PowerTracer: Tracing Requests in Multi-Tier Services to Reduce Energy Inefficiency

Gang Lu, Jianfeng Zhan, *Member, IEEE*, Haining Wang, *Senior Member, IEEE*, Lin Yuan, Yunwei Gao, Chuliang Weng, *Member, IEEE*, and Yong Qi

Abstract—As energy has become one of the key operating costs in running a data center and power waste commonly exists, it is essential to reduce energy inefficiency inside data centers. In this paper, we develop an innovative framework, called *PowerTracer*, for diagnosing energy inefficiency and saving power. Inside the framework, we first present a resource tracing method based on request tracing in multi-tier services of black boxes. Then, we propose a generalized methodology of applying a request tracing approach for energy inefficiency diagnosis and power saving in multi-tier service systems. With insights into service performance and resource consumption of individual requests, we develop (1) a bottleneck diagnosis tool that pinpoints the root causes of energy inefficiency, and (2) a power saving method that enables dynamic voltage and frequency scaling (DVFS) with online request tracing. We implement a prototype of PowerTracer, and conduct extensive experiments to validate its effectiveness. Our tool analyzes several state-of-the-practice and state-of-the-art DVFS control policies and uncovers existing energy inefficiencies. Meanwhile, the experimental results demonstrate that PowerTracer outperforms its peers in power saving.

Index Terms—Multi-tier web server, request tracing, diagnosing energy inefficiency, saving power

1 Introduction

To date energy cost has been a major part of the total cost of ownership (TCO) of a data center. With the continuing decrease of hardware prices, the portion of energy cost will grow even larger in the near future. Thus, energy efficiency (EE) has become a top priority for data center administrators. However, *energy inefficiency* is not uncommon in data centers. Here we define *energy inefficiency* as *a running state in which a computing system consumes power in a sub-optimal manner*. There are various causes of energy inefficiency, ranging from inappropriate architectures to system misconfigurations. For example, oversized physical infrastructure, defective power management governor, and inappropriate coordination among servers. Uncovering these sources is a critical task for efficiency improvement. Meanwhile, most of services in data centers adopt a multi-tier architecture [1]. Thus this paper focuses on developing innovative techniques to diagnose energy inefficiency mainly caused by inappropriate power management strategies and system misconfigurations, and accordingly improve the energy efficiency in a multi-tier service platform.

Although researchers have proposed a number of request tracing approaches to diagnosing performance problems of multi-tier services of black boxes [2], [3], [4] or white boxes [5], [6], [7], [8], [9], none of them take power management into account. Meanwhile, significant efforts have been made to improve energy efficiency from three different perspectives, (1) DVFS control policies [10], [11], [12], [13], (2) dynamic cluster reconfiguration by consolidating services through request distributions [12], [14], and (3) server consolidation by moving services onto virtual machines [15], [16], [17]. However, diagnosing energy inefficiency is quite different from improving energy efficiency. In other words, the focus of the traditional power saving approaches is not on how to disclose the source of energy inefficiency.

In this paper, we propose a generalized framework of applying a request tracing approach for energy inefficiency diagnosis and power saving in multi-tier service systems. Through kernel instrumentation, the request tracing tool can accurately profile the main activities of requests, in terms of *main causal path patterns* (in short, patterns), which represent *repeatedly executed* causal paths that account for significant fractions out of all requests; and then it can capture the server-side latency, especially the service time[1] of each tier in different patterns. Meanwhile, the request tracing tool enabled with the innovative resource-tracing feature can also collect resource consumption information of individual requests and therefore analyze resource consumption features in various forms, such as features in each tier and that of each pattern. Through power metering and examining the metrics with respect to possible root causes, we can finally decide the existence of energy inefficiency and find the root causes. In addition, the request tracing approach provides a sound basis for energy saving. First, in comparison with a brute-force approach, it costs less time in performance profiling, which is necessary for developing

