host running both daemons.
- Eight 2OU VM "blades" — each a QEMU guest with a HiNIC and a HiGPU.

While the "U" heights are visual rather than physical in the emulator, the layout maps one-to-one to physical ORv3 racks, so the same diagrams serve teaching, design, and real lab build-outs.

5 Educational and Research Applications

VDC is the lab environment for the HiCAIN training programme. Students run real distributed-training workloads, RDMA microbenchmarks (`ibv_rc_pingpong`, `perftest`), and MPI collective operations against the emulated fabric. Coursework that previously required a hyperscale lab now runs on a workstation. For research, VDC's open and modifiable architecture enables novel experiments in fabric scheduling, congestion control, in-network computing and security — particularly attractive in academic contexts where commercial silicon remains a black box.

6 Conclusion

VDC removes the hardware barrier that has limited AI/HPC infrastructure education and research to those with hyperscaler-grade budgets. By emulating the full datacenter — compute, fabric, NIC and switch — in software, while preserving the hardware semantics that matter, PacketFive makes the next generation of AI cluster operators, researchers and engineers possible. VDC is open source and available at `https://github.com/PacketFive/vdc`, with companion training material at `https://hicain.io`.