OptiTuner: On Performance Composition and Server Farm Energy Minimization Application

Jin Heo, *Student Member*, *IEEE*, Praveen Jayachandran, *Student Member*, *IEEE*, Insik Shin, *Member*, *IEEE*, Dong Wang, Tarek Abdelzaher, *Member*, *IEEE*, and Xue Liu, *Member*, *IEEE*

Abstract—This paper develops a software service for dynamic performance optimization and control in performance-sensitive systems. The next generation of performance-sensitive systems is expected to be more distributed and dynamic. They will have multiple "knobs" that affect performance and resource allocation. However, relying on the conglomeration of independent knob controls can become increasingly suboptimal. The problem lies in *performance composability* or lack thereof; a challenge that arises because individual optimizations in performance-sensitive systems generally do not compose well when combined. Performance adaptation in such systems needs to be carefully designed and implemented by holistically considering performance composability in order to achieve desired system performance. A flexible supporting software layer is therefore needed to easily apply different holistic performance management techniques. In this paper, we develop a software service, called *OptiTuner*, that monitors the current performance and the resource availability in performance-sensitive systems and allows easy implementation of different performance management schemes based on theoretical concepts of constrained optimization and feedback control. In order to show the efficacy of OptiTuner, we apply it to implement three holistic energy minimization techniques in a real-time web server farm comprising 18 machines. Using an industry standard e-Business benchmark, TPC-W, we demonstrate that the three approaches save up to 40 percent of total energy cost compared to the baseline approaches that do not holistically optimize the cost.

Index Terms—Performance composability, software service, energy minimization, data center, server farm.

1 Introduction

This paper develops a software service for performance management in large-scale performance-sensitive systems. The topic is motivated by the growing need for adaptive components to automatically manage and control system performance in modern software systems [7].

While adaptive capabilities can considerably diminish the need for human intervention, controlling and optimizing performance of large-scale performance-sensitive systems are becoming an extremely difficult problem. Subtle interactions between individually well-tuned components may result in adverse emergent behavior when combined contributing to significant performance degradation [8], [7]. For example, in our previous work [8], we presented an example of bad interactions between two energy saving policies in a three-tier web server farm, where the interactions of the two policies eventually caused the server farm to use more machines than needed, spending excessive energy. The contribution of our software service lies in facilitating the implementation of different holistic performance management mechanisms in large software systems based on principles of constrained optimization and control. Three such mechanisms are implemented and compared in a case study on energy minimization in server farms.

To address the problem of bad interactions between adaptive components, when designing adaptive components in performance-sensitive systems, *performance composability*, or the ability of individually well-optimized components to compose in ways that achieve good aggregate behavior, should be considered carefully. Various holistic design techniques for performance management such as optimization techniques and feedback control approaches can be considered to ensure proper adaptation to dynamic changes in the external environment or in system load that require manipulating performance knobs. A flexible supporting software layer is therefore required to register the available performance tuning mechanisms (called control knobs) in a large system and coordinate their operation to achieve or approach stated performance goals subject to applicable constraints. In this paper, we present *OptiTuner*, a software service designed to achieve the above.

OptiTuner provides simple abstractions and necessary services to support common operations of various performance optimization and feedback control techniques, such as constantly monitoring the current performance and resource availability of the target system and dynamically adjusting performance control knobs. OptiTuner then runs a collection of performance management modules connecting performance monitoring and control knob manipulation modules. All modules are pluggable and can be distributed, allowing flexible development for performance-sensitive systems.

- *J. Heo is with VMware, Inc., 3401 Hillview Ave, Palo Alto, CA 94301. E-mail: jinheo@gmail.com.*
- *P. Jayachandran is with IBM Research-India, A101, Skyline Atlantis, Old Madras Road, Benninganahalli, Bangalore 560093, India. E-mail: praveenj83@gmail.com.*
- *I. Shin is with the Department of Computer Science, KAIST, 335 Gwahangro, Yuseong-gu, Daejeon, South Korea 305-701. E-mail: insik.shin@cs.kaist.ac.kr.*
- *D. Wang and T. Abdelzaher are with the Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N Goodwin Ave, Urbana, IL 61801. E-mail: dwang24@illinois.edu, zaher@cs.uiuc.edu.*
- *X. Liu is with the Department of Computer Science and Engineering, 104 Schorr Center, University of Nebraska-Lincoln, Lincoln, NE 68588-0150. E-mail: xueliu@cse.unl.edu.*

Manuscript received 6 Mar. 2009; revised 14 Mar. 2010; accepted 17 Oct. 2010; published online 20 Jan. 2011.
Recommended for acceptance by C. Aykanat.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-03-0105.
Digital Object Identifier no. 10.1109/TPDS.2011.52.

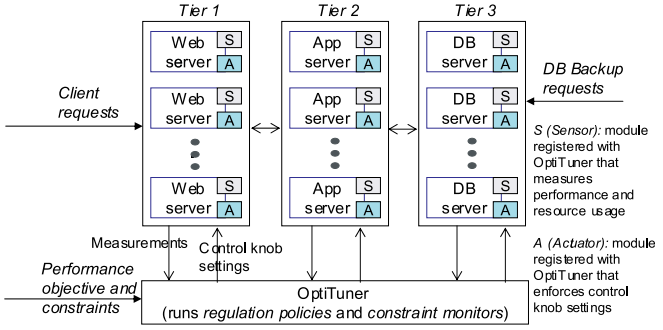


Fig. 1. An energy minimization application in a three-tier web server farm with OptiTuner.

