Components for Energy-Efficient Operating Systems

Seminar “Selected Chapters of System Software Techniques: Energy-aware Systems”

Clemens Lang

May 16, 2013

Motivation

Why?

- Battery technology stagnates
- CPUs and devices offer more and better power savings mechanisms

Motivation

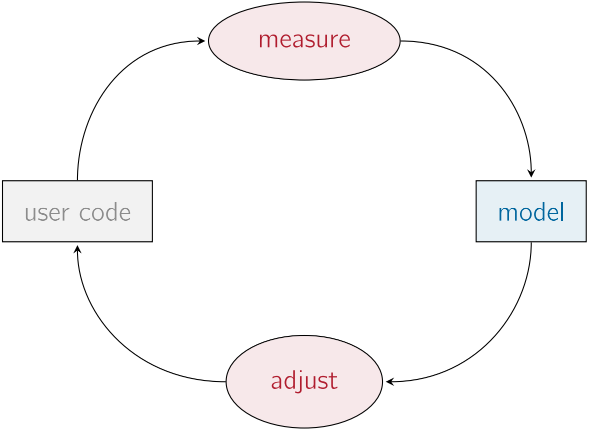
Why?

- Battery technology stagnates
- CPUs and devices offer more and better power savings mechanisms

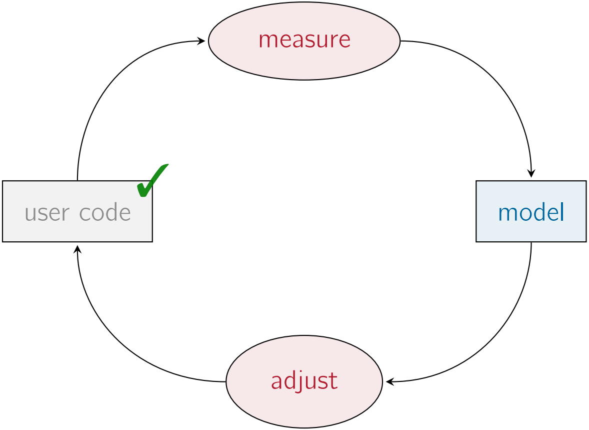
Question

How can operating systems be designed to efficiently use those mechanisms?

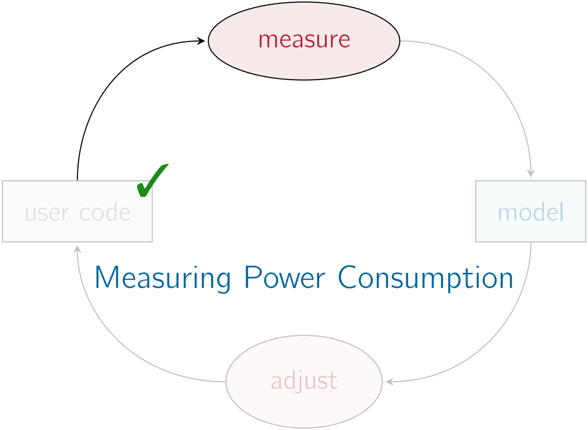
Outline



Outline



Outline



Measuring Power Consumption

Measuring Power Consumption

- How is power used?
  - **Static power consumption**: power dissipation
  - **Dynamic power consumption**: transistor switching

Measuring Power Consumption

- How is power used?
  - **Static power consumption**: power dissipation
  - **Dynamic power consumption**: transistor switching
- Can we influence static power usage?
  - If we can’t change it, do we still have to model it?
  - **Yes:** dynamic voltage scaling, factor in race-to-halt decisions

Identifying Key Power Consumers

Where is power dynamically used?

- **CPU**
  - High switching frequency
  - Different power usage characteristics depending on instructions executed

Identifying Key Power Consumers

Where is power dynamically used?

- **CPU**
  - High switching frequency
  - Different power usage characteristics depending on instructions executed
- **MMU & Caches**
  - Caches use a lot of energy
  - MMU contains caches (e.g., the TLB)
  - Power usage depending on access patterns

Identifying Key Power Consumers

Where is power dynamically used?

