HiPINEB 2017

Knapp: A Packet Processing Framework for Manycore Accelerators

Junhyun Shim, Joongi Kim, Keunhong Lee

Sue Moon

School of Computing, KAIST

2017. 2. 6.

KAIST

Advanced Networking Lab

Motivation

- Network in the era of Exascale computing and MSN vs ICN
  - Remains as a bottleneck in high performance computing
  - Is transitioning from custom HW to commodity HW
- Pulling high-performance out of commodity HW
  - Imperative in the time of NFV and SDN
  - NetMap, PacketShader, CuckwooSwitch, …

Key Software Enablers

- For custom HW to commodity HW transition
  - Userspace I/O (No kernel-user level copies)
  - Resource pooling (batching at all stages)
  - Compartmentalized allocation
    (NUMA local threads and memory use)
- Remaining challenges
  - Network packet processing is not uniform
  - Workload varies depending on Network Functions (NFs)
  - Accelerators add extra cycles needed: GPUs, Intel Xeon Phis

Intel Xeon Phi

- Knights Ferry – **Knights Corner** – Knights Landing – Knights Hill
- Many Integrated Core Architecture – 60 cores, 1.053 GHz, 4 HTs
- Instruction level vectorization – 16 INTs in single cycle throughput
- On-board 8GB GDDR5 RAM (doubles as disk)
- 2 simultaneous instructions per cycle (1 vector, 1 scalar)
  Loss if not running at least 2 concurrent threads per core
- **Runs its own Linux µOS.** Can run its own code.
  Unprecedented level of **transparency** in accelerator community
- Currently positioned as accelerator for scientific computing

Intel Xeon Phi
for
High-Speed Packet Processing?

KAIST

Advanced Networking Lab

Performance Metrics of Interest

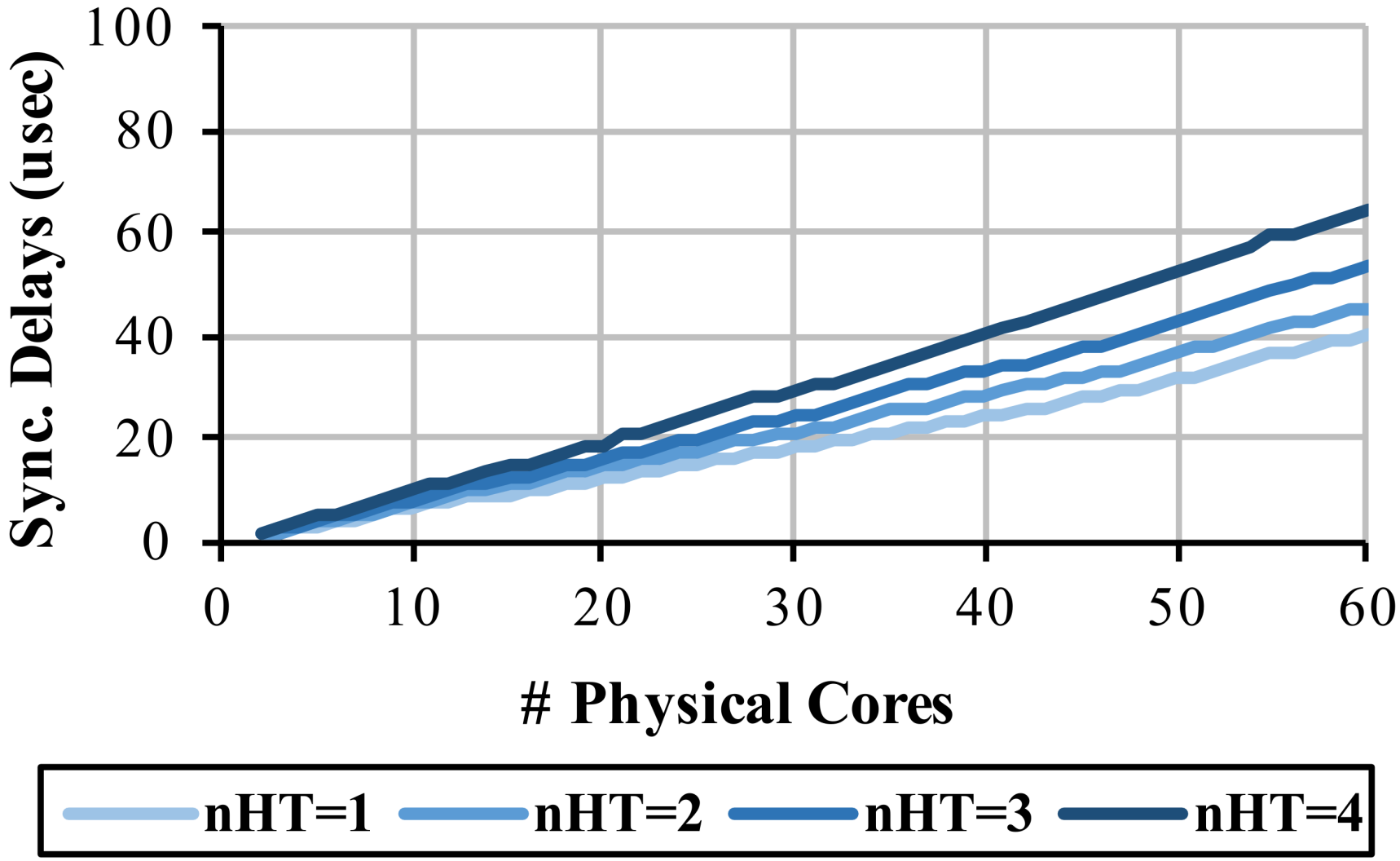
- Latency and throughput
- Following measures are critical
  - Integer operation throughput

