Power-Aware Throughput Control for Database Management Systems

Zichen Xu
The Ohio State University
xuz@ece.osu.edu

Xiaorui Wang
The Ohio State University
xwang@ece.osu.edu

Yi-Cheng Tu
University of South Florida
ytu@cse.usf.edu

Abstract

Performance has been traditionally regarded as the most important design goal for database management systems (DBMSs). However, in recent years, the increasing energy cost gradually rivals the benefit of chasing after performance. Therefore, there are strong financial incentives to minimize power consumption of a database system while maintaining its desired performance, so that the energy cost can be best amortized. Such a goal is challenging in practice because the power consumption of a database system varies significantly with the environment and workloads. Many modern hardware provide multiple modes with different power/performance tradeoffs. However, existing research has not used these power modes sufficiently to achieve the best tradeoff for database services due to the lack of the knowledge on database behavior under different power modes. In this paper, we present Power-Aware Throughput control (PAT), an online feedback control framework for energy conservation at the DBMS level. In contrast to heuristic-based tuning techniques commonly used in database systems, the design of PAT is based on rigorous control-theoretic analysis for guaranteed control accuracy and system stability. We implement PAT as an integrated component of the PostgreSQL system and evaluate it with workloads generated from various database benchmarks. The results show that PAT achieves up to 51.3% additional energy savings despite runtime workload dynamics and model errors, as compared to other competing methods.

1 Introduction

The rapid growth of energy-related research in databases is driven by the fact that *data centers are energy starving*. The increasing operating expenses of data centers (e.g., the electricity bill) quickly deplete the revenue earned from database services due to its accumulating demand of energy [18]. The power-performance tradeoff has now become a new key challenge in general purpose database system design [30].

Redesigning DBMS towards high energy efficiency has been discussed in the database community. Poess et al. [19] examine the power saving opportunities from different hardware systems. Lang et al. [11] report large energy savings by using the dynamic voltage and frequency scaling (DVFS) technique in CPUs. However, it is not a trivial task to harvest those opportunities in data processing while maintaining the desired performance . The DBMS performance could be very sensitive to the changes in hardware power modes. For example, tuning one step (25%) down in CPU frequency could result in about 30% performance degradation for CPU intensive queries; in addition, switching low-power modes in memory is a bad idea due to significant performance degradation for any DBMS queries, as shown in Fig.1, Section 2. Therefore, we cannot directly apply existing hardware power management techniques in DBMSs for the energy conservation.

It is also difficult to provide performance guarantees in a DBMS due to workload variations and environment dynamics. We need an adaptive architecture that could promptly monitor query statistics from DBMS and determine whether/to what extend adaption should be performed. Attempting to solve the problem, some studies employ simple hill-climbing strategies to make such important adaption decision [11, 10]. These *ad hoc* control solutions cannot provide desired control performance, such as zero steady-state error and short settling time bound [4]. Although there are many control work done at the OS level, such as [27, 26, 17], they are not feasible due to the lack of critical database information that is needed for making adaptation decisions.

To address the aforementioned problems, we first need to understand the nature of the DBMS's response to the changes of different hardware power modes ("knobs"). Specifically, we need a quantitative system model in the adaptive framework that describes how the DBMS performance changes in response to knobs tuning. Second, the adaptive framework needs to be implemented in light weight without affecting the normal DBMS operation. Finally, the control algorithm shall be robust such that it could tolerate errors from estimation in the DBMS optimizer and workload variations.

In this paper, we present Power-Aware Throughput control (PAT), an online feedback control framework for energy conservation at the DBMS level, to address the above challenges. Our solution takes advantage of well-established techniques from the field of control theory, to deal with systems that are subject to unpredictable dynamics [4]. In this solution, we formulate energy conservation with performance control at the DBMS level into a feedback control problem and tackle it with a proportional-integral (PI) controller based on the DBMS system model. Specifically, this paper makes the following contributions:

- We explore the relationship among query statistics, the DBMS throughput, hardware power states, and the active power consumption[1] via empirical studies. Our results show that 1) there exists great energy savings when tuning DVFS for processing I/O intensive queries; 2) The relationship between the DBMS throughput and the CPU frequency is an approximated linear model when DBMS workloads are steady; 3) the ratio of I/O intensive queries in the workload plays a major role in the workload statistics that affect the performance of the control.

- As one of the first attempts to introduce classic control theory into the energy management in DBMSs, we design PAT to control the DBMS throughput while minimizing the active power consumption.

- We design and implement a query classifier based on the fuzzy set theory. The classifier provides important information, such as the ratio of I/O queries, which plays a key role in achieving effective throughput control. The fuzzy-logic-based design also provides new insights to the classic problem of query clustering.

- We implement PAT within the real DBMS – PostgreSQL and evaluate it with various baselines. The results show that, PAT has significantly more energy saving (51.3%) with the least control errors comparing with other control baselines.

The rest of the paper is organized as follows: we first discuss our study on characterization of database system in Section 2. Section 3 introduces the overall control framework; Sections 4 and 5 present the design and analysis of the workload classifier and controller in PAT, respectively. Section 6 talks about our empirical evaluation of the proposed control strategy. Section 7 compares our work with related work; Section 8 concludes the paper.

[1] we use the active power of the whole system for the measurement throughout this paper. Any power data, if without specification, is the active power of the system.

2 System Characterization Study

In this section, we report our findings based on empirical studies of database behavior as a foundation of control framework design.[2]

In our study, we focus on the DBMS throughput (query per second, QPS) as the main performance metric. The throughput, as the reciprocal of the average response time, is an important performance metric. For example, transaction processing performance council (TPC) uses throughput to define and rank the performance of different DBMS products [22]. To keep the DBMS throughput within a desired level is essential to avoid situations, such as overloading. We take controlling the response time of individual queries as a future work for the design of PAT, which will not be discussed in this paper.

