Power Management

Goal: Improve the battery life of mobile devices

CPU Voltage & Frequency Scaling

- **Dynamic CPU Frequency Scaling**
  - Adjust the frequency of a CPU on the fly
  - Conserve power & reduce heat (reduce need for a fan)
  - Reduce # of instructions per time
  - Goal: use this when processes are not CPU bound

- **Examples of CPU support**:
  - Intel SpeedStep
  - AMD PowerNow!, AMD Cool ‘n’ Quiet
  - ARM Intelligent Energy Manager (iEM)

- **OS management of voltage/frequency control**
  - Linux: cpuspeed (RedHat) or cpufrequtils (Ubuntu)

Managing CPU performance

- **Governors**
  - Pre-configured power schemes
  - Loaded as kernel modules. Governors include:
    - cpufreq_performance: run at maximum speed (default)
    - cpufreq_ondemand: dynamically increase/decrease based on load
      - Programmable threshold based on % CPU utilization
    - cpufreq_conservative: similar to ondemand but slower changes
    - cpufreq_powersave: run CPU at minimum speed
    - cpufreq_userspace: allow user to configure speeds

CPUfreq system in Linux

- User-level governors
  - powersaved
  - cpuspeed

- In-kernel governors
  - Performance
  - Powersave
  - Userspace
  - Ondemand

- CPU-specific drivers
  - p4-clockmod
  - speedstep-centrino
  - acpi

- Advanced Configuration & Power Interface

ACPI Power Management

- **Global states (G)**
  - G0/G1: Working
    - G1: Sleeping
      - G1/S1: all CPU caches flushed, CPU stopped; power to CPU & RAM is ON
      - G1/S2: CPU is powered off
      - G1/S3: Standby/Sleep; RAM is powered on
      - G1/S4: Hibernation: Copy all of RAM to a swap partition or file
  - G2/G5: Soft OFF: most systems off but the machine can wake from LAN, USB, keyboard, or real-time clock inputs
  - G3: OFF (only the real-time clock running)
ACPI Power Management

- **Device Control (D)**
  - D0: Fully on
  - D1, D2: intermediate
  - D3: OFF and not responsive to the bus

- **CPU states (C)**
  - C0: normal operating state
  - C1: halt; not executing but can start instantly
  - C2: Stop-clock: CPU keeps state but takes longer to start
  - C3: Sleep: cache may not be updated

- **Power: Voltage/Frequency scaling (P)**
  - P0: maximum voltage & frequency
  - Pn: voltage and/or frequency scaled

Sleep & Hibernation

- **Sleep (standby) mode**
  - Stop processor execution, keep RAM powered

- **Hibernate mode**
  - Save memory state onto non-volatile storage (disk/flash)
  - Most systems are shut off
  - except USB/LAN/alarmswitch wake detection
  - Suspend-to-disk
  - Suspend-to-RAM

- **Hybrid**
  - Store contents to disk and then sleep
  - If power to memory is lost then wake via disk restore
  - Examples:
    - Windows Vista Fast Sleep & Resume
    - OS X Safe Sleep

Power Management: BIOS Support

- **Old interface: APM**
  - BIOS call; actions fully handled by hardware

- **Most PCs support ACPI**
  - Advanced Configuration and Power Interface
  - Fan control, dock/undock detection, temperature sensing, device control...
  - Intel provides a fixed function interface for control
  - Other systems are hardware-specific

Example

- Hit a sleep key, close lid, ...
  1. Hardware interrupt – interrupts CPU: general purpose event
  2. OS interrupt handler
  3. User-level power management daemon listens to events via /proc/acpi/events
  4. User process decides that the action requires a suspend to RAM
  5. Suspend to RAM initiated

  a. Script/program does initial work: unloads various drivers that are not power-management-aware
  b. Initiate suspend by echoing the right state into /sys/power/state
    - E.g., echo ”mem” > /sys/power/state
  c. Kernel steps user-level actions (process execution)
  d. Goes through each device: calls suspend methods on each active driver
  e. Call ACPI methods: PTS (Prepare To Sleep), GTS (Go To Sleep)
  f. Address of kernel wakeup code written to an address in the FADT – Fixed Address Descriptor Table in the ACPI
  g. While values are ACPI to sequence the machine to suspend
    - S3 state: shut the machine down but keep RAM on.

Example: Waking up

- User presses the power button
  1. BIOS starts kernel
  2. BIOS checks the ACPI status register: system was suspended to RAM
  3. Jumps to the programmed wake address
  4. Executes kernel-provided real-mode x86 code
  5. Restores register state, switches the CPU to protected mode
  6. Now the kernel is running
  a. Kernel
    - calls the ACPI WAK method
  b. Resumes all drivers
  c. Resumes userspace (scheduling)
  - The shell script that was running when we suspended resumes and reloads drivers.
Tickless Kernel

• Traditional kernel:
  – Periodic tick
  – Always ticking … whether the processor is busy or not
  – Used for
    • Timer management
    • Time slice management
    • SMP load balancing
  – Wakeup during idle is bad
  • Does not let CPU go to deep sleep states
  • Hurts battery life

• Tickless kernel:
  – On-demand timer interrupts
  – Turn off periodic tick when the CPU is idle
  – Clock event wakeup programmed based on next event

• Keep the kernel quiet
  – Group timers to avoid multiple interrupts
  – Round timeout values
  – Defer the expiration of non-critical timers during idle