  => Back-of-envelope calculation concludes viable
  - Random memory access

  ⇒For address loookups

  ⇒Comparable to CPUs, but far worse than GPUs
  - Thread synchronization overhead

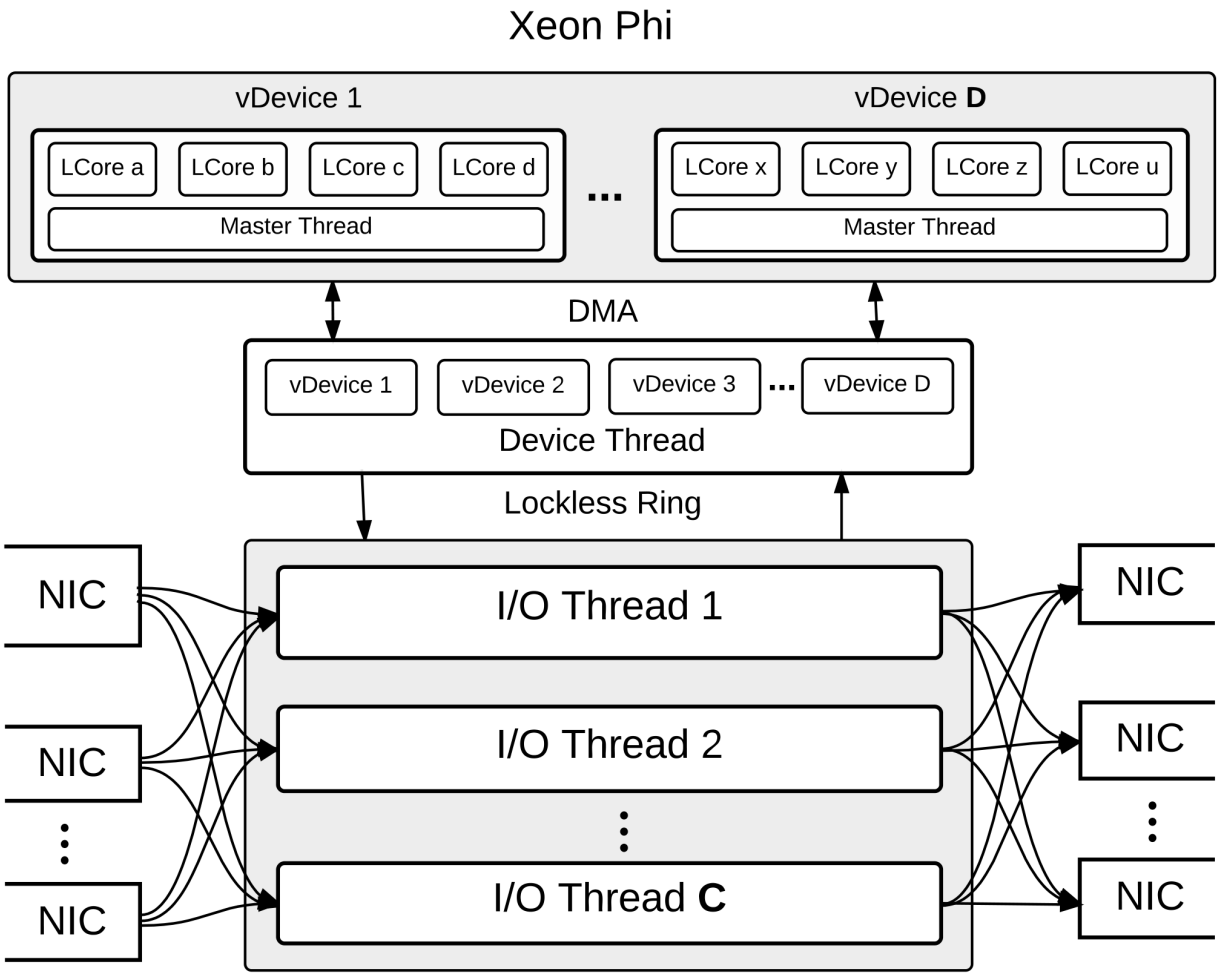
Synchronization Cost vs # Cores



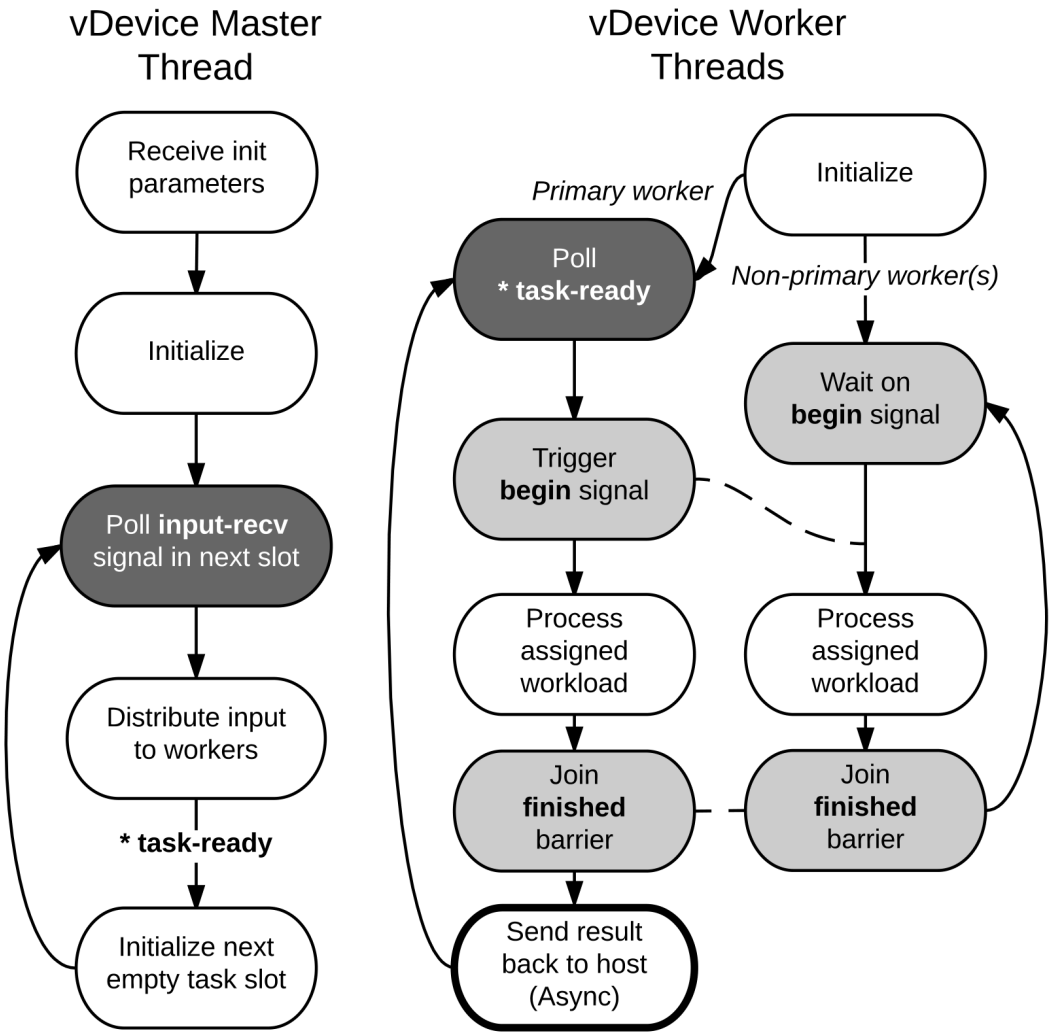
Knapp

- Knight's Corner as a Packet Processing framework
- Host-side of Knapp
  - Uses Intel DPDK for packet IO
  - One core per NUMA node for device comm
- Device-side of Knapp
  - vDevice partitions cores on the device
  - Each vDevice is associated with a packet processing application and two SCIF channels (control/data)

