Packets are Not Pages: Flow-Based Addressing Conserves Memory Bandwidth

(submitted to CoNEXT'26)

Agur Adams, Hui Sub Shim, Colin Drewes, Ben O'Keefe, Philip Levis and Zakir Durumeric

Stanford University

DAM Research Group Meeting

21 January, 2026

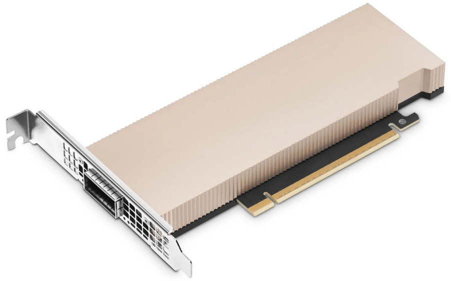
Motivation

Network line rates have significantly increased

40 Gbps
2009

→

800 Gbps
2024



*Images from Memory4Less and FS

Motivation

	Cores	NIC	DRAM	DRAM BW per core	NIC BW per core	Ratio
Google Cloud C3 2x Sapphire Rapids	176	200 Gbps	2x 8-ch DDR5	3.49 GB/s	0.14 GB/s	**24.93**
BlueField-3 SmartNIC DDR5	16	400 Gbps	2-ch DDR5	5.60 GB/s	3.13 GB/s	**1.79**

*Table data from *Lovelock: Towards Smart NIC-hosted Clusters* (2024) [1]

SmartNICs have limited memory bandwidth relative to their line rates

- Copying data to reassemble application bytestreams is expensive
- Careful cache management required

Motivation

Memory bottleneck on SmartNICs is due to a mismatch of abstractions

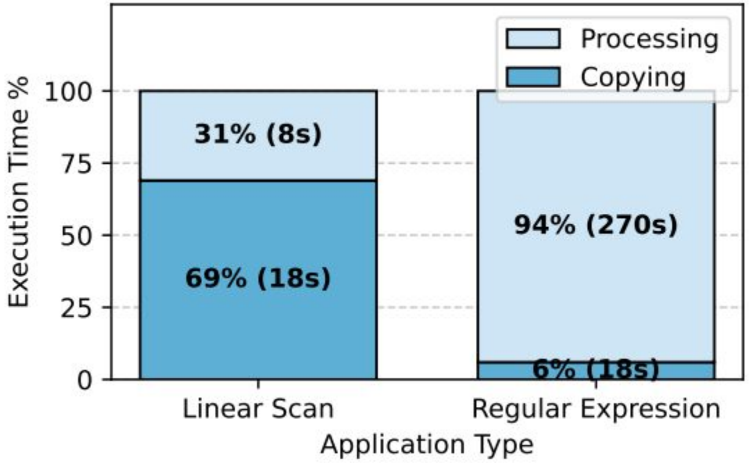
- Pages have a fixed size and their structures depends on high locality through a hierarchy of page tables
- Packets have variable size and low locality as a link linterleaves packets from many flows (*Fast & Safe IO Memory Protection* 2024 and *Ensō* 2023)
- Application code spends almost all its cycles and memory bandwidth accessing packets, but its memory is organized in terms of pages.

SmartNICs have limited memory bandwidth relative to their line rates

- Copying data to reassemble application bytestreams is expensive
- Careful cache management required

