FDIO : A Feedback Driven Controller for Minimizing Energy in I/O-Intensive Applications

Ioannis Manousakis, Manolis Marazakis and Angelos Bilas
Foundation for Research and Technology - Hellas (FORTH)
Institute of Computer Science (ICS)
100 N. Plastira Av., Vassilika Vouton, Heraklion, GR-70013, Greece

Abstract

The relatively low utilization of servers in data-center environments when running I/O-intensive applications is a key concern for efficiency. Energy optimization, by throttling power consumption, is an essential operational goal. Since processors are the most demanding of the components constituting a server, energy optimization has focused on regulating processor consumption. However, more recently memory and storage are increasingly becoming more demanding, collectively accounting for more than 40% of the overall energy consumption in typical system configurations. We argue that this trend necessitates tracking overall energy consumption rather than focusing on any single component. Although currently only processors expose energy-related controls at a fine granularity, we demonstrate that with a more holistic approach we can obtain significant efficiency benefits. Specifically, our feedback-based controller for Linux detects I/O-intensive phases in workloads, and adjusts processor operating frequencies accordingly, in a more effective manner than the standard CPU governors.

1 Introduction

Servers are becoming more energy-efficient, especially at low utilization levels, by taking advantage of new hardware infrastructure and OS policies. An example is the recent Intel Sandy and Ivy Bridge microarchitectures where the processor is able to scale or shut itself down when idle [9]. A further example is the build-in Linux *cpufreq* subsystem which controls the Dynamic Voltage and Frequency Scaling (DVFS) with a set of governors [7]. Currently, DVFS control is based purely on CPU utilization, assuming that the processor is the dominant energy consumer in a server system. However, in server environments the power consumption for memory can reach or even exceed the power consumption of processors [4, 2]. This is especially true if we also account for the power consumption of the server's storage system. As the design of existing governors does not account for this trend, their policies can be potentially harmful for the overall energy efficiency of the system.

On the other hand, even-increasing DRAM density and capacity result into larger portions of storage-heavy applications moving into main memory [6], essentially transforming them from I/O-bound to memory-bound. Although DRAM bandwidth is one order of magnitude higher than a cutting-edge SSD RAID array's, it still remains significantly slower than a server processor. Thus, opportunities for applying DVFS on the processor still exist, at relatively short time-scales that are currently beyond the reach of common control schemes.

Feedback-driven DVFS control is a novel direction in budgeting and capping the power consumption of servers [8]. In this work, we propose a similar approach to exploit optimization opportunities with various system-level energy metrics, different from the CPU utilization levels tracked by the standard Linux governors. Our contributions are as follows: (1) a novel feedback-driven controller for minimizing system-level energy metrics, such as raw energy, Energy Delay Product (EDP) and power consumption, in I/O-intensive workloads, (2) a real-time, hardware instrumentation scheme for measuring the actual power consumption of the processor, DRAM, storage and and the rest of the system peripherals, and (3) an evaluation of the behavior of the standard Linux governors with three well-known I/O workloads (nsort, TPC-H, and TPC-W). Our evaluation highlights opportunities for optimization beyond the reach of the standard Linux governors, in particular when the workload exhibits distinct phases of varying I/O intensity.

The rest of this paper is organized as follows. Section 2 presents our instrumentation on a real system. Section 3 compares the standard Linux governors with the best possible operating settings, and motivates the need for our *FDIO* controller, which is then presented in Section 4. Finally, Section 5 summarizes our conclusions.

2 Testbed and Power Instrumentation

Our test system is a *microserver* based on an Intel i5-2550 4-core processor, with permissible clock rates in the range from 1.6 to 3.3 GHz. The cache hierarchy consists of per-core L1 (32KB) and L2 (256KB) caches, and a shared L3 cache (6MB). The system has 16GB of DRAM, and two Intel X25-E Nand-Flash (SLC) SSDs, connected to a LSI MegaRAID 9265-8i controller. The SSDs are combined in a RAID-0 array, capable of I/O rates up to 500 MB/s for reads and 340 MB/s for writes. Our system runs Ubuntu server 12.04 LTS with Linux kernel 3.2 and the XFS filesystem.

