PacketShader 2.0: Design Considerations

FIF Testbed WG Meeting

2011.8.10. 3-6pm 충남대 공학2관 492호

Sue Moon

in collaboration with:

Joongi Kim, Seonggu Huh, Sangjin Han, Keon Jang, KyoungSoo Park [‡]

Advanced Networking Lab, CS, KAIST

[‡] Networked and Distributed Computing Systems Lab, EE, KAIST

KAIST

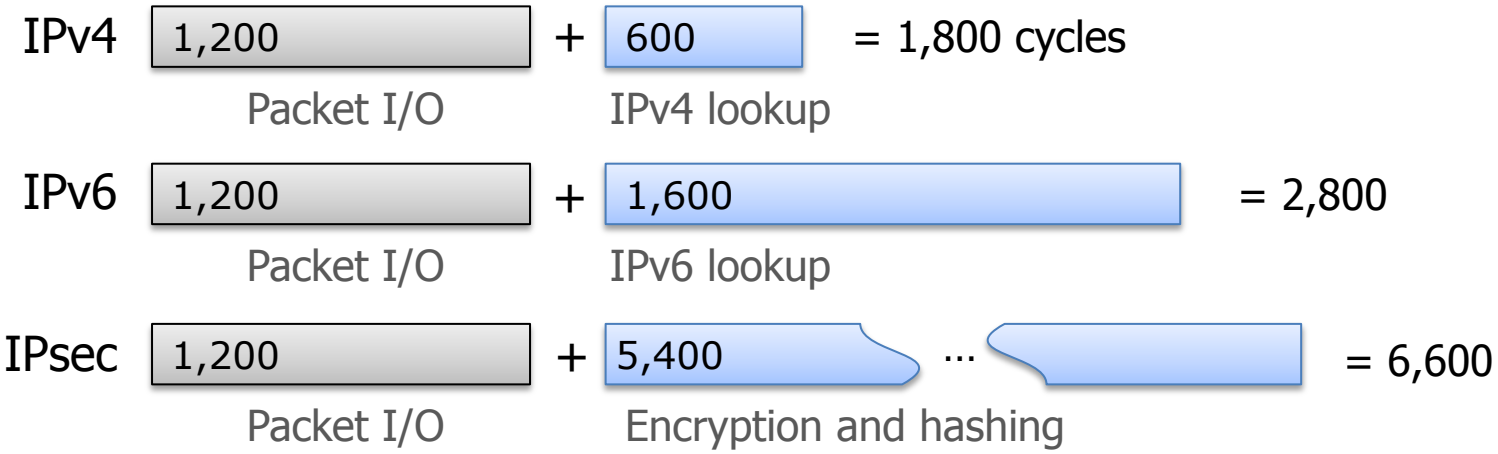
PacketShader 1.0

- GPUs
  - a great opportunity for fast packet processing
- v1.0: more of a forwarding engine
  - Optimized packet I/O + GPU acceleration
  - scalable with
    - # of multi-core CPUs, GPUs, and high-speed NICs
- Current Prototype
  - Supports IPv4, IPv6, OpenFlow, and IPsec
  - 40 Gbps performance on a single PC

CPU BOTTLENECK

Per-Packet CPU Cycles for 10G

Cycles needed



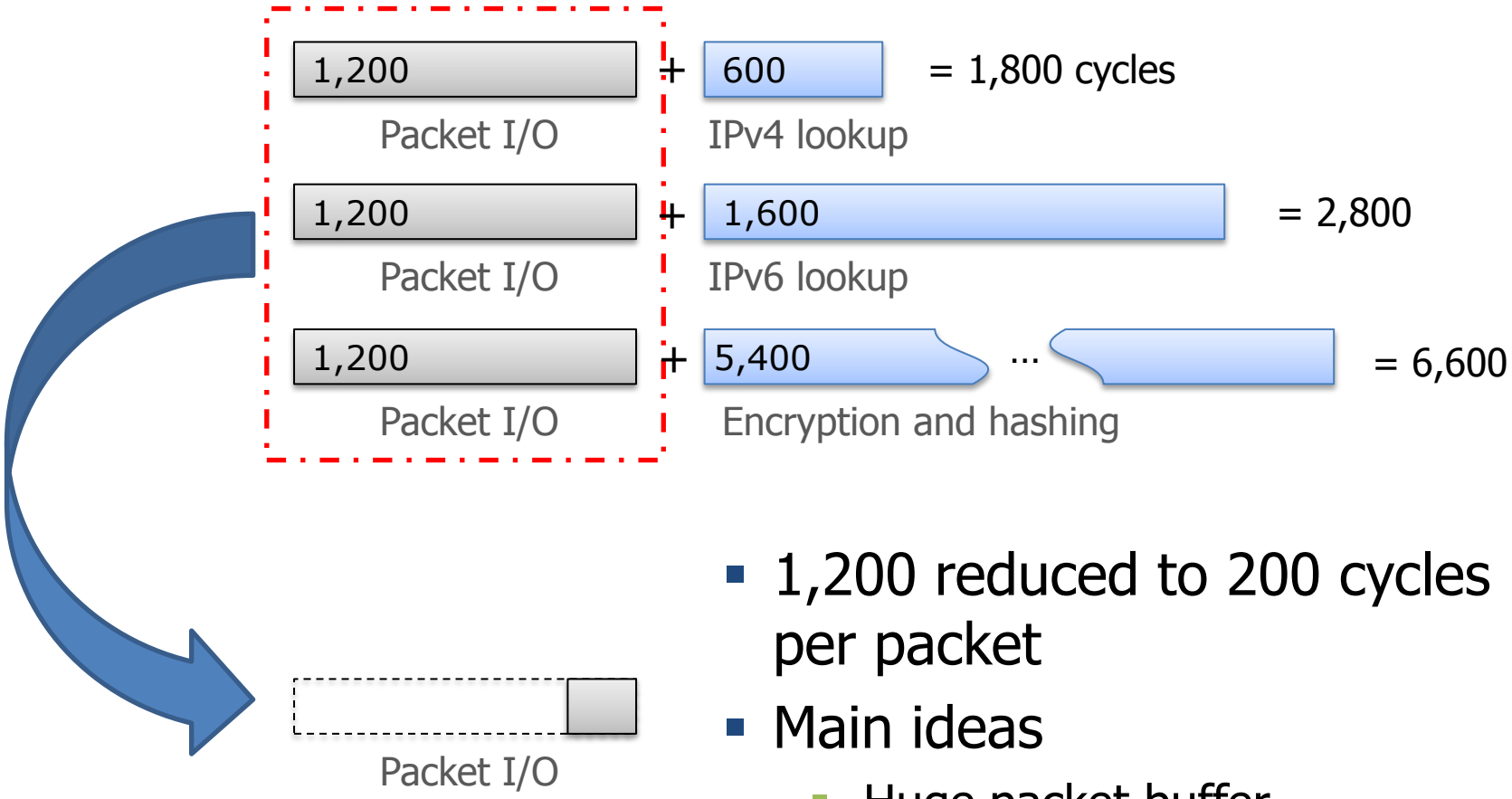
Your budget

10G, min-sized packets, dual quad-core 2.66GHz CPUs

(in x86, cycle numbers are from RouteBricks [Dobrescu09] and ours)

