ECE 471 – Embedded Systems
Lecture 23

Vince Weaver

http://www.eece.maine.edu/∼vweaver

vincent.weaver@maine.edu

26 November 2013

Announcements

- Project
- Don't forget HW#5
- Plan9 for Raspberry Pi

HW#4 Notes

- The displays we are using are LED not LCD
- Commenting code redux. It is subjective, but try to do better.
- Cable run question: USB cannot run 50 feet without help. 1-wire can, as can canbus.
- It's useful to check for error returns, especially on things like fopen()

Power Saving Strategies

big.LITTLE / Heterogeneous Computing

- ARM
- big = Cortex A15 = power hungry, fast, high-leakage
- little = Cortex A7 = low power, slow
- "big.LITTLE switcher" by Pitre. have 1:1, move from slow to fast when need the speed
- have all procs visible to Linux, schedule them with intelligent scheduler

- Can use cpufreq interface, “big” just seen as higher frequency operating point

Race to Idle

- Good strategy on high-leakage chips (Intel?)
- Depends on how CPU bound process is
- Example 1:
  - If 34W full speed, 24W half speed, 1W idle, total time 1s
  - 1s at half speed, 24W * 1s = 24J
  - 0.5s at full speed, 0.5s at idle: 34W * 0.5 + 1W*0.5=17.5J

- Example 2:
  - Instead, 34W full speed, 24W half, 20W idle
  - 1s at half speed, 24W * 1s = 24J
  - 0.5s at full speed, 0.5s at idle: 34W*0.5s + 20W*0.5s = 27J

Operating System Power Saving Strategies

- We look primarily at Linux, as it is open source and technical debates happen in the open
- Windows and OSX often have measurably better laptop Energy behavior due to tuning and better hardware testing

We previously discussed Power Governors

- With the ondemand governor the kernel controls DVFS

Tickless idle / NOHz

- Gets rid of the periodic timer tick (wakeups use Energy)
- Linux typically has periodic timer interrupt at 100, 250, or 1000Hz. Used to implement various timers, accounting, and context switch. Waste of energy if system is idle! (also, what if large IBM system with hundreds of VMs all doing nothing but ticking?)
- Use timers, only schedule a wakeup if needed
- Want to limit wakeups, as they bring CPU out of sleep

mode or idle

- Group close-enough timers together. deferrable timers
- Depends on userspace staying quiet if possible. Userspace does foolish stuff, like poll for file changes or drive status, blinking cursor, etc.
- Semi-related “NOHz tasks”: Turn off all interrupts, turn CPU into compute core for HPC

Suspend

- Linux supports three states:

1. Standby – minimal latency, higher energy
2. Suspend to RAM – similar to standby, lower energy. Everything except RAM refresh and wakeup events turned off
3. Suspend to Disk – even lower energy, high latency

Suspend to RAM

- Platform driver provides suspend-to-ram interface
- Often a controller supports fans, batteries, button presses, wakeup events, etc.
- ACPI interpreter runs in kernel, reads table or AML, essentially takes program from BIOS and runs in kernel interpreter
- PCI has D states, D0 (awake) to D3 (asleep). D1 and D2 are in between and optional and not used

- User can start suspend to RAM via ioctl or writing “mem” to /sys/power/state

What happens during Suspend to RAM

- grabs mutex (only one suspend at once). Syncs disk. Freezes userspace.
- suspends all devices. Down tree, want leaf suspended first
- disables non-boot CPUs
- disable interrupts, disable last system devices
- Call system sleep state init

What happens during Wakeup

- Wakeup event comes in (WOL, button, lid switch, power switch, etc.)
- CPU reinitialized (similar to bootup code)
- other CPUs reactivated
- devices resumed
- tasks unfrozen

- mutex released
- ISSUES: firmware re-load? where stored (problem if on disk or USB disk, etc. must store in memory?)
- Graphics card coming back, as X in userspace until recently. kernel mode setting helps

The Linux Scheduler

- People often propose modifying the scheduler. That is tricky.
- Scheduler picks which jobs to run when.
- Optimal scheduler hard. What makes sense for a long-running HPC job doesn't necessarily make sense for an interactive GUI session. Also things like I/O (disk) get involved.
- You don't want it to have high latency

- Linux originally had a simple circular scheduler. Then for 2.4 through 2.6 had an O(N) scheduler
- Then in 2.6 until 2.6.23 had an O(1) scheduler (constant time, no many how many processes).
- Currently the “Completely Fair Scheduler” (with lots of drama). Is O(log N). Implementation of “weighted fair queuing”
- How do you schedule? Power? Per-task (5 jobs, each get 20%). Per user? (5 users, each get 20%).

Per-process? Per-thread? Multi-processors? Hyper-threading? Heterogeneous cores? Thermal issues?

