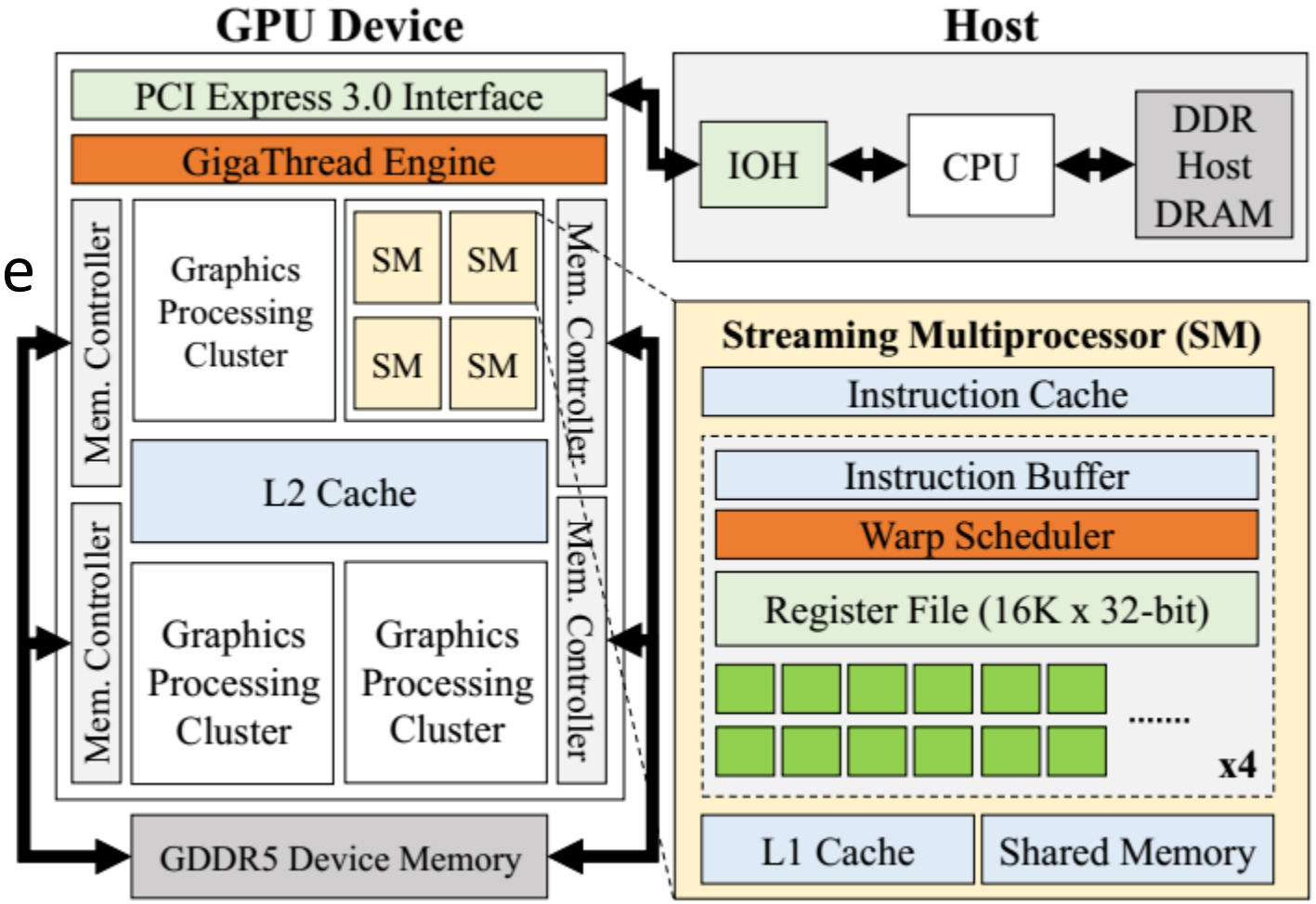
APUNet: Revitalizing GPU as Packet Processing Accelerator

Younghwan Go, Muhammad Jamshed, YoungGyoun Moon, Changho Hwang, and KyoungSoo Park
School of Electrical Engineering, KAIST

Discrete GPU and integrated GPU

- Discrete GPU
  Typical discrete GPU takes the form of a PCIe peripheral device that communicates with the host side via PCIe lanes.

