Computer Power Management Rules

- Jim Kardach, retired chief power architect, Intel
  http://www.youtube.com/watch?v=cZ6akewB0ps
HW1

- Has been posted on the online schedule.
- Due on March 3\textsuperscript{rd}, 1pm.
- Submit in class.
- **Hard deadline**: no homework accepted after deadline.
- **No collaboration** is allowed.
The Power Problem

- Processors improve performance at the cost of power.
  - Performance/watt remains low.

- Solution
  - Hardware offer mechanisms for saving power.
  - Software executes power management policies.
Power vs. Energy

- **Power:** Energy consumed per unit time
  - 1 watt = 1 joule/second
- **Power → heat**
- **Energy → battery life**
Why worry about energy?
Intel vs. Duracell

No Moore’s Law in batteries: 2-3%/year growth.
Trend in Power Density

- Hot plate
- i386
- i486
- Pentium®
- Pentium® Pro
- Pentium® II
- Pentium® III
- Pentium® 4
- Nuclear Reactor
- Rocket Nozzle
- Sun's Surface

Trend in Cooling Solution

The graph shows the trend in cooling solution costs based on thermal dissipation. As the thermal dissipation increases, the cost of the cooling solution also increases significantly. The cost is measured in dollars, and the thermal dissipation is measured in watts. The graph indicates a direct correlation between these two variables, with higher thermal dissipation leading to higher cooling solution costs.
Power

- Hardware support
- Power management policy
- Power manager
- Holistic approach
CMOS Power Consumption

- **Voltage drops**: power consumption $\propto V^2$.
- **Toggling**: more activity $\Rightarrow$ higher power.
- **Leakage** when inactive.
Power-Saving Features

Voltage drops
- Reduce power supply voltage.

Toggling
- Run at lower clock frequency.
- Reduce activity.
- Disable function units when not in use.

Leakage
- Disconnect parts from power supply when not in use.
Dynamic Voltage Scaling

- **Why voltage scaling?**
  - Power $\propto V^2$ $\rightarrow$ reduce power supply voltage saves energy.
  - Lower voltage $\rightarrow$ lower clock frequency.
    - Tradeoff between performance vs. energy.

- **Why dynamic?**
  - Peak computing demand is much higher than average.

- Changing voltage takes time
  - to stabilize power supply and clock
Examples

- **StrongARM SA-1100 takes two supplies**
  - VDD is main 3.3V supply.
  - VDDX is 1.5V.

- **AMD K6-2+**
  - 8 frequencies: 200-600 MHz.
  - Voltage: 1.4, 2.0 V.
  - Transition time: 0.4 ms for voltage change.

- **PowerPC 603**
  - Can shut down unused execution units.
  - Cache organized into subarrays to reduce active circuitry.
Intel SpeedStep

<table>
<thead>
<tr>
<th>Frequency</th>
<th>VID</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6GHz</td>
<td>29</td>
<td>1.1875V</td>
</tr>
<tr>
<td>1.87GHz</td>
<td>34</td>
<td>1.25V</td>
</tr>
<tr>
<td>2.13GHz</td>
<td>38</td>
<td>1.3V</td>
</tr>
<tr>
<td>2.4GHz</td>
<td>42</td>
<td>1.35V</td>
</tr>
</tbody>
</table>

Intel Core 2 Duo E6600

<table>
<thead>
<tr>
<th>P-State</th>
<th>Frequency</th>
<th>Voltage</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>1.6 GHz</td>
<td>1.484 V</td>
<td>25 Watts</td>
</tr>
<tr>
<td>P1</td>
<td>1.4 GHz</td>
<td>1.420 V</td>
<td>~17 Watts</td>
</tr>
<tr>
<td>P2</td>
<td>1.2 GHz</td>
<td>1.276 V</td>
<td>~13 Watts</td>
</tr>
<tr>
<td>P3</td>
<td>1.0 GHz</td>
<td>1.164 V</td>
<td>~10 Watts</td>
</tr>
<tr>
<td>P4</td>
<td>800 MHz</td>
<td>1.036 V</td>
<td>~8 Watts</td>
</tr>
<tr>
<td>P5</td>
<td>600 MHz</td>
<td>0.956 V</td>
<td>6 Watts</td>
</tr>
</tbody>
</table>

Intel Pentium M P states
Linux DVFS Governors

- **Performance**
  - Always set at the max frequency

- **Powersave**
  - Always set at the lowest frequency

- **Ondemand**
  - Automatically adjust the frequency according to CPU usage

- **Conservative**
  - Like ondemand, but in a more conservative way.