- **CPU**
  - High switching frequency
  - Different power usage characteristics depending on instructions executed
- **MMU & Caches**
  - Caches use a lot of energy
  - MMU contains caches (e.g., the TLB)
  - Power usage depending on access patterns
- **DRAM**
  - Periodic refresh ($\rightarrow$ static power usage)
  - Complex access electronics
  - Power usage depending on access patterns

Identifying Key Power Consumers

Where is power dynamically used?

- **CPU**
  - High switching frequency
  - Different power usage characteristics depending on instructions executed
- **MMU & Caches**
  - Caches use a lot of energy
  - MMU contains caches (e.g., the TLB)
  - Power usage depending on access patterns
- **DRAM**
  - Periodic refresh ($\rightarrow$ static power usage)
  - Complex access electronics
  - Power usage depending on access patterns
- **Devices**

  Not covered in this talk

Measuring Dynamic Power Consumption

- How can dynamic power consumption be **measured**?
  - Current measurement equipment is not available in off-the-shelf systems
    $\Rightarrow$ Available for calibration, but not when deployed
  - What tools are available **at runtime** to gauge power usage?

Measuring Dynamic Power Consumption

- How can dynamic power consumption be **measured**?
  - Current measurement equipment is not available in off-the-shelf systems
    ⇒ Available for calibration, but not when deployed
  - What tools are available **at runtime** to gauge power usage?
- Solution: Estimate power usage using **event counters**
  - Hardware counters for events (cache miss, cycle count, memory access, . . .)
  - Traditionally used for performance analysis
  - Problem: hundreds of countable events, but only a handful of counters
    ⇒ How can the ideal subset be chosen?

Measuring Dynamic Power Consumption

- How can dynamic power consumption be **measured**?
  - Current measurement equipment is not available in off-the-shelf systems
    $\Rightarrow$ Available for calibration, but not when deployed
  - What tools are available **at runtime** to gauge power usage?
- Solution: Estimate power usage using **event counters**
  - Hardware counters for events (cache miss, cycle count, memory access, . . .)
  - Traditionally used for performance analysis
  - Problem: hundreds of countable events, but only a handful of counters
    $\Rightarrow$ How can the ideal subset be chosen?
- Choosing subset of events
  - Run series of benchmarks with known behavior at all power saving configurations
  - Measure power consumption using dedicated hardware
  - Choose events correlating with power usage
  - **Note: hardware-specific!**

Maximizing Energy Efficiency: A Naïve Approach

$$\text{minimize} \quad \frac{\text{energy}}{\text{performance}} \left(= \frac{\text{power usage} \cdot \text{time}}{\text{time}^{-1}} = \text{power usage} \cdot \text{time}^2 \right)$$

Maximizing Energy Efficiency: A Naïve Approach

$$\text{minimize} \quad \frac{\text{energy}}{\text{performance}} \left(= \frac{\text{power usage} \cdot \text{time}}{\text{time}^{-1}} = \text{power usage} \cdot \text{time}^2 \right)$$

- Efficiency for
  - **CPU-bound** tasks: only little difference
  - **Memory-bound** tasks: higher efficiency at low speeds

Maximizing Energy Efficiency: A Naïve Approach

$$\text{minimize} \quad \frac{\text{energy}}{\text{performance}} \left(= \frac{\text{power usage} \cdot \text{time}}{\text{time}^{-1}} = \text{power usage} \cdot \text{time}^2 \right)$$

- Efficiency for
  - **CPU-bound** tasks: only little difference
  - **Memory-bound** tasks: higher efficiency at low speeds
- ⇒ run CPU-bound tasks at highest, memory-bound tasks at lowest speed
  - Low speeds significantly reduce performance
  - Users expect fast systems
  - **There is no free lunch:** performance vs. energy is a trade-off

