This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

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

Can Hankendi[1], Henry Hoffmann[2] and Ayse Kivilcim Coskun[1]

[1]ECE Department, Boston University, Boston, MA,

{hankendi, acoskun}@bu.edu

[2]Department of Computer Science, University of Chicago, Chicago, IL

hankhoffmann@cs.uchicago.edu

Abstract – Modern data centers become increasingly power limited, due to continuously increasing demand on cloud resources. As achieving low power and high performance are competing objectives, the efficient management of power/performance tradeoffs is one of the most challenging tasks in modern computing systems. In order to address the increasing management complexity, various types of adaptive techniques, such as system-level DVFS or application-level modifications have been proposed previously. In this work, we propose and implement an adaptive framework that jointly utilizes system and application-level adaptation to improve efficiency of multi-core servers. Our results show that joint-management reduces the power consumption up to 27% and improves the performance up to 2.7x under power and performance constraints.

1. Introduction

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 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

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 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:

- We first demonstrate how to improve the power/performance trade-off 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 2.7x 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 preset our results collected on real servers in Section IV.

2. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

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 Section 4.2.

In Figure 1, we show the power and performance (i.e., seconds elapsed processing each frame) for x264 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 trade-off curve for the power and performance space. This result in Figure 1 motivates the idea of using adaptive capabilities of applications to push the performance while reducing the power consumption through system-level control.

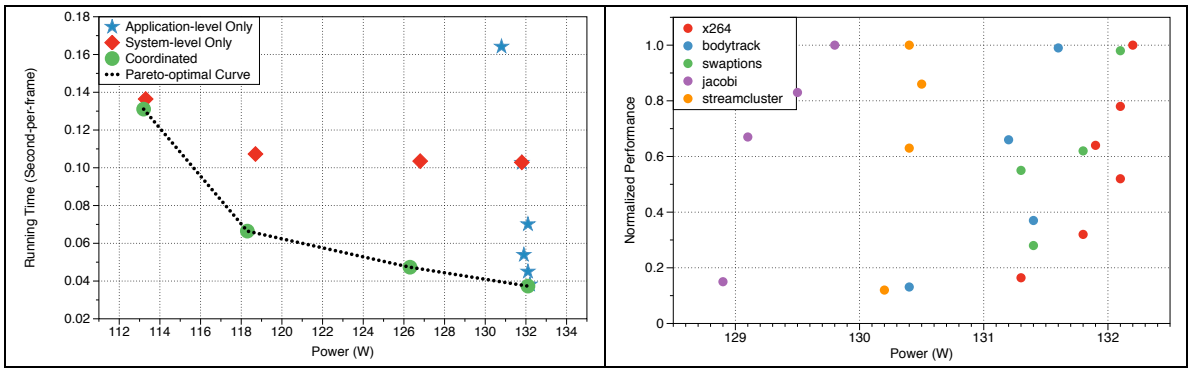

Figure 1: Power and performance tradeoff space for various adaptive techniques on Intel Xeon E5 multicore server when running *x264 (left)*. Proposed coordinated management extends the Pareto points to a more efficient operating point. Application adaptation has minimal impact on power consumption on all 5 of the applications (right).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

3. 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.

3.1. 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.

3.2. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

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.

3.3. 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.

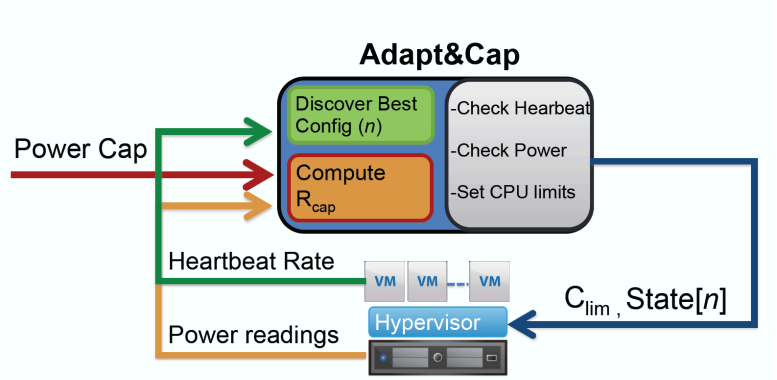

Figure 2: Adapt&Cap reads heartbeat rates and power measurements, and chooses the amount of CPU resources required and the optimum application state.

