

PacketShader: A GPU-Accelerated Software Router

Sangjin Han[†]

In collaboration with:

Keon Jang[†], KyoungSoo Park[‡], Sue Moon[†]

[†] Advanced Networking Lab, CS, KAIST

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

PacketShader: A GPU-Accelerated Software Router



High-performance

Our prototype: 40 Gbps on a single box

Software Router

- Despite its name, **not limited to IP routing**
 - You can implement whatever you want on it.
- Driven by software
 - Flexible
 - Friendly development environments
- Based on commodity hardware
 - Cheap
 - Fast evolution

Now 10 Gigabit NIC is a commodity

- From \$200 – \$300 per port
 - Great opportunity for software routers

amazon.com



Chelsio N320E Server Adapter - Network adapter - PCI Express x8 low profile - 10 Gigabit Ethernet - 2 ports

List Price: ~~\$579.00~~
Price: **\$543.99**
You Save: **\$35.01 (6%)**

In Stock.

Achilles' Heel of Software Routers

- Low performance
 - Due to CPU bottleneck



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


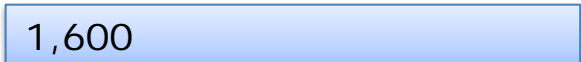
- Not capable of supporting even a single 10G port


CPU BOTTLENECK

Per-Packet CPU Cycles for 10G

Cycles
needed

IPv4  +  = 1,800 cycles
Packet I/O IPv4 lookup

IPv6  +  = 2,800
Packet I/O IPv6 lookup

IPsec  +  ...  = 6,600
Packet I/O Encryption and hashing

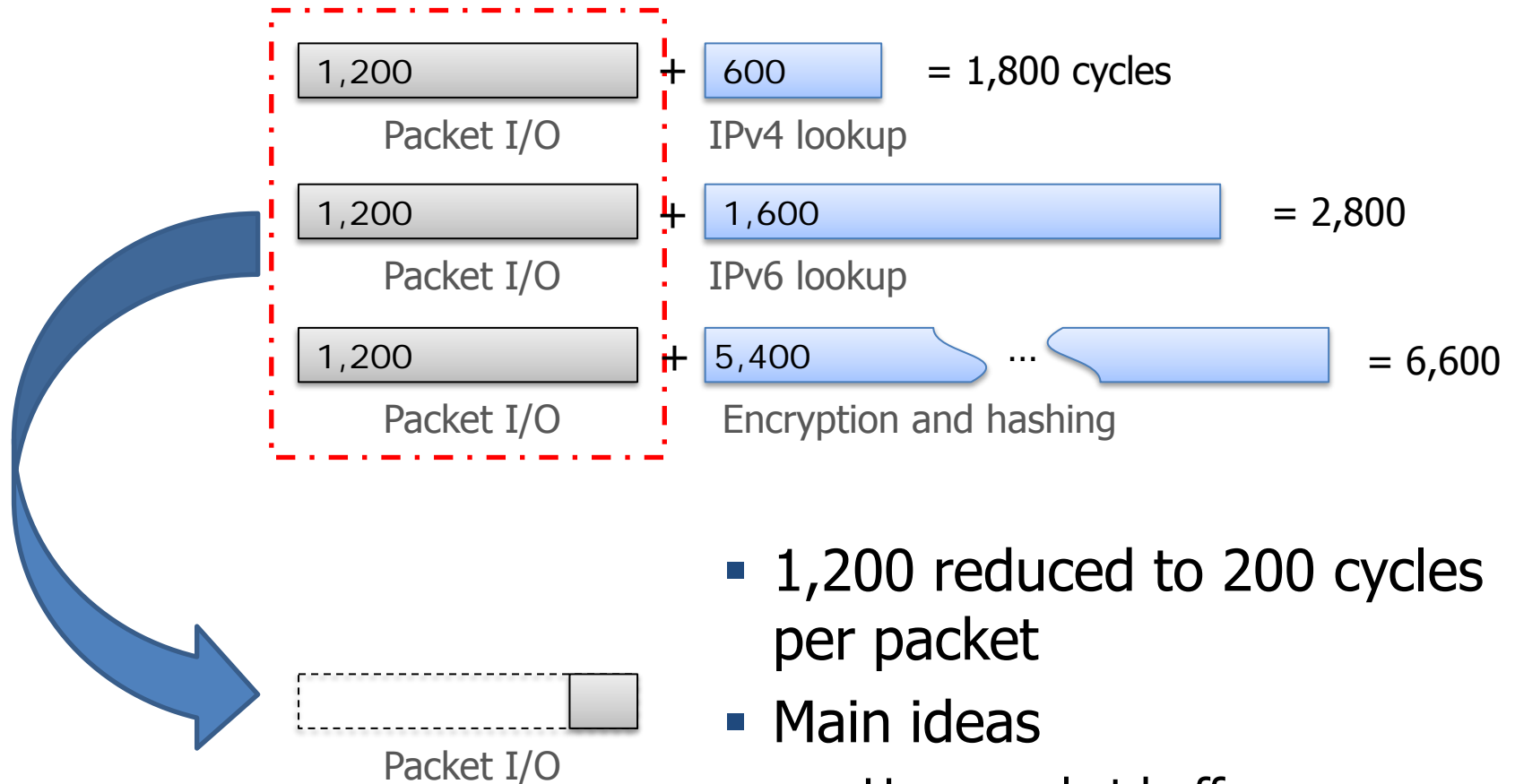
Your
budget

 1,400 cycles

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

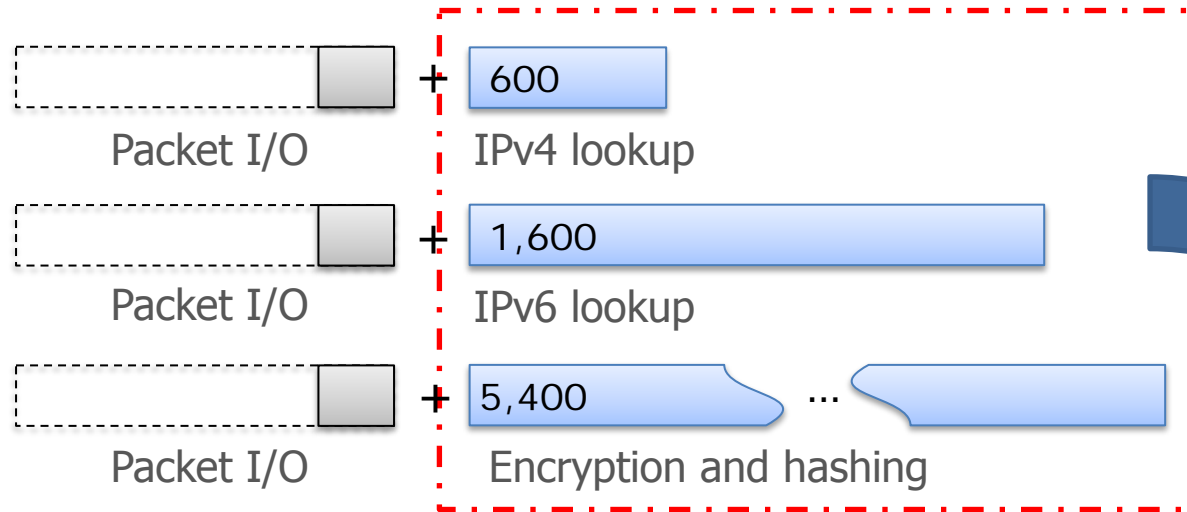
(in x86, cycle numbers are from RouteBricks [Dobrescu09] and ours)

Our Approach 1: I/O Optimization



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

Our Approach 2: GPU Offloading



- GPU Offloading for
 - Memory-intensive or
 - Compute-intensive operations
- Main topic of this talk



WHAT IS GPU?