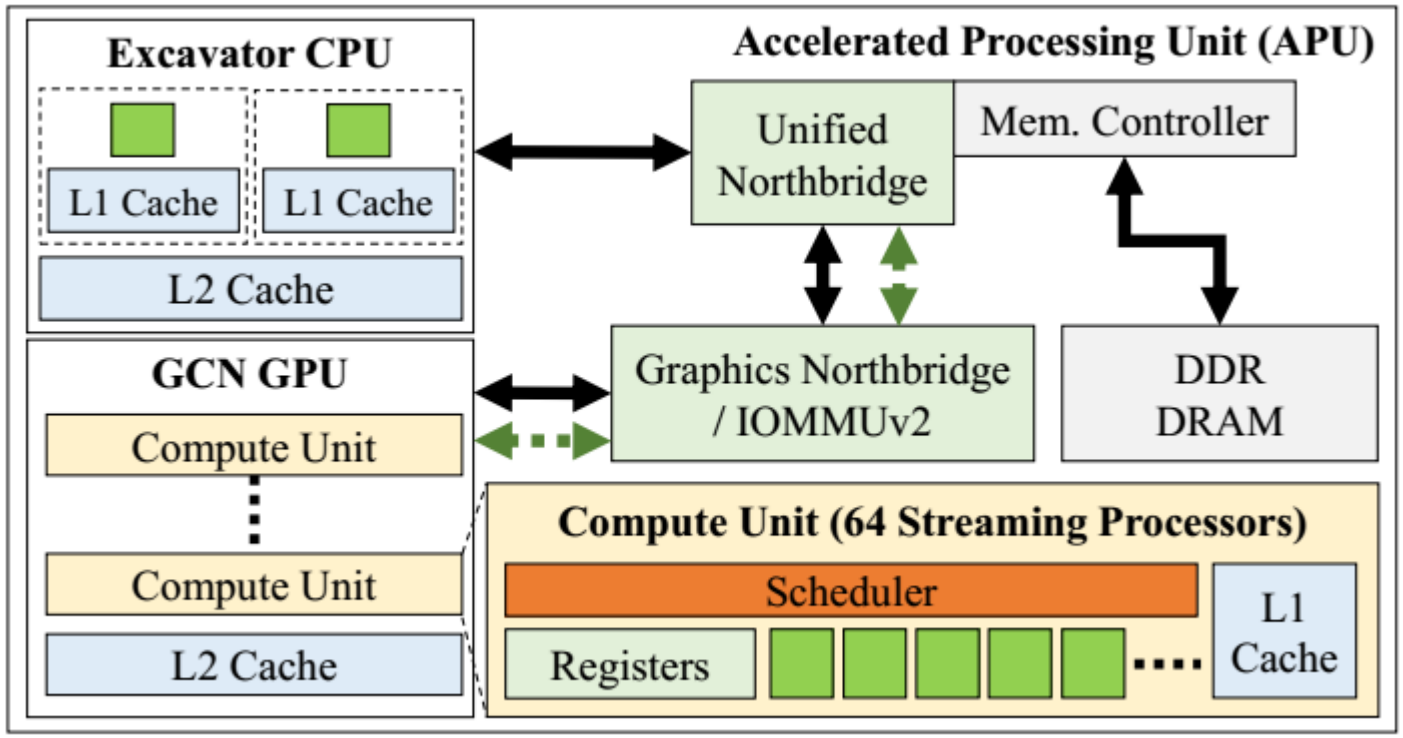


Discrete GPU and integrated GPU

- Integrated GPU

Integrated GPU is manufactured into the same chip as CPU and it shares the DRAM with CPU.

AMD's Accelerated Processing Unit (APU) or Intel HD Graphics



Comparison

- Memory

The memory of integrated GPU is shared with CPU. Discrete GPU has its own memory.

Advantage: shared memory results in efficient data sharing between CPU and GPU. Discrete GPU has to perform expensive DMA transactions over PCIe.

Disadvantage: shared memory greatly increases the contention on the memory controller and reduces the memory bandwidth for each processor.

- Computation capability

Typical integrated GPU has lower computation capacity than discrete GPU as it has a smaller number of processing cores.