Motivation

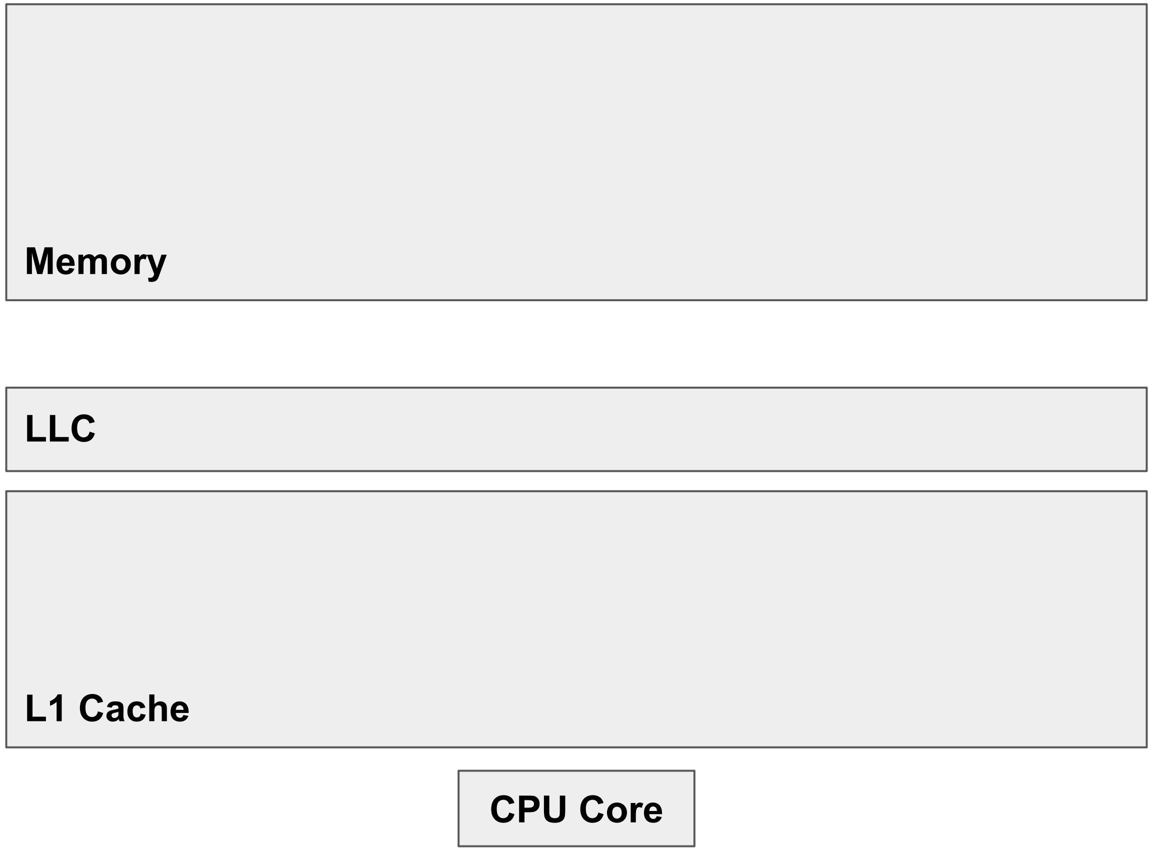
- Sent 64 million 1,518-byte packets at line rate of 200 Gbps
- Measured time spent copying packets in two common bytestream analysis tasks: 1) **linear scan** (memory-bound) and 2) **regular expression** search (compute-bound)
- Two NVIDIA BlueField-3 DPUs each with a 16 Armv8.2+ A78 Hercules cores and 32 GB of DDR5
- One core used on each side to eliminate intercore contention



