New Interfaces for Achieving Power Agility on Mobile Devices

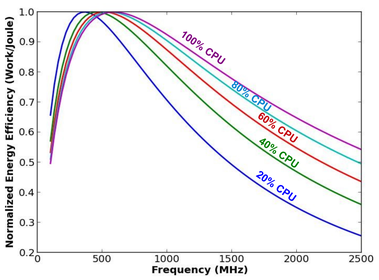
Guru Prasad, Scott Haseley, Geoffrey Challen
University at Buffalo
{gurupras,shaseley,challen}@buffalo.edu

Rizwana Begum, Mark Hempstead
Drexel University
{rb639,mhempstead}@drexel.edu

Motivation

Efficiency vs. Performance Tradeoffs

Energy-constrained mobile devices integrate multiple hardware components with *inherent* trade-offs between performance and efficiency. **As performance increases, efficiency in terms of work-per-joule decreases.**

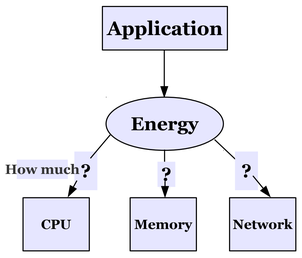


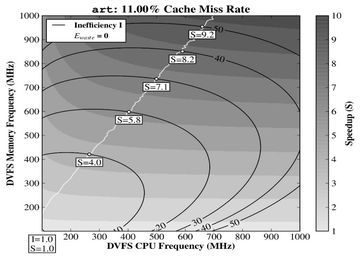
Example Hardware Performance-Efficiency Trade-offs

Component	Performance Metric
CPU	Speed
Memory	Bandwidth
Wifi	Latency

Cross-Component Choices

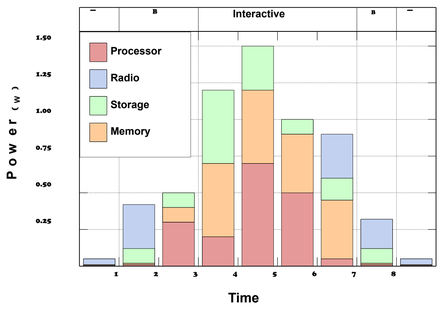
Multiple components with efficiency-performance trade-offs force applications to make choices about how to allocate available energy to achieve the best performance. **Incorrect component tuning can degrade performance and waste energy.**

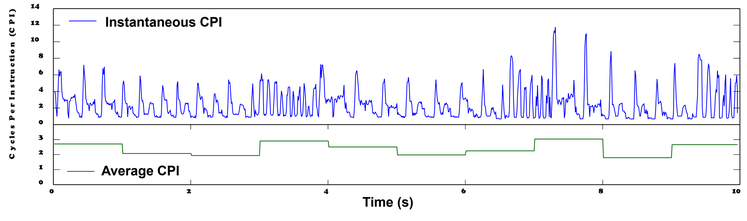




Application Phases

Since application needs change over time, maintaining a correct cross-component allocation requires frequent changes to device settings to respond to application needs. **We define the ability of a device to select and transition to right set of components and their settings as *power agility.***





http://blue.cse.buffalo.edu

Design Goals

Only slow down to save energy. Because they do not consider efficiency-performance tradeoffs, rate limiting approaches can waste energy *and* degrade performance.

Make application energy allocation explicit. Don't rely on the OS to guess changing application energy-performance dependencies and make cross-component energy allocations.

Constrain energy, not performance. Instead of treating energy usage as a side effect of hardware performance settings, allow hardware to maximize energy-constrained performance.

Allocating Inefficiency

We define ***inefficiency*** as the amount of extra energy used while executing a task above the minimum energy (E_{min}) the task required to execute:

$$\mathcal{I} = \frac{(E_{perf} + E_{waste})}{E_{min}} + 1$$

A portion of this extra energy (E_{perf}) improves performance, while a portion (E_{waste}) does not improve performance and is wasted.