GPU = Graphics Processing Unit

- The heart of graphics cards
- Mainly used for real-time 3D game rendering
 - Massively-parallel processing capacity



(Ubisoft's AVARTAR, from <http://ubi.com>)

CPU vs. GPU



CPU:
Small # of super-fast cores



GPU:
Large # of small cores

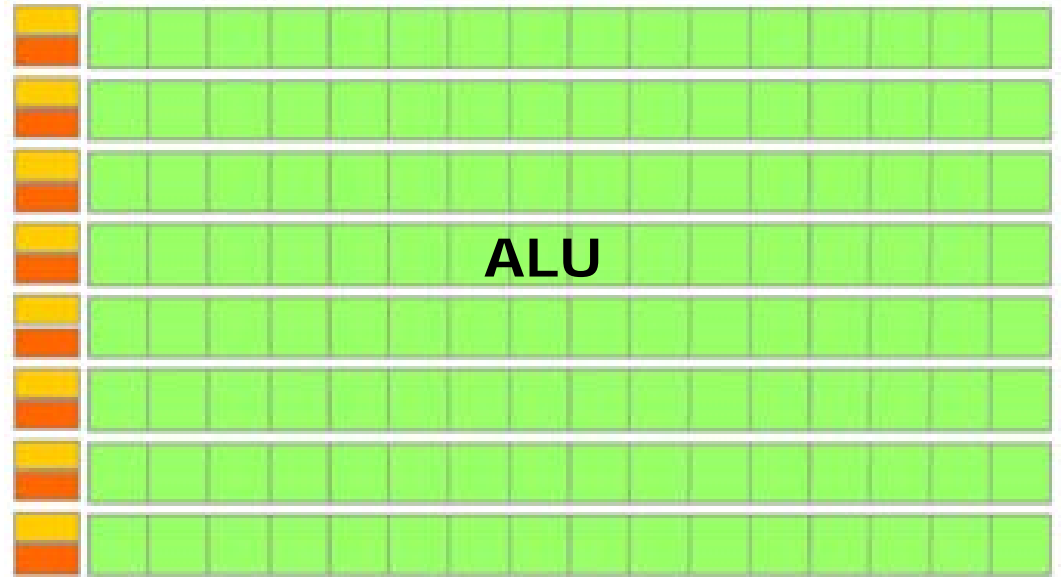
“Silicon Budget” in CPU and GPU



Xeon X5550:

4 cores

731M transistors



GTX480:

480 cores

3,200M transistors

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) \times
4 (# of cores) \times
4 (4-way superscalar)