The impact of hardware power modes with different DBMS workloads: to further understand the impact of low-power modes in different hardware components on the power consumption and the performance of database services, we use five power states of the memory (described in [3]), four discrete DVFS levels of the CPU (described in [27]), and the CPU C-state (described in [15] and labeled as "DVFS0"). To avoid possible bias from measurement errors, we repeat experiments using CPU intensive and I/O intensive workloads in several trials and collect the average result, demonstrated in Fig.1.

Fig.1(a) and Fig.1(b) show the DBMS performance and the power measurement of different power states in memory under two types of DBMS workloads. As we can see, a state transition in memory, such as from the active state to the active-standby state, can contribute to at most a 10% saving in active power. However, the power saving comes with a severe performance penalty as a 95% performance degradation in CPU workloads and a 98% degradation in I/O workloads after the transition. The penalty comes from unacceptable low I/O bandwidths from memory low power modes, which make any processing queries enter infinite cycles of I/O wait. Thus, although [3] claims energy savings from tuning power states in the memory, it may not be a feasible solution for database services. As a result, we find that any hardware power management techniques which increase per-page I/O cost may have a severe consequence on the DBMS throughput, which eventually leads to unacceptable high energy cost.

[2] details of the experiment setup can be found in Section 6.1.

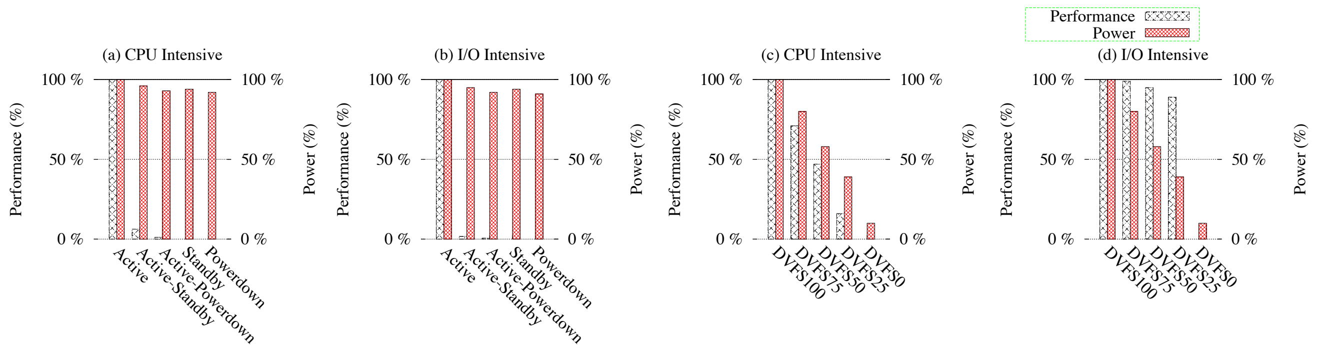

Figure 1: Performance (throughput) of a 100GB database system under low power states of the memory (Fig.a and Fig.b) and the CPU (Fig.c and Fig.d). All data are normalized to the normal scenario with active memory and CPU at 100% frequency. Subfigures are labeled by the workload type used in the test. DVFS0 is the CPU C-state, in which the system is in halt and there is no observed DBMS throughput.

Fig.1(c) and Fig.1(d) illustrate the results of many DBMS workloads running in different CPU power states. One observation is that, in both I/O intensive and CPU intensive queries, the active power cost monotonically decreases with the CPU frequency. This is in conformity with results reported in [14, 23]. The DBMS throughput, on the other hand, shows the same behavior. Such observations imply that CPU frequency and system performance are positively related and this gives us confidence in building an approximated linear system model between performance and power consumption. Nevertheless, comparing Fig.1(c) and Fig.1(d), the DBMS sensitivity[3] is different in CPU intensive and I/O intensive workloads. Apparently, one could harvest more power savings from I/O intensive queries without affecting their performance much.

Fig.1(c) and Fig.1(d) also demonstrate system reaction to the CPU C-state (DVFS0) in terms of power and performance. When the CPU is set to the C-state, the whole system is in the halt state. We did not observe any DBMS throughput although the active power consumption is low. At the same time, the delay of transiting in/out of the CPU C-state is so large that it jeopardizes the normal query execution in the DBMS, and leads to uncorrect query results. Thus, we do not implement the CPU C-state in PAT for power saving purposes but evaluate it in a simulation in our tech report [29].

The above experimental results show that CPU DVFS technique is a good candidate for the control actuator. Next we further explore the insight from results of Fig.1(c) and Fig.1(d).

CPU power states, the DBMS throughput and workload statistics: Fig.2(a), again, demonstrates the fact that the active power consumption is linearly related to the relative DVFS level. The power and the performance

[3] The sensitivity is defined as the change of performance in response to CPU frequency changes, as $\left(\frac{\text{throughput}}{\text{CPU frequency}}\right)$.