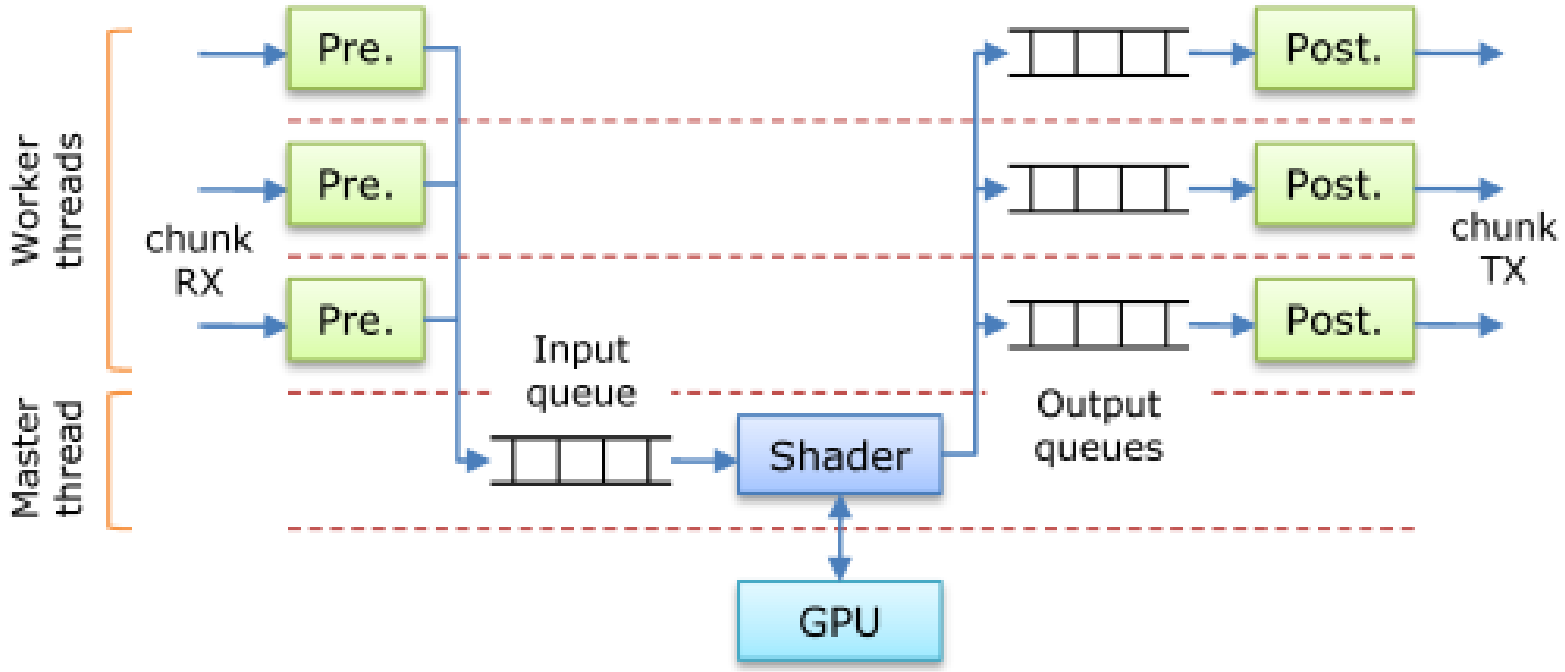
Summary of work

- It refutes a point that most benefits of GPU come from its memory access latency hiding instead of its high computational power raised by another paper published on NSDI 2015 (*Raising the Bar for Using GPUs in Software Packet Processing*).

- It employs integrated GPU in recent APU platforms as the packet processing accelerator to design an APU-based packet processing platform.

Background

- Most GPU-accelerated networked systems have employed discrete GPU to enhance the performance of network applications.



A previous paper

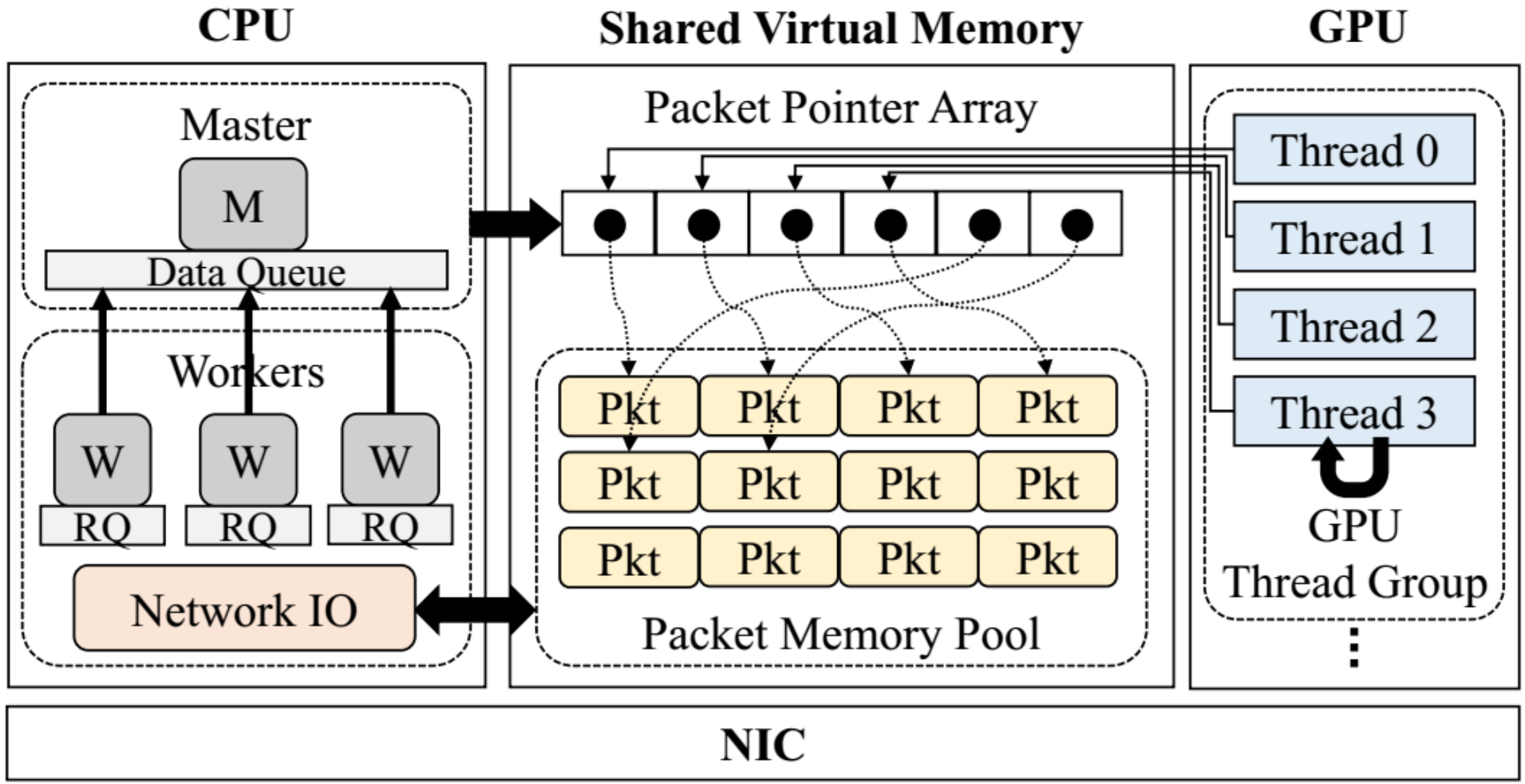
- It argues that for many packet processing applications, the key advantage of a GPU is not its computational power, but that it can hide the 60-200ns of latency required to retrieve data from memory.

