POWER MANAGEMENT
SELECTED ISSUES

COMP9242 2007/S2 Week 8
Laptops

• Obvious objective: prolong disconnected operating
  ➜ maximise battery lifetime
Laptops

- Obvious objective: prolong disconnected operating
  ➔ maximise battery lifetime

- Assume: fixed battery capacity $C$
  ➔ smaller discharge current $I = \frac{dQ}{dt}$ increases usable time
  ➔ objective is to reduce average power consumption $P = VI$
  ➔ reality more complex: strong current drains battery more
Laptops

- Obvious objective: prolong disconnected operating
  - maximise battery lifetime

- Assume: fixed battery capacity $C$
  - smaller discharge current $I = \frac{dQ}{dt}$ increases usable time
  - objective is to reduce average power consumption $P = VI$
  - reality more complex: strong current drains battery more

- Approaches:
  - spin down disk when idle
    - trade-off with response time
  - put peripherals into low-power mode when idle
  - blank screen
  - reduce clock frequency
**Power Management Issues**

Laptops

- **Obvious objective:** prolong disconnected operating
  - maximise battery lifetime

- **Assume:** fixed battery capacity $C$
  - smaller discharge current $I = \frac{dQ}{dt}$ increases usable time
  - objective is to reduce average power consumption $P = VI$
  - reality more complex: strong current drains battery more

- **Approaches:**
  - spin down disk when idle
    - trade-off with response time
  - put peripherals into low-power mode when idle
  - blank screen
  - reduce clock frequency
    - why does this help?
Power Management Issues

Desktops

- Objectives: prevent overheating of CPU core
Power Management Issues

Desktops

- Objectives: prevent overheating of CPU core
- Approaches:
  - reduce clock frequency
Desктопы

- Опричники: предотвратить перегрев ЦПУ ядра
- Аппрашес:
  ➜ снизить частоту тактовой частоты
  – почему это помогает?
Power Management Issues

Servers

- Objectives:
  - prevent overheating of CPU core
  - reduce overall power intake
  - reduce overall heat dissipation
**Power Management Issues**

Servers

- **Objectives:**
  - prevent overheating of CPU core
  - reduce overall *power intake*
  - reduce overall *heat dissipation*

- **Note:** about 1/3 of lifetime cost of hosting centres is power!
  - for computer power supply
  - for air conditioning
Power Management Issues

Servers

- Objectives:
  - prevent overheating of CPU core
  - reduce overall *power intake*
  - reduce overall *heat dissipation*

- Note: about 1/3 of lifetime cost of hosting centres is power!
  - for computer power supply
  - for air conditioning

- Approaches:
  - reduce clock frequency
  - spin down disks
  - shut down nodes
    - use virtualization to migrate load
General Power Management Issues

- Power-performance tradeoff:
  - transitioning between power modes takes time
  - time lost for transition to high-power mode degrades performance
General Power Management Issues

- Power-performance tradeoff:
  ➔ transitioning between power modes takes time
  ➔ time lost for transition to high-power mode degrades performance

- Cost-savings tradeoff:
  ➔ transitioning between power modes costs energy
  ➔ if transitioning too aggressively may consume more energy
General Power Management Issues

- **Power-performance tradeoff:**
  - transitioning between power modes takes time
  - time lost for transition to high-power mode degrades performance

- **Cost-savings tradeoff:**
  - transitioning between power modes costs energy
  - if transitioning too aggressively may consume more energy
  - rule of thumb for transition to low-power mode:
    - transition when wasted energy exceeds transitioning cost
Low-power modes

- Disk:  
  ➜ spin down
Low-power modes

- Disk:
  - spin down

- Memory:
  - loss-free low-power state
    - content not accessible until return to normal state
    - fast transition
    - moderate savings
  - destructive low-power state
    - content lost
    - slow transition
    - high savings
Low-power modes

- Disk:
  - spin down

- Memory:
  - loss-free low-power state
    - content not accessible until return to normal state
    - fast transition
    - moderate savings
  - destructive low-power state
    - content lost
    - slow transition
    - high savings

- CPU:
  - dynamic voltage and frequency scaling (DVFS)
Basis of DVFS:

1. DRAM circuits draw power only when switching
   - switching energy (approximately) constant per clock cycle:
     \[ P = VI \propto V^2 \]
   - switches per second = core frequency:
     \[ P \propto f \]
   - overall:
     \[ P(f) \propto fV^2 \]
Basis of DVFS:

1. DRAM circuits draw power only when switching
   ➔ switching energy (approximately) constant per clock cycle:
   \[ P = VI \propto V^2 \]
   ➔ switches per second = core frequency:
   \[ P \propto f \]
   ➔ overall:
   \[ P(f) \propto fV^2 \]

2. DRAM circuits switch faster under higher voltage
   ➔ can reduce voltage at lower frequencies (to a limit):
   \[ V_{min} \text{ monotonic in } f \]
   \[ P_{min} \propto f^2 \cdots f^3 \]
Basis of DVFS:

3. Work performed per clock cycle is independent of frequency
   - computation takes fixed number of cycles \( N \)
   - total computation time is:
     \[ T = \frac{N}{f} \]
   - energy for computation is:
     \[ E = PT \propto V^2 \]
     ... independent of \( f \)!
Basis of DVFS:

3. Work performed per clock cycle is independent of frequency
   ➜ computation takes fixed number of cycles $N$
   ➜ total computation time is:

   $$ T = \frac{N}{f} $$

   ➜ energy for computation is:

   $$ E = PT \propto V^2 $$

   ... independent of $f$!

   ➜ frequency scaling alone doesn’t save energy
   • but reduces core temperature

   ➜ energy saving comes from ability to reduce voltage with frequency
Basis of DVFS:

3. Work performed per clock cycle is independent of frequency
   ➜ computation takes fixed number of cycles $N$
   ➜ total computation time is:
   $$T = \frac{N}{f}$$
   ➜ energy for computation is:
   $$E = PT \propto V^2$$
   ... independent of $f$!
   ➜ frequency scaling alone doesn’t save energy
      • but reduces core temperature
   ➜ energy saving comes from ability to reduce voltage with frequency

• Let’s have a look at real hardware!
PLEB 2 (NICTA ’05):

- Single board computer based on Intel XScale (PXA255)
- Representative of a typical embedded system
Hardware Platform

- Three power supplies: CPU, Memory and IO
- Each power supply’s current is measured
- On-board micro-controller (AVR) for monitoring
- Samples taken periodically according to AVR clock
- \( f_{\text{max}} \) samples is 15 KHz, subdivided into three channels
## PXA255 Voltage and Frequency Settings

Valid voltage/frequency configurations:

<table>
<thead>
<tr>
<th>$V_{core}$ (V)</th>
<th>$f_{cpu}$ (MHz)</th>
<th>$f_{internalbus}$ (MHz)</th>
<th>$f_{mem}$ (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>1.0</td>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.1</td>
<td>300</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.3</td>
<td>400</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.3</td>
<td>400</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>
14 Macro benchmarks:

- most from MiBench, selected according to resource limitations
- automotive, industrial control, network, office, signal processing
- plus 4 more: gzip, MP3, image analysis, speech compression
1. Normalized CPU Energy Model

The diagram above illustrates the model predicted energy for different configurations of Vcore, fcpu, and fint bus. The x-axis represents the configurations, and the y-axis shows the CPU energy. The graph shows a significant increase in energy as the configurations change from 1.0 to 1.3 for Vcore, fcpu, and fint bus.
1. **Normalized CPU Energy**

The graph illustrates the model predicted energy consumption under different configurations. The x-axis represents the configurations with varying values for `Vcore`, `fcpu`, and `fint bus`. The y-axis shows the CPU energy. The graph compares model predicted energy versus CPU-intensive and memory-intensive scenarios. The configurations are marked with specific parameter values, such as `Vcore=1.0`, `fcpu=100`, and `fint bus=50`. The graph shows how energy consumption varies with these changes, highlighting the impact on energy efficiency.
1. B NORMALIZED MEMORY ENERGY

![Normalized Memory Energy Graph]

- **Configurations:**
  - **V\text{core}**: 1.0, 1.0, 1.1, 1.3, 1.3
  - **f\text{cpu}**: 100, 200, 300, 400, 400
  - **f\text{int bus}**: 50, 100, 100, 100, 200

- **Graph Legend:**
  - CPU-intensive
  - Memory-intensive

---

**COMP9242 WEEK 08**  
**Power Management 15**  
**UNSW 2007/S2**
2. Memory Power

![Graph showing power consumption for different configurations.](image-url)

- **Vcore**: 1.0, 1.1, 1.3
- **fcpu**: 100, 200, 300, 400
- **fintr bus**: 50, 100

The graph illustrates the power consumption for different configurations, with lines indicating Cpu-intensive (green) and Memory-intensive (blue) scenarios.
2. MEMORY POWER VS CPU POWER

