# Architecture and System Support for Power-Agile Computing

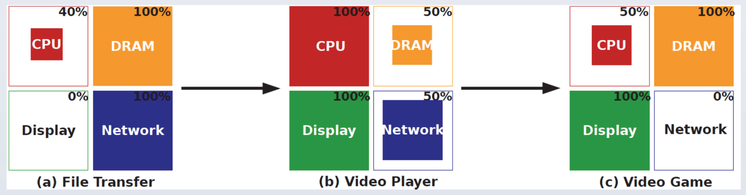

PIs: Geoffrey Challen[3] and Mark Hempstead[1,2]

Students: Rizwana Begum[2], Guru Prasad Srinivasa[3], Scott Haseley[3], David Werner[1,2], Jerry Ajay[3]

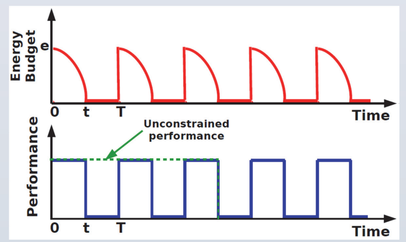
1. Computer Architecture Lab, Department of Electrical and Computer Engineering, Tufts University, Medford, MA
2. Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA
3. Blue, Department of Computer Science, University at Buffalo, Buffalo, NY


## The Need for Power-Agile Systems

- Battery life is the number one complaint among users
- Mobile devices have multiple hardware components each with its own power-performance tradeoffs
- Current Linux/Android power management methods are not capable of managing next generation hardware with multiple components and power management knobs
- The utilization of hardware components varies dynamically while a workload is executed




- Rate limiting approaches just result in more energy

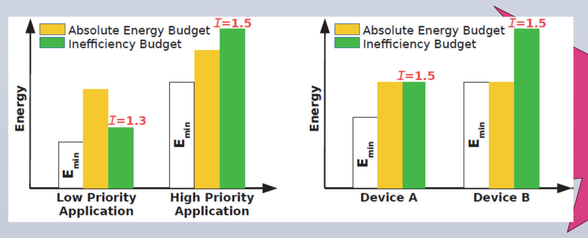



We Need Power-Agile Systems

## New Energy Constraint: Inefficiency

- $E_{min}$ – Minimum energy that the application could have consumed on the same device
- $E_{total}$ – Additional energy that the application can use to improve performance

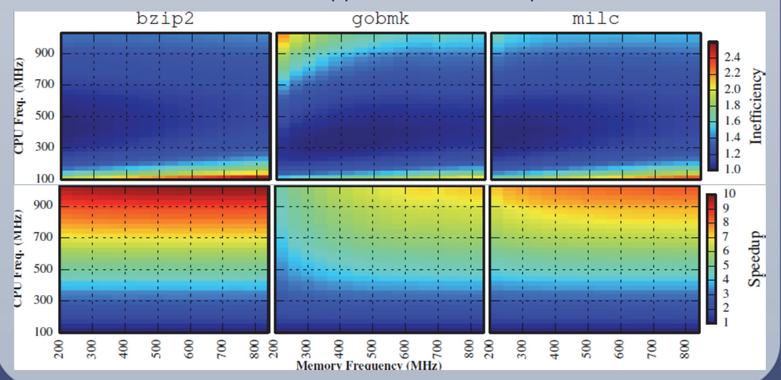
$$Inefficiency = \frac{E_{total}}{E_{min}}$$




Inefficiency is Device and Application Agnostic

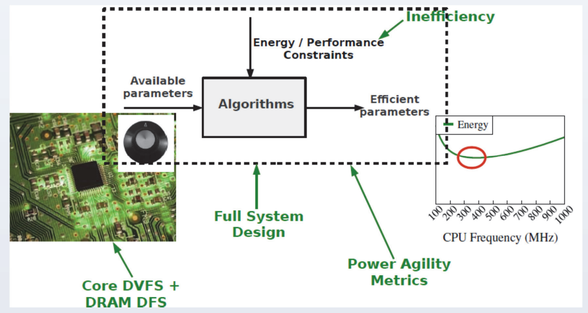
## Implementation Challenges

- Need cross component energy models to predict energy and performance for a new set of frequency settings
- $E_{min}$ is expensive to estimate using brute force methods
- Need management algorithms and system support for *Inefficiency*
- The best performing point is not necessarily the fastest point or highest energy point. It depends on the memory/CPU mix of application
- Need OS and hardware support for multi-component DVFS