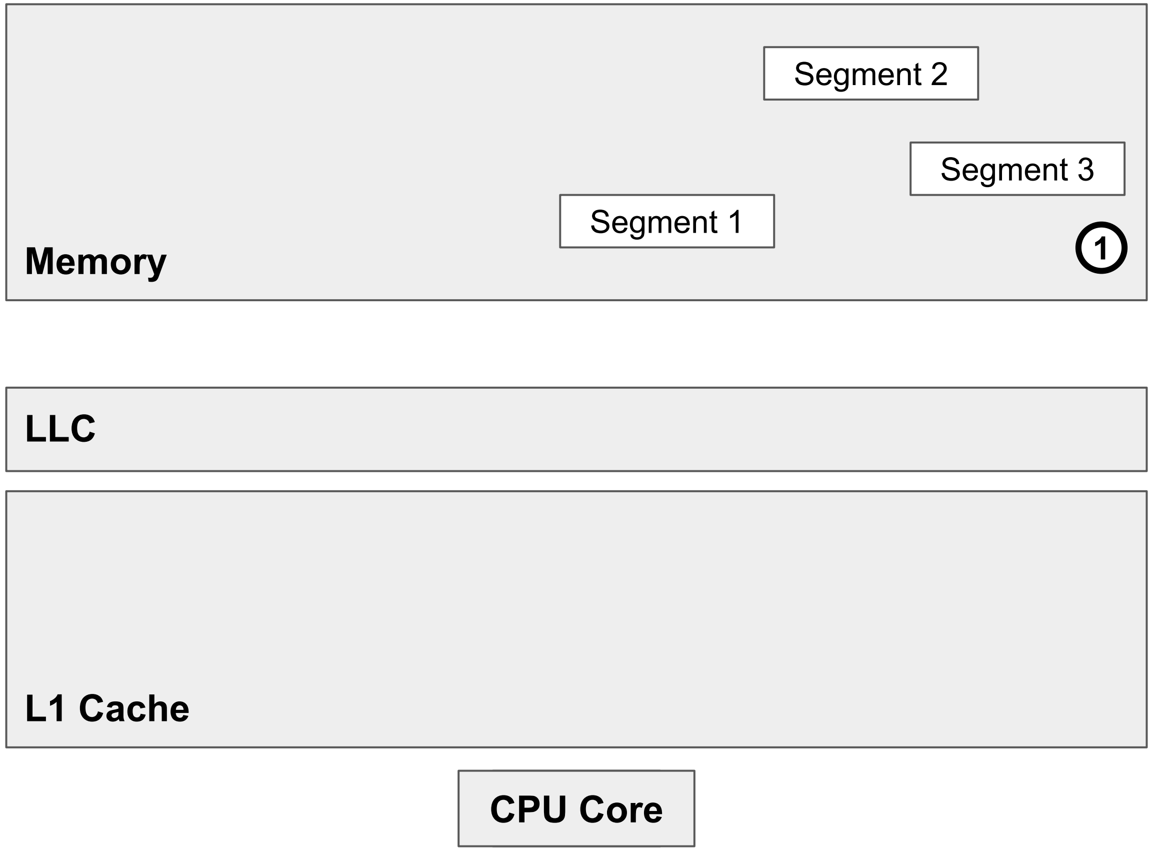
Overview

- While link speeds continue to increase, memory bandwidth, in comparison, remains stagnant
- Applications that inspect or process transport layer data (e.g., HTTP traffic, RPC requests), quickly run into memory bandwidth limitations on SmartNICs
- **Packets are not pages:** the memory bottleneck on SmartNICs is due, in part, to a mismatch of abstractions
- Propose a new flow-based addressing scheme and last-level cache controller for SmartNICs to present reassembled segment payloads as contiguous memory regions for efficient on-NIC computations
- Results show 5-6$\times$ reduction in DRAM bandwidth, 25.33-61.96% lower execution time, and 1.33-2.62$\times$ increase in per-core throughput