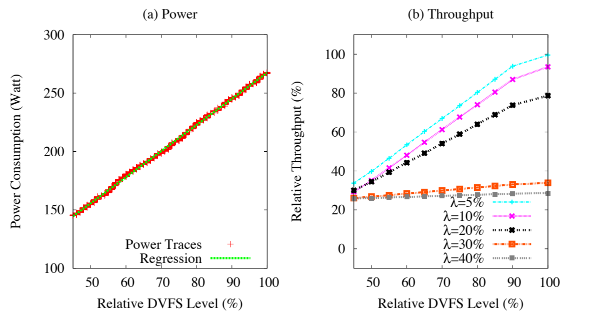

Figure 2: The impact of the CPU frequency (i.e., DVFS levels) on the power consumption (a) and the DBMS throughput (b). The five workloads in (b) differ by their λ – ratios of I/O intensive queries to the overall workload size. All data in (b) are normalized to the largest throughput of the workload with $\lambda = 5\%$ at the maximum CPU frequency.

data in Fig.2(b) are recorded from experiments running DBMS workloads with different statistics (i.e., the fraction of queries that are I/O intensive λ). An important observation from Fig.2(b) is that, there exists a linear relationship ($R^2 = 0.9633$) between throughput and CPU frequency for all DBMS workloads when λ is fixed. Therefore, we use the following linear model to describe the relationship between database throughput and CPU frequency,

$$r = A\lambda f + B \tag{1}$$

Where r is the DBMS throughput, f is the CPU frequency, and A, B are model coefficients.

Among all the workload characteristics, we found that the ratio of I/O-intensive queries λ is the major factor that affects the sensitivity, as shown in Fig.2(b). Our explanation is that, in our platform, Linux system uses Round-Robin as the process scheduling algorithm. Therefore, the more queries are bounded by I/O, the larger chance that those processes will skip their CPU time slices, thus keeping the CPU idle. As a result, a high

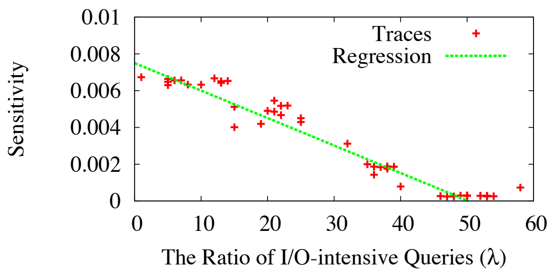

Figure 3: The relationship between workload's frequency-to-throughput ratio and the percentage of I/O-intensive queries in the workload (λ).

λ makes the system less sensitive to the CPU frequency changes. When λ is larger enough, as the grey line in Fig.2(b), the database performance has little change with the CPU frequency, where we could harvest the most energy savings. This sensitivity of the DBMS performance to the CPU frequency changes is measured as the slopes of all the throughput-DVFS lines in Fig.2(b).

Fig.3 shows that the relationship between λ and the sensitivity is also linear (with a goodness-of-fit $R^2 = 95\%$). Since λ is essential in our throughput control, it is necessary to estimate the value of λ to identify the workload at runtime. Note here, when the value of λ increases from 20% to 30% in Fig.2(b), the DBMS throughput drops heavily (50%) at the highest DVFS level. There is a value between 20% to 30%, we called it β, that defines an infection point. When the system crosses this point ($\lambda > \beta$), it will enter an I/O busy waiting state. This state is a "Limbo" that we are trying to avoid in our experiment. The value of β is a relative static number for any given systems. It can be found during the system identification process. In our experimental database system, the value of β is found to be 32%.[4]

3 The Framework of PAT

The control framework PAT is illustrated in Fig.4. The main components of PAT form a feedback control loop (indicated by the red arrow in Fig.4), including the PI controller (Controller), the throughput monitor of the DBMS (Plant),[5] and the CPU power state modulator (Actuator). The goal of the control loop is **to maintain the DBMS throughput at the set point R_s and minimize the power cost**. Specifically, the following steps are invoked in each control period,

1. The throughput monitor measures system throughput $r(i-1)$ in the last period. The *control error* is computed as $\Delta r(i) = R_s - r(i-1)$;

2. The controller receives the control error Δr and the workload statistic factor λ. Based on these values, it computes the *control signal* $f(i)$;

3. The CPU power state modulator receives the control output f, to calculate the new DVFS level and apply it in the CPU.

4. Exception: when $\lambda > \beta$ or the DVFS level is highest but the detected throughput still fails to meet the set point R_s, the CPU modulator will set the DVFS level to the active lowest state to save power for a short period of time t.

[4] The value of β needs to be calibrated when PAT is applied to a different system environment

[5] Note that the monitor itself is not the plant, the DBMS is.

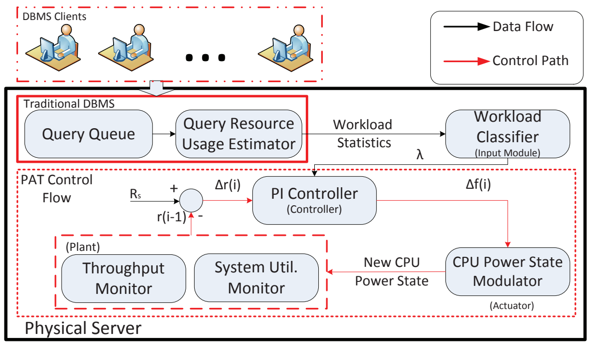

Figure 4: The power-aware throughput control architecture. The names in parentheses are given following control terminology.

Note here, the scale-down duration t is shall be smaller than control period T (introduced in Section 5), and the transaction time of different power state is at least one order of magnitude smaller than t.

3.1 Control Components

Before discussing more details about the two major components – the fuzzy workload classifier (FWC) and the PI controller, we briefly introduce the implementation of other components of PAT first.

