GASPP: A GPU-Accelerated Stateful Packet Processing Framework

Giorgos Vasiliadis, FORTH-ICS, Greece
Lazaros Koromilas, FORTH-ICS, Greece
Michalis Polychronakis, Columbia University, USA
Sotiris Ioannidis, FORTH-ICS, Greece
Network Packet Processing

• Computationally and memory-intensive

• High levels of data parallelism
  – Each packet can be processed in parallel

• Poor temporal locality for data
  – Typically, each packet is processed only once
GPU = Graphics Processing Units

- Highly parallel manycore devices
- Hundreds of cores
- High memory bandwidth
- Up to 6GB of memory
GPUs for Network Packet Processing

- Gnort [RAID’08]
- PacketShader [SIGCOMM’10]
- SSLShader [NSDI’11]
- MIDEA [CCS’11], Kargus [CCS’12]
- ...
GPUs for Network Packet Processing

- Gnort [RAID'08]
- PacketShader [SIGCOMM'10]
- SSLShader [NSDI'11]
- MIDeA [CCS'11], Kargus [CCS'12]
- ...

**Independent/Monolithic Designs**

1. A lot of CPU-side code even for simple apps
2. Explicit batching
3. Explicit data copies and PCIe transfers
Need a **framework** for developing **GPU accelerated packet processing** applications
GASPP Framework

TCP Flow State Management

AES

Regex Match

String Match

Firewall

Packet Reordering

Packet Scheduling

Packet Decoding

e.g., xOMB (UCSD)

Contribution of reusable modules: 30–80%
GASPP Framework

• Fast user-space packet capturing

• Modular and flexible

• Efficient packet scheduling mechanisms

• TCP processing and flow management support
GASPP Framework

• Fast user-space packet capturing

• Modular and flexible

• Efficient packet scheduling mechanisms

• TCP processing and flow management support
Fast user-space packet capturing

**Use a single user-space buffer between the NIC and the GPU**

**Stage packets back-to-back to a separate buffer**
Fast user-space packet capturing

<table>
<thead>
<tr>
<th>Packets size (#bytes)</th>
<th>Gbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>2.06</td>
</tr>
<tr>
<td>128</td>
<td>4.03</td>
</tr>
<tr>
<td>256</td>
<td>8.07</td>
</tr>
<tr>
<td>512</td>
<td>16.13</td>
</tr>
<tr>
<td>1024</td>
<td>32.26</td>
</tr>
<tr>
<td>1518</td>
<td>47.83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packets size (#bytes)</th>
<th>Gbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>13.76</td>
</tr>
<tr>
<td>128</td>
<td>18.21</td>
</tr>
<tr>
<td>256</td>
<td>20.53</td>
</tr>
<tr>
<td>512</td>
<td>19.21</td>
</tr>
<tr>
<td>1024</td>
<td>19.24</td>
</tr>
<tr>
<td>1518</td>
<td>20.04</td>
</tr>
</tbody>
</table>
Fast user-space packet capturing

<table>
<thead>
<tr>
<th>Packets size (#bytes)</th>
<th>Gbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>2.06</td>
</tr>
<tr>
<td>128</td>
<td>4.03</td>
</tr>
<tr>
<td>256</td>
<td>8.07</td>
</tr>
<tr>
<td>512</td>
<td>16.13</td>
</tr>
<tr>
<td>1024</td>
<td>32.26</td>
</tr>
<tr>
<td>1518</td>
<td>47.83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packets size (#bytes)</th>
<th>Gbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>13.76</td>
</tr>
<tr>
<td>128</td>
<td>18.21</td>
</tr>
<tr>
<td>256</td>
<td>20.53</td>
</tr>
<tr>
<td>512</td>
<td>19.21</td>
</tr>
<tr>
<td>1024</td>
<td>19.24</td>
</tr>
<tr>
<td>1518</td>
<td>20.04</td>
</tr>
</tbody>
</table>
Why staging is better than zero-copy
(for small packets)

Better space utilization => No redundant transfers
Selective scheme

- Packets are are *copied back-to-back* to a separate buffer, if the buffer occupancy is sparse
- *Otherwise*, they are transferred *directly* to the GPU
GASPP Framework

- Fast user-space packet capturing
- Modular and flexible
- Efficient packet scheduling mechanisms
- TCP processing and flow management support
Modular and Flexible

• Basic abstraction of processing: ‘`modules’’
  \[ \text{processPacket(packet)} \{ \ldots \} \]

• Modules are executed sequentially or in parallel

Module 1: **IP-learn**

Module 2: **Content Inspection**

Module 3: **Encryption**
Batch Processing Pipeline

RX → Module1 → Module2 → … → TX

time

[Diagram showing a workflow with RX, Module1, Module2, ..., TX connected by arrows, indicating the flow of data or process.]
Batch Processing Pipeline