Power-Aware Scheduler

- Most of this from various LWN articles
- Linux scheduler is complicated
- maintainers don't want regressions
- Can handle idle OK, maxed out OK. lightly loaded is a problem
- 2.6.18 - 3.4 was sched_mc_power_savings in sysctl but not widely used, removed

- “packing-small-tasks” patchset – move small patchsets to CPU0 so not wake up other sleeping CPUs
  small defined as 20% of CPU time
- knowledge of shared power lines. treat CPUs that must go idle together as a shared entity scheduling wise (buddy)
- how does this affect performance (cache contention)
- Shi’s power-aware scheduling
- move tasks from lightly loaded CPUs to others with

capacity

- if out of idle CPUs, then ramp up and race-to-idle
- Heterogeneous systems (such as big.LITTLE)
- Rasmussen mixed-cpu-power-systems patchset maxed out little CPU, move task to big CPU
- task tries to use the little CPUs first before ramping up big

Wake Locks and Suspend Blockers

- See "Technical Background of the Android Suspend Blockers Controversy" by Wysocki, 2010.

- Low-power systems want "opportunistic suspend"

- Google Android propose this interface, kernel developers push back

- System spends much of time in sleep, with just enough power to keep RAM going and power sources of events

- A Wake Lock prevents the kernel from entering low power state
- WAKE_LOCK_SUSPEND – prevent suspending
  WAKE_LOCK_IDLE – avoid idling which adds wakeup latency
- Try to avoid race conditions during suspend and incoming events. For example, system trying to suspend, incoming call coming in, don't let it lose events and suspend. Take lock to keep it awake until call over.
- Kernel high-quality timing suspended, sync with low-

quality RTC, time drifts

- Kernel developers not like for various reasons. All drivers have to add explicit support. User processes. What happens when process holding lock dies.

- You have to trust the apps (gmail) to behave and not waste battery, no way for kernel to override.

CPU Idle Framework?

- In kernel, kernel developers suggest it can be used instead of wake locks. Gives more control to kernel, doesn't trust userspace.
- Tracks various low-power CPU "C-states". Knows of Power consumption vs exit latency tradeoffs
- Lower C-states take power to come back, and might do things like flush the cache.
- kernel registers various C-state "governors" with info on

them.

The kernel uses the pm_qos value to choose which to enter.

- QOS say I need latencies better than 100us, so if suspend takes longer can't enter that suspend state
- /sys/devices/system/cpu/cpu0/cpuidle has power and latency values, among other things
- CPU idle stats, turbostat
- ACPI issues. Doesn't always accurately report C-states,

latencies

- ACPI_IDLE driver
- Alternate INTEL_IDLE as poorly written BIOSes not idling well on intel

Tools

- There are various tools that can show you status of power under Linux, configure settings, etc.

- Unfortunately you usually have to run these as root

Tools – Powertop

- Shows cstates, wakeups, suggested settings, gpu power
- On laptops with battery connected can estimate energy/power based on battery drain

Powertop–Overview

```
Summary: 344.6 wakeups/second,  0.0 GPU ops/seconds, 0.0 VFS ops/sec

        Usage         Events/s     Category       Description
      25.1 ms/s       268.6         Process        swirl -root
     100.0%                         Device         Audio codec hwC0D3: Intel
     100.0%                         Device         Audio codec hwC0D0: Cirru
259.1 M-BM-5s/s        29.6         kWork          od_dbs_timer
 11.1 M-BM-5s/s        17.8         Timer          menu_hrtimer_notify
      34.2 ms/s         2.0         Process        /usr/bin/X :0 vt7 -nolist
       1.2 ms/s        10.9         Timer          hrtimer_wakeup
326.0 M-BM-5s/s         4.9         Timer          tick_sched_timer
       5.1 ms/s         1.0         Process        powertop
 33.3 M-BM-5s/s         2.0         Interrupt      [3] net_rx(softirq)
484.3 M-BM-5s/s         1.0         Interrupt      [7] sched(softirq)
 75.4 M-BM-5s/s         1.0         Process        sshd: vince@pts/1
 46.6 M-BM-5s/s         1.0         Timer          watchdog_timer_fn
```

