Smart (Programmable) NICs

ECE/CS598HPN

Radhika Mittal

Offloading Distributed Applications onto SmartNICs using iPipe

Ming Liu, Tianyi Cui, Henry Schuh,
Arvind Krishnamurthy, Simon Peter, Karan Gupta

University of Washington, UT Austin, Nutanix

SIGCOMM'19

Slides from Arvind's SIGCOMM talk.

Programmable NICs

- Renewed interest in NICs that allow for customized per-packet processing
- Many NICs equipped with multicores & accelerators
  - E.g., Cavium LiquidIO, Broadcom Stingray, Mellanox BlueField
- Primarily used to accelerate networking & storage
  - Supports offloading of fixed functions used in protocols

Can we use programmable NICs to accelerate general distributed applications?

Talk Outline

- Characterization of multicore SmartNICs
- *iPipe* framework for offloading
- Application development and evaluation

SmartNICs Studied

	Vendor	BW	Processor	Deployed SW
LiquidIOII CN2350	Marvell	2X 10GbE	12 cnMIPS core, 1.2GHz	Firmware
LiquidIOII CN2360	Marvell	2X 25GbE	16 cnMIPS core, 1.5GHz	Firmware
BlueField 1M332A	Mellanox	2X 25GbE	8 ARM A72 core, 0.8GHz	Full OS
Stingray PS225	Broadcom	2X 25GbE	8 ARM A72 core, 3.0GHz	Full OS

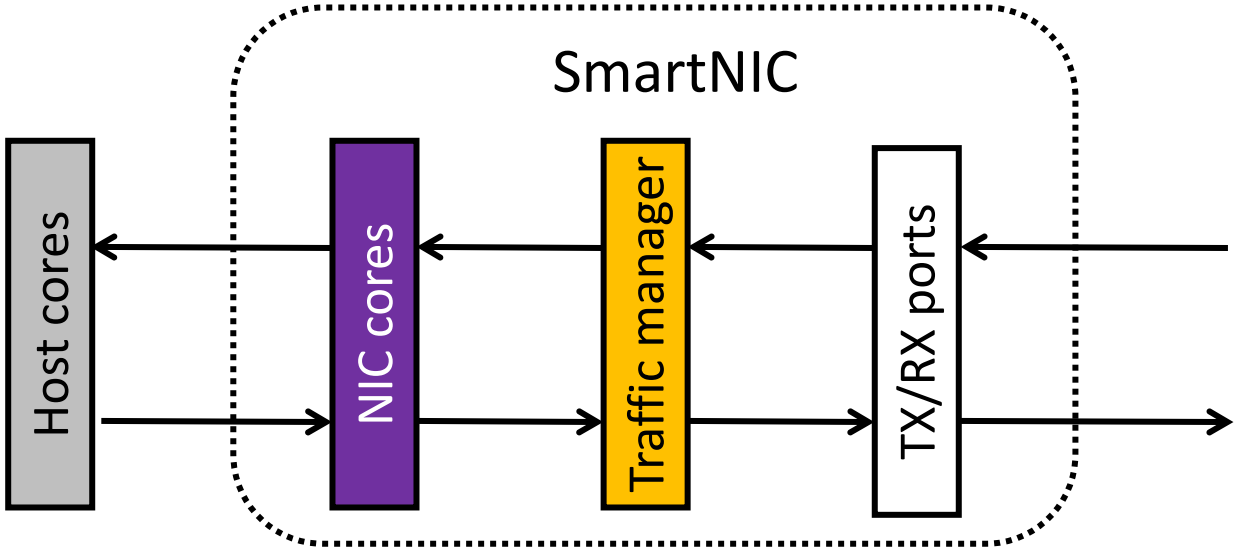
- Low power processors with simple micro-architectures
- Varying level of systems support (firmware to Linux)
- Some support RDMA & DPDK interfaces

Structural Differences

- Classified into two types based on packet flow
  - On-path SmartNICs
  - Off-path SmartNICs

