2013 IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum

Adaptive Power and Resource Management Techniques for Multi-threaded Workloads

Can Hankendi Ayse K. Coskun

Electrical and Computer Engineering Department
Boston University, Boston, MA 02215
{hankendi, acoskun}@bu.edu

***Abstract*—As today's computing trends are moving towards the cloud, meeting the increasing computational demand while minimizing the energy costs in data centers has become essential. This work introduces two adaptive techniques to reduce the energy consumption of the computing clusters through power and resource management on multi-core processors.**

We first present a novel power capping technique to constrain the power consumption of computing nodes. Our technique combines Dynamic Voltage-Frequency Scaling (DVFS) and thread allocation on multi-core systems. By utilizing machine learning techniques, our power capping method is able to meet the power budgets 82% of the time without requiring any power measurement device and reduces the energy consumption by 51.6% on average in comparison to the state-of-the-art techniques. We then introduce an autonomous resource management technique for consolidated multi-threaded workloads running on multi-core servers. Our technique first classifies applications according to their energy efficiency measure, then proportionally allocates resources for co-scheduled applications to improve the energy efficiency. The proposed technique improves the energy efficiency by 17% in comparison to state-of-the-art co-scheduling policies.

I. 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. Therefore, future computing clusters are required to be energy-efficient in order to be able to meet the continuously increasing computational demand. Moreover, administration and management of the data center resources has become significantly complex, due to increasing number of servers installed on data centers. Therefore, designing autonomous techniques to optimally manage the limited data center resources is essential to achieve sustainability in the cloud era.

The achievable maximum performance of a computing cluster is determined by (1) infrastructural/cost limitations (e.g, power delivery, cooling capacity, electricity cost) and/or (2) available hardware resources (e.g., CPU, disk size). Optimizing the performance under such constraints (i,e., power, resource) is critically important to improve the energy efficiency, therefore to reduce to cost of computing. Moreover, the emergence of multi-threaded applications on cloud resources bring additional challenges for optimizing the performance-energy tradeoffs under resource constraints, due to their complex characteristics such as performance scalability and inter-core communication.

In this work, we present two adaptive management techniques for multi-threaded workloads to improve the energy efficiency of multi-core servers under power and resource constraints. We first present a power capping technique to optimize the performance of multi-threaded applications under power constraints. The proposed technique adaptively manages the voltage-frequency (V-F) settings and the number of active cores depending on the characteristics of the applications. Our technique dynamically packs the active threads onto variable number of cores and jointly utilizes DVFS to optimize performance while meeting the power constraints. As a second step, we target optimizing resource sharing across applications that share the same physical resources to improve the energy efficiency. The optimum resource sharing among consolidated applications depends on the specific characteristics of the applications (e.g., scalability with increasing resources). In order to improve the energy efficiency of server nodes, we first present a technique to classify the applications according to their energy efficiency levels. Then, we present an adaptive resource allocation strategy for consolidating multi-threaded workloads on multi-core servers. At runtime, the proposed technique allocates resources proportional to the energy efficiency levels of the co-scheduled applications to improve the energy efficiency.

The rest of paper is organized as follows. In Section II, we discuss the significance of our approach to power capping and consolidation in comparison to state-of-the-art techniques. In Section III, we present our power capping technique, "Pack & Cap", and present our results based experiments on a real-life multi-core server. We present an autonomous resource allocation framework for virtual environments and demonstrate the success of our approach on a real-life system in Section IV. Section V discusses our future research directions and concludes the paper.

II. Background and Contributions

In this section, we first discuss the previous approaches in power capping and consolidation. We then present our specific contributions and the significance of our power capping and consolidation techniques in comparison to state-of-the-art approaches.

A. Power Capping

In recent years, power capping has become a desirable feature. Therefore, current processor vendors begin to provide peak power management features in commercial products.