Figure 3 provides the pseudo-code for the Adapt&Cap framework that consists of three major steps, which are (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 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 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

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

```
Start Configure
Read stateTable;
        Choose State_max
        Read P_max, HB_max                          // Assume highest HB and power
        Compute   P_HB = (P_max -P_idle) / HB_max   // P_HB: Dynamic power per HB
        Compute   CPU_HB = CPU_Used / HB_max        // CPU_HB: CPU usage per HB in MHz
End

Start PowerControl
n_iteration=1;
Read P_cap
Compute HB_cap = (P_cap -P_idle)/ P_HB              // Maximum achievable HB
Set CPU_limit = HB_cap / CPU_HB                     // Set CPU resource limits
Compute n_core = ceil(CPU_limit / f_core)           // Compute minimum number of cores
Pack: n_core                                        // Pack threads

        If (P_cur > (P_cap+ ε))                     // Fine tune the power control
                Set CPU_limit = CPU_limit – (f_core/n_iteration)
                Compute n_core = ceil(CPU_limit / f_core)
                Pack: n_core
                n_iteration++
        ElseIf (P_cur < (P_cap- ε))
                Set CPU_limit = CPU_limit + (f_core/n_iteration)
                Compute n_core = ceil(CPU_limit / f_core)
                Pack: n_core
                n_iteration++
        Else
        n_iteration=1;
End

Start HBControl
Read HB_target
        Set CPU_limit = HB_target / CPU_HB
        Compute n_core = ceil(CPU_limit / f_core)
        Pack: n_core
End
```

Figure 3: Pseudo-code 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 adjusts its decisions.

number of iterations. We use 2W as our **ε** value. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

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.

3.4. Experimental Setup

In this work, we target multi-core based servers that run multi-threaded applications. Our experimental setup includes an AMD 12-core Magny Cours (Opteron 6172) and an 8-core Intel Xeon E5 (2603). Magny Cours consists of two 6-core dies attached together on a single chip. Each 6-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 multi-socket system that includes two 4-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., 4 cores) shares 10 MB of L3 cache, and all 8 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 second 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 multi-threaded applications dominates the application execution time in real computing clusters. To evaluate our technique, we use 4 adaptive-applications from PARSEC 2.1 benchmark suite (i.e., bodytrack, swaptions, streamcluster and x264) 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.

4. Results

In this section, we present the benefits of the *Adapt&Cap* framework on real-life servers. We test our framework under two scenarios that are (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).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

4.1. 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 x264. We randomly change the power cap in every 8 seconds between 115W-135W for the Intel server, and between 85W-165W 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 and, in this way, it accurately tracks the power caps. The absolute value of the average tracking error is 1.8W and 1.2W for the Intel and AMD machines respectively.

4.2. 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 x264 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 approach increases the power tracking error by 3.7W when compared to the coordinated approach.