- *G. Lu, J. Zhan, L. Yuan, and Y. Gao are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. E-mail: {lugang, yuanlin}@ncic.ac.cn, {zhanjianfeng, gaoyunwei}@ict.ac.cn.*
- *H. Wang is with the Department of Computer Science, College of William and Mary, PO Box 8795, Williamsburg, VA. E-mail: hnw@cs.wm.edu.*
- *C. Weng is with Shannon (IT) Lab, Huawei.*
- *Y. Qi is with the Department of Computer Science, Xi'an Jiaotong University, Xi'an, Shangxi, China. E-mail: qiy@mail.xjtu.edu.cn.*

Manuscript received 23 Dec. 2012; revised 2 Jan. 2014; accepted 5 Mar. 2014. Date of publication 3 Apr. 2014; date of current version 8 Apr. 2015. Recommended for acceptance by C.-Z. Xu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2315625

1. Excluding the waiting time.

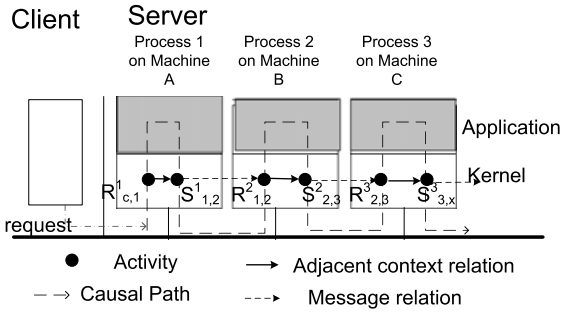


Fig. 1. Activities with causal relations in the kernel [4].

an accurate performance model; second, it reduces the complexity of the controller—a critical component in power saving systems.

We design and implement a prototype of the proposed framework for multi-tier services, called *PowerTracer* [18], which consists of an energy inefficiency diagnosis tool based on online request tracing and a power saving system equipped with accurate DVFS control. To trace the behaviors of serving a request, we adopt a precise request tracing tool (PreciseTracer [4]) and further enhance it with the capability of collecting and analyzing resource consumption. We name the enhanced tool *ResourceTracer*.[2] With insights into service performance and resource consumption of individual requests, all requests, or main causal path patterns, three categories of root causes of energy inefficiency can be pinpointed via individual comparison with the metrics. The diagnosis of a DVFS control policy which leverages the *average* server-side latency as the measured output reveals the different behaviors of causal path patterns. Inspired by the observations, we propose an accurate DVFS control policy to reduce energy inefficiency by adopting an empirical performance model based on the performance data from ResourceTracer. In the control, we use the service time in each tier of each pattern as the feedback input of the controller, which decides the best suitable frequencies and scales each CPU's frequency.

We conduct extensive experiments on a three-tier platform to validate the effectiveness of PowerTracer. For diagnosing energy inefficiency, we use several case studies to demonstrate the efficacy of PowerTracer, including state-of-the-art DVFS control policy—SimpleDVS [12] and state-of-the-practice DVFS control policy—OnDemand [19]. With regard to power saving, we employ three typical workloads—Rice University Bidding System (RUBiS) [20], RUBBoS [21], and TPC-W [22]—to evaluate the performance of PowerTracer, in terms of three metrics: total system power savings, request deadline miss ratio, and average server-side latency. Experimental results demonstrate PowerTracer not only uncovers existing energy inefficiencies but also outperforms its peers [12], [19] in power saving. Experiments also prove its portability and efficiency in virtual environments.

The remainder of this paper is organized as follows. We first present an overview of online request tracing in Section 2 and discuss the concept of energy inefficiency in multi-tier services in Section 3. Then we introduce the architecture of PowerTracer and its basic working mechanisms in Section 4. The implementation and performance evaluation of PowerTracer are respectively detailed in Sections 5 and 6. Finally, we survey related work in Section 7 and draw a conclusion in Section 8.

2. The source code can be downloaded from http://prof.ict.ac.cn/DCBenchmarks.

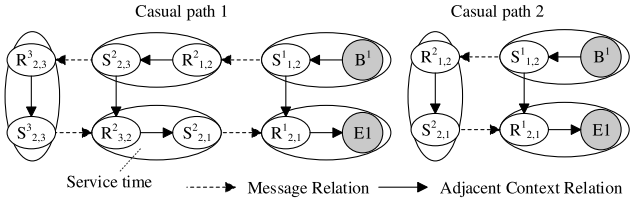


Fig. 2. Two different component activity graphs during serving a request.