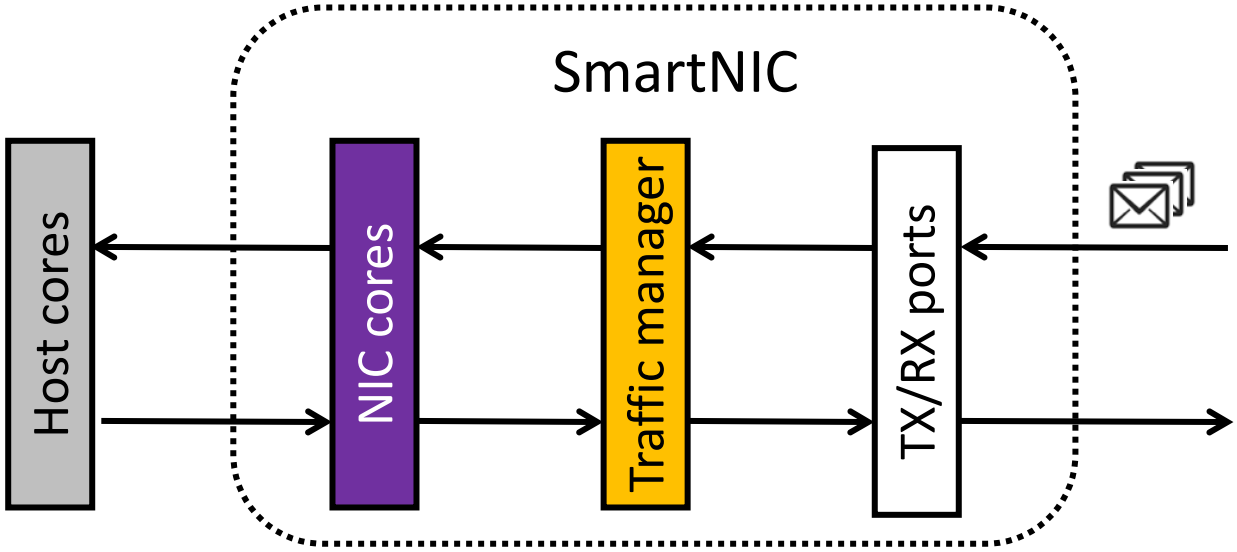
On-path SmartNICs

- NIC cores handle all traffic on both the send & receive paths



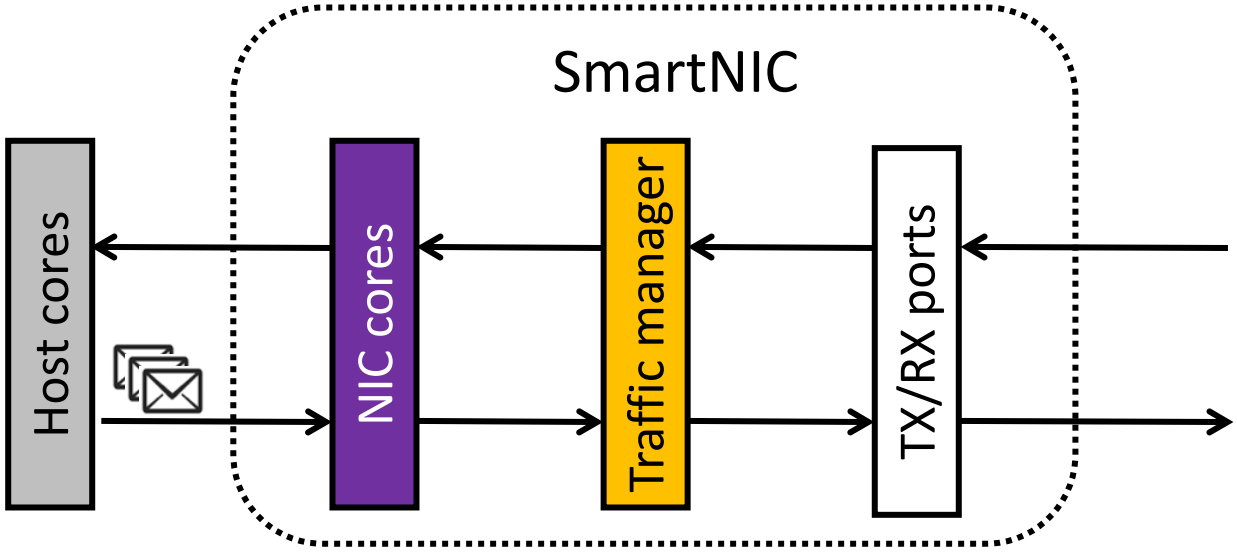
On-path SmartNICs: Receive path

- NIC cores handle all traffic on both the send & receive paths



On-path SmartNICs: Send path

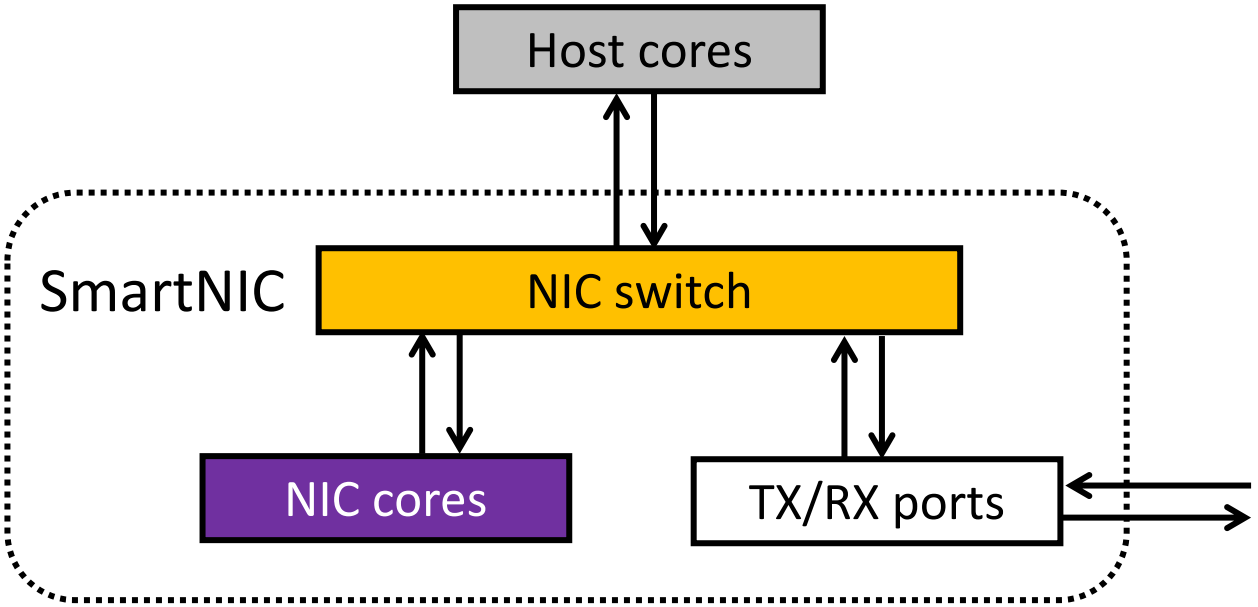
- NIC cores handle all traffic on both the send & receive paths



