

Adapt&Cap: Coordinating System- and Application-Level Adaptation for Power-Constrained Systems

Can Hankendi and Ayse Kivilcim Coskun
Boston University

Henry Hoffmann
University of Chicago

Editor's notes:

The paper proposes an adaptive framework for modern data centers that jointly manages application- and system-level (dynamic voltage and frequency scaling) adaptation to improve energy efficiency of multicore servers. The results of applying this framework show that significant power savings and performance improvements are possible with respect to current data center management techniques.

—David Atienza, *Ecole Polytechnique Federale de Lausanne*

■ **ENERGY-RELATED COSTS** are among the major contributors to the total cost of ownership of today's data centers and high-performance computing (HPC) clusters. Future computing clusters are required to be energy efficient to meet the continuously increasing computational demand. Furthermore, administration and management of the data center resources have become significantly complex due to an increasing number of servers installed in data centers [2]. Therefore, designing autonomous and adaptive techniques to manage data center resources is essential to achieve sustainability.

The achievable maximum performance of a computing cluster is determined by infrastructural/

cost limitations (e.g., power delivery, cooling capacity, electricity cost) and available hardware resources (e.g., CPU, disk size). Optimizing the performance under such constraints (i.e., power, compute resources) is critically important to improve energy efficiency and reduce computing costs. As power minimization and performance requirements are often

competing objectives, efficient management of these limited resources is a complex task. Continuously increasing demand on data center resources causes data center administrators to be more conservative on the management of the power resources due to power delivery and cost limitations. Therefore, constraining the peak power consumption of the servers via power capping is becoming a common practice for managing the energy costs and to comply with the power delivery limitations [1].

The aforementioned interplay between power and performance requirements and constraints adds to the complexity of data center management. In order to reduce the administration and management costs, designing adaptive solutions has become necessary. Traditional adaptive solutions employ system-level management knobs to comply with the power and performance requirements [12]. These system-level adaptive solutions use control knobs such as voltage/frequency selection (i.e., DVFS) or turning on/off cores. However, system-level

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/MDAT.2015.2463275

Date of publication: 31 July 2015; date of current version: 18 January 2016.

solutions lack the ability to optimize the performance of the application. Adaptive applications address the performance optimization problem by dynamically configuring application parameters depending on the hardware properties and the performance goals [7]. As application- and system-level decisions impact both the performance and the power consumption, uncoordinated decisions at these two levels can significantly hurt the overall energy efficiency of the system.

In this work, we propose a unified framework that takes advantage of both system- and application-level adaptability to: 1) improve performance under power caps; and 2) reduce power consumption under performance constraints. Our specific contributions in this paper are as follows.

- First, we demonstrate how to improve the power/performance tradeoff space by unifying system- and application-level adaptation.
- We propose a unified framework Adapt&Cap, which combines system- and application-level adaptations to improve performance while reducing the power consumption.
- We implement Adapt&Cap on real servers and demonstrate up to 27% power reduction and 2.7× performance improvement compared to system- or application-level only adaptation.

Next we provide our analysis on adaptive applications and systems. In Section III, we present the main components of Adapt&Cap and provide the details of our experimental setup. We present our results collected on real servers in Section IV.

Benefits of coordinating system- and application-level adaptation

Controlling the tradeoff between an application's accuracy and performance is a widely studied area [5]. Design of applications that can expose various control knobs to provide control over accuracy and performance targets is an emerging area of study [3]. Adaptive applications enable dynamic reconfiguration of execution parameters to meet user-defined performance and accuracy constraints [5].

On a cloud environment, where resources are limited, power, performance, and accuracy constraints are expected to be dynamically changing

due to changing user requirements, energy pricing, and cost management policies. Adaptive applications can meet these dynamically changing performance or accuracy targets by modifying a set of selected application parameters at runtime. Application parameters vary depending on the type of the application. For instance, for an image processing application, these parameters can be block sizes, motion search ranges, or color matrices. An adaptive application iteratively modifies its parameters (i.e., application control knobs) until the user-defined constraints are met.