To show the efficacy of OptiTuner, we implement and evaluate three different holistic approaches for performance management that have been widely used in achieving desired performance in computing systems—centralized optimization [12], [15], distributed optimization [2], [14], and multi-input and multioutput (MIMO) feedback control [4], [18]—in the context of energy minimization in a web server farm testbed of 18 machines. We show how OptiTuner can be used to apply the three techniques to implement power management for the application. Further, through a thorough evaluation process using an industry standard e-Business benchmark, TPC-W, we show that the three holistic energy minimization approaches can save more energy compared to the baselines, demonstrating that OptiTiuner is indeed useful for developing performance optimization and control for performance-sensitive systems.

The rest of the paper is organized as follows: In Section 2, we describe the design of OptiTuner and develop an energy minimization application in a three-tier web server farm using three different performance management techniques. We evaluate OptiTuner in Section 3 and the related work is presented in Section 4. We conclude the paper with Section 5.

2 Design and Approaches

To illustrate how OptiTuner can be used for designing and implementing performance-sensitive systems, we develop an energy cost minimization application for three-tier web server farms as shown in Fig. 1. We assume that a server farm runs a typical three-tier web application and provides a database backup service for other organizations to fully utilize its extra computing power. The total energy cost incurred by the system is composed of two parts: 1) the energy cost of server machines in the system less and 2) the utility achieved from database backup requests. The objective of the optimization problem is to minimize the total cost incurred by the system, subject to the delay constraints of client requests and a resource constraint on the number of available servers.

We use three different types of performance control knobs to adjust power consumption. First, we adjust the number of assigned machines by dynamically turning machines on and off. Second, dynamic voltage frequency scaling on an individual processor is used to change the speed and voltage of the processors in an active machine. Finally, we adjust the rate of the incoming database backup requests for the database tier using admission control. The three types of performance control knobs (that we will call On/Off, DVS, and BackupAC, respectively) need to be continuously coordinated to reduce bad interactions among them, thereby effectively minimizing energy. OptiTuner is designed to easily facilitate the implementation of holistic performance management approaches for such purpose.

Focusing on the critical issue of coordination of different knob settings in performance-sensitive systems, OptiTuner provides a supporting software layer for composing performance adaptation modules in ways that follow sound theoretical concepts of performance optimization and control. The contribution lies in the ease of applying different holistic optimization and control techniques simply by changing the policy modules that sit between performance measurement and control interfaces.

The common structure observed in different performance adaptation design techniques leads to a set of abstractions exported by OptiTuner. In OptiTuner, performance control knobs are governed by *regulation policies*. Regulation policies in OptiTuner are essentially performance adaptation modules periodically determining the values of the performance control knobs. In order to help the decision of regulation policies effectively, OptiTuner exports *performance sensors* that are thin software interfaces wrapped around pertinent performance measurements. *Actuators* are software modules that enforce the settings of the control knobs determined by regulation policies. *Constraint monitors* monitor the availability of resources to inform individual regulation policies of how much further performance can be improved without breaking the constraints. These abstractions are implemented as corresponding objects registered with OptiTuner.

In order to show the use and efficacy of OptiTuner, we implement three different holistic performance management approaches for the server farm energy minimization application that have been successfully applied to performance-sensitive systems recently: 1) a centralized optimization approach, 2) a distributed optimization approach, and 3) finally, a MIMO feedback control approach. We then evaluate and compare the three approaches in Section 3.

2.1 Centralized Performance Optimization

In this section, we implement a centralized optimization approach for an energy minimization application in server farms using OptiTuner. A centralized optimization approach is a natural choice for achieving good performance given multiple control knobs to tune. The goal of optimization is to find a best combination of the three types of control knobs in the server farm that minimizes power consumption.

2.1.1 Optimization Formulation

We assume that the load is evenly distributed across the machines at each tier in steady state. That is, all m_i machines belonging to a tier i operate at the same frequency f_i. Let $P_{tier}(i)$ be the power consumption of tier i in the system, which is a function of the DVS frequency level and number of machines operating in that tier. λ_{backup} is the rate at which database backup requests are admitted (in cycles/sec). The end-to-end delay bound of client requests is denoted by K. The total number of machines available is denoted by M, and each machine is either turned off or is operating at one of the three tiers. It is desired to minimize

$$\min \quad \sum_{i=1}^{3} P_{tier}(i) - h \cdot \log(\lambda_{backup}) \qquad (1)$$

$$\text{subject to} \quad \sum_{i=1}^{3} Delay(i) \le K, \quad \sum_{i=1}^{3} m_i \le M. \tag{2}$$

Observe that the formulation assumes that utility for backup requests increases logarithmically with backup request rate. Power consumption at tier i, $P_{tier}(i)$, is estimated based on the current frequency, f_i, and the number of machines operating at tier i:

$$P_{tier}(i) = m_i \cdot P_{machine} = m_i \cdot \left(A_i \cdot f_i^n + B_i\right), \tag{3}$$

where B_i is the power consumption that is independent of the current frequency and A_i is a positive constant that indicates the effect of the frequency on power consumption. A_i, B_i, and n can be measured using offline experiments. We will use the value 3 for n for the preliminary results in the next section.