- Tight integration of computing and communication

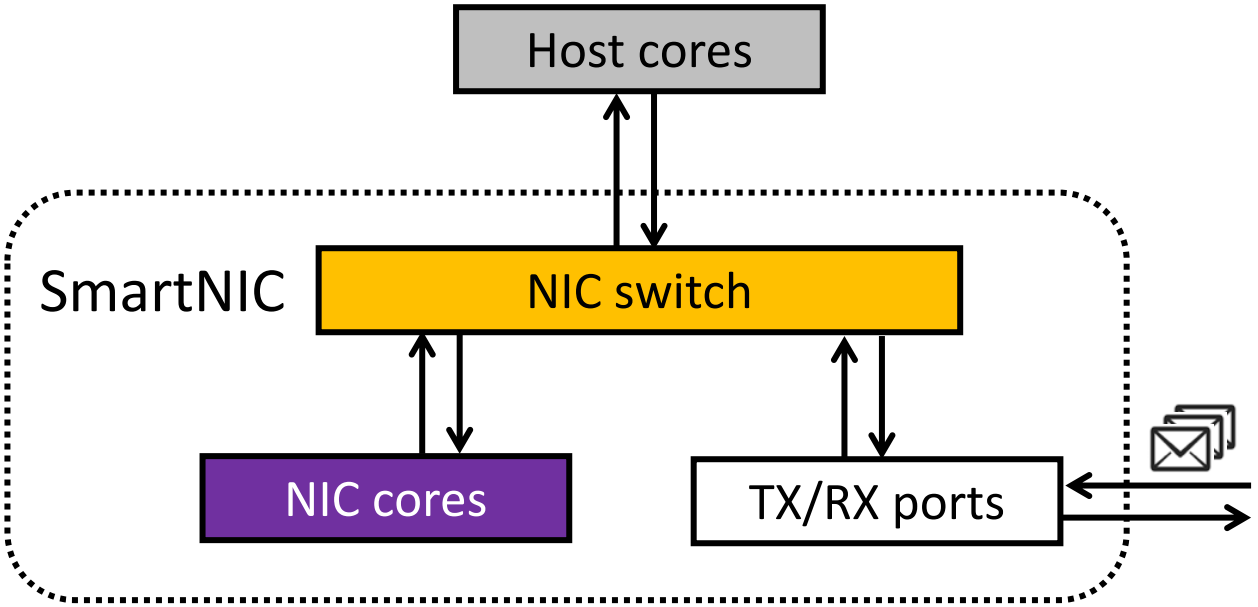
Off-path SmartNICs

- Programmable NIC switch enables targeted delivery



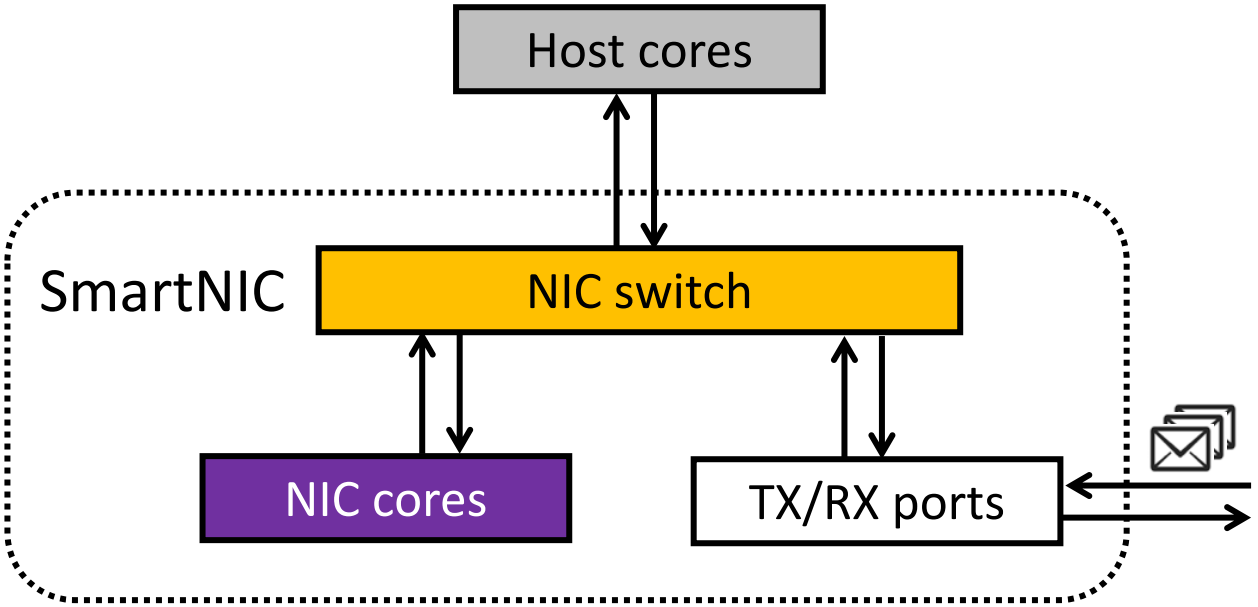
Off-path SmartNICs: Receive path

- Programmable NIC switch enables targeted delivery



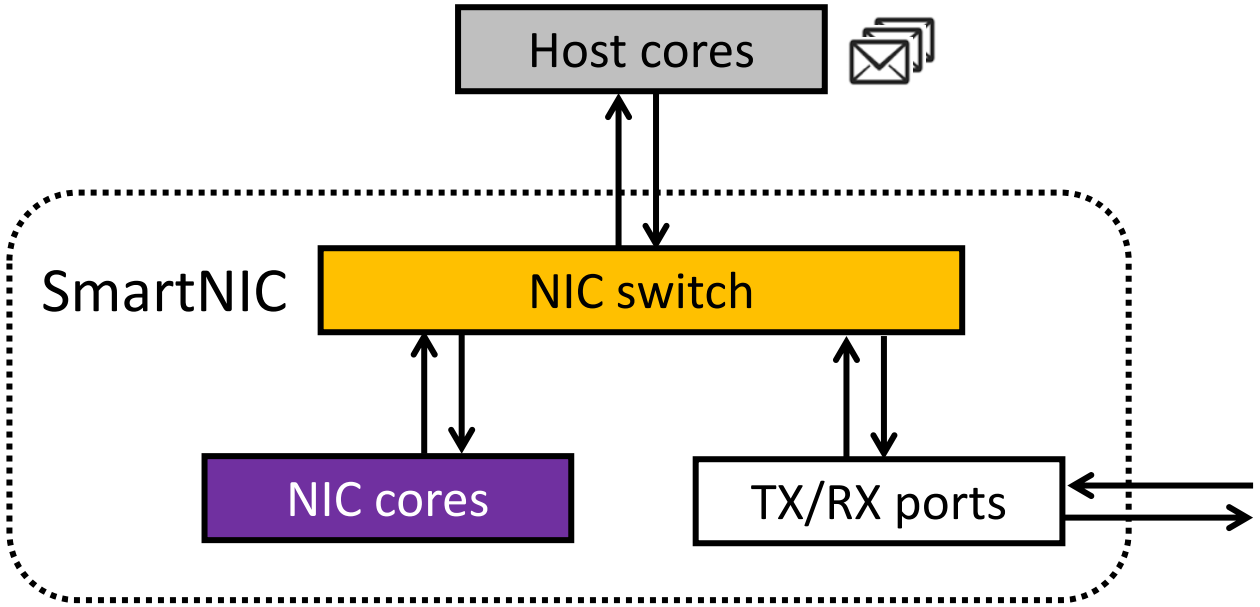
Off-path SmartNICs: Receive path