Although adjusting application parameters can be utilized to meet the performance and accuracy constraints, the impact of modifying the application parameters on the power consumption is limited. In order to meet the power constraints, system-level management techniques are necessary. Various system-level power management techniques have been proposed that utilize control knobs such as DVFS or adjusting the number of active cores [9], [11]. However, these power management techniques are agnostic about the potential adaptive capabilities of the applications. Independently managing system and application adaptation leads to uncoordinated management of power and performance requirements. This interaction may lead to destructive interference, where the application and system behavior oscillate and fail to meet both the performance constraint and the power cap with the desired accuracy, as we show in a later section.

In Figure 1, we show the power and performance (i.e., seconds elapsed processing each frame) for ×264 from the PARSEC suite [6] for three cases: 1) where we use adaptive capabilities only at the application-level (application level only); 2) only at the system-level (system level only); and 3) at both application and system level (coordinated). As Figure 1 shows, application-level decisions have minimal impact on the power, while providing the ability to adjust the performance for a wide range of targets. On the other hand, system-level decisions have a significant impact on power consumption, while providing a narrower performance range with respect to the adaptive application. Unifying the system- and application-level adaptability provides the most efficient tradeoff curve for the power and performance space. This result in Figure 1 motivates the idea of using

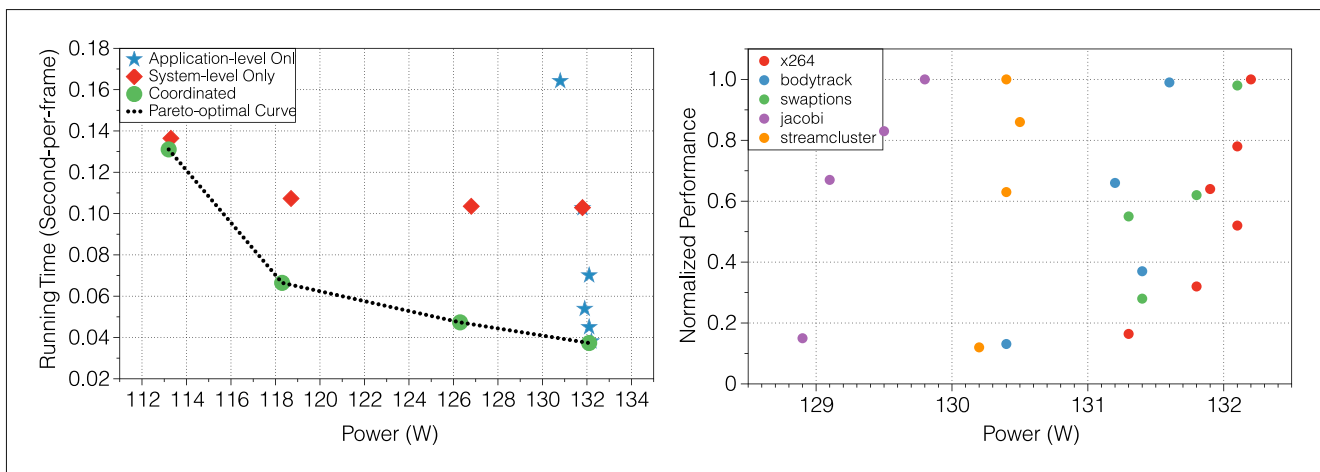


Figure 1. Power and performance tradeoff space for various adaptive techniques on Intel Xeon E5 multicore server when running $\times 264$ (left). Proposed coordinated management extends the Pareto points to a more efficient operating point. Application adaptation has minimal impact on power consumption on all five applications (right).

adaptive capabilities of applications to push the performance while reducing the power consumption through system-level control.

Adapt&Cap: unifying system- and application-level adaptation

In this section, we present the details of the proposed adaptive framework Adapt&Cap. Adapt&Cap combines an application-level adaptive framework (i.e., Heartbeats) with a system-level adaptive power management framework to: 1) maximize the performance under power constraints; and 2) minimize the power consumption under performance constraints. We first discuss the details of the application-level framework for adapting to changing performance and accuracy targets. We then introduce the system-level adaptation framework that adjusts the level of resource usage to closely follow the power constraints.

Application heartbeats

In order to create adaptive applications, we use the previously proposed PowerDial framework [5]. PowerDial dynamically adapts applications to changing performance and accuracy constraints by modifying configuration parameters at runtime. PowerDial first identifies the application parameters, then uses these parameters as runtime control knobs to adjust the tradeoffs between performance

and accuracy. PowerDial relies on the application heartbeats API [3], which dynamically reports performance and accuracy at runtime, to adjust its decisions. Furthermore, it allows users to dynamically request certain performance and accuracy targets during execution.