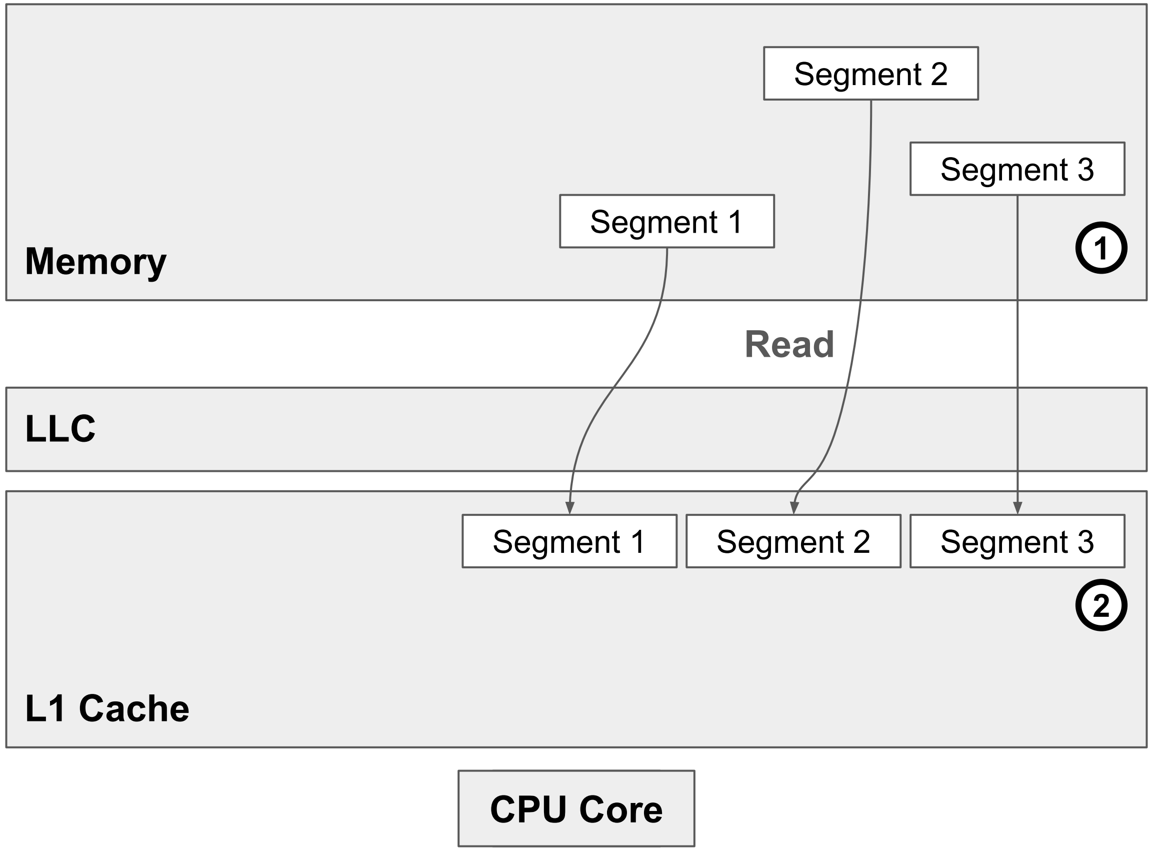
Problem



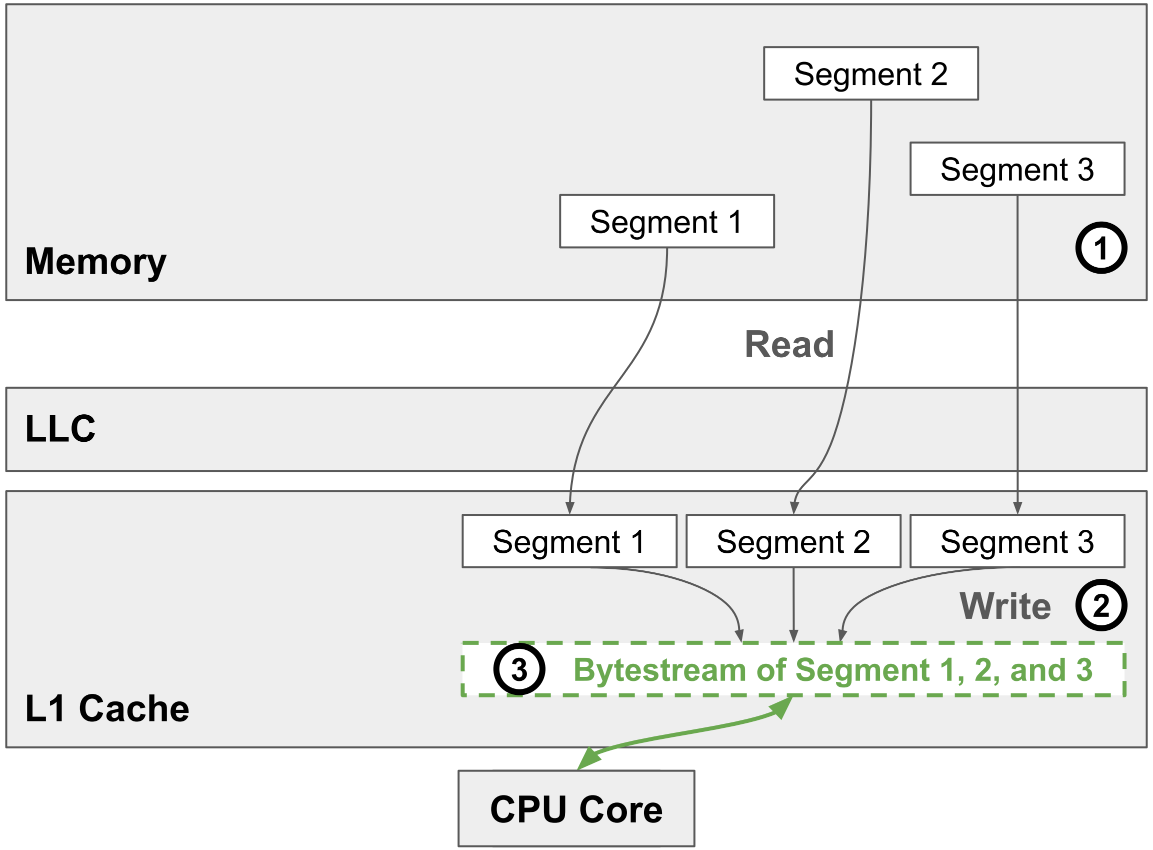
Problem



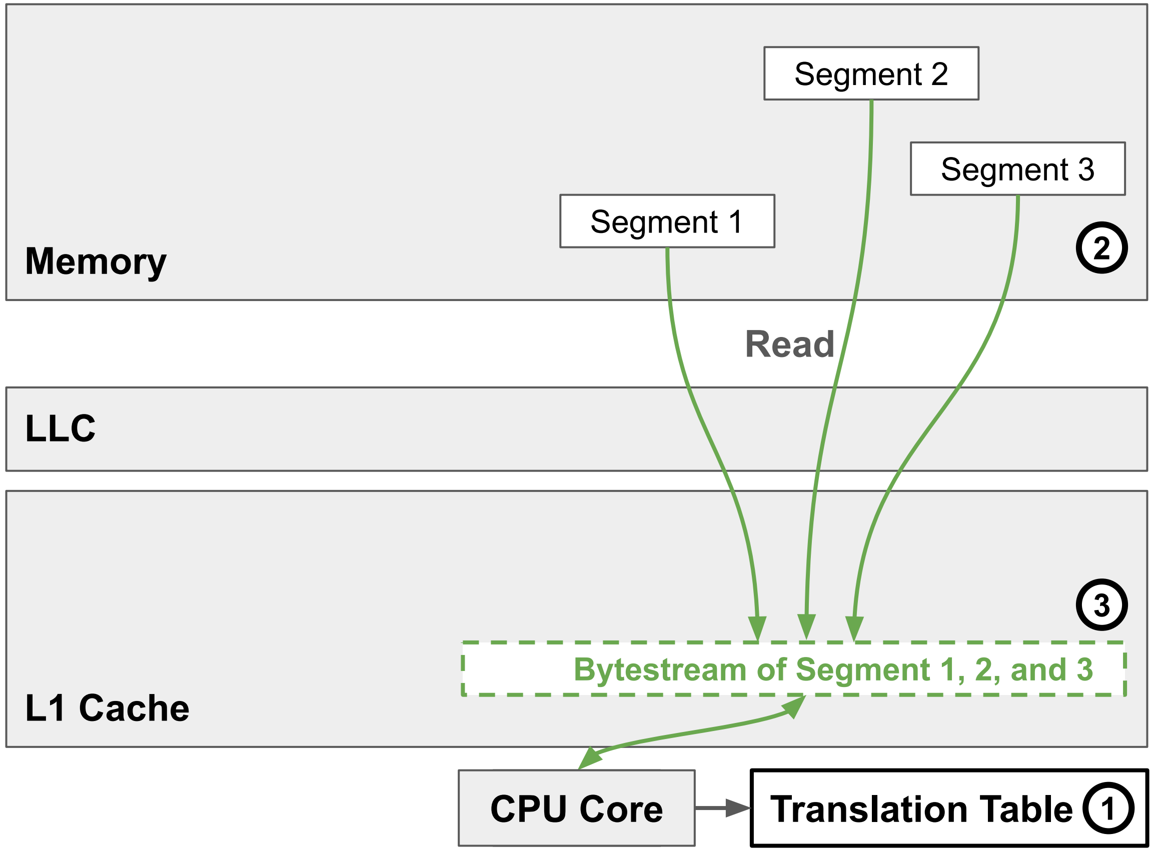
Problem


Segment 2
Segment 3
Segment 1
Memory
1
Read
LLC
Segment 1
Segment 2
Segment 3
2
L1 Cache
CPU Core

Problem


Segment 2
Segment 3
Segment 1