- Programmable NIC switch enables targeted delivery



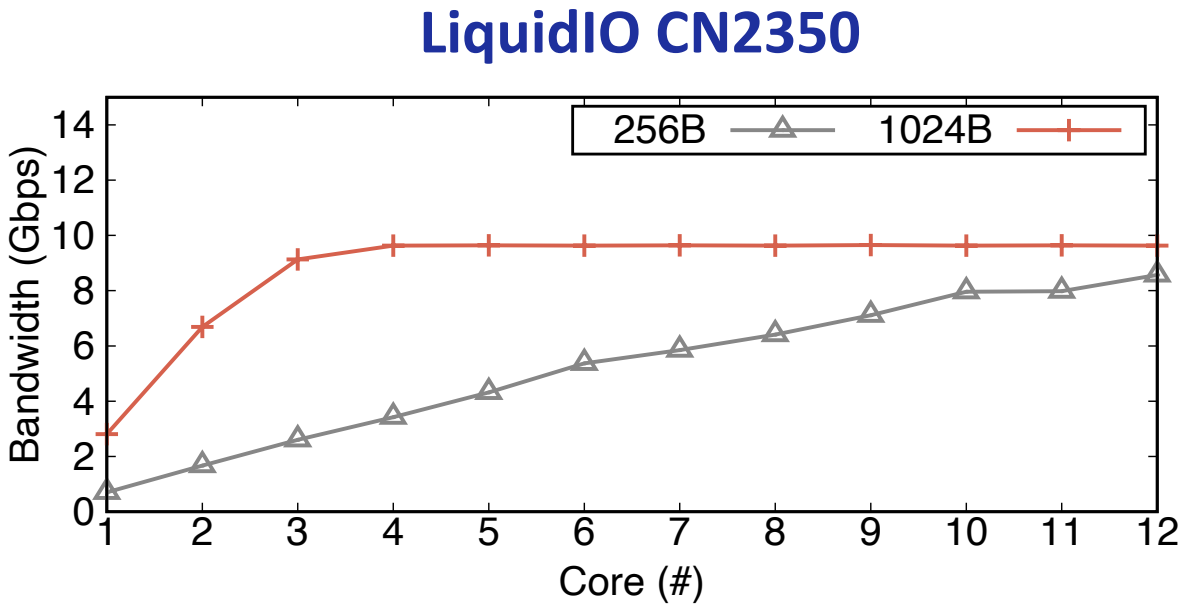
Off-path SmartNICs: Send path

- Programmable NIC switch enables targeted delivery
- Host traffic does not consume NIC cores
- Communication support is less integrated



Packet Processing Performance

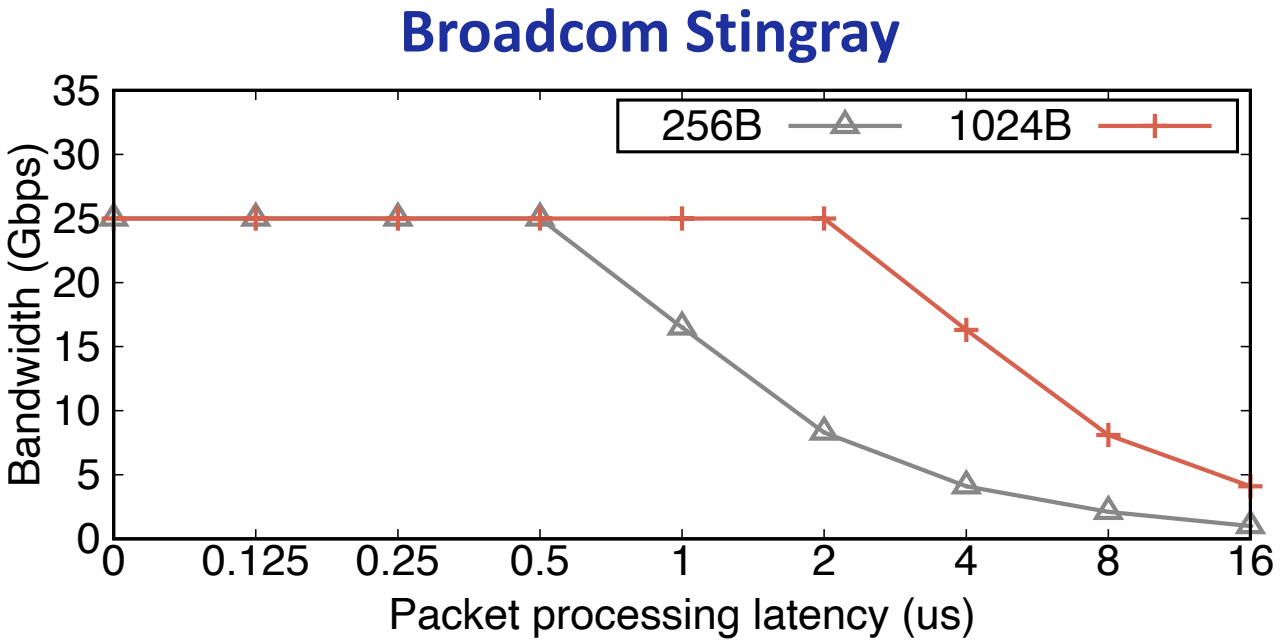
- Forwarding without any additional processing



