

KNOD: In-Kernel GPU Offload for Packet Processing

Taehee Yoo

LINE Investment Technologies

Hoyeon Lee

SUSE

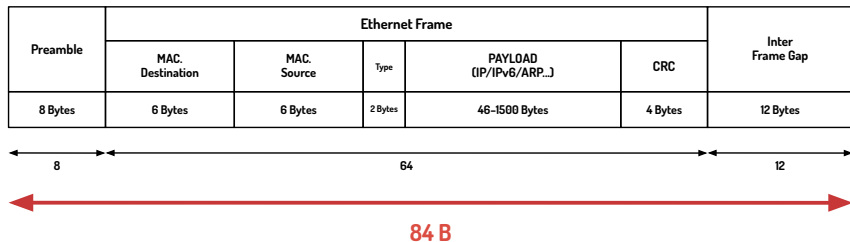


**LINUX
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026

Scaling Costs CPU Cores

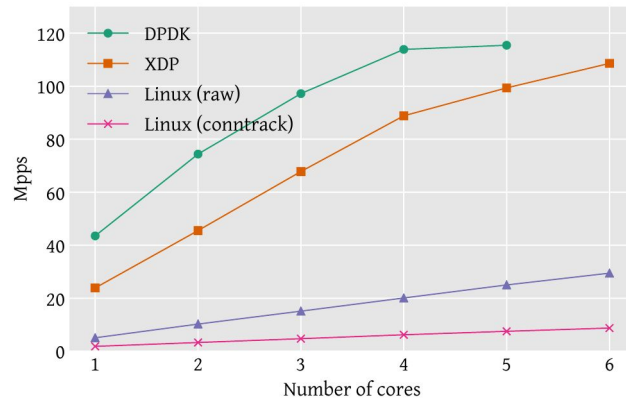
MINIMUM ETHERNET FRAME



$$100 \text{ Gb/s} \div 84 \text{ B} = 148,809,523 \text{ pps}$$

148.8 Mpps

XDP_DROP CPU SCALING

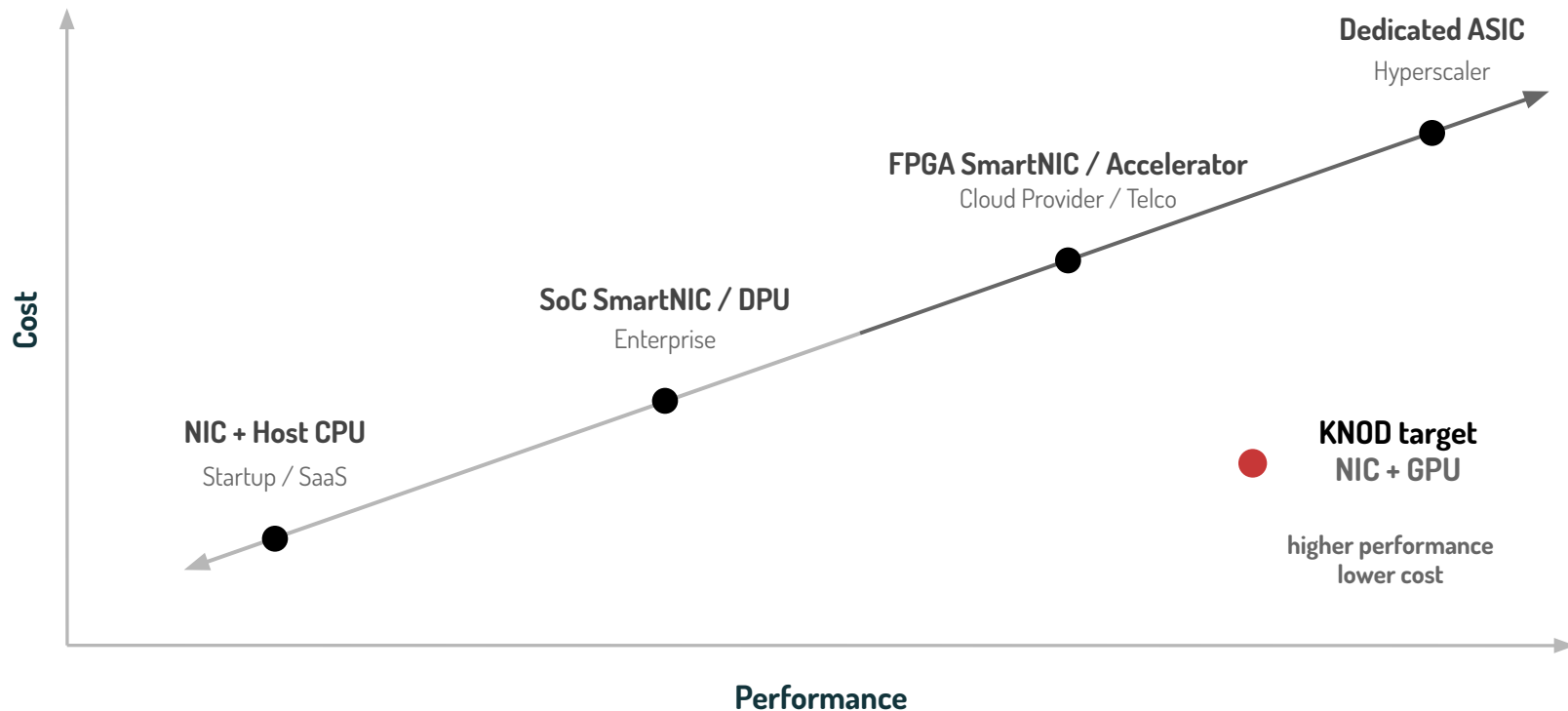


High packet rate

=> More CPU Cores

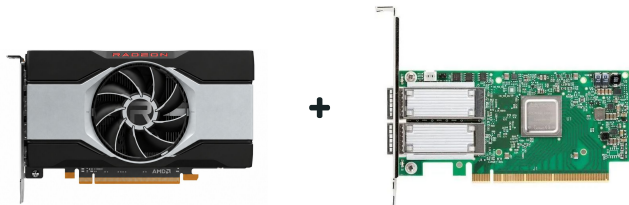
XDP CPU chart: CoNEXT '18, Fig. 3

Existing Approaches vs. KNOD



How Cheap?

KNOD development setup



Radeon RX 6600

secondary ~\$150-200

ConnectX-6 100G

secondary ~\$250-400

~\$400-600

Consumer GPU + 100G NIC

Other 100G-class hardware

FPGA SmartNIC / Accelerator

AMD Alveo U55C

2×100G FPGA card



\$4,747

- 10×

Dedicated appliance (ASIC)

Arista 7170-32C

32×100G Tofino switch

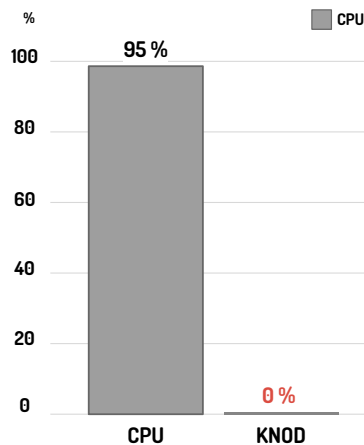


~\$14k

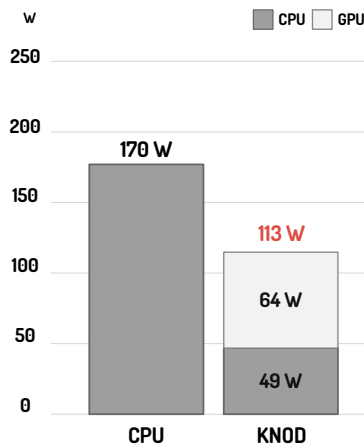
- 30×

public hardware prices

How Performant?