KAIST

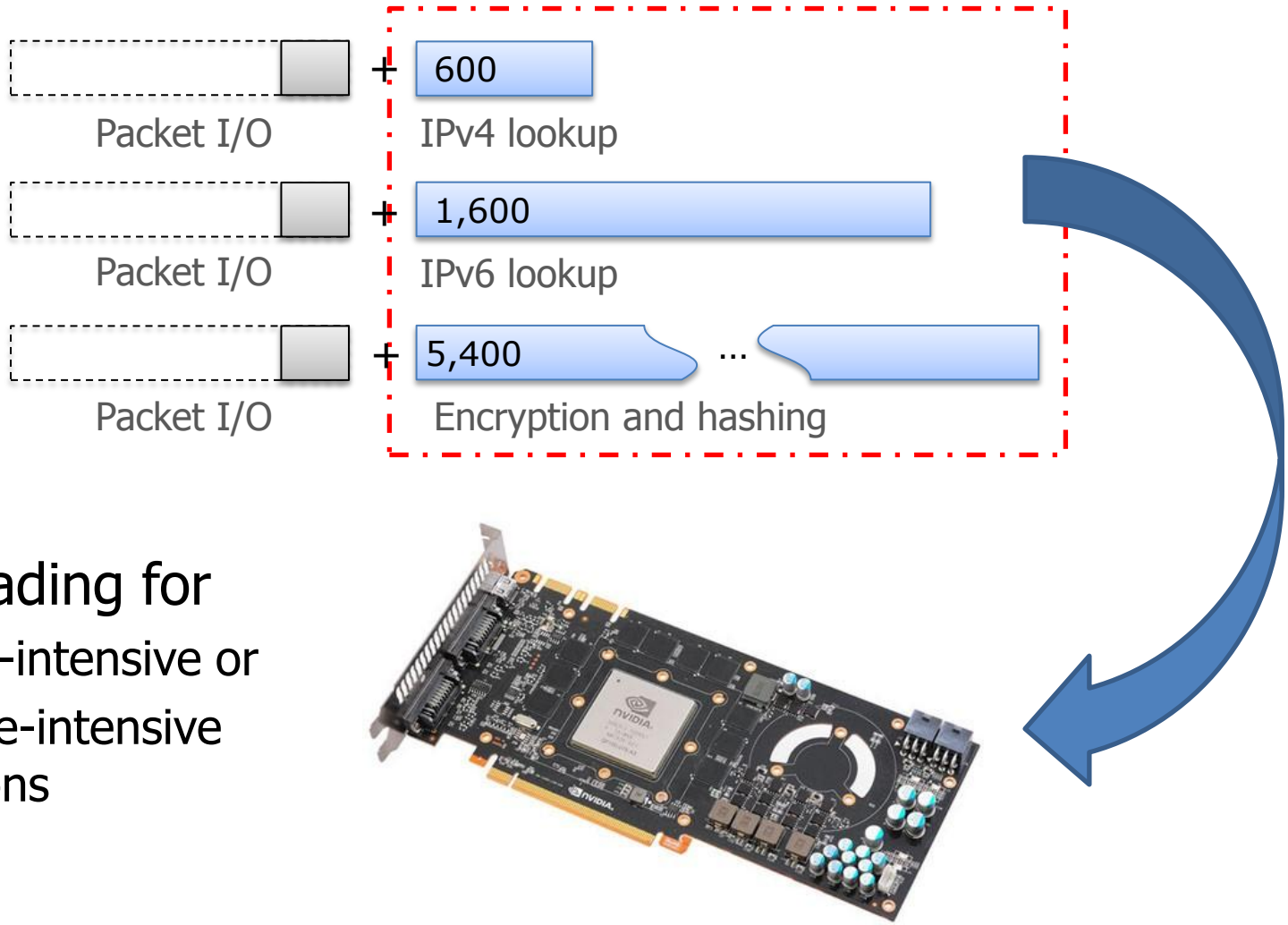
PacketShader Part 1: I/O Optimization



- 1,200 reduced to 200 cycles per packet
- Main ideas
  - Huge packet buffer
  - Batch processing

KAIST

PacketShader Part 2: GPU Offloading



- GPU Offloading for
  - Memory-intensive or
  - Compute-intensive operations

KAIST

OPTIMIZING PACKET I/O ENGINE

User-space Packet Processing

Packet processing in kernel is bad

- Kernel has higher scheduling priority; overloaded kernel may starve user-level processes.
- Some CPU extensions such as MMX and SSE are not available.
- Buggy kernel code causes irreversible damage to the system.

Processing in user-space is good

- Rich, friendly development and debugging environment
- Seamless integration with 3rd party libraries such as CUDA or OpenSSL
- Easy to develop virtualized data plane.

But packet processing in user-space is known to be 3x times slower!