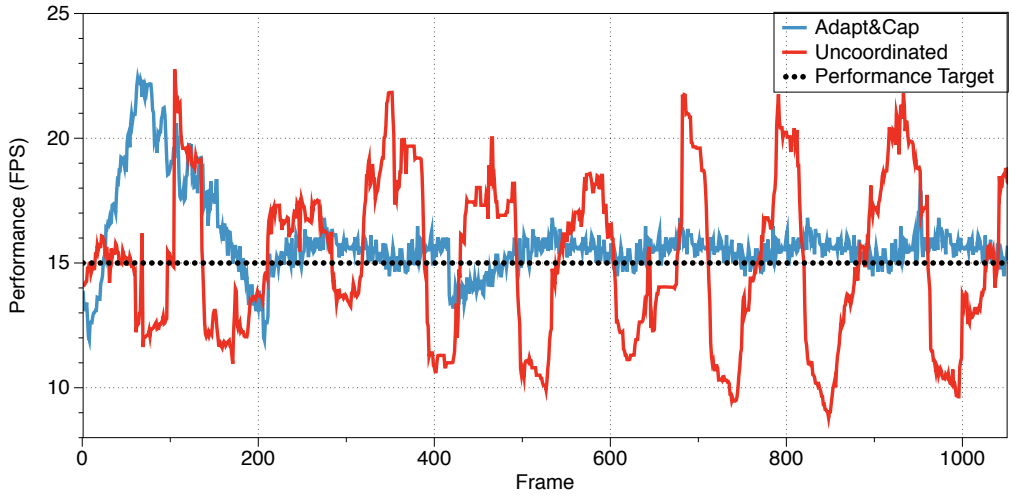

Figure 4: Uncoordinated approach shows oscillatory behavior, as system and application adaptation controls are not aware of other decisions that impact the performance of the system significantly.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

4.3. 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., x264, 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 in every 8 seconds. 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 ±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 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 multi-threaded 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.

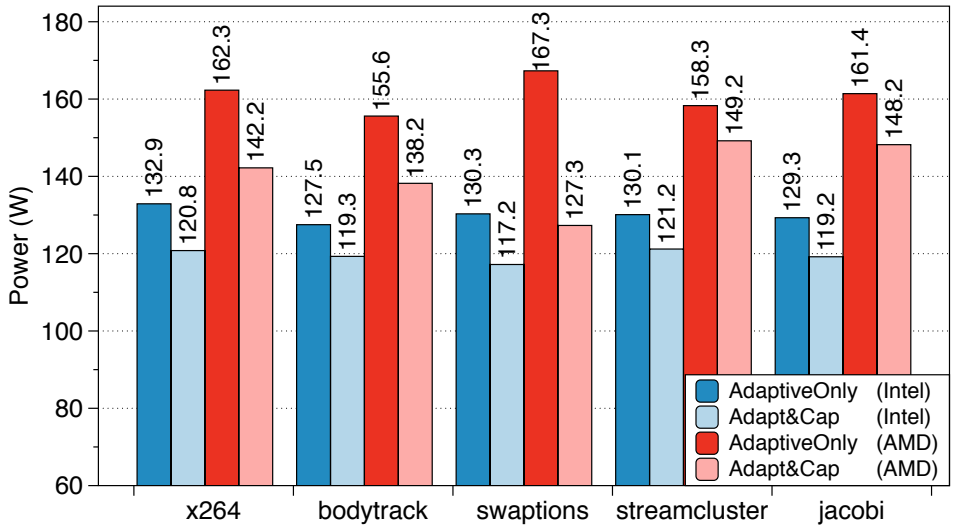

Figure 5: Comparison of power consumption for Adapt&Cap and only adaptive application under dynamically changing performance constraints. Adapt&Cap reduces the power consumption up to 27% compared to only application-level adaptation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

4.4. 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. Utilizing adaptive applications, *Adapt&Cap* improves the performance up to 2.7x when compared to running the default version of the application with vCap. In Figure 7, we show the performance improvements on Intel and AMD servers, where the default application performance is normalized to 1.

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 1.68x performance improvements on average.