![Graph showing the relationship between memory power and CPU power across various configurations.]
3. Normalized Total Energy

![Graph showing normalized total energy across different configurations.](image-url)
3. Normalized Total Energy vs Naïve Model

The diagram illustrates the normalized total energy for different configurations of $V_{core}$, $f_{cpu}$, and $f_{int bus}$. The x-axis represents the configurations, while the y-axis shows the normalized total energy. The graph compares model predicted energy with CPU-intensive and memory-intensive models.
Padding with idle power to equal total time

Benchmark X
slowest setting
longest execution time

$\text{t}$
Adding with idle power to equal total time

Benchmark X
slowest setting
longest execution time

Benchmark X
faster setting
shorter execution time
Padding with idle power to equal total time

Benchmark X
slowest setting
longest execution time

Benchmark X
faster setting
shorter execution time

Padding with
CPU Power
in Idle mode

$\mathbf{t}$
4. Normalised Total Energy Padded to Equal Time
What if we cannot measure memory power?
What if we cannot measure memory power?

- Model memory power via performance counters.
- Counting memory accesses leads to good results (WB02)
- However, XScale doesn’t have enough performance counters
What if we cannot measure memory power?

- Model memory power via performance counters.
- Counting memory accesses leads to good results (WB02)
- However, XScale doesn’t have enough performance counters
- Idea: count cache misses instead?
- Investigated via two micro-benchmarks
  - read: only load instructions
  - modify: load and then store to same location
5. Memory power vs cache miss rate
5. Memory power vs cache miss rate
What can we learn from this?

- The simple assumptions behind the common DVFS approach are wrong!
What can we learn from this?

- The simple assumptions behind the common DVFS approach are **wrong**!
  - memory stalls
  - static (leakage) power
  - memory power (and I/O devices)
  - multiple frequency/voltage domains
What can we learn from this?

- The simple assumptions behind the common DVFS approach are *wrong*!
  - memory stalls
  - static (leakage) power
  - memory power (and I/O devices)
  - multiple frequency/voltage domains

- Things aren’t as simple as some people think...
  - yet this model is used in many publications!
MEMORY STALLS

- Total time of memory access is independent of core frequency
- Cycles for memory access are proportional to core frequency
Memory stalls

- Total time of memory access is independent of core frequency
- Cycles for memory access are proportional to core frequency
Memory stalls

- Hence, with increasing core frequency:
  - time for CPU instructions drops
  - time for memory accesses is unchanged
MEMORY STALLS

- Hence, with increasing core frequency:
  - time for CPU instructions drops
  - time for memory accesses is unchanged
Real Program Behaviour

Bus: 66.4MHz, Memory: 132.7MHz

Normalised Time vs CPU Core Frequency (MHz)
- DRAM consumes power also when not switching ➔ in modern processors static power is starting to dominate!

- Static power is independent of frequency ➔ static energy is proportional to execution time!
**Static Power**

- DRAM consumes power also when **not** switching
  - In modern processors static power is starting to dominate!

- Static *power* is independent of frequency
  - Static *energy* is proportional to execution time!

- Hence, with increasing core frequency:
  - Dynamic energy decreases
  - Static energy increases
Static power

- DRAM consumes power also when **not** switching
  - in modern processors static power is starting to dominate!

- Static *power* is independent of frequency
  - static *energy* is proportional to execution time!

- Hence, with increasing core frequency:
  - dynamic energy decreases
  - static energy increases
  - optimal frequency depends on relative size of static and dynamic power
  - lowest energy may be at intermediate frequency!
What is Needed for Energy Management?

- Effect of DVFS on energy consumption depends on workload characteristics
  - need individual settings for each application
  - requires *a-priori* characterisation or run-time adjustment
What is Needed for Energy Management?

- Effect of DVFS on energy consumption depends on workload characteristics
  - need individual settings for each application
  - requires *a-priori* characterisation or run-time adjustment

- Minimising energy with DVFS requires:
  - prediction of *performance* (execution time) under DVFS
  - prediction of *power consumption* under DVFS
What is Needed for Energy Management?

