



Embedded Linux Power Management on the Intel® Atom™ Processor

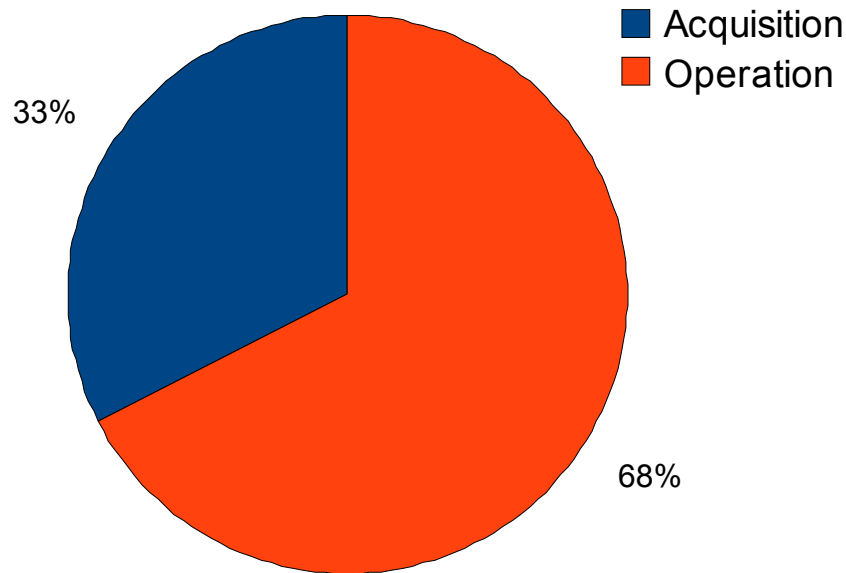
Brad Dixon
MontaVista Software

Kaitlin Murphy
Intel Corporation

- **Power Management (PM) Fundamentals**
- **Designing for PM**
- **PM Technologies in Linux[®]**
- **Stitching it Together**
- **Integrated in MontaVista Linux**
- **Available Tools**

Retail POS automation

The TCO of a POS deployment is highly influenced by operation costs:



- Many based on Intel Architecture and a variation of the standard “PC” platform.
- Wide array of unusual on-board and off-board peripherals
 - Magstripe readers
 - PIN entry devices
 - Receipt printers
 - Touchscreens
 - Multiple displays
 - Cash drawers
- Most commonly wall-socket powered with a local UPS for emergency operation.
- Large retailers network POS devices.



- **Power efficiency saves big money**
 - Drop power usage 33% on 5000 terminals: Save **\$1 million** in 7 years.
- **Power efficiency is on the mind of POS system purchasers**
 - Retailers consider energy efficiency as a factor when selecting POS systems:
 - Big Box: over 70%
 - Grocery: over 50%
 - Small Box: over 50%
- **As retail hours stretch...**
 - Basic on/off power saving is limited.
 - Only 50% of big box retailers use on/off.
 - Saving power while *open for business* is essential.



- **Power management is a system level design goal... not a software level design goal.**

Two big ways:

- **Turn stuff off**

- Clock trees, caches, displays, radios, USB, memory, anything you can get your hands on.

- **Clock stuff down and power it at a lower voltage**

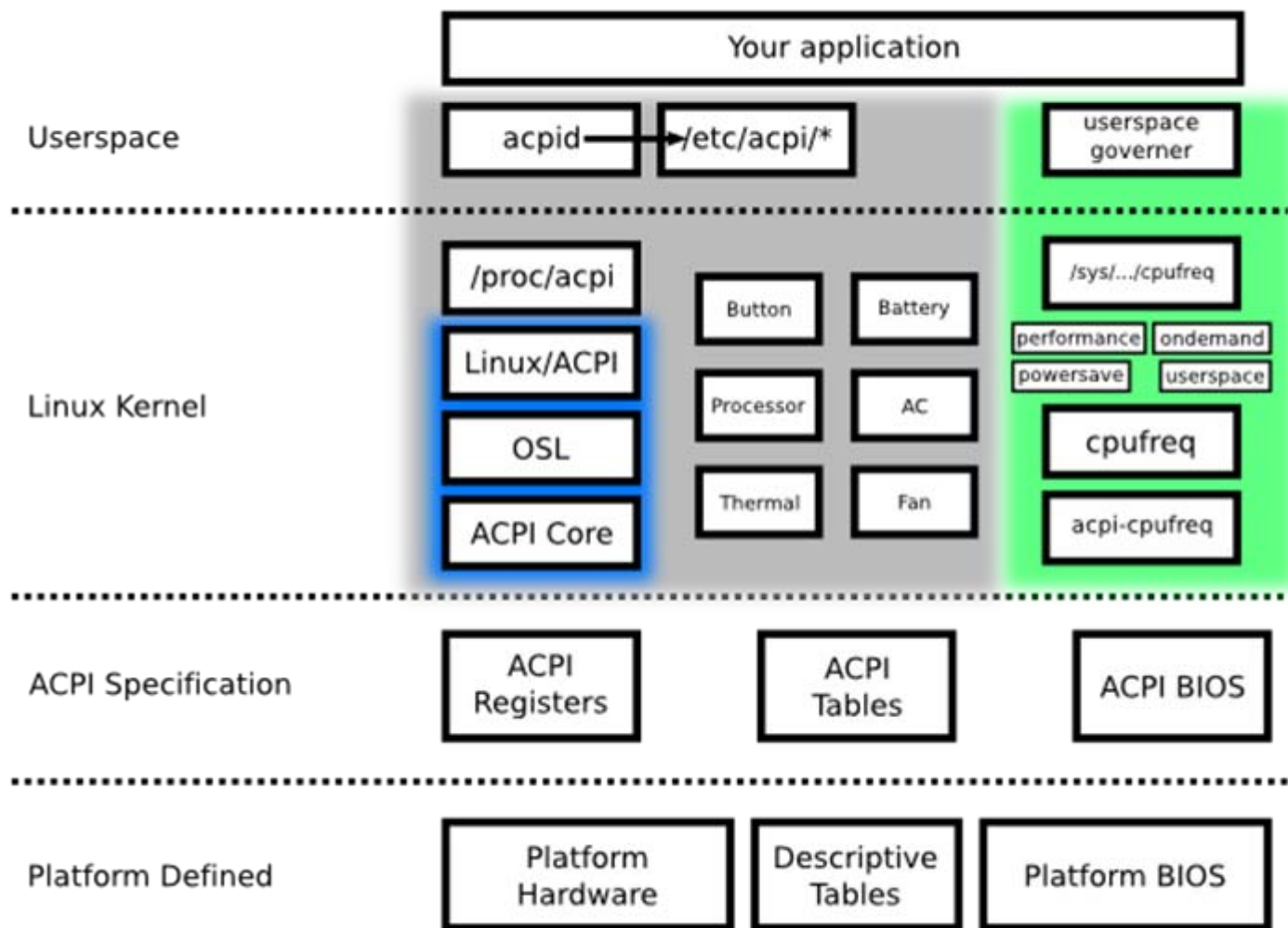
- $P = CV^2f$ in CMOS
- Switching capacitance
- Voltage (which also relates to frequency)
- Frequency