In order to determine the control variables for the parameters, PowerDial generates a state table that stores various configuration parameters to create the adaptive version of a statically configured application. PowerDial generates the state table at compile time by profiling the applications on representative inputs provided by the user. For each combination of parameter settings, PowerDial profiles all representative inputs and records the speedup and accuracy loss. It then computes the average speedup and accuracy loss across all inputs, and sorts these average values into the set of Pareto optimal states. By convention, PowerDial stores configurations so that the slowest configuration is the first entry in the state table and the fastest configuration is the last.

In this work, we use two types of adaptive applications that are created with PowerDial and with a loop-perforation technique [8] to show the applicability of our technique to virtually any software that has adjustable parameters. Other application domains that rely on iterative algorithms, such as graph applications, are also good candidates to create adaptive versions.

vCap dynamic capping framework

vCap is a dynamic power capping framework that we designed for virtualized environments [4]. vCap is built on top of VMware ESXi and manages the resource usage of virtual machines (VMs) to maximize performance while meeting the power and user-defined application performance constraints. vCap is designed for two main purposes: 1) to closely meet power constraints; and 2) to achieve desired performance targets through utilizing the system-level resource usage control knobs (i.e., CPU resource limits and active number of cores). vCap receives both power and performance feedback and estimates the performance/power tradeoff during runtime to make decisions. Based on the given power and performance constraints, vCap first estimates the minimum number of active cores that are required to meet the power constraints and packs the applications threads on to a smaller number of virtual cores to reduce the power consumption. vCap then utilizes CPU resource limit control knobs to fine-tune the performance and power tradeoff. CPU resource limits are available on most virtualization solutions and essentially limits the CPU time of the VMs through time slicing.

Adapt&Cap

Adapt&Cap maximizes performance through utilizing adaptive applications and minimizes the power consumption by employing system-level management. Adapt&Cap is built on top of the vCap framework and extends the capabilities of vCap by taking advantage of the performance optimization capabilities of the adaptive applications. In Figure 2, we illustrate the overall flow of the Adapt&Cap framework. Our framework accepts two types of constraints either from the user for performance (i.e., heartbeat rate) or the system administrator for power (i.e., power cap). Both power consumption and the heartbeat rates are periodically fed to the closed-loop controller to adjust and tune its decisions.

Figure 3 provides the pseudocode for the Adapt&Cap framework that consists of three major steps: 1) configuring the adaptive application (*Configure*); 2) controlling the power consumption (*PowerControl*); and 3) meeting performance constraints (*HBControl*). Each adaptive application comes with built-in state tables, which include various combinations of the application parameters. As a first step, Adapt&Cap discovers the adaptive

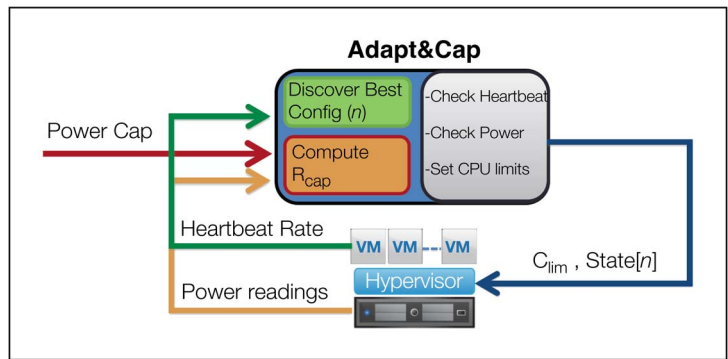


Figure 2. Adapt&Cap reads heartbeat rates and power measurements, and chooses the amount of CPU resources required and the optimum application state.

states of the application within the code and chooses the state that achieves the highest performance. It then measures the performance and power consumption at the highest state (i.e., n). Based on these measurements at the highest performance state, we derive the dynamic power consumed per heartbeat (P_{HB}), and the CPU resource used per heartbeat (CPU_{HB}) assuming that the power and performance are linearly correlated. A later fine-tuning stage enables compensating for the potential inaccuracies caused by the linearity assumption.