- Effect of DVFS on energy consumption depends on workload characteristics
  - need individual settings for each application
  - requires \textit{a-priori} characterisation or run-time adjustment

- Minimising energy with DVFS requires:
  - prediction of \textit{performance} (execution time) under DVFS
  - prediction of \textit{power consumption} under DVFS

- Needs to happen in real time if \textit{a-priori} characterisation is infeasible
  - can use performance counters (WB02)
  - needs to be done with low overhead
Realistic Execution Time Model

\[ T = \frac{C_{cpu}}{f_{cpu}} + \frac{C_{bus}}{f_{bus}} + \frac{C_{mem}}{f_{mem}} + \frac{C_{io}}{f_{io}} + \ldots \]

- \( C_x \) represents the number of clock cycles for \( f_x \).
- \( C_x \) is workload-specific, independent of frequency.
- Assumes no dependences — works well in practice.
Realistic Execution Time Model

\[ T = \frac{C_{cpu}}{f_{cpu}} + \frac{C_{bus}}{f_{bus}} + \frac{C_{mem}}{f_{mem}} + \frac{C_{io}}{f_{io}} + \ldots \]

- \( C_x \) represents the number of clock cycles for \( f_x \).
- \( C_x \) is workload-specific, independent of frequency.
- Assumes no dependences — works well in practice.

\( C_x \): characterise online using performance counters

\[ C_{bus} = \alpha_1 PMC_1 + \alpha_2 PMC_2 + \ldots \]
\[ C_{mem} = \beta_1 PMC_1 + \beta_2 PMC_2 + \ldots \]

\( C_{cpu} \) inferred from the other results)
Estimating $C_{cpu}$:

$\rightarrow$ $C_{cpu}$ is difficult to estimate using performance counters alone
Estimating $C_{cpu}$:

- $C_{cpu}$ is difficult to estimate using performance counters alone
- But...

$$C_{cpu} = C_{tot} - \frac{f_{cpu}}{f_{bus}} C_{bus} - \frac{f_{cpu}}{f_{mem}} C_{mem} + \ldots$$

- If we can measure cycles, we can estimate $C_{cpu}$.
- Or... If we can estimate the off-chip cycles, we can calculate the on-chip cycles.
The data:

- Typical embedded platform (PLEB 2, XScale based)
- Cycle counter, 2 performance counters, 13 events
- 22 frequency setpoints with different $f_{cpu}$, $f_{bus}$ and $f_{mem}$
- 37 benchmarks run to completion at each setpoint for all frequency settings
- All performance counter events were measured end-to-end for each benchmark/frequency
- Benchmarks were partitioned for calibration and validation
- Measurements: Cycles, Frequencies, Performance counters
Which counters are the best to use?

Per workload:

\[
T = \frac{C_{cpu}}{f_{cpu}} + \frac{C_{bus}}{f_{bus}} + \frac{C_{mem}}{f_{mem}} + \frac{C_{io}}{f_{io}} + \ldots
\]

Per platform:

\[
C_{bus} = \alpha_1 PMC_1 + \alpha_2 PMC_2 + \ldots
\]

\[
C_{mem} = \beta_1 PMC_1 + \beta_2 PMC_2 + \ldots
\]

\rightarrow\text{Fit for each benchmark for the average } PMC_x

\rightarrow\text{Exhaustive parameter selection based on } bic \text{ or } R^2.$
Best correlation of power and PMC readings for different number of counters.

Pickling Performance Counters

COMP9242 Week 08
Power Management 37
UNSW 2007/S2
Making predictions:

How long would it have taken to run at XX MHz?

\[ C'_\text{tot} = C_\text{tot} - \frac{f_{\text{cpu}}}{f_{\text{bus}}} C_{\text{bus}} - \frac{f_{\text{cpu}}}{f_{\text{mem}}} C_{\text{mem}} + \frac{f'_{\text{cpu}}}{f'_{\text{bus}}} C_{\text{bus}} + \frac{f'_{\text{cpu}}}{f'_{\text{mem}}} C_{\text{mem}} \]

Errors:

- Predicting the performance at the maximum frequency
- 2-parameter: avg 1.7%, max 7%
- CPU frequency only: avg 10%, max 36%
Relative performance:

→ Often we want a normalised performance (e.g. 50% of maximum).

\[ s = \frac{t^{\text{max}}}{t'} = \left( \frac{f'_{\text{cpu}}}{f_{\text{cpu}}^{\text{max}}} \right) \times \left( \frac{C_{\text{tot}}^{\text{max}}}{C'_{\text{tot}}} \right) \]

This is a linear equation in terms of the PMCs.
Power/Energy model basics:

For our system:
Power/Energy model basics:

→ Events each use an amount of energy
→ An event may use energy in more than one voltage domain

For our system:

\[ E_{events} = V_{cpu}^2 (\alpha_0 PMC_0 + \cdots + \alpha_m PMC_m) + \beta_0 PMC_0 + \cdots + \beta_m PMC_m \]
Power/Energy model basics:

→ *Events* each use an amount of energy
→ An event may use energy in more than one voltage domain
→ Clock cycles count as events

For our system:

\[ E_{\text{events}} = V_{\text{cpu}}^2 (\alpha_0 PMC_0 + \cdots + \alpha_m PMC_m) + \beta_0 PMC_0 + \cdots + \beta_m PMC_m \]

\[ E_{\text{freqs}} = V_{\text{cpu}}^2 (\gamma_1 f_{\text{cpu}} + \gamma_2 f_{\text{bus}} + \gamma_3 f_{\text{mem}}) \Delta t + (\gamma_4 f_{\text{cpu}} + \gamma_5 f_{\text{bus}} + \gamma_6 f_{\text{mem}}) \Delta t \]
Power/Energy model basics:

- **Events** each use an amount of energy
- An event may use energy in more than one voltage domain
- Clock cycles count as events
- Static power models power not related to events or voltages.
- Constant IO power for the benchmarks tested.

For our system:

\[
E_{\text{events}} = V_{\text{cpu}}^2 (\alpha_0 \text{PMC}_0 + \cdots + \alpha_m \text{PMC}_m) + \beta_0 \text{PMC}_0 + \cdots + \beta_m \text{PMC}_m
\]

\[
E_{\text{freqs}} = V_{\text{cpu}}^2 (\gamma_1 f_{\text{cpu}} + \gamma_2 f_{\text{bus}} + \gamma_3 f_{\text{mem}}) \Delta t + (\gamma_4 f_{\text{cpu}} + \gamma_5 f_{\text{bus}} + \gamma_6 f_{\text{mem}}) \Delta t
\]

\[
E_{\text{static}} = P_{\text{static}} \Delta t
\]
Power/Energy model basics:

- Events each use an amount of energy
- An event may use energy in more than one voltage domain
- Clock cycles count as events
- Static power models power not related to events or voltages.
- Constant IO power for the benchmarks tested.

For our system:

\[
E_{\text{events}} = V_{\text{cpu}}^2 (\alpha_0 PMC_0 + \cdots + \alpha_m PMC_m) + \beta_0 PMC_0 + \cdots + \beta_m PMC_m
\]

\[
E_{\text{freqs}} = V_{\text{cpu}}^2 (\gamma_1 f_{\text{cpu}} + \gamma_2 f_{\text{bus}} + \gamma_3 f_{\text{mem}}) \Delta t + (\gamma_4 f_{\text{cpu}} + \gamma_5 f_{\text{bus}} + \gamma_6 f_{\text{mem}}) \Delta t
\]

\[
E_{\text{static}} = P_{\text{static}} \Delta t
\]

Time is essential for either power or energy estimation
The Data:

- Same as ETM data
- End-to-end energy measured using a shunt resistor
- Measurements triggered using a GPIO
- Each workload sample is used to predict the power for every other sample
- Frequency and voltage are varied independently
**Power/Energy model**

Power error data (using a measured run-time):

<table>
<thead>
<tr>
<th>Counters</th>
<th>Param.</th>
<th>$R^2$</th>
<th>Max Err (%)</th>
<th>Avg Err (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.9836</td>
<td>7.46</td>
<td>2.14</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.9871</td>
<td>6.94</td>
<td>2.31</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.9904</td>
<td>4.85</td>
<td>1.26</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.9922</td>
<td>3.78</td>
<td>1.16</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>0.993</td>
<td>3.68</td>
<td>0.92</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>0.9938</td>
<td>2.94</td>
<td>0.89</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>0.9947</td>
<td>2.75</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Energy error data (estimated run-time — four counters):

- Max: 4.9%, Avg: 1.5%
Using the models

Sound approach:

• Determine appropriate PMC set and weights via systematic process
  – once per platform
• Use these to predict performance and power for actual load at run time
• Context-switch DVFS settings
**Using the Models**

**Sound approach:**
- Determine appropriate PMC set and weights via systematic process
  - once per platform
- Use these to predict performance and power for actual load at run time
- Context-switch DVFS settings

**Using these results:**
- Thread-level DVFS.
- Identify appropriate frequency choices
- *Energy · Delay*
- Sleep mode vs. DVFS trade-offs
Performance predictions at run-time:

- A toy system: frequency scaling in Linux
- Measures the performance counters after each time slice
- Predicts the performance at all of the other setpoints for that time-slice
- Assuming temporal locality, chooses the setpoint with the closest to a chosen performance
- Error: avg 1.9%, max 7%

Not useful in itself, but demonstrates the model