Application-Aware Power Management

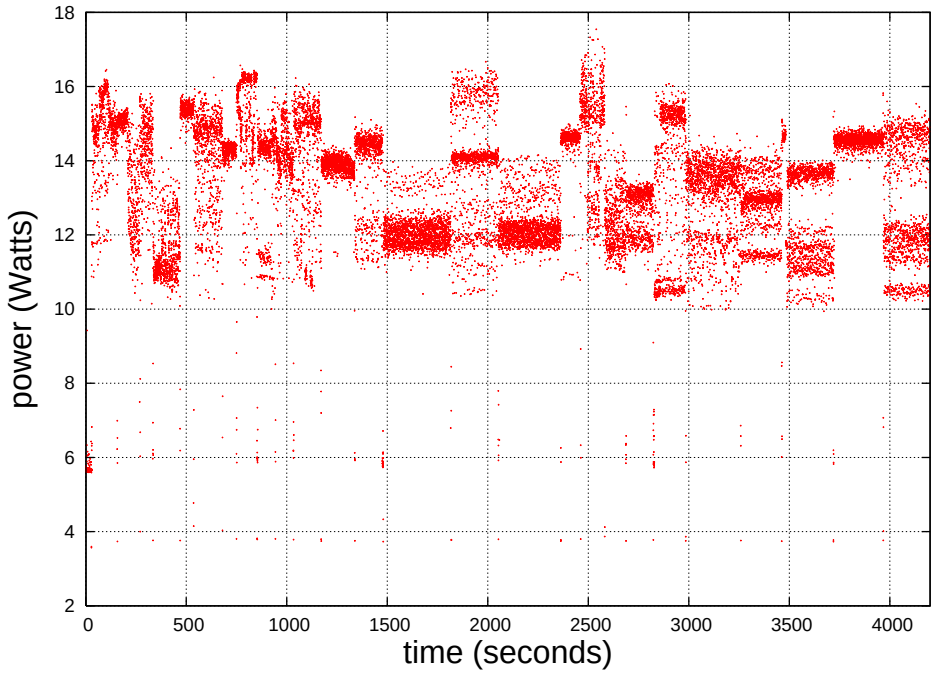
Karthick Rajamani, Heather Hanson, Juan Rubio, Soraya Ghiasi, Freeman Rawson*

Power-Aware Systems, IBM Austin Research Lab
**University of Texas at Austin*

Outline

- Motivation
- Application-Aware Power Management Methodology
- Infrastructure
  - Implementation
  - Evaluation
- Solutions
  - Power-aware Performance Maximizer
  - Performance-conscious Power Saver
- Conclusions

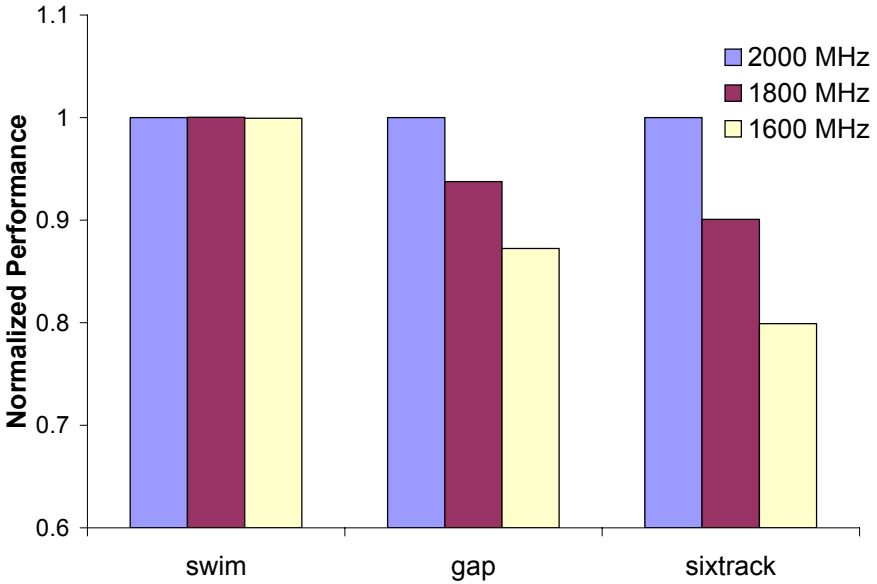
Power Consumption is Workload Dependent



- Power samples over time
- SPEC CPU2000 workloads
- Intel Pentium-M 755 processor running at 2GHz

