Super Networking Performance

Tom Herbert

6/14/2011

The need for speed

- Want *much* higher networking performance
  - High performance mode for apps that need it
  - Optimize for latency, throughput, CPU utilization
  - No negative impact to low end or standard path
  - Scale to 100 cores, 40 and 100 Gbps
- Motivated by applications, new technology
  - Networking shouldn't be bottleneck for tightly coupled computing model or low latency apps
  - Technology drivers: fast network storage/memory, flash, HFT and HPC transactions

Onload as proof of concept

- Stack in userspace (not offload)
  - LD_PRELOAD for sockets
  - Poll device queues directly for lowest latency
  - HW support just MQ + flow steering
- netperf RR (Solarflare onload)
  - TCP: 8.5 usecs RTT, 4M tps/6 cores, 0.8M/1 core
  - UDP: 8.0 usecs RTT latency, 4M tps/4 cores, 1M/1 core
- Load balancer application
  - Raw queue access to user space
  - 1.3M tps not accelerated (16 cores)
  - 3M tps on one core

Some stack experiments

- Force NAPI polling
  - Hack driver to always return budget
- kNetperf
  - Data path implemented in kernel thread

Test	50% RTT	90% RTT	99% RTT
Default	34	38	43
Force polling	27	28	32
kNetperf	30	34	36
Polling+kNetperf	17	18	27

Performance goals

- Latency
  - Unloaded latency: 5 usec. RTT (over TCP)
  - At 5M tps, 99th% latency 15 usecs RTT
  - High priority blocked by lower for 1 MTU at most
- Throughput
  - One CPU can do 40Gbps streaming
  - 25M tps on a single system
- CPU utilization
  - One CPU can do 5M tps
  - Linear scaling pps with number of CPUs

Techniques

- Per flow packet steering
  - Programmable 4-tuple filters mapping to queues
  - Accelerated RFS and more
- HW QoS
  - High priority packet waits at most one MTU time for a low priority packet
- Spin polling
  - Poll HW queues directly from read/poll syscalls
  - Tradeoff low latency for CPU

mmap networking

- Sockets
  - Like PF_PACKET, but extend to protocol sockets (UDP, TCP, etc.)
  - One syscall just to initiate IO and polling
- Device buffers
  - Like FreeBSD netmap
  - Combine with mmap sockets and flow steering
  - Per queue buffer
  - Zero copy send and receive

Miscellany

- In development
  - Byte queue limits
  - Doc on packet steering
  - SO_REUSEPORT
  - TCP fast open
  - TCP proportional rate recovery
- Open
  - HTB: interface lock is still a pain
  - Netdev flags: what is needed?
  - Sendgroup: as pseudo multicast
  - Device rate limiting: integrate into stack?