After deriving the power/performance relationship of an application at its best performing configuration, Adapt&Cap individually checks the power and performance constraints to adjust the CPU usage limits (CPU_{limit}) and to make thread packing decisions. For a given power cap, Adapt&Cap first computes the maximum achievable performance (HB_{cap}), then it computes the maximum amount of CPU resources that will not violate the power constraints (CPU_{limit}). Based on the CPU_{limit} , we derive the minimum number of active cores that can provide enough CPU resources to meet the computed CPU_{limit} .

In order to compensate for the potential inaccuracies in the power and performance estimation, Adapt&Cap performs fine-tuning on its decisions. In case of a tracking error that is larger than ϵ , Adapt&Cap iteratively adjusts the CPU_{limit} . We start with a granularity of 1-core (i.e., resource limits corresponding to 1-core) to increase or decrease the CPU resources allocated. Until the tracking error is within the ϵ range, we increase the granularity of fine-tuning by dividing the adjustment range with the number of iterations. We use 2 W as our ϵ value.

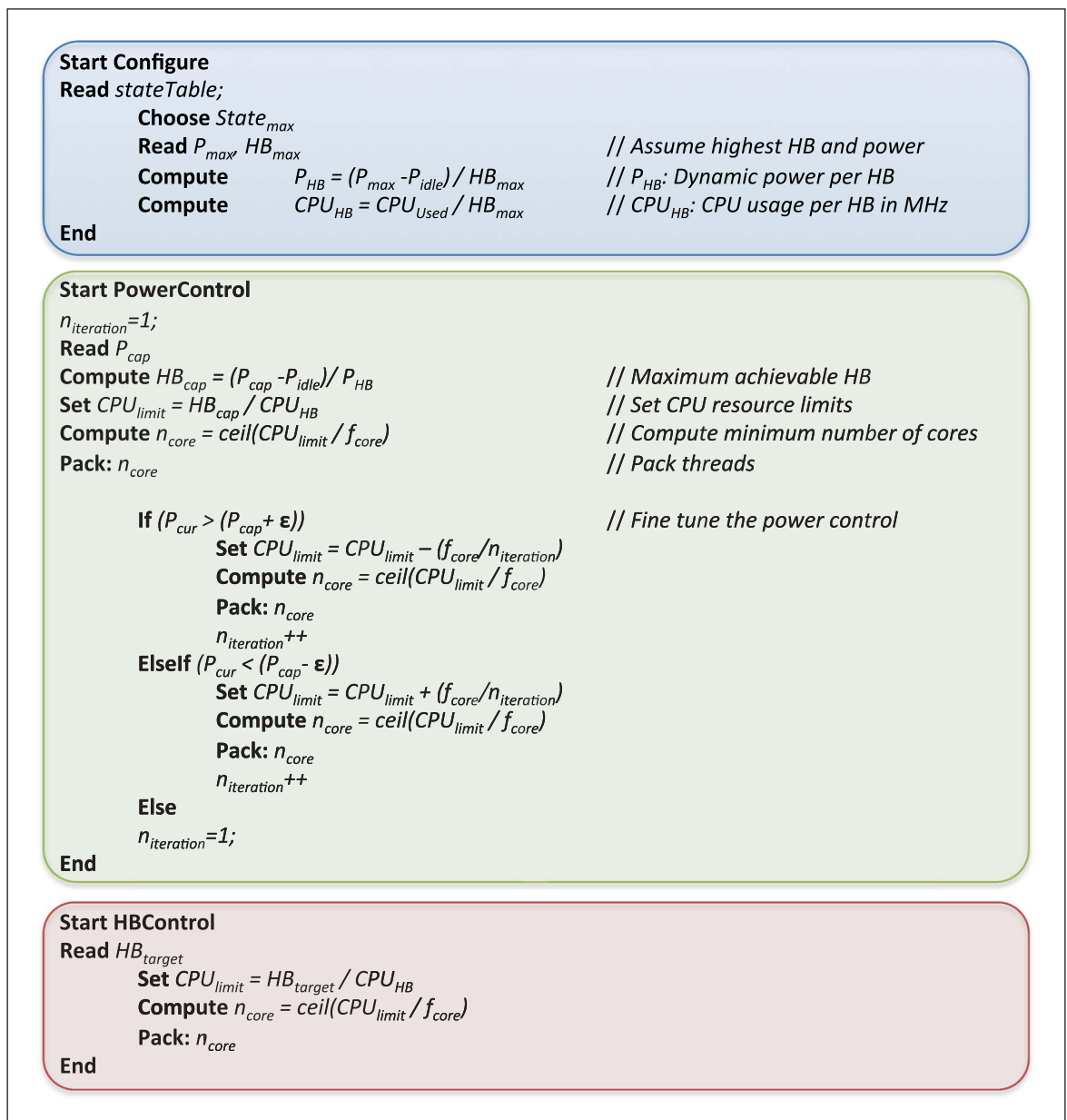


Figure 3. Pseudocode for Adapt&Cap control modules. Adapt&Cap first discovers the higher performance application state (blue box), then periodically checks power (green box) and performance (red box) requirements to adjust its decisions.

As a result, in each iteration, we achieve a finer control on the power consumption. Similarly, for the performance control, Adapt&Cap gets the performance requirement as an input (HB_{target}), and computes the necessary amount of CPU resources that will meet the performance constraints. The overhead of monitoring and management is around 1%.