Memory
1
Read
LLC
Segment 1
Segment 2
Segment 3
Write
2
3
Bytestream of Segment 1, 2, and 3
L1 Cache
CPU Core

Solution

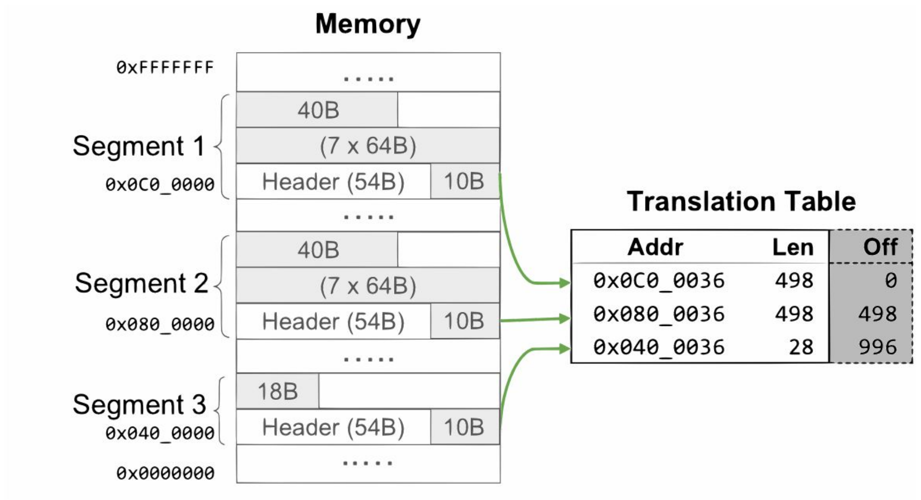


Segment 2
Segment 3
Segment 1
Memory
2
Read
LLC
3
Bytestream of Segment 1, 2, and 3
L1 Cache
CPU Core
Translation Table
1