In our testbed, f_i has eight different discrete values, while optimization works more efficiently when all variables are continuous. We effectively approximate f_i as a continuous variable by replacing it with U_i which is continuous. After getting U_i, we can closely approximate the given U_i by choosing an effective frequency that makes the utilization closest to U_i. Alternatively, a time-multiplexed combination of frequencies can be used as described in [13]. To replace f_i with U_i, the steady-state result of an M/M/1 queuing model [11], $utilization = \lambda/\mu$, is used, where μ is the service rate and λ is the arrival rate. We use a queuing model, because it has been widely used for estimating various performance metrics for computing systems, such as delay and resource utilization [10]. It yields

$$U_i = \frac{\lambda_i/m_i}{f_i} = \frac{\lambda_i}{f_i m_i}, \tag{4}$$

where λ_i denotes the rate of arrival of requests to tier i and U_i denotes the utilization of a machine at tier i. Using the relationship shown in (4), we rewrite (3):

$$P_{tier}(i) = m_i \cdot P_{machine} = m_i \cdot \left(\frac{A_i \lambda_i^3}{U_i^3 m_i^3} + B_i\right). \tag{5}$$

Also, λ_{backup} is estimated using the relationship between U_{backup}, m_3, and f_3 from (4):

$$\lambda_{backup} = U_{backup} \cdot m_3 f_3 = (U_3 - U_{user}) m_3 f_3, \tag{6}$$

where U_{backup} is the utilization of backup requests alone at the third tier and U_{user} denotes the utilization of user requests.

Assuming the load is equally distributed over the machines at each tier, the delay experienced at each tier i, $Delay(i)$, is estimated using the steady-state result of M/M/1 queuing:

$$Delay(i) = \frac{C_i}{f_i - \frac{\lambda_i}{m_i}} = \frac{m_i C_i}{\lambda_i} \cdot \frac{\frac{\lambda_i}{m_i f_i}}{1 - \frac{\lambda_i}{m_i f_i}} = \frac{m_i C_i}{\lambda_i} \cdot \frac{U_i}{1 - U_i}, \tag{7}$$

where C_i is a constant in cycles per request to normalize the delay, since f_i is measured in cycles and λ_i is estimated in cycles/sec rather than in requests/sec.

Finally, the resulting cost minimization problem is formulated with the control knobs,

$$\mathbf{x} = [U_1\ U_2\ U_3\ U_{backup}\ m_1 m_2]^{\mathrm{T}},$$

and the two constraints as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{3} m_i \left(\frac{A_i \lambda_i^3}{U_i^3 m_i^3} + B_i\right) - h \cdot \log(U_{backup} \cdot m_3 f_3) \\ \text{subject to} \quad & \sum_{i=1}^{3} \frac{m_i C_i}{\lambda_i} \cdot \frac{U_i}{1 - U_i} \le K, \\ & \sum_{i=1}^{3} m_i \le M. \end{aligned} \tag{8}$$

Observe that in the current testbed, the On/Off knob in the third tier is inactive due to data migration problems (we will explain about our testbed, in detail, in Section 3). Therefore, in our testbed, the centralized optimization approach determines six knob settings instead of seven. This also applies to the other two design techniques explained below. Hence, m_3 is a constant and the goal of optimization is to find the set of six knob settings, $\mathbf{x} = [U_1\ U_2\ U_3\ U_{backup}\ m_1\ m_2]^{\mathrm{T}}$, that minimizes the cost function subject to the two constraints.

2.1.2 Implementation of Regulation Policies in OptiTuner

We use Shor's r-algorithm to solve the constrained minimization problem shown in (8), because the success of the algorithm for nonlinear optimization has been reported recently [1].

Since it is a centralized solution, we implement a regulation policy running on a dedicated machine that periodically solves the optimization formulation to determine control knob settings at each control period t. One of the machines at each tier is then selected as the leader to run a regulation policy that periodically contacts the central optimization solver to set the control knobs for the next control period $t + 1$.

We also implement tierwise sensor objects called *TierSensor* that collect performance measurements and resource usage information such as the utilization, request rate, and response time for the corresponding tier that are required to solve the optimization problem. They aggregate performance measurements collected from sensors instrumented in individual machines. Similarly, we implemented two tierwise actuators, *TierDVSActuator* and *TierOn/OffActuator*, that act on the DVS and On/Off actuators instrumented in individual machines.