CPU Usage



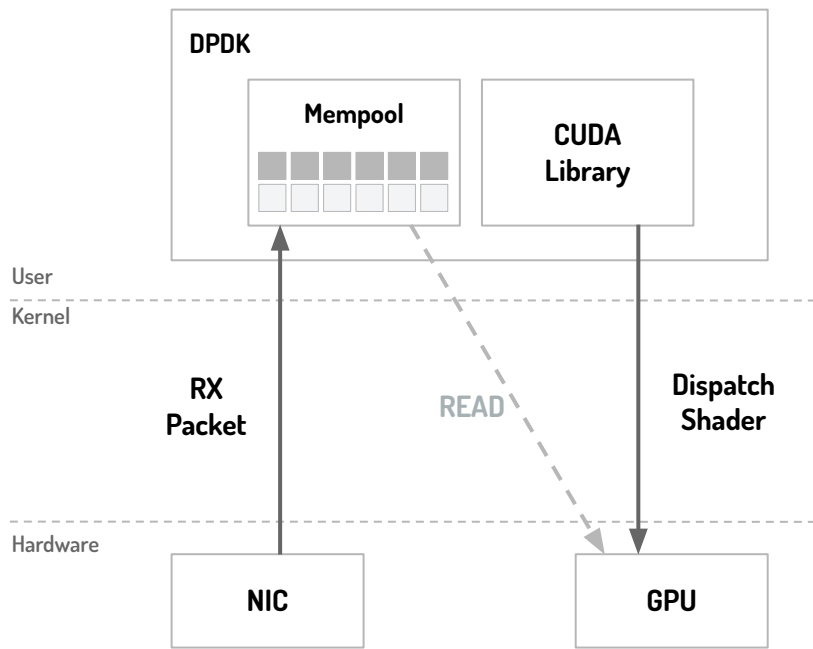
Power Consumption

**Katran-based LB
about 64 Mpps**

**=> not line-rate yet,
but still improving**

GPU packet processing

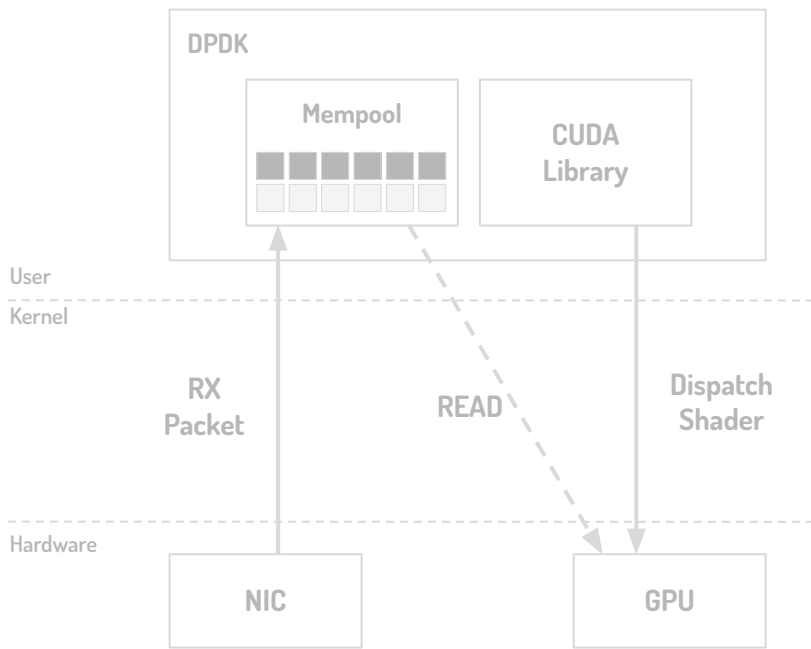
`l2fwd-nv · DPKD + CUDA`



`l2fwd-nv -m 0 (Host pinned mode)`

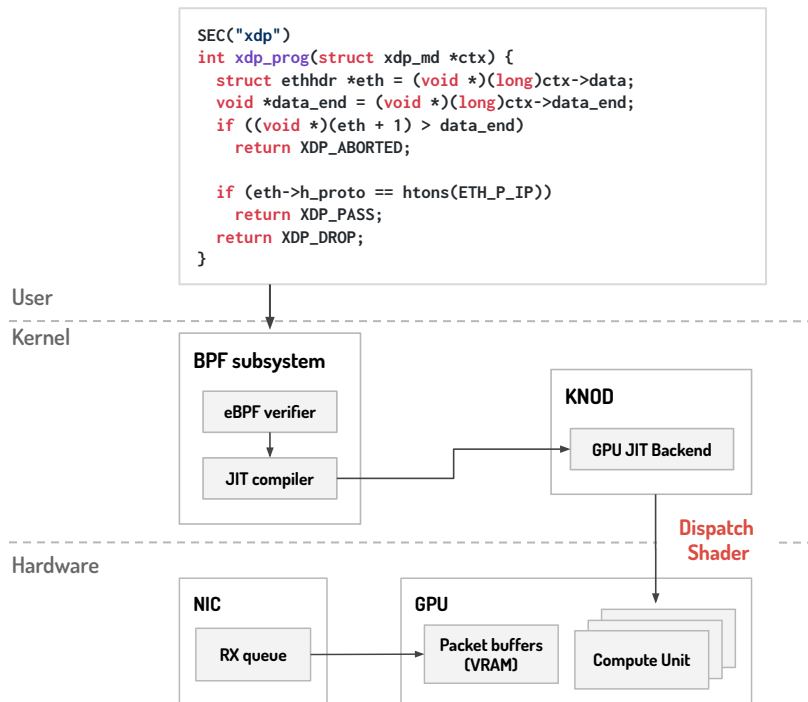
GPU packet processing with XDP

I2fwd-nv · DPDK + CUDA



I2fwd-nv -m 0 (Host pinned mode)

KNOD · XDP



High-Performance XDP Offload with Commodity GPU

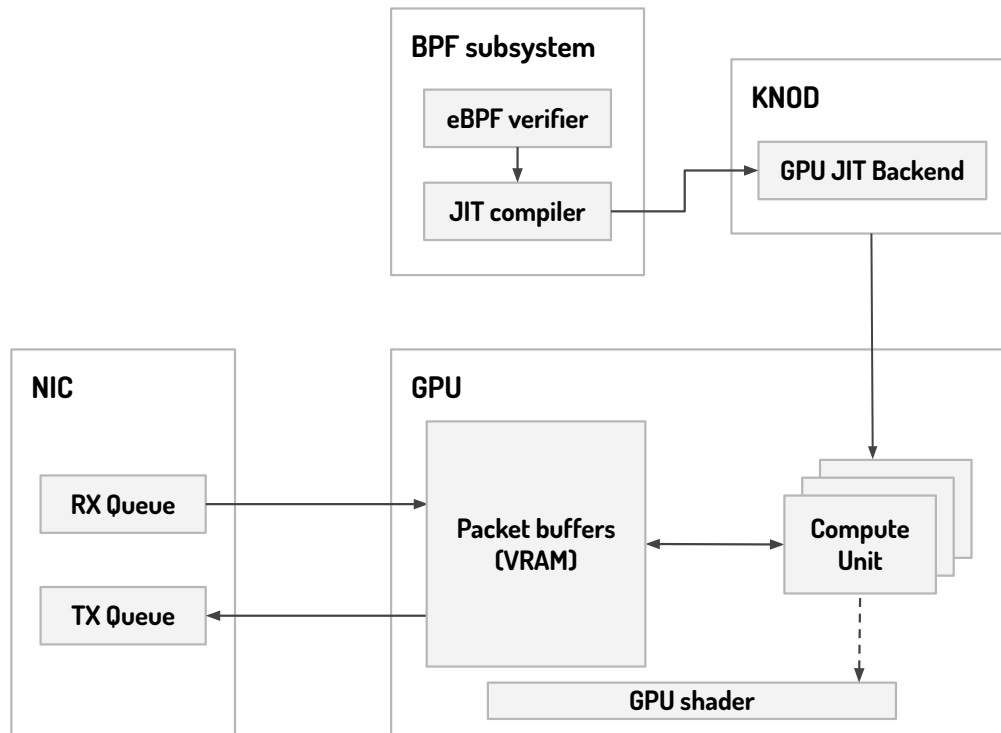
Overview

Overview

- NIC/GPU binding

- BPF-to-GPU JIT

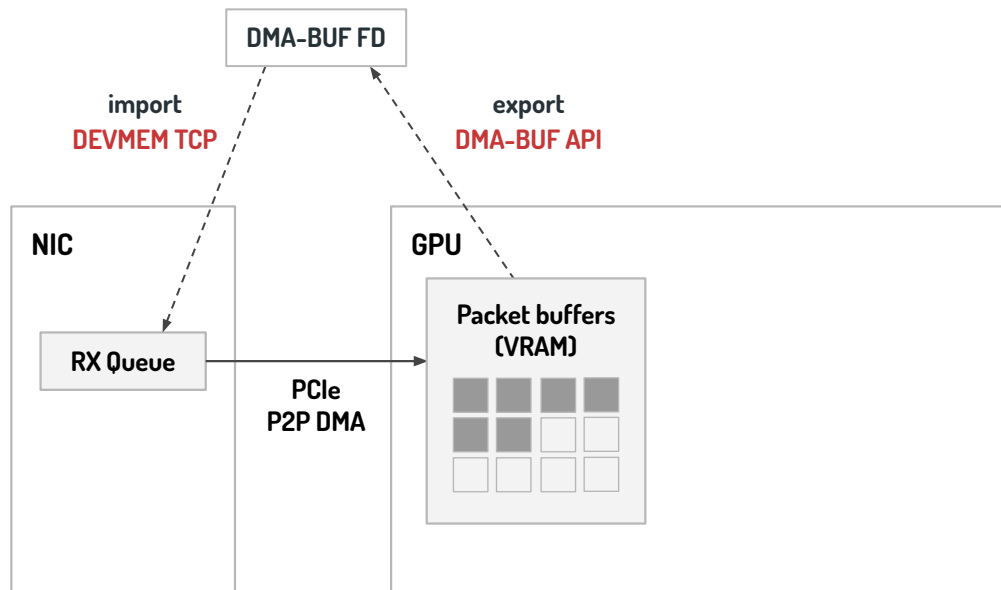
- GPU Shader execution



Overview

- NIC/GPU binding

- Use VRAM as RX Queue backing memory
- Direct packet writes to VRAM via PCIe P2P DMA

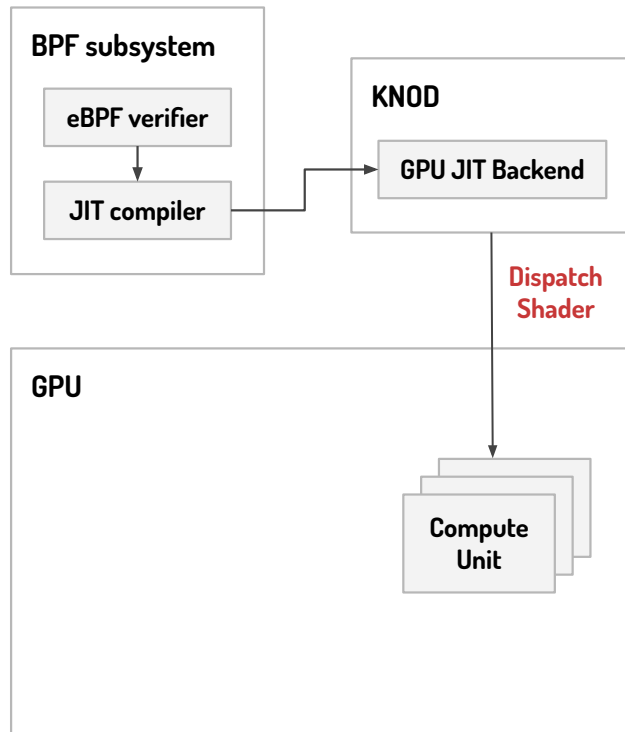


Overview

- NIC/GPU binding

- BPF-to-GPU JIT

- Place BPF Register/Stack to GPU
- Dispatch JIT-ed BPF Program Shader to GPU



Overview

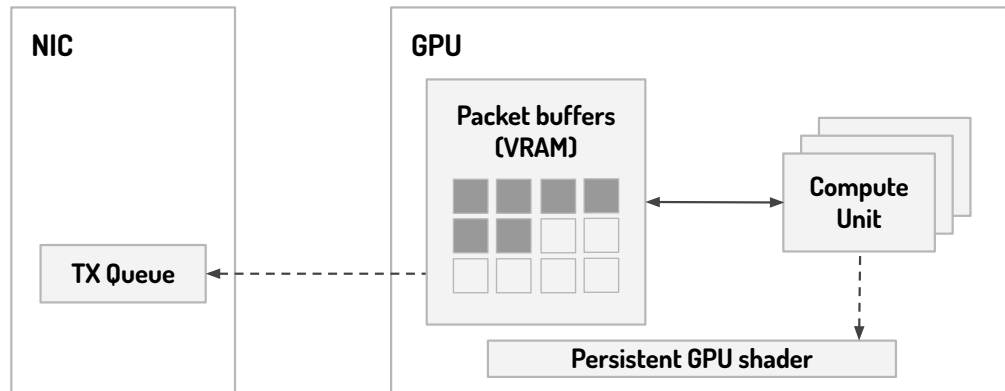
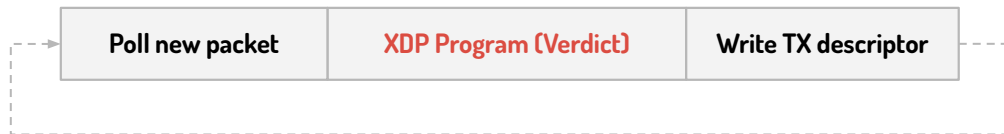
- NIC/GPU binding

