APUNet: Revitalizing GPU as Packet Processing Accelerator

Younghwan Go, Muhammad Asim Jamshed, YoungGyoun Moon, Changho Hwang, and KyoungSoo Park

School of Electrical Engineering, KAIST
GPU-accelerated Networked Systems

• Execute same/similar operations on each packet in parallel
  • High parallelization power
  • Large memory bandwidth

• Improvements shown in number of research works
  • PacketShader [SIGCOMM’10], SSLShader [NSDI’11], Kargus [CCS’12], NBA [EuroSys’15], MIDEA [CCS’11], DoubleClick [APSys’12], …
Source of GPU Benefits

• GPU acceleration mainly comes from memory access latency hiding
  • Memory I/O → switch to other thread for continuous execution
Memory Access Hiding in CPU vs. GPU

• Re-order CPU code to mask memory access (G-Opt)*
  • Group prefetching, software pipelining

Questions:
Can CPU code optimization be generalized to all network applications?
Which processor is more beneficial in packet processing?

*Borrowed from G-Opt slides

*Raising the Bar for Using GPUs in Software Packet Processing [NSDI'15]
Anuj Kalia, Dong Zhu, Michael Kaminsky, and David G. Anderson
Contributions

• Demystify processor-level effectiveness on packet processing algorithms
  ➔ CPU optimization benefits light-weight memory-bound workloads
  ➠ CPU optimization often does not help large memory workloads
  ➔ GPU is more beneficial for compute-bound workloads
  ➠ GPU’s data transfer overhead is the main bottleneck, not its capacity

• Packet processing system with integrated GPU w/o DMA overhead
  • Addresses GPU kernel setup / data sync overhead, and memory contention
  • Up to 4x performance over CPU-only approaches!
Discrete GPU

• Peripheral device communicating with CPU via a PCIe lane

- High computation power
- High memory bandwidth
- Fast inst./data access
- Fast context switch

Require CPU-GPU DMA transfer!
Integrated GPU

• Place GPU into same die as CPU → share DRAM
  • AMD Accelerated Processing Unit (APU), Intel HD Graphics

- High computation power
- Fast inst./data access
- Fast context switch
- Low power & cost

No DMA transfer!
CPU vs. GPU: Cost Efficiency Analysis

• Performance-per-dollar on 8 popular packet processing algorithms
  • Memory- or compute-intensive
  • IPv4, IPv6, Aho-Corasick pattern match, ChaCha20, Poly1305, SHA-1, SHA-2, RSA

• Test platform
  • CPU-baseline, G-Opt (optimized CPU), dGPU w/ copy, dGPU w/o copy, iGPU

<table>
<thead>
<tr>
<th>CPU / Discrete GPU</th>
<th>APU / Integrated GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>AMD RX-421BD (4 @ 3.4 GHz)</td>
</tr>
<tr>
<td>GPU</td>
<td>AMD R7 Graphics (512 @ 800 MHz)</td>
</tr>
<tr>
<td>RAM</td>
<td>16 GB (DIMM DDR3 @ 2133 MHz)</td>
</tr>
<tr>
<td>Cost</td>
<td>CPU: $1143.9</td>
</tr>
</tbody>
</table>

CPU Intel Xeon E5-2650 v2 (8 @ 2.6 GHz)
GPU NVIDIA GTX980 (2048 @ 1.2 GHz)
RAM 64 GB (DIMM DDR3 @ 1333 MHz)
Cost CPU: $1143.9 dGPU: $840

GPU NVIDIA GTX980 (2048 @ 1.2 GHz)
RAM 64 GB (DIMM DDR3 @ 1333 MHz)
Cost CPU: $1143.9 dGPU: $840

CPU AMD RX-421BD (4 @ 3.4 GHz)
GPU AMD R7 Graphics (512 @ 800 MHz)
RAM 16 GB (DIMM DDR3 @ 2133 MHz)
Cost iGPU: $67.5
Cost Effectiveness of CPU-based Optimization

- G-Opt helps memory-intensive, but not compute-intensive algorithms
- Computation capacity as bottleneck with more computations

<table>
<thead>
<tr>
<th></th>
<th>IPv6 table lookup</th>
<th>AC pattern matching</th>
<th>SHA-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized perf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0 (w/ copy)</td>
</tr>
<tr>
<td>G-Opt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Detailed analysis on CPU-based optimization in the paper 😊
Cost Effectiveness of Discrete/Integrated GPUs

- Discrete GPU suffers from DMA transfer overhead
- Integrated GPU is most cost efficient!