CPU Power State Modulator: PAT uses Intel's SpeedStep technique (10ms overhead) to tune the CPU DVFS level. An interesting issue is that the Intel Xeon CPU E5645 used in our platform (as well as many other DVFS-enabled CPUs) only supports several discrete CPU frequency levels. However, PAT needs to set a value of the DVFS level within a normalized continuous range [0 – 100]. Therefore, the task of the modulator is to approximate the desired value using a combination of the supported discrete frequency levels. For example, to get 2.23 GHz CPU frequency during one control period, the modulator would output pseudo frequency signal sequence as {2.67, 2, 2, 2.67, 2, 2} to emulate the average CPU frequency as 2.23 GHz. To realize such idea, we implemented a first-order delta-sigma modulator in the system, which is commonly used in analog-to-digital signal conversion [12].

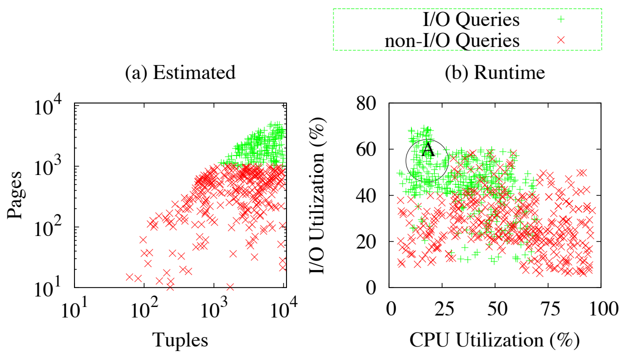

Figure 5: The estimated resource demand (Fig.(a)) and the real resource usage (Fig.(b)) of 500 different queries. Each node in the figure stands for one query.

Throughput Monitor: the throughput monitor is implemented as a daemon program that collects the number of finished queries in each control period. The monitor traces every exit signal from DBMS threads and records the sum in period *i*. The monitor maintains throughput data from the past *n* periods.

System Utilization Monitor: the system utilization monitor is a program that records system status (e.g., CPU utilization) when PAT is active. We use the collected data for the performance analysis.

Query Resource Consumption Estimator: the resource estimator is a tool that retrieves the run-time query estimation information from the query optimizer of the DBMS. Based on such information, FWC could define the query type and update the corresponding parameter λ. Note that the original resource estimator in the DBMS could be highly unreliable. We calibrate this estimator before using it for estimation. The detailed work can be found in the tech report [29].

4 Fuzzy Workload Classifier

As we learned from the observations in Section 2, the key factor in the workload statistics used in the system model is the percentage of I/O intensive queries in the current running workload, namely λ. Thus, to successfully build the system model in order to control a composite workload, the model shall update the value of λ on-the-fly. To solve this problem, we need to classify queries based on its I/O intensity.

4.1 Main Challenge

Fig.5(a) shows the classification results of 500 queries based on a static threshold – 1,000 demanding pages, which is the size as the L3 cache in the server. Each query is labeled as I/O-intensive if the I/O cost is more than 1,000 pages (green node) or non-I/O intensive (red node), otherwise. However, such a rule-based method fails when the resource estimation given by the query optimizer is not accurate enough to reflect the actual requested resource at runtime. Fig.5(b) shows the real resource usage of the same set of queries. The results show only a part of identified I/O queries are real I/O-intensive queries (e.g., those nodes in circle A in Fig.5(b)). The above empirical results show that simple rule based classification fails at obtaining the accurate λ value.

We propose a classification approach based on fuzzy set theory to solve our problem. Fuzzy-based methods are particularly suitable for systems with complex behaviors. They are designed to handle an unpredictable environment with limited number of rules to reach sufficient accuracy [25, 21]. Our FWC collects workload statistics and creates fuzzy rules to identify runtime resource consumption patterns of queries in the workload.

4.2 Fuzzy Classifier Design

In FWC, Sugeno-type fuzzy rules [21] are generated from the clustered data for modeling database workloads. The input for the FWC is the resource demand of the incoming query and the output is the aggregated estimation of runtime resource utilization. For the i^{th} query, its resource demand vector is denoted as $[d^i_{CPU}, d^i_{I/O}]^T$ and the estimated CPU and I/O utilization as $[u^i_{CPU}, u^i_{I/O}]^T$. The number of fuzzy rules shall be the same as the number of clusters in the estimated resource demand map [8]. In our case, there are two clusters: I/O-intensive and non-I/O-intensive. The member functions of the fuzzy rules are linear functions generated via rigorous mathematical tools from Matlab [13]. The rule base is constructed as follows:

$$R_j: \text{ IF } [d^i_{CPU}, d^i_{I/O}]^T \in \text{ cluster } X_j, \text{ THEN } [u^i_{CPU}, u^i_{I/O}]^T = M_j[d^i_{CPU}, d^i_{I/O}]^T + N_j$$

where X_j is the cluster determined by clustering technique, M_j and N_j are parameters from the fuzzy set associated membership functions obtained from the learning process. The symbol $\in$ stands for the distance between the node and the center of cluster X_j. The procedure of workload classification are as follows:

1. *Evaluation*: compute the appropriate fuzzy rule output $[u^i_{CPU}, u^i_{I/O}]^T$ based on the input resource demand vector $[d_{CPU}, d_{I/O}]^T$ using the corresponding membership functions M_j and N_j;

2. *Implication calculation*: obtain implication p_j of each fuzzy set R_j and calculate the confidence t_j that the query belongs to fuzzy set R_j based on the implication weight over all $\frac{\Sigma(p_j)-p_j}{\Sigma(p_j)}$;

3. *Aggregation result*: the output of all fuzzy rules are aggregated and inversely translated into the av-

erage utilization vector $[\sum_j t_j u^i_{CPU}, \sum_j t_j u^i_{I/O}]^T$ from all rules with confidence t_j.

Although there still exist errors from workload classification, those errors are bounded and smaller than the acceptable maximum tolerance (overshoot) in the controller design. Due to the page limit, we put a detailed analysis and examples of FWC in our tech report [29]. The accuracy of FWC is evaluated in Section 6.

