Pushing the Limits of Kernel Networking

Networking Services Team, Red Hat
Alexander Duyck
August 19th, 2015
Agenda

- Identifying the Limits
  - Memory Locality Effect
  - Death by Interrupts
  - Flow Control and Buffer Bloat
  - DMA Delay
- Performance
  - Synchronization Slow Down
  - The Cost of MMIO
  - Memory Alignment, Memcpy, and Memset
  - How the FIB Can Hurt Performance
- What more can be done?
Identifying the Limits

- With 60B frames achieving line rate is difficult
  - Only 24B of additional overhead per frame
  - $10\text{Gb/s} / 125\text{MB/Gb} / 84\text{Bpp} = 14.88\text{Mpps, 67.2nspp}$
- L3 cache latency on Ivy Bridge is about 30 cycles
  - Each nanosecond an E5-2690 will process 2.6 cycles
  - $30 \text{cycles} / 2.6 \text{cycles/ns} = 12\text{ns}$
- To achieve line rate at 10G we need to do two things
  - Lower processing time
  - Improve scalability
Memory Locality Effect

- NUMA – Non-uniform memory access
Memory Locality Effect

- **DDIO - Data Direct I/O**
  - Xeon E5 26XX Feature
  - Local socket only
  - No need for memory access

- **XPS – Transmit Packet Steering**
  - Transmit packets on local CPU

```bash
echo 01 > /sys/class/net/enp5s0f0/queues/tx-0/xps_cpus
echo 02 > /sys/class/net/enp5s0f0/queues/tx-1/xps_cpus
echo 04 > /sys/class/net/enp5s0f0/queues/tx-2/xps_cpus
echo 08 > /sys/class/net/enp5s0f0/queues/tx-3/xps_cpus
```
Death by Interrupts

- Interrupts can change location based on irqbalance
- Too low of an interrupt rate
  - Overrun ring buffers on device
  - Add unnecessary latency
  - Overrun socket memory if NAPI shares CPU
- Too high of an interrupt rate
  - Frequent context switches
  - Frequent wake-ups
- Interrupt moderation schemes often tuned for benchmarks instead of real workloads
Flow Control and Buffer Bloat

- Flow control can significantly harm performance
  - Adds additional buffering, adding extra latency
  - Creates head-of-line blocking which limits throughput
    - Faster queues drop packets waiting on slowest CPU
- Some NICs implement per-queue drop when disabled
- Disabling it requires just one line in ethtool
  
  ```bash
  ethtool -A enp5s0f0 tx off rx off autoneg off
  ```
DMA Delay

- IOMMU can add security but at significant overhead
  - Resource allocation/free requires lock
  - Hardware access required to add/remove resources
- If you don't need it you can turn it off
  \[
  \text{intel_iommu=off}
  \]
- If you need it for virtualization (KVM/XEN)
  \[
  \text{iommu=pt}
  \]
- Some drivers include mitigation strategies
  - Page reuse
Performance Data Ahead!!!

- Single socket Xeon E5-2690v3
- Dual port 82599ES
  - Assigned addresses 192.168.100.64 & 192.168.101.64
  - Disabled flow control
  - Pinned IRQs 1:1
  - Used ntuple filter to force flows to specific queues
- CPU C states disabled via cpu /dev/cpu_dma_latency
- Traffic generator sent IP data w/ RR source address
  - Each frame sent 4 times before moving to next address
- Your Experience May Vary
Routing Performance

![Graph showing routing performance for RHEL 7.1 with threads on the x-axis and packets per second on the y-axis. The graph illustrates the increase in packets processed per second as the number of threads increases. The performance peaks around 8 threads and then slightly decreases with 12 threads.]
Synchronization Slow Down

- Synchronization primitives come at a heavy cost
  - `local_irq_save/resore` costs 10s of ns
    - Not needed when all requests are in same context
  - `rmb/wmb` flush pipelines which adds delay
    - Needed for some architectures but not others
- Updated kernel to remove unnecessary bits in 3.19
  - NAPI allocator for page fragments and skb
  - `dma_rmb/wmb` for DMA memory ordering
The Cost of MMIO

- MMIO write to notify device can cost hundreds of ns
- Latency shows up as either Qdisc lock, or Tx queue unlock overhead
- xmit_more was added to 3.18 kernel to address this
  - Reduces MMIO writes to device
  - Reduces locking overhead per packet
  - Reduces interrupt rates as packets are coalesced
  - Allows for 10Gbps line rate 60B packets w/ pktgen
Memory Alignment, Memcpy, and Memset

- Partial cache-line writes come at a cost
  - Most architectures now start with `NET_IP_ALIGN = 0`
  - On x86 partial writes trigger a read, modify, write cycle
- String ops change implementation based on CPU flags
  - erms and rep_good can have impact on performance
  - KVM doesn't copy CPU flags by default
- tx-nocache-copy
  - Enabled use of movntq for user to kernel space copy
  - Enabled by default for kernels 3.0 – 3.13
  - Prevents use of features such as DDIO

```bash
ethtool -K enp5s0f0 tx-nocache-copy off
```
How the FIB Can Hurt Performance

- Starting w/ version 4.0 of kernel fib_trie was rewritten
  - FIB statistics were made per CPU and not global
  - Penalty for trie depth significantly reduced
  - Kernel 4.1 merged local and main trie for further gains

- Recommendations for kernels prior to 4.0
  - Disable CONFIG_IP_FIB_TRIE_STATS in kernel config
  - Avoid assigning addresses such as 192.168.122.1
    - IPs in the range 192.168.122.64 – 191 can reduce depth by 1
  - Use class A reserved addresses to reduce trie walk
    - 10.x.x.x likely will contain fewer bits than 192.168.x.x
What More Can be Done?

- SLAB/SLUB bulk allocation
  - [https://lwn.net/Articles/648211/](https://lwn.net/Articles/648211/)
- Tuning interrupt moderation to work in more cases
  - Pktgen with 60B packets
- Explore optimizing users for memset/memcpy()
  - build_skb()
- Find a way to better use xmit_more on small packets
- Explore shortening Tx/Rx queue lengths
Routing Performance

The graph shows the routing performance in packets per second as a function of the number of threads. The performance is measured for three versions of RHEL:

- RHEL 7.1
- RHEL 7.2 Alpha
- Tweaked 7.2 Alpha

As the number of threads increases, the performance of all three versions improves. However, the Tweaked 7.2 Alpha version consistently outperforms the other two, reaching a maximum of approximately 14,000,000 packets per second at 12 threads, compared to 10,000,000 for RHEL 7.1 and 12,000,000 for RHEL 7.2 Alpha.

This demonstrates the benefits of optimization in the Tweaked 7.2 Alpha version, highlighting its superior performance in kernel networking.
Questions?

• Alexander Duyck
  • alexander.h.duyck@redhat.com
  • AlexanderDuyck@gmail.com