- **Question #1: How does the software find out what the hardware can do and how to tweak it?**

- Systems designed to be “PC” compatible do this very differently

Advanced Configuration and Power Interface

- Primary purpose is to abstract the hardware's power management details from the operating system.
- Enumerates a slew of hardware attributes to the system
 - Processors and their power management states
 - Relationship between processors and their resources (NUMA)
 - On board hardware like APIC's and embedded controllers
- Enumerates and glues power related “toys” to the OS
 - Lid switches
 - Thermal zone monitoring
 - Fans
 - AC and batteries
 - Brightness and volume controls
 - Oddball buttons



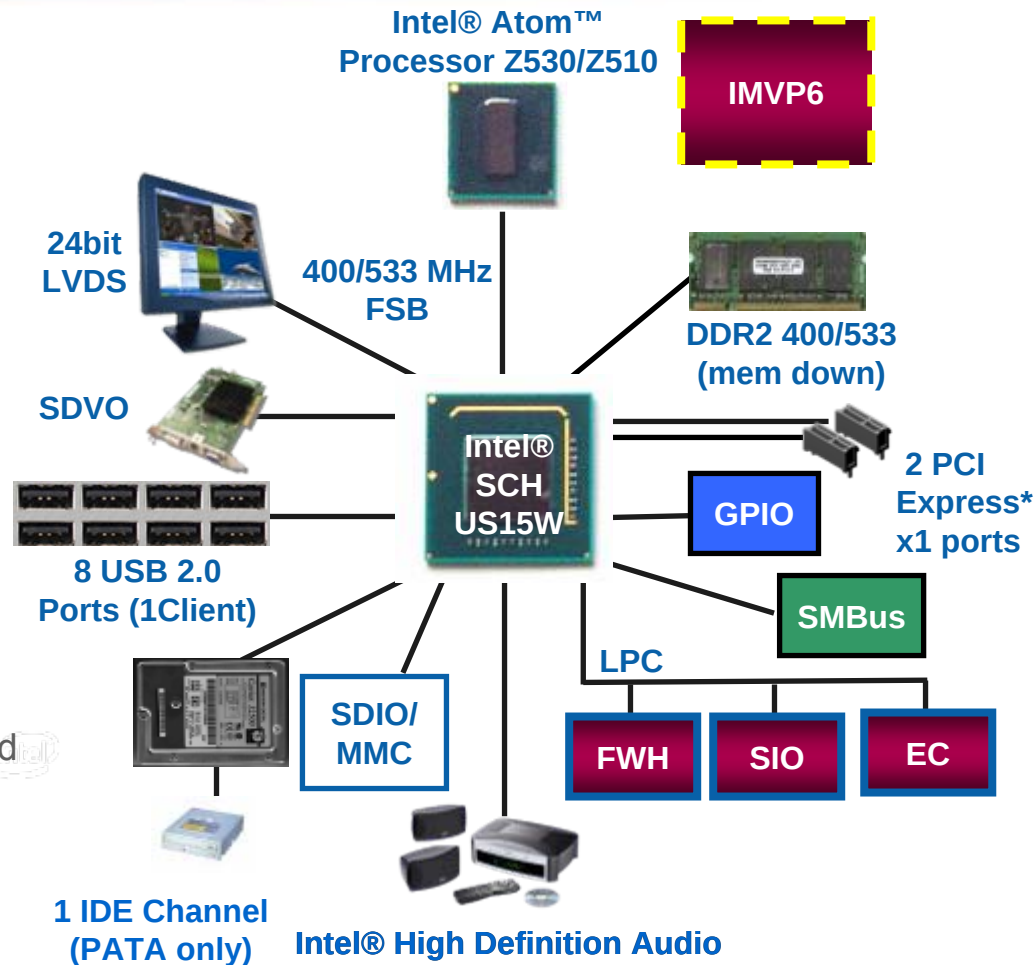
Intel® Atom™ Processor and Intel® System Controller Hub US15W Block Diagram