- BPF-to-GPU JIT

- GPU Shader execution

- Persistent GPU Shader polls new packet
- JIT-ed XDP Program verdicts
(write TX descriptor)

Persistent GPU Shader

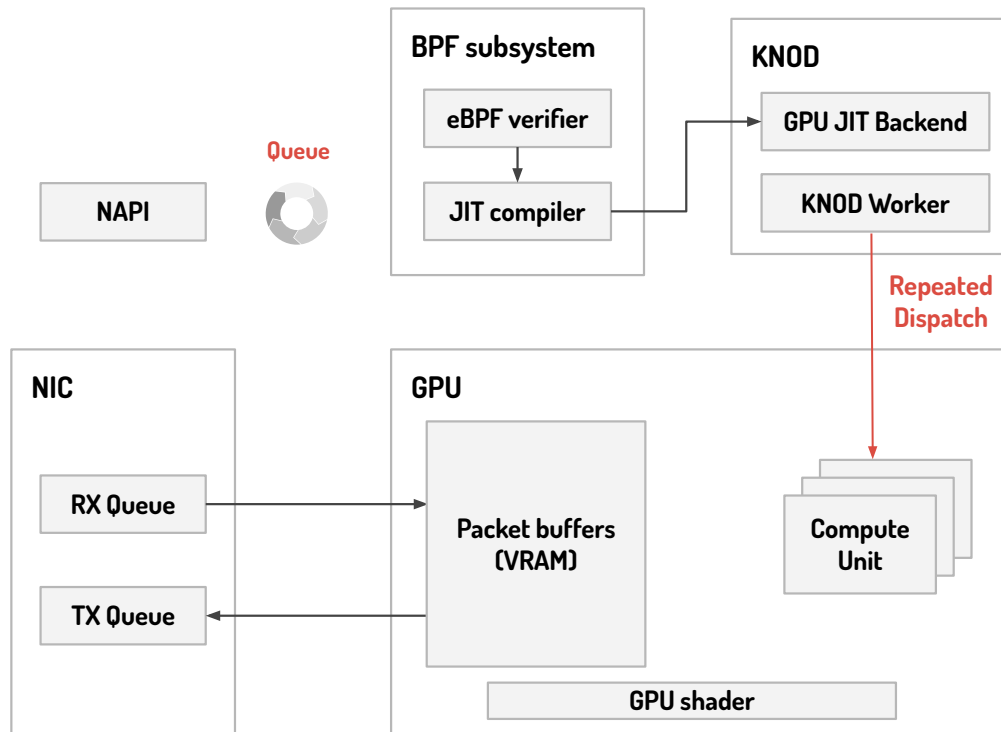


Development History

Development History

- Per-batch shader (RFC)

- NAPI handles packet flow (RX/verdict)
- Worker prepares batches and dispatches via AQL

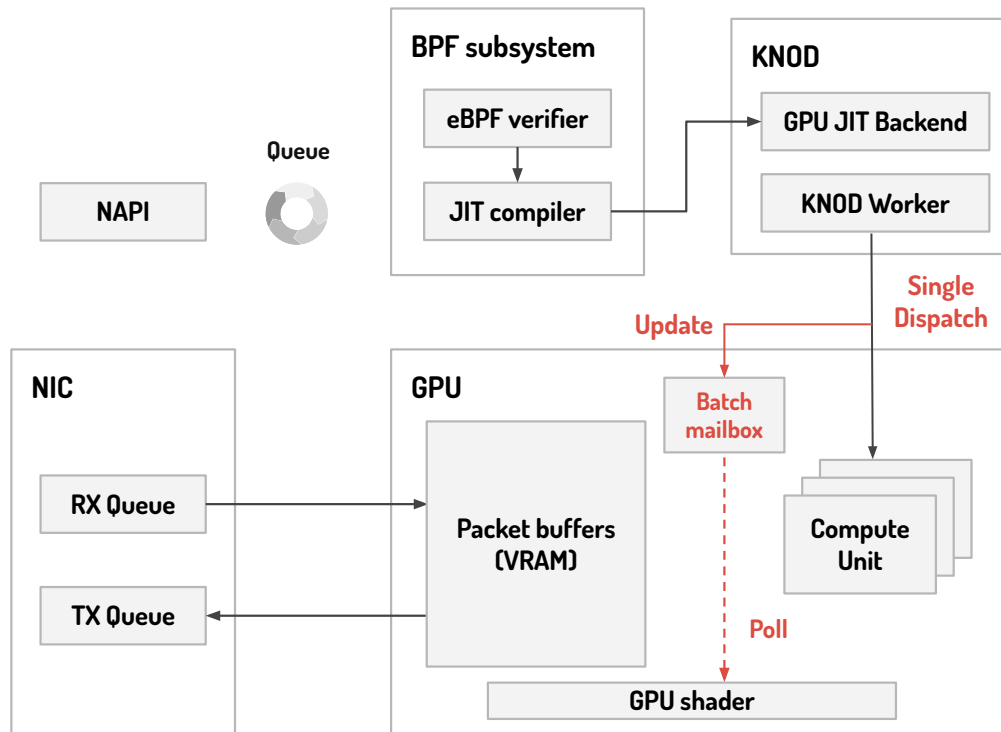


Development History

- Per-batch shader (RFC)

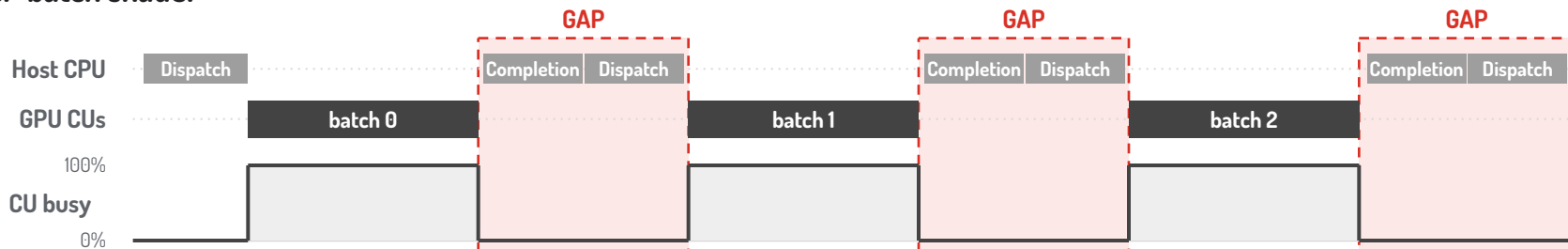
- Persistent shader

- No per-batch GPU dispatch
- CPU still handles metadata delivery and verdicts



Persistent shader: dispatch overhead

Per-batch shader



Persistent shader



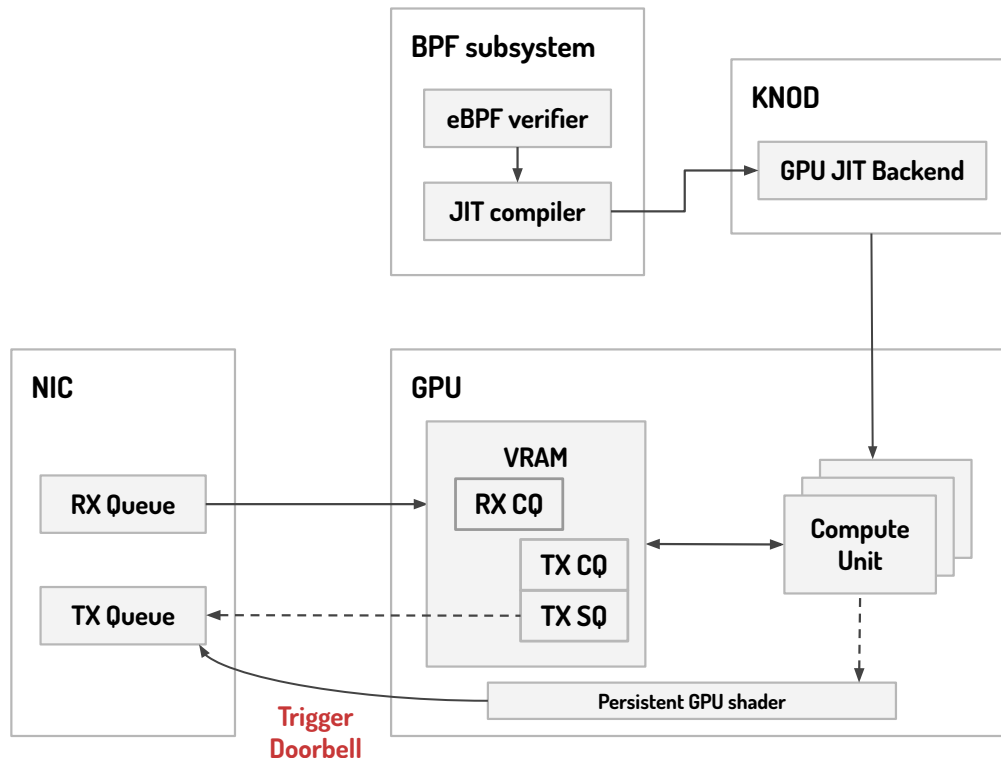
Development History

- Per-batch shader (RFC)