➔ Our solution: (1) batching + (2) better core-queue mapping

Inefficiencies of Linux Network Stack

Compact metadata →
Batch processing →
Software prefetch →

Functional bins	% of cycles
skb (de)allocation	8.0%
skb initialization	4.9%
NIC device driver	13.3%
Compulsory cache misses	13.8%
Memory subsystem	50.2%
Others	9.8%
Total	100.0%

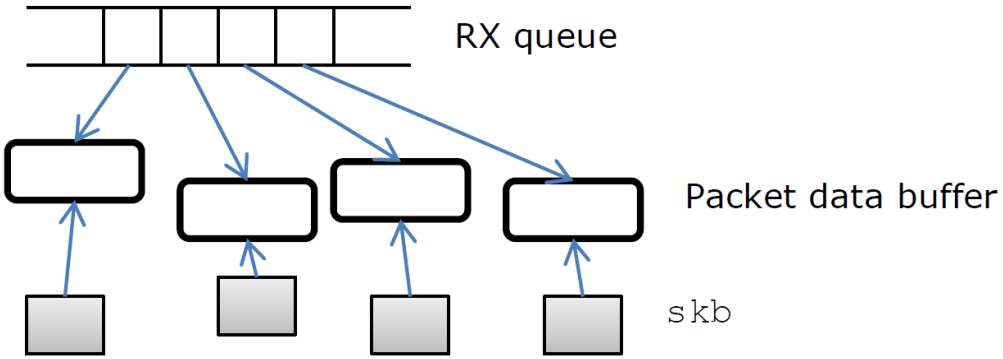
Huge packet buffer

CPU cycle breakdown in packet RX

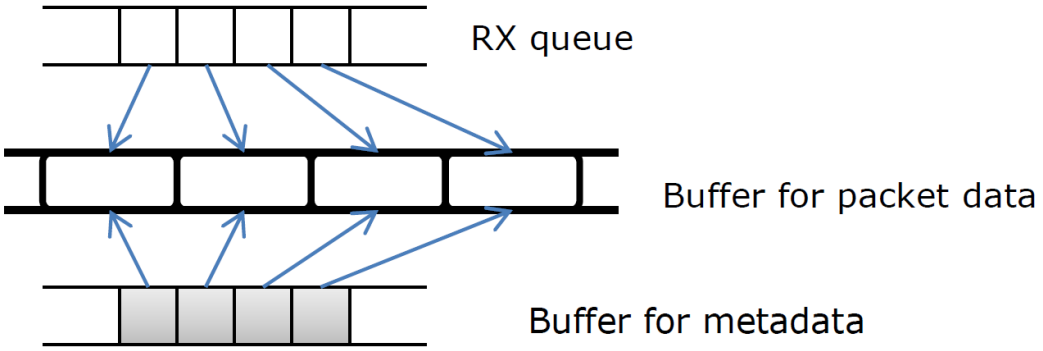
KAIST

Huge Packet Buffer

eliminates per-packet buffer allocation cost



Linux per-packet buffer allocation



Our huge packet buffer scheme

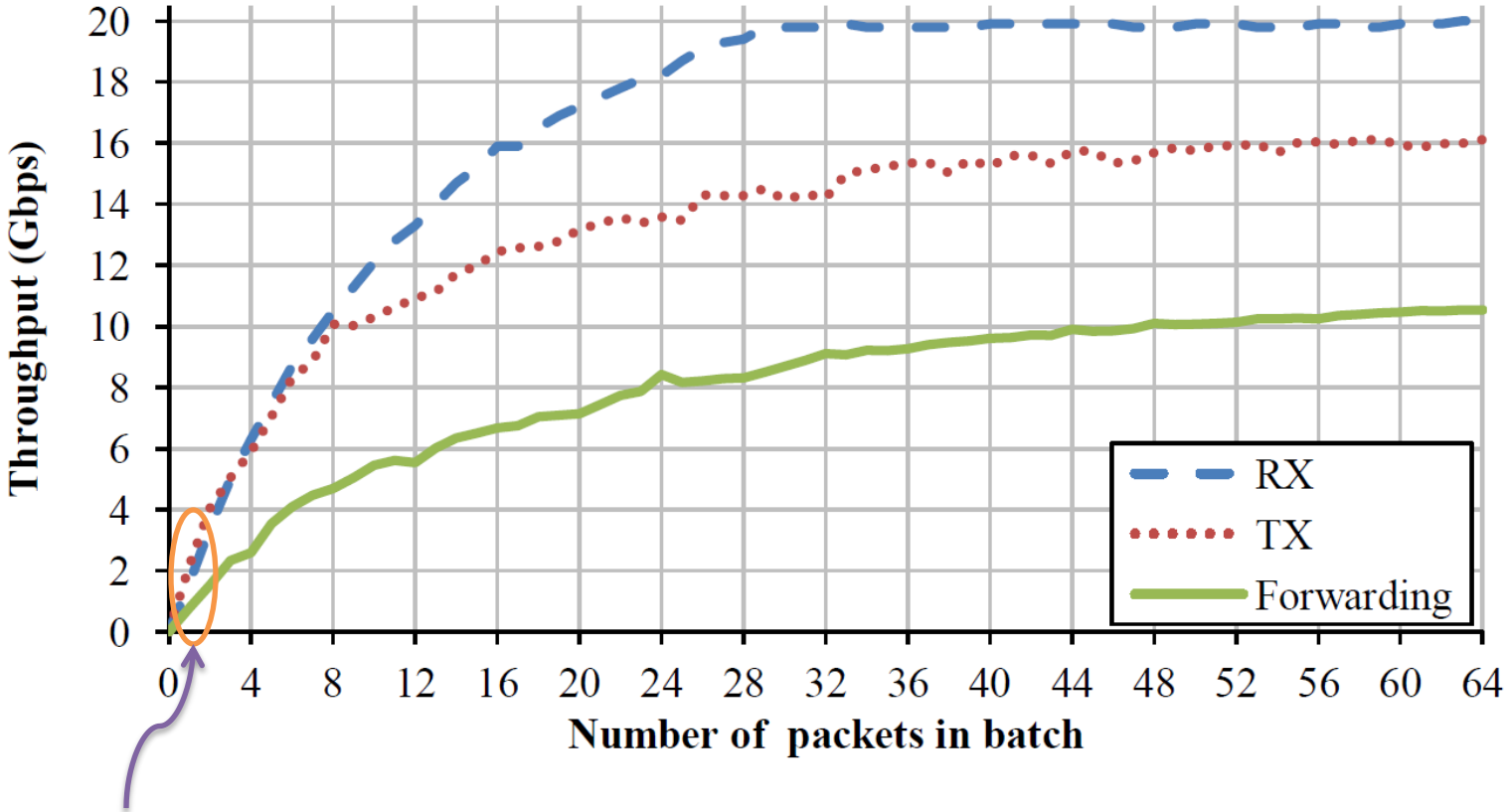
Batch Processing

amortizes per-packet bookkeeping costs.