- Variation in power consumption greater than 35% of maximum observed.
  - Consequence of more efficient processor designs.
- Comparable to impact from power management actions.
  - Workload-agnostic choice for power management settings would be overly conservative or aggressive.
- Variation is at constant 'system-perceived' utilization.
  - Conventional CPU utilization observations cannot be used to anticipate/explain this power variation

Impact of Power Management Mechanism is Workload-Specific



- Performance normalized to 2GHz performance.
- SPEC CPU2000 workloads – *swim, gap, sixtrack.*
- Intel Pentium-M 755 processor at 3 different *p-states (voltage-frequency pairs)*.

- Performance impact of power management action ranges from inconsequential to exactly proportional to extent of setting-change.
- Workload-agnostic exercise of power management mechanism would lead to unpredictable performance impact.
- Variation is at constant 'system perceived' utilization.
  - Conventional CPU utilization observations cannot be used to anticipate/explain this performance variation.

Benefits of being Application-Aware

- Reduce/eliminate margins provided for safe operation
  - Increased performance with reliability
- Better understanding of workload-specific impact from power management actions
- Enable more advanced power management techniques
  - Capping to explicit power levels
  - Explicit performance trade-offs/assured performance levels for power/energy reduction.

Proposed Methodology

- Three phases
  1. Monitor – workload behavior/activity.
  2. Predict/Estimate – given behavior under current conditions estimate impact of different power management actions.
     - E.g. activity under new voltage and frequency – change in power, impact on performance.
     - Using custom, well-characterized models for activity, power and performance estimations across operating state changes.
  3. Manage – given the optimization goal and constraints on power, performance etc change to operating points that best meet constraints.

- Iterate periodically to
  - Adapt to change in workload characteristics.
  - Adapt to change in constraints and/or environmentals.

Infrastructure for Application-aware Power Management

- Monitor
  - Using performance counters for low overhead, fast access, adequate information.
    - Decoded instructions, Retired Instructions, DCU Miss Outstanding Cycles (or Bus Transactions for Memory).
  - Custom Linux and MS Windows drivers – samples counters up to 400 times a second with no impact on workloads relying on system-provided timer facilities.
    - Larger variance on MS Windows.
- Actuate
  - Low-overhead drivers for voltage and frequency (and clk-throttle) control – write to MSRs (*cpufreq-based* in Linux and custom for MS Windows).
- Manage
  - User-level application running under real-time/high priority that periodically monitors, estimates, assesses constraint compliance and changes operating states as required.