Knapp Architecture



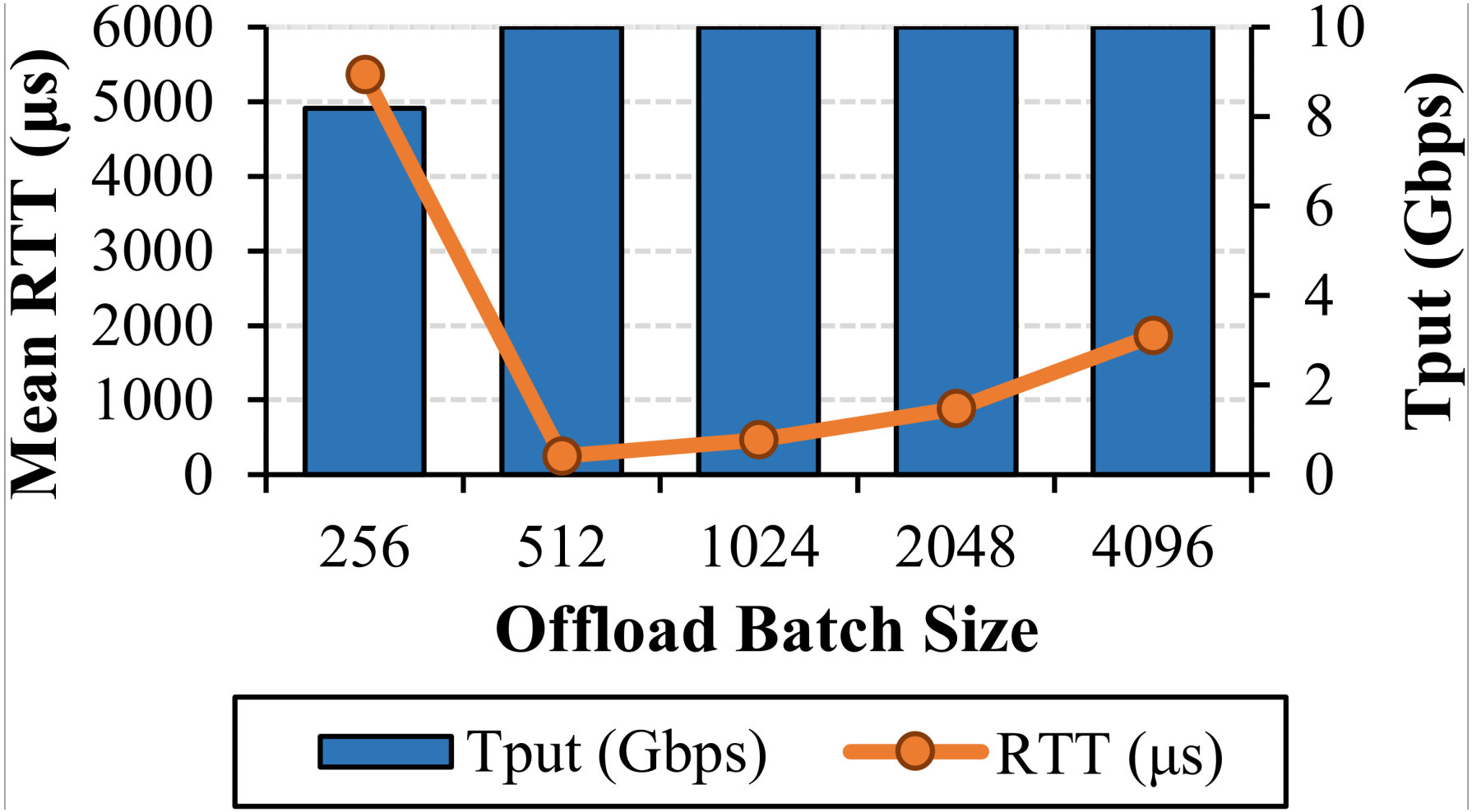
Control Flow of vDevice Pipeline



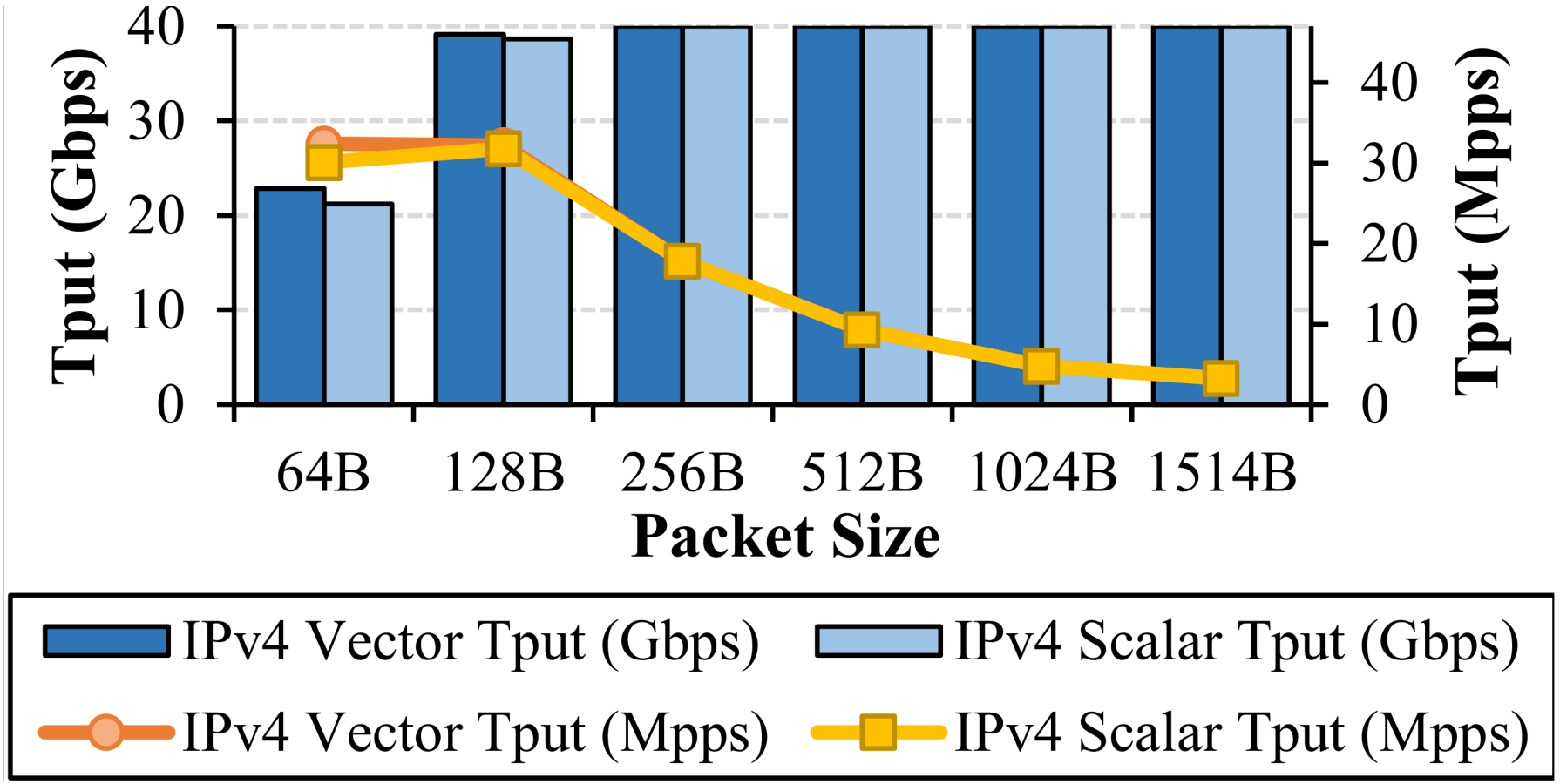
Evaluation Configuration

Category	Specification
CPU	1x Intel Xeon E5-2670 Sandy Bridge, octa-core 2.6 GHz
RAM	32 GB
NIC	2x Intel 82599ES dual-port 10GbE, total 40 Gbps
MIC	1x Intel Xeon Phi 5110P 60 1.053 GHz Atom cores, 8 GB RAM, 320 GB.s, PCIe 2.0