Intel® Atom™ Processor Z530 & Z510

- Low Power Microarchitecture on 45nm technology
- 512 KB L2 cache
- Ultra small 13x14mm package
- Performance and power optimized for small form-factor, ultra-low power embedded platforms (TDP 2.2W)
- Hyper-Threading Technology**

Intel® System Controller Hub US15W

- Integrated GMCH+ICH (single chip)
- 22x22mm package
- Up to 2 GB of DDR2 400/533 MHz memory down
- Ultra low power integrated graphics with 2D and 3D HW accelerator
- Integrated High Definition Video Decoder
- Expansion capability meeting USB2.0, SDIO/MMC and PCI-Express* standards
- 2.3W TDP



Highly Integrated 2-Chip Solution for Low Power, Small FF, Embedded Platforms

- **ACPI is an open-industry standard primarily used for power management.**
- **ACPI defines different types of power saving states**
 - G states (Global States)
 - S states (System states, which fall under G1)
 - P states (Performance States)
 - D states (Device states)
 - C states (Processor Power States)
- Combinations of ACPI states can be used to achieve platform power reduction



C states

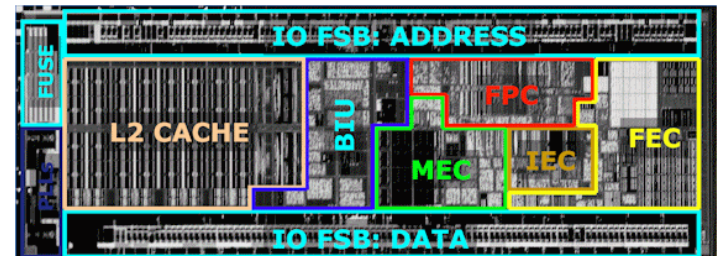
	C0 HFM	C0 LFM	C1/C2	C4	C6
Core voltage					
Core clock			OFF	OFF	OFF
PLL				OFF	OFF
L1 caches			 flushed	 flushed	 off
L2 caches				 Partial flush	 off
Wakeup time	active	active			
Power					

What is Deep Power Down Technology?

- Deep Power Down Technology (C6) is a new low power state designed to reduce power consumption due to leakage current.
- On software's request, the processor saves its state in a power-preserved domain (SRAM) and shuts down the power to the chip (core and caches)
- In this mode,
 - All caches are flushed.
 - The core is not functional and the state is not retained.
 - No snoops or MSIs from chipset are permitted while in C6.
- On exiting this state, power and clocks are restored, CPU is reset, state is restored, and execution resumes seamlessly.

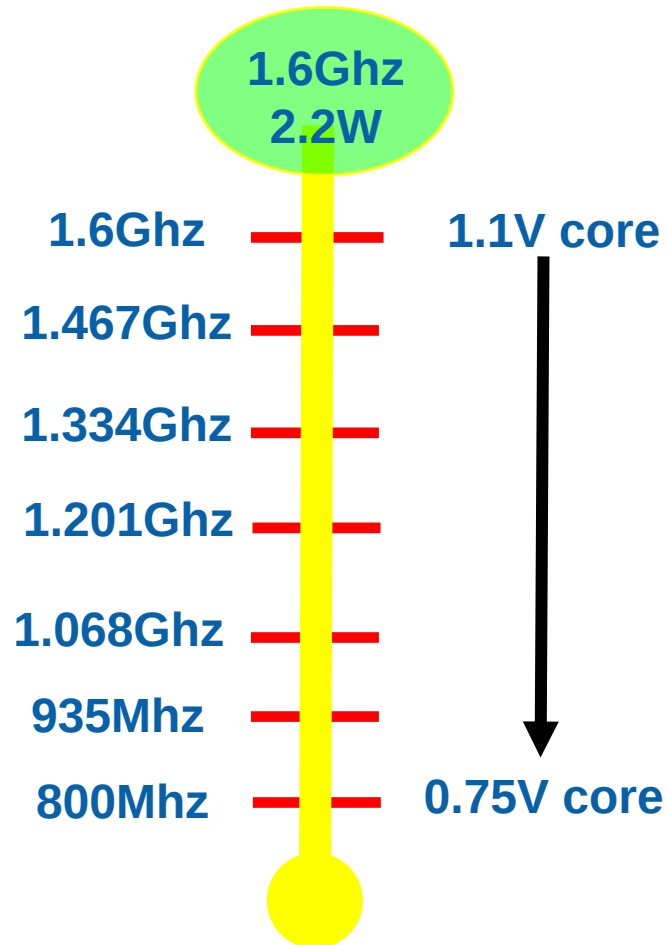
Benefits:

- Much greater leakage savings than in C4.
- Exit latencies well within today's C-state exit latencies.
- No software intervention needed for context save/restore.