- It implements a framework called G-Opt that can accelerate CPU-based packet handling programs by automatically hiding memory access latency. G-Opt achieves the same or even better performance compared with GPU accelerated program.

APUnet

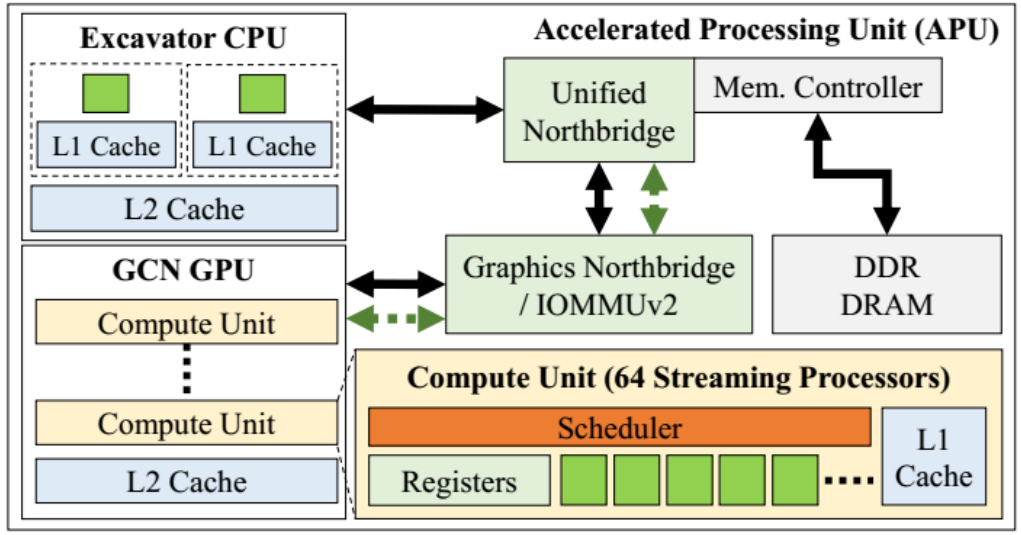
- There are many compute-bound algorithms that do benefit from the parallel computation capacity of GPU while CPU-based optimizations fail to help.
- The relative performance advantage of CPU over GPU in most applications is due to data transfer bottleneck in PCIe communication of discrete GPU rather than lack of capacity of GPU itself .
- To avoid the PCIe bottleneck, APUnet employs APU framework to build a packet processing platform.

Overall framework of APUnet



Challenges

- First, sharing of DRAM incurs contention on the memory controller and bus, which potentially reduces the effective memory bandwidth for each processor.
- Second, frequent GPU kernel launch and teardown incur a high overhead in packet processing.
- Last, APU does not ensure cache coherency across CPU and GPU and it requires expensive atomics operations to synchronize the data in the cache of the two processors.