2.2 Distributed Performance Optimization

A disadvantage of the above approach is that it is centralized. In this section, we describe distributed performance optimization using OptiTuner. This approach uses optimization decomposition [2], recently applied at length in network theory, to break complex systemwide global optimization problems into less coupled subproblems that can be optimally solved at runtime in a distributed fashion.[1]

We can use the same optimization formulation shown in (8) since it is a convex problem. In order to derive subproblems from the optimization formulation, we first get the Lagrangian of the problem (8) as

1. For details about applying optimization decomposition to performance-sensitive systems in general, please refer to our full-length paper published in the IEEE Explorer digital library with the electronic version of this paper, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.52.

$$L(x,p_1,p_2)=\sum_{i=1}^{3} m_i\left(\frac{A_i\lambda_i^3}{U_i^3 m_i^3}+B_i\right)-h\cdot\log(U_{backup}\cdot m_3 f_3) + p_1\cdot\left(\sum_{i=1}^{3}\left(\frac{m_i C_i}{\lambda_i}\cdot\frac{U_i}{1-U_i}\right)-K\right) + p_2\cdot\left(\sum_{i=1}^{3}(m_i)-M\right), \tag{9}$$

where $p_1, p_2 \geq 0$ denote the Lagrange multipliers for the two constraints.

By differentiating the Lagrangian with respect to each of the optimization variables, we can decompose the problem into subproblems that adapt their knob settings individually. Six different regulation policies are created to iteratively adjust the six knob settings, $x=[U_1\ U_2\ U_3\ U_{backup}\ m_1\ m_2]^T$ (m_3 is a constant in our testbed). A constraint monitor is created to adapt the two price values, $p1$ and $p2$, for the two constraints.

By differentiating the Lagrangian with respect to U_1 and U_2, the update equations for the DVS policies at tiers 1 and 2 are obtained as

$$U_i(t+1)=\left[U_i(t)-\alpha_{U_i}\left(-\frac{3A_i\lambda_i^3}{m_i^2 U_i(t)^4}+\frac{p_1 m_i C_i}{\lambda_i(1-U_i(t))^2}\right)\right]^+. \tag{10}$$

Once U_i is calculated, the corresponding frequency, f_i, is determined using (4). Since the frequency value is discrete in a real system, we choose the closest frequency setting to the calculated value, f_i. The DVS Policy at tier 3 updates its knob setting U_3 similarly:

$$U_3(t+1)=\left[U_3(t)-\alpha_{U_3}\cdot\left(-\frac{3A_3\lambda_3^3}{m_3^2 U_3(t)^4} + \frac{p_1 m_3 C_3}{\lambda_3(1-U_3(t))^2}-\frac{h}{U_3(t)-U_{user}}\right)\right]^+. \tag{11}$$

The DVS policy at tier i determines its control knob setting, U_i (hence f_i) every second. For all tiers, we use 0.05 for the step size constants, α_{U_i}. Observe that all time scales and step sizes used in this approach are empirically chosen. Formal stability analysis of convex optimization solvers can be found in [2].

Since the control knob settings U_{backup} and U_3 are related to each other, we calculate U_{backup} based on U_3:

$$U_{backup}(t+1)=\frac{\lambda_{backup}}{\lambda_3}U_3(t+1). \tag{12}$$

With the calculated U_{backup}, the BackupAC policy adjusts the portion of the incoming database backup requests accordingly using a control-theoretic approach.

In the same way as the DVS policies, the On/Off policies at tiers 1 and 2 update their control knob settings m_1 and m_2 at each invocation period according to the update rules derived by differentiating the Lagrangian with respect to m_1 and m_2. The constraint monitor periodically adjusts p_1 and p_2 using the update rules obtained in a similar way. The details can be found in our full-length paper.

2.2.1 Implementation of Regulation Policies in OptiTuner

The decomposition of the optimization problem results in regulation policies to adjust the six different performance control knobs and a constraint monitor to adjust the two price values for the constraints.

Three *DVSPolicy* objects were implemented to perform the adaptation rules, described in (10) for the first and the second tier and (11) for the third tier. Similarly, two *On/OffPolicy* objects were implemented to perform the adaptation rules for m_1 and m_2. The *BackupACPolicy* object executes the update rules in (12). As all policies work on tiers instead of individual machines, they are placed in the leader machines. The *ConstraintMonitor* object implements the update rules for the two constraints and runs on the leader machine of each tier.

As in the centralized optimization approach, the tierwise sensor *TierSensor* objects, are used to collect performance data and two actuators, *TierDVSActuator* and *TierOn/OffActuator*, are used to enforce the control knob settings.

2.3 MIMO Feedback Control

In this section, we implement a MIMO controller using OptiTuner. The implemented MIMO controller dynamically finds the control knob settings,

$$\mathbf{x}=[U_1\ U_2\ U_3\ U_{backup}\ m_1\ m_2]^{\mathrm{T}},$$

to keep the end-to-end delay constraint. At the same time, by adjusting the three types of power saving control knobs, power consumption is reduced. The MIMO controller designed in this paper is based on the one used in [18]. We chose it because the MIMO controller they used is directly applicable to our energy minimization application. This also proves the usefulness of OptiTuner because it can be used to implement an existing design technique without difficulties.

2.3.1 Controller Design

Since the goal of the controller is to minimize the control error, we first define the control error $e(t)$ at each control period t as the difference between the current average delay $D(t)$ and the end-to-end delay constraint K, hence, $e(t)=D(t)-K$. The controller tries to keep the error as small as possible by adjusting the control input (i.e., the control knobs, $\mathbf{x}$).