We have designed and implemented an instrumentation scheme, outlined in Figure 1, that is capable of high-rate measurement capture. We take advantage of the physical layout to achieve a breakdown of the total power consumption into four components: CPU, Memory, Storage and Motherboard Peripherals. The 12V line powers the processor and the storage controller, the 5V line powers the memory, and the 3.3V line powers the rest of the motherboard peripherals. A separate 5V line powers the two SSDs. To infer the Storage dimension, we calculate the power demands of the LSI controller from its datasheet and sum it with the power demands of the SSDs. We exclude all the system hard disks from our measurements. Four high-precision *Hall effect* current sensors constantly monitor the three ATX power-supply power lines (+12.0, +5.0, +3.3 Volts). Analog sensor values are converted into digital values, and transmitted over a USB interface to a dedicated data logger board. The data logger includes a high-speed analog-to-digital converter (ADC) operating at a frequency of 40KHz. Code running on the data logger continuously reads the ADC outputs, applies a running-average filer, and interrupts the processor with a rate of 1KHz to report the values. On the test server, a daemon periodically collects the measurements from the data logger, and makes them available for monitoring and control. Figure 6 in Section 3 illustrates the detailed information that we extract from our instrumentation.

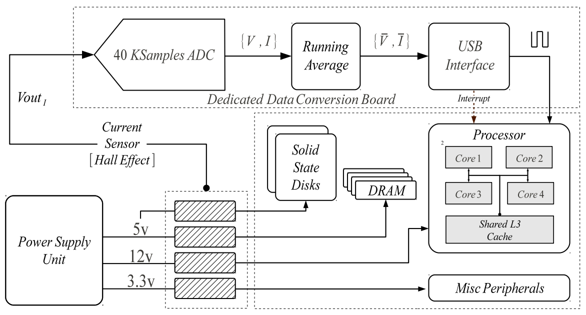


Figure 1: Per-component Power Instrumentation.

3 Application Characterization

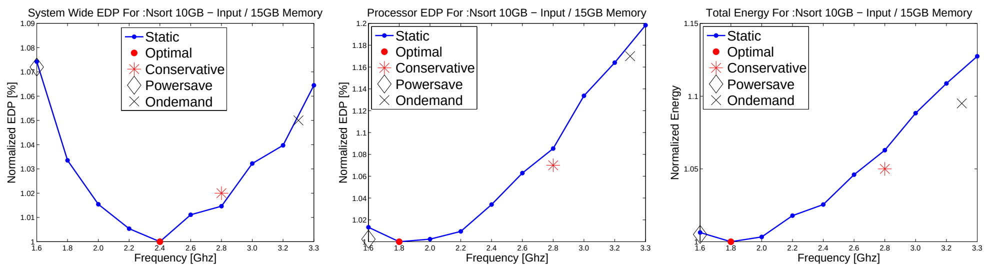
We use our instrumentation to quantify the impact of different DVFS settings on various energy-related metrics, for the following I/O-intensive workloads: (1) `TPC-W` [3] with 36000 emulated clients, (2) `nsort` [5], and (3) `TPC-H` [1] (queries `Q1` and `Q3`). We explore all the possible DVFS operating points to minimize the following metrics: system-wide EDP, Processor-only EDP, and raw system energy. We calculate EDP as the product of the workload execution time and its equivalent energy consumption. We compare our results against the the standard Linux governors: Powersave, Performance, Conservative, and Ondemand [7].

Our results (Figures 2, 3, 4, 5) show that by tuning the CPU DVFS setting we can conserve a significant amount of energy as compared with the default `Ondemand` governor. The DVFS settings range from 1.6 to 3.3GHz, with a step of 0.2GHz. On each figure, we mark the best operating point with a red dot. For each data point in the figures of this section, we run the corresponding workload 10 times and report the average. We flush the buffer-cache before each run.

Beyond reducing the raw energy, we can reduce EDP by 50% (i.e. double the system energy efficiency). Moreover, we observe a weakness of the two adaptive Linux governors (Conservative, Ondemand) to optimize for those metrics, as their operation relies only on CPU utilization. Finally, we present a breakdown of system energy consumption along four components, arguing that the storage is fast becoming a significant challenge in reducing the energy consumption of data-centers.