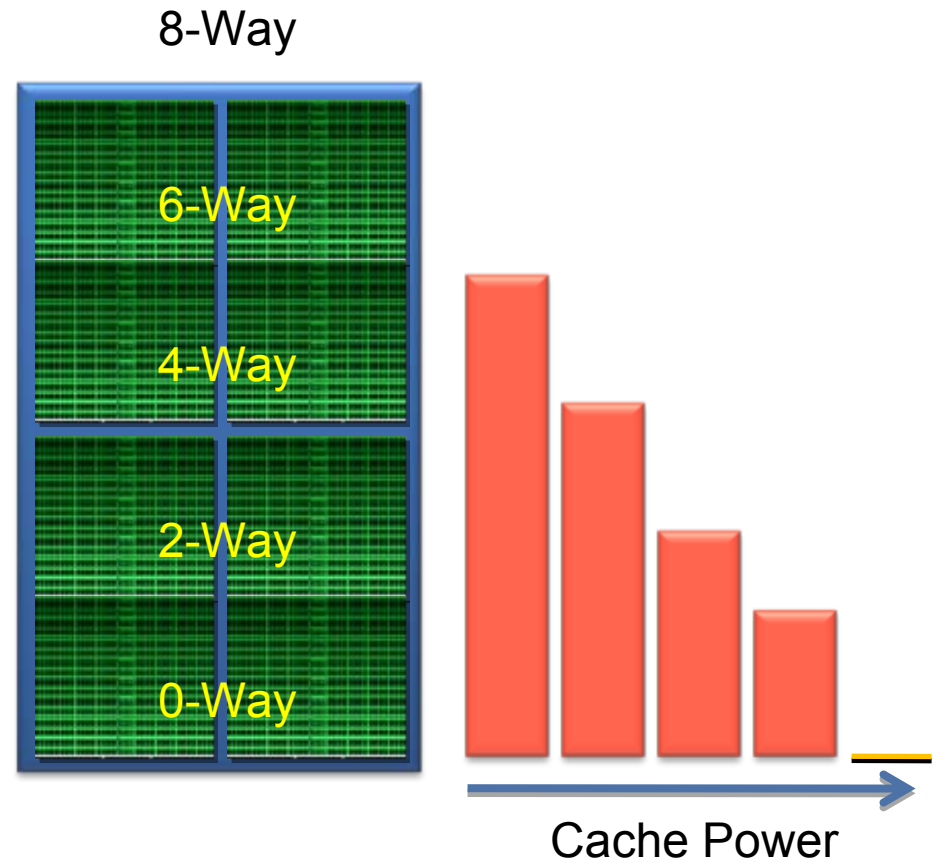
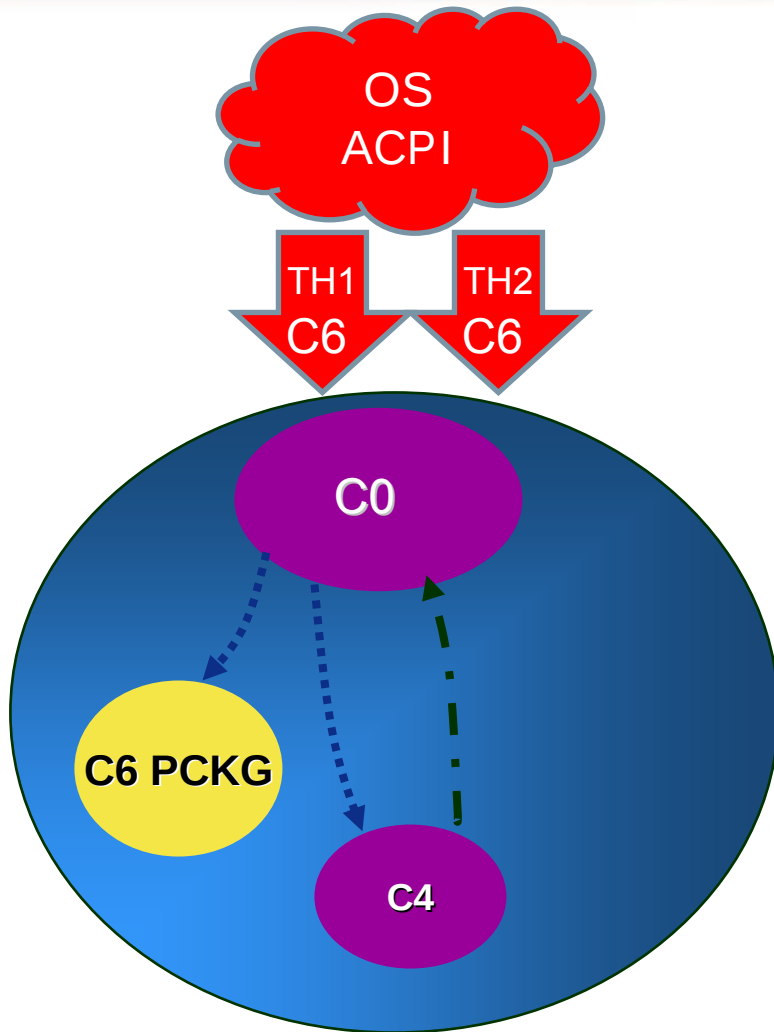


Enhanced Intel SpeedStep® Technology

- Multiple frequency and voltage operating points provide flexibility for power and performance optimization
- No chipset dependency
- Full software control of operating points to ease platform design
- Ultra-fast CPU transition time ($<10\mu\text{s}$ PLL lock time)
- Transitions may be used for efficient thermal monitoring



L2 Dynamic Cache Sizing



- 1. Enumerate system devices**
- 2. Determine degrees of power management freedom for each device**
- 3. Identify constraints**
- 4. Identify product use cases**
- 5. Define power management policies**

You'll need a full list of every power consuming device in the system

- CPU
- External interrupt controller (if any)
- Video controller
- Keyboard
- LCD
- Backlight
- Memory banks

For each device study what can be controlled to optimize power management

- Voltage: operating voltages of the CPU or peripherals
- Frequency: alter bus frequencies

Document the operating ramifications of altering each of these parameters

- Will altering the PLL frequency impose a lengthy settling time?

Are there relationships between devices in the system that will constrain the frequency and voltage settings?

- For example: Does Bluetooth require a certain bus frequency to operate at the speeds required?
- Does the bus frequency mandate a certain CPU voltage?

