PacketShader: A GPU-Accelerated Software Router

Some images and sentence are from original author Sangjin Han’s presentation.

Presenter: Hao Lu
Why? What? How?

• Why used software routers?
• What is GPU?
• Why use GPU?
• How to use GPU?
• What is PacketShader’s design?
• How is the performance?
• If have time, configuration of the system.
Software Router

• Not limited to IP routing
  • You can implement whatever you want on it.

• Driven by software
  • Flexible

• Based on commodity hardware
  • Cheap
What is GPU?

- Graph process units.

15 Streaming Multiprocessors consist 32 processors = 480 cores
Why use GPU?

Benefit:

• Higher computation power
  - 1-8 v.s. 480

• Memory access latency
  - Multi-thread to hide the latency
  - CPU has miss register (up to 6)

• Memory bandwidth
  - 32GB v.s. 177GB

Down Sides:

• Thread start latency
• Data transfer rate
How to use GPU?

- GPU is used for highly parallelizable tasks.
- With enough threads to hide the memory access latency

1. Batching
2. Parallel Processing in GPU
PacketShader Overview

- Three stages in a streamline
  - Pre-shader
    - Fetching packets from RX queues.
  - Shader
    - Using the GPU to do what it need to be done
  - Post-shader
    - Gather the result and scatter to each TX queue
IPv4 Forwarding Example

1. IP addresses
2. Forwarding table lookup
3. Next hops

- Checksum, TTL
- Format check
- ...

Some packets go to slow-path

Update packets and transmit
Scaling with Multi-Core CPU

- Problems:
  - GPU are not as efficient if more than one CPU access it.
Another view

Figure 7: PacketShader software architecture
Optimization

- Chuck Pipelining:

- Gather/Scatter

- Concurrent Copy and Execution
Performance: hardware

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
<th>Qty</th>
<th>Unit price</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Xeon X5550 (4 cores, 2.66 GHz)</td>
<td>2</td>
<td>$925</td>
</tr>
<tr>
<td>RAM</td>
<td>DDR3 ECC 2 GB (1,333 MHz)</td>
<td>6</td>
<td>$64</td>
</tr>
<tr>
<td>M/B</td>
<td>Super Micro X8DAH+F</td>
<td>1</td>
<td>$483</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA GTX480 (480 cores, 1.4 GHz, 1.5 GB)</td>
<td>2</td>
<td>$500</td>
</tr>
<tr>
<td>NIC</td>
<td>Intel X520-DA2 (dual-port 10GbE)</td>
<td>4</td>
<td>$628</td>
</tr>
</tbody>
</table>

Table 2: Test system hardware specification (total $7,000)

Figure 3: Block diagram of our server
Performance: IPv4 Forwarding

- Algorithm: DIR-24-8-BASIC
  - It requires one memory access per packet for most cases, by storing next-hop entries for every possible 24-bit prefix.

- Pre-shade:
  - Require slow path => Linux TCP/IP stack
  - Else, Update TTL and checksum.

![Graph showing throughput vs packet size]
Performance: IPv6 Forwarding

- Same idea of IPv4, more memory access
Performance: OpenFlow

- OpenFlow is a framework that runs experimental protocols over existing networks. Packets are processed on a flow basis.
- The OpenFlow switch is responsible for packet forwarding driven by flow tables.
Performance: IPsec

- IPsec is widely used to secure VPN tunnels or for secure communication between two end hosts.
- Cryptographic operations used in IPsec are highly compute-intensive

![Graph showing throughput and speedup for different packet sizes]
Configuration of the System

• Problem:
  1. Linux Network Stack Inefficiency.
  2. NUMA (None uniform memory access)
  3. Dual-IOH Problem

• Solutions:
  1. Better Driver, use Huge Packet Buffer
  2. NUMA aware driver
  3. In research
Network Stack Inefficiency

1. Frequent allocation/deallocation memory
2. skb too large (208 bytes)
NUMA

• None Uniform Memory Access due to RSS.

• Solution: Reconfigure RSS to configure RSS to distribute packets only to those CPU cores in the same node as the NICs.
Dual-IOH Problem

• Asymmetry on Data transfer rate.

• Cause: Unknown!!