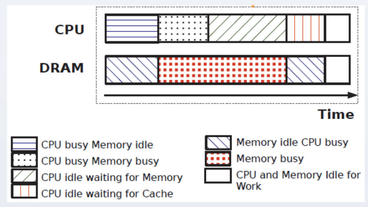

## Memory and Core DVFS under Inefficiency Constraints

### System Overview

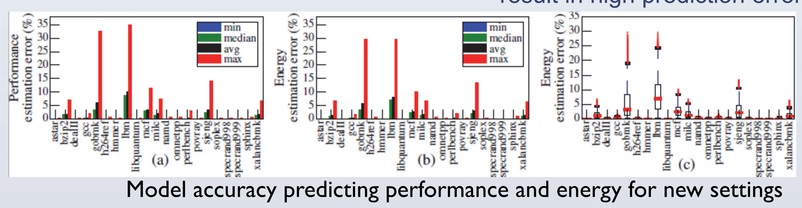



- The system runs a set of energy management algorithms to search for $E_{min}$ and maximize performance under the *Inefficiency* constraint
- To estimate energy and performance, the system collects workload statistics from hardware performance counters every fixed number of instructions and feeds them into the energy/performance model

### Cross-Component Predictive Energy and Performance Models



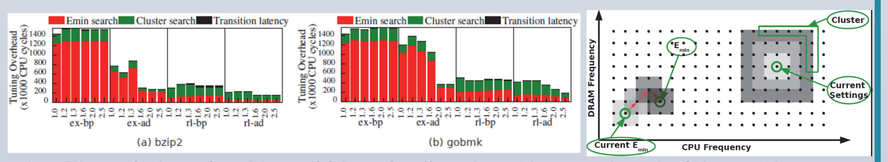

- Use *stats* collected from workload to specify *states*
- Scale *busy, idle, waiting,* and *overlap* states by the new target frequency
- Fixed amount of work makes scaling *easy*
- These models predict performance and energy for a new set of frequency settings with an average error of 4%
- More than 50% of predictions have less than 3% error
- Only two workloads have max errors above 10%
- Large jumps in frequency result in high prediction error




Model accuracy predicting performance and energy for new settings

### Reducing Overhead with Relative Search Algorithms

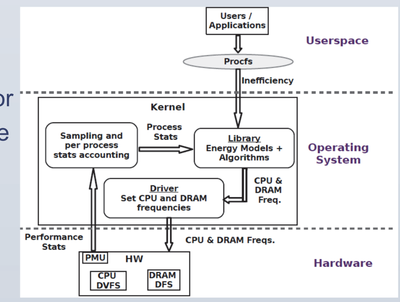
- The exhaustive algorithm (ex) searches all frequency settings for $E_{min}$
- Best performing (bp) does an exhaustive search for the best settings under the *Inefficiency* constraint
- Expensive and brute-force
- Relative (rl) uses the previous $E_{min}$ frequency settings as a starting point
- Stops immediately if new settings found within the Inefficiency constraint
- Adaptive algorithm (ad) skips the search if stats are unchanged (application stable)
- Includes logic to expand search and detect local minima




- The relative algorithms (rl-bp, rl-ad) reduce the overhead of the settings and $E_{min}$ search so that it is cheap enough to include in Android systems

### Simulator Implementation

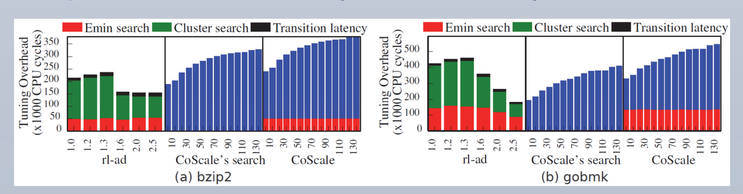
- Modified the gem5 full-system simulator
- Added hardware performance counters
- Added memory frequency scaling




- Modified the frequency governors in Android to implement both core and memory scaling
- Built energy performance models and algorithms into the Android kernel
- Provide *procfs* interface to set *Inefficiency*

## Agile-Android: Putting it All Together: Algorithms under Inefficiency Constraints

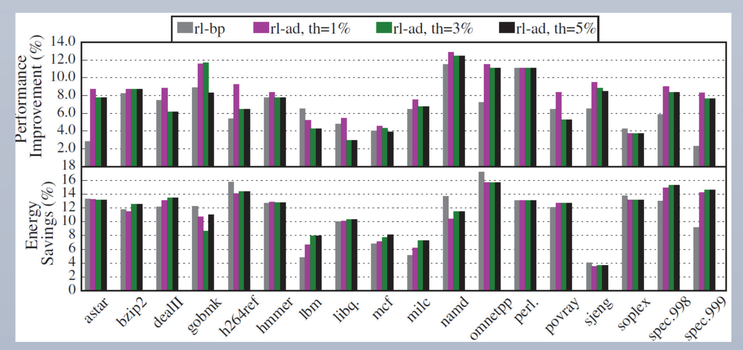
- Compared to CoScale (ASPLOS 2012) our search algorithms reduce overhead and stay under *Inefficiency* constraints
- CoScale uses performance bounds and not *Inefficiency* so we had to use the models to find the appropriate performance bound to stay under *Inefficiency*