- Simple queuing theory:
  - input traffic > capacity of the system
  - → RX queues fills up
- Dequeue and process multiple packets
  - It improves overall throughput

Effect of Batched Packet Processing

64-byte packets, two 10G ports, one CPU core



Without batching: 1.6 Gbps for RX, 2.1 Gbps for TX, 0.8 Gbps for forwarding

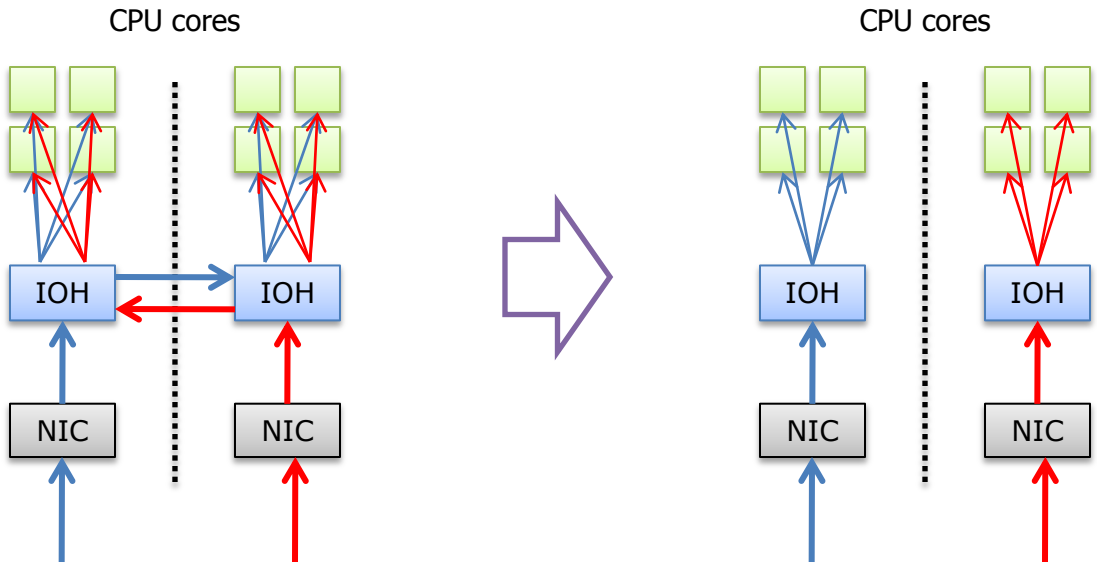
➔ batching is essential!

KAIST

NUMA –Aware RSS

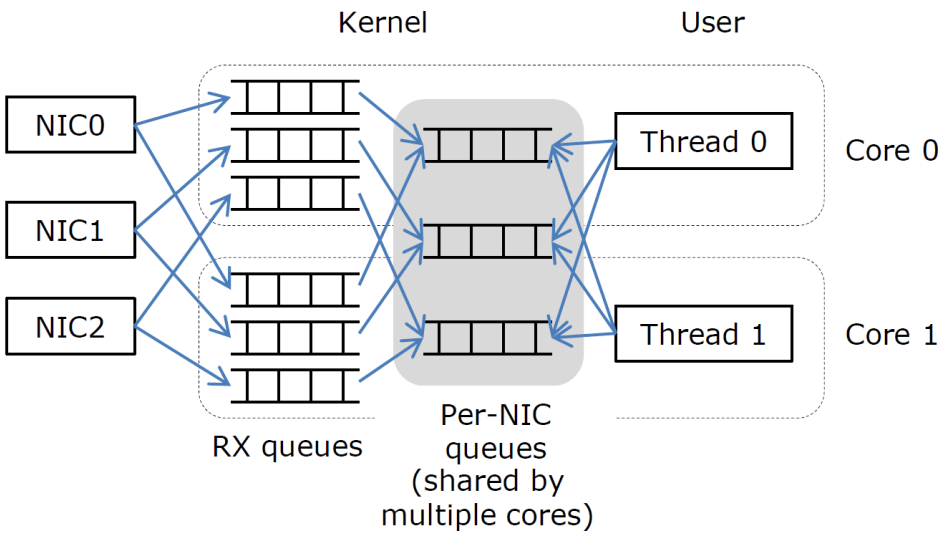
- RSS (Receive-Side Scaling) default behavior
  - RSS-enabled NICs distribute incoming packets into all CPU cores.

- To save bandwidth between NUMA nodes, we prevent packets from crossing the NUMA boundary.



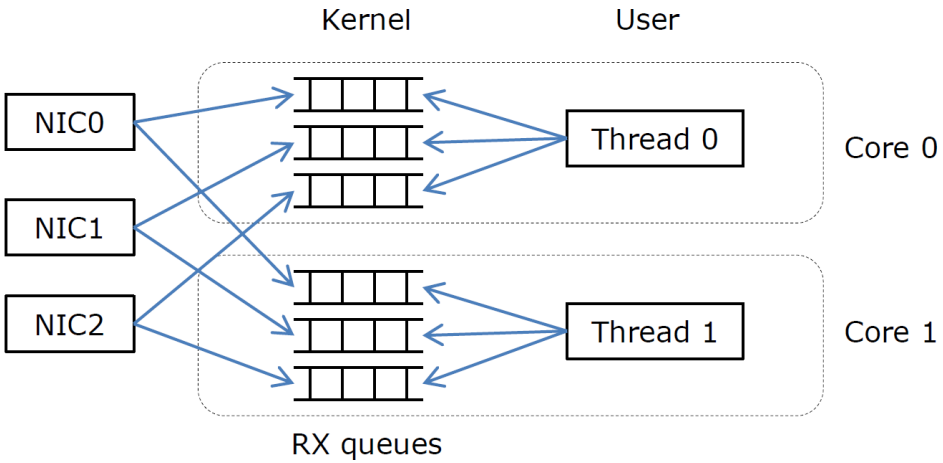
KAIST

Multiqueue-Aware User-space Packet I/O



Existing scheme (ex. libpcap):

Per-NIC queues cause cache bouncing and lock contention



Our multiqueue-aware scheme:

Memory access is partitioned between cores

KAIST

GPU FOR PACKET PROCESSING

Advantages of GPU for Packet Processing

1. Raw computation power
2. Memory access latency
3. Memory bandwidth

- Comparison between
  - Intel X5550 CPU
  - NVIDIA GTX480 GPU

(1/3) Raw Computation Power

- Compute-intensive operations in software routers
  - Hashing, encryption, pattern matching, network coding, compression, etc.
  - GPU can help!