Flow-Based Addressing

Goal: Make the payloads of a series of packets addressable as a contiguous regions in memory without a copy

- SmartNIC populates and manages a translation table of all transport-layer segments of a flow
- Table stores physical address, length and offset into the bytestream
- These fields let the SmartNIC CPU map any byte position in the flow to its location in memory



Addr	Len	Off
0x0C0_0036	498	0
0x080_0036	498	498
0x040_0036	28	996

FPGA Implementation

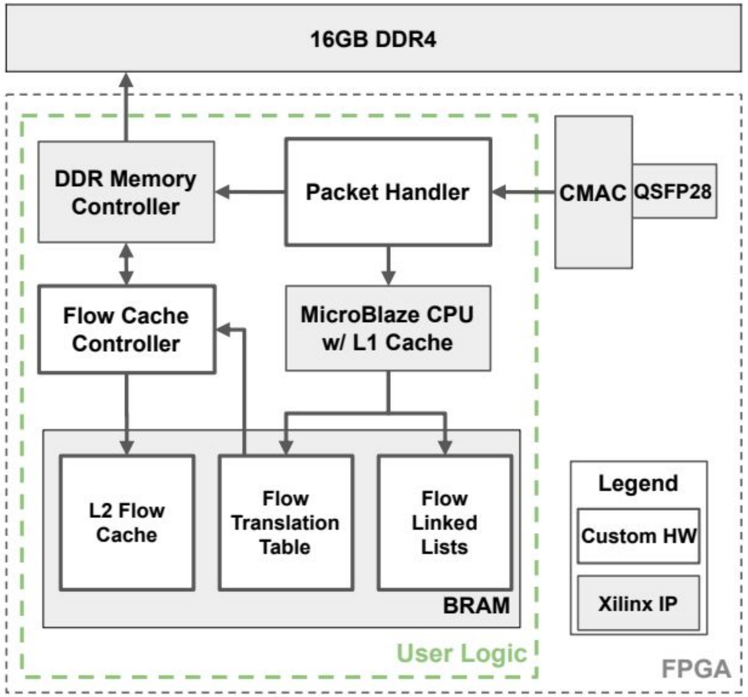
Alveo U250 FPGA-Based SmartNIC with AMD's OpenNIC framework and MicroBlaze 64-Bit RISC CPU with L1 cache

Hardware modules:

1) **Packet Handler** for line-rate packet reception and metadata extraction
2) **Flow cache controller** for retrieving unaligned flow data from DRAM into flow cache

Data structures:

1) Flow-linked list with per-flow metadata
2) Translation table that maps logical flow offsets to physical memory addresses
3) L2 flow cache stores assembled bytestream



FPGA Implementation

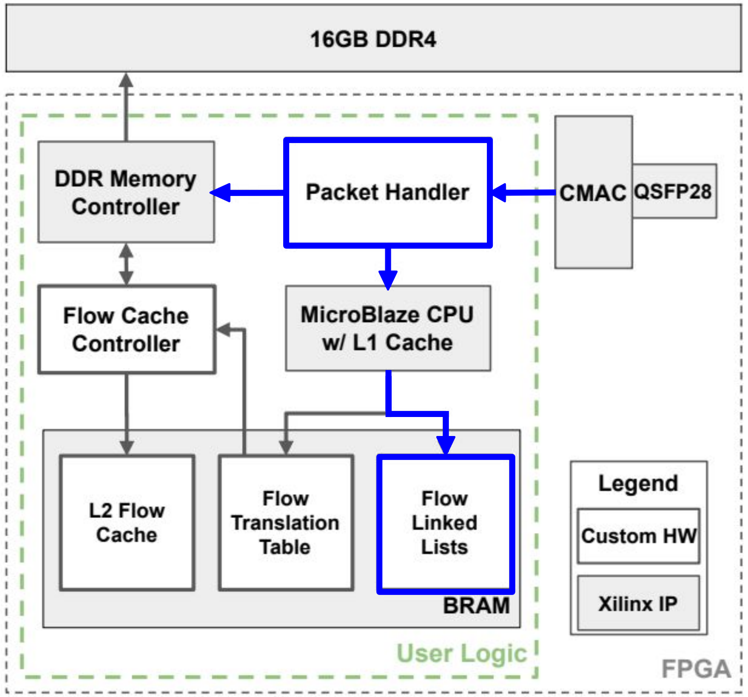
Alveo U250 FPGA-Based SmartNIC with AMD's OpenNIC framework and MicroBlaze 64-Bit RISC CPU with L1 cache