2 Background: Overview of Online Request Tracing

As shown in Fig. 1, a request triggers a series of *interaction activities* in the OS kernel or shared libraries, e.g., sending or receiving messages. Those activities happen under specific contexts (processes or kernel threads) of different components. We record an activity of sending a message as $S^i_{i,j}$, which indicates a process i sends a message to a process j. We record an activity of receiving a message as $R^j_{i,j}$, which indicates a process j receives a message from a process i. Activity types of interest include: *BEGIN*, *END*, *SEND*, and *RECEIVE*. The SEND and RECEIVE activities are those of sending and receiving messages. A BEGIN activity marks the start of servicing a new request, while an END activity marks the end of servicing a request.

Causal path. When an individual request is serviced, a series of activities that have causal or happened-before relationship constitute a causal path. For example, in Fig. 1, the activity sequence $\{R^1_{c,1}, S^1_{1,2}, R^2_{1,2}, S^2_{2,3}, R^3_{2,3}, S^3_{3,x}\}$ constitutes a causal path. For each individual request, there exists a causal path exposing all activities with causal relations in the life cycle of serving the request. We propose *a directed acyclic graph* $G(V, E)$ to represent a causal path, where vertices V are activities of components and edges E represent those causal relations between two activities, as shown in Fig. 2. For a multi-tier service, the server-side latency can be defined as the time difference between the time stamp of the BEGIN activity and that of the END activity in its corresponding causal path. The service time of each tier can be defined as the accumulated processing time between SEND activities and RECEIVE activities. For each tier, its role in serving a request can be measured in terms of service time percentage, which is the ratio of service time of the tier to the server-side latency.

Causal path pattern. After correlating those activity logs into causal paths, we can classify the causal paths into different *patterns*. A pattern is a theme of recurring causal paths with common characteristics. Usually, the method of classifying causal paths into patterns is agile; and we can conduct the classification based on path shapes, server-side latencies, resource consumptions, etc. After that, we use *main patterns* to account for the patterns of which the number of causal paths take significant proportions out of all causal paths. For each pattern, we figure out the average server-side latency and the average service time of each tier.

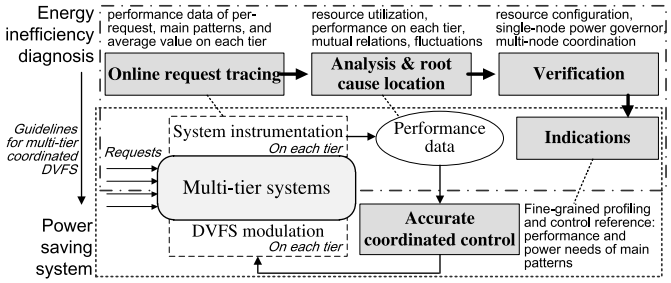


Fig. 3. The overall framework of PowerTracer.

We use a four-tuple *<pattern ID, the average server-side latency of pattern ID, the average service time per tier, current load>* to represent the online performance data of a pattern in a multi-tier service.

Metrics and overhead. An online request tracing system provides fine-grained performance data of individual requests for multi-tier services of black boxes. Useful metrics include throughput, server-side latency, service time of each tier, resource utilization of each tier, and resource consumption of each request, etc. In addition, by observing the number of BEGIN activities, which mark the beginning of serving new requests, we can derive the current load level of the services. The employment of such tools aims at discovering hidden knowledge of requests, optimizing scheduling algorithms, and identifying performance bottleneck. In this paper, we exploit online request tracing for both energy inefficiency diagnosis and DVFS-based power saving. Note that the overhead of request tracing which is generally introduced by instrumentation and simultaneous data operations can potentially reduce the reliability of further analysis. Luckily, the previous work [23] has demonstrated that the overhead of request tracing is minor.

