RADOS-NKV

Key Value Namespaces

Kyle Bader

Chief Architect, Data and AI, Ceph

IBM Storage

AGENDA

- Introduction
- GPU Initiated
- What is the target?
- The road ahead

Garth Gibson told us in 1997 that the storage drive itself needed to become a secure, network-addressable, intelligent object engine capable of processing drive-level semantic commands.

Hardware couldn’t realize it then. **AI requires it now**.

RADOS-NKV

Exposes RADOS, a reliable object storage service through the NVMe key-value command set.

A GPU kernel can initiate IO *or computation*, and data bypasses the CPU with peer-to-peer DMA

RADOS-NKV

Store : Retrieve : Delete : Exist : List : Execute

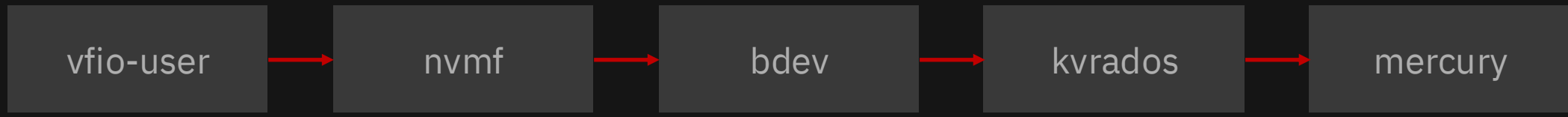
Namespace	→	Namespace
Key	→	Object ID
Value	→	Object Data

Max inline key : 16-bytes

Max extended key: 255-bytes

Values up to 128MB,
default max of 64MB

RADOS-NKV



pci device emulation

controller cmd dispatch

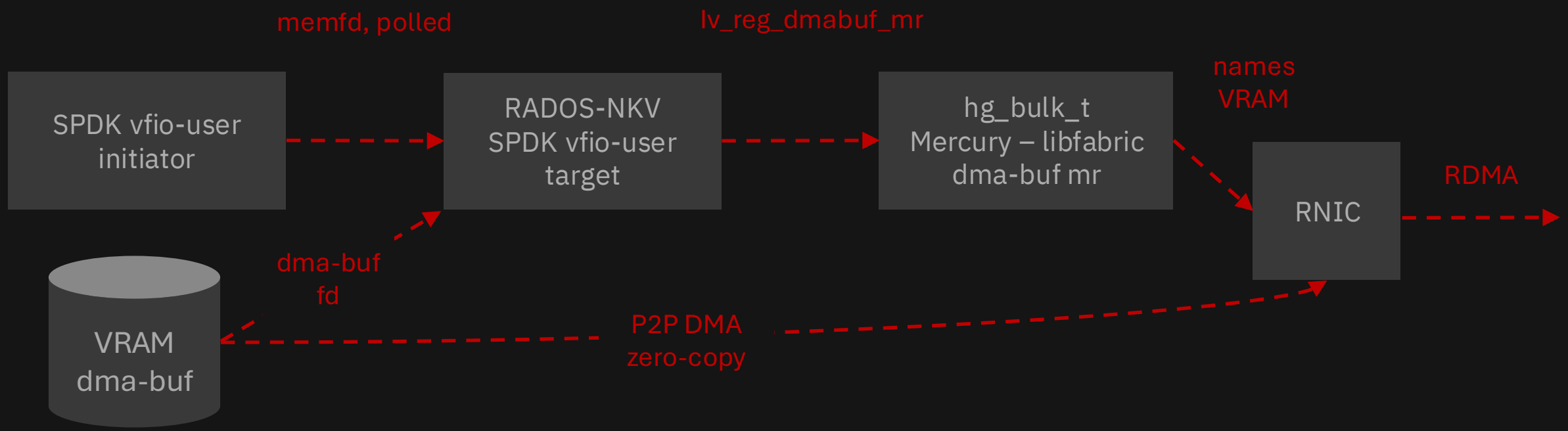
bdev kv cmd passthrough

out-of-tree bdev backend (key-value)