5 Throughput Controller Design

5.1 System Modeling

Building an accurate mathematical model of the system to be controlled is of great importance to the entire control loop design. We build the model of the DBMS throughput and the power consumption based on observations in Section 2. Let us denote the length of the control period as T and the throughput within the i^{th} period as $r(i)$. Given $r(i)$, our control goal is to guarantee that the DBMS throughput r could be converged to the set point R_s after a finite number of control periods (*settling time*). Note here, for better establishing the model we scale those two values into percentage. Thus, Δr and f are now the relative control error and the related frequency setting, respectively. For example, $f = 100\%$ means that CPU is running at its highest frequency. In the experiment, the minimum available frequency is 40% of the maximum frequency.

Here we update the system model in Eq. (1) as:

$$\Delta r(i) = \lambda A f(i) + B \tag{2}$$

For the convenience of the control analysis, Eq. (2) is transformed in the z-domain as:

$$R(z) = \lambda A F(z) \tag{3}$$

where $R(z), F(z)$ are the z-transform of signal $\Delta r(i), f(i)$, respectively. Thus, the system transfer function of the DBMS throughput to the frequency change in Fig.2 is:

$$G(z) = \frac{R(z)}{F(z)} = \lambda A \tag{4}$$

We test the system with sinusoidal inputs in Fig.6. Fig.6 demonstrates that our model is sufficiently close to the actual system with $R^2 = 0.9152$.

5.2 Controller Design

The goal of the controller design is to meet the following goals:

- *stability*, the throughput shall settle into a bounded range in response to a bounded reference input;

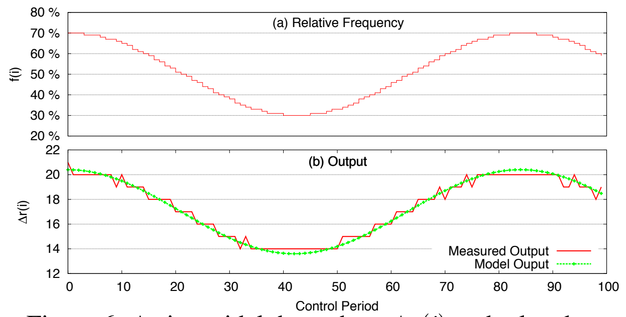

Figure 6: A sinusoidal throughput $\Delta r(i)$ and related control signal $f(i)$.

- *zero steady state error*, when the system enters the steady state, the throughput shall settle to the set point with zero errors; and

- *short settling time*, the system shall settle to the set point before the specified deadline.

Based on the control theory, we design a *Proportional-Integral* (PI) control that has been widely adopted in industry control systems. We select the PI controller for its nice property of the zero-state-error and its fast response [4, 7]. The PI controller can also provide robust control performance despite modeling error and input/output disturbances. It has the following form in the discrete time domain:

$$f(i) = k_P \Delta r(i) + k_I \sum_{1}^{i} (\Delta r(j)) \tag{5}$$

where $\Delta r(i)$ is the control error at i^{th} period. $f(i)$ is the frequency offset. k_I and k_P are control parameters. Those parameters can be analytically chosen to guarantee the system stability and zero steady-state error. From Eq. (5), we have the controller transform function in the z-domain as:

$$C(z) = \frac{z(k_I + k_P) - k_P}{z - 1} \tag{6}$$

Overall, the transfer function $\mathbf{F}(z) = G(z)C(z)$ is,

$$\mathbf{F}(z) = \frac{\lambda A k_p (z-1) + \lambda A k_I z}{(1 + \lambda A(k_I + k_P))z - (\lambda A k_P + 1)} \tag{7}$$

We use the Root-Locus method [7] to design the control coefficients k_I and k_P to guarantee stability and zero steady-state error. The poles of the transfer function are $-0.26 \pm 0.8i$. As both eigenvalues are inside one unit circle, the closed-loop system in our experiments is stable [4]. The values of the system model parameters in Eq. (2) are $A = 4.329$ and $B = 24.329$, based on our characterization study. λ is provided at runtime by FWC. Based on the result of control analysis, control parameters $k_I = 0.5$ and $k_P = 1.06$. More details of the control analysis are in the tech report [29].

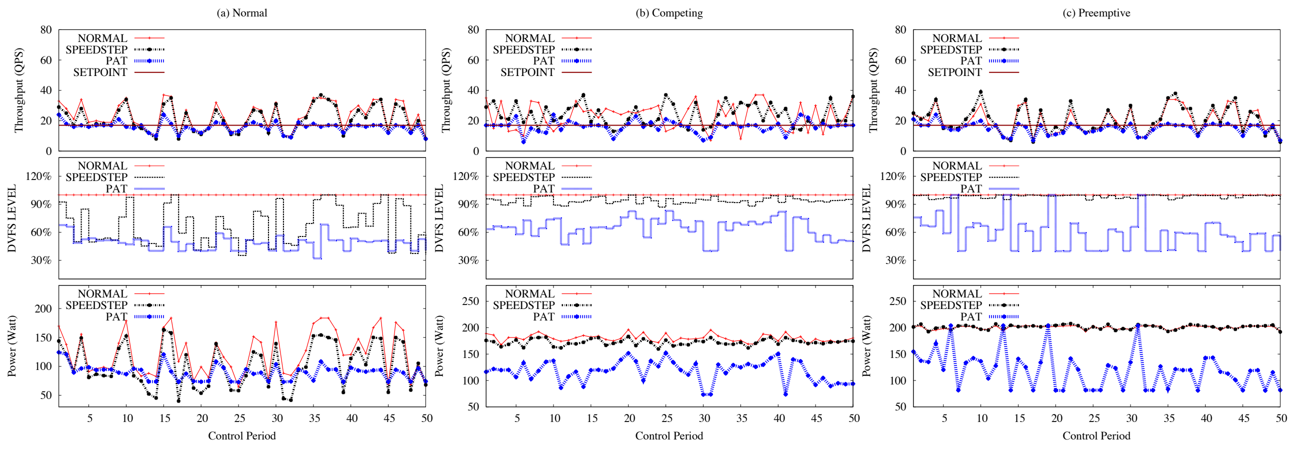

Figure 7: A snapshot of the database throughput, the DVFS control signal and the active power consumption of NORMAL, SPEEDSTEP and PAT in three different system settings.