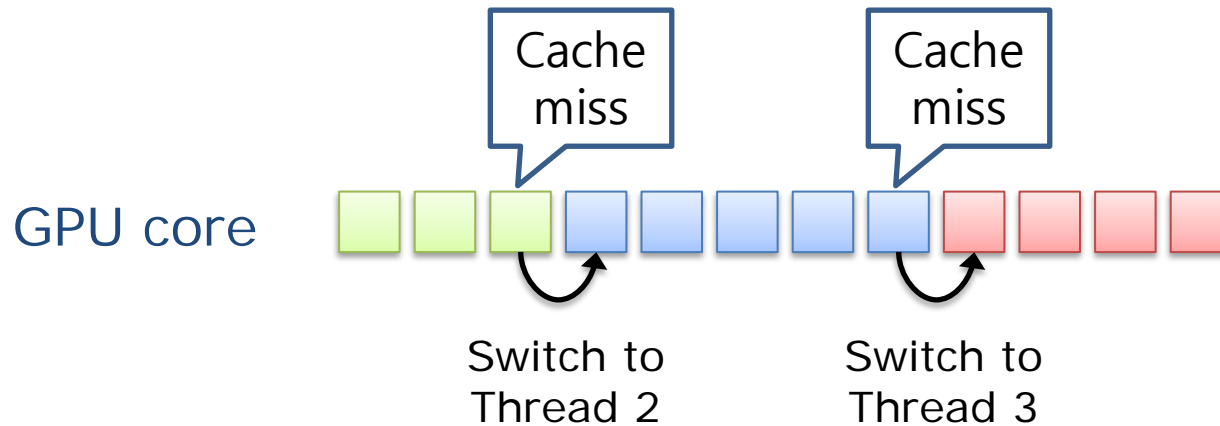
<



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

(2/3) Memory Access Latency

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

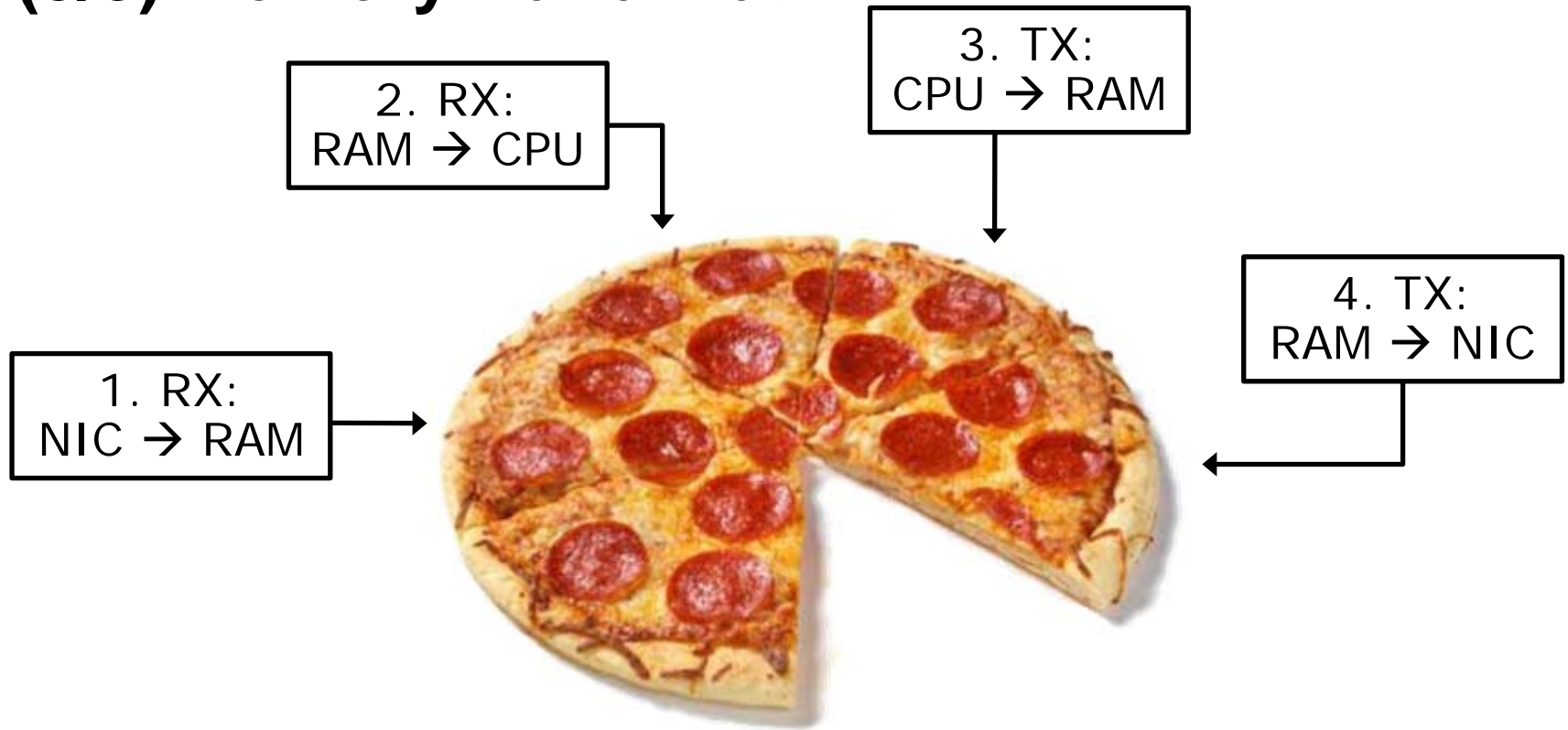


(3/3) Memory Bandwidth



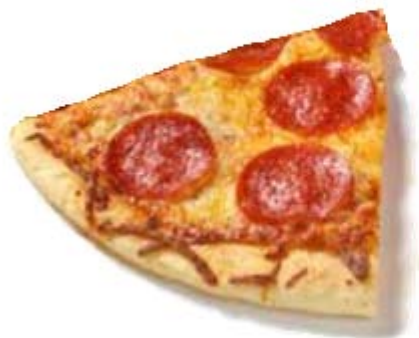
CPU's memory bandwidth (theoretical): 32 GB/s

(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

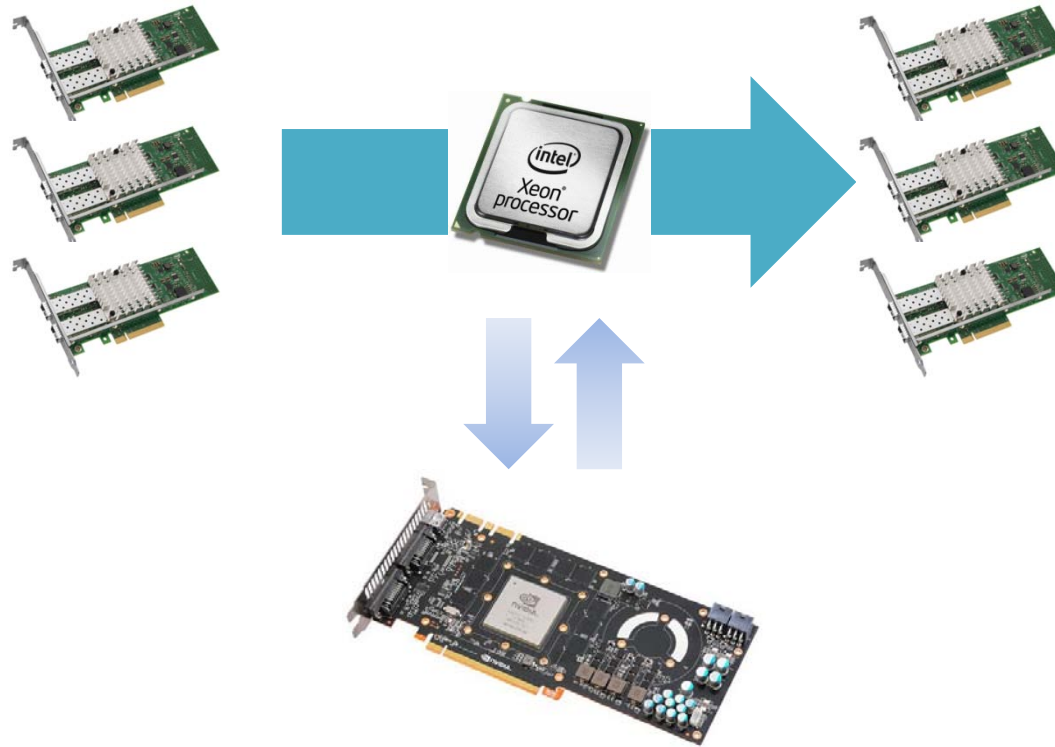
(3/3) Memory Bandwidth



~~Your budget for packet processing can be less 10 GB/s~~
GPU's memory bandwidth: 174GB/s

HOW TO USE GPU

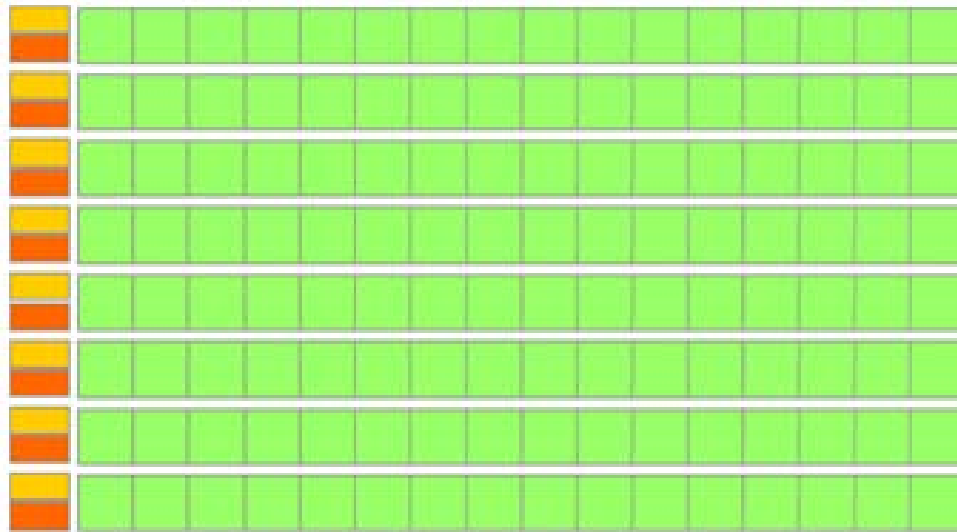
Basic Idea



Offload core operations to GPU
(e.g., forwarding table lookup)

Recap

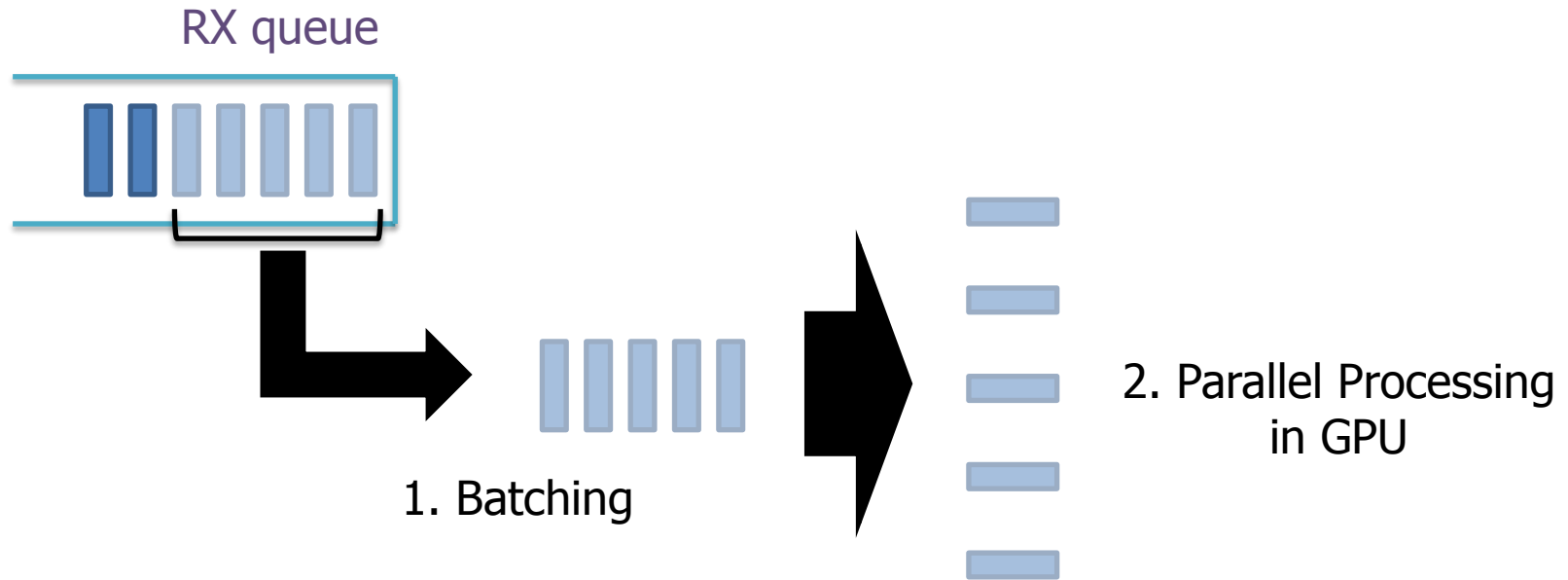
- For GPU, more parallelism, more throughput