rpc framework

RADOS-NKV

CPU Initiated, Bare Metal



RADOS-NKVX

OSD host service

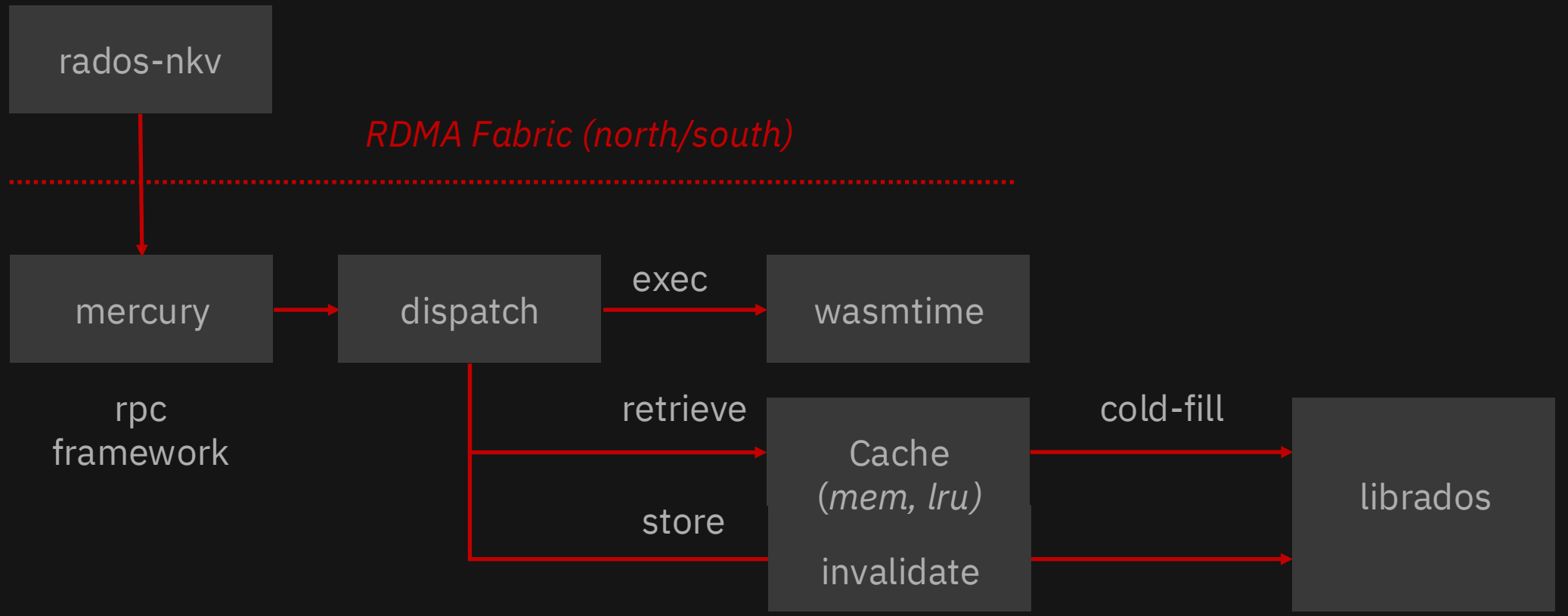
Pluggable Execution Runtime

- WebAssembly reference
- Velox (future)