3 Energy Inefficiency of Multi-Tier Services

For data centers, energy is delivered as electricity consumed by servers, air conditioners, lights, and other infrastructures. We selectively focus on servers' energy consumption. For a service node, its energy efficiency can be defined as the ratio of the service work completed under a service-level agreement (SLA) to the corresponding energy consumption. If a multi-tier service is deployed on multiple servers, EE is also equivalent to the ratio of the service throughput to the average power, as depicted in Equation (1). We define *energy inefficiency* as *a running state in which a computing system consumes power in a sub-optimal manner*. Usually, we can conclude the existence of energy inefficiency if the root causes are revealed and further optimization can improve energy efficiency

$$EE = \frac{Service\ completed}{Energy} = \frac{Throughput}{Average\ power}. \quad (1)$$

From Equation (1), we can observe that energy inefficiency is caused by two main factors: poor performance and over-provisioned power. Considering poor performance issues can be resolved through existing performance bottleneck pinpointing tools, our work mainly focuses on the second factor: over-provisioned power. Specifically, performance is obtained through utilizing over-provisioned resources, such as more processors, larger volume and higher bandwidth of memory and storage than that the workloads really need, which usually need more power supply and then consume additional power. The existence of this scenario can be diagnosed through integrated measurement of performance, resource consumption, and power consumption.

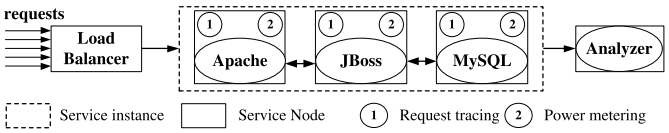


Fig. 4. The architecture of the energy inefficiency diagnosis tool.

4 System Design

The framework of PowerTracer is depicted in Fig. 3. It consists of two main parts: an energy inefficiency diagnosing tool and a power saving system. The diagnostic process generally includes four steps: 1) to run workloads and collect performance data through online request tracing; 2) to analyze the energy efficiency in terms of expected metrics; 3) to classify and verify the root causes; 4) to develop practical guidelines for further improvement. Following the guidelines from a case study of multi-tier coordinated DVFS, we design the power saving system with an offline performance model and fine-grained control based on request tracing. This section presents the detailed descriptions of both parts.

4.1 Diagnosing Energy Inefficiency

4.1.1 Diagnosis Components

Our target service applications run in a multi-tier architecture, and services are replicated or distributed on a cluster of servers [1]. We call an instance of multi-tier service *a service instance*. In our benchmarks, for example, for RUBiS, it includes an Apache server, a JBOSS Server, and a MySQL database. The load balancer like Linux Virtual Server (LVS) [24] is responsible for distributing requests to many service instances for load balancing. As shown in Fig. 4, the energy inefficiency diagnosing tool consists of three modules: ResourceTracer, power metering, and Analyzer. The power metering module, which is responsible for collecting power consumption of each node, is simple and straightforward. We use a latest commercial power analyzer to measure the power consumption. The detailed descriptions of ResourceTracer and Analyzer are given as follows.

ResourceTracer. Most existing request tracing approaches can only figure out latencies of requests. Some tools, such as Magpie [5] developed by Microsoft Research, can profile request resource consumption in multi-tier service systems. Unfortunately, these tools work in white boxes, that is to say, source code of target systems is needed for instrumentation. As an online request tracing tool for multi-tier services of black boxes, PreciseTracer only measures latencies.

To profile request resource consumption of each request, we develop a new tool *ResourceTracer* by

TABLE 1
Average Service Time and the Amount of CPU Jiffies of Two Main Patterns on Each Tier

		Apache	JBoss	MySQL
Average service time(ms)	pattern 1	0.07	1.27	0.98
	pattern 2	1.16	2.74	10.93
	all requests	0.21	1.55	2.40
Amount of CPU jiffies	pattern 1	11855	41838	25234
	pattern 2	8212	346	5737
	all requests	20128	42684	53125

extending PreciseTracer. What makes ResourceTracer different from PreciseTracer is that it records resource consumption information besides latencies, such as CPU time, memory page faults, disk R/Ws, and net I/Oes. These microcosmic statistics help observe the features of resource consumption behaviors of each request. Moreover, ResourceTracer also monitors resource utilization of individual physical nodes, providing macro-scopes in the status of the servers.

Analyzer. This module analyzes request causal paths generated by resource tracing. The core function, *Classifier*, is responsible for classifying a large variety of causal paths into different patterns, and extracting performance data and resource consumption for main patterns according to their fractions.