978-0-7695-4979-8/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPSW.2013.258

IEEE computer society

AMD has introduced *PowerCap Manager* for 45 nm Opteron processors [1]. Intel recently introduce Running Average Power Limiting (RAPL) methodology with the Sandybridge architecture, which gives ability to cap peak power consumption on Intel processors [2]. For data center power management, HP and Intel jointly offer a power capping technique, which adjusts power caps according to busy/idle states of the nodes [3]. This technique utilizes the DVFS states and the throttling (idle cycle insertion) capabilities at the chip-level. In addition to sleep modes, power nap modes, in which the system can enter and exit from low-power modes in milliseconds, are proposed to cope with the demand variation patterns in data centers [4].

This paper brings the following important innovations over the state-of-the-art: (1) Our technique targets caps on peak server power, while most prior techniques (such as throttling and DVFS) focus on maintaining an average power consumption value; (2) we do not require any modifications to the hardware beyond fundamental DVFS capabilities, which are already included in most processors; (3) we do not require power metering during runtime which saves large investment in power metering infrastructure; (4) we use thread packing in addition to DVFS, which increases the power range and improves performance compared to applying DVFS alone. Our approach is also applicable to single-threaded applications running on multi-core based systems.

B. Consolidation and Resource Management

One line of work on consolidation and resource management targets to optimize multiple nodes through techniques, such as turning off idle cores and load balancing across server nodes [5]. At the single server node level, a deeper understanding of workload characteristics (i.e., other than utilization) is required to reduce resource contention and improve energy efficiency of the server nodes. Bhadauria *et al.* propose co-scheduling algorithms based on bus and cache accesses of consolidated applications [6]. Authors propose co-scheduling policies that manage allocated time and space share of applications to improve energy efficiency. Their proposed approach relies on offline energy-delay measurements to make co-scheduling decisions. Dhiman *et al.* propose a VM migration technique based on application characteristics to improve the energy efficiency [7]. Their proposed technique identifies the workloads that have complementary characteristics and co-schedule them on the same physical resources. However, their proposed technique assumes equal resource allocations for co-scheduled applications. These recent co-scheduling policies target pairing applications that have contrasting characteristics to improve energy efficiency.

In contrast to previous work in consolidation and resource management, we utilize resource management knobs on virtualized systems to dynamically adjust the resources, while previous work considers equal resource allocation regardless of the application characteristics [7]. The proposed technique utilizes runtime monitoring to classify applications according to their energy efficiency characteristics, and do not rely solely on offline characterization [6]. Moreover, our technique is able to provide user-defined performance guarantees for individual co-scheduled applications.

III. Adaptive Power Capping

In order to be able to optimize the performance under power constraints, we propose utilizing a logistic regression model. The logistic regression model takes the performance monitoring data as the input, and outputs the probability of being optimum for each candidate operating point (i.e., v-f setting, active core count). In this section, we present the details of our power capping technique to optimize the performance of multi-threaded applications under power constraints. Then, we present our results based on experiments on a commercial multi-core server.

A. Methodology

Our approach for runtime thread packing and DVFS control requires an offline step to train the logistic regression model. In the offline step, we use an extensive set of performance data collected for the parallel workloads in the PARSEC benchmark suite [8] to train the classifiers. Each classifier takes performance counter and per-core temperature measurements as inputs, and outputs the system operating point with the highest probability of maximizing performance within a given power constraint. A classifier instance is trained for each desired power constraint. During runtime, we recall the model associated with the desired power constraint using a lookup table. Then, the control unit sets the system operating point with the highest probability of being optimal. In this way, our model is able to constrain the system power under a power cap at runtime without using expensive power measurement devices.

The offline characterization step makes use of an L_1-regularized multinomial logistic regression (MLR) classifier [9]. While previous techniques require manual inputs to select the performance metrics that are most relevant to energy, power and delay optimization, we use L_1-regularization to systematically find the relevant inputs and mask irrelevant ones. In the offline characterization step, we use an extensive set of power, temperature and performance counter data collected for each PARSEC benchmark at all feasible system operating points (V-F and thread packing combinations). For each workload's parallel phase (region of interest, ROI), we divide the data into 100 billion μ-op execution intervals. We then train an MLR classifier for each desired power constraint.

