Building a Single-Box 100Gbps Software Router

The 17th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)
May 5th, 2010
Long Branch, New Jersey

Sangjin Han, Keon Jang, KyoungSoo Park, Sue Moon
Software Routers

• Runs on commodity-off-the-shelf (COTS) servers (mostly x86-based)

• Software is usually open-sourced, but there are many commercial ones as well.

• Control plane
  – Many options: Zebra, Quagga, XORP, ...

• Data plane
  – TCP/IP stack of general OS (Linux, FreeBSD) or
  – Dedicated SW (e.g. Click)
# Traditional Routers vs. SW Routers

<table>
<thead>
<tr>
<th></th>
<th>Traditional routers</th>
<th>Software routers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Price</strong></td>
<td>$10 ~ $1M (CRS-1 40Gbps)</td>
<td>$500 ~ $5,000</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Wide range (100Mbps ~ multi-tera)</td>
<td>1 ~ 5Gbps</td>
</tr>
<tr>
<td><strong>HW</strong></td>
<td>Proprietary</td>
<td>Off-the-shelf</td>
</tr>
<tr>
<td><strong>SW</strong></td>
<td>Proprietary</td>
<td>Third party (many opensource)</td>
</tr>
<tr>
<td><strong>Specialized ASIC</strong></td>
<td>Yes (only for high-end)</td>
<td>No</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>Proven (?)</td>
<td>Doubtful</td>
</tr>
<tr>
<td><strong># of developers</strong></td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td><strong># of engineers</strong></td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td><strong>Upgrade</strong></td>
<td>$$$ or often impossible</td>
<td>Replace with newer parts</td>
</tr>
<tr>
<td><strong>Evolution cycle</strong></td>
<td>Long</td>
<td>Short</td>
</tr>
<tr>
<td><strong>When troubled, any room for excuses? 😊</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Concerns over Software Routers

- **Performance**
  - High per-packet cost
    - low throughput for min-sized packets
  - Over 40% of packets are min-sized.
    - TCP ACK and other control packets
  - Severe performance degradation for additional functions
    - E.g. SSL decryption offload is 10+ times more expensive than plain TCP forwarding

- **High availability**
  - Critical for enterprise market

- **Others**: form factor, port density, ease of deployment, ...
What Makes SW Routers Slow?

1. RX: DMA to RAM
2. CPU read
3. Packet processing
4. CPU write-back
5. TX: DMA to NIC

**Intel Core (old)**

- Memory is attached to the Northbridge
- e.g. Intel Core 2 quad

**Intel Nehalem (new)**

- Memory controller is integrated in CPU
- e.g. Intel Core i7

---

**Diagram:**

- **CPU**
- **NB**
- **RAM**
- **NIC**

**Intel Core (old):**

1. RX: DMA to RAM
2. CPU read
3. Packet processing (CPU)
4. CPU write-back
5. TX: DMA to NIC

Memory bus was the bottleneck

**Intel Nehalem (new):**

1. RX: DMA to NIC
2. CPU reads
3. Packet processing (CPU)
4. CPU write-back
5. TX: DMA to NIC

Now CPU becomes the new bottleneck

Thanks to DDR3 and triple-channel architecture, memory bus is not a bottleneck any longer.
Distributed PC-based Router

Khan, Birke, Manjunath, Sahoo, Bianco, HPSR 2008
8.33 Gbps or 6.35 Gbps w/o Ethernet O/H
8.33 x 4 / 2 = 15.77 Gbps on 4-PC config
PacketShader

- Integrated memory controller and dual IOHs
- Aggregate 80Gbps → the system must be highly efficient
- 8 CPU cores, 8 10G ports, 2 NUMA nodes, 2 GPUs, 2 IOH...
  - Scalability is the key
- 28.8 Gbps for 64B packets
How to Deliver 100 Gbps

• CPU cycles
• I/O Capacity
• Memory Bandwidth
Inefficiencies of Linux Network Stack

<table>
<thead>
<tr>
<th>Functional bins</th>
<th>% of cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>skb (de)allocation</td>
<td>8.0%</td>
</tr>
<tr>
<td>skb initialization</td>
<td>4.9%</td>
</tr>
<tr>
<td>NIC device driver</td>
<td>13.3%</td>
</tr>
<tr>
<td>Compulsory cache misses</td>
<td>13.8%</td>
</tr>
<tr>
<td>Memory subsystem</td>
<td>50.2%</td>
</tr>
<tr>
<td>Others</td>
<td>9.8%</td>
</tr>
<tr>
<td>Total</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

CPU cycle breakdown in packet RX

- Compact metadata
- Batch processing
- Software prefetch

Huge packet buffer
CPU Cycle Breakdown

• RouteBricks

  1,229 CPU cycles per NIC-to-NIC packet forwarding
  1,850 CPU cycles per NIC-to-NIC packet routing

  64B packets at 100 Gbps = 149 Mpps

  ⇒ 1,229 x 149 = 183 GHz

  ⇒ 1,859 x 149 = 277 GHz

• Intel X7560 CPU = 8 x 2.26 GHz on 4 sockets

  8 x 2.26 x 4 = 72.3 GHz
Optimization in PacketShader

1. Remove dynamic per-packet buffer allocation and use static buffers
2. Perform prefetch over descriptors and packet data to mitigate compulsory cache misses
3. Minimize cache bouncing and eliminate false sharing between CPU cores

⇒ Factor of 6 reduction in CPU cycles
⇒ 200 CPU cycles per packet
Optimization #1

RX queue

Packet data buffer

skb

Buffer for packet data

Buffer for metadata
Optimization #2

Without batching: 1.6 Gbps for RX, 2.1 Gbps for TX, 0.8 Gbps for forwarding

⇒ batching is essential!
I/O Capacity (I)

• PCI Express
  – 10 GbE NIC has PCIe x8
  – PCIe 2.0 = 2.5 GHz per lane => 20 Gbps / 8 lanes
  – Effective B/W = 12.3 Gbps per NIC

• PCIe 2.0 upgrade to 5 GHz
  – Effective B/w up over 20 Gbps

• How many PCIe 2.0 x8 slots?
I/O Capacity (II)

• QuickPath Interconnect (QPI)
  – CPU socket-to-socket link for remote memory
  – IOH-to-IOH link for I/O traffic
  – CPU-to-IOH for CPU to peripheral connections

• Today’s QPI link
  – 12.8 GB/s or 102.4 Gbps
Memory Bandwidth

• For 100Gbps forwarding we need 400 Gbps in memory bandwidth + bookkeeping

• Current configuration
  – triple-channel DDR3 1,333 MHz
  – 32 GB/s per core (theoretical) and 17.9GB/s (empirical)

• On NUMA system
  – More nodes
  – Careful placement...
Summary

• Two major bottlenecks
  – CPU cycles
  – I/O bandwidth

• Message
  – Find source of extra CPU cycles
  – Expect improvement in IOH chipsets and multi-IOH configuration