Solutions

- Zero-copy Packet Processing
- Persistent Thread Execution
- Group Synchronization

Zero-copy Packet Processing

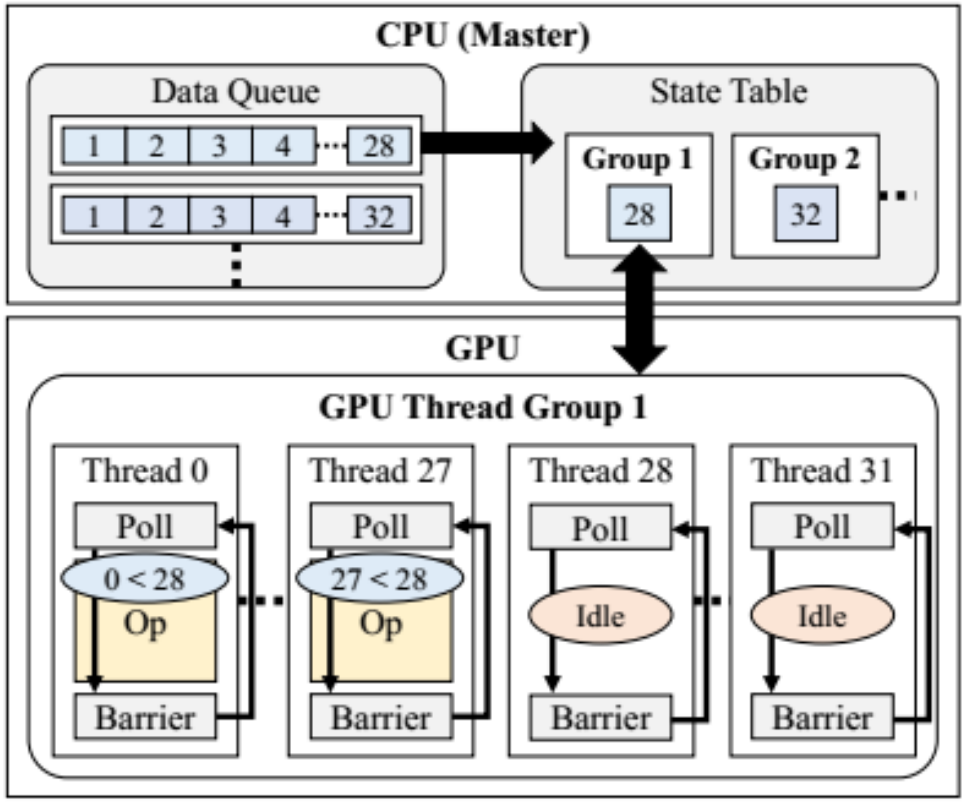
- OpenCL supports shared virtual memory (SVM) that presents the same virtual address space for both CPU and GPU.
- In APU, one can implement zerocopying with SVM and its OpenCL API.
- They update the DPDK code to turn off the hugepage mode, and make it use the SVM memory allocator to create packet buffers.

Persistent Thread Execution

- The basic idea is to keep a large number of GPU threads running to process a stream of input packets continuously.
- At every dequeuer of master's data queue, master groups the packets, and pass them to GPU as a batch. Each CU(compute unit) in GPU has four SIMT units and each unit consists of 16 work items (or threads) that execute the same instruction stream in parallel.
- They use 32 as the batch size in the implementation, and threads in two SIMT units (that we call as GPU thread group) process the packets in parallel.

Persistent Thread Execution

- If the number of dequeued packets is smaller than the group size, keeping the remaining GPU threads in the group idle to make the number align with the group size.
- When new packets arrive, the master thread finds an idle thread group and activates the group by setting the value of its group state to the number of packets to process.



Group Synchronization

- It exploits the LRU cache replacement policy of GPU to evict dirty data items in GPU cache to memory.

- It exploit idle GPU thread groups that are currently not processing packets such that they carry out dummy data I/O on a separate memory block and cause existing data in L2 cache to be evicted.