Before adjusting the control knob values $\mathbf{x}$, they are normalized by subtracting their average, $\mathbf{\Delta x(t)}=\mathbf{x(t)}-\overline{\mathbf{x}}$, where $\overline{\mathbf{x}}$ is the average of $\mathbf{x(t)}$. This is the process of linearization around the operating point because our controller assumes a linear system. Hence, the designed controller periodically adjusts a control knob change vector $\mathbf{\Delta x(t)}=[\Delta x_1(t),\ \ldots,\ \Delta x_n(t)]$ with given the current error $e(t)$ to minimize the next error $e(t+1)$, where n is the number of control knobs.

In order to effectively design a controller, we need a dynamic model describing the relationship between the control input $\mathbf{\Delta x}(t)$ and the output $e(t)$. We apply *system*

identification to derive a linear model for designing our controller as follows:

$$e(t) = \mathbf{A}\ e(t-1) + \mathbf{B}\ \mathbf{\Delta x}(t-1), \tag{13}$$

where $\mathbf{A}$ is a $n \times 1$ matrix and $\mathbf{B}$ is a $n \times n$ matrix and n is the number of control knobs.

Using the linear model identified, we design our controller using the Linear Quadratic Regulator (LQR) control [6]. The LQR is a well-known control technique that provides practical feedback gains for MIMO control problems. We determine the feedback gain matrix $\mathbf{F}$ for the LQR controller using the matlab *dlqry* function:

$$\mathbf{F} = [\mathbf{K_P} \quad \mathbf{K_I}], \tag{14}$$

where $\mathbf{K}_P$ and $\mathbf{K}_I$ are constant matrix. Given the gain matrix $\mathbf{F}$, the controller calculates a control knob change vector $\mathbf{\Delta x}$ at each control period t:

$$\mathbf{\Delta x}(t) = -\mathbf{F}[e(t-1) \quad v(t)]^{\mathrm{T}}, \tag{15}$$

where $v(t)$ represents the accumulated error used for applying the integral gain $\mathbf{K}_I$ to reduce residual error. $v(t)$ is defined as follows:

$$v(t) = \lambda\ v(t-1) + e(t-1), \tag{16}$$

where λ is a forgetting factor indicating the importance of the past accumulated error. We use 0.8 for λ.

2.3.2 Implementation with OptiTuner

We implement a regulation policy running on a dedicated machine for the designed MIMO controller. The leader machine at each tier then periodically contacts the controller (the regulation policy in the dedicated machine) to extract the control knob values for the next control period. We use the same sensor objects and actuator objects as in the centralized optimization and distributed optimization approaches to compare performance fairly.

This reuse is also the main reason that OptiTuner simplifies the development of different control and optimization policies. While the theory behind each policy may be different, they all share the same sensor and actuator modules, which, in the system implementation, account for the majority of the code. Moreover, they all share the need for communication across these modules, which is handled by the service.

3 Evaluation

In this section, we show the evaluation results of OptiTuner with an energy minimization application in our three-tier web server farm testbed of 18 machines.

3.1 Testbed Setup

In this section, we briefly explain our testbed for a three-tier web server farm with a total of 18 machines. Four machines are used to run a load balancers, a backup request generator, and a client workload generator. An admission control mechanism for backup requests is installed on the same machine with the backup request generator. Tiers 1 and 2 were initially given five machines each and tier 3 was given four machines. Each machine belongs to a specific tier and this per-tier allocation is static. The On/Off Policy in the third tier is inactive in the current testbed, since the database clustering solution does not fully support consistent data migration between replicas.

We installed a three-tier web application on our testbed based on TPC-W, a transactional web benchmark specifically designed for evaluating e-commerce systems. The performance objective is to minimize energy cost subject to the end-to-end delay constraint of 0.5 sec (500 msec) and the resource constraint of the given 14 machines.

Power consumption is estimated based on frequency samples of processors measured every second, using (3). It allows us to evaluate various energy saving approaches on system settings with different DVS capabilities. In (3), B represents the fixed energy consumption regardless of the current frequency and A is a coefficient for calculating the effect of frequency changes. More detailed explanation about the power estimation is presented in our previous paper [8].

3.2 Experimental Results

We evaluate six different energy saving approaches including the three holistic approaches presented in Section 2 and three baseline approaches.[2]

- The Ondemand governor approach uses only DVS and BackupAC knobs. DVS knobs are adjusted by the Linux Ondemand governor [16], which is a dynamic in-kernel CPU frequency governor that changes CPU frequency levels based on utilization. BackupAC policy controls incoming backup requests independent of the Linux Ondemand governor based on the delay constraint.
- The independent approach uses three independent regulation policies for On/Off, DVS, and BackupAC knobs. They independently determine their knob settings based on the current load.
- The RTSS07-independent approach jointly adapts DVS and On/Off knobs using feedback-based heuristics [8], with a BackupAC policy running independently.
- The centralized approach jointly adapts DVS, On/Off, and BackupAC knobs by solving the global optimization formulation (explained in Section 2.1).
- The distributed approach has the DVS, On/Off, and BackupAC policies for the control knobs coadapted as per the optimization decomposition methodology described (explained in Section 2.2).
- The MIMO control approach adapts DVS, On/Off, and BackupAC knobs as determined by the MIMO controller designed (explained in Section 2.3).