This is likely the most crucial step to making a successful product.

- Major operating modes: powered off, on battery, A/C powered, “fake off”, emergency power conserve
- Uses: Record video, play video, place call, play music, take picture, etc.
- Constraints: Can't play and record video; Can't play music and make a call; play video and take picture

From the user's perspective...

how you turn *on*

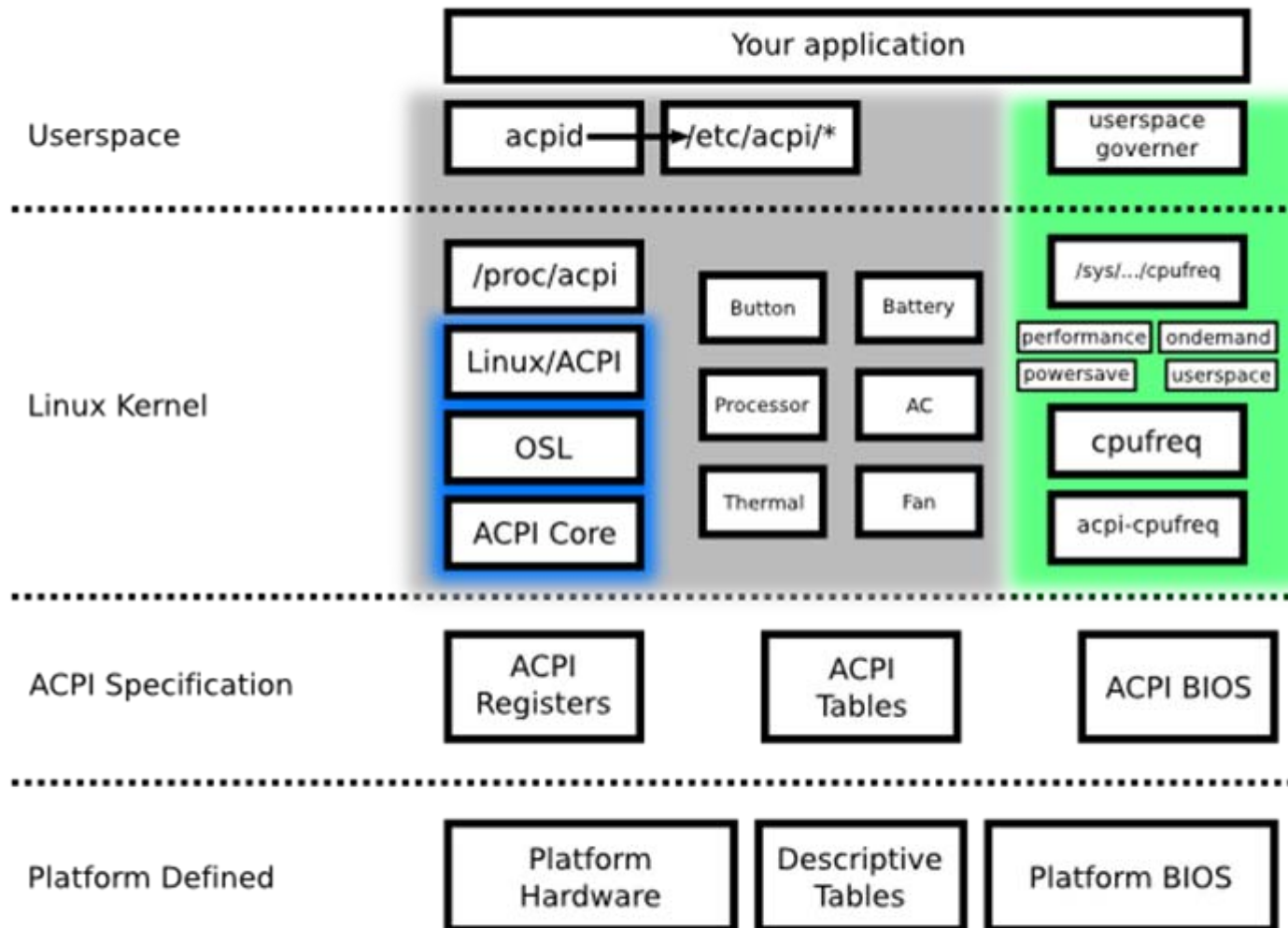


is as important as what you turn *off!*



Before we get into that we'd better learn what we can control and how we can control it

- **Saving power while the CPU is *active***
 - Voltage and frequency scaling of the CPU using cpufreq
- **Saving power while the CPU is *inactive***
 - Idle scaling
 - Dynamic tick
 - Deferrable timers
 - Mitigate wakeups using PowerTop and system tuning



- **CPUfreq**

- Create a processor driver: ✓
- Define operating points: ✓
- Modify standard drivers to respond to CPUfreq notifications: ✓
- Select and configure the governor

- **Power management aware device drivers**

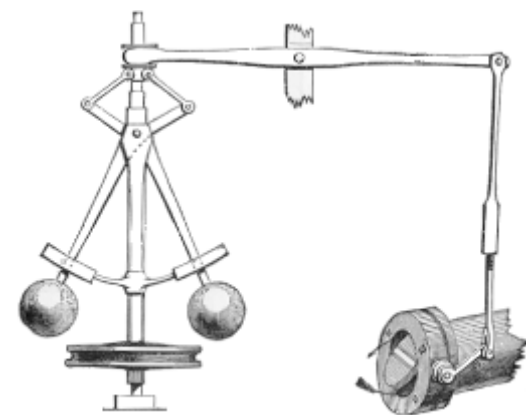
- Implementing power management in a device driver
- Handling CPUfreq notifications in a device driver
- Suspend/Resume hooks
- Clock framework