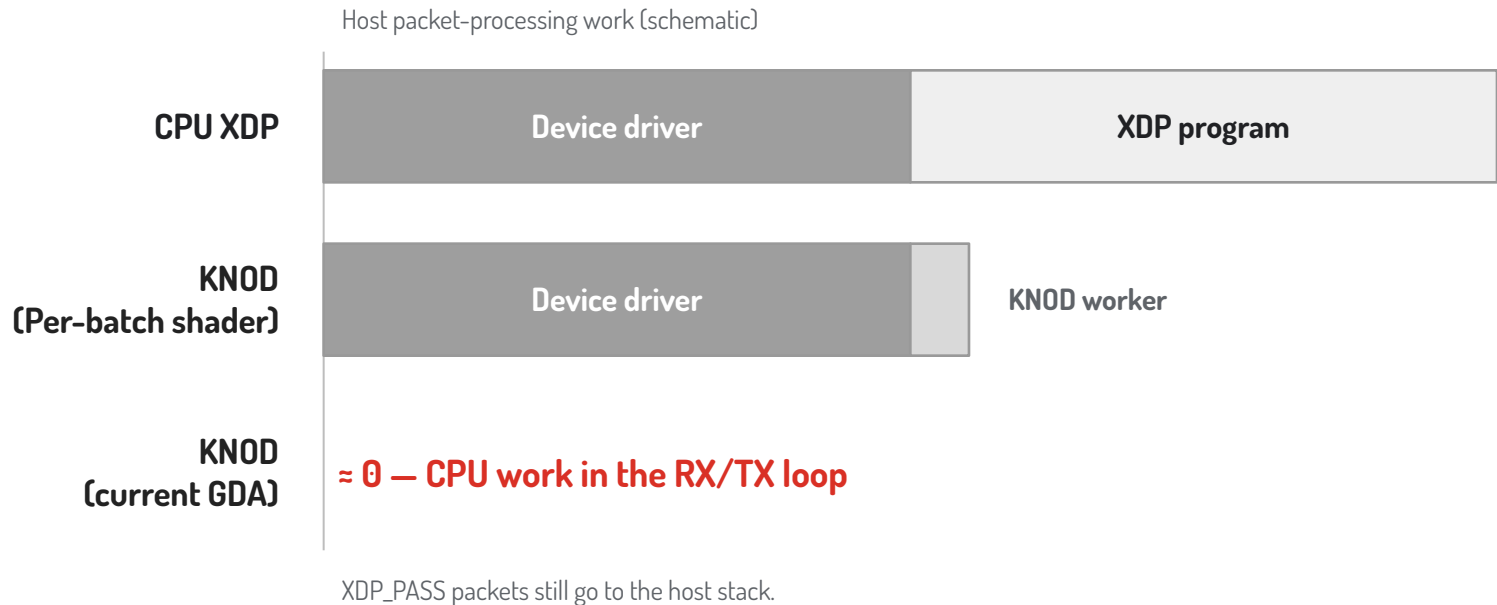
- Persistent shader

- GPU-driven packet I/O (GDA)

- Shader polls CQ, posts TX and returns RX buffers
- Remove batch worker and verdict queue
- Host handles setup, management and XDP_PASS

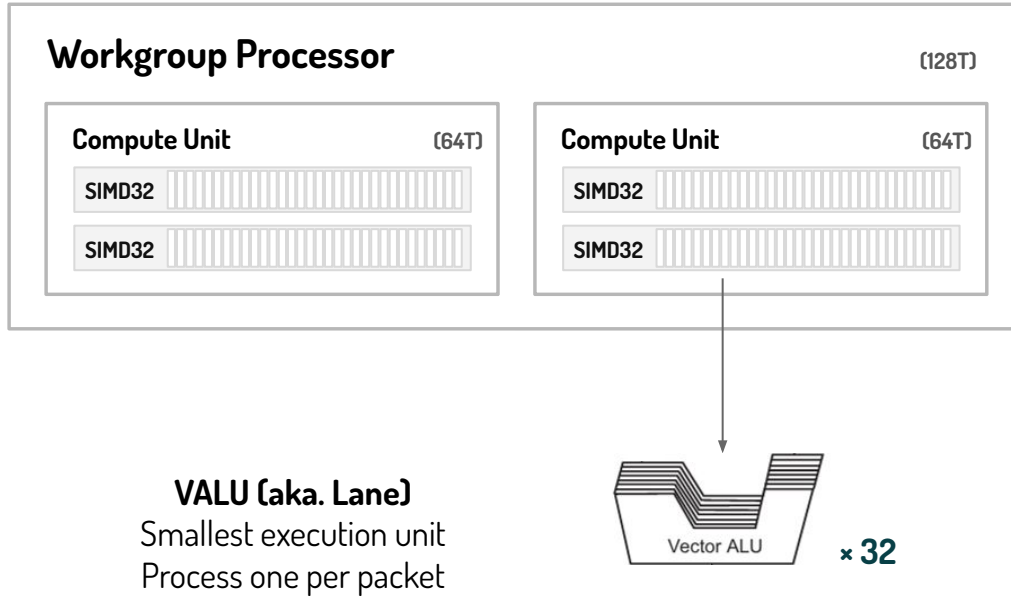


GDA Benefits

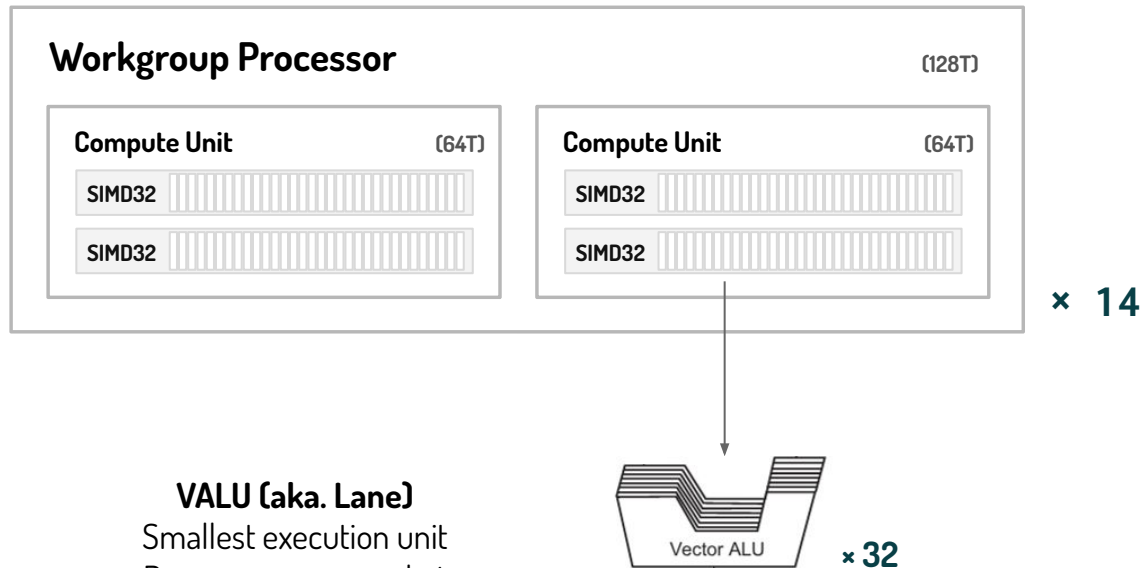


BPF-to-GPU Model

GPU execution model (ex. RDNA)



GPU execution model (ex. RDNA)