### Summary

- Mobile systems must be agile and able to respond to changes in workload to manage energy
- *Inefficiency* provides a relative device and application agonistic constraint for energy management algorithms
- Our system has been implemented in the Android kernel and tested in full-system simulation

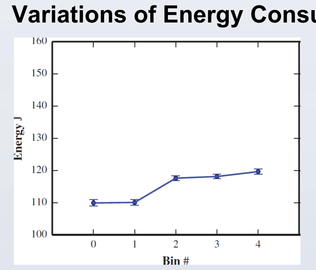
- Through the use of *Performance Clusters* the management algorithm can stay with the current settings if they are within a *performance threshold*
- The plot across all spec benchmarks normalized to *CoScale* for an *Inefficiency* of 1.3
- For a cluster threshold of 1% we see an average energy savings of 9.6% and performance improvement of 8.5% compared to *CoScale*
- Further analysis shows that *CoScale* picks the best performing settings more accurately than *rl-ad* but its search cost is high



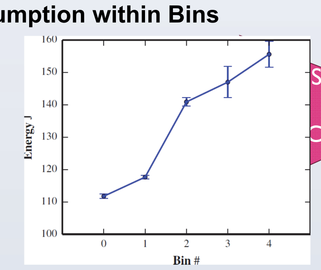

## Addressing Process Variations and Thermal Management for Energy-Efficiency

- The microprocessors for smartphones are not all the same; they are binned after manufacture with different supply voltages
- In addition, when phones get too hot they are throttled either to a lower frequency or fewer cores
- How does this affect energy efficiency and performance?
- We measured phones from different bins in a thermal chamber

### Variations of Energy Consumption within Bins




Default bin voltage

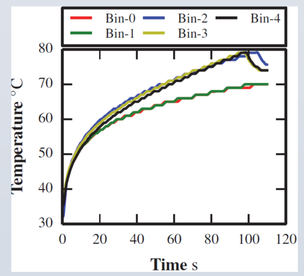


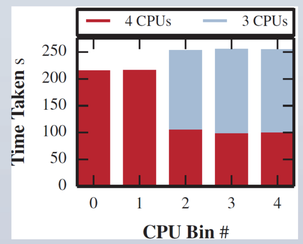

Uniform voltage

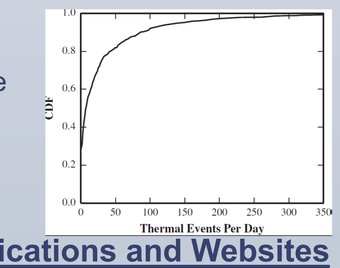
Smartphones are not Created Equal

### Thermal Emergencies Affect Performance

- Measured Energy and performance of Nexus 5 phones
- Without heat sinks or fans phones will heat up and be throttled, reducing performance
- We observed that more power hungry and leaky transistors more thermal emergencies (bins 2,3,4)
- Less efficient bins will spend more time in slower settings and take longer to execute an application
- Used the PhoneLab testbed to observe thermal emergencies per device per day
- Need better methods to reduce thermal emergencies
- Need methods to identify bins when purchasing phones




## For More Information: Publications and Websites


- *Guru Prasad Srinivasa, Rizwana Begum, Scott Haseley, Mark Hempstead, and Geoffrey Challen,* ***Separated by Birth: Hidden Differences Between Seemingly-Identical Smartphone CPUs****. HotMobile 2017.*
- *Rizwana Begum, Guru Prasad Srinivasa, Geoffrey Challen, and Mark Hempstead.* ***Algorithms for CPU and DRAM DVFS Under Inefficiency Constraints****. ICCD, 2016.*
- *Rizwana Begum, Guru Prasad Srinivasa, David Werner, Mark Hempstead and Geoffrey Challen.* ***Energy-Performance Trade-offs on Energy-Constrained Devices with Multi-Component DVFS****. IISWC, 2015.*
- *Rizwana Begum and Mark Hempstead.* ***Power Agility Metrics: Measuring Dynamic Characteristics of Energy Proportionality****. ICCD 2015.*
- *Rizwana Begum, Mark Hempstead, Guru Prasad, and Geoffery Challen,* ***New Interfaces for Achieving Power Agility on Mobile Devices****. HotMobile, Feb 2014.*
- *Geoffrey Challen, Mark Hempstead.* ***The Case for Power-Agile Computing****. HotOS 2011.*
- https://www.bluegroup.systems/projects/poweragility/
- https://sites.tufts.edu/tcal/