Estimation Models

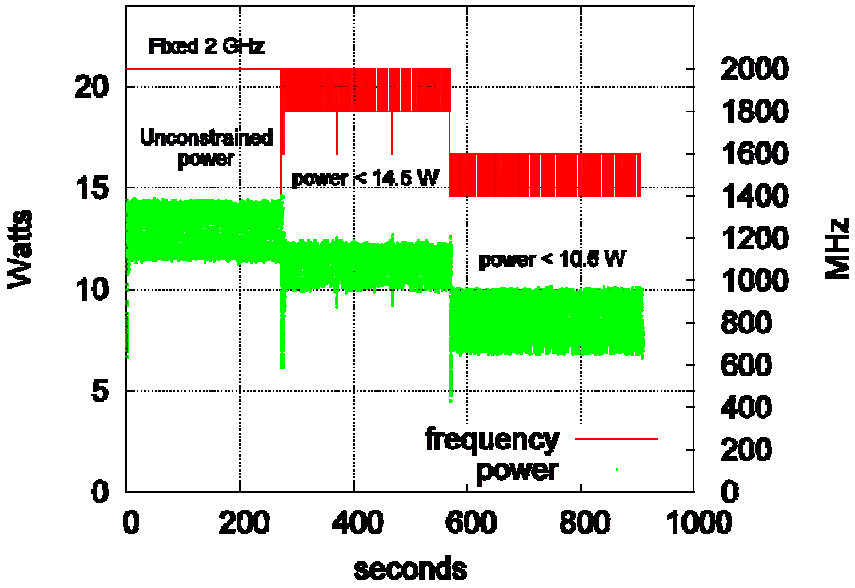
- Based on characterization using custom loop-based micro-kernels
  - Compute versus memory intensive
  - Varying usage of memory hierarchy
- Power Estimation
  - Estimation for fixed activity – operating-state-specific linear models based on *Instructions Decoded/cycle*
- Activity Estimation
  - *Conservative* estimation of impact on activity where activity is used to estimate power.
- Performance Estimation
  - Performance impact equated to impact on Instructions retired per second.
  - Models for estimating Instructions retired per cycle using corresponding DCU Miss Outstanding cycles counter and state change information.

Evaluating Power Management Solutions

- SPEC CPU2000 workloads
  - Variability across different runs addressed using median-execution-time run among three repetitions.
- Performance metric - Execution time of each benchmark
- Power delivery to chip sampled at sense resistors between VRMs and chip at 100 Hz
  - Oversubscription tracked for 100 ms intervals.
  - Energy savings also inferred from sampled power.

Power-Aware Performance Maximizer

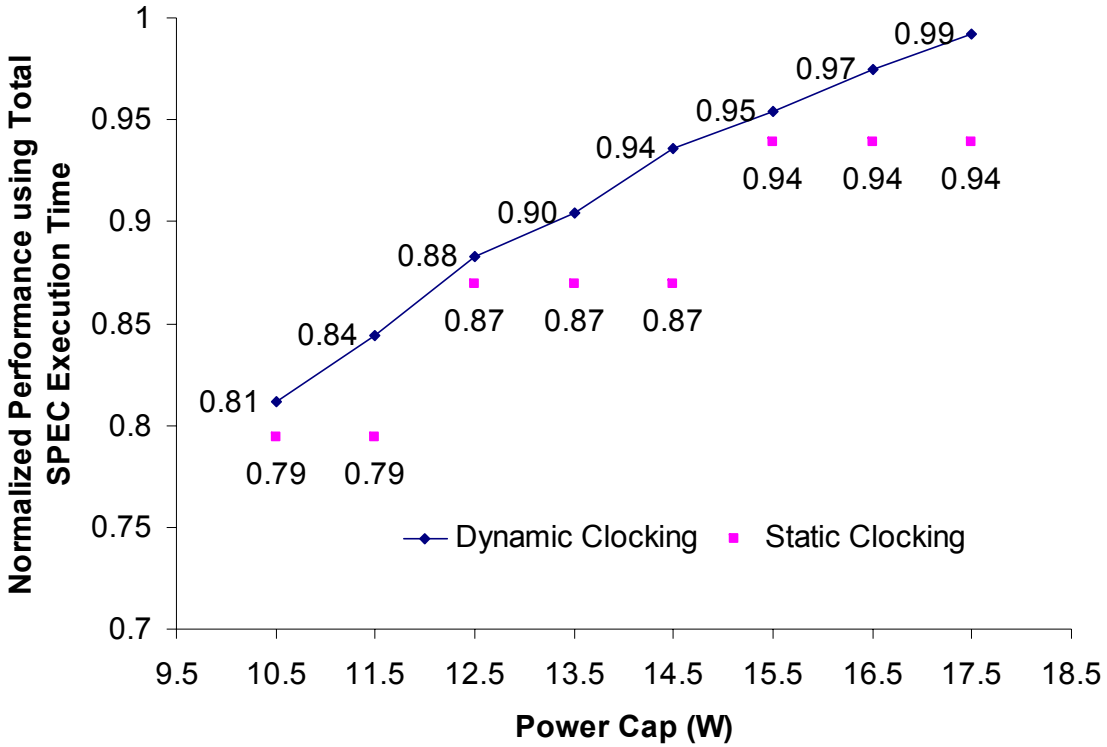
- Exploit DVFS capability to obtain higher performance.
- Dynamically trade-off application's activity-slack for increased performance using higher frequencies.
  - Where power-cap is set using worst-case workload, most real workloads can exploit higher frequencies for same power cap.
- Continue to maintain power consumption below specified levels.
  - Adapt to changing power constraints.