Instructions/sec

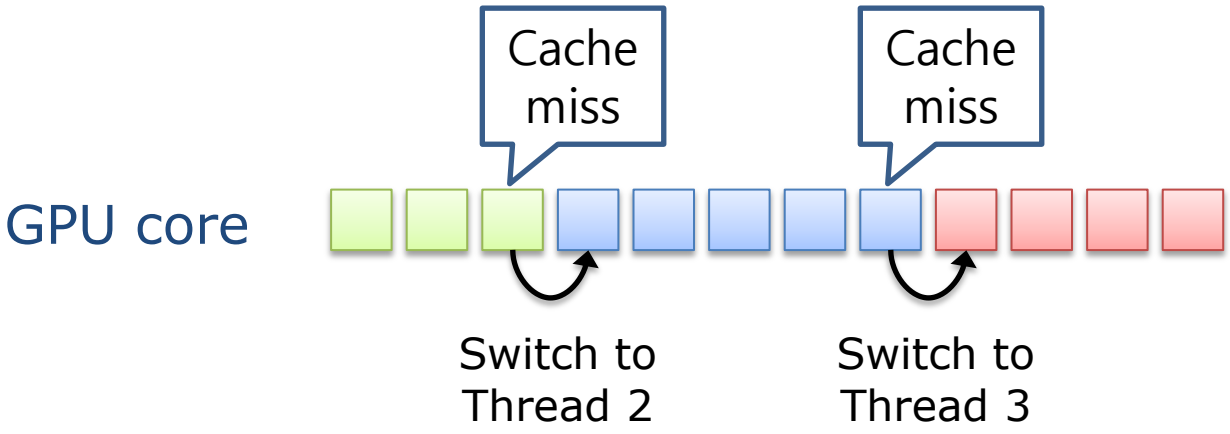
CPU: 43×10^9
= 2.66 (GHz) ×
4 (# of cores) ×
4 (4-way superscalar)

<

GPU: 672×10^9
= 1.4 (GHz) ×
480 (# of cores)

(2/3) Memory Access Latency

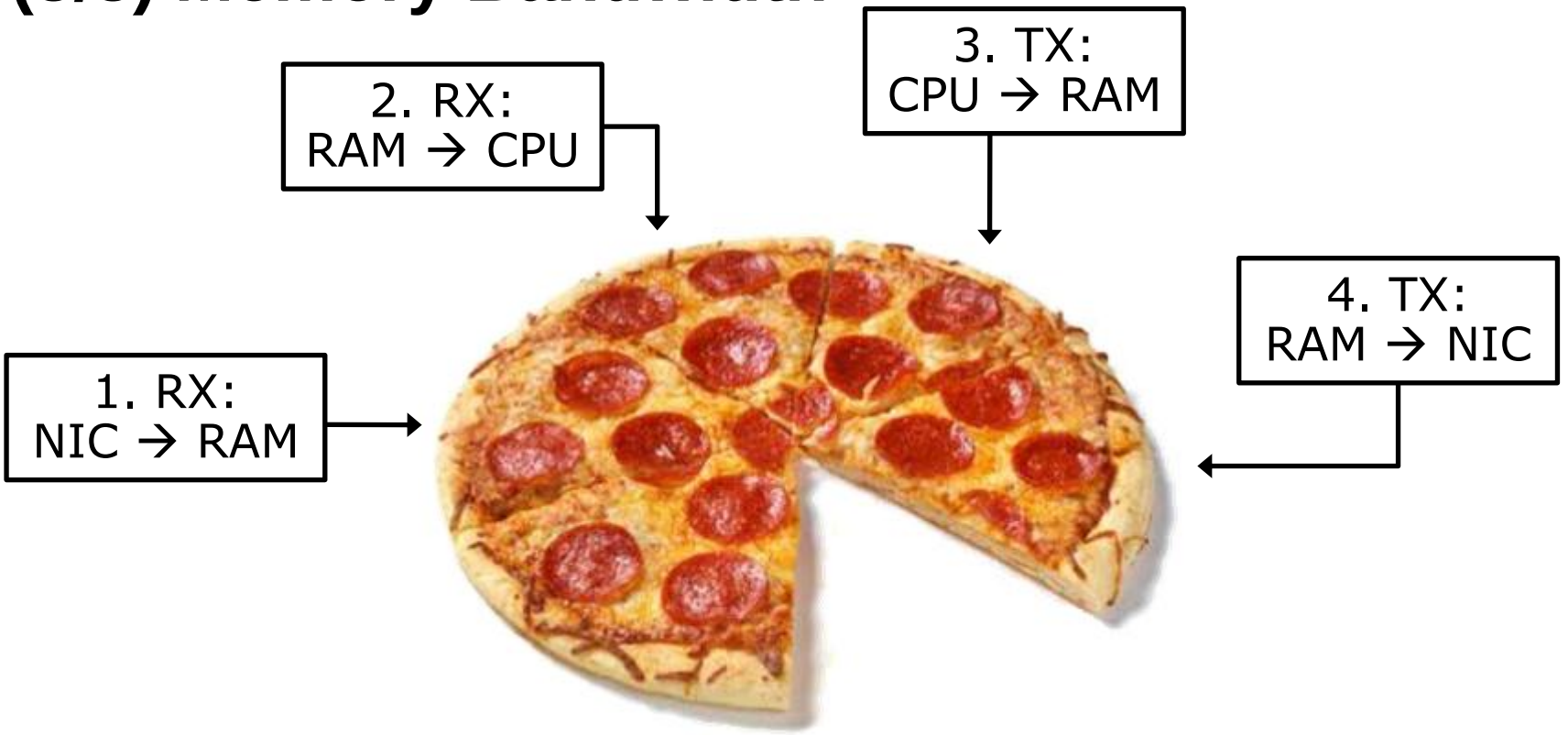
- Software router → lots of cache misses
  - GPU can effectively hide memory latency