Hardware modules:

1) **Packet Handler** for line-rate packet reception and metadata extraction
2) **Flow cache controller** for retrieving unaligned flow data from DRAM into flow cache

Data structures:

1) **Flow-linked list** with per-flow metadata
2) Translation table that maps logical flow offsets to physical memory addresses
3) L2 flow cache stores assembled bytestream



FPGA Implementation

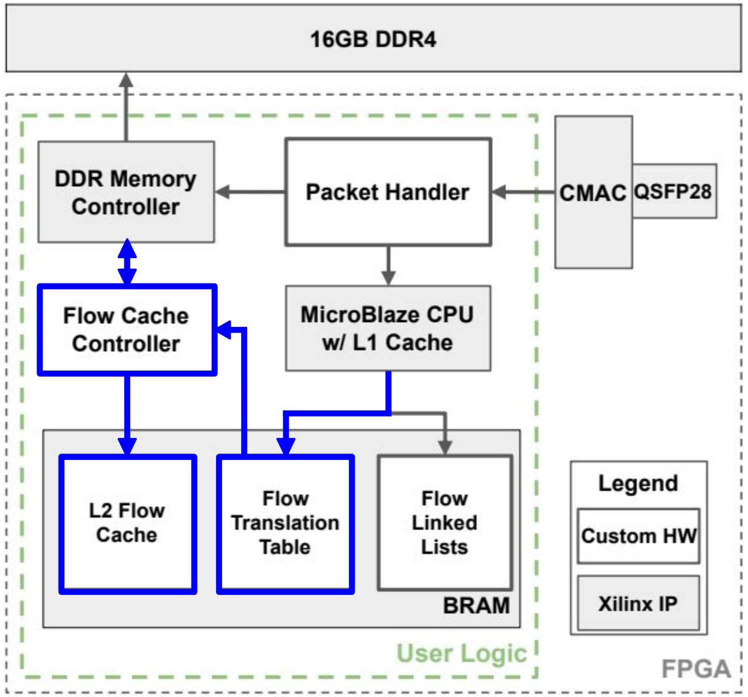
Alveo U250 FPGA-Based SmartNIC with AMD's OpenNIC framework and MicroBlaze 64-Bit RISC CPU with L1 cache

Hardware modules:

1) **Packet Handler** for line-rate packet reception and metadata extraction
2) **Flow cache controller** for retrieving unaligned flow data from DRAM into flow cache

Data structures:

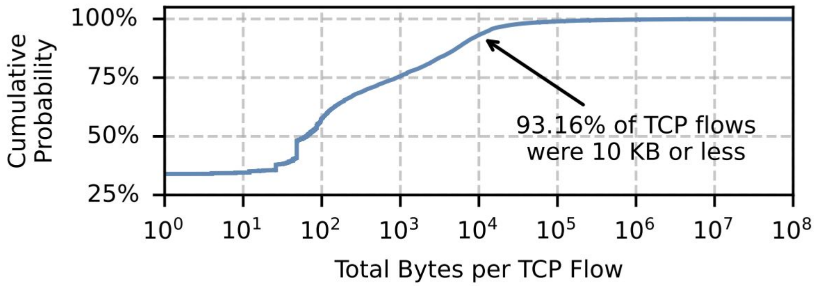
1) Flow-linked list with per-flow metadata
2) **Translation table** that maps logical flow offsets to physical memory addresses
3) **L2 flow cache** stores assembled bytestream



FPGA Implementation Challenges

1) How large should the flow cache and the flow translation table be?

Measured over 1.6 billion flows over four hours; **93.16% TCP flows are ≤ 10 KB**



2) How do you dealing with unaligned accesses and maintain throughput?

