

# Saving Power With Laptop Mode

Bart Samwel ([bart@samwel.tk](mailto:bart@samwel.tk))

---

---

# *Outline*

1. Linux laptop issues
2. Power management economics
3. Laptop Mode
4. Laptop Mode Tools



# Desktop Challenge: The Laptop

- The *laptop* is the major desktop arena challenge.
  - Specific issues for laptops:
    - Suspend to disk
      - Works OK
    - Suspend to RAM
      - YMMV
    - Hardware support
      - Often not off-the-shelf HW
      - Wireless networking
    - Power management
  - Must work well out-of-the-box.
  - Easy for Apple, hard for M\$, harder for Linux.
- 
-

# *Economics of Power Management*

- If:  
time spent optimizing  $\geq$  time gained working  
Then:  
*you don't gain a thing.*
- Again: it should work well out-of-the-box.

# What's The Gain?

- Gain per battery cycle:  
battery life increase \* fraction of cycles that battery is emptied
  - Loss per battery cycle:  
efficiency decrease \* average work time in a battery cycle +  
time spent optimizing / expected # of battery cycles
  - Example:  
Gain: 30 minutes \* 20% = 6 minutes  
Loss: 3% \* 120 minutes +  
(8\*60) / 300 = 3.6 + 1.6 = 5.2 minutes
- 
-

# Maximum Gain Per Component

- Amdahl's Law:

$$b = P / (P - p)$$

p = power used by component

P = total power used by system

b = battery life increase factor

- Potential battery life gain:

$$b * B - B$$

B = battery life with component active



# *Typical Power Distribution*

- Hard drive (idle): 1.0 W
  - Backlight (full): ~5 W (?)
  - CPU: 5 - 25 W (original Pentium M)
  - Wireless:
    - 1.0 – 2.0 W (powersave off)
    - 0.2 – 1.0 W (powersave on)
    - 3 – 4 W (sending/receiving)
  - RAM: negligible. 4 hour operating battery life / 12 days standby battery life, so  $< 1.3\%$  of power.
  - Total: 10 – 40 W
  - YMMV!
- 
-

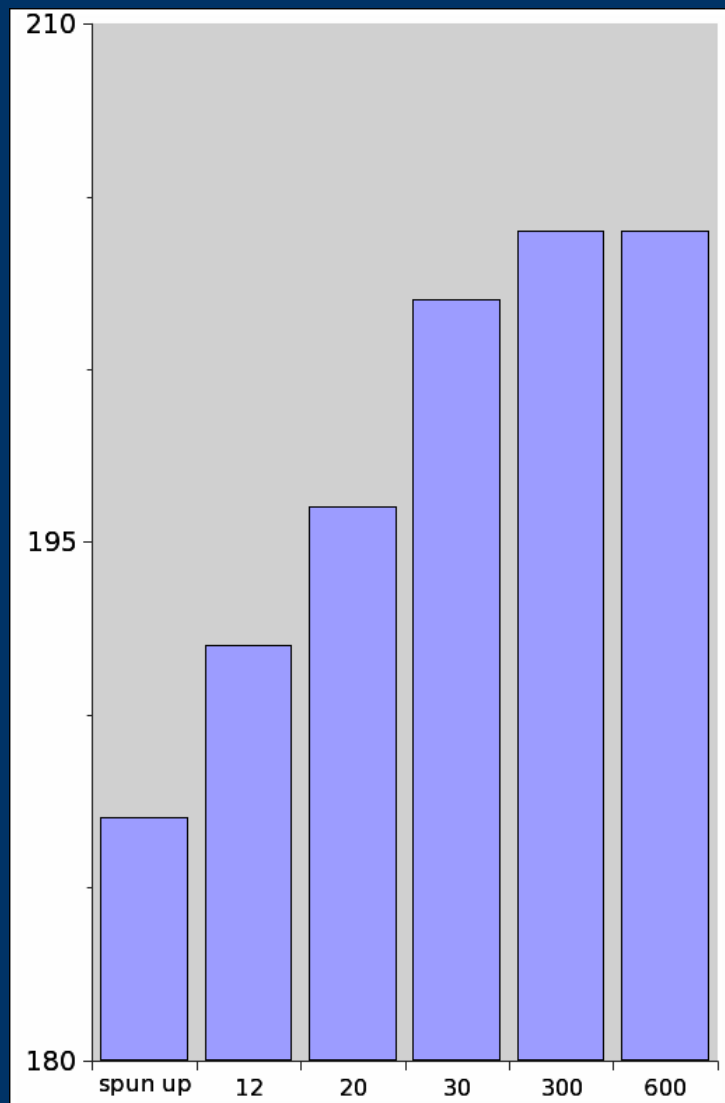
# *Maximum Gains (by Amdahl's Law)*

- Backlight:  $120 * 35 / 30 - 120 = 20$  minutes
    - Realistic? No.
  - CPU:  $120 * 35 / 15 - 120 = 160$  minutes
  - Backlight:  $280 * 15 / 10 - 280 = 140$  minutes
    - Gain for every optimization grows with other optimizations!
    - Realistic? No, but dimming helps much more now!
  - Hard drive:  $280 * 15 / 14 - 280 = 20$  minutes
    - => Laptop Mode.
- 
-