Figs. 2, 3, and 4 show performance comparison (total energy cost) of the six approaches with different values of the parameter B representing different DVS capabilities. From the figures, the three holistic approaches (the centralized, distributed, and MIMO control) save more total energy cost in all three different system settings than the other three approaches. For example, at high loads, the

2. We only show a part of the evaluation results due to the space limit. For all evaluation results, please refer to our full-length paper published in the IEEE Explore digital library, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.52.

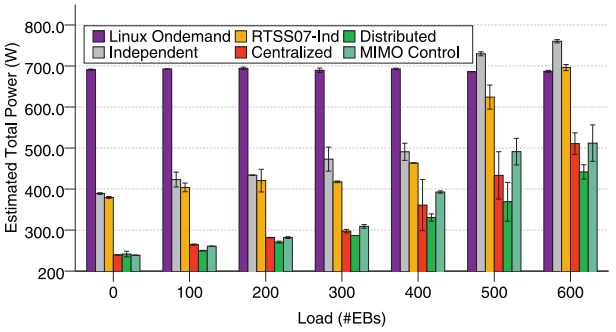


Fig. 2. Performance (total energy cost) comparison between various approaches with backup requests: When B is 70 percent of the maximum power.

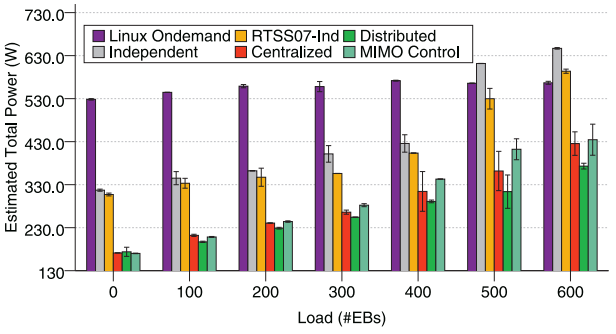


Fig. 3. Performance (total energy cost) comparison between various approaches with backup requests: When B is 50 percent of the maximum power.

distributed approach saves at least 15 percent and up to 40 percent over the Independent approach. Also, the distributed approach saves as much as 40 percent and at least 20 percent at low loads depending on the DVS configuration compared to the Linux Ondemand approach.

Among the top three approaches, the distributed approach performs slightly better than the centralized and MIMO control approaches. One of the possible reasons why the distributed approach performs slightly better than the centralized approach is that the distributed approach uses a measured delay when adjusting the price value for the delay constraint, thereby accurately responding to workload changes. On the contrary, the centralized approach uses an estimated delay when solving the minimization problem. Further, the distributed approach performs a little better than the MIMO control approach because the MIMO control approach presented in this paper minimizes power only indirectly by keeping the end-to-end delay around a set point, while manipulating energy savings.

While the Linux Ondemand approach performs poorly at low workloads (with more machines turned on than required) in terms of energy cost, it performs better than the independent and the RTSS07-independent approaches at high load (see Figs. 2, 3, and 4). This is because, as the workload increases, the settings of all three control knobs are not jointly adapted in these approaches. Hence, they incur more cost than the Linux Ondemand approach that uses only two control knobs.

Fig. 5 shows the average number of machines used during the experiments. We only consider tier 1 and tier 2 machines to calculate averages because the number of machines at tier 3 is fixed. The Linux Ondemand approach shows the largest values as expected, as they use a fixed

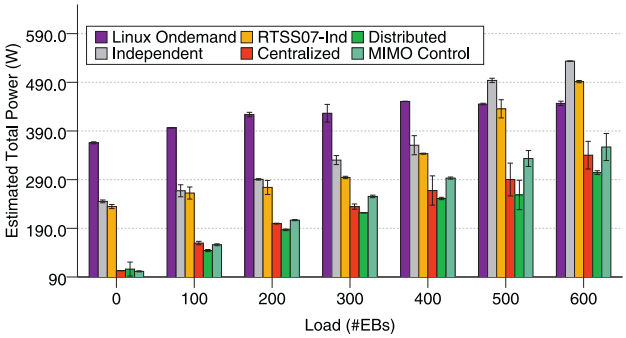


Fig. 4. Performance (total energy cost) comparison between various approaches with backup requests: When B is 30 percent of the maximum power.

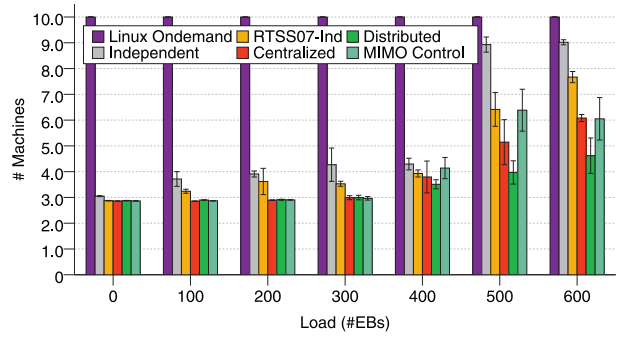


Fig. 5. Average number of machines for tiers 1 and 2.