Implementation for Performance Maximizer

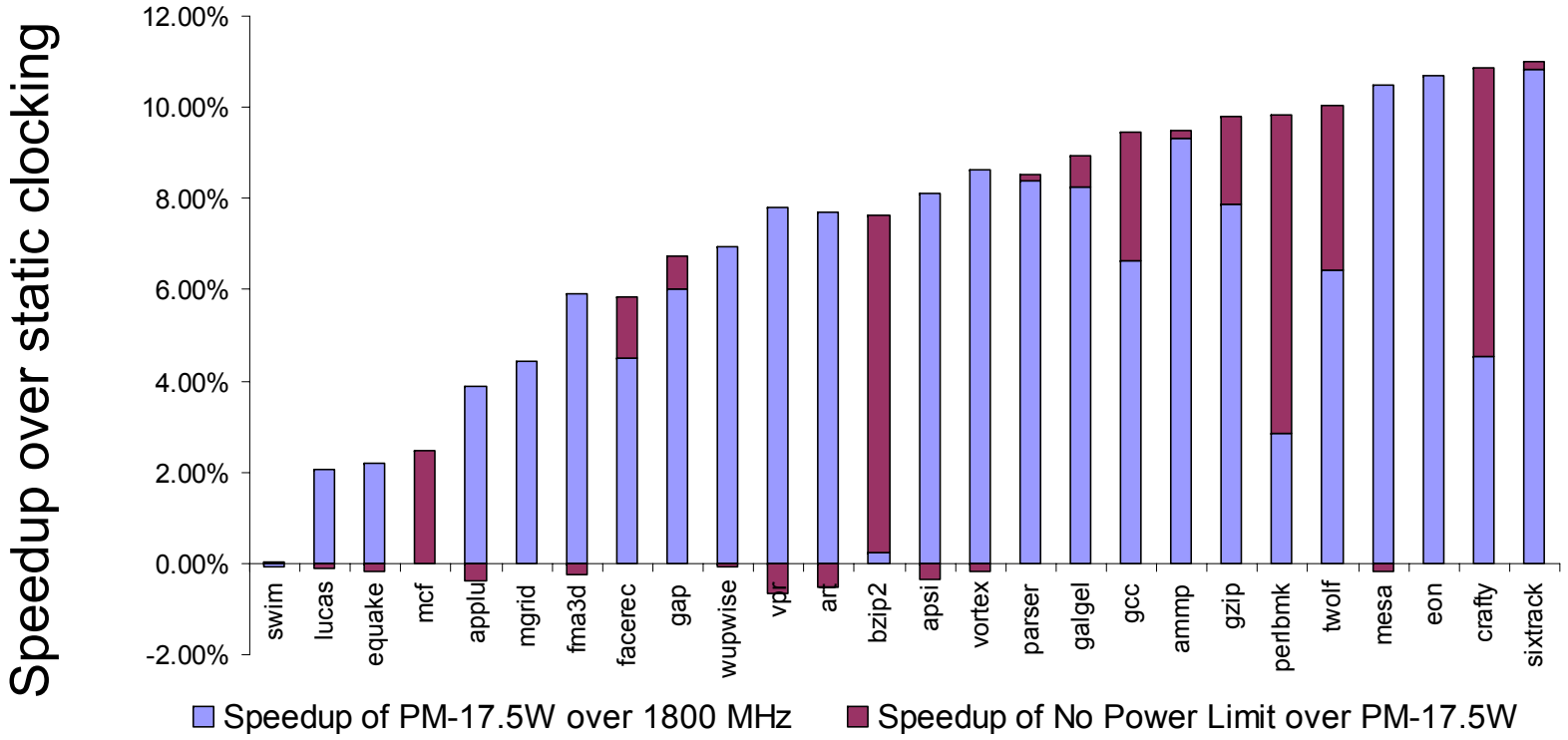
- User-specified explicit power cap – maximize performance by dynamically choosing maximum frequency possible within power cap.
- Three phases
  - Monitor *Instructions Decoded* counter to get Decoded Instructions per cycle (DPC).
    - Estimate DPC and then power for each *p-state* (frequency).
  - Choose as new frequency the highest frequency for which estimated power is still below desired limit.
- Models
  - Power(f) = DPC * α(f) + β(f)
  - DPC(f') = DPC(f) * f/f' , f' < f; DPC(f) , f' > f.