(3/3) Memory Bandwidth

CPU’s memory bandwidth (theoretical): 32 GB/s

(3/3) Memory Bandwidth



CPU's memory bandwidth (empirical) < 25 GB/s

(3/3) Memory Bandwidth

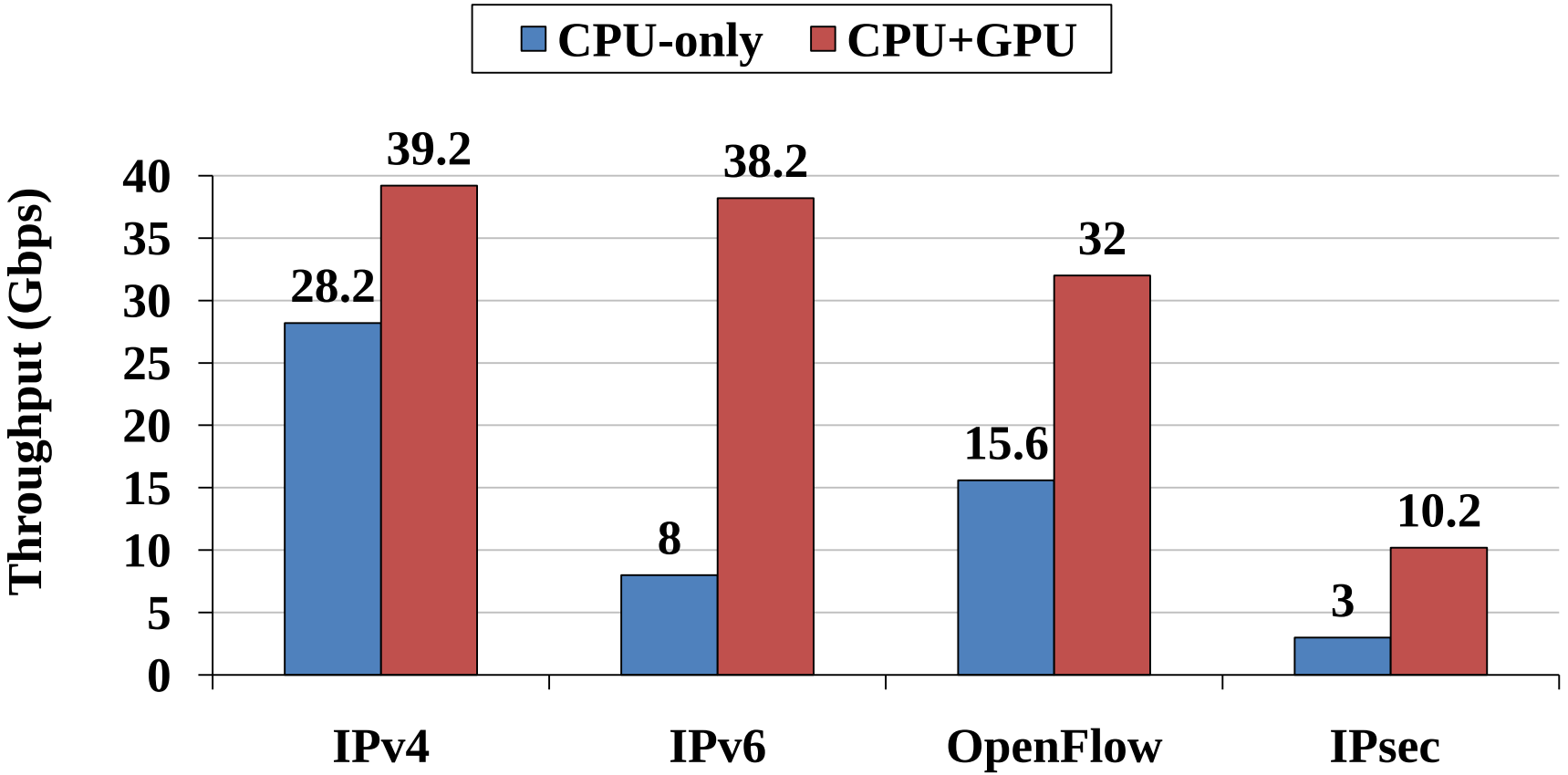
Your budget for packet processing can be less 10 GB/s

(3/3) Memory Bandwidth

~~Your budget for packet processing can be less 10 GB/s~~

GPU's memory bandwidth: 174GB/s

Results (w/ 64B packets)



	IPv4	IPv6	OpenFlow	IPsec
GPU speedup	1.4x	4.8x	2.1x	3.5x

KAIST

Results

Year	Ref.	H/W	IPv4 Throughput
2008	Egi *et al.*	Two quad-core CPUs	3.5 Gbps
2008	"Enhanced SR" Bolla *et al.*	Two quad-core CPUs	4.2 Gbps
2009	"RouteBricks" Dobrescu *et al.*	Two quad-core CPUs (2.8 GHz)	8.7 Gbps
2010	PacketShader (CPU-only)	Two quad-core CPUs (2.66 GHz)	28.2 Gbps
2010	PacketShader (CPU+GPU)	Two quad-core CPUs + two GPUs	39.2 Gbps