Allocating a task inefficiency allows it to use extra energy to run faster but does not interfere with scheduling or stop tasks from running. Because inefficiency allocation never "saves" energy only by delaying work, it is a particularly good fit for interactive tasks.

Our current power-agile system design allocates inefficiency proportional to scheduling priorities, allowing simple integration with existing task prioritization.

Status and Challenges

AgileAndroid

We are implementing a power-agile smartphone platform called *AgileAndroid* based on the Android Open Source Project (AOSP) platform sources. Our modified operating system and platform will run both in the `gem5` simulator, to enable hardware experimentation, and on existing Android smartphones. We have added support for dynamic voltage and frequency (DVFS) scaled cores and memory to `gem5`.

Challenges and Future Work

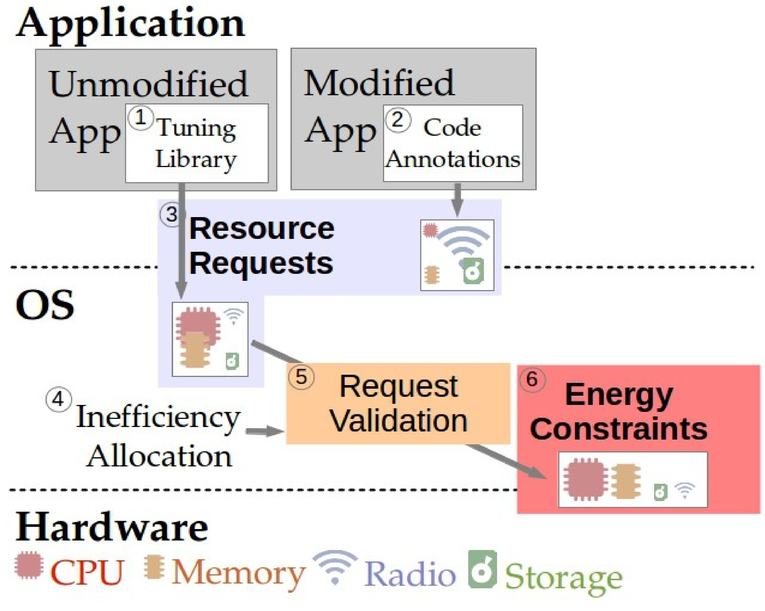
- Tasks accessing **shared components** on multicore systems may have allocated them different amounts of inefficiency. Reordering within the scheduling queue may help align resource requests.
- While hardware improvements are reducing the **transition latency** between voltage and frequency domains, other components may still present high overhead to change energy constraints.
- Until hardware components support energy constraints directly, supporting **legacy hardware** will require drivers that can map energy constraints to performance settings.
- Inefficiency provides a natural model for identifying opportunities to enable **race-to-sleep behavior** when sleep states are available and idle costs are high, but this is unaddressed in our current design.
- More work at the OS-hardware level will be required to integrate **new hardware features** with energy-performance implications such as computational sprinting.

Power-Agile System Architecture

Making inefficiency allocation explicit requires **new interfaces** between the application, OS, and hardware. At the application-OS interface we introduce ***resource requests*** allowing applications to explicitly distribute their allocated inefficiency between components with efficiency-performance tradeoffs. At the OS-hardware interface we communicate energy requirements directly to hardware using ***energy constraints***.

Applications can use either *automated tuning libraries* (1) or *explicit code annotations* (2) to determine how to allocate energy between hardware components to maximize performance. This allows our power-agile design to support both unmodified legacy apps as well as ones rewritten to use the resource request mechanism.

These requests are communicated to the operating system (3), which maintains a per-task inefficiency allocation (4) representing that tasks ability to consume extra energy to improve performance. After validating each resource request (5), the OS uses the request to set per-component hardware energy constraints (6) directly. Hardware components then run as fast as possible without exceeding their inefficiency constraint, allowing hardware to conserve energy on timescales that cannot be managed by the OS.



http://dpac.ece.drexel.edu/