Evaluation

- Test Setup

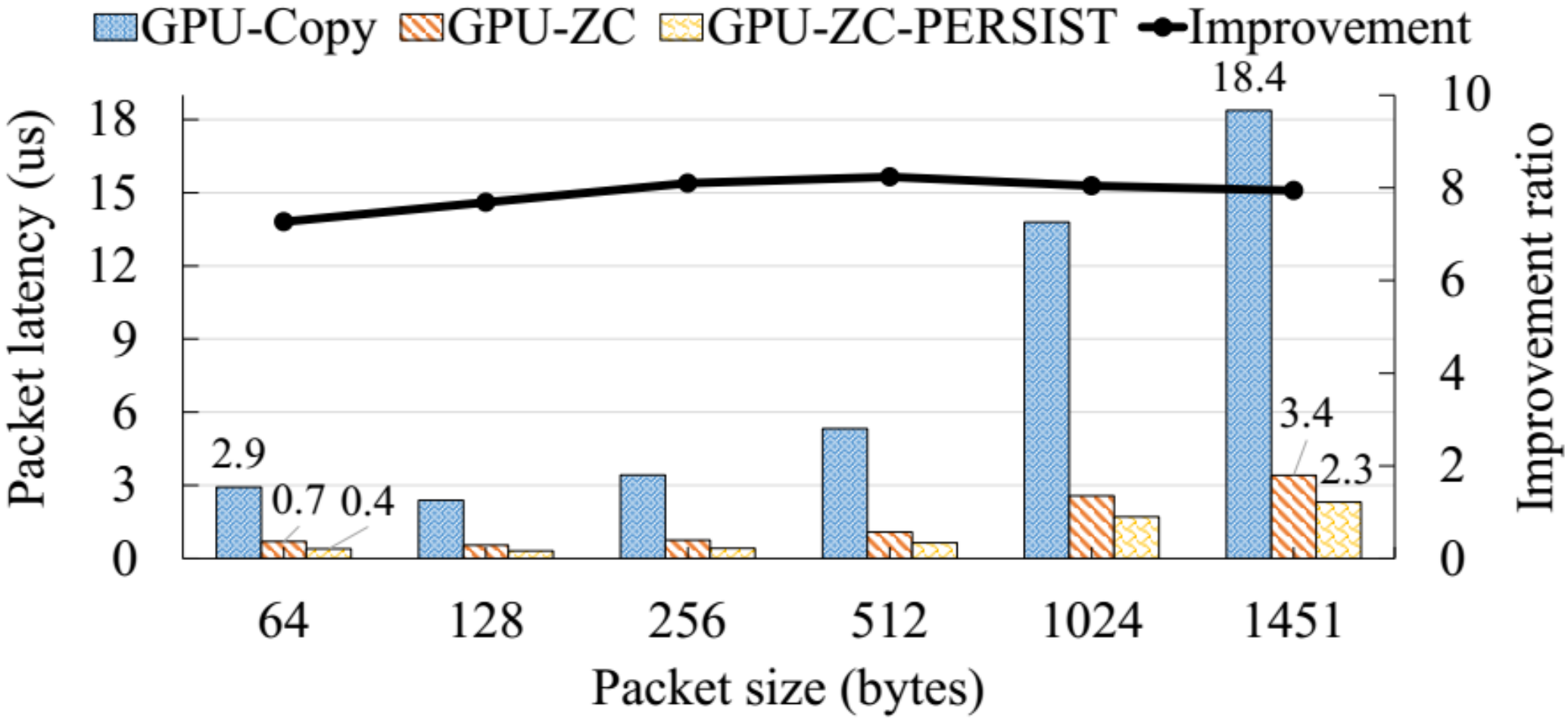
Evaluating APUNet on the AMD APU platform as a packet processing server.

APUNet uses one CPU core as a master communicating with GPU while three worker cores handle packet I/O.

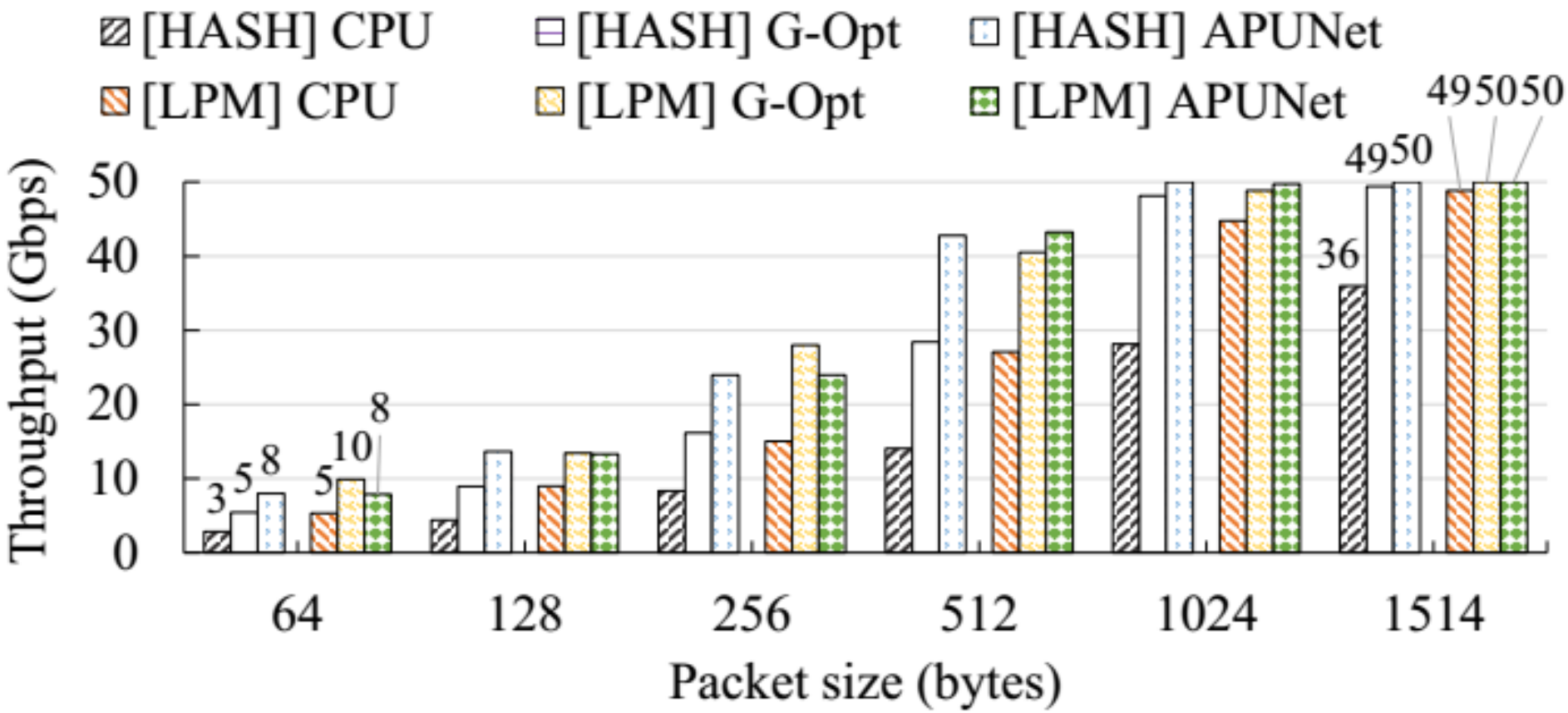
Using a client machine equipped with an octa-core Intel Xeon E3-1285 v4 (3.50 GHz) and 32GB RAM.

