Processing of hardware interrupts in Linux

Petr Holášek, Red Hat
August 17, 2015
HW and kernel
Interrupt

- Hardware interrupt vs softIRQ
- Interrupt ReQuest from hardware
- In system represented as interrupt vector
- Pin-based vs MSI(-X)
Pin-based IRQ

• Triggered by electronic signal
• Pin can be shared
• Possible race condition
MSI(-X)

- Message Signaled Interrupts
- Introduced with PCI 2.2
- Triggered after write to an address
- Improved version called MSI-X
Interrupt controller

• APIC
  • LAPIC (local APIC) - at CPU
  • IOAPIC (I/O APIC) – at device
• Using system bus
• APIC bus deprecated
Interrupt handler

- Handles received interrupt
- Need for speed
- Most of the work deferred
- Using tasklets or workqueues
Interrupt handler

- Multiple CPUs cannot parallelize interrupt handler
- Only one interrupt handler running on CPU at time
- CPUs can alternate in handling the handler
Kernel interfaces

• The only visible info for user
  • /proc/interrupts
  • /proc/irq/<irqnum>/...
  • /sys/devices/.../irq
  • /proc/stats
Interrupt affinity

- Mask of possibly receiving processors
- `/proc/irq/X/smp_affinity`
- Hexadecimal mask or list
- Its value doesn't mean much
Interrupt distribution

• Should be done on multiprocessor systems
• Storage devices, NICs
• Risk of CPU overload or cache misses
Hardware topology

- NUMA node
- Package
- Cache domain – L2 or L3
- CPU
- `numactl` tool
Optimal affinity layout

• Identify and group all high-volume interrupts

• Move them to unique single CPUs

• Spread out lower-volume interrupts among other CPUs

• Do it within the device NUMA node
irqbalance
Irqbalance

• Interrupts load balancing daemon

• Can improve performance and save power

• https://github.com/Irqbalance/irqbalance

• Support for NUMA
Irqbalance basics

- Balancing of interrupts is complex task
- Periodic review of system
- Affinity management among heterogeneous systems
Irqbalance basics 2

• Don't migrate interrupt out of home NUMA node

• CPU load - time spent in interrupt and softIRQ context
Irqbalance algorithm 1

- Parse all available interfaces
- Evaluate overloaded processors
- Evaluate the busiest IRQs
- Rebalance IRQ on processors
Irqbalance algorithm 2

- Set new `smp_affinity` values
- Wait some time and repeat
Irqbalance options

- Can respect `affinity_hint` set by driver
- Can ignore selected IRQs
- Can ignore isolated CPUs
Alternatives to irqbalance
“Premature optimization is the root of all evil.”

Donald E. Knuth
Manual pinning

- Recent irqbalance 1.x addresses most of the discovered bugs
- But sometime manual pinning is still better
- Real-time, HPC
Rules of manual pinning

• Don't set affinity mask to all CPUs
• Move affinity to device rather than to process
• Let the scheduler do its work
• Consider faulty hardware
Kernel IRQ balancing

- Dropped by 8b8e8c in 2008
- Return is not planned so far
- Interrupt locality ideas
Give irqbalance a second chance

• Explore recent version
• Some new features are coming soon
• Try to compare manual pinning and irqbalance