Results for Performance Maximizer



- Performance compared to no-power constraint performance.
- Static-clocking frequency determined by *hot loop* power.
  - Close for subset of limits, but not feasible for supporting multiple limits or varying limits

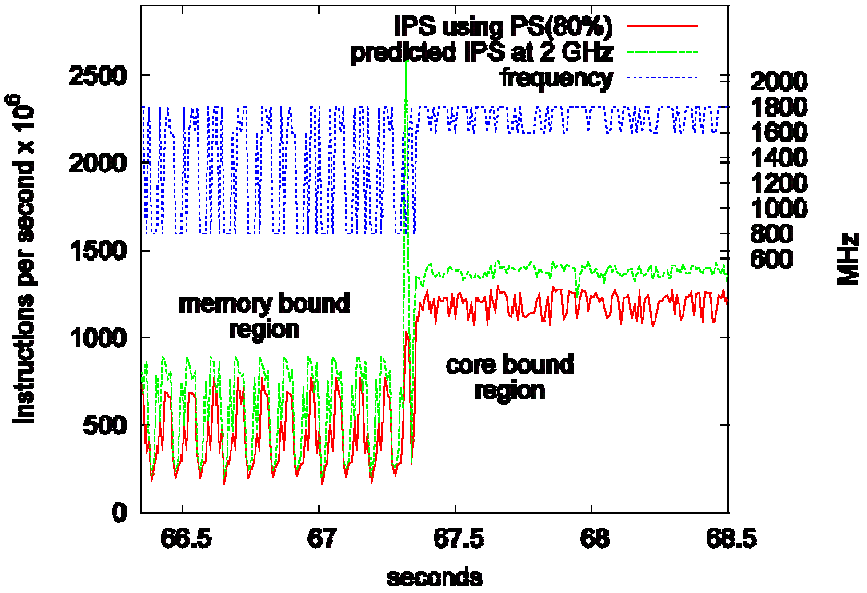
Results for Performance Maximizer



- High memory (DRAM) and resource stall applications to left.
- High core-active applications are power-limited e.g. crafty, perlbmk, bzip2 – unable to reach *unconstrained* potential.

Performance-conscious Power Saver

- Exploit DVFS capability to generate energy savings.
- Dynamically trade-off *user-specified slack* in performance requirement.
  - User-given slack in performance requirement translated into appropriate usage of lower frequencies.
  - Reducing frequencies allow super-linear reductions in both active/switching and leakage power (by using lower voltages).
- Continue to maintain performance at user-specified slack levels.
  - In contrast, most prior energy saving solutions exploit 'natural' slack in performance requirements – smaller savings.