GTX480: 480 cores

Parallelism in Packet Processing

- The key insight
 - Stateless packet processing = parallelizable



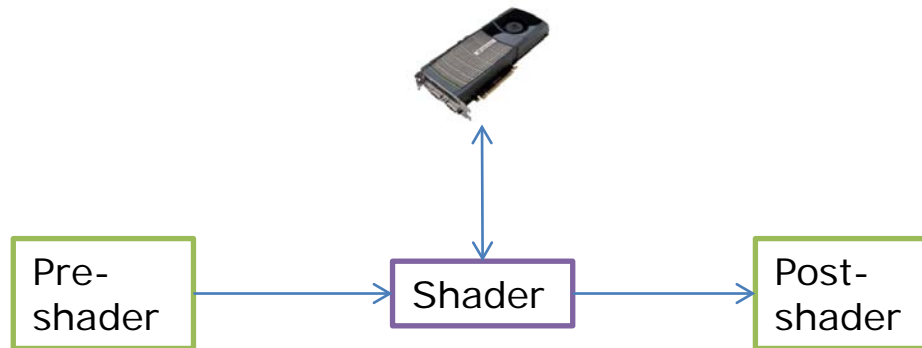
Batching → Long Latency?

- Fast link = enough # of packets in a small time window
- 10 GbE link
 - up to 1,000 packets only in 67 μ s
- Much less time with 40 or 100 GbE

PACKETSHADER DESIGN

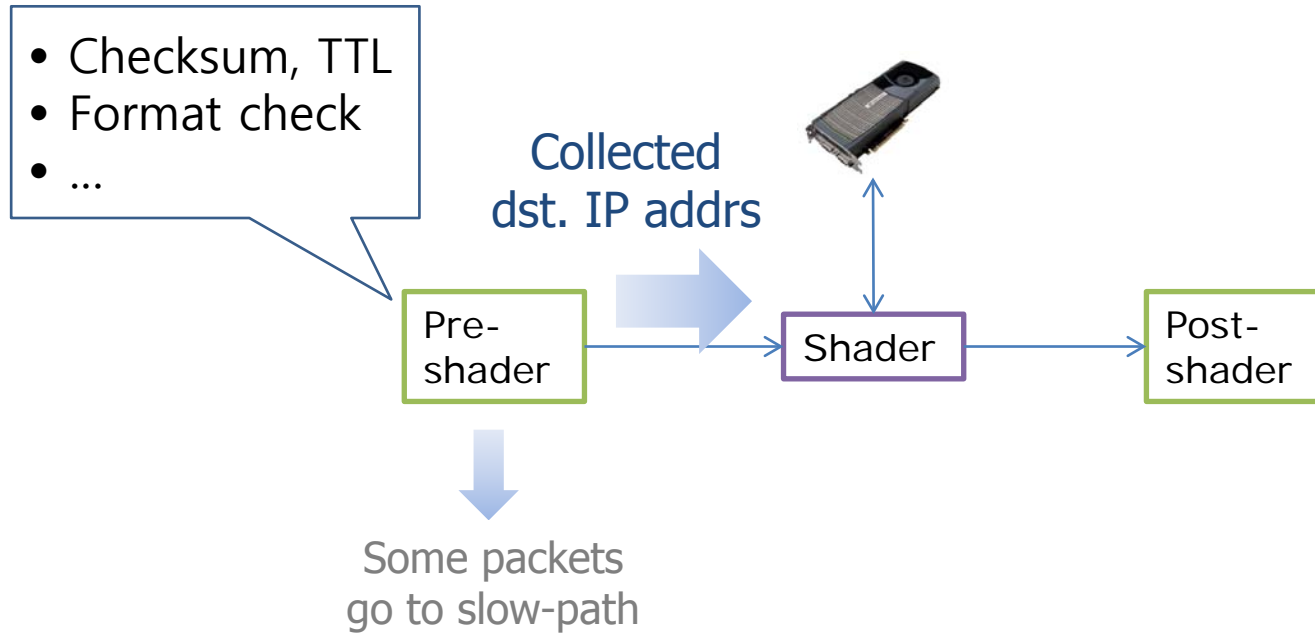
Basic Design

- Three stages in a streamline



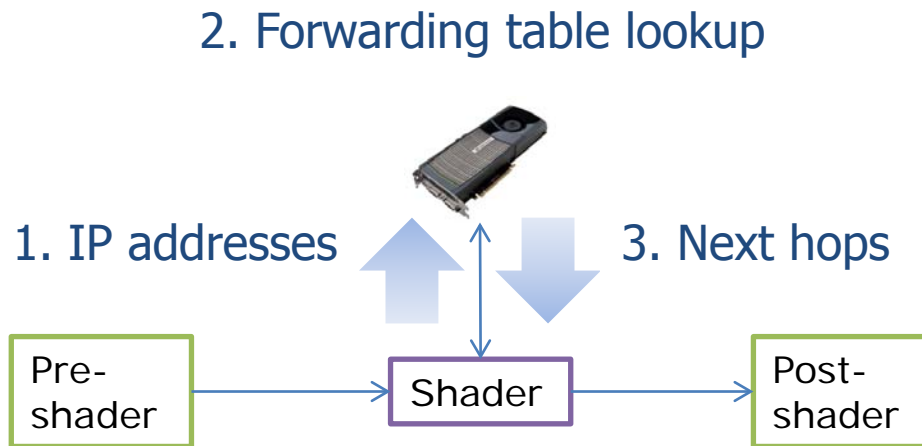
Packet's Journey (1/3)

- IPv4 forwarding example



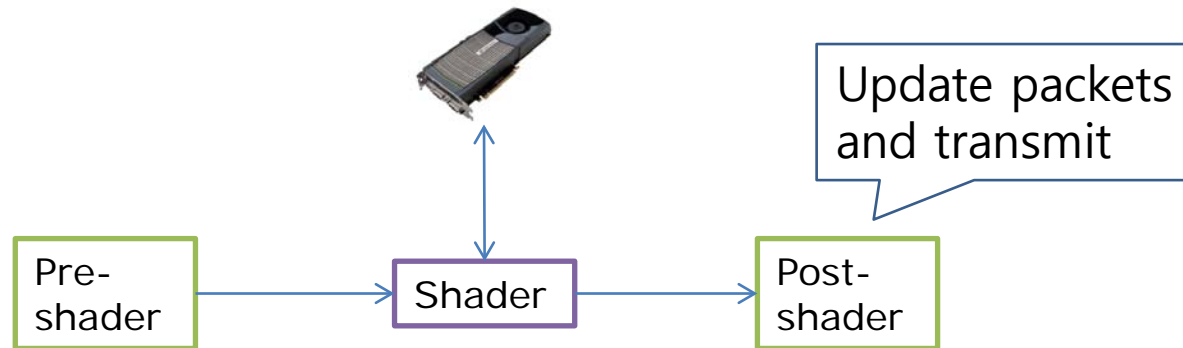
Packet's Journey (2/3)