Kernel

User

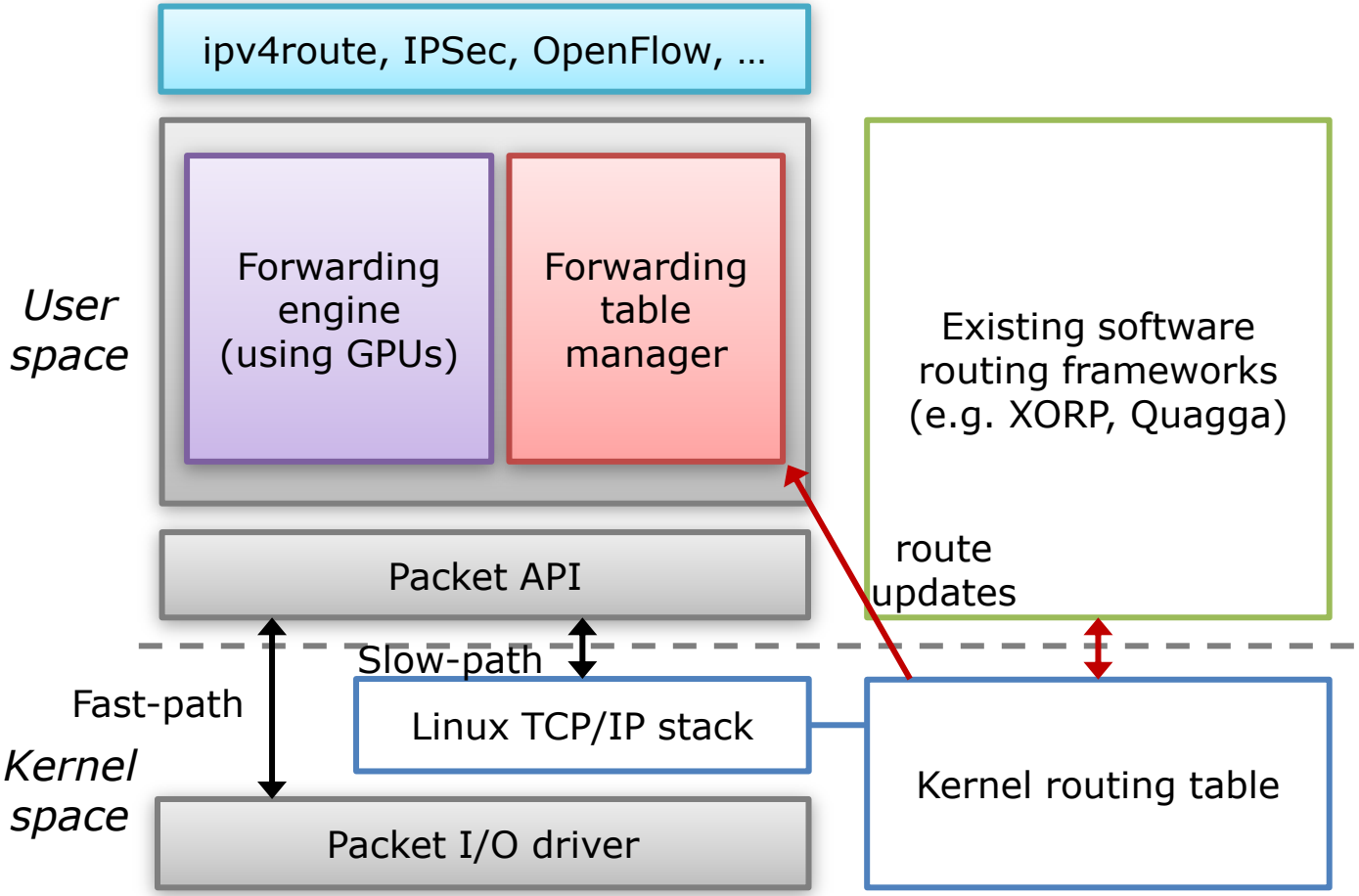
What PacketShader is not

- Working router
  - Control plane missing
  - Microbenchmarked for only single appllications (protocols)
  - Basic protocols not implemented (e.g. ARP, ICMP, ...)

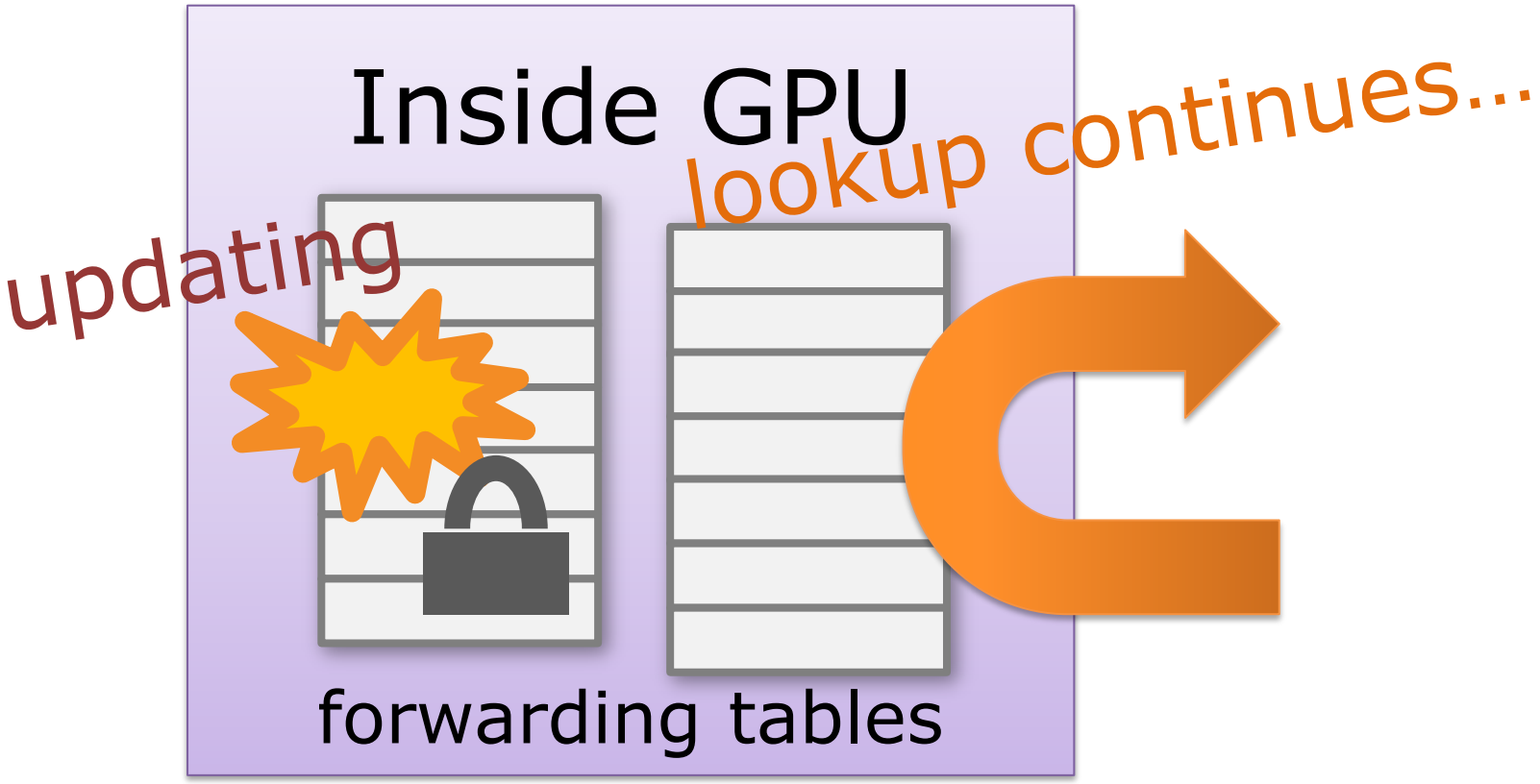
PacketShader 2.0

- Control plane integration
  - Dynamic routing protocols with Quagga or XORP
- Opportunistic offloading
  - CPU at low load
  - GPU at high load
- Multi-functional, modular programming environment
  - Integration with Click? [Kohler99]