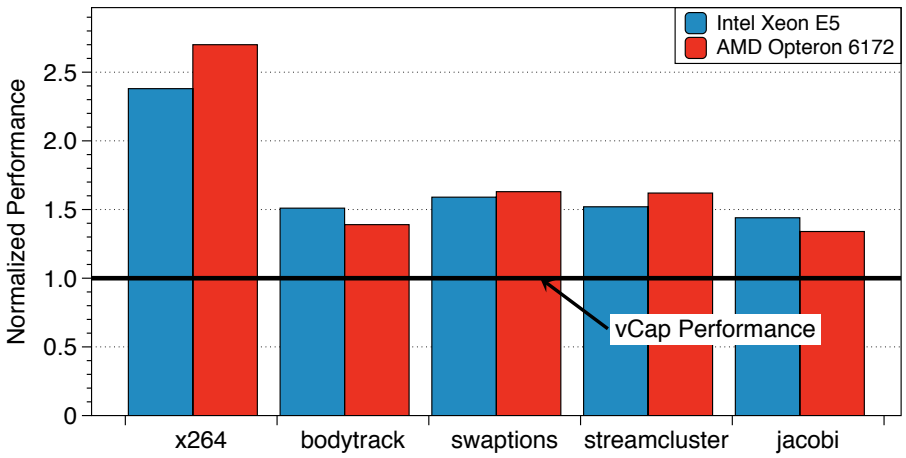

Figure 6: Performance results for Adapt&Cap under dynamically changing power constraints. Adapt&Cap improves the performance up to 2.7x compared to vCap under the same power constraints.

5. Conclusion

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 multi-core servers and show that unifying system and application-level adaptability

improves the performance by 1.68x and reduces the power by 22% on average, when compared to system-only or application-only adaptations.

6. Acknowledgments

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

7. References

[1] R. Nathuji and K. Schwan, “VPM Tokens: Virtual Machine-aware Power Budgeting in Datacenters”. In *International Symposium on High Performance Distributed Computing (HPDC)*, 2008, pp. 119-128.

[2] L. Borovick, “The Benefits of a Virtualized Approach to Advanced-Level Network Services,” International Data Corporation (IDC), Whitepaper, February 2011.

[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 *Proceedings of the 7th International Conference on Autonomic Computing (ICAC)*, 2010, pp. 79-88.

[4] C. Hankendi. S. Reda, A. K. Coskun, "vCap: Adaptive Power Capping for Virtualized Servers". In *Proceedings of IEEE International Symposium on Low Power Electronics and Design (ISLPED),* 2013, pp. 415-420.

[5] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, “Dynamic Knobs for Responsive Power-aware Computing”. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2010, pp. 199-212.

[6] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The PARSEC Benchmark Suite: Characterization and Architectural Implications”. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2008.

[7] Bailey, P. E., Lowenthal, D. K., Ravi, V., Rountree, B., Schulz, M., & de Supinski, B. R., “Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems”. In *Proceedings of the 43rd International Conference on Parallel Processing (ICPP),* 2014.

[8] Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. 2011. Managing Performance vs. Accuracy Trade-offs with Loop Perforation. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering* (ESEC/FSE '11). ACM, New York, NY, USA, 124-134.

[9] G. Dhiman, G. Marchetti, and T. Rosing, “vGreen: A System for Energy-efficient Computing in Virtualized Environments,” in *ISLPED*, 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 *ASPLOS*, 2008, pp. 48–59.

[11] K. Ma and X. Wang, “PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs,” in *PACT*, 2012, pp. 13–22.

[12] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, “RAPL: Memory Power Estimation and Capping,” in *ISLPED*, 2010, pp. 189–194.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2015.2463275, IEEE Design and Test

8. Biographies

Can Hankendi received his M.S. degree in Electrical Engineering from University of Southern California, Los Angeles, in 2010. He is currently a Ph.D candidate in Electrical and Computer Engineering Department at Boston University. His research interest is adaptive techniques for resource and power management on multi-core servers for multi-threaded workloads.

Henry Hoffmann received the PhD and SM degrees in electrical engineering and computer science from MIT in 2013 and 2003, respectively. He is currently an Assistant Professor in the Department of Computer Science at the University of Chicago where his research focuses on adaptive techniques for meeting computing systems' accuracy, performance, energy, and resilience constraints.

Ayse K. Coskun (M06) received the M.S. and Ph.D. degrees in Computer Science and Engineering from the University of California, San Diego. She is currently an Assistant Professor at the Department of Electrical and Computer Engineering, Boston University (BU), Boston, MA. She currently is an associate editor of IEEE Embedded Systems Letters. She is a member of the IEEE and ACM.