- Quantifies the default forwarding tax of SmartNICs
- Dependent on packet size workload

Processing Headroom

- Forwarding throughput as we introduce additional per-packet processing



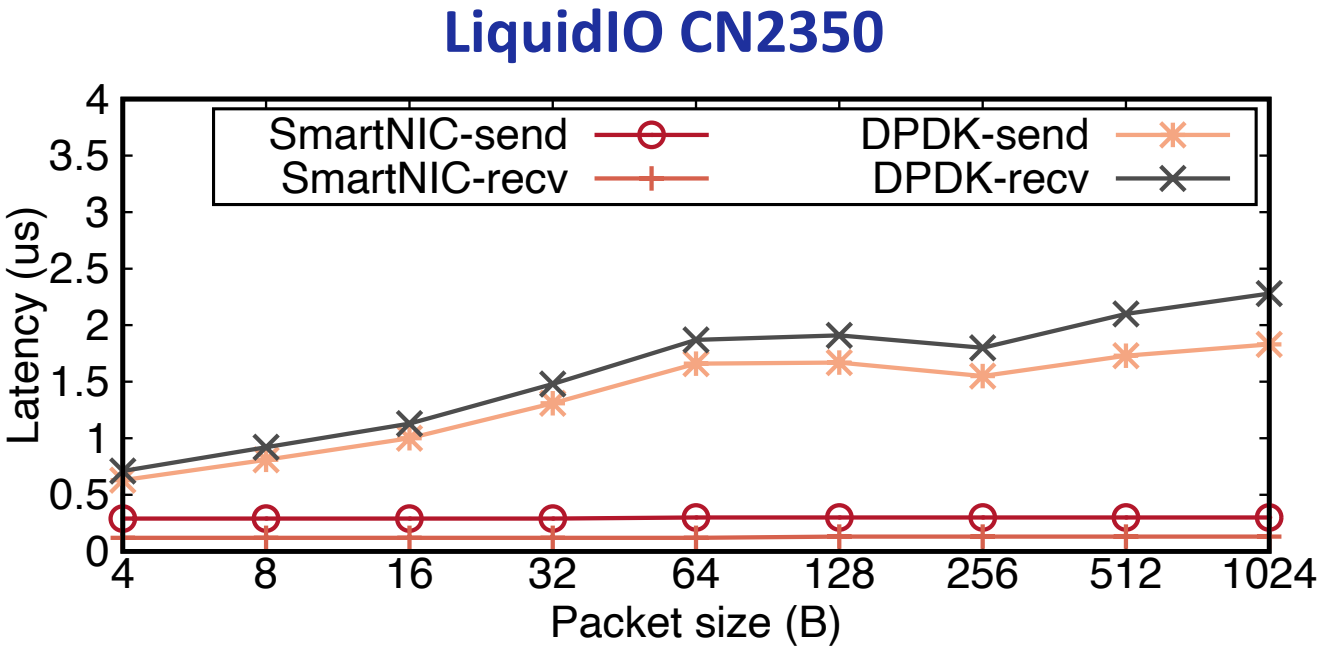
- Headroom is workload dependent and only allows for the execution of tiny tasks

Compute Performance

- Evaluated standard network functions on the SmartNIC cores
- Execution affected by cores' simpler micro-architecture and processing speeds
  - Suitable for running applications with low IPC
- Computations can leverage SmartNIC's accelerators but tie up NIC cores when batched
  - E.g., checksums, tunneling, crypto, etc.

Packet Processing Accelerators

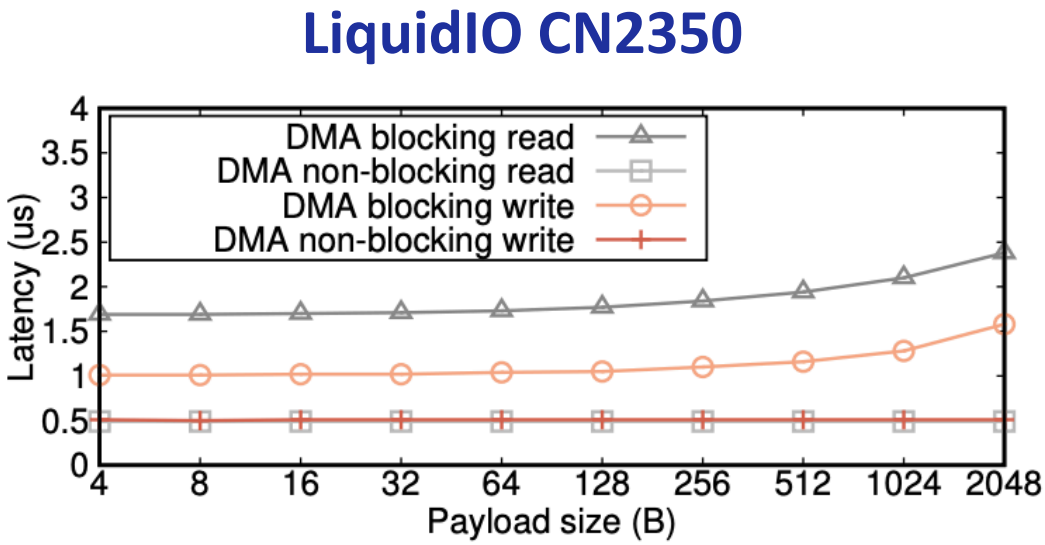
- On-path NICs provide packet processing accelerators
  - Moving packets between cores and RX/TX ports
  - Hardware-managed packet buffers with fast indexing