Implementation for Power Saver

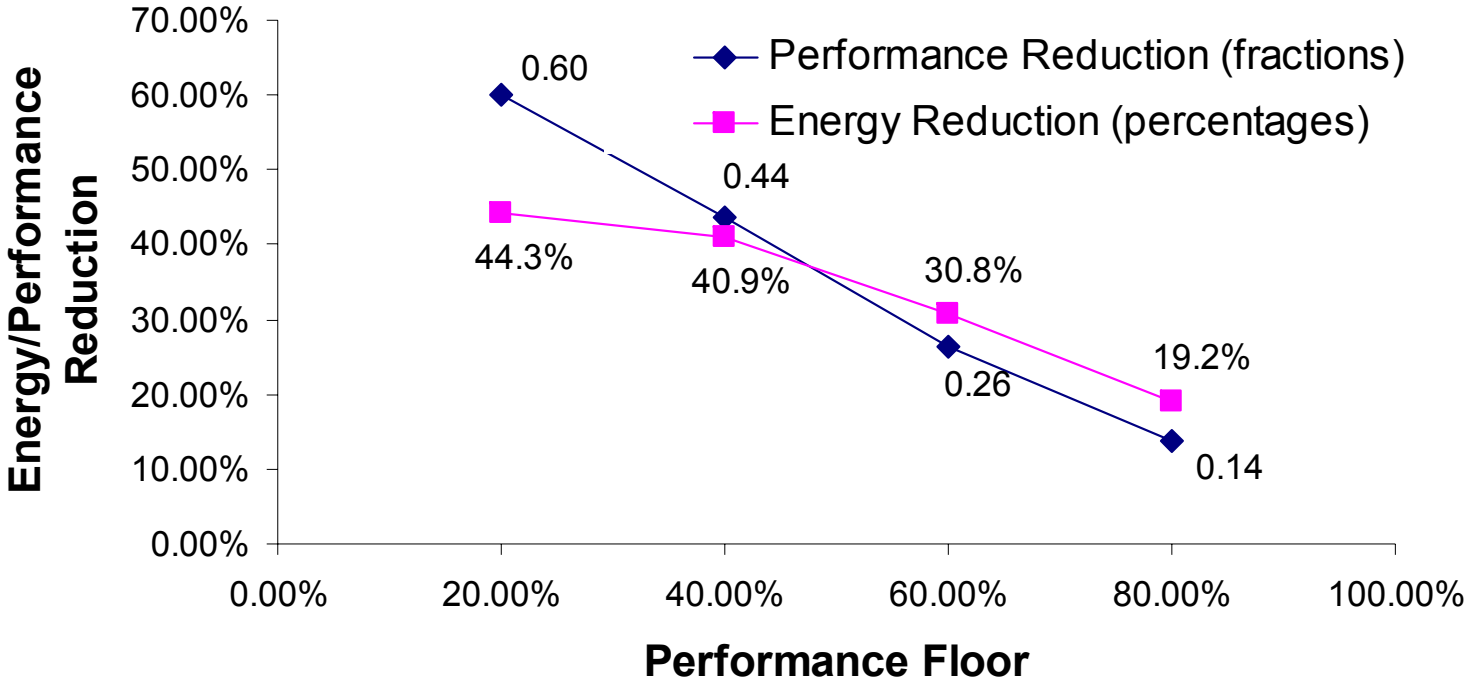
- Requirement on performance – X% of peak performance
  - Equated to Retired Instructions per Second (IPS) that is X% of the IPS at peak frequency.

- Three phases
  - Monitor *Instructions Retired* (IPC/IPS) and *DCU Miss Outstanding Cycles* (DCU) counters.
  - Estimate IPS at each *p-state* (frequency) including peak.
  - Choose as new frequency the lowest frequency for which estimated IPS is still above desired lower bound.