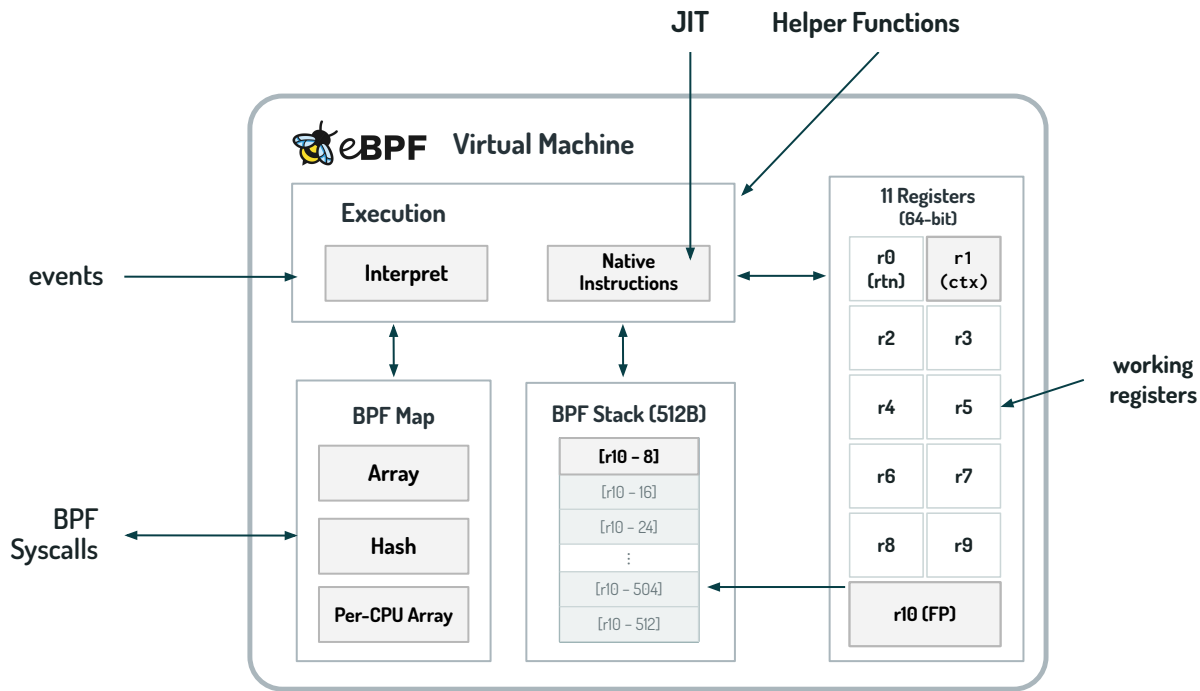


VALU (aka. Lane)
Smallest execution unit
Process one per packet

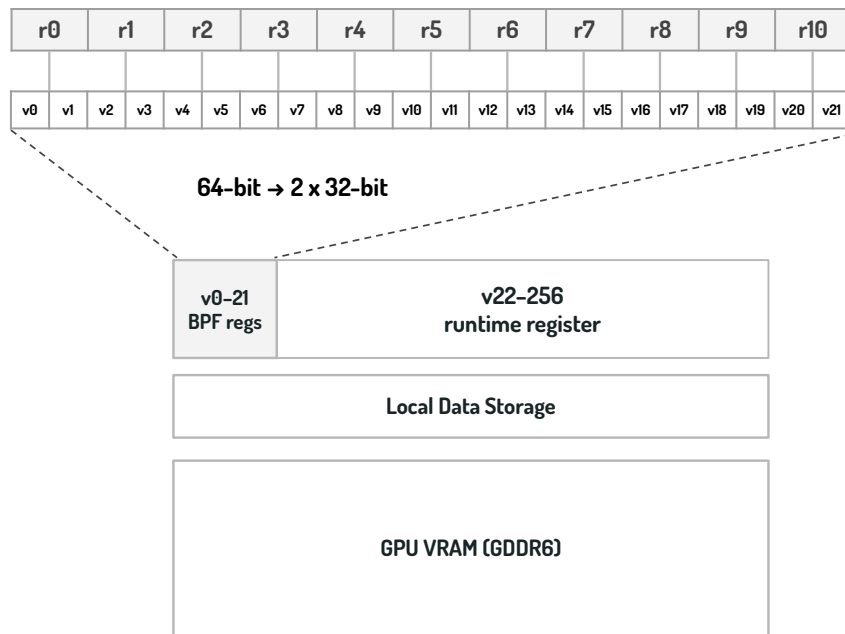
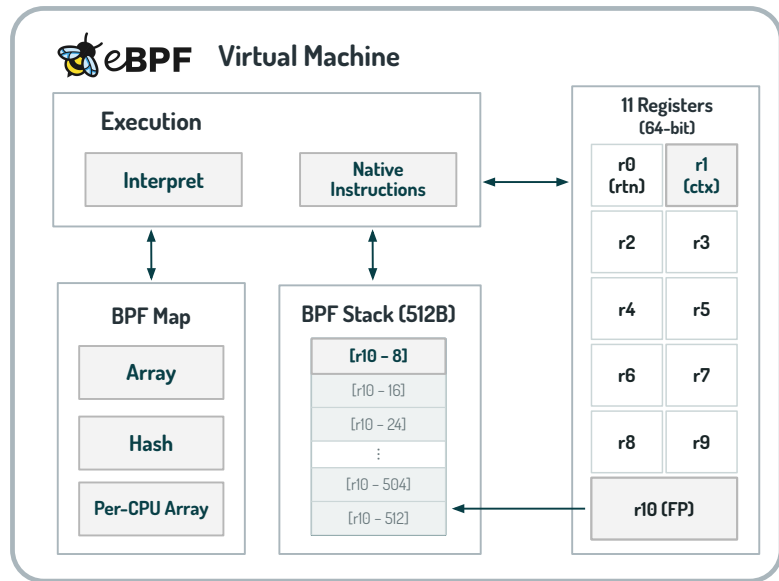
Unit	Count
Workgroup Processor	14
Compute Unit	28
SIMD32	56
Total Lane	1,792

(ex. Radeon RX 6600)

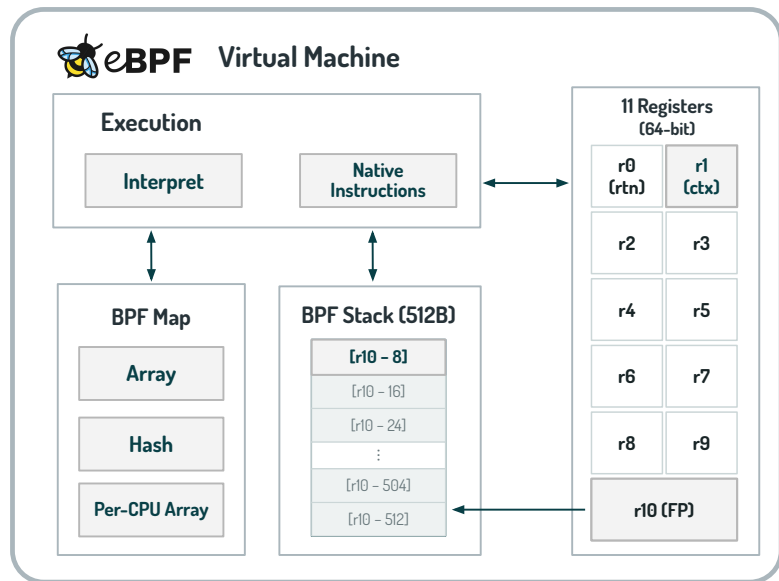
BPF execution model



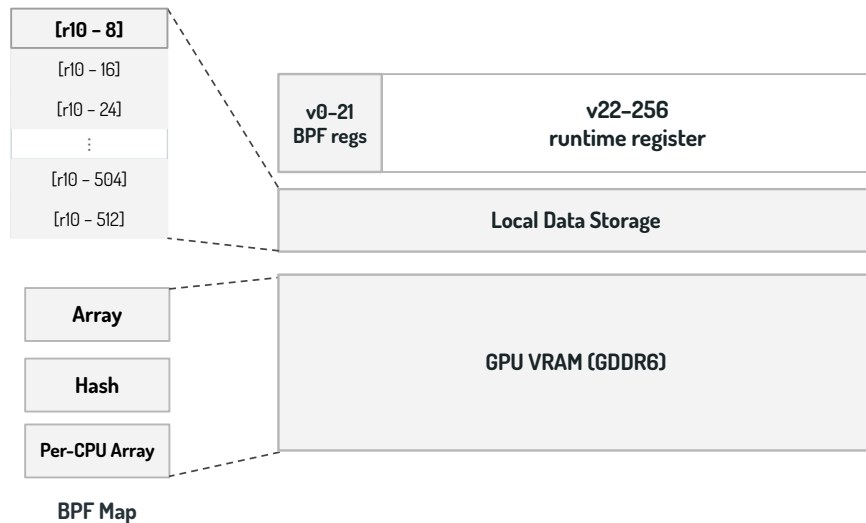
BPF execution model



BPF execution model



BPF Stack (512B)



BPF-to-GPU JIT Examples

r1 += r2

BPF_ALU64 | BPF_ADD | BPF_X

register pairs

r1:	v3	v2
-----	----	----

r2:	v5	v4
-----	----	----

r3 *= r4

BPF_ALU64 | BPF_MUL | BPF_X

r3:	v7	v6
-----	----	----

r4:	v9	v8
-----	----	----

Scratch
Register

v22

AMDGPU ISA

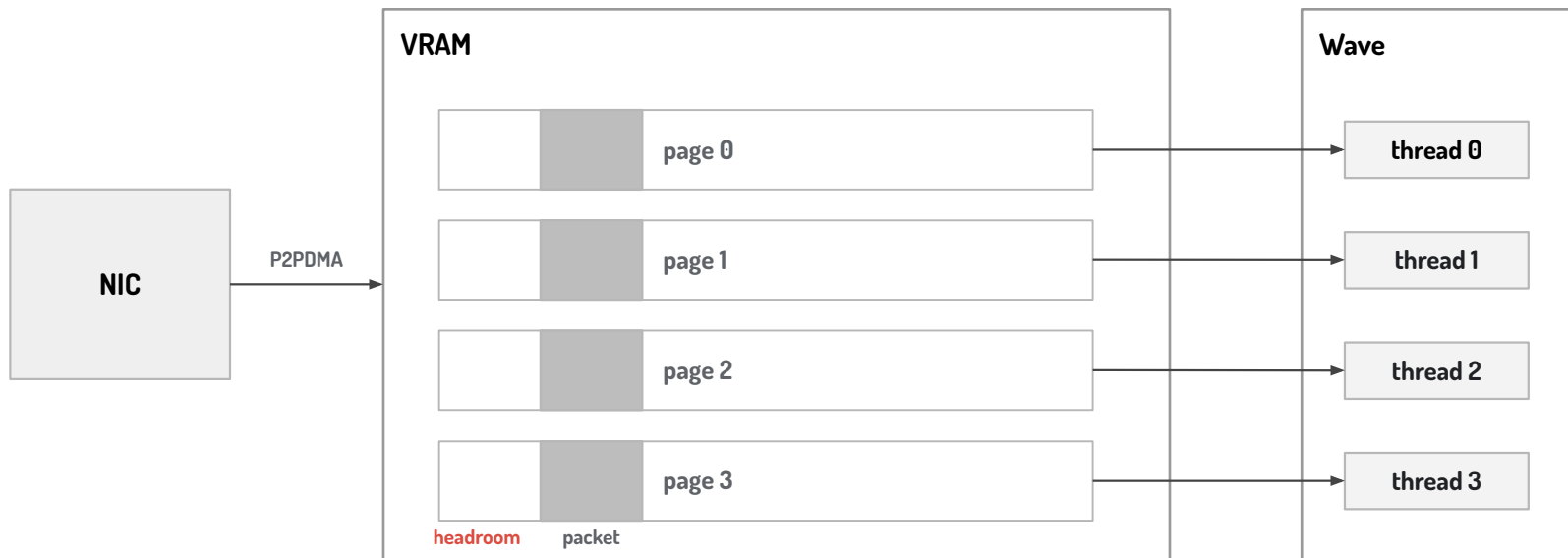
```
v_add_co_u32    v2, vcc_lo, v2, v4
v_add_co_ci_u32 v3, vcc_lo, v3, v5, vcc_lo
```

AMDGPU ISA

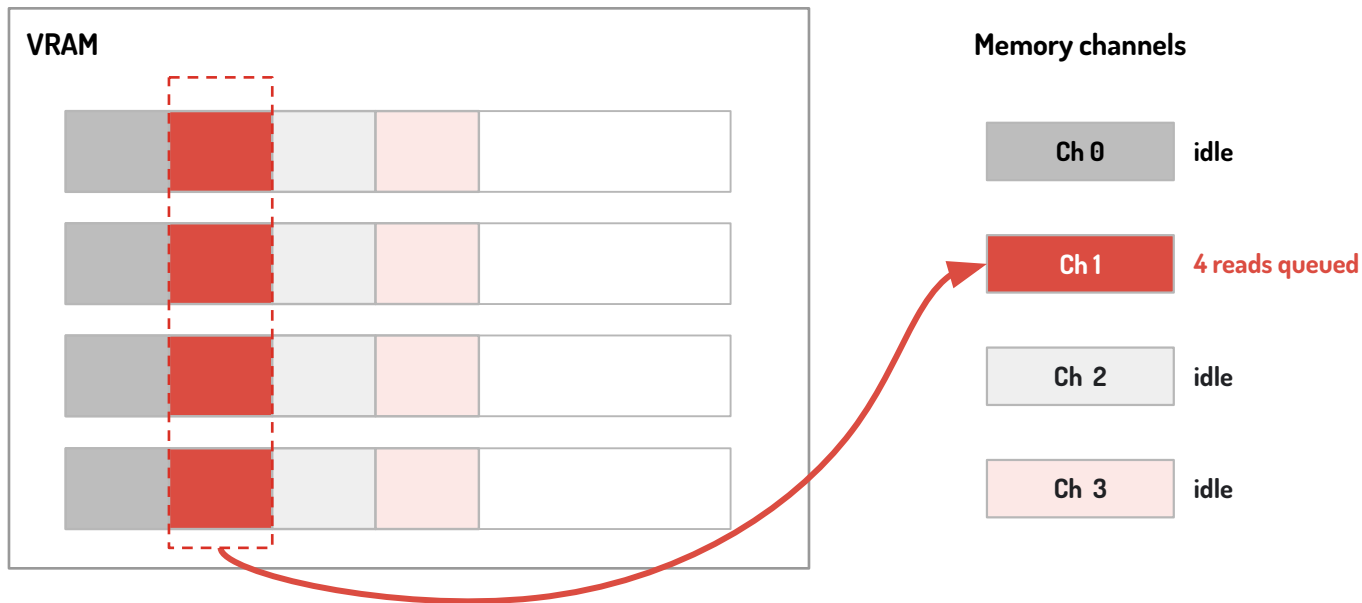
```
v_mul_lo_u32    v22, v7, v8
v_mul_hi_u32    v7,  v6, v8
v_add_nc_u32    v7,  v7, v22
v_mul_lo_u32    v22, v6, v9
v_add_nc_u32    v7,  v7, v22
v_mul_lo_u32    v6,  v6, v8
```