- Fast and packet-size independent messaging

Host Communication

- Traverse PCIe bus either through low-level DMA or higher-level RDMA/DPDK interfaces



- Non-trivial latency and overhead
  - Useful to aggregate and perform scatter/gather

iPipe Framework

- Programming framework for distributed applications desiring SmartNIC offload
- Addresses the challenges identified by our experiments

- Host communication overheads ⇒ distributed actors
- Variations in traffic workloads ⇒ dynamic migration
- Variations in execution costs ⇒ request scheduler

Actor Programming Model

- Application logic expressed using a set of actors
- Each actor has well-defined local object state and communicates with explicit messages
- Migratable actors; supports dynamic communication patterns

Actor Scheduler

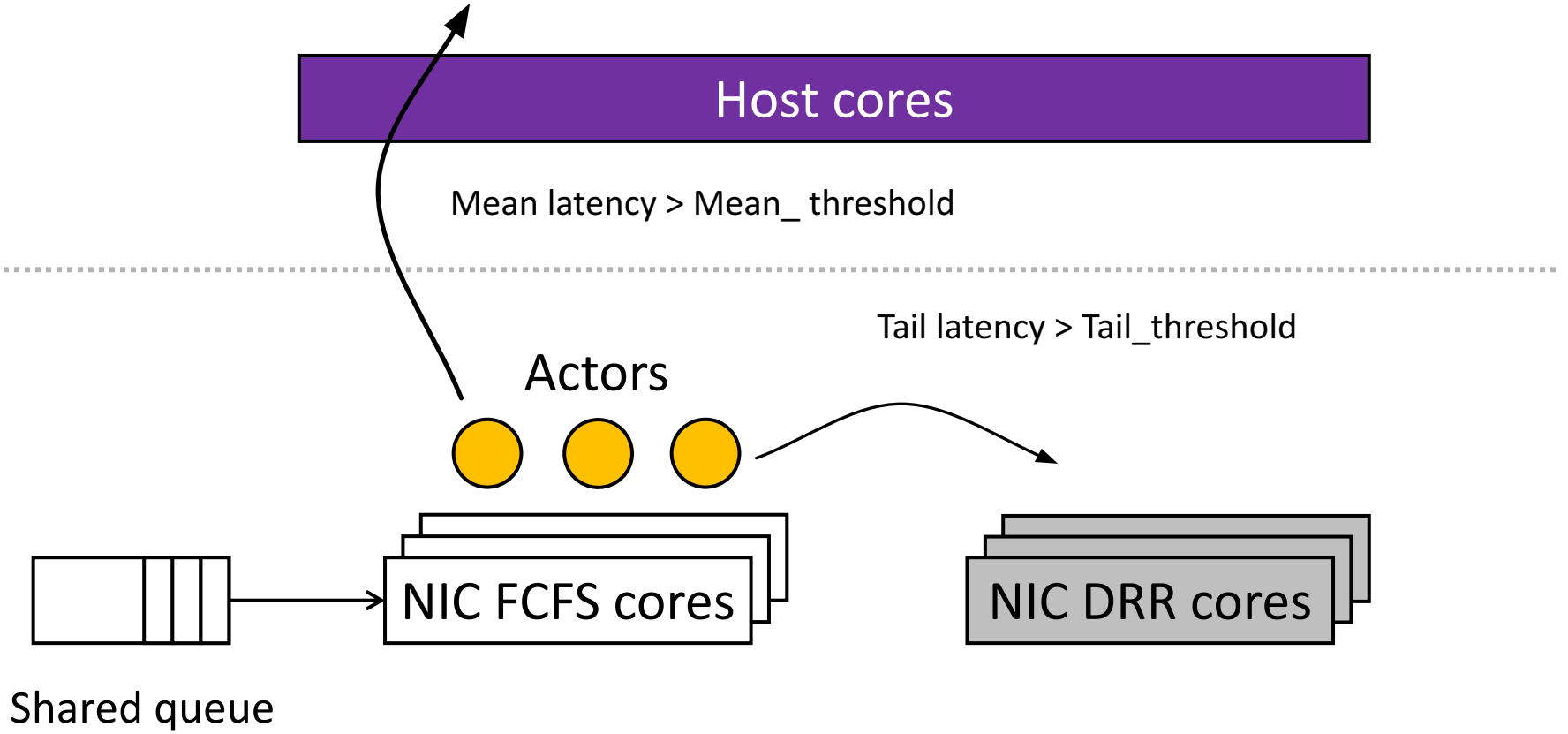
- Goal is to maximize SmartNIC usage, and
  - Prevent overloading and ensure line-rate communications
  - Provide isolation and bound tail latency for actor tasks
- Theoretical basis:
  - Shortest Job First (SJF) optimizes mean response time for arbitrary task distributions
  - If the tail response time is to be optimized:
    - First come first served (FCFS) is optimal for low variance tasks
    - Processor sharing is optimal for high variance tasks

iPipe's Hybrid Scheduler

- Design overview:
  - Combine FCFS and deficit round robin (DRR)
    - Use FCFS to serve tasks with low variance in service times
    - DRR approximates PS in a non-preemptible setting
  - Dynamically change actor location & service discipline
    - Monitor bounds on aggregate mean and tail latencies
    - Profile the mean and tail latency of actor invocations