6 Performance Evaluation

6.1 Experimental Setup

Our test bed contains an open-source database PostgreSQL (version 8.3.18) running under Redhat 5 (kernel version 3.0.0). The data server is a DELL PowerEdge R710 with 12-core Intel Xeon E5645. A client feeds the server with a typical database workload generated from TPC tools [22] and SDSS traces [20]. We use a WattsUp power meter ($\pm 1.5\%$ error, 1 Hz sampling frequency [1]) to measure the power consumption.

We have designed several baselines for evaluation.

1) *NORMAL and TRADITION*: those two baselines are common Advanced Configuration and Power Interfaces (ACPIs) in modern servers. *NORMAL* is when system runs with the maximum CPU frequency and *TRADITION* sets a static CPU frequency based the offline workload analysis.

2) *SPEEDSTEP, HEURISTIC and SCTRL*: *SPEEDSTEP* is the ACPI policy in BIOS that tunes the CPU frequency according to the system load. *HEURISTIC* is an *ad hoc* control solution with the DBMS performance set point R_s. *SCTRL* is an OS-level feedback control solution with the DBMS performance set point R_s. Comparing with PAT, it contains the basic control loop with throughput monitor to detect the DBMS throughput except the FWC and any internal parts of the DBMS in Fig.4. Note that, when those control solutions control the CPU frequency, other power management policies are turned off.

6.2 Performance of PAT

To study the impact of PAT on performance and energy savings, we have designed three scenarios from daily DBMS operations. 1) *Ideal environment*: the database process is the only user of all the computational resources; 2) *Competing environment*, there exists a set of pure CPU intensive programs in the system competing for the CPU resource. 3) *Preemptive environment*, there exists a set of high-priority (OS-level) processes which randomly occupies the CPU resource assigned for database processes. Fig.7 shows the database throughput, the DVFS control signal, and the active power consumption of the system using NORMAL, Speedup and PAT ($R_s = 17$QPS) in above scenarios in 50 control periods.

In Fig.7(a), SPEEDSTEP and PAT provide significantly larger energy savings than NORMAL does. Comparing with SPEEDSTEP, PAT controls the throughput performance strictly to the setpoint, and the maximum overshoot (throughput exceeding the set point) is much smaller.

In the competing scenario in Fig.7(b), the database throughput is greatly affected by the competing CPU-intensive processes, which are injected into the system follows a Poisson distribution. The noise from resource competition between database processes and CPU-intensive processes hurts the control performance of PAT. However, PAT could tolerate such noise and control the throughput back to the setpoint within 3 periods. On the other hand, because SPEEDSTEP controls the CPU frequency based on the total system utilization, it usually sets the DVFS level near the highest level.

Fig.7(c) demonstrates the results in the preemptive scenario. The preemptive behavior of system processes leads to a low DBMS throughput due to the interrupt and resource occupation. It is often the case when the actuator fails to handle the overshoot exceeding its control limit. PAT treats this case as the exception and tunes down the CPU frequency to save more energy, such as the 6th, 13th, 18th, etc. period in Fig.7(c).

Overall, PAT saves up to 51.3% of the energy con-

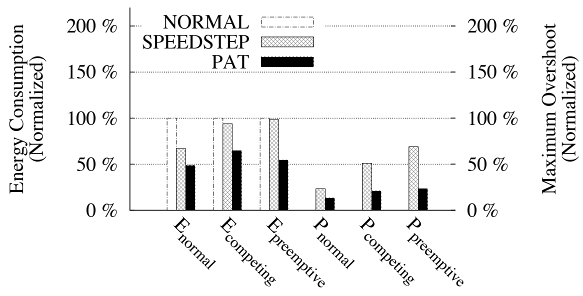

Figure 8: The normalized energy consumption and the performance overshoot of NORMAL, SPEEDSTEP and PAT in three scenarios. The energy cost is normalized to the data of NORMAL.

sumption (15% more than the SPEEDSTEP), comparing with NORMAL in the ideal environment, shown in Fig.8. Its maximum performance overshoot[6] is less than half of the overshoot in SPEEDSTEP, especially in the competing and preemptive scenarios. This is because SPEEDSTEP does not take the DBMS performance as the control goal and the system dynamic gives more error in the last two scenarios. Here we show the advantage of PAT by comparing with ACPI baselines. To further study the performance and the robustness of PAT, we test it with other control baselines to control the DBMS throughput.

6.3 Control Performance Comparison

Fig.9 is the snapshot of the database throughput, DVFS setup and the power consumption of four controllers in the ideal system environment.

TRADITION cannot control the throughput to the set point because the workload does not always follow the pattern in the offline analysis. This is a typical problem of open control. First, finding a good static DVFS for one workload in one system scenario needs extensive experimental work and complex learning processes. Second, it could easily fail under workload variations.