The power and performance control mechanism of Adapt&Cap is implemented to prioritize the hard

constraints (i.e., power) over soft constraints (i.e., performance). Therefore, decisions to improve the performance are overwritten in case of a power violation to obey the power constraints. As oppose to disjoint management schemes, Adapt&Cap is an opportunistic approach that does not solely meet the requirements in one dimension, but also targets to improve the efficiency in both power and performance dimensions.

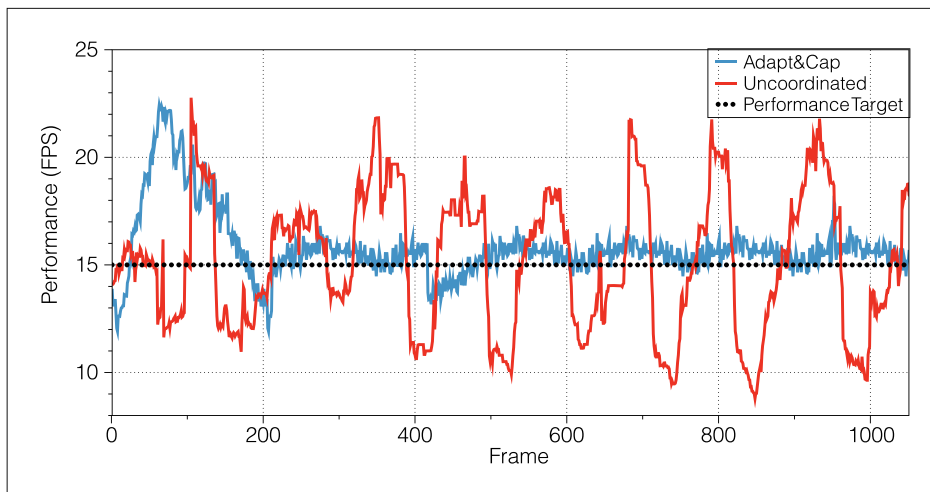


Figure 4. Uncoordinated approach shows oscillatory behavior, as system and application adaptation controls are not aware of other decisions that impact the performance of the system significantly.

Experimental setup

In this work, we target multicore-based servers that run multithreaded applications. Our experimental setup includes an AMD 12-core Magny Cours (Opteron 6172) and an eight-core Intel Xeon E5 (2603). Magny Cours consists of two six-core dies attached together on a single chip. Each six-core die includes a 12-MB shared L3 cache, and each core has a 512-KB private L2 cache. All cores also share a 16-GB off-chip memory. The other server is a multisocket system that includes two four-core Intel Xeon E5-2603 processors. Each core has 32 KB of private L1 and 256 KB of private L2 cache; each processor (i.e., four cores) shares 10 MB of L3 cache, and all eight cores share a 32-GB off-chip memory. We virtualize both servers with the VMware ESXi 5.1 hypervisor and create VMs with Ubuntu Server 12.04 guest OS.