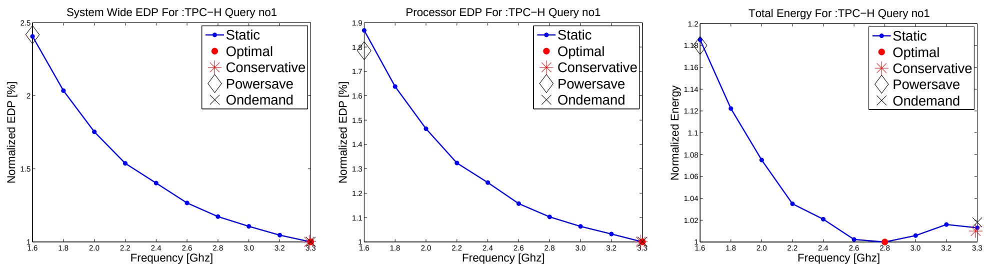
Figure 2 presents our results with `nsort` (10GB input file, that fits in system memory). This workload consists of two phases, one I/O bound and one CPU bound. The best DVFS point for the total energy is at 1.8GHz, which close to the minimum permissible frequency. At this operating point, the system consumes 10% less energy to complete the task as compared to the Ondemand governor and 6% less as compared with the Conservative governor. Powersave also achieves equivalent low energy, by setting the DVFS state at 1.6 GHz. However, for the best system EDP, the optimal DVFS setting is different: 2.4GHz. Compared to the optimal configuration, Conservative matches the EDP, while Powersave and Ondemand result into 7% and 5% higher EDP. For Processor-only EDP, only Powersave achieves comparable efficiency with the optimal DVFS point.

Figures 3 and 4 illustrate results with the TPC-H workload, for queries `Q1` and `Q3` respectively. Q1 exhibits one uniform phase with 100% CPU utilization. Thus, the best operating point for all our metrics lies close to the maximum DVFS setting (3.3Ghz). Powersave achieves the worst energy efficiency, as it results in 18% more En-



(a) system-wide EDP (b) processor-only EDP (c) energy

Figure 2: Analysis of `nsort` execution: system-wide EDP, processor-only EDP, system energy.



(a) system-wide EDP (b) processor-only EDP (c) energy

Figure 3: Analysis of `TPC-H/Q1` execution: system-wide EDP, processor-only EDP, system energy.

ergy, 235% higher EDP and 185% higher Processor-only EDP. On the other hand, both the Ondemand and Conservative governors achieve near-optimal behavior for all the above metrics.

The `TPC-H/Q3` workload differs from `TPC-H/Q1`, as it exhibits distinct phases with varying CPU utilization. The best operating point in terms of energy and processor-only EDP is 2.8GHz, while the 3.2GHz setting leads to the best system EDP. The Conservative governor is again the closest to the optimal state, with 5% higher values in all metrics. On the contrary, the Ondemand governor leads to 13% higher energy, 135% higher Processor-only EDP and 146% higher system EDP, as it is incapable of tracking the frequent and relatively short-lived spikes in the CPU utilization. Finally, the Powersave governor is again the worst, resulting in 16%, 152% and 180% higher than the optimal scores for Energy, EDP and Processor-only EDP.

Figure 5 present results with the `TPC-W` workload. This workload reads 4GB of data from the two SSDs and performs SQL operations on them. It does not exhibit distinct execution phases, with CPU utilization close to 100% at all times. The optimal operating point for `TPC-W` is the 2.8GHz, for all our metrics. Both the Ondemand and Conservative governors result into 12% higher energy, 8% higher EDP and 10% higher Processor EDP. Similar to the `TPC-H` workloads, the Powersave governor achieves the highest values for all our metrics.

Figure 6 presents the breakdown of energy consumption for our workloads, as four components: CPU, Memory, Storage, and Rest of System Peripherals. For each workload, we present the breakdown for the minimum and the maximum permissible operating frequency of the processor. The storage system consists of the two SSDs and the storage controller. We believe that in data-center environment, hosting many micro-servers, both the CPU and Memory components will be optimized over time in terms of energy proportionality; this trend will make the Storage contribution to overall system energy consumption more pronounced over time. At the minimum CPU frequency setting, the Storage contribution ranges from 15% to 20% of the total system energy, roughly equal to half the CPU energy consumption. On the other hand, at the maximum CPU frequency setting, the Storage contribution is in the range from 10% to 17% (for the `TPC-H/Q1` and `nsort` workloads, correspondingly).

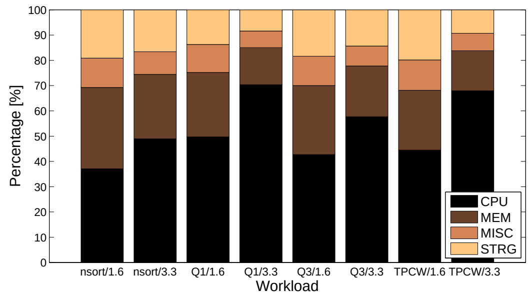
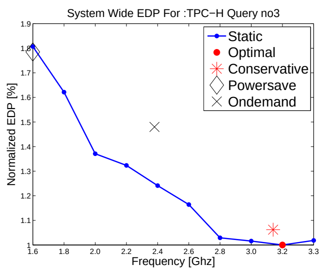
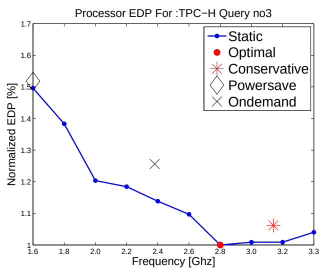


Figure 6: Energy breakdown for various workloads.