The stage of classifying causal paths addresses the following two problems: first, it is difficult to leverage each individual causal path from massive request traces as guidelines for diagnosing energy inefficiency or DVFS modulation; second, causal paths are different and the overall statistics of performance information (e.g., the average server-side latency used in [12]) of all paths hide the diversity of patterns. Analyzer characterizes online performance data of different patterns at different tiers, and provides insights into subtle influences induced by different system configurations or power management strategies, disclosing where energy inefficiency hides.

As presented in Section 3, the causes of energy inefficiency for a multi-tier service are complicated by serving of an individual request distributed on multiple servers. From the system view, we pay attention to three categories of root causes: resource overprovision from the operating system, defective power management strategy in a single-tier, and inappropriate power governors with poor or inconsistent multi-node coordination. The purpose of energy inefficiency diagnosis is to uncover the hidden information implying for low utilization of resources, defects of a power management strategy, lack of coordination or inconsistent coordination with cross-node governors.

The general steps of locating the root causes are 1) tracing request, 2) monitoring the performance, resource utilization, and power consumption, and 3) analyzing these data individually with expected metrics. For the root causes, the metrics we can inspect mainly include: available resource capacity, resource utilization, the performance and resource consumptions on each tier, and the change trend of the performance metrics. Besides, we can observe these metrics from different perspectives: performance data of an individual request throughout its causal path, average characteristics of all requests, or

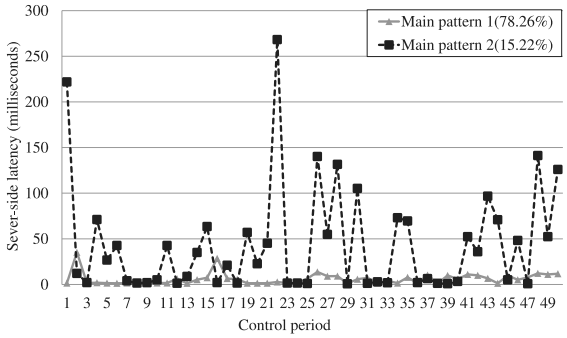


Fig. 5. The server-side latencies of two main patterns. The benchmark is RUBiS's mixed workload with 500 concurrent clients.

main causal path patterns. Through observing these fine-grained and coarse-grained performance data online, we can discover the possible root causes of energy inefficiency. Section 6 presents several case studies of diagnosing energy inefficiency.

4.1.2 An Example of Energy Inefficiency Diagnosis

For a multi-tier service, CPU overprovisioning has motivated much work on developing accurate power governors to mitigate the inefficiency. Here we show how our system can help diagnose energy inefficiency in a multi-tier coordinated DVFS control system.

The simplest power governor is a single-node ad-hoc DVFS controller, which separately modulates a server's CPU frequencies based on its resource utilization or performance. However, through the energy inefficiency diagnosis tool, we uncover that each tier of services has different resource requirements and takes different proportion of the server-side latency. Table 1 shows that the CPU jiffies used in the MySQL tier is the highest and that of the JBoss tier ranks secondly. We also find that the service time of the first tier (Apache server), which is 10^{-2} to 10^{-1} milliseconds on average, contributes little to the server-side latency, which is 10 to 100 milliseconds. The experimental results while causal paths of requests [25]. This observation, as well as the requirement of meeting SLAs, inspires us to implement a multi-tier coordinated DVFS control system, called *PowerTracer_NP*, which step-by-step modulates the CPU frequency of the node whose service time takes the largest proportion of all tiers, and then the CPU frequency of the node with less service time. The average server-side latency which reflects the overall running state of the service and implies for subsequent resource requirements is adopted as the measured output of the feedback control.