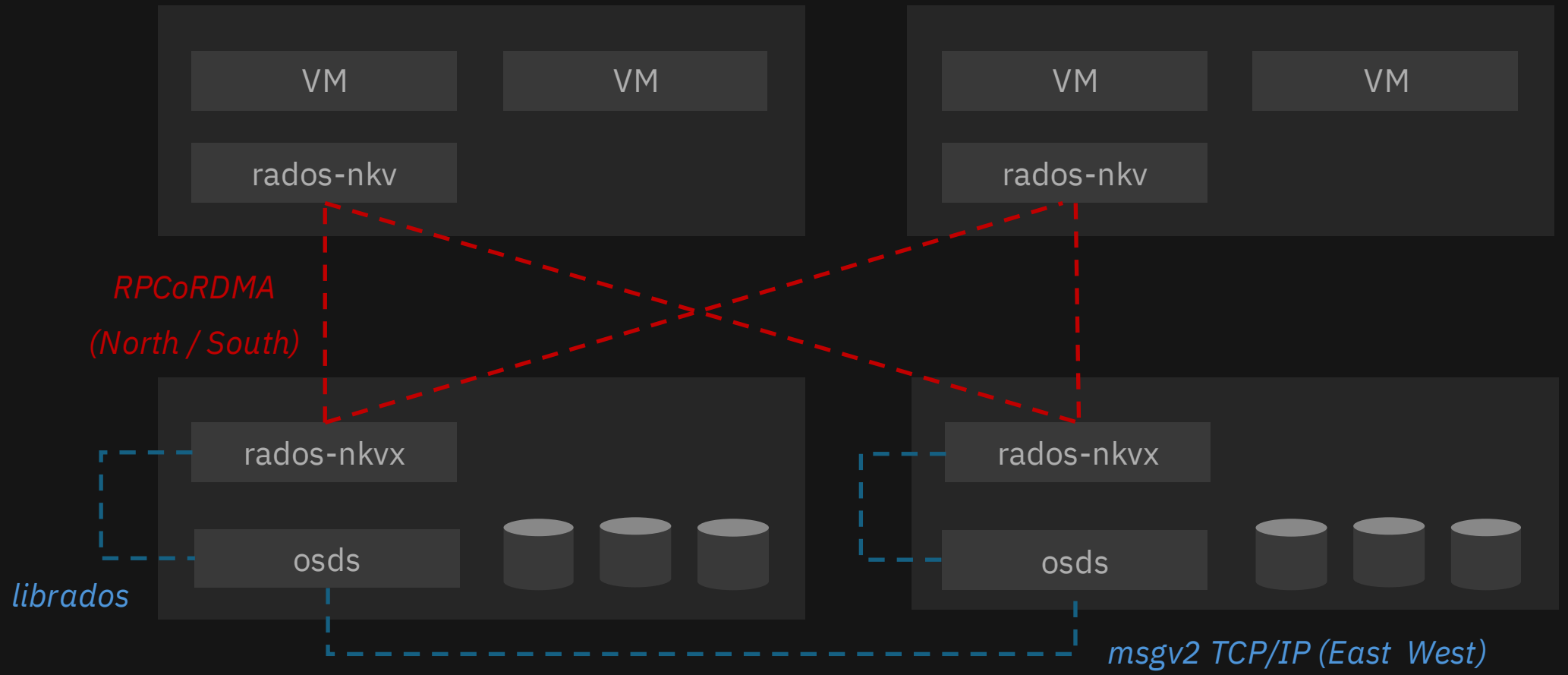
Local Cache

- In-memory (LRU)
- Cold-fill via librados
- Invalidate on store

RADOS-NKVX


rados-nkv
RDMA Fabric (north/south)
mercury
rpc framework
dispatch
exec
wasmtime
retrieve
store
Cache
(mem, lru)
invalidate
cold-fill
librados