# Laptop Mode

- Laptop Mode = kernel feature to make disk accesses “bursty”.
- Included in Linux  $\geq 2.4.23$  and  $\geq 2.6.6$ .
- Flag: `/proc/sys/vm/laptop_mode`
- Behaviour: automatically “sync” after a couple of seconds when I/O becomes idle
- Various tweaks needed to make this work:
  - `/proc/sys/vm/dirty_expire_centisecs`
  - `/proc/sys/vm/dirty_writeback_centisecs`
  - `commit=N` (ext3, ReiserFS)
  - Similar kernel parameters for XFS
  - Set spindown timeout on hard drives using `hdparm`

# Benchmark Results



- $X$  = time between spinups
- $Y$  = battery life
- Average I/O rate / min held constant
- Linux Journal 9/2004
- Hardware:
  - PowerBook G4 (new)
  - 5400 rpm disk
- Observe:
  - Gain 16 mins at most
  - Max gain at 5 mins spindown time
  - $>5$  useful for noise!

# Laptop Mode Tools

- Once: a simple control script for *laptop mode*.
  - Now: a general power saving tool featuring:
    - Laptop Mode control
    - Disk access profiler (`lm-profiler`)
    - HD idle timeout, readahead, power mgmt, write cache control with IDE, SATA and SCSI support.
    - Automatically start/stop programs when on battery
    - Frequency scaling control (min, max, scaling governor), CPU throttling control.
    - Automatic `syslog.conf` swizzling (to avoid syncs)
    - X and terminal blanking settings control
    - Automatic hibernation on low battery
- 
-

# Modes in LMT

- Orthogonal modes:
    - On AC / on battery
    - Laptop mode active/inactive  
(Should have been called “Powersave mode”, has nothing to do with the kernel's *laptop mode* feature.)
  - Everything can be configured for combinations of these two modes that make sense.
  - “Laptop mode” can be enabled based on AC/battery state, lid state, etc.
  - Battery low: data loss sensitive features disabled.
    - Delayed disk writes
    - HD write cache
- 
-

# Starting/stopping programs

- The idea: a power mode is a *runlevel patch*. A *stop* and a *start* directory for every mode, e.g.:  
    /etc/laptop-mode/lm-ac-stop  
    /etc/laptop-mode/lm-ac-start
  - “Stop” scripts: called with “stop” argument when entering mode, with “start” when leaving. “Start” scripts similarly, with “start” / “stop”.
  - Init scripts can be linked directly into these directories:  
    cron -> /etc/rc.d/init.d/cron  
    squid -> /etc/rc.d/init.d/squid
- 
-

# *The Laptop Mode Profiler*

- Uses kernel flag `/proc/sys/vm/block_dump`, which reports disk I/O and inode dirtying in `dmesg` output.
  - Filters kernel output for *direct* writes and reads (that cause disk spinups). => Works with LM off!
  - Records names of programs that access disk.
  - Uses `netstat` to find listening network daemons.
  - Finds init scripts corresponding to recorded disk accessing and network listening programs.
  - Links these init scripts into the batt-stop directory.
- 
-

# Swizzling Syslog Settings

- Syslogd syncs after writing log entry:

```
auth.*    /var/log/auth.log
```

- Disabled like this:

```
auth.*    -/var/log/auth.log
```

- But:

- You want auth.log to sync when connected.
- You don't want this when you're on the road.

- LMT can automatically link `/etc/syslog.conf` to a specific `syslog.conf` depending on power state.
  - Tool `lm-syslog-setup` automatically sets this up.
- 
-

# *LMT Hardware Support*

- Laptop mode tools supports ACPI best.
    - Prevent data loss when battery low
    - Enabling when lid closed
    - Automatic hibernation when battery low
    - CPU frequency scaling / throttling
  - Apple hardware supported:
    - `pbbuttonsd`: automatically started on unplug from AC
    - `pmud`: automatic start on AC unplug configurable
  - APM support is limited.
    - Can detect power state, but not started automatically.
    - No support for battery level events.
- 
-

# *The Future*

- Support other power saving tricks:
    - Wifi powersave mode on/off
    - Network interfaces up/down
    - Backlight level adjustment (hardware specific)
  - Modularize non-core functionality using program start/stop architecture.
  - Improve Apple hardware support.
- 
-

Questions?