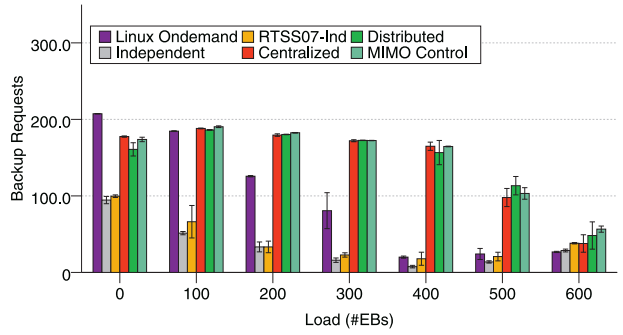


Fig. 6. Served backup requests.

number of machines. Interestingly, at high loads, the independent approach uses almost all available machines, hence spending more energy. Similarly, the RTSS07-independent approaches uses more machines than the top three approaches at high loads. This is an instance of undesirable interactions between regulation policies when executed independently. This result shows the need for design techniques to coordinate the adjustment of control knobs. The centralized, distributed, and MIMO control approaches show smaller average values than the three baseline approaches confirming their control knob adjustment is coordinated, saving more energy.

This ill-coordinated behavior at high workloads in the independent and the RTSS07-independent approaches is also shown in Figs. 6 and 5. At high loads, the number of backup requests served by these approaches actually increases with system load as shown in Fig. 6.

Finally, the three holistic approaches—the centralized, distributed, and MIMO control—do not compromise on throughput or delay in order to achieve the reduced energy cost (We do not show the figures here due to the page

length limit). By admitting more backup requests when the user load is low, and by always coordinating adjustment of control knobs, they save considerable amount of energy cost compared to the baselines.

4 Related Work

Utility maximization has been widely adopted in many areas such as real-time systems and wireless sensor networks [12], [8], [15]. In this paper, we implement a centralized optimization approach using OptiTuner and show that it can effectively coordinate adjustment of multiple control knobs.

Utility maximization problems can be also solved in a distributed way under certain assumptions (e.g., convexity of problems). Recent breakthroughs in networking literature have led to the development of a mathematical theory for optimally layering network protocols using *optimization decomposition* techniques [2], [3], [17] to maximize the global utility of the involved network components. In this work, we show how optimization decomposition techniques can be applied to distributed performance optimization and resource management problems in performance-sensitive systems.

MIMO control has been successfully used to maintain the desired performance in performance-sensitive systems [4], [18]. In this paper, we implement an MIMO feedback algorithm used in [18] to show that OptiTuner can be easily used to apply an existing design technique. Further, through evaluation, we show that the presented MIMO control approach exhibits comparable performance to the two optimization approaches, successfully reducing undesirable interactions between different power saving mechanisms.

With an expanding number of server machines in large server farms and data centers, a large portion of maintenance cost is due to energy bills [5], [8]. This is because they are typically overprovisioned based on offline analysis. Hence, significant research efforts have been expended to save energy consumption of computing systems [5], [8], [13], [18]. In this work, we consider the problem of applying multiple energy saving mechanisms (such as DVS and switching off machines), and jointly optimizing them for performance subject to resource constraints.

5 Conclusion

In this paper, we presented a software service for achieving performance composability in distributed performance-sensitive systems. OptiTuner provides proper abstractions and necessary services to help the implementation of performance management approaches for coordinating different knob settings. To evaluate and demonstrate the efficacy of OptiTuner, we implemented three different holistic design techniques that have been widely used in achieving desired performance in computing systems—centralized optimization, distributed optimization, and MIMO feedback control approaches—and applied them to an energy minimization application in a web server farm. Results from a testbed with 18 machines showed that those approaches were able to reduce total energy consumption considerably compared to the baseline approaches through intelligently coordinating control of multiple performance knobs.

Acknowledgments

This work is an extended version of a workshop paper titled "OptiTuner: An Automatic Distributed Performance Optimization Service and a Server Farm Application" in Proceedings of the Fourth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks [9].

References

[1] J.V. Burke, A.S. Lewis, and M.L. Overton, "The Speed of ShorS R-Algorithm," *IMA J. Numerical Analysis,* vol. 28, no. 4, pp. 711-720, 2008.

[2] M. Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle, "Layering as Optimization DecomPosition: A Mathematical Theory of Network Architectures," *Proc. IEEE,* vol. 95, no. 1, pp. 255-312, Jan. 2007.

[3] R.L. Cruz and A. Santhanam, "Optimal Routing, Link Scheduling, and Power Control in Multihop Wireless Networks," *Proc. IEEE INFOCOM,* vol. 1, pp. 702-711, 2003.

[4] Y. Diao, G. Neha, J.L. Hellerstein, S. Parekh, and D.M. Tilbury, "Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics with Application to the Apache Web Server," *Proc. Network Operations and Management Symp. (NOMS '02),* pp. 219-234, 2002.

[5] E.N. Elnozahy, M. Kistler, and R. Rajamony, "Energy Conservation Policies for Web Servers," *Proc. USENIX Symp. Internet Technologies and Systems,* 2003.

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

[7] J. Heo and T. Abdelzaher, "AdaptGuard: Guarding Adaptive Systems from Instability," *Proc. Sixth Int'l Conf. Autonomic Computing (ICAC '09),* pp. 77-86, 2009.