The inputs to the MLR classifier include a set of workload metrics, which are functions of the system performance-counter values (e.g., *μ-ops retired, load locks, cache misses, resource stalls*, etc.), per-core temperatures, and the current operating point. Given the inputs during runtime, the logistic regression calculates the probability of each candidate operating point being optimal under power caps. The output with the highest probability is then chosen as the current operating point. At runtime, the system logs performance counter and temperature data, and calculates the probability of each operating point being optimal using the set of weights derived from the MLR classifier corresponding to the current

power constraint p_c. The runtime overhead of the proposed technique is minimal, as the model weights are accessed in the form of a lookup table.

B. Experimental Results

All experiments are performed on a server including an Intel Core i7 940 45nm quad-core processor, running the 2.6.10.8 Linux kernel OS. We control the system operating points (V-F settings and thread-packing combinations) using Linux C library interfaces. To implement data collection and runtime control, we interface our data measurement and control apparatus to a MATLAB module compiled as a C-shared library. This module is configured to read lookup tables generated offline, buffer incoming performance counter and temperature data, and periodically output control decisions to a control unit. The runtime overhead for each runtime activation of the control algorithm is in the range of 10-50ms.

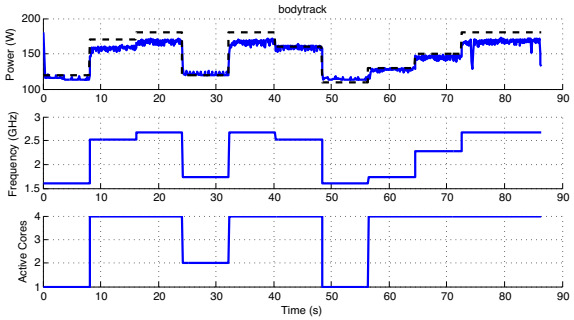

Fig. 1. Demonstration of DVFS and thread-packing control for bodytrack under changing power caps [10].

During the execution of each parallel workload, we periodically change the power constraint to a random value in the 110W - 180W range, and measure the percentage of the execution time for which the power is within a tolerance of the cap value. We do not utilize any power measurements during runtime control. Overall, we are able to constrain the power consumption within the given cap 96% of the time within a 5W margin beyond the power cap. In Figure 1, we demonstrate the adaptive power capping capabilities of our approach. We also compare thread packing with using a fixed smaller number of threads, i.e., thread reduction. While 1-thread fixed or 2-thread fixed corresponds to running 1 or 2 threads, thread packing corresponds to executing applications with 4 threads *packed* on 1 or 2 cores on a quad-core machine. Thread packing is capable of matching the lower bound on the power cap associated with the 1-thread case, but achieves an average of 51.6% reduction in energy. When compared to the 2-thread case, thread packing is able to achieve a better power range, and an average of 15.6% reduction in energy.

IV. Energy-efficient Consolidation

In this section, we present our autonomous resource management technique on consolidated virtual environments. The proposed technique first classifies applications according to their energy efficiency levels through runtime monitoring. Based on the energy efficiency levels of co-scheduled applications, our technique adaptively adjusts the resources allocated for each application at runtime. We first discuss our experimental methodology, then we demonstrate the success of our approach on a real-life server.