- **Userspace**
  - Set at a fixed frequency by the user
Ondemand

- Initial implementation in 2.6.9
- For all CPUs
  - if (> 80% busy) then P0 (max frequency)
  - if (< 20% busy) then down by 20%
- Multiple improvements since 2.6.9
Get & Set CPU Frequency

Get the current frequency:
- `/sys/devices/system/cpu/cpu[X]/cpufreq/scaling_cur_freq`
- Example: 2400000 (2.4GHz)

Frequency & governors available:
- `/sys/devices/system/cpu/cpu[X]/cpufreq/scaling_available_frequencies`
- Example: 2400000 2133000 1867000 1600000
- `/sys/devices/system/cpu/cpu[X]/cpufreq/scaling_available_governor`
- Example: ondemand userspace performance powersave conservative

Set the frequency:
- Root privilege
- `echo userspace > /sys/devices/system/cpu/cpu[X]/cpufreq/scaling_governor`
- `echo 2133000 > /sys/devices/system/cpu/cpu[X]/cpufreq/scaling_setspeed`
Clock Gating

- Applicable to *clocked* digital components
  - Processors, controllers, memories
- Stop clock ➔ stop signal propagation in circuits

- Short transition time
  - Clock generation is not stopped
  - Only clock distribution is stopped

- Relatively high power consumption
  - Clock itself still consumes energy
  - Cannot prevent power leaking
Supply Shutdown

- Disconnect parts from power supply when not in use.

✔ General
✔ Save most power

✘ Long transition time
Example: SA-1100

Three power modes:

- Run: normal operation.
- Idle: stops CPU clock, w. I/O logic still powered.
- Sleep: shuts off most of chip activity
SA-1100 SLEEP

▼ RUN → SLEEP
- (30 µs) Flush to memory CPU states (registers)
- (30 µs) Reset processor state and wakeup event
- (30 µs) Shut down clock

▼ SLEEP → RUN
- (10 ms) Ramp up power supply
- (150 ms) Stabilize clock
- (negligible) CPU boot
## Intel Core Duo Processor SV

<table>
<thead>
<tr>
<th>Name</th>
<th>Vcc</th>
<th>Watt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C0</strong> High Frequency Mode (P0)</td>
<td>1.3</td>
<td>31</td>
</tr>
<tr>
<td><strong>C0</strong> Low Frequency Mode (Pn)</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td><strong>C1</strong> Auto Halt Stop Grant (HFM)</td>
<td></td>
<td>15.8</td>
</tr>
<tr>
<td><strong>C1E</strong> Enhanced Halt (LFM)</td>
<td></td>
<td>4.8</td>
</tr>
<tr>
<td><strong>C2</strong> Stop Clock (HFM)</td>
<td></td>
<td>15.5</td>
</tr>
<tr>
<td><strong>C2E</strong> Enhanced Stop Clock (LFM)</td>
<td></td>
<td>4.7</td>
</tr>
<tr>
<td><strong>C3</strong> Deep Sleep (HFM)</td>
<td></td>
<td>10.5</td>
</tr>
<tr>
<td><strong>C3E</strong> Enhanced Deep Sleep (LFM)</td>
<td></td>
<td>3.4</td>
</tr>
<tr>
<td><strong>C4</strong> Intel Deeper Sleep</td>
<td>0.85</td>
<td>2.2</td>
</tr>
<tr>
<td><strong>DC4</strong> Intel Enhanced Deeper Sleep</td>
<td>0.80</td>
<td>1.8</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Microcontroller</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>AT90LS8535</td>
<td>ATmega163</td>
</tr>
<tr>
<td>Program memory (KB)</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>RAM (KB)</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>Active Power (mW)</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Sleep Power (μW)</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>Wakeup Time (μs)</td>
<td>1000</td>
<td>36</td>
</tr>
<tr>
<td>Nonvolatile storage</td>
<td>24LC256</td>
<td>AT45DB041B</td>
</tr>
<tr>
<td>Chip</td>
<td>I²C</td>
<td>SPI</td>
</tr>
<tr>
<td>Connection type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size (KB)</td>
<td>32</td>
<td>512</td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radio</td>
<td>TR1000</td>
<td>TR1000</td>
</tr>
<tr>
<td>Data rate (kbps)</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>Modulation type</td>
<td>OOK</td>
<td>ASK</td>
</tr>
<tr>
<td>Receive Power (mW)</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>Transmit Power at 0dBm (mW)</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Power Consumption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum Operation (V)</td>
<td>2.7</td>
<td>2.7</td>
</tr>
<tr>
<td>Total Active Power (mW)</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>Programming and Sensor Interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expansion</td>
<td>none</td>
<td>51-pin</td>
</tr>
<tr>
<td>Communication</td>
<td>IEEE 1284 (programming) and RS232 (requires additional hardware)</td>
<td>19-pin</td>
</tr>
<tr>
<td>Integrated Sensors</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Power Consumption
Computer with Wireless NIC