✓ = Implemented for you by MontaVista

Best to consult the kernel source code and [Documentation/cpu-freq/cpu-drivers.txt](#)

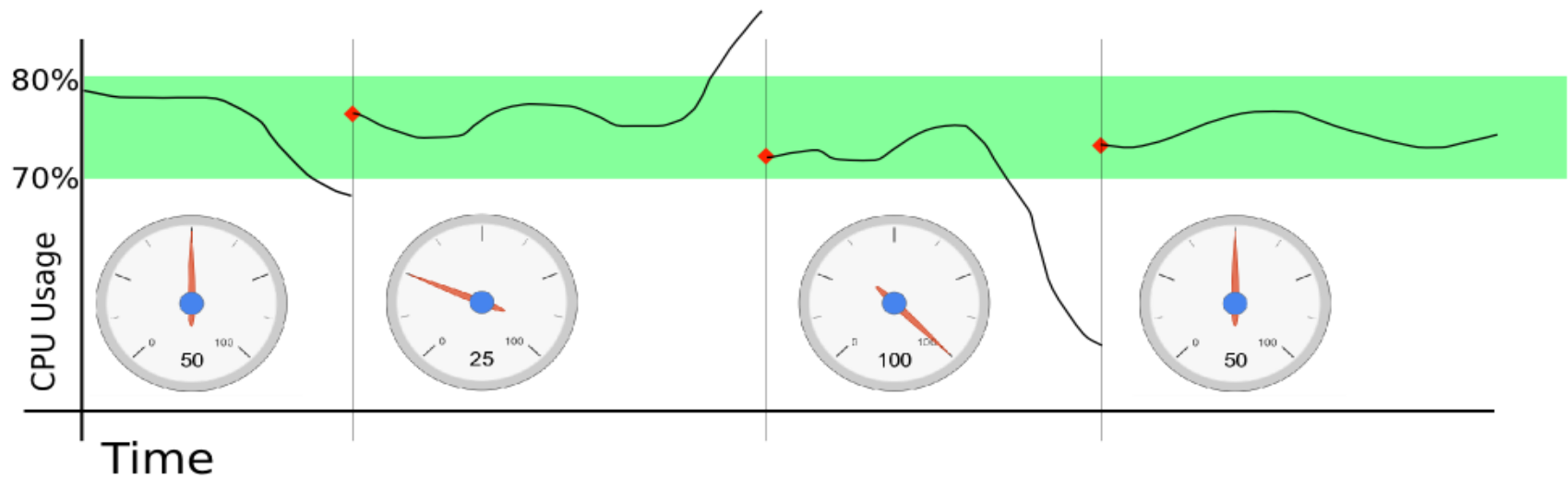
- `cpufreq_driver.name`
- `cpufreq_driver.owner`
- `cpufreq_driver.init`
- `cpufreq_driver.verify`
- `cpufreq_driver.[setpolicy|target]`
- `cpufreq_driver.exit`
- `cpufreq_driver.resume`
- `cpufreq_driver.attr`

- **Task: Decide how and when to change operating points.**
- **Four options provided:**
 - performance: statically set highest power operating point
 - powersave: statically set lowest power operating point
 - userspace: permit any application running as root to set the operating point
 - ondemand: set the operating point based on current CPU usage



- **Works by altering the operating point to minimize idle time.**
- **Lots of control knobs**
 - `sampling_rate`
 - `sampling_rate_max`
 - `sampling_rate_min`
 - `up_threshold`
 - `powersave_bias`
 - `ignore_nice_load`

ondemand Governor Example



- **Can reside in either userspace or kernelspace**
- **Interfaces via the /sys filesystem**
- **Can do whatever you wish it to do once the userspace in-kernel governor has been activated.**
- **Two other userspace utilities exist:**
 - `cpufreq-info`: returns information about cpufreq configuration
 - `cpufreq-set`: permits you to control kernel settings

Three areas to focus on:

- **Wise power management: minimizing power usage of the driver in regular operations**
 - Staying “off” between close() and open()
 - Staying “off” if the transceiver/PHY indicates no connection
 - Gating off unused clocks
 - Switching off unused power
 - Using lower voltages
- **System sleep: Preparing the driver to respond to system wide low-power sleep requests**
- **Responding to cpufreq notifications**

- Create a struct `platform_driver` in your driver
- Register the platform driver
- Implement driver specific suspend and resume functions
- Use `/sys/power/state` as a test interface

```
#include <linux/platform_device.h>

static struct platform_driver sample_driver = {
    .suspend = sample_driver_suspend,
    .resume  = sample_driver_resume,
    .driver  = {
        .owner = THIS_MODULE,
        .name  = "sample_driver",
    },
};
```

