Cross-Component Energy Management of Energy Constrained Devices

Rizwana Begum

Next generation smart phones must be energy efficient to provide long battery life. Over the past few decades, researchers have built a large body of research focusing on conserving energy in mobile environments. Most prior work has optimized energy consumption of single components in isolation (e.g., CPU or memory or network etc.), while a few efforts that have focused on system-wide energy conservation used performance constraints as their input. Managing energy of individual components without considering the cross-component effects may have a negative impact on performance and energy [5]. Similarly, operating mobile devices under performance constraints doesn’t provide a guarantee of longer battery lifetime. My research focus is to design a system that is capable of tuning multiple components of a device simultaneously to deliver best performance staying under given energy budget.

Energy management of multi-component devices is explored before. For example, CoScale optimizes energy consumption of servers by scaling frequency of cores and DRAM [5], while Chidambaram et al. scale frequency of GPU and IP cores in addition to CPUs and DRAM to reduce energy consumption of handhelds [6]. These efforts are focused on conserving energy under performance constraints such as acceptable performance loss or the available slack in the system. Performance constraints are good for server systems that are wall powered and have quality of service requirements. However for mobile devices, it is appropriate to operate under energy constraints to prolong and guarantee a minimum battery lifetime. A few efforts that work under energy constraints either use absolute energy or rate of energy consumption as their constraints. Both of these constraints are difficult to put into practice as the absolute energy needs vary across applications and devices, so do these constraints. I introduced the concept of Inefficiency, a relative energy constraint—used to specify the amount of additional energy that can be used to improve performance [4].

Energy management approaches designed to operate under performance constraints can’t be directly extended to work under energy constraints; Computing the performance bound for a desired energy savings target is a daunting task as it requires oracle knowledge of applications and devices [2]. I designed a holistic approach that dynamically tunes voltage and frequency of cores and only frequency
of DRAM to deliver best performance staying under given inefficiency budget. Our system profiles the applications periodically at runtime, uses the algorithms and cross-component performance and energy models that we developed to select the optimal settings that deliver best performance staying under given inefficiency budget.

Performance and energy models developed in the past, that consider the coupling between cores and DRAM are used only for in-order cores and are not validated against ground truth [5]. I developed models that predict performance and energy at target frequency and voltage settings DRAM using the application statistics collected at a given pair of frequency settings. The cross-component models consider the impact of scaling frequency of one component on the performance and energy of the other component. The performance model works by collecting the time spent in busy, idle and waiting states for each component and scaling it systematically to predict execution time. The energy models use empirical power models for cores and micron power models for DRAM—extended to integrate frequency scaling—to predict energy consumption. Our models are validated across 21 SPEC benchmarks and have average error of 4% and maximum error of less than 10% [2].

Energy management algorithms play a pivotal role in our system. Algorithms use the models to select the optimal frequency settings that deliver best performance under given inefficiency budget. Inefficiency is defined as the ratio of \( E_{total} \) and \( E_{min} \), where \( E_{total} \) is the total energy that the application can consume and \( E_{min} \) is the minimum energy that the application could have consumed on the same given device. Our algorithms first estimate \( E_{min} \) and then multiply it with given inefficiency budget to compute \( E_{total} \). Using the models, cluster of frequency settings that result in energy consumption of less than or equal to \( E_{total} \) are filtered. Among the filtered settings, frequencies that deliver best performance are chosen and system is transitioned to the selected settings. The algorithms search for 1) \( E_{min} \) and 2) the cluster of possible frequency settings, therefore have higher tuning cost compared to the algorithms operating under performance constraints—that only search for later. I designed relative and adaptive search algorithms that have 24% lower tuning cost compared to the state of the art performance constrained system.

Our system is capable of adapting it’s energy consumption with changing workload characteristics by tuning performance-energy knobs of cores and DRAM. In general, we call the ability of the systems to detect, select and transition to efficient power settings dynamically during the execution of the application as Power Agility [3]. I introduced the metrics to measure how well a device is achieving power
agility and classified the metrics further to identify if the device is inefficient in selecting the right settings (Selection Power Agility) or transitioning to the selected settings (Transition Power Agility) [1].

References