- Display: 36%
- CPU/ Memory: 21%
- Hard Drive: 18%
- Wireless LAN: 18%
- Other: 7%
Power

- Hardware support
- Power management policy
- Power manager
- Holistic approach
Approaches

- **Static Power Management**
  - Does not depend on activity.
  - Example: user-activated power-down.

- **Dynamic Power Management**
  - Adapt to activity at run time.
  - Example: automatically disabling function units.
Dynamic Power Management

- Inherent tradeoff: energy vs. performance
- Fundamental premises
  - Non-uniform workload during operation
  - Possible to predict workload with some degree of accuracy
## PowerPC 603 Activity

Percentage of time idle for SPEC integer/floating-point:

<table>
<thead>
<tr>
<th>unit</th>
<th>Specint92</th>
<th>Specfp92</th>
</tr>
</thead>
<tbody>
<tr>
<td>D cache</td>
<td>29%</td>
<td>28%</td>
</tr>
<tr>
<td>I cache</td>
<td>29%</td>
<td>17%</td>
</tr>
<tr>
<td>load/store</td>
<td>35%</td>
<td>17%</td>
</tr>
<tr>
<td>fixed-point</td>
<td>38%</td>
<td>76%</td>
</tr>
<tr>
<td>floating-point</td>
<td>99%</td>
<td>30%</td>
</tr>
<tr>
<td>system register</td>
<td>89%</td>
<td>97%</td>
</tr>
</tbody>
</table>
Problem Formulations

- Minimize energy under performance constraints
  - Real-time applications

- Optimize performance under energy/power constraints
  - Battery lifetime (energy)
  - Temperature (power)
Power Down/Up Cost

- Going into/out of an inactive mode costs
  - time
  - energy

- Must determine if going into an inactive mode is worthwhile.

- Model power states with a Power State Machine (PSM)
SA-1100 Power State Machine

$P_{ON} = 400 \text{ mW}$

$P_{OFF} = 50 \text{ mW}$

$P_{OFF} = 0.16 \text{ mW}$

$P_{TR} = P_{ON}$
Greedy Policy

- Immediately goes to sleep when system becomes idle

- Works when transition time is negligible
  - Ex. between IDLE and RUN in SA-1100

- Doesn’t work when transition time is long!
  - Ex. between SLEEP and RUN/IDLE in SA-1100
  - Need better solutions!
Break-Even Time $T_{BE}$

- Minimum idle time required to compensate for the cost of entering an inactive state.

- Enter an inactive state is beneficial only if idle time $> T_{BE}$. 
Break-Even Time

\[ P_{TR} \leq P_{ON} \]

- **\( P_{TR} \)**: Power consumption during transition
- **\( P_{ON} \)**: Power consumption when active
- **\( T_{BE} \)** of an inactive state is the total time it takes to enter and leave the state

\[ T_{BE} = T_{TR} = T_{ON,OFF} + T_{OFF,ON} \]

- \( T_{BE} = 160 \text{ ms} + 90 \mu \text{s} \) for SLEEP in SA-1100
SA-1100 Power State Machine

\[ P_{\text{ON}} = 400 \text{ mW} \]

\[ P_{\text{OFF}} = 50 \text{ mW} \]

\[ P_{\text{OFF}} = 0.16 \text{ mW} \]

\[ P_{\text{TR}} = P_{\text{ON}} \]
Break-Even Time

$P_{TR} > P_{ON}$

- $T_{BE}$ must include additional inactive time to compensate for extra power consumption during transition.

$$T_{BE} = T_{TR} + T_{TR}(P_{TR} - P_{ON})/(P_{ON} - P_{OFF})$$

- Reduce $T_{BE}$ → save more energy
  - Shorter $T_{TR}$
  - Higher power difference between $P_{ON} - P_{OFF}$
  - Lower $P_{TR}$
Inherent Exploitability