No Free Lunch

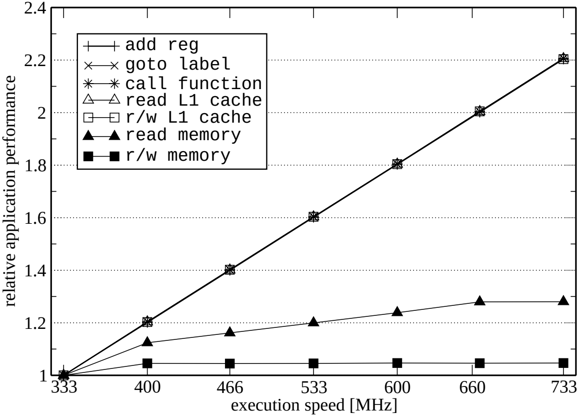
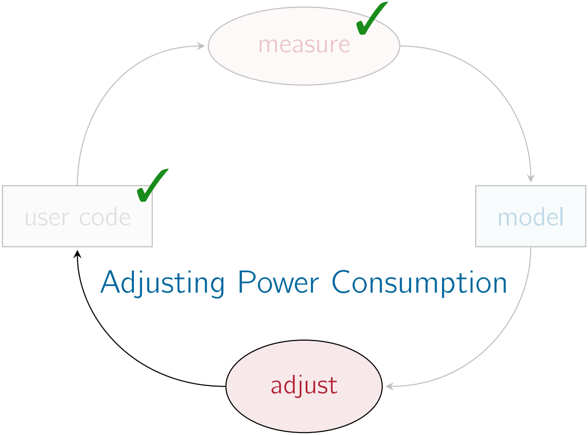


Figure: Normalized performance at different clock speeds. From [WB02].

Outline



Adjusting Power Consumption

Adjusting Power Consumption

- **Dynamic frequency scaling**
  - Adjust core frequency in discrete steps at run-time
  - Triggered by writing into hardware-specific register

Adjusting Power Consumption

- **Dynamic frequency scaling**
  - Adjust core frequency in discrete steps at run-time
  - Triggered by writing into hardware-specific register
- **Dynamic voltage scaling**
  - Similar to DFS, but for voltage
  - Lower voltages are only available at lower clock speeds
    $\Rightarrow$ Used together with DFS as DVFS
  - DVS **affects static power consumption**
  - $E \propto V^2 \Rightarrow$ high impact!

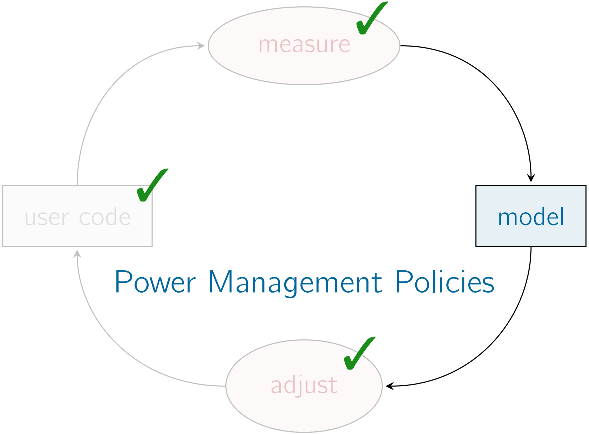
Adjusting Power Consumption (cont'd)

- **Sleep states** (C-states)
  - $\mathrm{C}0, \mathrm{C}1, \ldots, \mathrm{C}3$, more depending on hardware
  - Higher number: lower energy usage
  - $\mathrm{C}0$: executing instructions
  - $\mathrm{C}1$: `hlt`
  - $\mathrm{C}n, n > 1$: turn off features (e.g., caches and cache coherence) to save power

Adjusting Power Consumption (cont'd)

- **Sleep states** (C-states)
  - $\mathrm{C}0, \mathrm{C}1, \ldots, \mathrm{C}3$, more depending on hardware
  - Higher number: lower energy usage
  - $\mathrm{C}0$: executing instructions
  - $\mathrm{C}1$: `hlt`
  - $\mathrm{C}n, n > 1$: turn off features (e.g., caches and cache coherence) to save power
- **Switching overhead**
  - Switching to and from a power saving configuration takes significant time
  - Rule of thumb: **higher savings** $\Leftrightarrow$ **higher switching time**
  - Prediction problem: Will switching save energy?