A. Methodology

We performed all experiments on an AMD 12-core Magny Cours (6172) server, virtualized by VMware vSphere 5.0 ESXi hypervisor. Magny Cours is a single-chip processor that comprises two 6-core dies (similar to AMD Istanbul architecture) attached side by side. Each core has a 1 MB private L2-cache and each 6-core die has a 6 MB shared L3-cache and a local NUMA node. All cores share a 16 GB off-chip memory. We create two VMs, *VM-0* and *VM-1*, on top of the hypervisor. Our target environment is HPC-type applications, which already utilizes the cores for smaller number of cores. In order to be able to evaluate the impact of increasing resources on the performance of HPC applications, we limit our experiments with 2 VMs. Each VM runs Ubuntu Server 12.04 as its OS and is initialized with either 12 or 6 virtual CPUs (vCPU) depending on the experiment and 8 GB of virtual memory. We use the default `vmkperf` utility to poll the following performance counter data from the physical CPUs at every 2 seconds: CPU cycles, retired instructions, and L3-cache misses. These metrics determine the performance and power characteristics of the applications, as shown in previous work [11].

Each vCPU runs as a process (i.e., world in ESXi) on the hypervisor, thus it is possible to derive VM-level performance measurements through configuring `vmkperf` to poll performance counter readings for vCPUs. We use `esxtop` utility to collect VM-level CPU utilization at every 2 seconds. We measure system power by using a *Wattsup PRO* power meter with a 1 second sampling rate, which is the minimum sampling rate provided for this meter. As total system power determines the electricity cost of a server, we choose to evaluate system power rather than component power (i.e., processor, disk, etc.). We run PARSEC multi-threaded benchmarks in our experiments as a representative set of multi-threaded workloads in HPC clusters. We evaluate the parallel phases of the applications, as they dominate the compute cycles on real-life systems.

B. Autonomous Resource Management

We first study the degree of performance isolation and actual performance of HPC-like applications in virtualized servers. Our observations show that, due to high performance isolation in virtual servers, it is more important to adjust resources allocated to co-scheduled applications rather than choosing the applications pairs.

Following our observations, we propose to adjust resources of co-scheduled applications according to their energy-efficiency level [12]. In order to classify application energy-efficiencies, we utilize Instruction-per-cycle (IPC)*CPU utilization metric. We evaluate throughput-per-watt as the energy efficiency metric, as it evaluates the useful work done (retired instructions) for each watt consumed. By utilizing the IPC*CPU utilization metric, we classify applications using DBSCAN (Density-based spatial clustering of applications

with noise) clustering algorithm. DBSCAN discovers the clusters automatically based on a density reachability threshold, ε [13]. For 13 PARSEC benchmarks, DBSCAN discovers 4 application classes, as *Class-4* are the most power efficient benchmarks, where as the *Class-1* corresponds to the lowest power efficiency class. We then distribute the available resources proportional to the energy efficiency class of the applications. The results of the offline benchmark classification with DBSCAN is implemented as Lookup table (LUT) to be used at runtime. We first sample the IPC and CPU utilization of the applications, then we access the LUT to determine the class of the current sample (i.e., phase) of the application. Therefore, we are able to capture the potential phases that might occur during the execution of an application. In case an outlier phase or application is detected, we rerun the DBSCAN to classify the benchmarks to include the outliers.

In vSphere ESXi environment, it is possible to adjust the limits of the allocated resources (i.e., CPU and memory) for each VM. Resource limits restrict the resource usage of the VMs through forced idleness. We utilize CPU resource limits settings as our main control knob for resource allocation management according to our application classification scheme. We utilize a multi-core desktop as a management node to adjust the resource limits of individual application. Management node monitors VM performance and collect appropriate performance statistics (i.e., retired instructions, clock cycles. CPU utilization) from the ESXi hypervisor. The resource allocation routine is triggered through the vSphere SDK to enforce resource allocation decisions based on the energy efficiency class of the co-scheduled applications.

In order to evaluate impact of our technique on the overall energy efficiency of a cluster in a real-life scenario, we generate 50 random workload sets, each consists of 10 benchmarks (i.e., 5 co-scheduling pairs). In Figure 2, we compare our approach with application selection based co-scheduling policies (i.e., using MPC*Utilization and IPC*Utilization metrics), and also with hybrid techniques (e.g., Proposed technique together MPC*Utilization application matching policy). We also report average throughput-per-watt, throughput, power and energy with respect to the baseline case, where each application is given the maximum amount of resources without any limits. The proposed technique together with MPC*Utilization application selection policy improves the energy efficiency by 21% on average, whereas MPC*Utilization policy alone improves the energy efficiency by only 4% with respect to the baseline.

