Hardware Selection and Kernel Tuning for High Performance Networking

Dec 07, 2006
SLAC, Berlin

by

Harald Welte <laforge@gnumonks.org>

About the Speaker

Who is speaking to you?

- an independent Free Software developer
- Linux kernel related consulting + development for 10 years
- one of the authors of Linux kernel packet filter
- busy with enforcing the GPL at gpl-violations.org
- working on Free Software for smartphones (openezx.org)
- ...and Free Software for RFID (librfid)
- ...and Free Software for ePassports (libmrtd)
- ...and Free Hardware for RFID (openpcd.org, openbeacon.org)
- ...and the worlds first Open GSM Phone (openmoko.com)

Hardware selection is important

Hardware selection is important

- linux runs on about anything from a cellphone to a mainframe
- good system performance depends on optimum selection of components
- sysadmins and managers have to undestand importance of hardware choice
- determine hardware needs before doing purchase !

Network usage patterns

Network usage patterns

- TCP server workload (web server, ftp server, samba, nfs-tcp)
  - high-bandwidth TCP end-host performance
- UDP server workload (nfs udp)
  - don’t use it on gigabit speeds, data integrity problems!
- Router (Packet filter / IPsec / ...) workload
  - packet forwarding has fundamentally different requirements
  - none of the offloading tricks works in this case
  - important limit: pps, not bandwidth!

Contemporary PC hardware

Contemporary PC hardware

- CPU often is extremely fast
  - 2GHz CPU: 0.5nS clock cycle
  - L1/L2 cache access (four bytes): 2..3 clock cycles
- everything that is not in L1 or L2 cache is like a disk access
  - 40..180 clock cycles on Opteron (DDR-333)
  - 250.460 clock cycles on Xeon (DDR-333)
- I/O read
  - easily up to 3600 clock cycles for a register read on NIC
  - this happens synchronously, no other work can be executed!
- disk access
  - don’t talk about it. Like getting a coke from the moon.

Hardware selection

Hardware selection

- CPU
  - cache
    - as much cache as possible
    - shared cache (in multi-core setup) is great
  - SMP or not
    - problem: increased code complexity
    - problem: cache line ping-pong (on real SMP)
    - depends on workload
    - depends on number of interfaces!
    - Pro: IPsec, tc, complex routing
    - Con: NAT-only box

Hardware selection

Hardware selection

- RAM
  - as fast as possible
  - use chipsets with highest possible speed
  - amd64 (Opteron, ..)
    - has per-cpu memory controller
    - doesn’t waste system bus bandwidth for RAM access
  - Intel
    - has a traditional ’shared system bus’ architecture
    - RAM is system-wide and not per-CPU

Hardware selection

Hardware selection

- Bus architecture
  - as little bridges as possible
    - host bridge, PCI-X / PXE bridge + NIC chipset enough!
  - check bus speeds
  - real interrupts (PCI, PCI-X) have lower latency than message-signalled interrupts (MSI)
  - some boards use PCIe chipset and then additional PCIe-to-PCI-X bridge :(

Hardware selection

Hardware selection

- NIC selection
  - NIC hardware
    - avoid additional bridges (fourport cards)
    - PCI-X: 64bit, highest clock rate, if possible (133MHz)
- NIC driver support
  - many optional features
    - checksum offload
    - scatter gather DMA
    - segmentation offload (TSO/GSO)
    - interrupt flood behaviour (NAPI)
  - is the vendor supportive of the developers
    - Intel: e100/e1000 docs public!
  - is the vendor merging his patches mainline?
    - Syskonnect (bad) vs. Intel (good)

Hardware selection

Hardware selection

- hard disk
  - kernel network stack always is 100% resident in RAM
  - therefore, disk performance not important for network stack
  - however, one hint:
    - for SMTP servers, use battery buffered RAM disks (Gigabyte)

Network Stack Tuning

Network Stack Tuning

- hardware related
  - prevent multiple NICs from sharing one irq line
    - can be checked in /proc/interrupts
    - highly dependent on specific mainboard/chipset
  - configure irq affinity
    - in an SMP system, interrupts can be bound to one CPU
    - irq affinity should be set to assure all packets from one interface are handled on same CPU (cache locality)

Network Stack Tuning

Network Stack Tuning

- 32bit or 64bit kernel?
  - most contemporary x86 systems support x86_64
  - biggest advantage: larger address space for kernel memory
  - however, problem: all pointers now 8bytes instead of 4
  - thus, increase of in-kernel data structures
  - thus, decreased cache efficiency
  - in packet forwarding applications, ca. 10% less performance

Network Stack Tuning

Network Stack Tuning

- firewall specific
  - organize ruleset in tree shape rather than linear list
  - conntrack: hashsize / ip_conntrack_max
  - log: don’t use syslog, rather ulogd-1.x or 2.x

Network Stack Tuning

Network Stack Tuning

- local sockets
  - SO_SNDBUF / SO_RCVBUF should be used by apps
  - in recent 2.6.x kenrnels, they can override /proc/sys/net/ipv4/tcp_[rw]mem
  - on long fat pipes, increase /proc/sys/net/ipv4/tcp_adv_win_scale

Network Stack Tuning

Network Stack Tuning

- core network stack
  - disable rp_filter, it adds lots of per-packet routing lookups
  - check linux-x.y.z/Documentation/networking/ip-sysctl.txt for more information

Links

Links

- The Linux Advanced Routing and Traffic Control HOWTO
  - http://www.lartc.org/
- The netdev mailinglist
  - netdev@vger.kernel.org