Outline



Managing Power: Policies

- Event counters span multidimensional space
  - Optimization methods **find optimal configuration for each point**
  - Changing the objective function (and the constraints) yields different **policies**

Managing Power: Policies

- Event counters span multidimensional space
  - Optimization methods **find optimal configuration for each point**
  - Changing the objective function (and the constraints) yields different **policies**
- **Maximum degradation** policy
  - minimize P subject to $pT \leq T_{\text{opt}}$
  - i.e., **minimize power consumption** P,
    but only **up to a performance loss** of $(1 - p)$ %
  - Weißel et al.: $p = 0.9$ works well, up to 37 % saved

Managing Power: Policies (cont'd)

- **Generalized energy-delay** policy
  - minimize $P^{1-\alpha} \cdot T^{1+\alpha}, \alpha \in [-1; 1]$
  -

α	policy behavior
1	maximum performance, race-to-halt
0	minimize energy usage (remember $E := \int_T P = \bar{P}T$)
-1	minimize power consumption
$0 < \alpha < 1$	throttle depending on the workload

  - Snowdon et al.: up to 30 % saved for a 4 % performance loss

Managing Power: Policies (cont'd)

- **Generalized energy-delay** policy
  - minimize $P^{1-\alpha} \cdot T^{1+\alpha}, \alpha \in [-1; 1]$
  -

α	policy behavior
1	maximum performance, race-to-halt
0	minimize energy usage (remember $E := \int_T P = \bar{P}T$)
-1	minimize power consumption
$0 < \alpha < 1$	throttle depending on the workload

  - Snowdon et al.: up to 30 % saved for a 4 % performance loss
- Adjustable policies
  - Note the parameters!
  - **User experience matters**, user-adjustable policies help

Generalized Energy Delay

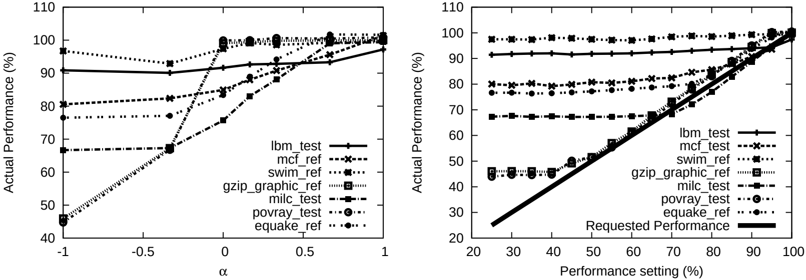


Figure: Generalized energy-delay policy. From [SLSPH09].

Challenges: Is It Really That Simple?

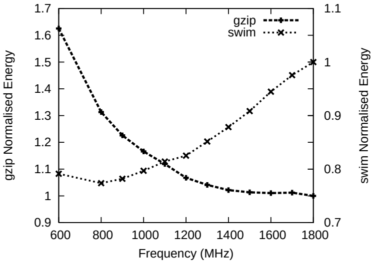


Figure: Normalized energy consumption of two benchmarks. From [SLSPH09].

- **Quality of workload prediction**
  - Bad analysis → wrong power saving decision
  - Bad prediction → sleep state overhead

Challenges: Is It Really That Simple?

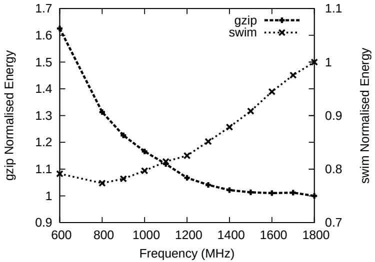


Figure: Normalized energy consumption of two benchmarks. From [SLSPH09].

- **Quality of workload prediction**
  - Bad analysis → wrong power saving decision
  - Bad prediction → sleep state overhead
- **Multiple and dependent variables**
  - Multiple adjustable values → more test data required
  - Snowdon et al.: memory performance depends on CPU frequency
  - Not all effects are measurable using event counters

Challenges: Is It Really That Simple? (cont'd)

- **Race-to-halt or** run at **lower frequency?**

Challenges: Is It Really That Simple? (cont'd)

- **Race-to-halt or** run at **lower frequency?**
- Switching overhead
  - **Switch** to higher C-state **or wait?**
  - Run at suboptimal frequency/voltage or switch?