HEURISTIC gives a relatively better control performance, comparing with the SPEEDSTEP. However, when facing an ever-changing workload, HEURISTIC fails to commit to a steady state in an acceptable time. For example, data in control period 20 to 30 in Fig.9(a) show how HEURISTIC fails to handle the "M" shape throughput pattern. While SCTRL and PAT could both commit to the setpoint in 4 periods, the tuning of HEURISTIC oscillates in many steps, which results in less energy savings. Solving the problem will eventually leads to the same feedback controller design in PAT.

SCTRL treats the DBMS processing as a black box. It settles to the setpoint faster than HEURISTIC. However, when all DBMS processes are in I/O busy waiting,

[6] the performance overshoot is measured by P_{max}/R_s, where P_{max} is the maximum performance and R_s is the set point

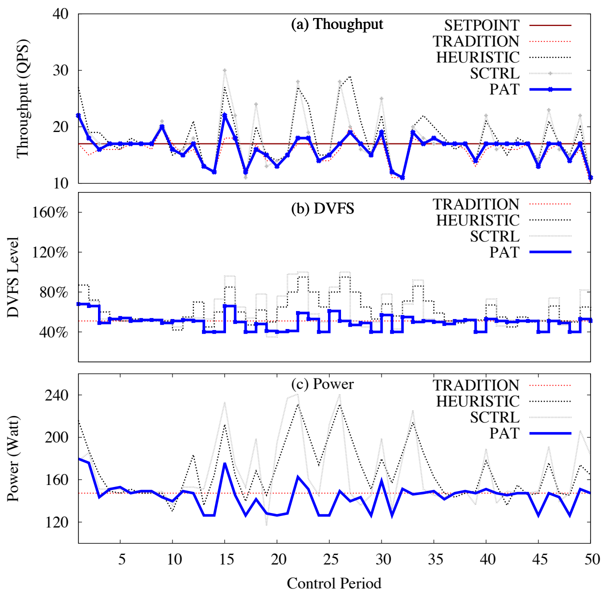

Figure 9: A snapshot of the database throughput, the DVFS control signal and the active power consumption of the four control solutions.

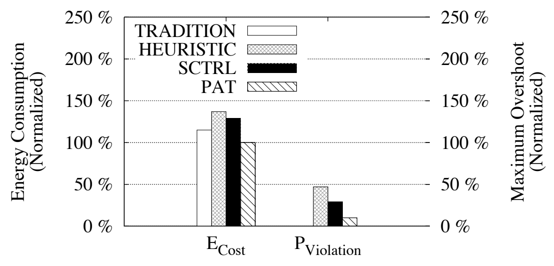

Figure 10: The normalized energy consumption and the performance overshoot of four control solutions.

SCTRL would uselessly set the DVFS level to the maximum, and waste energy.

We conduct the overall performance evaluation of the five control technique in Fig.10. While PAT achieved the 20% more energy savings than that of TRADITION, HEURISTIC and SCTRL only got 56% and 74% of energy savings achieved by PAT because of the failure to commit steady state (HEURISTIC) and the unnecessary setting of the highest DVFS level (SCTRL). Comparing the performance violation, PAT has the smallest maximum overshoot than the other two control methods.

6.4 The performance of FWC

Our fuzzy workload classifier provides a high prediction accuracy of the query resource consumption pattern. The classification result of the tested workload above is shown in Fig.11. The accuracy is above 90% for the two testing traces. FWC provides high accuracy in the system model for controller design in PAT. Theoretically, PAT could tolerate up to 45% overshoot from model and environment based on the controller design. As shown

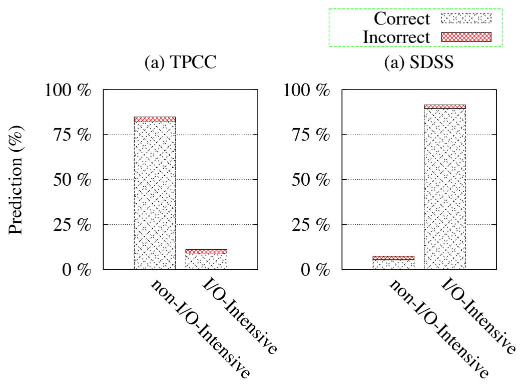

Figure 11: The prediction result of the query processing pattern using FWC.

in Fig.7 and Fig.9, the biggest difference of the workload trends is almost 40% but PAT could still control the throughput to the set point in a few periods.

7 Related Work

Reduction of energy consumption has become an active topic in the DBMS community. Harizopoulos et al. [6] and Graefe [5] introduce a new paradigm of DBMS design concerning energy efficiency. Recent work [30] shows that there exists energy-efficient query plans in DBMS. Another work by Harizopoulos [23] suggests that most energy efficient plans come from DBMS running in the active low power mode. Poess and Nambiar [19] examine multiple storage components in the system for energy saving potentials. Lang et al. [11] claim that it is worthwhile to scale hardware performance to control in DBMS's query processing in the distributed environment. In contrast to their work, we argue that applying hardware scaling technique to DBMS design for energy-saving purposes is not a trivial task and propose a systematic solution that relies on rigorous control loop design. As compared to heuristics-based strategies, our solution provides analytical assurance of control accuracy and system stability.

Application of mathematical control theory has been conducted in several topics in the DBMS area. Tu et al. [24] introduce this technique to handle load shedding in data stream systems by using a classic P feedback controller that successfully avoids noticeable streaming tuple delays with lower data loss. Kang et al. [9] create Chronos by applying a similar feedback model in controlling number of transactions to a baseline. Our paper, unlike those two, is one of the first attempts targeting at energy savings while preserving performance in database systems. It is inspired by the fact that most database servers are running in relatively low utilization – energy proportionality can be achieved if we make database run under low power modes of hardware.