- IPv4 forwarding example

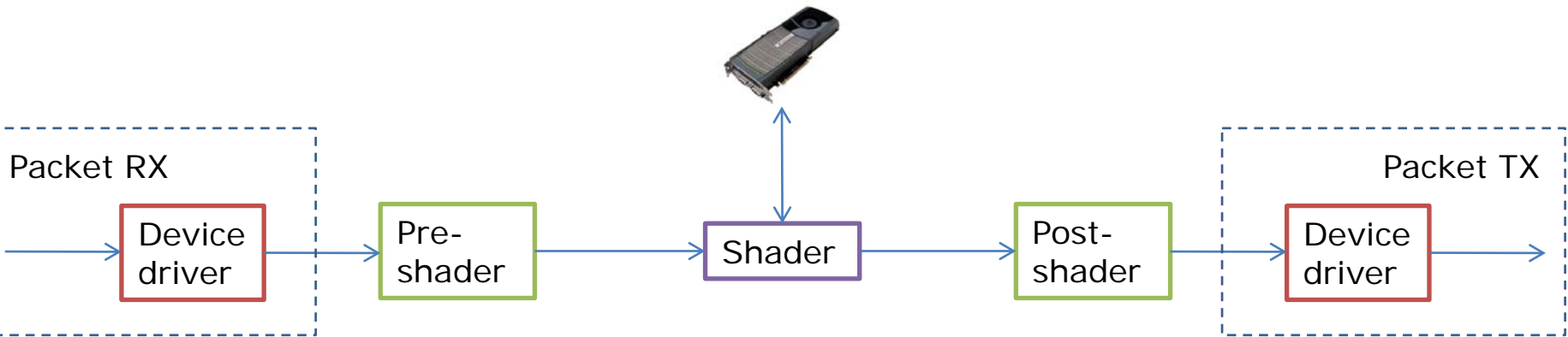


Packet's Journey (3/3)