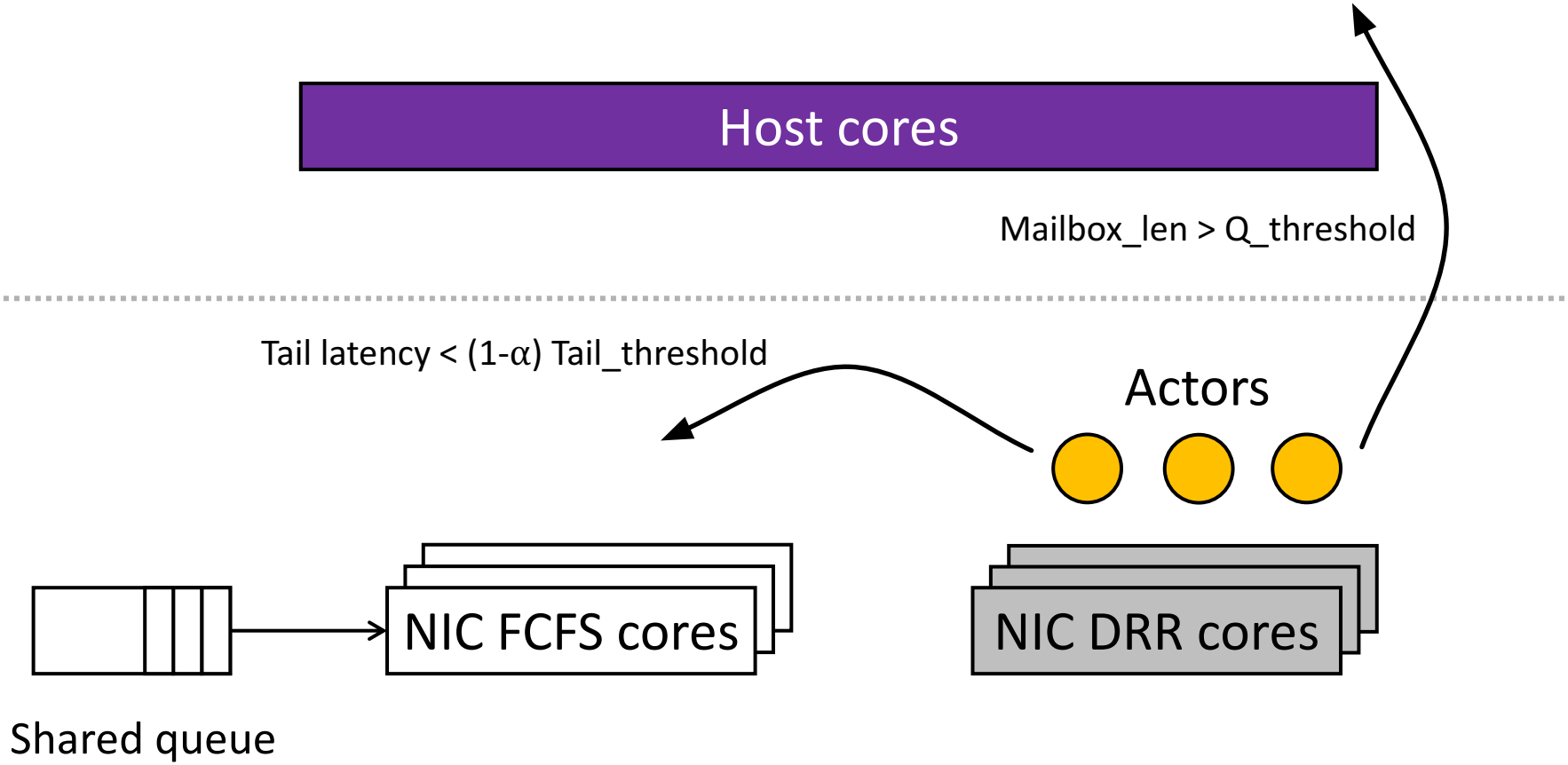
FCFS Scheduling

- FCFS cores fetch incoming requests from a shared queue and perform run-to-completion execution



DRR Scheduling

- DRR cores traverse the runnable queue and execute actor when its deficit counter is sufficiently high