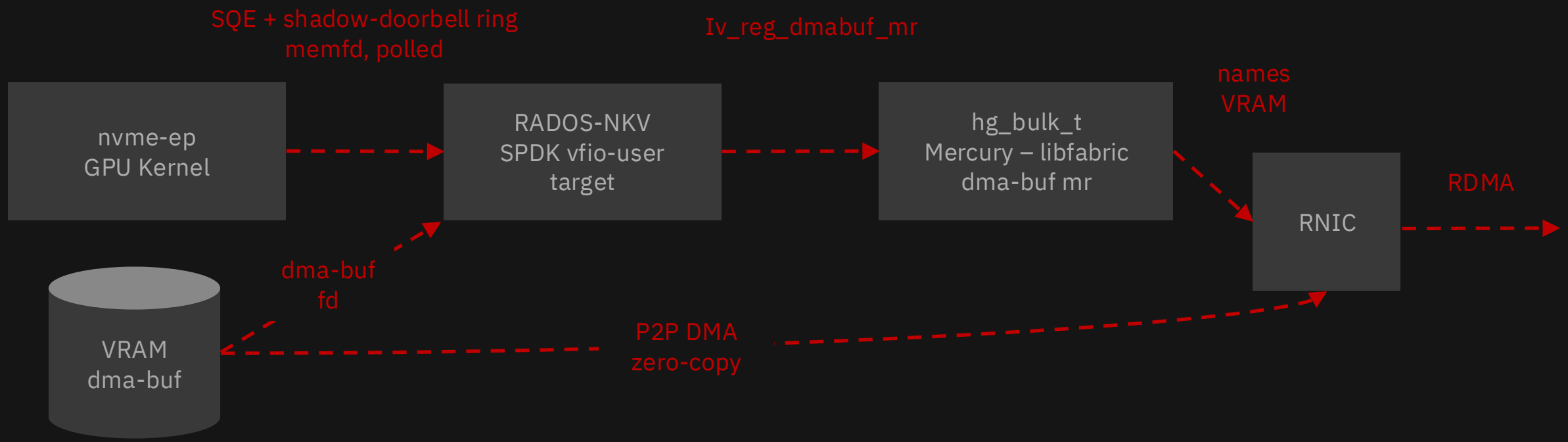
RADOS-NKV{X}


VM
VM
rados-nkv
VM
VM
rados-nkv
RPCoRDMA
(North / South)
rados-nkvx
osds
rados-nkvx
osds
librados
msgv2 TCP/IP (East West)

