Towards High Performance Software Virtual Routers on x86 Commodity Hardware

CoNEXT 2008

Mickaël Hoerdt – Lancaster University/UK
m.hoerdt@comp.lancs.ac.uk

In collaboration with
N. Egi, L. Mathy
A. Greenhalgh, M. Handlev
F. Huici (NEC Europe)
Introduction

- Traditionally, high performance networking devices:
  - Built as custom hardware
  - Made of various specialized “CPUs”
  - Specialized multi-cores

- Software routers several orders of magnitude slower

- BUT recent advances in commodity HW architecture
  - Multi-core (with very high clock frequency)
  - Buses (usual bottlenecks) are disappearing
The question

- On Commodity Hardware :
  - How viable are software routers ?
  - How viable are virtual routers ?
Outline

- Part I: Performance of software routers
- Part II: Viability of virtual routers
Performance of software routers

System under test

- **Dell PowerEdge 2950 [SMP architecture]**
  - 2 Intel Xeon X5355 CPUs (quad-core @2.66GHz)
    - 32K L1d, 32K L1i, 4M L2 shared d&i cache (shared by 2 cores)
  - 8*1GB system memory
    - PC-5300
  - 12 Gigabit Ethernet ports
    - On 3 NICs
    - Each NIC had 4 PCIe lanes
    - Intel 82571EB controllers
  - Software
    - Click in Linux kernel
    - Xen VMM/Openvz
    - Oprofile, etc.
Performance of software routers

Software router raw aggregated performance: Single-core

- 1.5 Mpps, 1024 bytes = 12 Gb/s = Line rate

Uniprocessor one way and two way forwarding performances – 1024 bytes packets
Performance of software routers

Software router **raw** aggregated performance : **Single-core**

- Uniprocessor one way and two way forwarding performances – 64 bytes packets
- 2.4 Mpps, 64 bytes = CPU limitation
  - No improvement in aggregate rate when more flows are added
  - Probably a CPU limitation.
- 2.4 Mpps would mean up to 20 gigabit interfaces to saturate a core with 1024 bytes packets
Performance of software routers

Software router raw aggregated performance: Multi-core

- Scales linearly for first 4 flows
  - Confirm that single core was indeed CPU-limited

- Hard limit at 7.1 Mpps, 64 bytes
  - Adding more flows (and more cores) does not increase the limit
  - Can’t be hard bound on throughput limited at 3.64 Gb/s but did 12 Gb/s with large packets

Multiprocessor one way way forwarding performances – 64 bytes packets
Performance of software routers

Packet life in memory
So where is the bottleneck?

- Packets are being dropped from the input NIC queue
  - CPU does not allocate DMA buffers fast enough?
  - NIC cannot use the allocated buffers fast enough?
- Checking the number of packets polled in at a time shows that it is the second case (many empty polls)
For each data transfer:
NIC must:
1- Obtain packet buffer address in Memory
2- DMA the packet From/To Memory
3- Update packet descriptor in Memory when finished
6 memory accesses by the NIC when forwarding.

Measure of the number of CPU memory transaction per packets

So in total 10 memory accesses per packets with CPU and NICs

About 4
Memory specs:
- Frequency: 3ns/cycles (333 MHz)
- Throughput: 16 bytes per half clock cycles
- CAS: 5 clock cycles
- RAS: 5 clock cycles

If memory accesses are contiguous the throughput is high.
- Because CAS and RAS don’t have to be repaid.

Assuming packets are not accessed contiguously in memory:
- Then each memory access has to be CAS asserted (best case)
- Then each packets cost 5x10=50 memory cycles
- Which is 50*3=150 ns per packet or 6.7 Mpps
- We measured 7.1 Mpps

Conclusion: The bottleneck is certainly memory latency
Performance of software routers

Software router IPv4 aggregated performances - Multicore

- 5 Extra memory accesses
  - Read/Write the TTL: 2
  - Rewrite MAC address: 1
  - Checksum computation: 2
  - 15 memory accesses per packets
  - 15*5 = 75 memory cycles
  - 225 ns per packet = 4.4 Mpps

- 1024 bytes packets
  - Line rate.

Multiprocessor two ways
IPv4 forwarding performances – 64 bytes packets
With appropriate FP-to-core mapping,
- 7 Mpps raw forwarding, 4.9 Mpps IPv4 forwarding
- This is more than enough for the edge
- Faster than many commercial solutions

Lesson learned:

- The bottleneck is memory access latency
- Plenty of spare CPU cycles
  - Surplus scales with number of cores
Part II

- What about running several (virtual) routers on a single box?
Virtual Routers

Challenges

- Isolating VRs from each other
- Low overhead of the virtualization
- Fair scheduling despite the scarce resource being the memory rather than CPU cycles
- Mapping Cores to VRs in a performant manner
# Viability of virtual routers?

## Two techniques to virtualize a router

1. **Container based**
   - User space
   - Control Plane
   - Kernel space
   - Forwarding Plane

2. **Hypervisor based**
   - User space
   - Control Plane
   - Kernel space
   - Forwarding Plane
   - Control-forwarding interface
   - Memory domain

<table>
<thead>
<tr>
<th></th>
<th>1. Container based</th>
<th>2. Hypervisor based</th>
</tr>
</thead>
<tbody>
<tr>
<td>User space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control Plane</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kernel space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forwarding Plane</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory domain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network interface</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Control-forwarding interface
- Hypervisor
Virtualization techniques **kernel** forwarding overhead (minimum sized packets)

- Container based (OpenVz+Click implementation): 7.1 Mpps
- Hypervisor based (Xen+Click implementation): 6.8 Mpps, 4% overhead.
Sharing NICs between virtual routers: software classification

- Once polled, i.e. copied in memory, a packet has passed the bottleneck
- Scheduling dictated by arrival rate
- We could re-queue and do weighted fair queueing, but...
- Dropping excess packets is very wasteful as main cost is polling in the first place
  - Dropping low priority packets is NOT going to improve aggregate forwarding performance much

What we need is hardware classification...

Viability of virtual routers?
Sharing NICs: hardware classification

- Fairness not perfect
  - saturation point (partly) depends on traffic distribution
- Close enough for most purposes

Viability of virtual routers?
Conclusions

- Memory latency is bottleneck in SW (VR) routers
  - Scales in terms of number of memory controllers, not cores!

- High performance SW VR
  - Is forwarding engine virtualization
  - Needs multi-queue NICs
  - Hero numbers: Currently fastest VR on commodity HW in the world

- Commodity HW + familiar OS = better scope for programmability
  - Deployment of new protocols
Future work

- Develop new kernel mechanisms to break our 7 Mpps “hero number”
  - Great scope with NUMA

- Fair Hierarchical scheduling

- Platform coming “soon” in open source
  - URL: http://nrg.cs.ucl.ac.uk/vrouter/