How to do nothing efficiently or better laziness.

No idle tick on x86-64

Andi Kleen, SUSE Labs ak@suse.de

## Idle loop basics

- □ Executed when the CPU is idle
- □"Hottest" code path in kernel
- □ Optimizing idle is important contrary to folklore
  - Saving power
  - Virtualization
  - OLow latency
- □ Idling longer can result in better performance

### Idle loop II

- □ Waits for an interrupt or a reschedule.
- □ Special hardware and firmware support to idle better
  - OHLT, MONITOR/MWAIT, SMIs,
  - OTurns off parts of the CPU
  - OAllows hypervisor to schedule other processes
- □ Flavours
  - OHLT, poll, MONITOR/MWAIT, ACPI

#### Timer basics

- □ Regular timer interrupt each jiffie
  - OBasic time unit (usually 1ms, 4ms, 10ms)
  - OMinimum unit of sleep without special hardware
- □Time of the day (xtime)
- □Kernel timers (add\_timer)
- □ Rescheduling
- □ Process/interrupt time accounting

#### Timer tradeoffs

- ☐ Shorter jiffie gives more accurate timers
  - oe.g. 10ms wasn't enough for MPEG
  - ○1ms was a bit too wasteful, now back at 4ms
- ☐But wakes up the idle loop
- □100, 250, 1000, ... times
  - Costs power
  - Problem for virtualization

### Timer flavours

□xtime vs gettimeofday vs jiffies

□ PIT

□**HPET** 

□ APIC

□TSC

□pmtimer

#### no idle tick

- □Some other architectures already have it
  - os390, i386, arm, xen
  - OHave various problems
- □x86-64 implementation from scratch
- □CONFIG\_NO\_IDLE\_HZ
  - O/proc/sys/kernel/hz\_timer sysctl

# Implementation

- □ Idle notifier
  - OAlso useful e.g. for fixing up oprofile
- □When entering idle turn off timer
  - Or rather ask timer subsystem for next event
- □ Catch up with time on exiting
- □ Reprogram timer without losing ticks
  - OAdds more drift on some timers
  - Can avoid this by using different timer for backing time

### CPU sleep states

- □Lots of sleep states on PCs: C1-C4, S1-S3, P states, G\* ...
  - Only care about C states here
- □Increasing latency
  - Can do lots of work in SMM mode
  - OACPI FADT tells us about them
  - ○On this laptop 1us, 1us, 85us, 185us
  - OSMP traditionally only C1, but changing
- □But useless when latency is near jiffie

## Moving between sleep states

- □ Ideally want average sleep time much larger than latency
- □Linux ACPI algorithm needs idle ticks
- □Code from Thomas Renninger to estimate average sleep time
  - OBus mastering needs to be taken into account in C3
  - OAnd then go directly into right sleep state
  - Sampling problem

#### Generic Problems

- □ Accounting
  - ○Can batch on wakeup
- □ Read Copy Update
  - OAdvanced locking that needs regular feedback from all CPUs.
  - Currently uses bad hack
  - ○To tell other CPUs without adding too much synchronization
- □ Rescheduling tick on SMP
  - Push rebalancing to busy CPUs

## x86-64/PC specific problems

- □PIT is nasty to reprogram and loses time if you do it
  - Original PC-AT time with bizarre slow ioport interface
- ☐ HPET support is often missing
  - OAnd HPET implementations differ
  - Still problems with implementation
- □Local APIC timer is not reliable
  - ○Fast, easy to program
  - ○Needed on SMP
  - OStops in deeper C states and accuracy issues

### Sleep disturbances

- □ Various subsystems
  - OFortunately only a few really bad offenders
  - ONeed to fix a lot of kernel code
- □USB polling
  - OPrevents entering C3 big problem.
- □User space
  - "wiggling applets eat your battery"
  - OPower needs to be kept in mind when designing desktops!
  - ONeed better tools

# Sample profile - console bootup, typing

3752 i8042\_timer\_func+0
2708 process\_timeout+0
1869 rh\_timer\_func+0
1143 it\_real\_fn+0
131 delayed\_work\_timer\_fn+0
113 e1000\_watchdog+0
98 cfq\_idle\_slice\_timer+0
89 neigh\_periodic\_timer+0
48 wb\_timer\_fn+0
44 acpi\_thermal\_run+0
26 commit\_timeout+0
8 kd\_nosound+0

### KDE bootup, konqueror, konsole

1166 i8042 timer func+0 1070 process timeout+0 kmix 792 process\_timeout+0 swapper 764 process timeout+0 kded 604 rh timer func+0 593 process timeout+0 X 540 it real fn+0 X 214 process\_timeout+0 kicker 197 process\_timeout+0 powersaved 153 process timeout+0 ksplashx 144 cfg idle slice timer+0 136 process\_timeout+0 kdesktop 113 process timeout+0 klipper 95 process\_timeout+0 suseplugger 94 process timeout+0 konsole 88 process\_timeout+0 konqueror 80 process\_timeout+0 watchdog/0 67 process\_timeout+0 smpppd 63 process\_timeout+0 kwin 45 process\_timeout+0 kwrapper 45 delayed\_work\_timer\_fn+0 40 process\_timeout+0 blogd

39 process\_timeout+0 udev

#### **Improvements**

- □next timer interrupt lookup in heap inefficient
  - OBetter to keep track of wakeup point
- □ Inaccurate timers for better batching
- □ No tick even when not idle
  - ○Needed for normal wakeup <1ms
  - OAccounting is difficult
  - OEasy would be to do a low frequency tick and speed up on demand
  - OPerformance counters?