We utilize the application heartbeats API to measure the application-specific performance metrics. We measure the system power by using a Wattsup PRO power meter with a 1-s sampling rate. As the total system power determines the electricity cost of a server, we focus on the system power rather than the component power (i.e., processor, disk, etc.). In all of our experiments, we only evaluate the parallel phases of the applications, as the parallel phase of multithreaded applications dominates the application execution time in real computing clusters. To evaluate our technique, we use four adaptive applications from PARSEC 2.1 benchmark suite (i.e., bodytrack, swap-

tions, streamcluster, and $\times 264$) that are generated by the PowerDial system and another application that is generated by the loop perforation technique (i.e., Jacobi) [8]. For all baseline experiments, we use the default application parameters that are built in to the original code of the application.

Results

In this section, we present the benefits of the Adapt&Cap framework on real-life servers. We test our framework under two scenarios: 1) dynamically changing performance constraints; and 2) dynamically changing power caps. First, we show that Adapt&Cap provides lower power under dynamically changing performance constraints. We then show that Adapt&Cap can provide higher performance under the same power constraints when compared to algorithms that do not leverage the adaptive features of the applications (i.e., vCap).

Power tracking performance

In Figure 4, we show the runtime behavior of Adapt&Cap under dynamically changing power caps on the Intel Xeon server when running the adaptive version of $\times 264$. We randomly change the power cap every 8 s between 115 and 135 W for the Intel server, and between 85 and 165 W for the AMD server based on the power dynamics of these two different systems. Adapt&Cap reacts to dynamically changing power constraints by tuning the CPU resource usage and the number of active cores

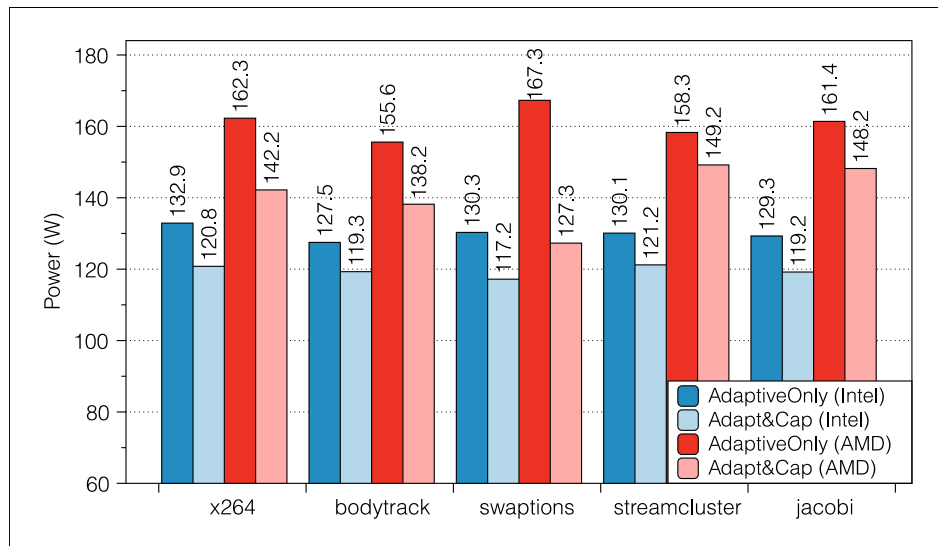


Figure 5. Comparison of power consumption for Adapt&Cap and only adaptive application under dynamically changing performance constraints. Adapt&Cap reduces the power consumption up to 27% compared to only application-level adaptation.

and, in this way, it accurately tracks the power caps. The absolute value of the average tracking error is 1.8 and 1.2 W for the Intel and AMD machines, respectively.

Benefits of coordinating system- and application-level adaptation

In order to illustrate the benefits of coordinated approach of Adapt&Cap, we show the performance trace for two approaches under the same performance constraints in Figure 5. In both cases, we run $\times 264$ under a constant performance target of 15 FPS. For the uncoordinated case, we independently activate application-level and system-level control, whereas Adapt&Cap simply uses the best application configuration together with system-level control. As Figure 5 shows, uncoordinated approach creates oscillatory behavior, as system- and application-level adaptation continuously adjusts their decisions for satisfying the same goal, whereas Adapt&Cap control stabilizes after a few iterations. Overall, for 161 experiments with various power and performance constraints, Adapt&Cap reaches to a stable control point after the third iteration 91% of the time.