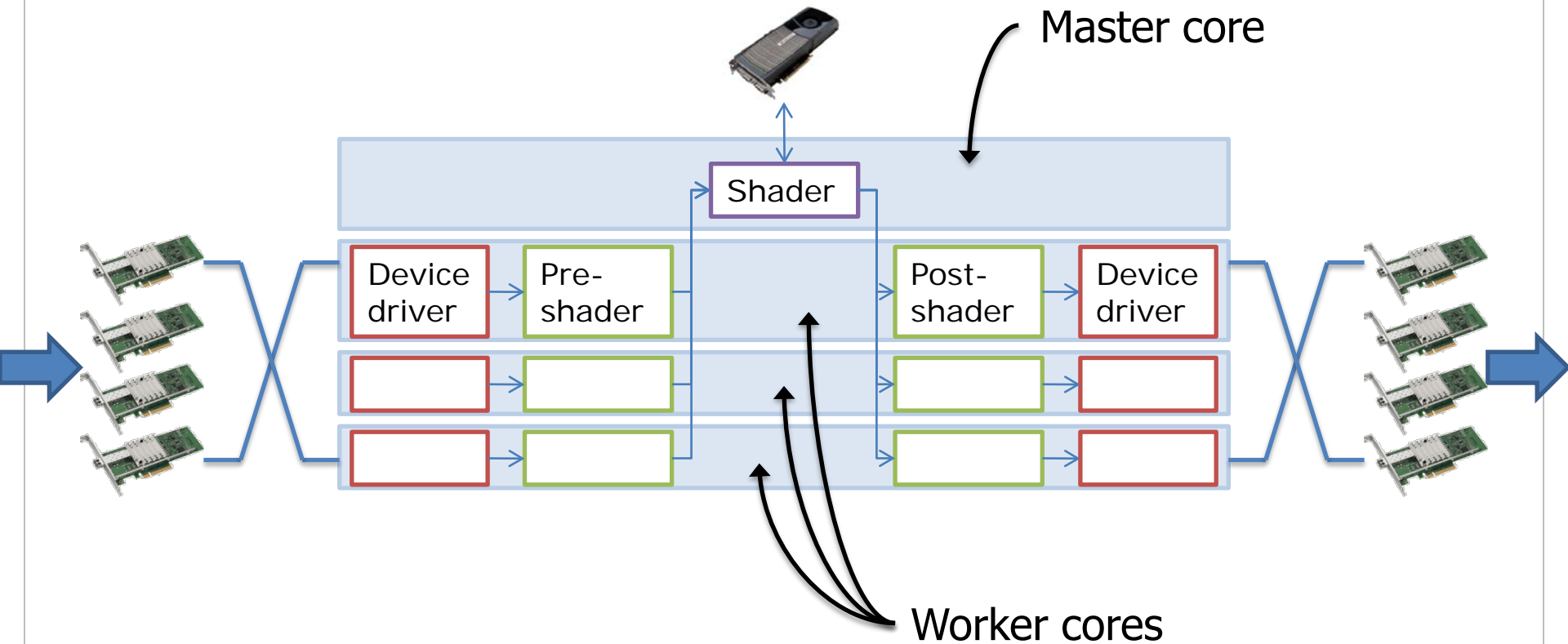
- IPv4 forwarding example



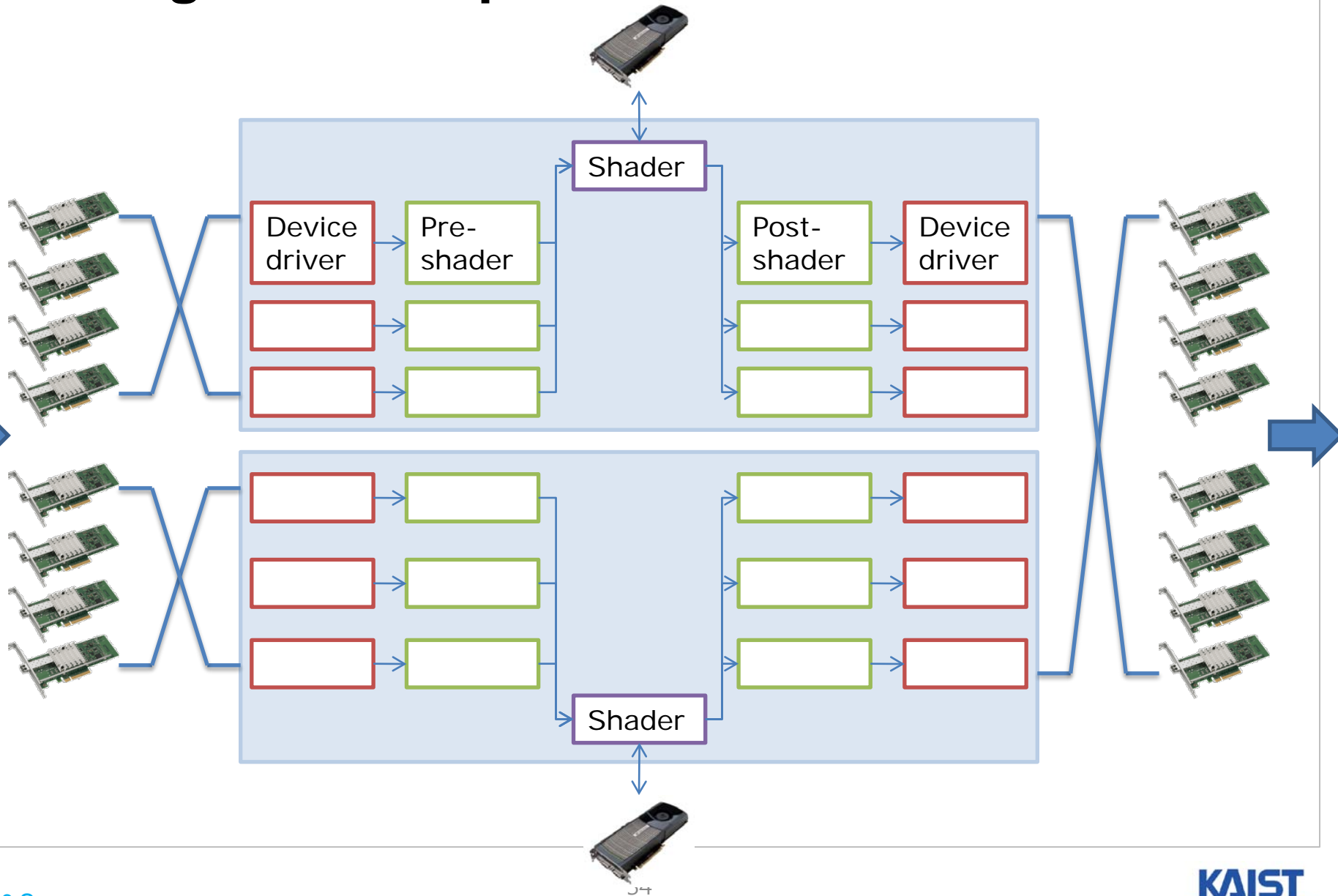
Interfacing with NICs



Scaling with a Multi-Core CPU



Scaling with Multiple Multi-Core CPUs



EVALUATION

Hardware Setup

CPU:



Quad-core, 2.66 GHz

Total 8 CPU cores

NIC:



Dual-port 10 GbE

Total 80 Gbps

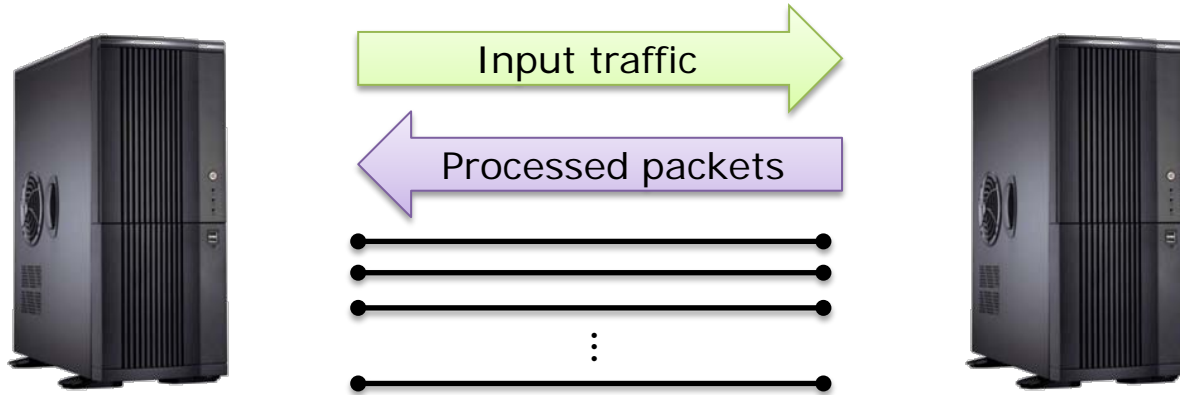
GPU:



480 cores, 1.4 GHz

Total 960 cores

Experimental Setup

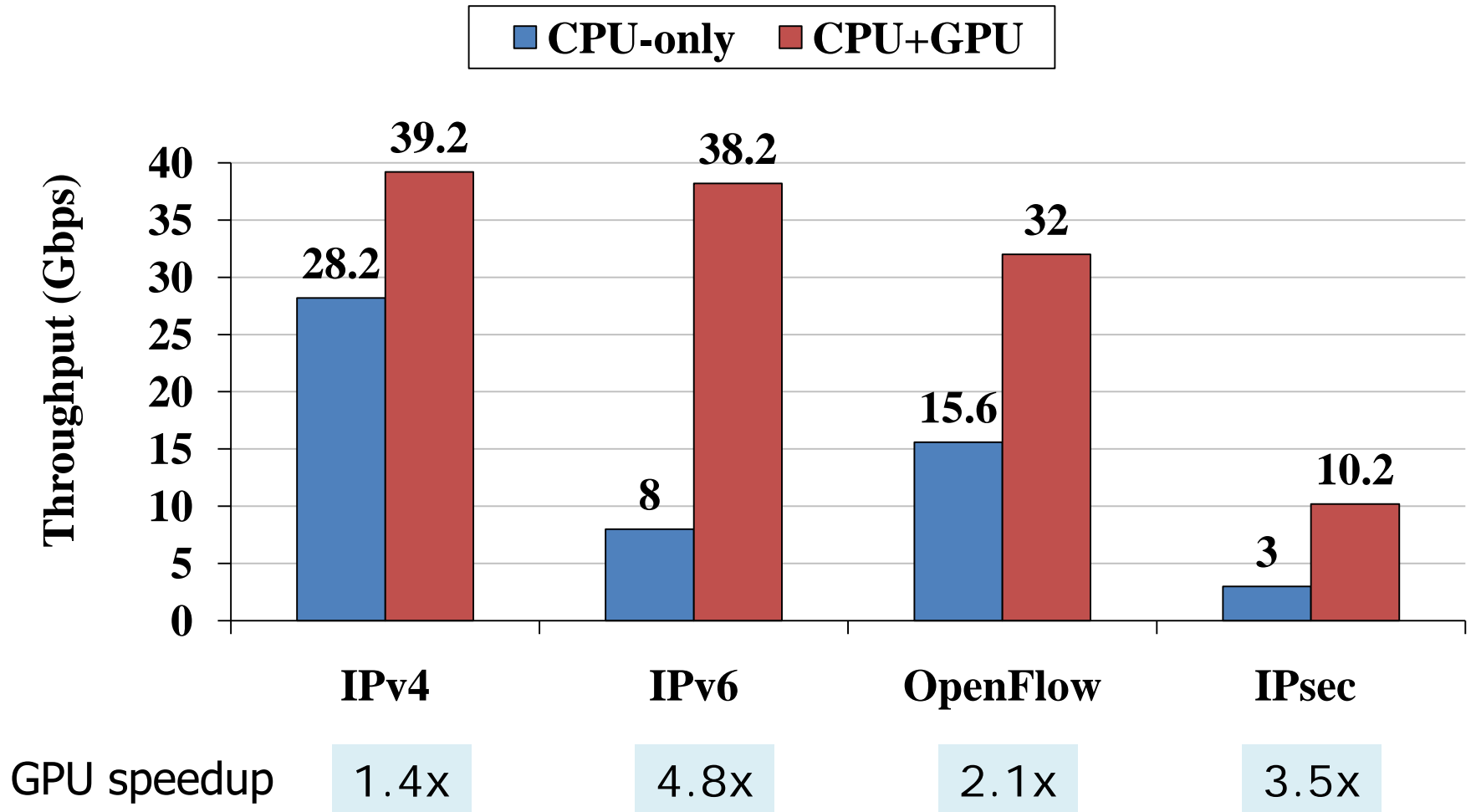


Packet generator
(Up to 80 Gbps)