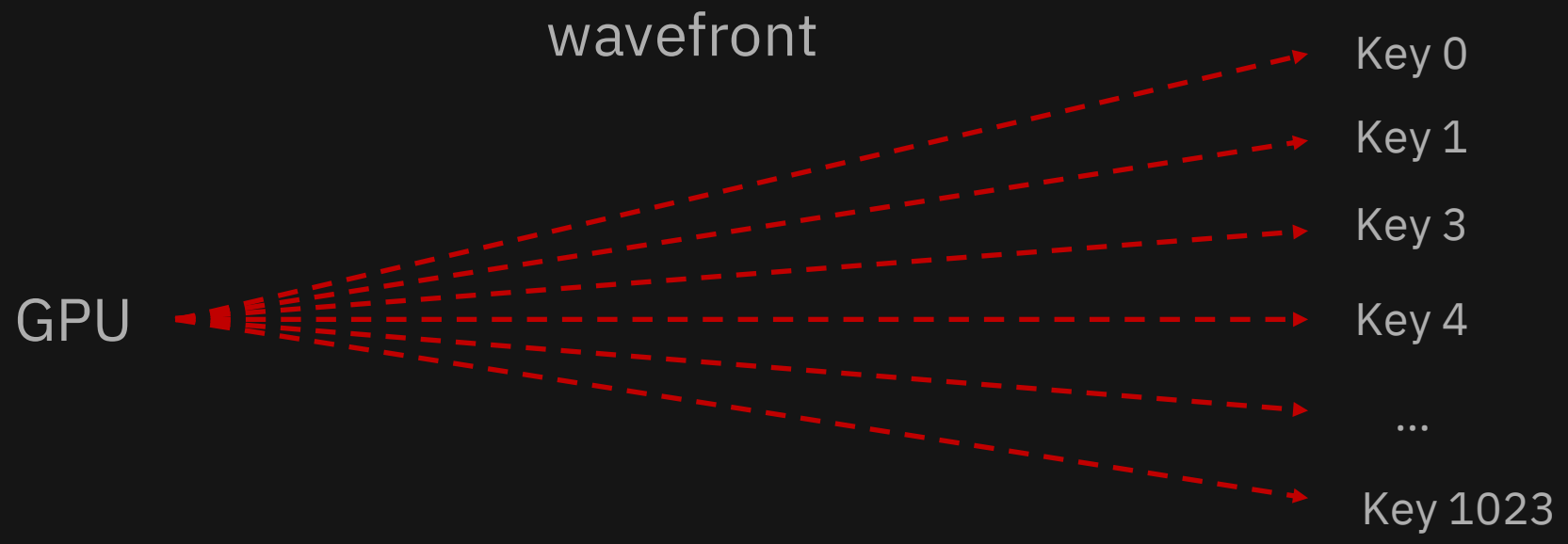
GPU INITIATED

RADOS-NKV

GPU Initiated, Bare Metal



RADOS-NKV

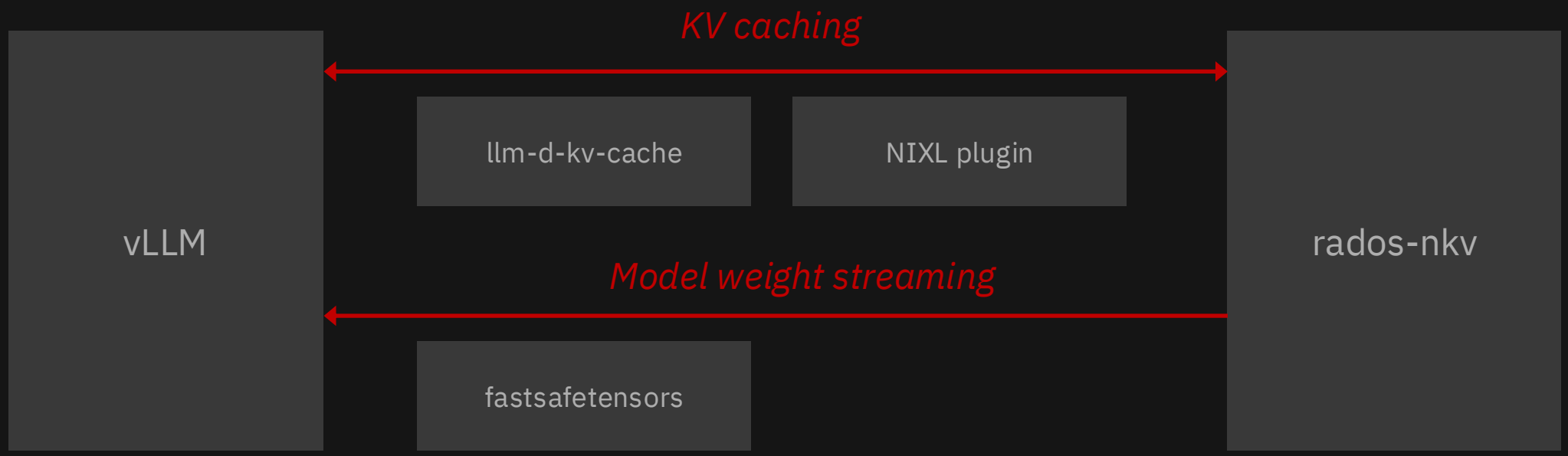


one SQ drives many ops

What is the target?

RADOS-NKV

Inference use-cases



RADOS-NKV

Performance, scale, deployment

Oberon and HELIOS class

- 18x compute trays
  - 2x CPU
  - 4x GPU
  - 4x 800GbE AI (E-W)
  - 2x 400GbE Storage (N-S) (rados-nkv)

Ceph context class cluster (C4)

- 36x storage nodes
  - 400-800GbE RDMA, rados-nkvx (N-S)
  - 400GbE TCP/IP, OSDs (E-W)
  - Gen5 EDSFF NVMe
  - At least 16TB usable / GPU

The road ahead

RADOS-NKV

Governance, Packaging and validation

- Move project to Ceph organization
- Packaged: Fedora 44, UBI – RPMs and containers
- Validated end-to-end on Strix Halo (iGPU, UMA), g6.xlarge (dGPU)

RADOS-NKV

Land in upstream SPDK

PR28298	bdev/nvme KV command-set passthrough	Ben Walker, NVIDIA
PR27457	env: spdk_mem_register_dma_buf	Ben Walker, NVIDIA
PR28950	env: spdk_mem_register_external_fd	Kyle Bader, IBM

27457 → dGPU, rados-nkv on a DPU attached to GPU host

28950 → dGPU, rados-nkv on the GPU host itself

RADOS-NKV

Ecosystem, going generic

SPDK (vfio-user) plugin for NIXL (wip-generic-spdk)

- Generic block or key-value, not rados-nkv specific
- 16-byte / 128-bit keys from positional lineage hashing

SPDK (vfio-user) plugin for fastsafetensors

- Supplants rados-nkv-weights
- Stream weights from rados-nkv to vLLM

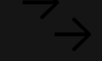
RADOS-NKV

Kubernetes CSI driver

Provision key-value namespaces via PersistentVolumeClaims

rados-nkv on

- GPU host
- GPU-host-resident DPU
- DPU Vendor agnostic, modeled after SPDK CSI



Questions?