Coordinating system- and application-level adaptation also achieves better power tracking accuracy due to reduced oscillation. Uncoordinated ap-

proach increases the power tracking error by 3.7 W when compared to the coordinated approach.

Reducing power under performance constraints

We next test Adapt&Cap under dynamically changing performance constraints. We compare the benefits of Adapt&Cap with the adaptive versions of the applications that can track the performance requirements with its internal control through parameter adjustments (i.e., AdaptiveOnly). We only evaluate the parallel portions (regions of interest) of the PARSEC benchmarks (i.e., $\times 264$, bodytrack, swaptions, streamcluster) and the whole execution of Jacobi. The range between maximum and minimum performance varies among applications; therefore, we randomly change the performance requirements within the predetermined maximum and minimum ranges for each application. We dynamically change the performance requirement of the applications every 8 s. We use the same performance traces for both techniques.

In Figure 6, we report the average system-level power consumption of two real servers. Both approaches (i.e., AdaptiveOnly and Adapt&Cap) meet the performance requirements within $\pm 2\%$. However, in both systems, Adapt&Cap significantly reduces the power consumption by utilizing system-level control knobs. Although adaptive capabilities of the applications are useful to meet

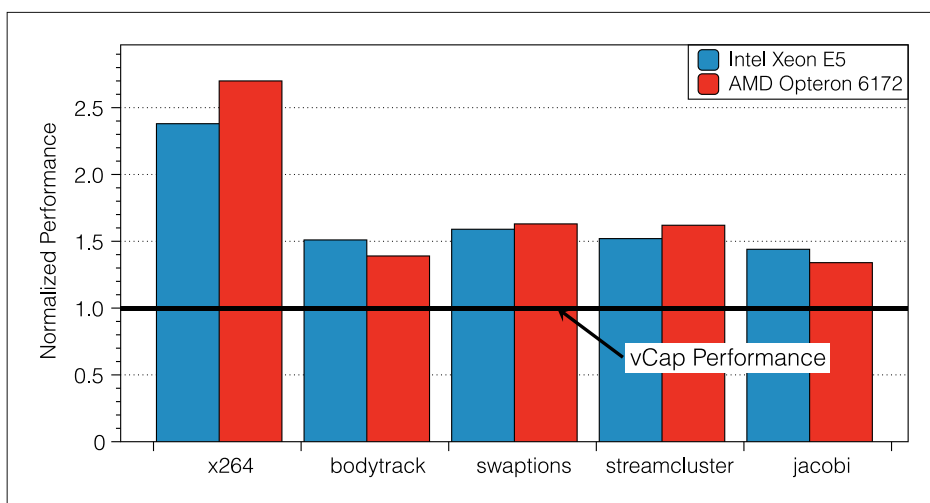


Figure 6. Performance results for Adapt&Cap under dynamically changing power constraints. Adapt&Cap improves the performance up to 2.7× compared to vCap under the same power constraints.

the performance requirements, AdaptiveOnly consumes more or less the same amount of power regardless of the performance targets. The underlying reason is that the modifications to the application parameters have negligible impact on the power consumption. Therefore, Adapt&Cap can achieve the same performance at a much lower power cost by utilizing additional system-level management knobs. Furthermore, Adapt&Cap provides more efficient execution for multithreaded applications regardless of the application characteristics. Our application set covers both highly memory-bound applications (i.e., streamcluster), as well as CPU-bound applications, such as x264 and swaptions. On average, Adapt&Cap achieves up to 27% power reduction when compared to AdaptiveOnly approach, and consistently provides lower power for all applications.

Improving performance under power constraints

In the second set of experiments, we evaluate Adapt&Cap under dynamically changing power caps and compare the performance of Adapt&Cap with vCap, which is an adaptive yet application agnostic power management technique and runs the default versions of the applications. For each system (i.e., Intel, AMD), we create separate power cap traces, as the power ranges of these two systems vary significantly. In Figure 6, we show the perfor-

mance improvements on Intel and AMD servers, where the default application performance with vCap is normalized to 1. Adapt&Cap improves the performance by up to 2.7x by utilizing adaptive applications when compared to vCap running default applications.

Depending on the default configuration parameters, achievable performance range, and the underlying platform, performance improvements show significant variation. However, Adapt&Cap consistently outperforms the vCap and provides 1.68× performance improvements on average.