Recently, feedback control theory has been successfully applied to energy efficient control for data center servers at the system and hardware levels [2, 26, 17]. Existing solutions of power and performance control for enterprise servers attempt to tackle the problem in two separate ways. Performance-oriented control solutions focus on altering power to meet the system-level performance budget while reducing power consumption in a best effort manner [16]. However, those solutions do not have any explicit internal information from software, such as DBMS. As a result, there could be undesirable performance degradation. In the other way, power-oriented control solutions treat power as the first-class control target and maximize the performance within the power budget [2, 26, 28]. In DBMS, its throughput could not be maximized by control at the system level because the resource are evenly distributed (Round Robin scheduling in Linux). Thus, we need to build the control loop by taking DBMS statistics into consideration.

8 Conclusion and Future Work

The contradictory requirements of high performance and low energy consumption have attracted a lot of talents working on database system design. The low-power modes of hardware provide opportunities for power saving with predictable performance degradation. In this paper, we tackle the problem of maximizing energy savings under a user-specified performance bound in database systems. We argue that such a problem is non-trivial due to the dynamics in database workloads and environment. Therefore, based on the results of our evaluation, traditional offline analysis and heuristic solutions are not effective. We propose our solution as a feedback control framework based on system characteristics. Unlike heuristic-based adaptive solutions widely used in database tuning, PAT provides performance guarantees over the power control on hardware. We implement PAT with the PostgreSQL engine and the empirical results demonstrate that PAT can achieve high energy efficiency with small violation of SLA. One immediate future work is to consider the performance bound of individual queries using DVFS as the global control actuator.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments. This research is supported by the US National Science Foundation (NSF) grants IIS-1117699, IIS-1156435, and CSR-1143607.

References

[1] Watts up power meter. `http://goo.gl/AI7so`.

[2] M. Chen, X. Wang, R. Gunasekaran, H. Qi, and M. Shankar. Control-based real-time metadata matching for information dissemination. In *RTCSA*, pages 133–142, 2008.

[3] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini. Memscale: active low-power modes for main memory. *SIGPLAN Not.*, 46(3):225–238, Mar. 2011.

[4] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, 1990.

[5] G. Graefe. Database servers tailored to improve energy efficiency. In *SETMDM*, pages 24–28, 2008.

[6] S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan. Energy efficiency: The new holy grail of data management systems research. In *CIDR*, 2009.

[7] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31.

[9] K.-D. Kang, J. Oh, and S. H. Son. Chronos: Feedback control of a real database system performance. In *RTSS*, pages 267–276, 2007.

[10] M. Kunjir, P. K. Birwa, and J. R. Haritsa. Peak power plays in database engines. In *Proc. of EDBT*, EDBT '12, pages 444–455. ACM, 2012.

[11] W. Lang, R. Kandhan, and J. M. Patel. Rethinking query processing for energy efficiency: Slowing down to win the race. *IEEE Data Eng. Bull.*, 34(1):12–23, 2011.

[12] C. Lefurgy, X. Wang, and M. Allen-Ware. Server-level power control. In *ICAC*, page 4, 2007.

[13] MATLAB. *version 7.13.0 (R2011b)*. The MathWorks Inc., Natick, Massachusetts, 2011.

[14] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch. Power management of online data-intensive services. In *ISCA*, pages 319–330, New York, NY, USA, 2011. ACM.

[15] D. Meisner and T. F. Wenisch. Dreamweaver: architectural support for deep sleep. In *ASPLOS*, pages 313–324, 2012.

[16] R. G. Melhem, D. Mossé, and E. N. Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. Computers*, 53(2):217–231, 2004.

[17] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated control of multiple virtualized resources. In *EuroSys*, pages 13–26, 2009.

[18] M. Poess and R. O. Nambiar. Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results. *PVLDB*, 1(2):1229–1240, 2008.

[19] M. Poess and R. O. Nambiar. Tuning servers, storage and database for energy efficient data warehouses. In *ICDE*, pages 1006–1017, 2010.

[20] Sloan Digital Sky Survey. `http://goo.gl/hOVDP`.

[21] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *Ieee Transactions On Systems Man And Cybernetics*, 15(1):116–132, 1985.

[22] Transaction Processing Performance Council. `http://www.tpc.org`.

[23] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the energy efficiency of a database server. In *SIGMOD*, pages 231–242, 2010.

[24] Y.-C. Tu, S. Liu, S. Prabhakar, and B. Yao. Load shedding in stream databases: A control-based approach. In *VLDB*, pages 787–798, 2006.

[25] L. Wang, J. Xu, M. Zhao, Y. Tu, and J. A. B. Fortes. Fuzzy modeling based resource management for virtualized database systems. In *MASCOTS*, pages 32–42, 2011.

[26] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller. Ship: Scalable hierarchical power control for large-scale data centers. In *PACT*, pages 91–100, 2009.

[27] Y. Wang, X. Wang, M. Chen, and X. Zhu. Partic: Power-aware response time control for virtualized web servers. *IEEE Trans. Parallel Distrib. Syst.*, pages 323–336, 2011.

[28] Q. Wu, P. Juang, M. Martonosi, L.-S. Peh, and D. W. Clark. Formal control techniques for power-performance management. *IEEE Micro*, 25(5):52–62, 2005.

[29] Z. Xu. Model Evaluation of PAT, a comprehensive study. Technical report. `http://goo.gl/Cd2sN` (Link is shortened).

[30] Z. Xu, Y.-C. Tu, and X. Wang. Exploring power-performance tradeoffs in database systems. In *ICDE*, pages 485–496, 2010.