Both server and client communicate through a dual-port 40 Gbps Mellanox ConnectX-4 NIC (MCX414A-B [3]).

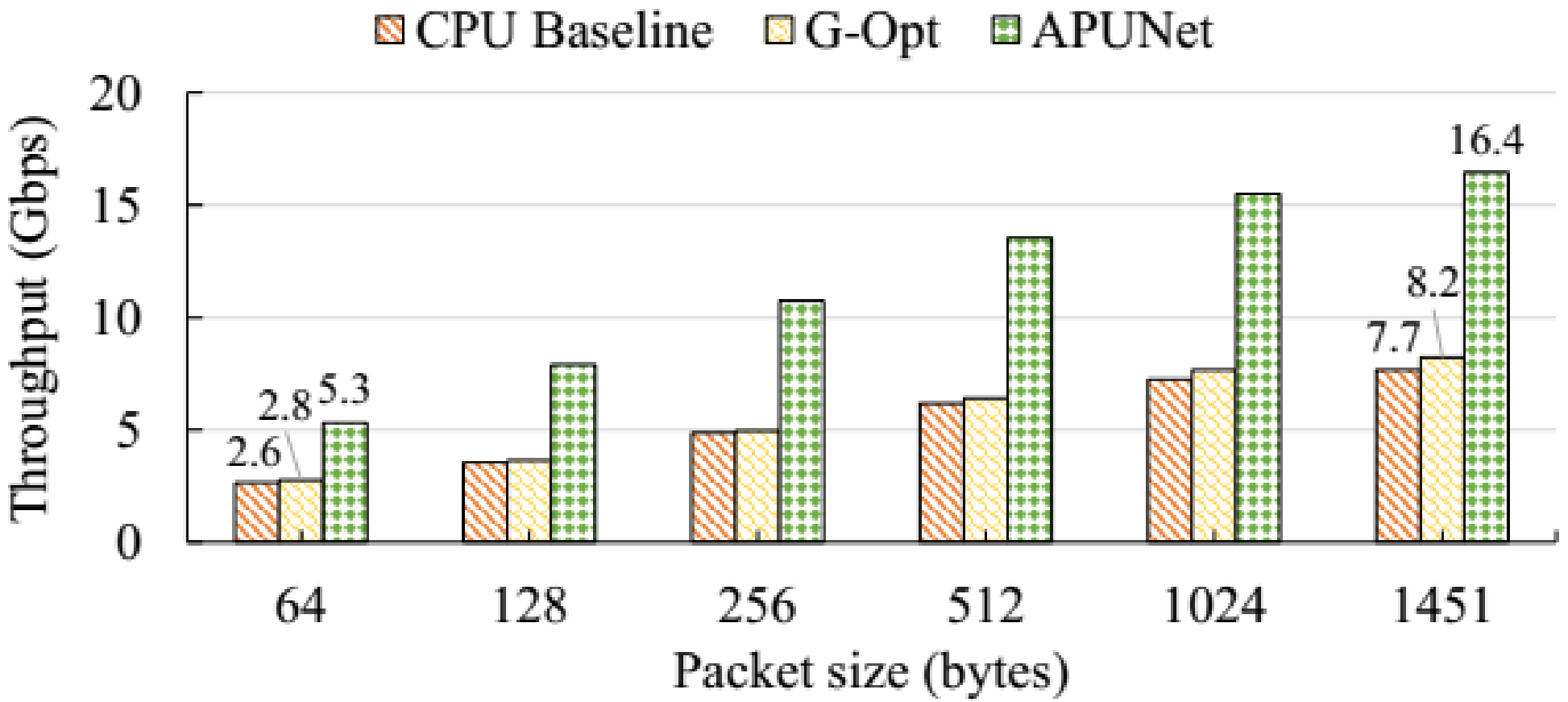
Latency


GPU-Copy
GPU-ZC
GPU-ZC-PERSIST
Improvement
Packet latency (us)
Improvement ratio
18
15
12
9
6
3
0
10
8
6
4
2
0
2.9
0.7
0.4
18.4
3.4
2.3
64
128
256
512
1024
1451
Packet size (bytes)