WITH THE INCREASING degree of heterogeneity in today's data centers, it has become essential to design adaptive systems as well as adaptive applications that can optimize power and performance under various hardware configurations and/or dynamically changing constraints. System-level adaptations are necessary to manage the power consumption, while application-level adaptations enable autonomous performance optimization at runtime. However, the interplay between power and performance requires designing power management techniques that are aware of the application-level adaptation capabilities.

In this work, we propose Adapt&Cap, which combines application- and system-level adaptation to improve the energy efficiency. We implement Adapt&Cap on two real multicore servers and

show that unifying system- and application-level adaptability improves the performance by 1.68× and reduces the power by 22% on average, when compared to system-only or application-only adaptations ■

Acknowledgment

We would like to thank B. Havey for his contributions during our initial study of adaptive applications.

References

- [1] R. Nathuji and K. Schwan, "VPM tokens: Virtual machine-aware power budgeting in datacenters," in *Proc. Int. Symp. High Performance Distrib. Comput.*, 2008, pp. 119–128.
- [2] L. Borovick, "The benefits of a virtualized approach to advanced-level network services," in *Proc. Int. Data Corporation*, Feb. 2011, white paper.
- [3] H. Hoffmann, J. Eastep, M. D. Santambrogio, J. Miller, and A. Agarwal, "Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments," in *Proc. 7th Int. Conf. Autonom. Comput.*, 2010, pp. 79–88.
- [4] C. Hankendi, S. Reda and A. K. Coskun, "vCap: Adaptive power capping for virtualized servers," in *Proc. IEEE Int. Symp. Low Power Electron. Design*, 2013, pp. 415–420.
- [5] H. Hoffmann et al., "Dynamic knobs for responsive power-aware computing," in *Proc. 16th Int. Conf. Architect. Support Programm. Lang. Oper. Syst.*, 2010, pp. 199–212.
- [6] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. Int. Conf. Parallel Architect. Compilat. Tech.*, 2008, pp. 71–82.
- [7] P. E. Bailey et al., "Adaptive configuration selection for power-constrained heterogeneous systems," in *Proc. 43rd Int. Conf. Parallel Process.*, 2014, pp. 371–380.
- [8] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, "Managing performance vs. accuracy trade-offs with loop perforation," in *Proc. 19th ACM SIGSOFT Symp./13th Eur. Conf. Found. Softw. Eng.*, 2011, pp. 124–134.
- [9] G. Dhiman, G. Marchetti, and T. Rosing, "vGreen: A system for energy-efficient computing in virtualized environments," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2009, pp. 243–248.
- [10] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: Coordinated multi-level power management for the data center," in *Proc. 13th Int. Conf. Architect. Support Programm. Lang. Oper. Syst.*, 2008, pp. 48–59.
- [11] K. Ma and X. Wang, "PGCapping: Exploiting power gating for power capping and core lifetime balancing in CMPs," in *Proc. 21st Int. Conf. Parallel Architect. Compilat. Tech.*, 2012, pp. 13–22.
- [12] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory power estimation and capping," in *Proc. ACM/IEEE Int. Symp. Low-Power Electron. Design*, 2010, pp. 189–194.

Can Hankendi is currently a Postdoctoral Researcher at AMD Research. His research interest is adaptive techniques for resource and power management on multicore servers. Hankendi has a PhD from the Electrical and Computer Engineering Department, Boston University, Boston, MA, USA.

Ayse Kivilcim Coskun is currently an Assistant Professor at the Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA. Kivilcim Coskun has an MS and a PhD in computer science and engineering from the University of California San Diego, La Jolla, CA, USA. She is an Associate Editor of IEEE Embedded Systems Letters. She is a member of the IEEE and the Association for Computing Machinery (ACM).

Henry Hoffmann is currently an Assistant Professor in the Department of Computer Science, University of Chicago, Chicago, IL, USA, where his research focuses on adaptive techniques for meeting computing systems' accuracy, performance, energy, and resilience constraints. Hoffmann has a PhD and an SM in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA (2013 and 2003, respectively).

■ Direct questions and comments about this article to Can Hankendi, Electrical and Computer Engineering Department, Boston University, Boston, MA, USA; hankendi@bu.edu.