- Model
  - $IPC(f') = IPC(f)$, $DCU/IPC < 1.21$

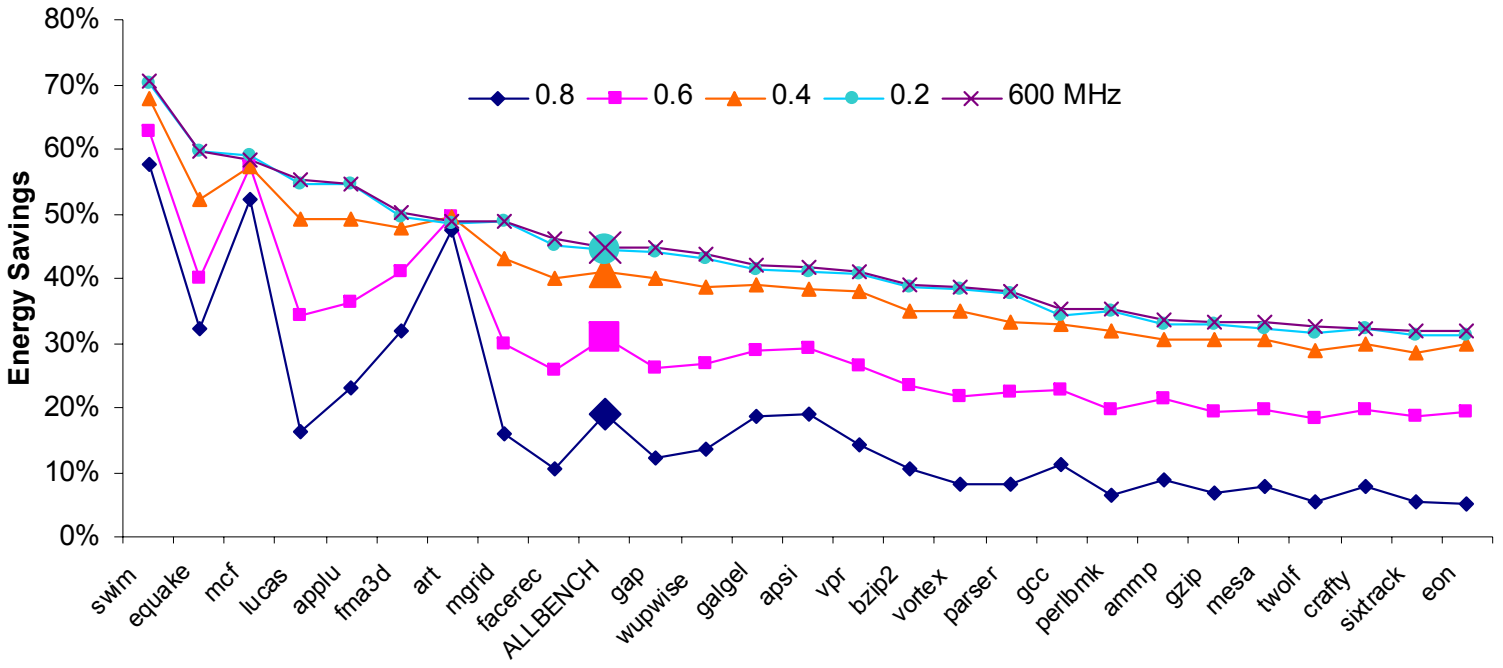
    $= IPC(f) * (f/f')^{0.81}$, $DCU/IPC \geq 1.21$

Results for Power Saver



- Performance reduction and energy reduction compared to full speed.
- Meets corresponding performance floor
  - Discretized p-states – one hurdle to extracting all allowed performance reduction.

Results for Power Saver



- 600 MHz – upper bound on savings, sorted by savings@600MHz
- Workload characteristics dictates savings
  - Memory-bound to the left of ALLBENCH – exploit lowest frequencies.
  - Core-bound to right of ALLBENCH.

Conclusions

- Power-efficient/modern architectures lead to highly workload-dependent power consumption; conversely, impact of power management actions are also highly workload-specific.
- Application-aware solutions can exploit superior knowledge to increase performance and obtain greater energy savings.
- We propose a practical methodology for incorporating application-awareness in power management.
  - Our proposed infrastructure is simple, low-overhead and easily realizable – practical.
- Application-aware methodology enables new, sophisticated power management solutions.