Powertop – Idle Stats

```
          Package    |            Core     |            CPU 0       CPU 2
                     |                     | C0 active   1.3%         0.4%
                     |                     | POLL        0.0%    0.0 ms  0.
                     |                     | C1-IVB      0.4%    0.3 ms  0.
C2 (pc2)    1.1%     |                     |
C3 (pc3)    0.0%     | C3 (cc3)    0.4%    | C3-IVB      0.4%    0.3 ms  0.
C6 (pc6)    1.5%     | C6 (cc6)    0.0%    | C6-IVB      0.0%    0.0 ms  0.
C7 (pc7)   90.1%     | C7 (cc7)   94.9%    | C7-IVB     96.4%    7.4 ms 99.

                     |            Core     |            CPU 1       CPU 3
                     |                     | C0 active   0.6%         1.1%
                     |                     | POLL        0.0%    0.0 ms  0.
                     |                     | C1-IVB      0.0%    0.1 ms  0.
                     |                     |
                     | C3 (cc3)    0.0%    | C3-IVB      0.0%    0.3 ms  0.
                     | C6 (cc6)    0.0%    | C6-IVB      0.0%    0.0 ms  0.
                     | C7 (cc7)   96.0%    | C7-IVB     98.8%   26.2 ms 97.
```

Powertop – Frequency Stats

```
            Package |            Core    |            CPU 0       CPU 2
                    |                    | Actual     1202 MHz     1198 M
Turbo Mode    0.0%  | Turbo Mode    0.0% | Turbo Mode   0.0%         0.0%
2.50 GHz      0.0%  | 2.50 GHz      0.0% | 2.50 GHz     0.0%         0.0%
2.40 GHz      0.0%  | 2.40 GHz      0.0% | 2.40 GHz     0.0%         0.0%
2.31 GHz      0.0%  | 2.31 GHz      0.0% | 2.31 GHz     0.0%         0.0%
2.21 GHz      0.0%  | 2.21 GHz      0.0% | 2.21 GHz     0.0%         0.0%
2.10 GHz      0.0%  | 2.10 GHz      0.0% | 2.10 GHz     0.0%         0.0%
2.00 GHz      0.0%  | 2.00 GHz      0.0% | 2.00 GHz     0.0%         0.0%
1.91 GHz      0.0%  | 1.91 GHz      0.0% | 1.91 GHz     0.0%         0.0%
...
1500 MHz      0.0%  | 1500 MHz      0.0% | 1500 MHz     0.0%         0.0%
1400 MHz      0.0%  | 1400 MHz      0.0% | 1400 MHz     0.0%         0.0%
1300 MHz      0.0%  | 1300 MHz      0.0% | 1300 MHz     0.0%         0.0%
1200 MHz      2.4%  | 1200 MHz      2.4% | 1200 MHz     2.4%         0.0%
Idle         97.6%  | Idle         97.6% | Idle        97.6%       100.0%
```

Powertop – Device Stats

```
       Usage     Device name
        4.7%       CPU use
      100.0%       Audio codec hwC0D3: Intel
      100.0%       Audio codec hwC0D0: Cirrus Logic
        0.0 ops/s  GPU
      100.0%       USB device: IR Receiver (Apple, Inc.)
      100.0%       USB device: BRCM20702 Hub (Apple Inc.)
      100.0%       USB device: usb-device-0424-2512
      100.0%       PCI Device: Broadcom Corporation BCM4331 802.11a
      100.0%       PCI Device: Intel Corporation Xeon E3-1200 v2/3r
      100.0%       PCI Device: Intel Corporation 3rd Gen Core proce
      100.0%       Radio device: btusb
      100.0%       USB device: Bluetooth USB Host Controller (Apple
      100.0%       USB device: USB Keykoard (USB)
      100.0%       USB device: Dell USB Mouse (Dell)
      100.0%       PCI Device: Broadcom Corporation NetXtreme BCM57
```

Powertop – Tunables

```
>> Bad         VM writeback timeout
   Bad         Enable SATA link power Managmenet for host0
   Bad         Enable SATA link power Managmenet for host1
   Bad         Enable SATA link power Managmenet for host2
   Bad         Enable SATA link power Managmenet for host3
   Bad         Enable SATA link power Managmenet for host4
   Bad         Enable SATA link power Managmenet for host5
   Bad         Enable Audio codec power management
   Bad         NMI watchdog should be turned off
   Bad         Autosuspend for USB device Bluetooth USB Host Controller
   Bad         Autosuspend for USB device USB Keykoard [USB]
   Bad         Autosuspend for USB device IR Receiver [Apple, Inc.]
   Bad         Autosuspend for USB device Dell USB Mouse [Dell]
   Bad         Runtime PM for PCI Device Intel Corporation 7 Series/C210
   Bad         Runtime PM for PCI Device Intel Corporation Xeon E3-1200
   Bad         Runtime PM for PCI Device Intel Corporation 3rd Gen Core
```

Tools – Cpufreq

- `cpufreq-info` (no root) shows info of current governor and frequency states, etc.
- `cpufreq-set` (needs root) – set governor or frequency
- `cpurfreq-apert` (needs root) – shows aperf/mperf settings from MSR. Useful for determining frequency values?