#1 Control-plane Integration



Double Buffering



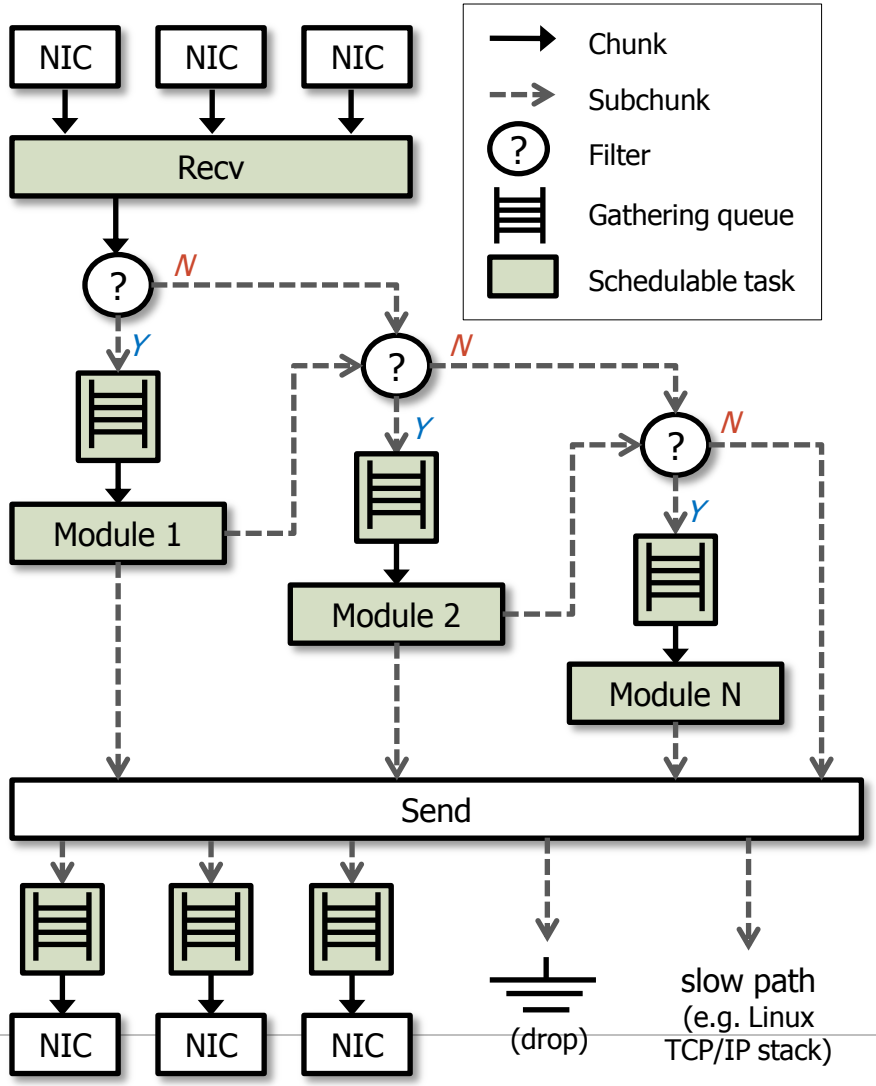
#2 Opportunistic Offloading

- Implemented in SSLShader, our GPU-based SSL accelerator
- Threshold-based switching between CPU-only and CPU+GPU operations

#3 Multi-functional , modular programming environment

#3 Multi-functional, modular programming environment

- Cascading



KAIST

Factors behind PacketShader 1.0 Performance

- Batching, batching, batching!
  - The IO engine (modified NIC driver) uses continuous huge packet buffers called "*chunks*".
  - The user-level process *pipelines* multiple chunks.
  - The GPU processes multiple chunks *in parallel*.
- Hardware-aware optimizations
  - No NUMA node crossing
  - Minimized cache conflicts among multi-cores

Remaining Challenges

- 100+ Gbps speed
- Stateful processing
  - Intrusion detection systems / firewalls

Review of I/O Capacity for 100+ Gbps

- QuickPath Interconnect (QPI)
  - CPU socket-to-socket link for remote memory
  - IOH-to-IOH link for I/O traffic
  - CPU-to-IOH for CPU to peripheral connections
- Today's QPI link
  - 12.8 GB/s or 102.4 Gbps
- Sandy Bridge
  - On recall at the moment
  - Expected to boost performance to 60 Gbps w/o modification

Review of Memory B/W for 100+ Gbps

- For 100Gbps forwarding we need 400 Gbps in memory bandwidth + bookkeeping
- Current configuration
  - triple-channel DDR3 1,333 MHz
  - 32 GB/s per core (theoretical) and 17.9GB/s (empirical)
- On NUMA system
  - More nodes
  - Careful placement…

Future Work

- Consider other architectures
  - AMD's APU
  - Tilera's tiled many-core chips
  - Intel's MIC
- Become a platform for all new FIA architectures
  - Advantage over NetFPGA, ServerSwitch, ATCA solutions
- Who in Korea will take it to full development?

Positioning

- Commercial competitor?
  - Core routers with 100+ Tbps capacity? No.
  - Edge routers with 100+ Gbps capacity with complex features? Maybe
  - Experimental platforms?
    - NetFGPA
    - ServerSwitch
    - RouteBricks
    - ATCA-based boxes

Collaboration opportunities with US

- PlanetLab #1
- OpenFlow / NetFPGA initiatives #1
- NSF GENI

- NSF FIA
  - XIA
  - NDN
  - MobilityFirst
  - Nebula

KAIST

A Collaboration opportunity with EU: OneLab2

- Extension of OneLab
- Open call by 2011.9.15.
- Work Packages
  - Control Plane Interoperability
  - Experimental Plane Interoperability
  - Wireless Testbeds
  - Wired Testbeds
- Private PlanetLab Korea (PPK) already federated
- What more needs to be done?
  - Long-term commitment
  - Designate a technical correspondent
  - Join as a partner by September
  - Apply for support to MKE or KCC

Venues for publicity opportunities

- ACM SIGCOMM conferences / workshops / poster sessions
- SOSP/OSDI, EuroSys, APSys
- ACM CCR (6pg only, 3mon turn-around)
- USENIX ;login

- GENI meetings
- EU FP7 meetings
- AsiaFI summer school, CFI conference

For more details

https://shader.kaist.edu/

QUESTIONS?

THANK YOU!