Implementation Details

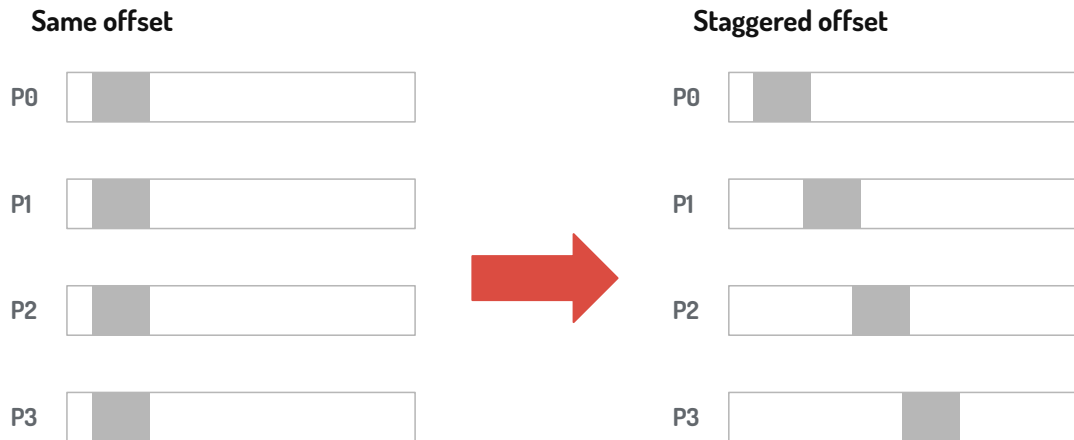
Data Plane



Memory Access Pattern



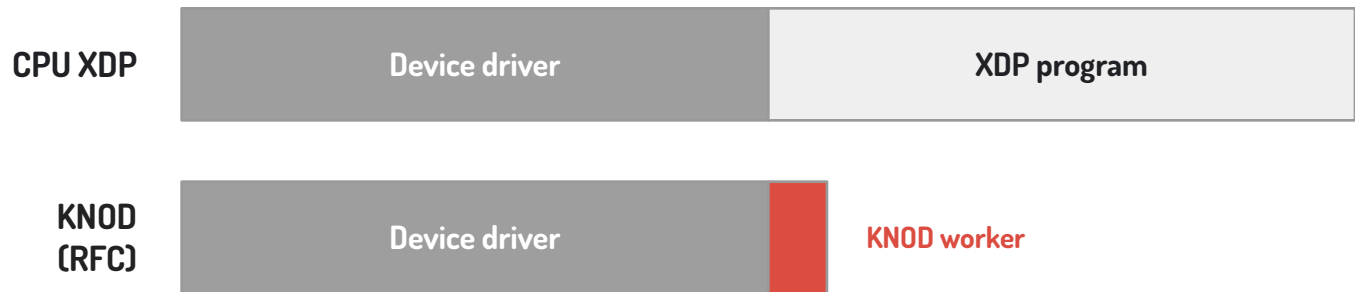
Stagger



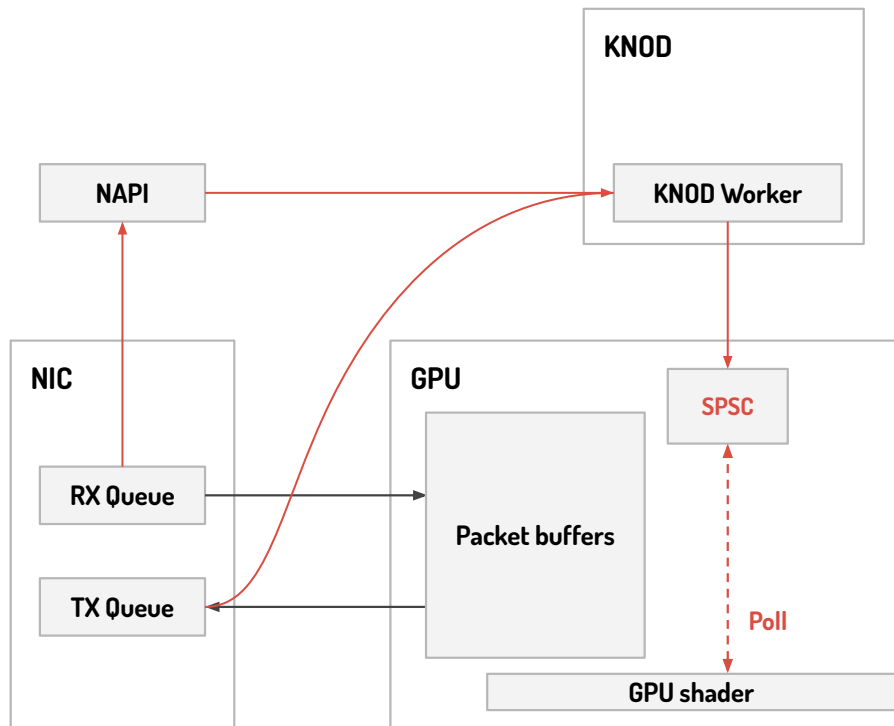
- Same header field, different page offsets
- Break the repeated low address bits
- Aim to spread reads across memory channels

CPU Utilization

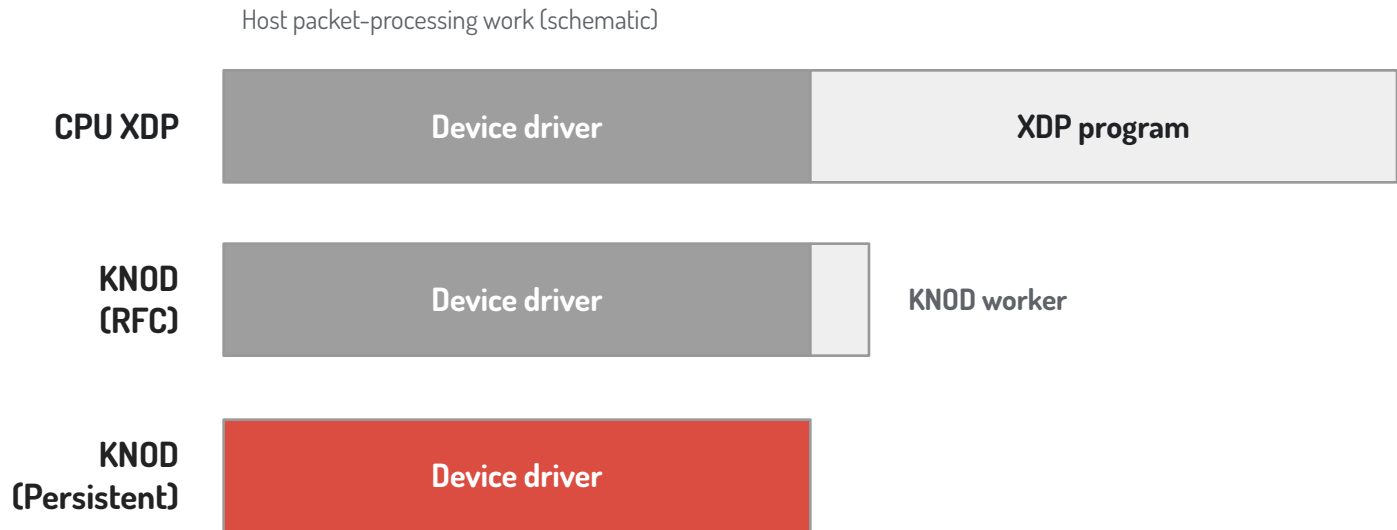
Host packet-processing work (schematic)