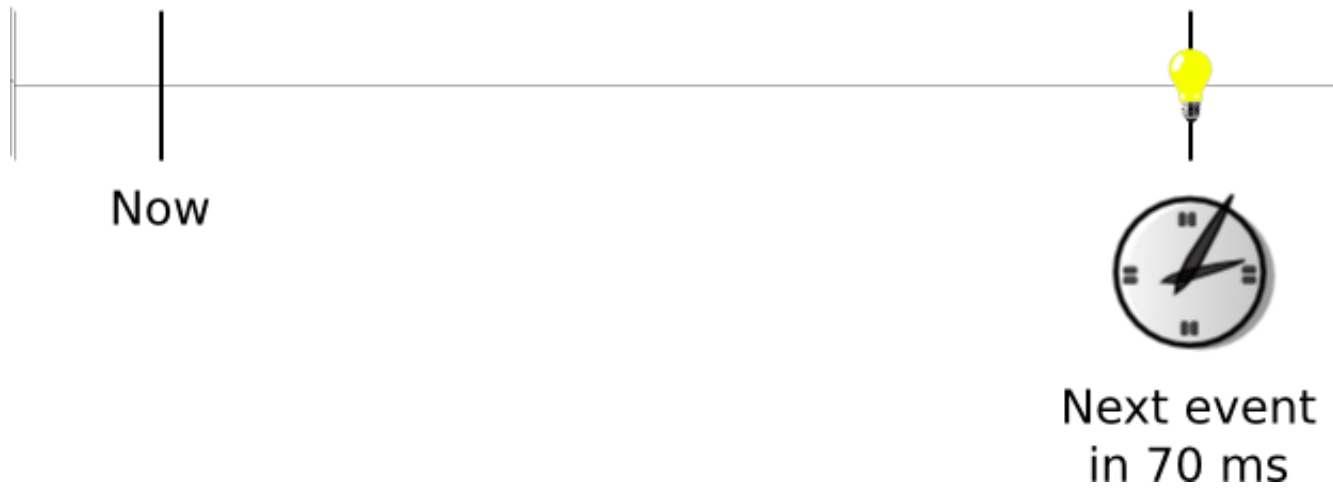
- **Your driver can register with cpufreq to get notified of power events:**
 - CPUFREQ_PRECHANGE: sent immediately before a new operating point is set
 - CPUFREQ_POSTCHANGE: sent immediately after a new operating point is set
 - CPUFREQ_RESUMECHANGE: sent if the cpufreq subsystem determines that an operating point was changed during system suspend

Before we get into that we'd better learn what we can control and how we can control it

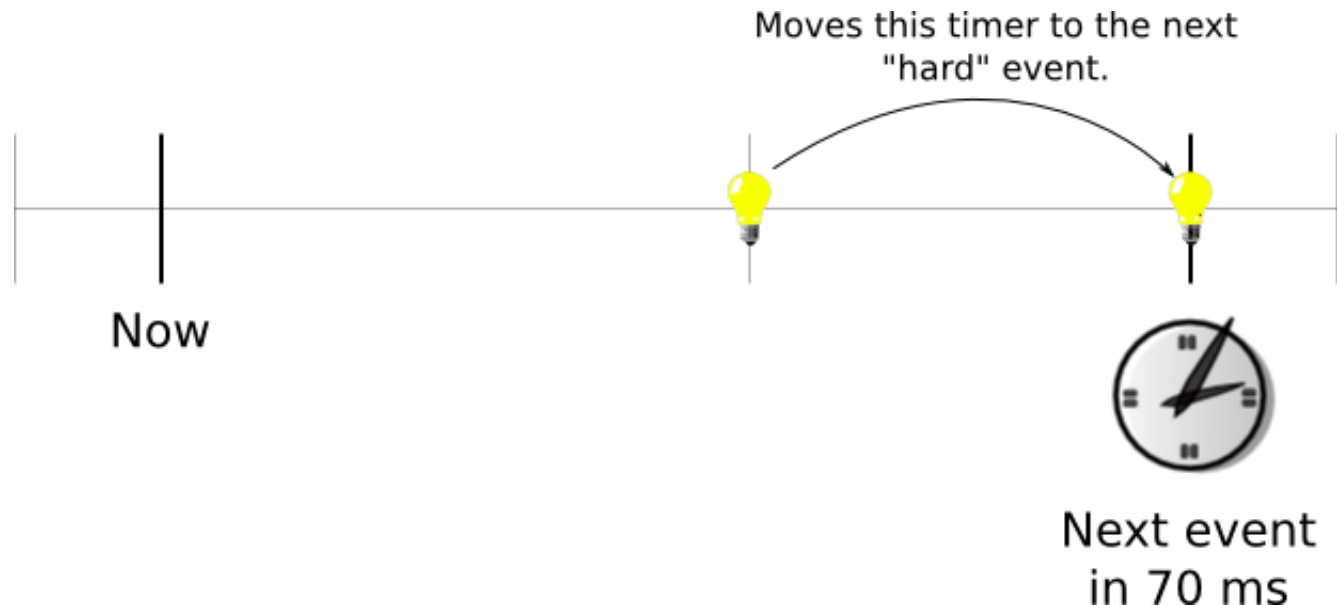
- **Saving power while the CPU is *active***
 - Voltage and frequency scaling of the CPU using cpufreq
- **Saving power while the CPU is *inactive***
 - Idle scaling
 - Dynamic tick
 - Deferrable timers
 - Mitigate wakeups using PowerTop and system tuning

- **Idle Scaling: Turning down the power consumption during idle periods**
 - If you've done cpufreq well you've got the job done already!
- **Dynamic Tick: coalescing ticks to avoid unnecessary wakeups**

the system snoozes soundly...