Built **custom data realignment engine** for byte extraction and sequential writing

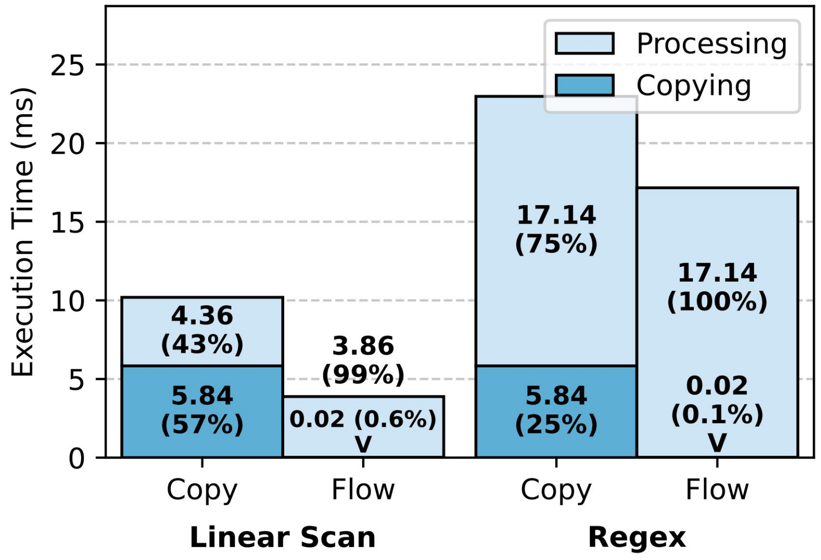
3) What is the right strategy for flow retrieval and managing packet metadata?

Configurable strategies for selective retrieval and/or whole bytestream

Evaluation

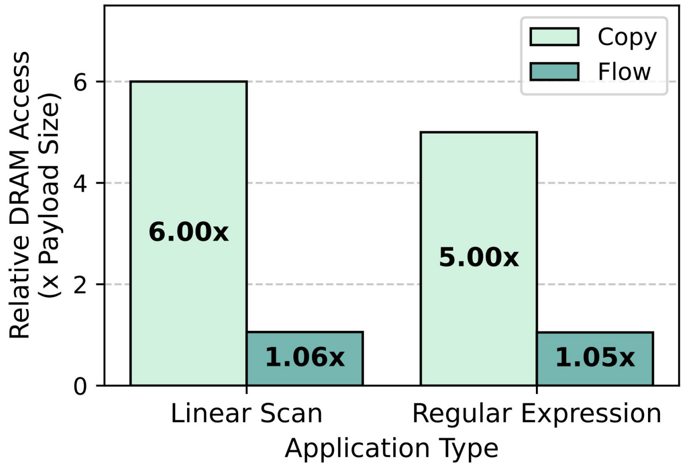
Microbenchmarks of 10 fixed size 1,510 byte packets with `pktgen`

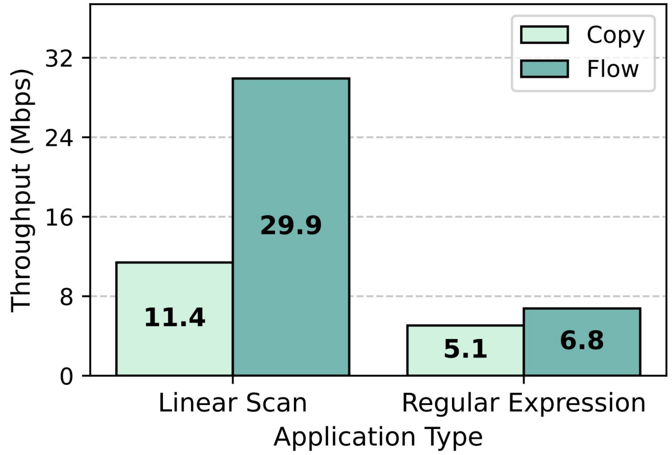
- **Copying falls** from 5.84 ms to 0.02 ms (99.66%) as the flow cache controller builds the contiguous buffer in L2 flow cache
- **Processing drops** from 4.36 ms to 3.86 ms (11.47%) by reading from the L2 flow cache rather than DRAM
- **Total time decreases** by 61.96%



Evaluation

- Copy-based design performs 6✕ and 5✕ payload size transfers (read segments, write contiguous buffer, process buffer)
- Per-core throughput with flow-based addressing is 2.62✕ and 1.33✕ traditional page-based addressing
- Benefits come from eliminating the copy stage and reducing processing time by sourcing data from the L2 flow cache instead of DRAM





Discussion and Conclusion

- **Packets are not pages**: SmartNICs inherit page-based memory management from CPUs, but packet workloads differ fundamentally
- Rather than accelerate an operation through a new offload, remove an operation entirely
- Flow-based addressing improves SmartNIC performance by removing memory inefficiencies rather than increasing compute throughput
- Need to re-examine memory addressing in high performance network devices

Questions?