8 × 10 GbE links

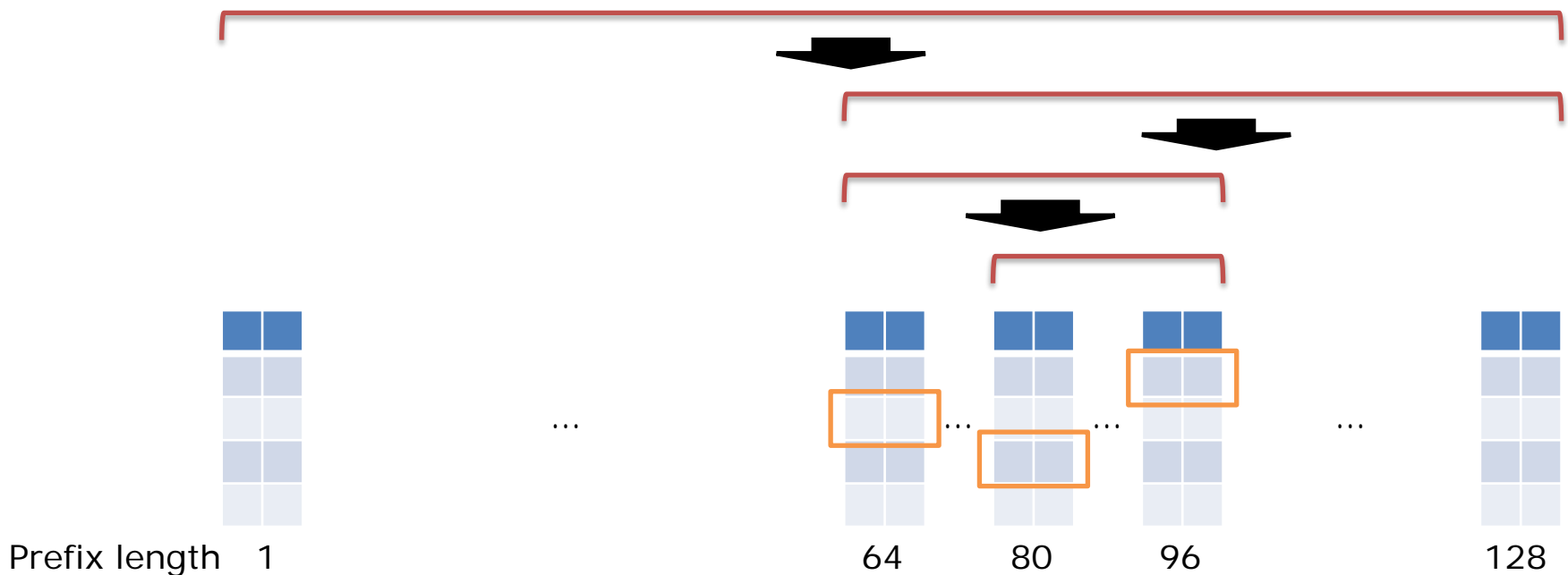
PacketShader

Results (w/ 64B packets)

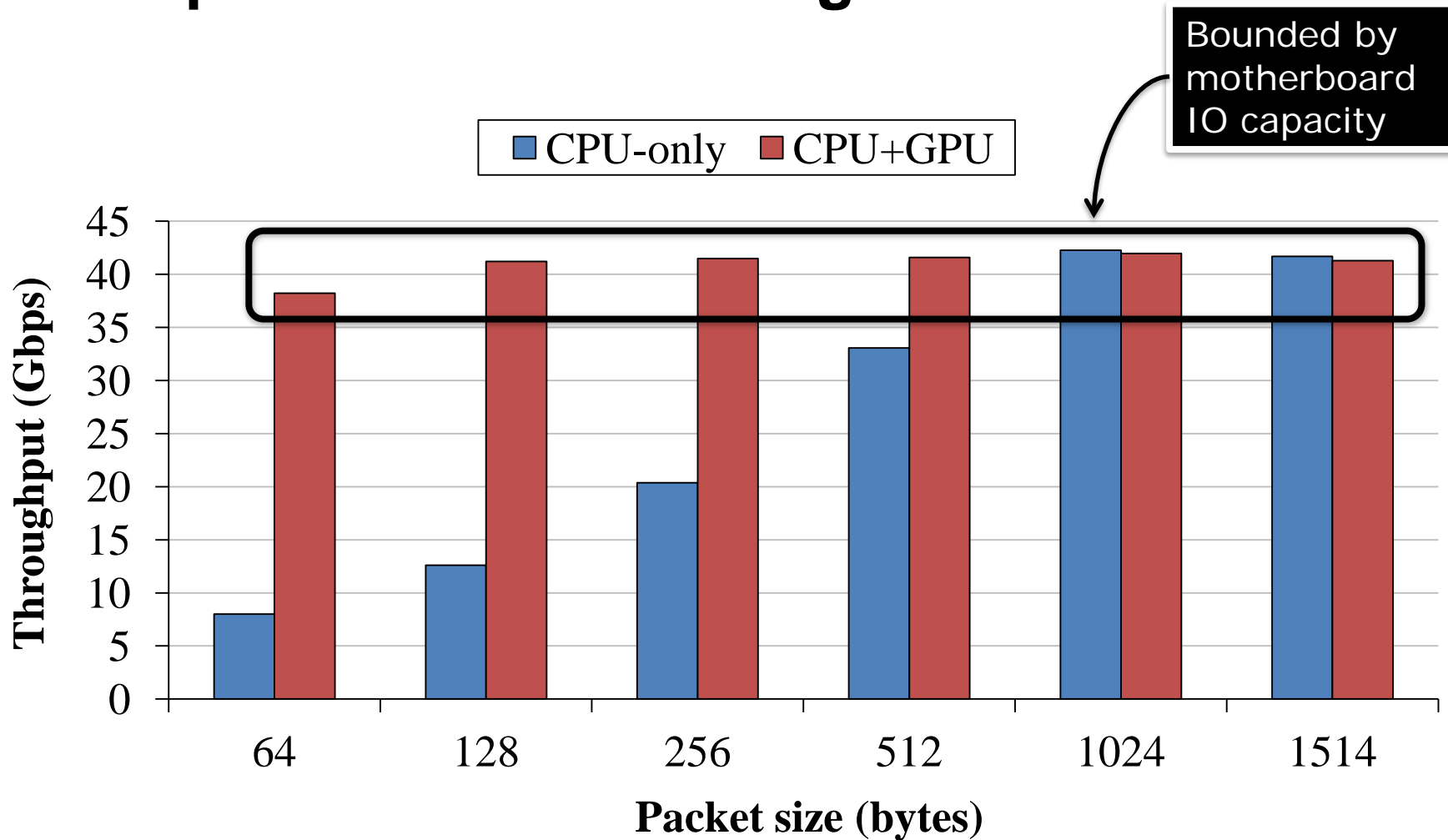


Example 1: IPv6 forwarding

- Longest prefix matching on 128-bit IPv6 addresses
- Algorithm: binary search on hash tables [Waldvogel97]
 - 7 hashings + 7 memory accesses