Challenges: Is It Really That Simple? (cont'd)

- **Race-to-halt or** run at **lower frequency?**
- Switching overhead
  - **Switch** to higher C-state **or wait?**
  - Run at suboptimal frequency/voltage or switch?
- Power-supply efficiency and temperature

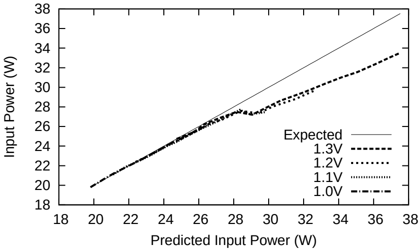


Figure: Actual vs. predicted input power of a Dell Latitude D600. From [SLSPH09].

  - Power-supply efficiency doesn't necessarily scale linearly
  - Influence of temperature (on efficiency, power required for cooling)

Notes on Implementation

- Predict behavior **per process**
  - Simpler prediction of behavior
  - Needs modifications in
    - dispatcher
    - process control block
  - **Events keep counting** in interrupts/during task switch

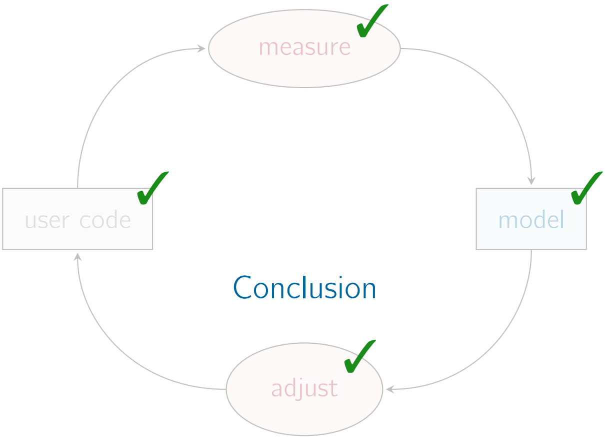
Notes on Implementation

- Predict behavior **per process**
  - Simpler prediction of behavior
  - Needs modifications in
    - dispatcher
    - process control block
  - **Events keep counting** in interrupts/during task switch
- **Avoiding overhead** is crucial
  - Reformulate to avoid floating point operations
  - Pre-compute lookup tables
  - Favor simple decision rules

Notes on Implementation

- Predict behavior **per process**
  - Simpler prediction of behavior
  - Needs modifications in
    - dispatcher
    - process control block
  - **Events keep counting** in interrupts/during task switch
- **Avoiding overhead** is crucial
  - Reformulate to avoid floating point operations
  - Pre-compute lookup tables
  - Favor simple decision rules
- Snowdon et al. implemented *Koala* for Linux 2.6.24.4

Outline



Conclusion

- Power Management
  - is **heuristic**
  - is **predictive**
  - involves **hardware-specific**s

Conclusion

- Power Management
  - is **heuristic**
  - is **predictive**
  - involves **hardware-specific**s
- There is no free lunch: **Performance** ↔ **Energy**

Conclusion

- Power Management
  - is **heuristic**
  - is **predictive**
  - involves **hardware-specific**s
- There is no free lunch: **Performance** $\leftrightarrow$ **Energy**
- Manufacturers also providing the OS are at advantage

Conclusion

- Power Management
  - is **heuristic**
  - is **predictive**
  - involves **hardware-specific**s
- There is no free lunch: **Performance** $\leftrightarrow$ **Energy**
- Manufacturers also providing the OS are at advantage
- **Lessons learned:** write predictable applications

Q&A Session

Questions & Answers

Thank you for your attention.

References

David C. Snowdon, Etienne Le Sueur, Stefan M. Petters, and Gernot Heiser.
Koala: a platform for os-level power management.
In *Proceedings of the 4th ACM European conference on Computer systems*, EuroSys '09, pages 289–302, New York, NY, USA, 2009. ACM.

Andreas Weißel and Frank Bellosa.
Process cruise control: event-driven clock scaling for dynamic power management.
In *Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 238–246. ACM, 2002.