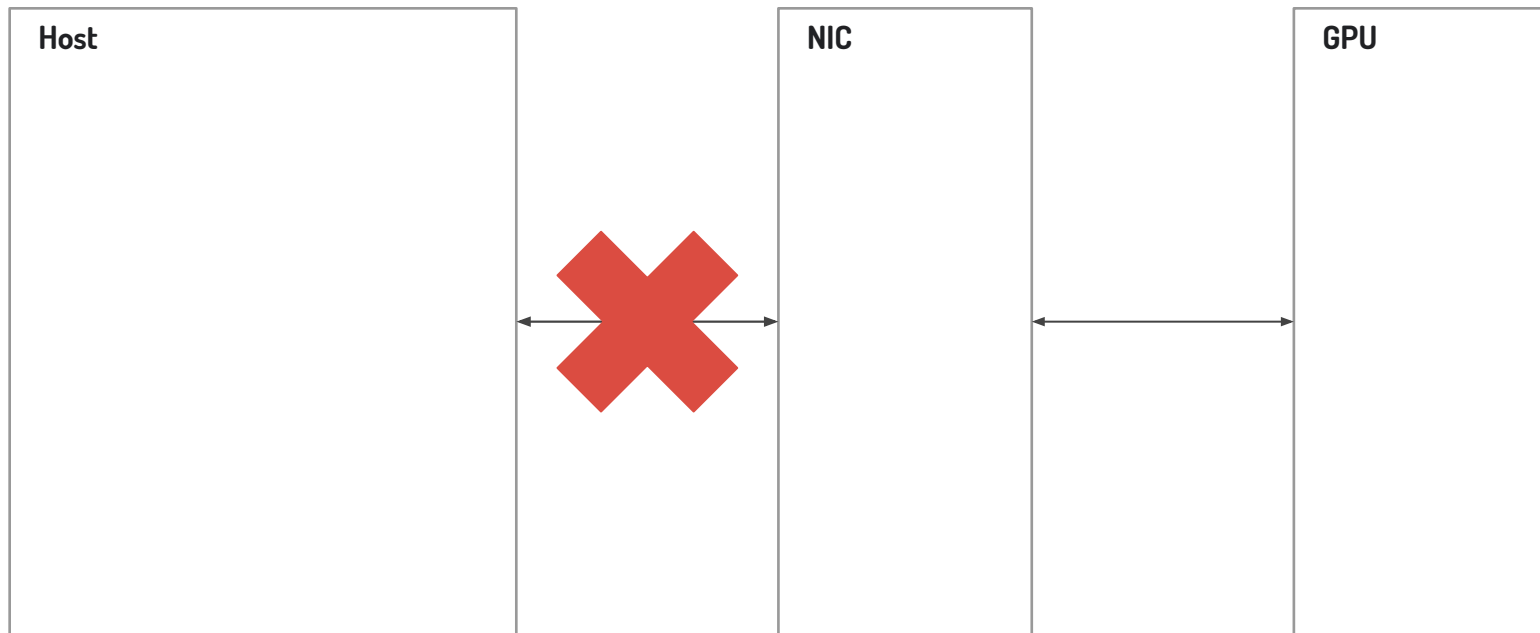
Persistent Shader



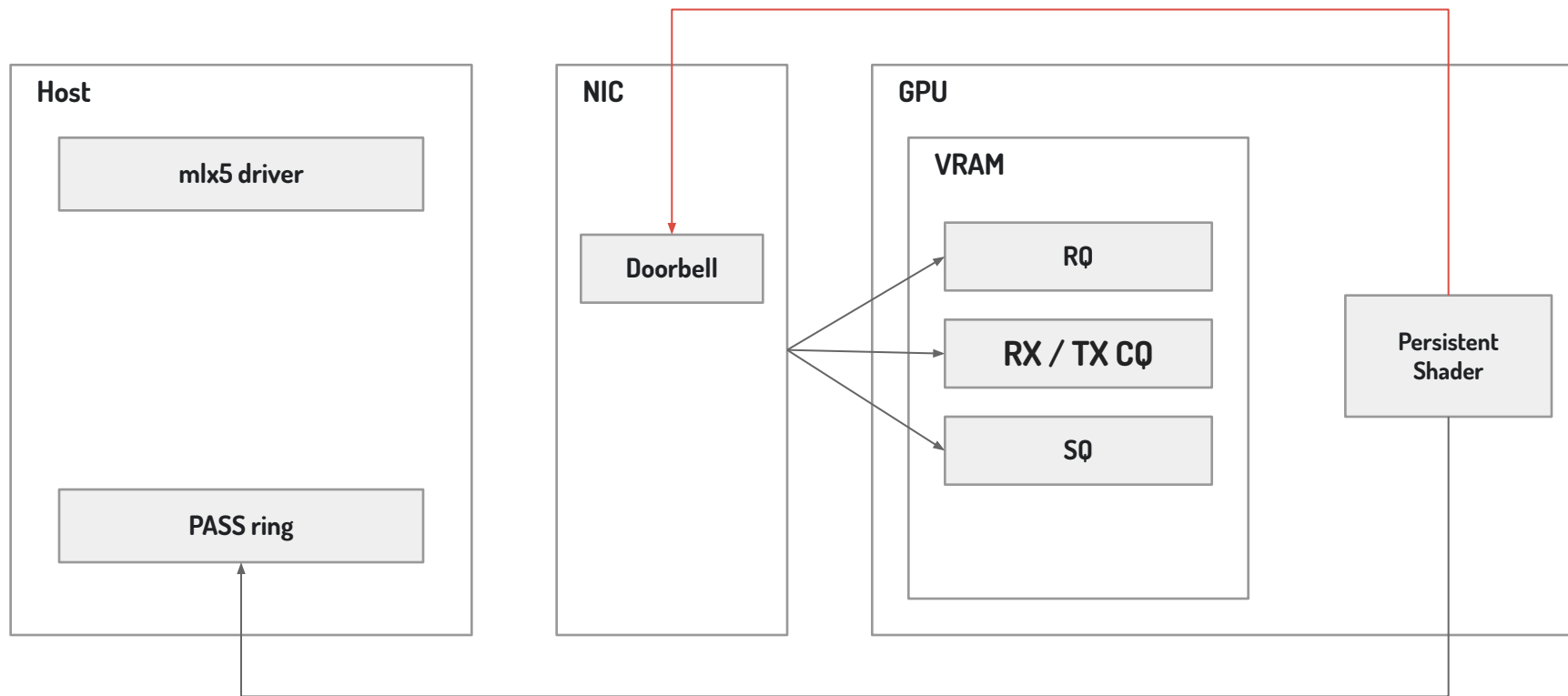
CPU Utilization



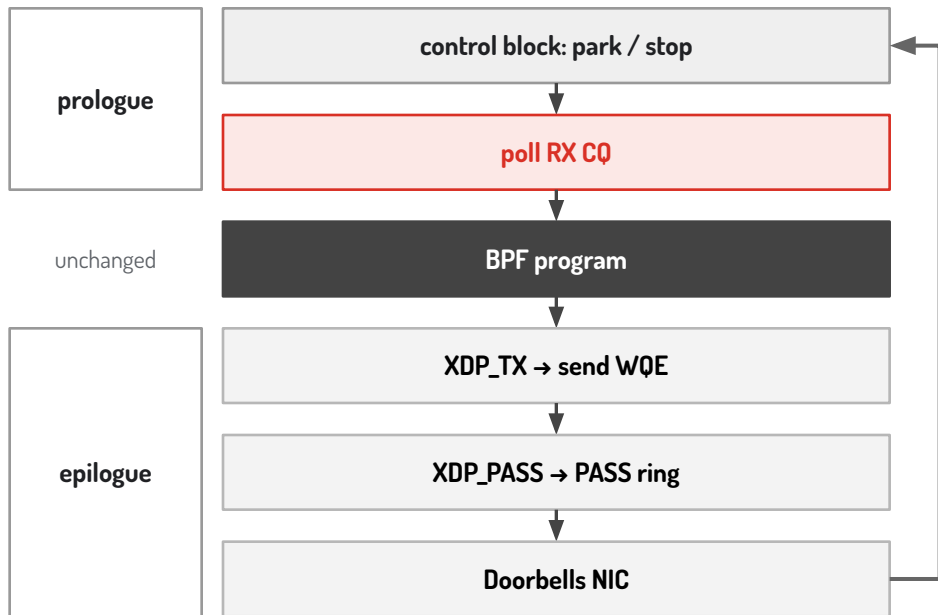
GDA (GPUDirect-Async)



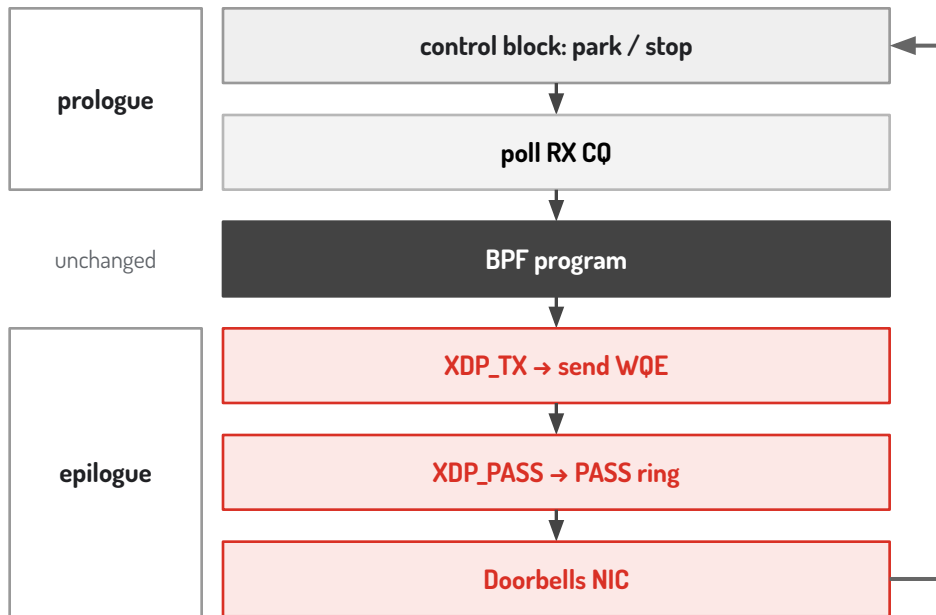
GDA (GPUDirect-Async)