- Achievable energy saving depends on workload!
  - Distribution of idle periods

- Given an idle period $T_{idle} > T_{BE}$
  - $E_S(T_{idle}) = (T_{idle} - T_{TR})(P_{ON} - P_{OFF}) + T_{TR}(P_{ON} - P_{TR})$

- Assumptions
  - No performance penalty.
  - Ideal manager with knowledge of workload in advance.
Inherent Exploitability based on real workload

Graph a) shows the break-even time (ms) against power (in mW) for different activities: editing, development, and games. Graph b) presents the break-even time (ms) against power (in mW) for sleep and idle states.
Time-Power Product
Workload-independent Metric

\[ C_S = T_{BE} P_{OFF} \]

- An inactive state with lower \( C_S \) may save more energy
- Only a crude estimate
  - May not be representative of real power savings
Predictive Techniques

- Interested event: $p = \{T_{\text{idle}} > T_{\text{BE}}\}$
  - Predict based on history

- Observed event: $o$
  - Triggers state transition

- Objective: predict $p$ based on $o$
Metrics

- **Safety**: conditional probability \( \text{Prob}(p \mid o) \)
  - If an observed event happens \( \Rightarrow \) the probability of \( T_{\text{idle}} > T_{BE} \)
  - Ideally, safety = 1.

- **Efficiency**: \( \text{Prob}(o \mid p) \)
  - If \( T_{\text{idle}} > T_{BE} \) \( \Rightarrow \) the probability of correctly predicting.

- Overprediction \( \Rightarrow \) high performance penalty \( \Rightarrow \) poor safety
- Underprediction \( \Rightarrow \) wastes energy \( \Rightarrow \) poor efficiency
Fixed Timeout Policy

- Enter inactive state when system has been idle for $T_{TO}$
  - $o: T_{idle} > T_{TO}$

- Wake up in response to activity

- Hypothesis: If system has been idle for $T_{TO} \rightarrow$ it will continue to be idle for $T_{idle} - T_{TO} > T_{BE}$
Increasing $T_{TO}$ improves safety, but reduces efficiency.

Highly workload dependent

Karlin’s result: $T_{TO} = T_{BE}$ → Energy consumption is at most twice the energy consumed under an ideal policy.
Impact of Timeout Threshold

(a) Conditional probability against timeout
(b) Power (mW) against timeout
Impact of Workloads

![Graph a) and b) showing the impact of workloads on efficiency and power consumption.](image)
Critique: Fixed Timeout

- How to set timeout threshold?
  - Tradeoff between safety and efficiency
  - Works best when workload traces are available

- Fundamental limitations
  - Always waste energy before reaching the timeout threshold
  - Always incur performance penalty for wake up
Possible Improvement

- **Predictive shutdown**
  - shut down *immediately* when an idle period starts.
  - avoid wasting energy before reaching the timeout threshold.
  - more efficient, less safe.

- **Predictive wakeup**
  - wake up when the *predicted* idle time expires, even if no new activity has occurred.
  - avoid performance penalty for wakeup.
  - less efficient, safer.
Predictive Shutdown
Threshold-based Policy

- Observation: short active period tends to be followed by long idle period.
- If active period < threshold, the following idle period is predicted to be longer than $T_{BE}$.
- What is the right threshold?
  - Workload dependent
  - Require offline analysis
Threshold-based Predictive Shutdown

(a) Idle period (ms) vs. Last active period (ms)

(b) Conditional probability vs. Threshold (ms)

- Solid line: safety
- Dashed line: efficiency
Predictive Wakeup
Regression-based Algorithm

- Predict the length of an idle period based on
  - preceding active period
  - previous n pairs of idle/active periods
- More complicated than fixed timeout
  - Need to maintain history information
- Depend on offline analysis and traces to determine the regression function and parameters
Adapt to Workload Changes

- Grade n timeout thresholds based on history
  - Use the best one for prediction
  - Use weighted average of n thresholds

- Adjust timeout
  - Increase timeout threshold if causing too many shutdowns
  - Decrease timeout threshold if causing too few shutdowns

- Stochastic techniques
Critiques: History-based Predictors

- Depend on short-term correlation between past & future
  - Hold in many workloads
  - Fail when the correlation is weak