Our approach:
Use integrated GPU to accelerate packet processing!
Contents

- Introduction and motivation
- Background on GPU
- CPU vs. GPU: cost efficiency analysis
- Research Challenges
  - APUNet design
  - Evaluation
- Conclusion
Research Challenges

• Frequent GPU kernel setup overhead
  • Overhead exposed w/o DMA transfer

• High data synchronization overhead
  • CPU-GPU cache coherency

• More contention on shared DRAM
  • Reduced effective memory bandwidth

APUNet: a high-performance APU-accelerated network packet processor
Persistent Thread Execution Architecture

- Persistently run GPU threads without kernel teardown
- Master passes packet pointer addresses to GPU threads
Data Synchronization Overhead

- Synchronization point for GPU threads: L2 cache
  - Require explicit synchronization to main memory

![Diagram showing shared virtual memory, master thread, CPU, GPU, and synchronization points.]

- Can process one request at a time!
- Need explicit sync!
Solution: Group Synchronization

- Implicitly synchronize group of packet memory GPU threads processed
- Exploit LRU cache replacement policy

For more details and tuning/optimizations, please refer to our paper 😊
Zero-copy Based Packet Processing

Option 1. Traditional method with discrete GPU

<table>
<thead>
<tr>
<th>NIC</th>
<th>CPU</th>
<th>COPY</th>
<th>GPU</th>
<th>Standard (e.g., mmap)</th>
<th>GDDR (e.g., cudaMalloc)</th>
</tr>
</thead>
</table>

High overhead!

Option 2. Zero-copy between CPU-GPU

<table>
<thead>
<tr>
<th>NIC</th>
<th>COPY</th>
<th>CPU</th>
<th>GPU</th>
<th>Standard (e.g., mmap)</th>
<th>Shared (e.g., clSVMAlloc)</th>
</tr>
</thead>
</table>

High overhead!

- Integrate memory allocation for NIC, CPU, GPU

<table>
<thead>
<tr>
<th>NIC</th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Shared (e.g., clSVMAlloc)

No copy overhead!
Evaluation

- How well does APUNet reduce latency and improve throughputs?
- How practical is APUNet in real-world network applications?

**APUNet (AMD Carrizo APU)**
- RX-421BD (4 cores @ 3.4 GHz)
- R7 Graphics (512 cores @ 800 MHz)
- 16GB DRAM

**40 Gbps NIC**
- Mellanox ConnectX-4

**Client (packet/flow generator)**
- Xeon E3-1285 v4 (8 cores @ 3.5 GHz)
- 32GB DRAM
Benefits of APUNet Design

- **Workload:** IPsec (128-bit AES-CBC + HMAC-SHA1)

### Packet Processing Latency

**Throughput (Gbps)**

<table>
<thead>
<tr>
<th>Atomics</th>
<th>Group sync</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.93</td>
<td>5.31</td>
</tr>
<tr>
<td>5.7x</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packet size (bytes)</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>1451</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet latency (us)</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

**Synchronization Throughput (64B Packet)**

- **5.4x**
- **1.5x**
Real-world Network Applications

- 5 real-world network applications
  - IPv4/IPv6 packet forwarding, IPsec gateway, SSL proxy, network IDS

**IPsec Gateway**

<table>
<thead>
<tr>
<th>Packet size (bytes)</th>
<th>Throughput (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>2.6 2.8</td>
</tr>
<tr>
<td>1451</td>
<td>5.3 7.7 8.2</td>
</tr>
</tbody>
</table>

**SSL Proxy**

<table>
<thead>
<tr>
<th>Number of concurrent connections</th>
<th>HTTP trans/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>1791 1801 3583</td>
</tr>
<tr>
<td>8192</td>
<td>1539 1540 4241</td>
</tr>
</tbody>
</table>

G-Opt vs CPU Baseline:
- 2x throughput improvement
- 2.75x higher HTTP trans/sec
Real-world Network Applications

• **Snort-based Network IDS**
  • Aho-Corasick pattern matching

• **No benefit from CPU optimization!**
  • Access many data structures
  • Eviction of already cached data

• **DFC* outperforms AC-APUNet**
  • CPU-based algorithm
  • Cache-friendly & reduces memory access

---

*DFC: Accelerating String Pattern Matching for Network Applications [NSDI’16]
Byungkwon Choi, Jongwook Chae, Muhammad Jamshed, KyoungSoo Park, and Dongsu Han
Conclusion

• Re-examine the efficacy of GPU-based packet processor
  • GPU is bottlenecked by PCIe data transfer overhead
  • Integrated GPU is the most cost effective processor

• APUNet: APU-accelerated networked system
  • Persistent thread execution: eliminate kernel setup overhead
  • Group synchronization: minimize data synchronization overhead
  • Zero-copy packet processing: reduce memory contention
  • Up to 4x performance improvement over CPU baseline & G-Opt

APUNet
High-performance, cost-effective platform for real-world network applications
Thank you.

Q & A