(a) system-wide EDP


(b) processor-only EDP

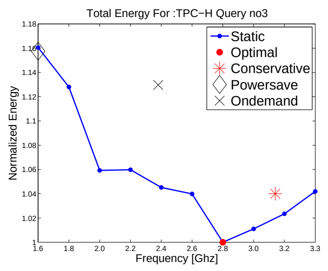
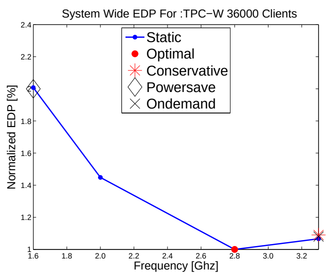
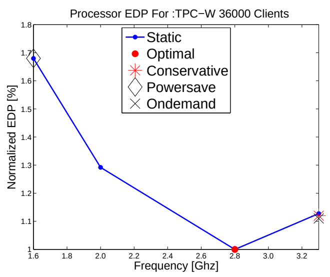

(c) energy

Figure 4: Analysis of `TPC-H/Q3` execution: system-wide EDP, processor-only EDP, system energy.


(a) system-wide EDP


(b) processor-only EDP

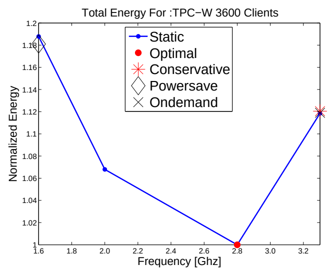

(c) energy

Figure 5: Analysis of `TPC-W` (36000 clients) execution: system-wide EDP, processor-only EDP, system energy.

4 FDIO Controller

In Section 3 we illustrate deficiencies of the standard Linux governors when running I/O-intensive applications. We improve upon these governors by introducing *FDIO*, a feedback-driven controller that strives to minimize system-level energy-related metrics, such as EDP, rather than relying solely on CPU utilization levels. The main control knob remains the CPU DVFS levels; however, *FDIO* is in principle ready to make use of additional control knobs as they become available on servers. Figure 7 illustrates its principle of operation.

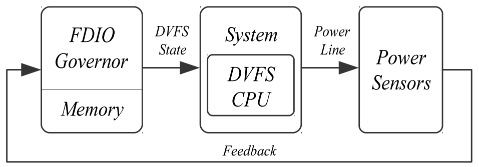


Figure 7: The FDIO Controller.

FDIO captures measurements from our instrumentation, and over a short time interval (on the order of a few 10s of milliseconds) performs an exhaustive check of all possible DVFS settings to determine which settings results in the best energy score. *FDIO* relies on the instrumentation's capability to capture short-duration transients in the energy consumption for each of the sys-

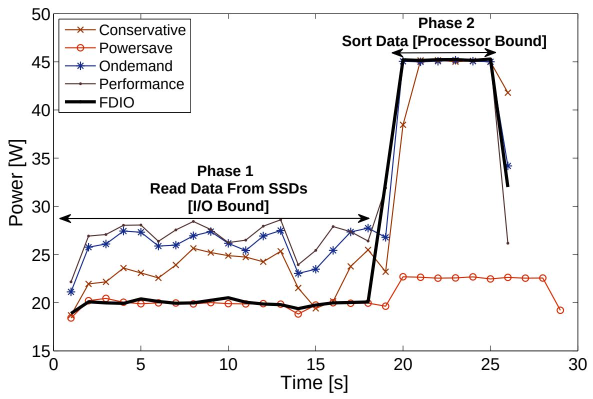


Figure 8: Power timeline for the `nsort` workload.

tem components. Therefore, *FDIO* is capable of tracking phases with different intensity of I/O activity in the course of executing a workload. Such I/O-intensive phases, even if they are short-lived, present opportunities for energy savings without a negative impact on overall system performance. By actually setting the control knobs for brief periods to all permissible values, we actually measure the impact of alternative control knob configurations on both performance and energy consumption, and obtain a ranking of the possible operating states. In a sense, *FDIO* operates as a *meta-scheduler*, emulating the behavior of the standard Linux governors and selecting the most appropriate one for each period. By observing system-level rather than component-level metrics, and by responding fast to workload changes, *FDIO* offers us more opportunities for optimization even in very dynamic run-time environments.