RX Module1 Module2 ••• TX

RX batch
Batch Processing Pipeline

RX ➔ Module1 ➔ Module2 ➔ ... ➔ TX

RX batch ➔ copy to GPU ➔ Batch processing
Batch Processing Pipeline

RX batch → Module1 → Module2 → TX

copy to GPU

Batch processing
Batch Processing Pipeline

RX batch

Module1

Module2

TX batch

RX

Module

TX

Batch processing

copy to GPU

copy to CPU
GASPP Framework

- Fast user-space packet capturing
- Modular and flexible
- Efficient packet scheduling mechanisms
- TCP processing and flow management support
Single Instruction, Multiple Threads

- Threads within the same **warp** have to execute the same instructions

- **Great for regular computations!**
Parallelism in packet processing

• Network packets are processed in batches
  – More packets => more parallelism
Dynamic Irregularities

- Received network packets mix is very dynamic
• Received network packets mix is very dynamic
  – *Different packet lengths*
• Received network packets mix is very dynamic
  – *Different packet lengths*
  – *Divergent parallel module processing*
Dynamic Irregularities

![Bar chart showing dynamic irregularities across different modules and warps over time.](chart.png)
Dynamic Irregularities

![Graph showing dynamic irregularities with modules and warps]
Dynamic Irregularities

module 1 | module 2 | module 3

warp 1 | warp 2 | warp 3 | warp 4 | warp 5 | warp 6 | warp 7

time
Dynamic Irregularities
Dynamic Irregularities

Low warp occupancy
Packet grouping

Batch Size

Batch Size
Packet grouping

- Harmonized execution
- Symmetric processing
GASPP Framework

- Fast user-space packet capturing
- Modular and flexible
- Efficient packet scheduling mechanisms
- TCP processing and flow management support
TCP Flow State Management

- Maintain the state of TCP connections
TCP Stream Reassembly

Batch Size

connection1  connection2  connection3
TCP Stream Reassembly

Batch Size

- connection1
- connection2
- connection3

Sequential processing
TCP Stream Reassembly

Batch Size

Sequential processing
TCP Stream Reassembly

Batch Size

connection1  connection2  connection3

Sequential processing

Packet-level parallel processing
• Key insight
  – Packets $<A, B>$ are consecutive if $Seq_B = (Seq_A + len_A)$
TCP Stream Reassembly

Batch Size

H(Seq)  H(seq+len)

A  A  B  C  ...
TCP Stream Reassembly

Batch Size

H(Seq) H(seq+len)
TCP Stream Reassembly

Batch Size

A

B

C

H(Seq)

H(seq+len)
TCP Stream Reassembly

Batch Size

Parallel Processing
Other TCP corner cases

• TCP sequence holes

• Out-of-order packets
Other TCP corner cases

- TCP sequence holes
- Out-of-order packets

Read the paper for details
Evaluation

• Forwarding
• Latency
• Individual Applications
• Consolidated applications
**Evaluation Setup**

- GASPP machine has:
  - 2x NUMA nodes (Intel Xeon E5520 2.27GHz quad-core CPUs)
  - 2x banks of 6GB of DDR3 1066MHz RAM
  - 2x Intel 82599EB network adapters (with dual 10GbE ports)
  - 2x NVIDIA GTX480 graphics cards
Basic Forwarding

![Graph showing throughput (Gbit/s) vs packet size (bytes). The graph compares the performance of different packet sizes for 64-byte packets (solid line), GASPP transfers with all PCIe data transfers included (grey bars), GASPP transfers without PCIe data transfers included (blue bars), and without any GPU transfers (purple line). The x-axis represents packet sizes from 64 to 1518 bytes, and the y-axis represents throughput in Gbit/s. The graph shows that GASPP achieves higher throughput compared to CPU (8x cores) and without any GPU transfers.]
Latency

- **8192 batch:**
  - CPU: 0.48 us
  - GASPP: 3.87 ms

- **1024 batch:** 0.49 ms
  - Same performance for basic forwarding
  - ...but 2x-4x throughput slowdown for heavyweight processing applications
Individual Applications

- Each application is written as a GPU kernel
  - No CPU-side development

- Speedup over a single CPU-core

<table>
<thead>
<tr>
<th>Applications</th>
<th>GASPP (8192 batch)</th>
<th>GASPP (1024 batch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall</td>
<td>3.6x</td>
<td>3.6x</td>
</tr>
<tr>
<td>StringMatch</td>
<td>28.4x</td>
<td>9.3x</td>
</tr>
<tr>
<td>RegExMatch</td>
<td>173.1x</td>
<td>36.9x</td>
</tr>
<tr>
<td>AES</td>
<td>14.6x</td>
<td>6.5x</td>
</tr>
</tbody>
</table>
Consolidating Applications

GASPP: - 1.19x 2.12x 1.93x

GASPP reduces irregular execution by 1.19-2.12X
Conclusions

• What we offer:
  – Fast inter-device data transfers
  – GPU-based flow state management and stream reconstruction
  – Efficient packet scheduling mechanisms

• Limitations
  – High packet processing latency
GASPP: A GPU-Accelerated Stateful Packet Processing Framework

Giorgos Vasiliadis, FORTH-ICS, Greece
Lazaros Koromilas, FORTH-ICS, Greece
Michalis Polychronakis, Columbia University, USA
Sotiris Ioannidis, FORTH-ICS, Greece