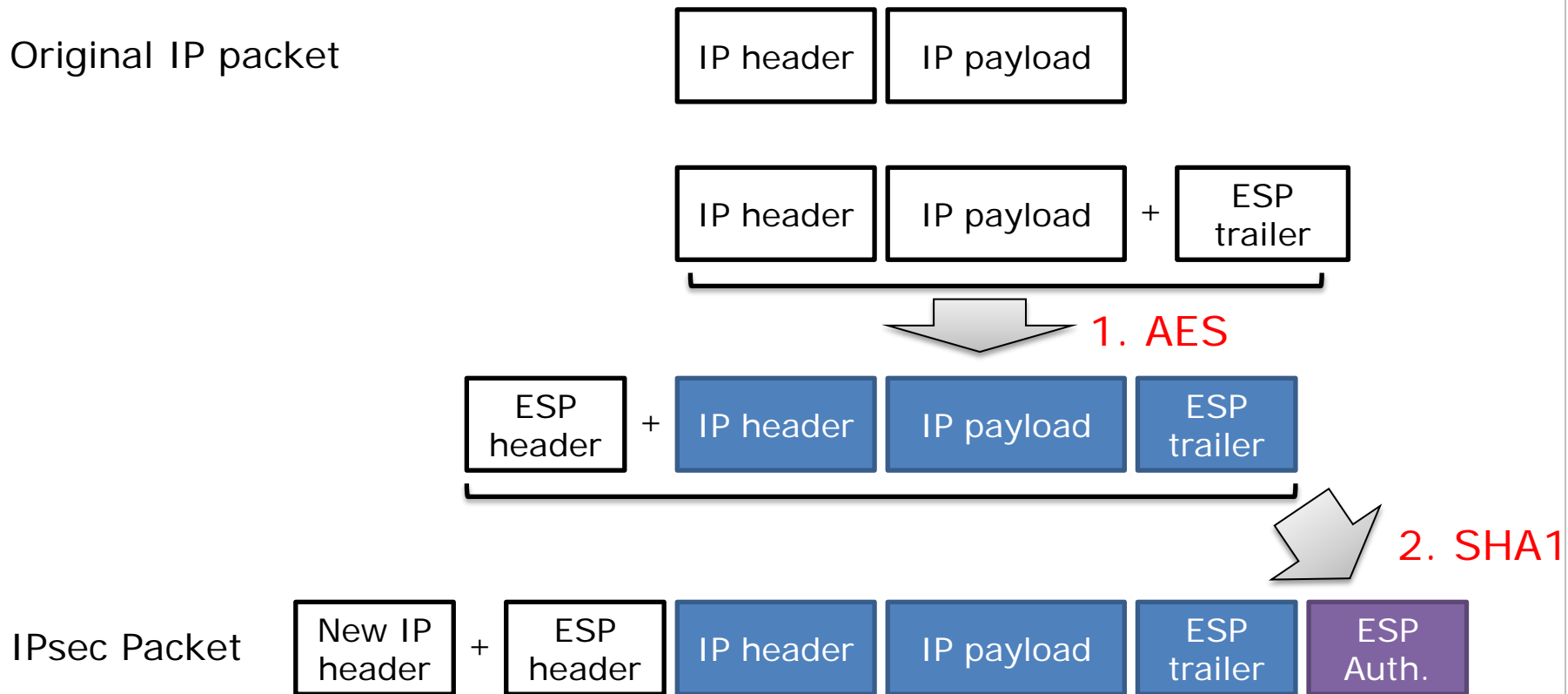
Example 1: IPv6 forwarding



(Routing table was randomly generated with 200K entries)

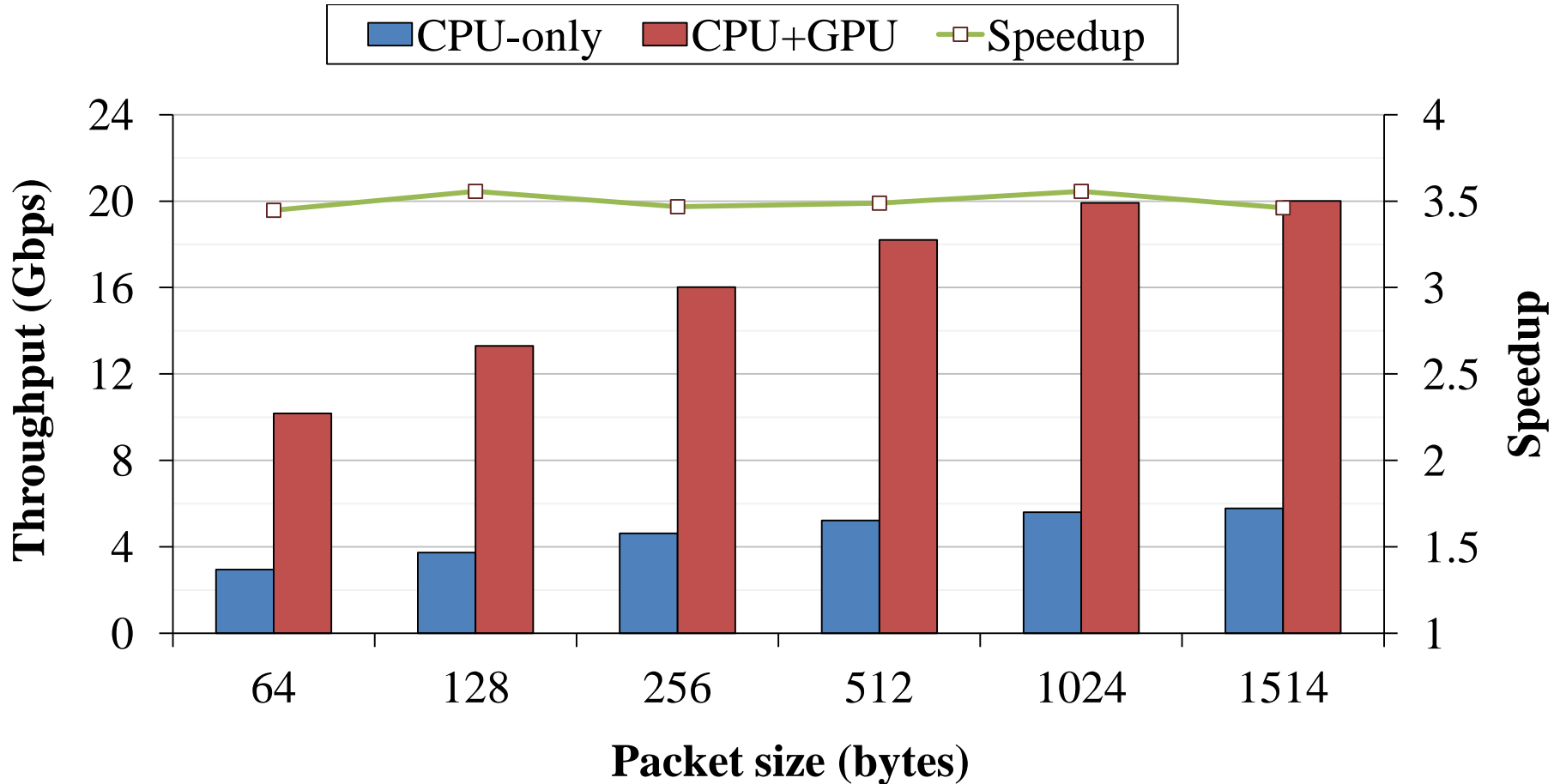
Example 2: IPsec tunneling

- ESP (Encapsulating Security Payload) Tunnel mode
 - with AES-CTR (encryption) and SHA1 (authentication)



Example 2: IPsec tunneling

- 3.5x speedup



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

Conclusions

- GPU
 - a great opportunity for fast packet processing
- PacketShader
 - 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

Future Work

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