FDIO relies on continuously tracking the run-time power consumption of the overall system. Preferably, this is achieved via appropriate instrumentation; however, in principle *FDIO* could operate based on a model of overall power consumption, provided that this model is validated against representative workloads and its parameters can be adjusted reasonably fast in response to changing run-time conditions.

To demonstrate the effectiveness of *FDIO*, we present a case study around an execution of the `nsort` benchmark, comparing with the standard Linux governors. This benchmark consists of two phases: (1) I/O-intensive data retrieval from the storage devices, and (2) CPU-intensive sorting in the system memory. Using the best operating frequency for each phase as determined by *FDIO*, we compare with the standard Linux governors. Figure 8 illustrates CPU power consumption over time. During the first phase, the most effective DVFS setting is the minimum permissible operating frequency for the CPU cores. During this phase, CPU utilization is around 20%. *FDIO* matches the behavior of the (static) *Powersave* governor. The *Ondemand* and *Conservative* governors are sensitive to the CPU utilization fluctuations; correspondingly, their decisions result in higher energy consumption without any performance benefit, i.e. worse EDP, during this phase of `nsort`. During the second (CPU-intensive) phase, the best DVFS setting is the maximum permissible operating frequency for the CPU cores. With the exception of *Powersave*, all governors select this setting. The *Powersave* governor results in the same energy consumption as the other governors, but its EDP score is twice as high as compared to *FDIO*. Overall, in this case study *FDIO* predicts the best DVFS operating point for both workload phases, and achieves the best energy consumption and EDP scores among all governors, by being able to quickly detect transitions in the overall energy consumption, thus taking advantage of more opportunities for fine-tuning the DVFS operating points.

5 Conclusions

In this paper we explore energy-saving opportunities in servers running I/O-intensive workloads. We present a hardware instrumentation scheme capable of measuring real power consumption of a running system at a fine time granularity, and obtaining four components: Processors, Memory, Storage, (rest of) Peripherals. We find that the non-processor components can account for up to 60% of the overall energy consumption. We use our instrumentation to characterize a set of I/O-intensive workloads (TPC-W, TPC-H, nsort), for different DVFS settings of the processors. We find that the standard Linux governors that only track processor utilization are ineffective for these workloads. The standard governors fail to minimize important energy-related metrics such as EDP, leading to inefficiencies at the overall system level. We present *FDIO*, a feedback-driven controller that uses run-time power consumption and performance metrics to perform optimal DFVS selection. Our controller is capable of identifying distinct phases during the course of workload execution, and performs exhaustive space exploration to determine the optimal setting for the DVFS control knob. *FDIO* is also capable to take advantage of other power-related control knobs, as we expect to find in upcoming generations of server systems.

Acknowledgments

We thankfully acknowledge the support of the European Commission under the 7th Framework Programs through the IOLANES (FP7-ICT-248615) and CumuloNimbo (FP7-257993) projects.

References

[1] Council, T. P. P. Tpc-h benchmark specification. *Published at http://www. tcp. org/hspec. html* (2008).

[2] Deng, Q., Meisner, D., Ramos, L., Wenisch, T. F., and Bianchini, R. Memscale: active low-power modes for main memory. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems* (New York, NY, USA, 2011), ASPLOS XVI, ACM, pp. 225–238.

[3] García, D. F., and García, J. Tpc-w e-commerce benchmark evaluation. *Computer 36*, 2 (2003), 42–48.

[4] Manousakis, I., and Nikolopoulos, D. S. Btl: A framework for measuring and modeling energy in memory hierarchies. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium on* (2012), IEEE, pp. 139–146.

[5] Nyberg, C., Koester, C., and Gray, J. Nsort: a parallel sorting program for numa and smp machines, 1997.

[6] Ousterhout, J., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G., Rosenblum, M., Rumble, S. M., Stratmann, E., and Stutsman, R. The case for ramclouds: scalable high-performance storage entirely in dram. *SIGOPS Oper. Syst. Rev. 43*, 4 (Jan. 2010), 92–105.

[7] Pallipadi, V., and Starikovskiy, A. The ondemand governor. In *Proceedings of the Linux Symposium* (2006), vol. 2, sn, pp. 215–230.

[8] Reda, S., Cochran, R., and Coskun, A. Adaptive power capping for servers with multithreaded workloads. *Micro, IEEE 32*, 5 (Sept.-Oct.), 64–75.

[9] Rotem, E., Naveh, A., Rajwan, D., Ananthakrishnan, A., and Weissmann, E. Power-management architecture of the intel microarchitecture code-named sandy bridge. *Micro, IEEE 32*, 2 (2012), 20–27.