cpufreq-info

```
analyzing CPU 3:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 0 1 2 3
  CPUs which need to have their frequency coordinated by software: 3
  maximum transition latency: 10.0 us.
  hardware limits: 1.20 GHz - 2.50 GHz
  available frequency steps: 2.50 GHz, 2.50 GHz, 2.40 GHz, 2.30 GHz,
     2.20 GHz, 2.10 GHz, 2.00 GHz, 1.90 GHz, 1.80 GHz, 1.70 GHz,
     1.60 GHz, 1.50 GHz, 1.40 GHz, 1.30 GHz, 1.20 GHz
  available cpufreq governors: conservative, powersave, userspace,
        ondemand, performance
  current policy: frequency should be within 1.20 GHz and 2.50 GHz.
                  The governor ``ondemand'' may decide which speed to use
                  within this range.
  current CPU frequency is 1.20 GHz.
  cpufreq stats: 2.50 GHz:0.99%, 2.50 GHz:0.00%, 2.40 GHz:0.00%,
                 1.70 GHz:0.00%, 1.60 GHz:0.03%, 1.50 GHz:0.00%,
                 1.40 GHz:0.01%, 1.30 GHz:0.01%, 1.20 GHz:98.95%  (54321)
```

Powertop – aperf/mperf

- mperf is a counter that counts at the maximum frequency the CPU supports
- aperf counts at the current running frequency
- current frequency (for things like detecting TurboBoost) can be detected by the ratio

Tools – x86_energy_perf_policy

- allows adjusting the msr that tells how aggressive turbo mode is, among other things. hint at a performance vs power preference

- comes in Linux source tree in tools/power/x86/x86_energy_

Tools – Turbostat

- shows cstates, RAPL information, turboboost, other things from MSRs
- comes in Linux source tree in tools/power/x86/turbostat

Turbostat Output

```
./turbostat -S
  %c0  GHz  TSC SMI    %c1    %c3    %c6    %c7 CTMP PTMP   %pc2   %pc3
 1.34 1.99 2.29   0   2.72   0.05   0.01  95.88   44   45   2.84   0.02
 1.24 2.23 2.29   0   1.94   0.13   0.00  96.69   45   46   2.88   0.15
 1.56 1.77 2.29   0   2.98   0.11   0.00  95.35   43   47   2.63   0.12
 1.42 1.84 2.29   0   2.51   0.05   0.00  96.03   45   45   2.66   0.03
```

```
...
%pc6   %pc7  Pkg_W  Cor_W GFX_W
2.96  86.14   2.31   0.43  0.00
2.97  87.63   2.30   0.43  0.00
2.73  85.67   2.32   0.43  0.00
2.74  86.88   2.30   0.41  0.00
```

Tools – Sensors

- no need for root if configured right
- shows temps, fans, etc
- Various other sensors from i2c bus, etc.

Sensors Part 1

```
vince@mac-mini:~$ sensors
applesmc-isa-0300
Adapter: ISA adapter
Exhaust  :   1798 RPM  (min = 1800 RPM, max = 5500 RPM)
TA0P:         +37.0C
TA0p:         +37.0C
TA1P:         +37.8C
TA1p:         +37.8C
TC0C:         +42.0C
TC0D:         +44.8C
TC0E:         +42.8C
TC0F:         +43.2C
TC0G:         +99.0C
TC0J:          +0.2C
TC0P:         +40.8C
TC0c:         +42.0C
TC0d:         +44.8C
```

Sensors Part 2

```
TC0p:          +40.8C
TC1C:          +42.0C
TC1c:          +42.0C
TCGC:          +42.0C
TCGc:          +42.0C
TCPG:         +103.0C
TCSC:          +43.0C
TCSc:          +43.0C
TCTD:           -0.2C
TCXC:          +42.8C
TCXc:          +42.8C

coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +46.0C  (high = +87.0C, crit = +105.0C)
Core 0:         +42.0C  (high = +87.0C, crit = +105.0C)
Core 1:         +45.0C  (high = +87.0C, crit = +105.0C)
```

Example

Gumstix Overo – Naïve 300x300 double-precision floating point matrix-matrix multiply repeated 10 times.

Frequency	Idle P	Load P	Time	Energy
125MHz	2.5W	2.6W	112s	291J
250MHz	2.5W	2.7W	57.8s	156J
500MHz	2.7W	3.0W	31.6s	95J
550MHz	2.8W	3.1W	29.3s	91J
600MHz	2.8W	3.2W	27.4s	87J

Example – Governors

Gumstix Overo

- ondemand: 27.57s
- performance: 27.23s
- powersave: 111.0s

Example – Mixed Load

Gumstix Overo – Do one MatrixMatrix job every 5 minutes.

Frequency	Active Energy	Idle Energy	Total
125MHz	291J	470J	761J
250MHz	156J	605J	761J
500MHz	95J	723J	818J
550MHz	91J	756J	847J
600MHz	87J	761J	848J