V. Future Work

As an initial step to improve the energy efficiency of data centers, our research targets improving the efficiency of single server nodes. We plan to test and extend our techniques for larger scale computing clusters with additional resource management capabilities (e.g., VM migration). In addition, there are many diverse set of applications that run on the cloud resources. Currently, our techniques target the HPC domain. However, we plan to extend the application domain to enterprise and big data domains as they are the potential candidate that are expected to dominate the compute cycles on clusters.

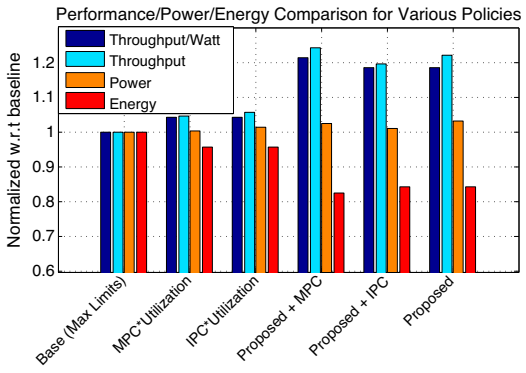

Fig. 2. Average throughput-per-watt, throughput, power and energy comparison normalized w.r.t baseline case.

As a future research direction in power capping, we plan to combine our resource management and power capping techniques to improve energy efficiency of consolidated servers under power constraints. We plan to utilize the virtual control knobs for power management, and then our current resource management framework can easily be extended to operate efficiently under power constraints. Combining two approaches will provide a robust and efficient framework that can be utilized for both resource, power and cost management for large scale computing clusters.

References

[1] T. Samson, "AMD brings power capping to new 45nm Opteron line," http://www.infoworld.com/d/green-it/amd-brings-power-capping-new-45nm-opteron-line-906, 2009.

[2] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory Power Estimation and Capping," in *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, 2010, pp. 189–194.

[3] "HP-Intel Dynamic Power Capping," http://www.hpintelco.net/pdf/solutions/SB_HP_Intel_Dynamic_Power_Capping.pdf, 2009.

[4] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS '09, 2009.

[5] X. Fan, W. dietrich Weber, and L. A. Barroso, "Power Provisioning for a Warehouse-sized Computer," in *In Proceedings of International Symposium on Computer Architecture (ISCA)*, 2007, pp. 13–23.

[6] M. Bhadauria and S. A. McKee, "An Approach to Resource-aware Co-scheduling for CMPs," in *ACM International Conference on Supercomputing*, 2010, pp. 189–199.

[7] G. Dhiman, G. Marchetti, and T. Rosing, "vGreen: A System for Energy-efficient Computing in Virtualized Environments," in *ISLPED*, 2009, pp. 243–248.

[8] C. Bienia, "Benchmarking Modern Multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[9] E. Alpaydin, *Introduction to Machine Learning*, 2004.

[10] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps," in *International Symposium on Microarchitecture (MICRO)*, 2011, pp. 175–185.

[11] M. Khan, C. Hankendi, A. Coskun, and M. Herbordt, "Software Optimization for Performance, Energy, and Thermal Distribution: Initial Case Studies," in *Green Computing Conference and Workshops (IGCC)*, 2011, pp. 1 –6.

[12] C. Hankendi and A. Coskun, "Adaptive Energy-Efficient Resource Sharing for Multi-threaded Workloads in Virtualized Systems (CHANGE-DAC)," in *International DAC Workshop on Computing in Heterogeneous, Autonomous and Goal-oriented Environments*, 2012.

[13] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.