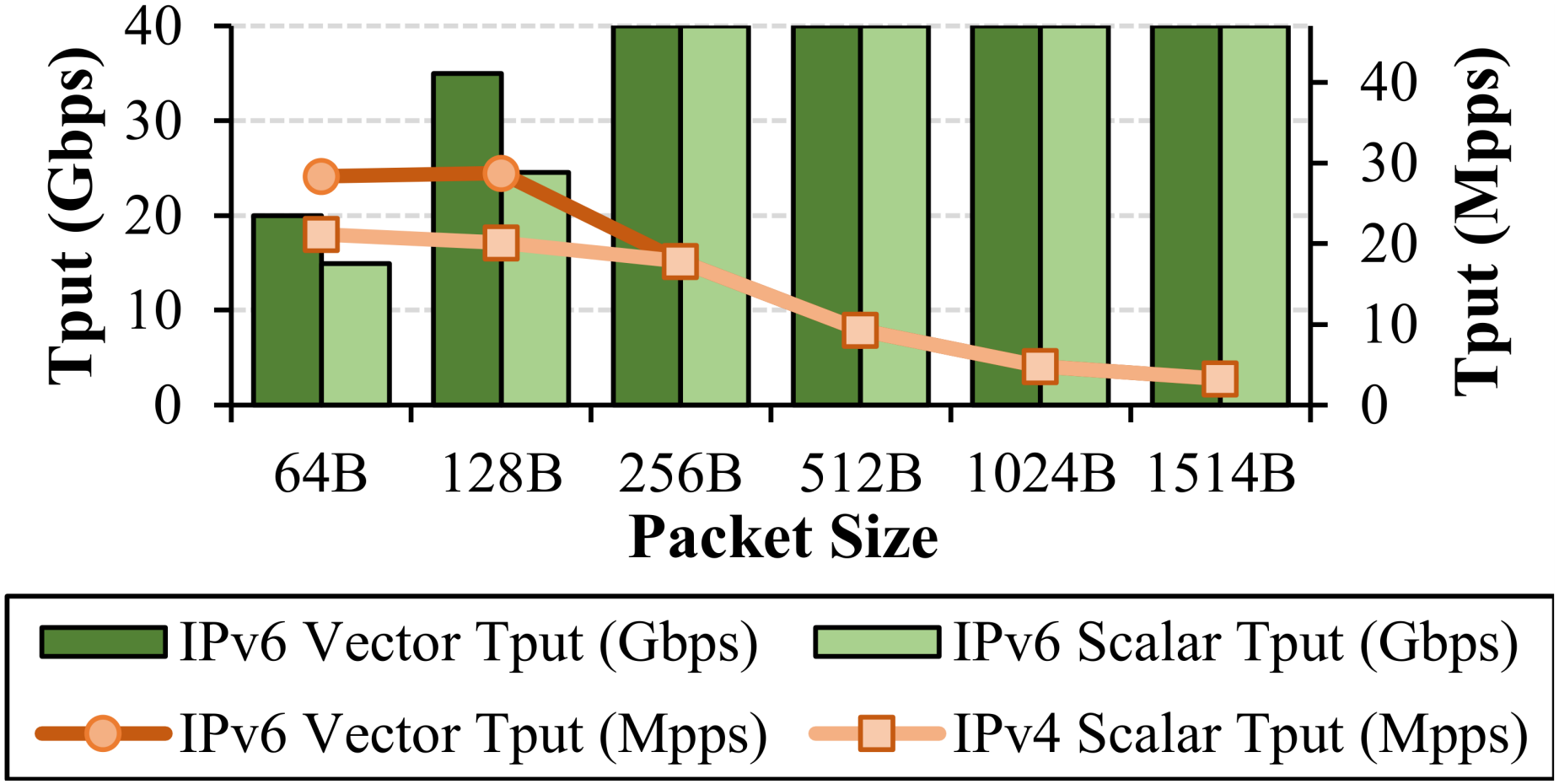
Packet Forwarding Latency



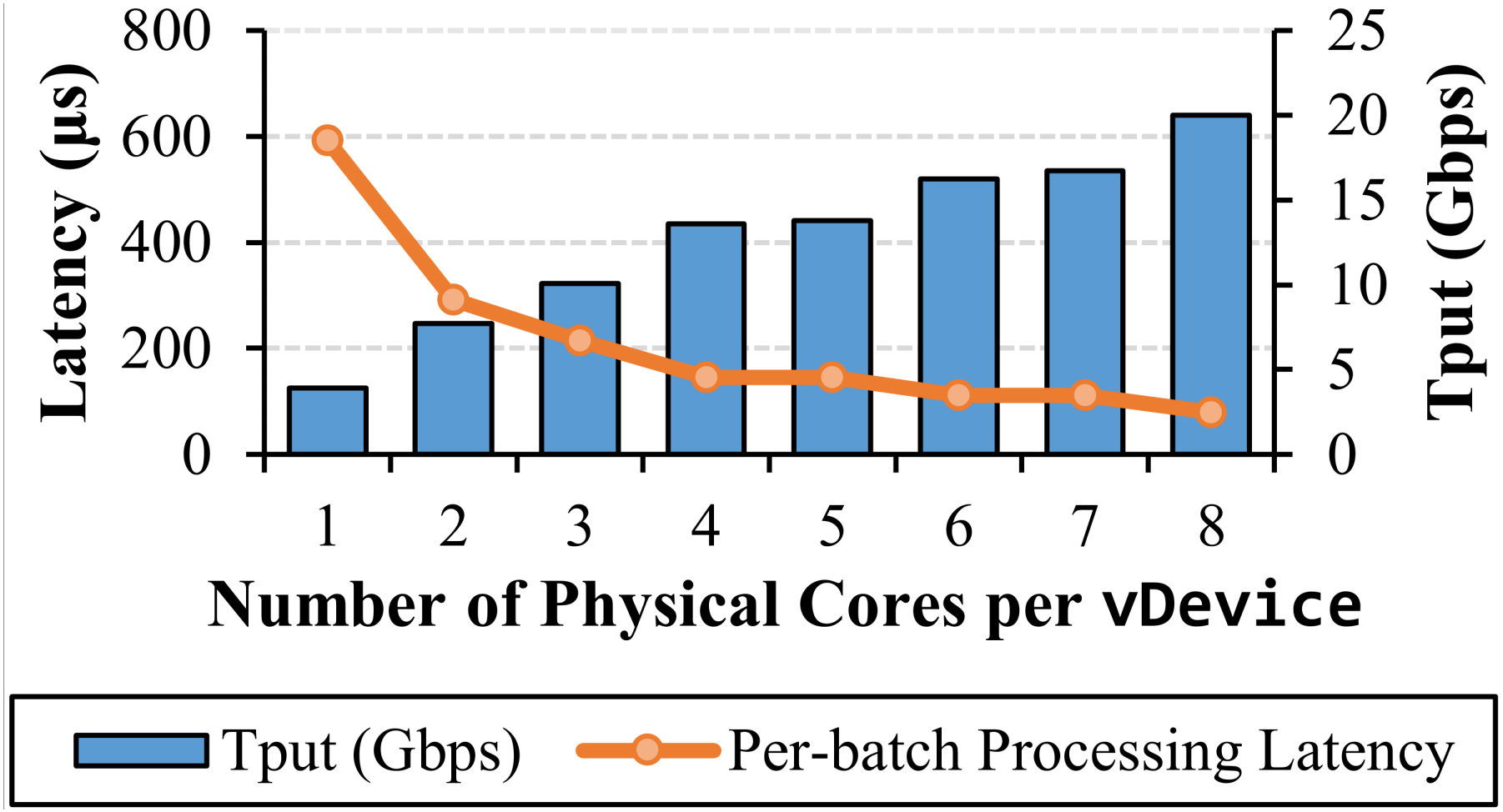
Packet Forwading Throughput: IPv4



Packet Forwading Throughput: IPv6



Core Scalability in vDevice: Case Study of Vectorized IPv6



Conclusions and Future Work

- Intel Xeon Phi feasible as an accelerator
  - Remaining challenges: manual vectorization to extract most out of Xeon Phi architecture
  - Knight's Landing as a stand-alone processor work well
- Future Work
  - Explore the implementation behind Peer Direct™/GPU Direct (P2P DMA technology among PCIe devices)
  - Add GPU-like Xeon Phi daemon interface to NBA
  - Extend to other common router apps (IPSec, NAT, IPv6)

Q&A