- API so that drivers can notify the kernel that the timer wakeup is needed but that the precise time of the wakeup is flexible.
- Use `init_timer_deferrable()`
- Example usage: flashing LED that indicates an email has arrived



Mitigate wakeups with Powertop

```
bdixon@bd-laptop: ~  
File Edit View Terminal Tabs Help  
bdixon@bd-laptop: ~ bdixon@bd-laptop: ~  
PowerTOP version 1.9 (C) 2007 Intel Corporation  
  
Cn          Avg residency      P-states (frequencies)  
C0 (cpu running) ( 1.9%)      1.60 Ghz      1.5%  
C1          0.0ms ( 0.0%)      1000 Mhz      0.0%  
C2          3.8ms (98.1%)      800 Mhz       0.0%  
C3          0.0ms ( 0.0%)      600 Mhz      98.5%  
  
Wakeups-from-idle per second : 257.9 interval: 10.0s  
no ACPI power usage estimate available  
  
Top causes for wakeups:  
75.1% (199.3)      ruby : do_nanosleep (hrtimer_wakeup)  
6.8% ( 18.0)      <interrupt> : uhci_hcd:usb1, uhci_hcd:usb2, uhci_hcd:usb3,  
4.1% ( 11.0)      <interrupt> : libata  
2.5% ( 6.6)      USB device 2-2 : SCR33x USB Smart Card Reader (SCM Microsys  
1.5% ( 4.0)      <kernel module> : usb_hcd_poll_rh_status (rh_timer_func)  
1.1% ( 2.9)      trackerd : schedule_timeout (process_timeout)  
  
Suggestion: Disable 'hal' from polling your cdrom with:  
hal-disable-polling --device /dev/cdrom 'hal' is the component that auto-opens a  
  
Q - Quit R - Refresh K - kill hald-addon-storage
```

- **Less Watts: Intel's Linux Power Management website**
 - <http://lesswatts.org>
- **Open Device, Insert Code**
 - <http://www.mvista.com/blogs/dixon>
- **Benchmarking of Dynamic Power Management Systems**

Frank Dols
CELF Embedded Linux® Conference 2007

Linked from <http://www.mvista.com/power>

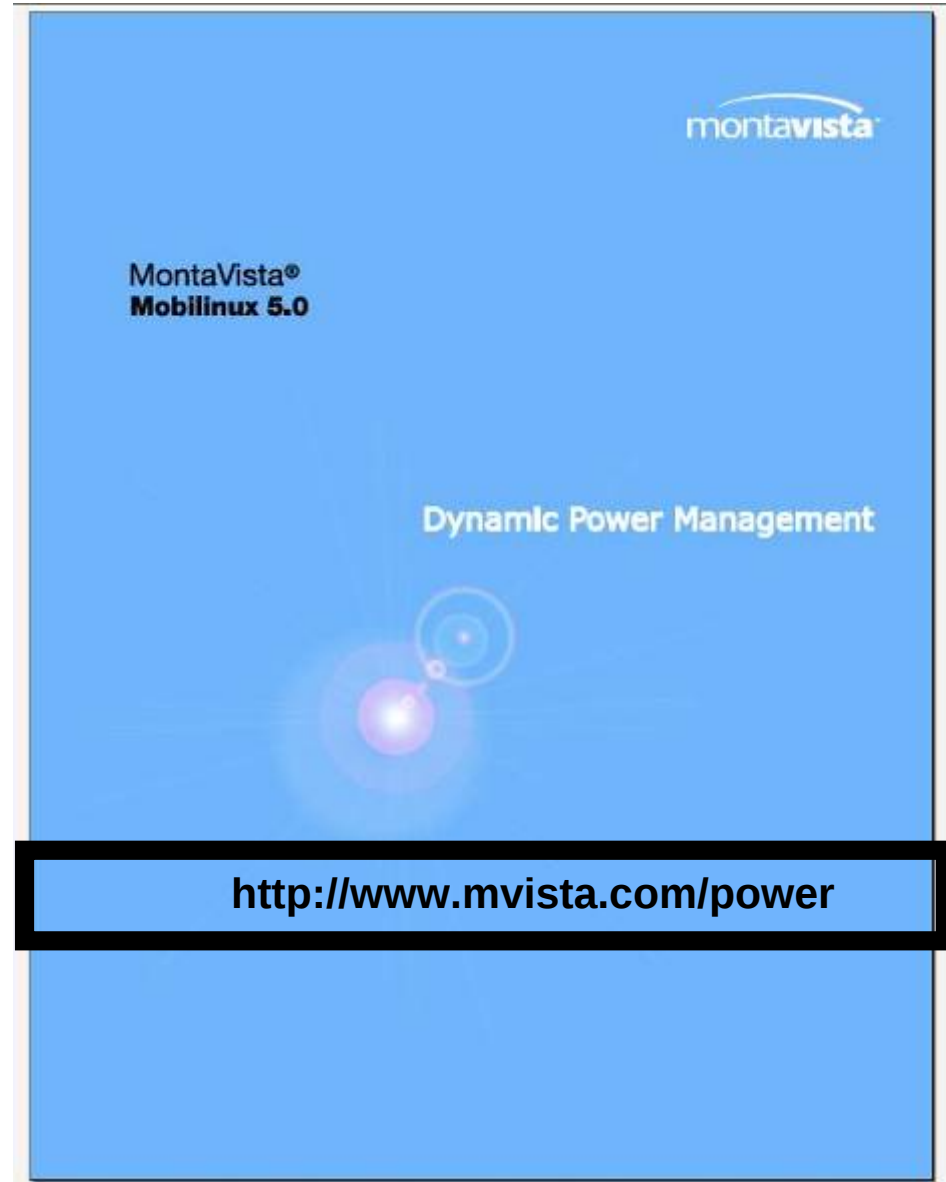
Free for the asking...

“MontaVista Mobilinux® 5.0 Dynamic Power Management”

60 pages of technical
details on power
management

Useful to all considering or
using Linux®

Sample code for power
aware device drivers and
governors





Embedded Linux Power Management on the Intel® Atom™ Processor

Brad Dixon
MontaVista Software

Kaitlin Murphy
Intel Corporation