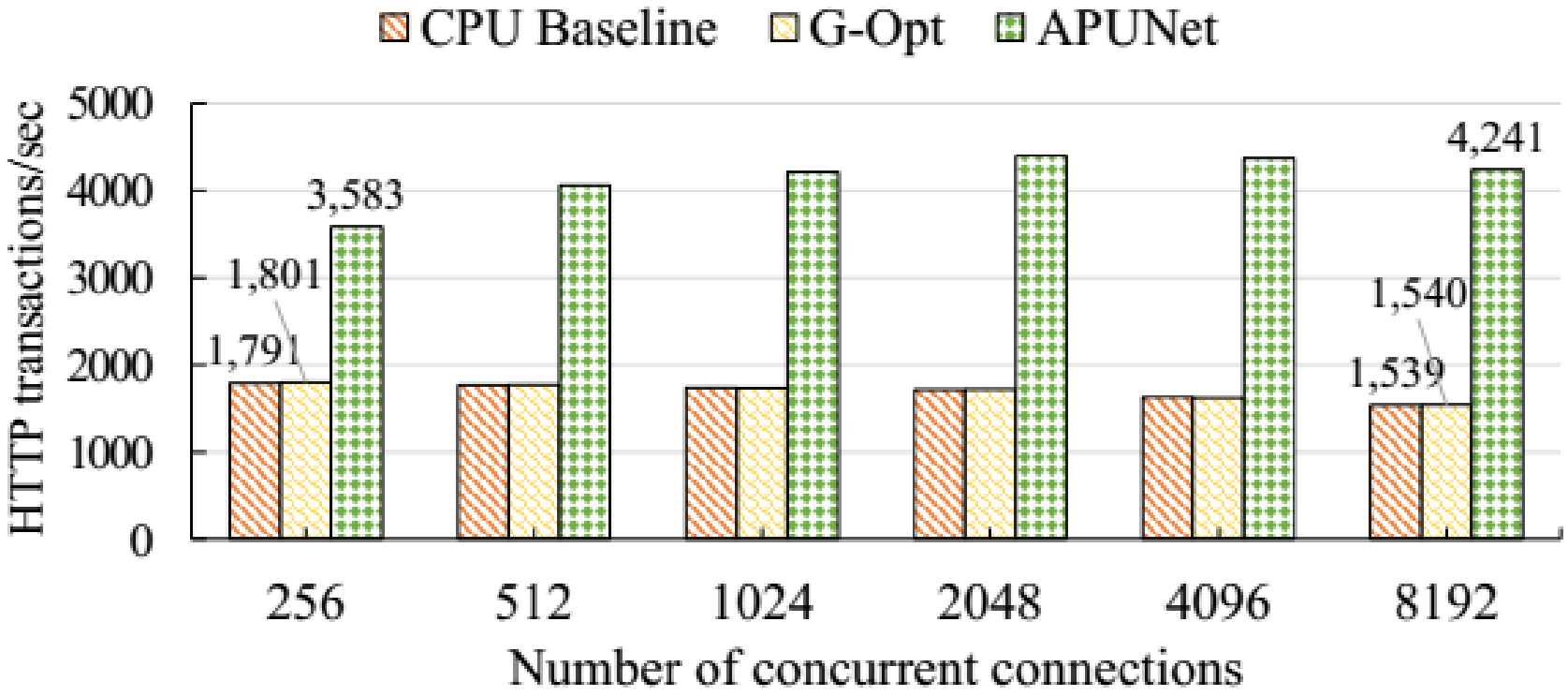
IPv6 forwarding



IPsec gateway



SSL reverse proxy



Lessons learned

- The benefits of GPU come from not only its parallel computation capability but also its memory access latency hiding feature.

- Learned the architecture of AMD GPU.

- Learned the warp scheduling inside the GPU SM.