[8] J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher, "Integrating Adaptive Components: An Emerging Challenge in Performance-Adaptive Systems and a Server Farm Case-Study," *Proc. 28th IEEE Real-Time Systems Symp. (RTSS '07)* , pp. 227-238, 2007.

[9] J. Heo, P. Jayachandran, I. Shin, D. Wang, and T. Abdelzaher, "OptiTuner: An Automatic Distributed Performance Optimization Service and a Server Farm Application," *Proc. Fourth Int'l Workshop Feedback Control Implementation and Design in Computing Systems and Networks (FeBID '09),* 2009.

[10] M.G. Kienzle and K.C. Sevcik, "Survey of Analytic Queueing Network Models of Computer Systems," *ACM SIGMETRICS Performance Evaluation Rev.,* vol. 8, no. 3, pp. 113-129, 1979.

[11] L. Kleinrock, *Queueing Systems. Volume 1: Theory.* Wiley-Interscience, 1975.

[12] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A Scalable Solution to The Multi-Resource QoS Problem," *Proc. IEEE Real-Time Systems Symp. (RTSS '99),* pp. 315-326, 1999.

[13] C. Lefurgy, X. Wang, and M. Ware, "Server-Level Power Control," *Proc. IEEE Int'l Conf. Autonomic Computing (ICAC '07),* p. 4, 2007.

[14] X. Lin, N. Shroff, and R. Srikant, "A Tutorial on Cross-Layer Optimization in Wireless Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, no. 8, pp. 1452-1463, Aug. 2006.

[15] X. Liu, Q. Wang, W. He, M. Caccamo, and L. Sha, "Optimal Real-Time Sampling Rate Assignment for Wireless Sensor Networks," *ACM Trans. Sensor Networks,* vol. 2, no. 2, pp. 263-295, 2006.

[16] V. Pallipadi and A. Starikovskiy, "The Ondemand Governor," *Proc. Linux Symp.,* vol. 2, 2006.

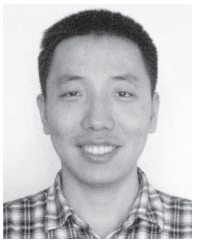
[17] W. Shu, X. Liu, Z. Gu, and S. Gopalakrishnan, "Optimal Sampling Rate Assignment with Dynamic Route Selection for Real-Time Wireless Sensor Networks," *Proc. IEEE Real-Time Systems Symp. (RTSS '08),* 2008.

[18] Y. Wang, X. Wang, M. Chen, and X. Zhu, "Power-Efficient Response Time Guarantees for Virtualized Enterprise Servers," *Proc. IEEE Real-Time Systems Symp. (RTSS '08),* pp. 303-312, 2008.

JIn Heo received the BS degree in computer engineering from Seoul National University, Korea, in 1999, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 2010. He is currently a member of Technical Staff at VMware working on improving network performance of VMware's virtualization products. His research interests include performance and energy management in data centers, cloud computing, and virtualization. He is a student member of the IEEE.

Praveen Jayachandran received the BTech and MTech dual degree in computer science from the Indian Institute of Technology, Chennai, India, in 2005 and the PhD degree from the University of Illinois at Urbana-Champaign in 2010. His research interests include real-time and distributed systems, sensor networks, and wireless networks. He is a student member of the IEEE.

Insik Shin received the BS degree in computer science from Korea University in 1994, the MS degree in computer science from Stanford University in 1998, and the PhD degree in compositional schedulability analysis in real-time embedded systems from the University of Pennsylvania in 2006. He is currently an assistant professor in the Department of Computer Science at KAIST, South Korea, since 2008. He has been a postdoctoral research fellow at Malardalen University, Sweden, and a visiting scholar at the University of Illinois, Urbana-Champaign, until 2008. His research interests include cyber-physical systems and real-time embedded systems. He is a member of the IEEE.



Dong Wang received the BEng degree in communication and information systems from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2004 and the MS degree in electrical and computer engineering from Peking University, Beijing, China, in 2007. He is now a PhD student of Computer Science Department at the University of Illinois at Urbana Champaign. His research interests include cost-aware prediction for data fusion systems and energy-aware real-time scheduling of heterogenous sensor platforms.

Tarek Abdelzaher received the PhD degree from the University of Michigan, Ann Arbor, in 1999, under Professor Kang Shin. From August 1999 to 2005, he was an assistant professor at the University of Virginia. Then, he joined the University of Illinois at Urbana Champaign as an associate professor with tenure. His research interests include operating systems, networking, sensor networks, distributed systems, and embedded real-time systems. Especially, he is interested in developing theory, architectural support, and computing abstractions for predictability in software systems, motivated by the increasing software complexity and the growing sources of nondeterminism. Applications range from sensor networks to large-scale server farms and from avionics to homeland defense. He is a member of the IEEE.

Xue Liu received the BS and MS degrees in mathematics and automatic control from Tsinghua University, China, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 2006. He is the Samuel R. Thompson associate professor in the Department of Computer Science and Engineering at the University of Nebraska-Lincoln. From 2007-2009, he was an assistant professor in the School of Computer Science at the McGill University, Canada. His research interests include real-time and embedded systems, cyber-physical systems, server and data center performance modeling and management, and software reliability. He is a member of the IEEE.