- Workload in many embedded systems are more predictable than PCs
  - Workload (e.g., periodic tasks) known *a priori*
  - Specialized application
ESSAT

**Efficient Sleep Scheduling based on Application Timing**

- Reduce radio power consumption by exploiting the timing properties of periodic queries in sensor networks
- Sleep scheduling incurs low delay penalty

---

Power

- Hardware support
- Power management policy
- Power manager
- Holistic approach
Power Manager

- Usually implemented in software (OS) for flexibility
- Hardware and software co-design
  - Software implements policy
  - Hardware implements power saving mechanisms
- Need standard interfaces to deal with hardware diversity
  - Different vendors
  - Different devices: processor, sensor, controller ...
ACPI
Advanced Configuration and Power Interface

Open standard for power management services.
http://www.acpi.info/
Used as **contract** between hardware and OS vendors
ACPI Global Power States

- G3: mechanical off – no power consumption
- G2: soft off – restore requires full OS reboot
- G1: sleeping state
  - S1: low wake-up latency with no loss of context
  - S2: low latency with loss of CPU/cache state
  - S3: low latency with loss of all state except memory
  - S4: lowest-power state with all devices off
- G0: working state
Intel Core i7 C States

![Diagram showing C states for Intel Core i7]

- **C0**: Core clock and PLL are active, shared cache and core caches are flushed, wakeup time and core idle power are active.
- **C1**: Core clock and PLL are off, shared cache and core caches are flushed, wakeup time is active, core idle power is off.
- **C3**: Core clock and PLL are off, shared cache and core caches are flushed, wakeup time is active, core idle power is off.
- **C6**: Core clock and PLL are off, shared cache and core caches are flushed, wakeup time is off, core idle power is approximately 0.

* Rough approximation
## Intel Pentium M P States

<table>
<thead>
<tr>
<th>P-State</th>
<th>Frequency</th>
<th>Voltage</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>1.6 GHz</td>
<td>1.484 V</td>
<td>25 Watts</td>
</tr>
<tr>
<td>P1</td>
<td>1.4 GHz</td>
<td>1.420 V</td>
<td>~17 Watts</td>
</tr>
<tr>
<td>P2</td>
<td>1.2 GHz</td>
<td>1.276 V</td>
<td>~13 Watts</td>
</tr>
<tr>
<td>P3</td>
<td>1.0 GHz</td>
<td>1.164 V</td>
<td>~10 Watts</td>
</tr>
<tr>
<td>P4</td>
<td>800 MHz</td>
<td>1.036 V</td>
<td>~8 Watts</td>
</tr>
<tr>
<td>P5</td>
<td>600 MHz</td>
<td>0.956 V</td>
<td>6 Watts</td>
</tr>
</tbody>
</table>
Device Power States

- Device power state is **invisible** to the user.
  - Devices may be inactive when the system is in the working state.
- Each device may be controlled by a **separate** power management policy.
Power

- Hardware support
- Power management policy
- Power manager
- Holistic approach
Holistic View of Power Consumption

- Instruction execution (CPU)
- Cache (instruction, data)
- Main memory
- Other: non-volatile memory, display, network interface, I/O devices
Mote

- **System view when switching from sleep to active**

Sources of Energy Consumption

Relative energy per operation (Catthoor):

- memory transfer: 33
- external I/O: 10
- SRAM write: 9
- SRAM read: 4.4
- multiply: 3.6
- add: 1
Optimize Memory System

- Different instructions $\rightarrow$ Different energy consumption

- Energy: register $<<$ cache (SRAM) $<<$ memory (DRAM)

- Optimizing memory system $\rightarrow$ significant energy saving
Cache Behavior

Sweet spot in cache size:

- **Too small**: waste energy on memory accesses;
- **Too large**: cache itself burns too much power.
Impacts of Cache Size

(a) MPEG: energy

(b) MPEG: exec time

(c) bsort: energy

(d) bsort: exec time
Optimizations

- Reduce memory footprint
  - Reduce code size
  - Analyze/test footprint to find right size: stack, heap...

- Find correct cache size
  - Analyze cache behavior (size of working set)

- Minimize memory and cache access
  - Use registers efficiently → less cache access
  - Identify and eliminate cache conflicts → less memory access

- Better performance → More idle time!
Reading

- Textbook 3.7
- **Required:** Sections I, II, III.A, III.B, IV of
- **Interesting:** Intel Inside...Your Smartphone