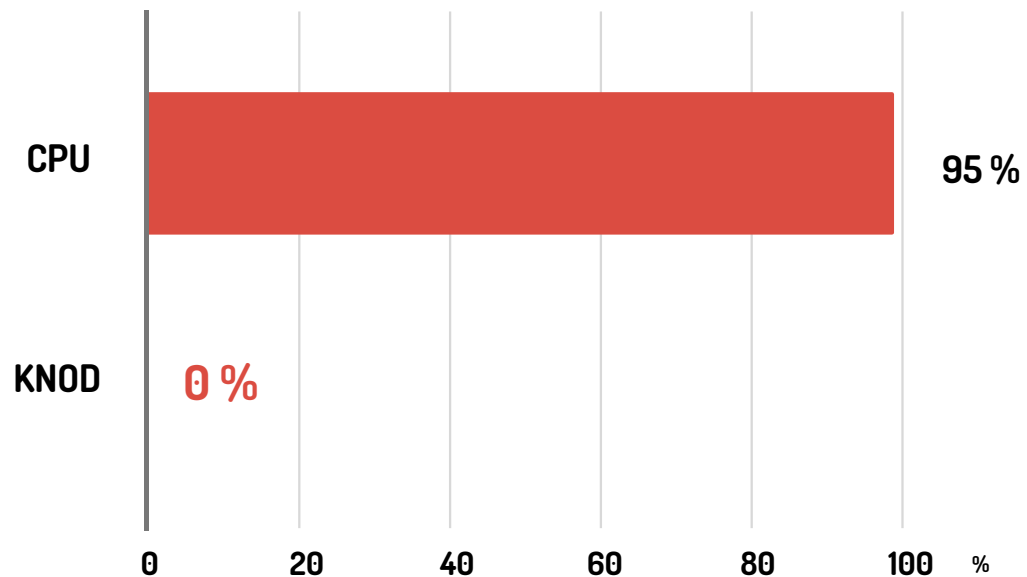
GDA in the Persistent shader



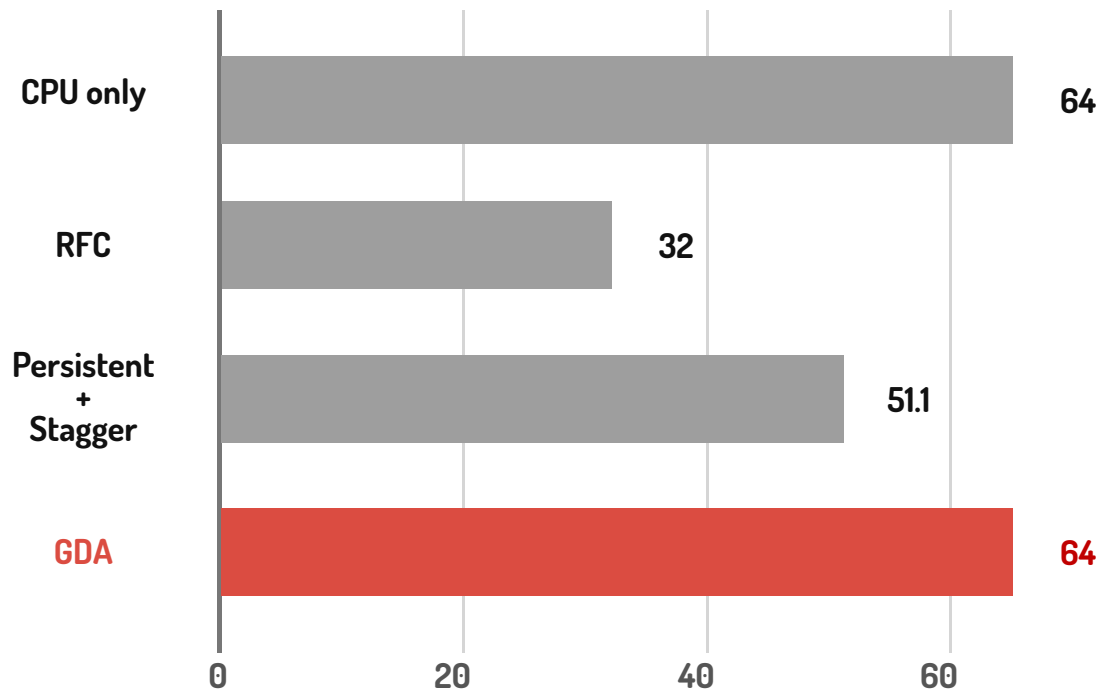
GDA in the Persistent shader



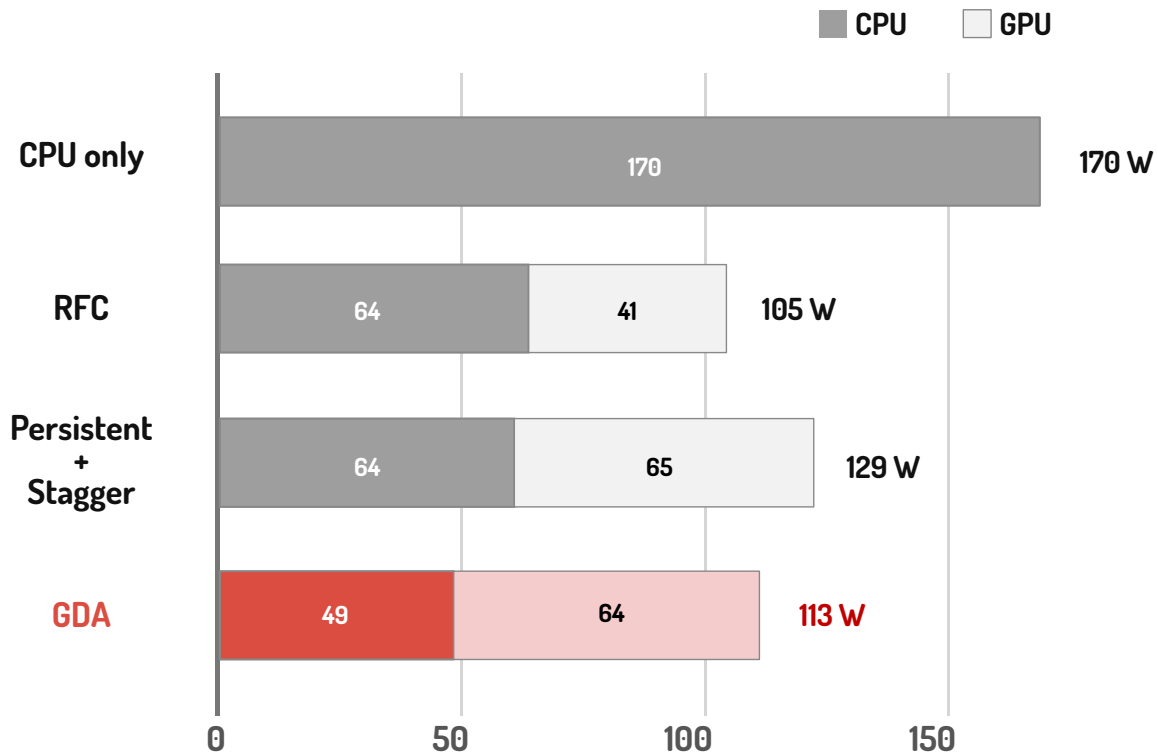
CPU Utilization



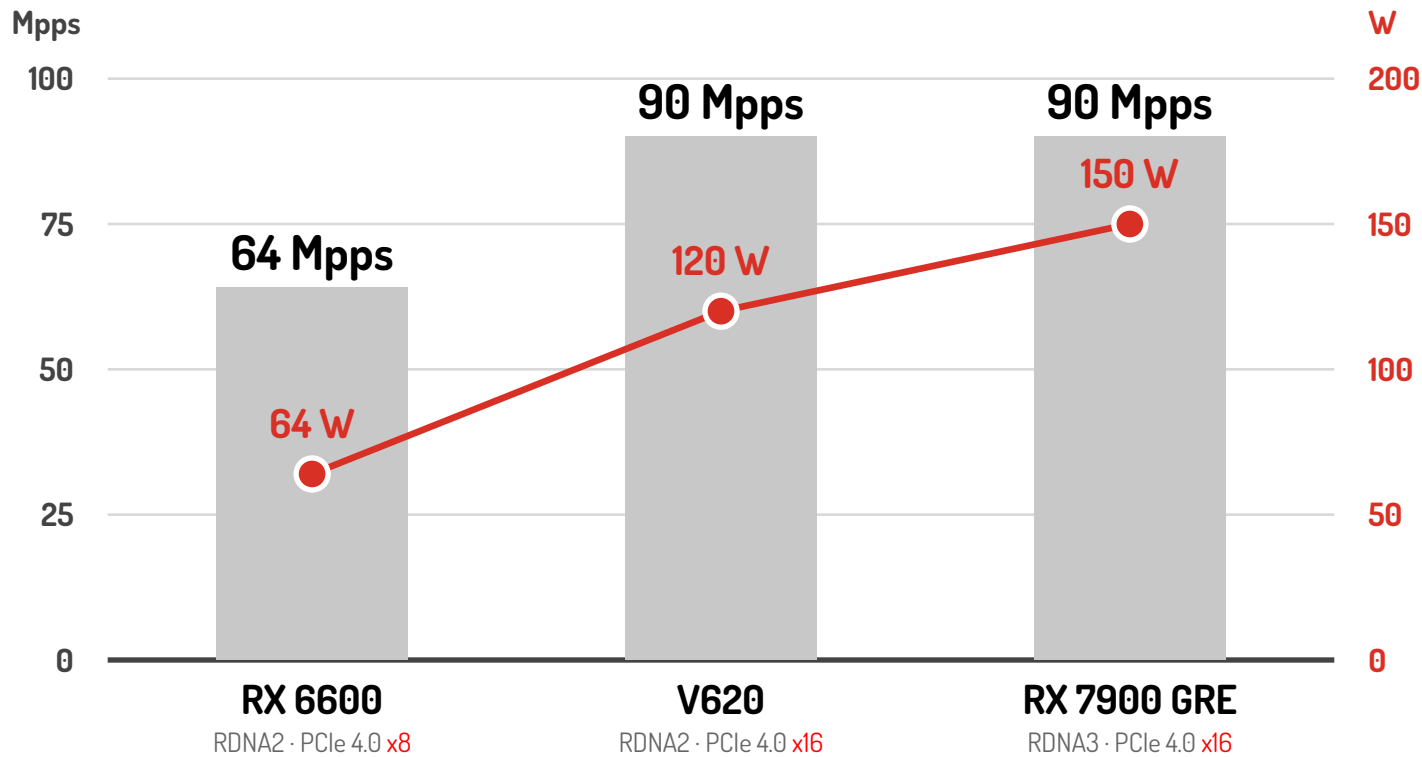
Throughput (Mpps, RX6600)



Power consumption (W)



Throughput and power by GPU



Future Work

Open Questions

Power Consumption

- Persistent shader consumes ~24–26 W on idle

XDP_REDIRECT

- No redirect is supported.

Performance Bottleneck

- Ceiling: 64 Mpps (x8), ~90 Mpps (x16)
- **PCIe switch / root complex?** (limit on PCIe TLP message rate)
- **NIC feature enable?** (compressed CQE or MPWQE)

TODO

Other Vendor

- GPU: Intel Arc graphics
- NIC: Broadcom NIC (bnxt), Intel NIC (ice)
- APU: CPU with Integrated GPU

References

Project development github.com/TaeheeYoo/knod

RFC Proposal lore.kernel.org/netdev/20260719175857.4071636-1-ap420073@gmail.com

Phoronix Article phoronix.com/news/KNOD-Network-Offload-AMD-GPUs

LPC 2025 lpc.events/event/19/contributions/2267/

Interested? Join us!

Q & A

LINUX PLUMBERS CONFERENCE
PRAGUE, CZECHIA / OCT. 5-7, 2026