Applications Built Using iPipe

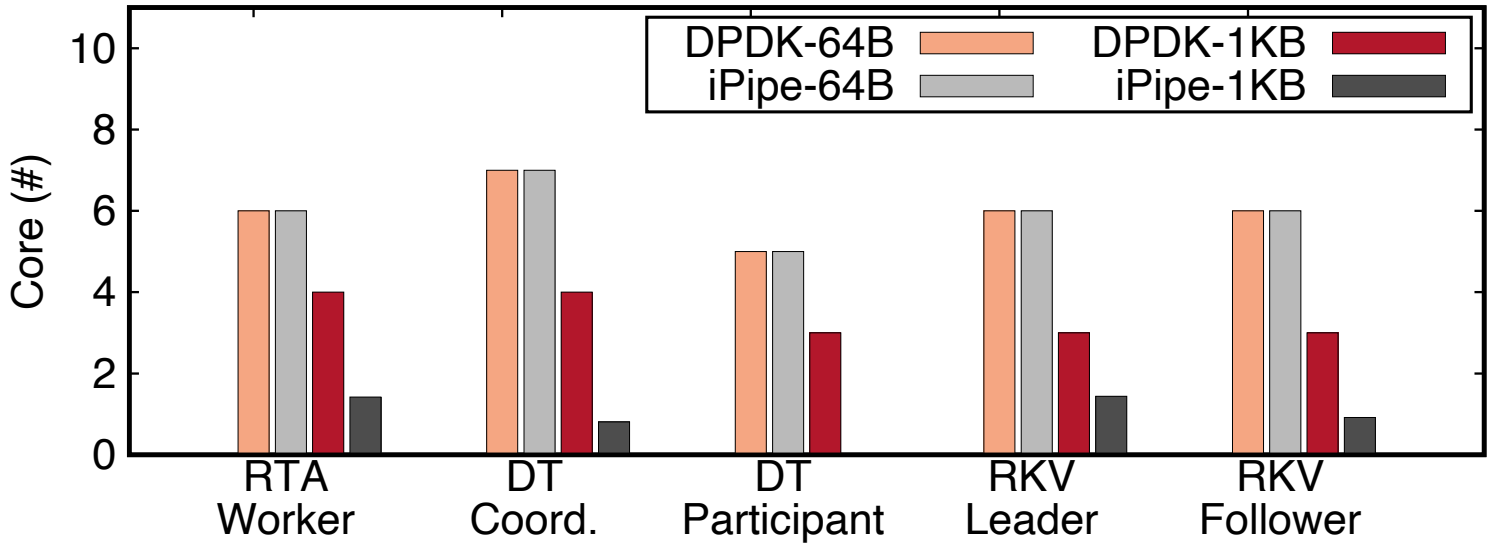
- Replicated and consistent key-value store
- Real time analytics
- Transaction processing system

Evaluation

- Application benefits:
  - Core savings for a given throughput
  - Or higher throughput for a given number of cores
  - Latency & tail latency gains

Host Core Savings for LiquidIO CN2360

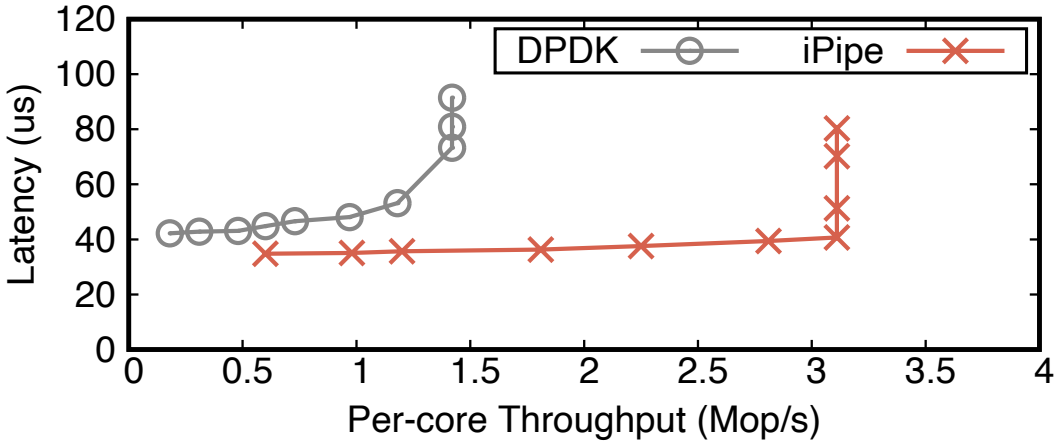
- Testbed:
  - Supermicro servers, 12-core E5-2680 v3 Xeon CPUs



- Offloading adapts to traffic workload
- Average reduction in host core count is 73% for 1KB packets

RKV Store Latency/Throughput (LiquidIO CN2360)

- Fixed the host core count and evaluated the improvement in application throughput



- 2.2x higher throughput and 12.5us lower latency

Summary

- Performed an empirical characterization of SmartNICs
  - Significant innovation in terms of hardware acceleration
  - Off-path and on-path designs embody structural differences
  - SmartNICs can be effective but require careful offloads
- iPipe framework enables offloads for distributed applications
  - Actor-based model for explicit communication & migration
  - Hybrid scheduler for maximizing SmartNIC utilization while bounding mean/tail actor execution costs
- Demonstrated offloading benefits for distributed applications

Your Opinions

Pros

- Good understanding of SmartNICs
- Adapts to traffic workload
- Tackles not just performance but also security concerns.
- Tested on three different applications.
- Extensive evaluation

Your Opinions

Cons

Your Opinions

Cons

- Can it scale to 40Gbps or 100Gbps bandwidth?
- How well does it scale with number of applications?
- Can we run out of NIC memory for memory-bound tasks?
- Is offload really useful for such apps?
- Coexistence of iPipe with other offloads.
- 10% overhead of the iPipe framework.
- Needs redesigning of applications.
- Comparison with other approaches?

Your Opinions

Ideas

- Compare with FPGA-based NICs
- Enable such a framework on FPGA-based NICs
- Coexistence with network functions in SmartNICs
- Fair-sharing of NIC resources across multiple tenants, handling tasks with different prioirities
- More evaluation on on-path and off-path smartNICs
- Attacks that by-pass their security provisions.

Other applications

- Load balancing / request steering
  - RPCValet, ASPLOS'19
  - A Case for Informed Request Scheduling at the NIC, HotNets'19
- Remote memory calls, HotNets'20
- Network functions
  - ClickNP, SIGCOMM'16
  - FlowBlaze, NSDI'19
- Caching for key-value stores
  - IncBricks, ASPLOS'17
- ……

Class on Dec 1st

- Pick a paper of your choice on a related topic.
  - The paper should not have already been discussed in class.
- No need to submit reviews.
- Instead prepare a 4mins presentation on the paper
  - What problem is it trying to solve and why?
  - How?
  - Result.