However, from our further analysis of causal paths, we observe that requests of different causal path patterns differ in the server-side latency, resource consumption, and service time on each tier, especially on the tiers of JBoss and MySQL. Fig. 5 shows the server-side latencies of the top two patterns under the control of PowerTracer_NP, which take up 78.26 and 15.22 percent of all causal paths, respectively. The requests of pattern 1 are mostly have less SEND-RECEIVE activities as depicted in Fig. 2, while causal paths of requests of pattern 2 have more back-end interactive activities. We can also see that the two patterns

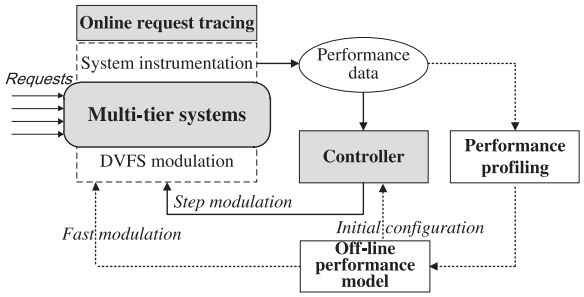


Fig. 6. The architecture of the power saving system.

perform very differently together with the different service time shown in Table 1. In other words, *the DVFS control policy which only considers the average server-side latency ignores the different requirements of all requests on different tiers, which will over-supply resources for serving partial requests on some tiers*. On the other hand, the evaluation in Section 6.3 will also show that the improved controller through choosing individual server-side latencies of the top N causal path patterns as the measured output can save more power than that of PowerTracer_NP, which further verifies the existence of energy inefficiency in PowerTracer_NP as well as the fact that choosing the average server-side latency as the measured output is too coarse-grained.

4.2 Accurate DVFS Control Based Power Saving

As shown in Fig. 6, we apply online request tracing on a power saving system, which consists of four major modules: the online request tracing module, the initial configuration module, the controller module, and the scaler module.

The online request tracing is described in Section 2. The initial configuration module aims to establish an empirical performance model offline under different load levels. Through this model, the fast modulation figures out an optimal configuration for the controller, and the step modulation obtains its reference inputs.

Based on the initial configuration, Controller quickly calculates the optimal setting of DVFS modulation for different load levels. Controller first runs the fast modulation based on the performance model, followed by step modulation loops that include alternate sampling and controlling periods. After online request tracing produces four-tuple performance data for the top N patterns, Controller makes decisions on scaling clock frequencies. For each service instance, Controller only relies on the single-node DVFS modulation as opposed to varying multiple CPU frequencies simultaneously at control periods. Note that Controller chooses the performance of main patterns out of all request paths as the basis of DVFS modulation.

4.2.1 Offline Performance Model

Different from the approach proposed in [26], the offline performance model aims to generate detailed performance profiles of multi-tier services under varying workloads with different DVFS modulations. It can be represented by Equation (2) as follows:

$$\begin{cases} D_{L,1} = \sum_{j=1}^{M} f_{L,1,j}(F_j) + \gamma_{L,1} \\ \quad\vdots \\ D_{L,i} = \sum_{j=1}^{M} f_{L,i,j}(F_j) + \gamma_{L,i} \\ \quad\vdots \\ D_{L,N} = \sum_{j=1}^{M} f_{L,N,j}(F_j) + \gamma_{L,N,} \end{cases} \tag{2}$$

N is the number of main patterns singled out by our request tracing system. M is the number of tiers of a multi-tier service instance. $D_{L,i}(i = 1, 2, \ldots, N)$ is the average server-side latency of pattern i with the current load L, and $\gamma_{L,i}(i = 1, \ldots, N)$ is the network latency of pattern i with the current load L. Function $f_{L,i,j}(F_j)$ represents the average service time of each tier j in pattern i, when the clock frequencies run with $F_j(j = 1, \ldots, M)$ and the current load is L.

For each load level, the function set f is derived as follows: first, through online request tracing, we measure the service time of each tier while traversing all discrete CPU frequency values offered by each server. In a test environment, we vary the DVFS setting of a node, and apply a variety of loads before collecting each tier's performance characteristics, independent of other tiers. As the available CPU frequency levels and service tiers are usually low, the number of offline experiments is very limited. Second, we derive functions $f_{L,i,j}$ between the average service time of each tier in main patterns and the CPU frequencies of nodes by the normal quadratic polynomial fitting tool. Experimental results demonstrated the model has a high fitting coefficient $R^2 > 97$ percent.

This empirical performance model has two main functions for accurate DVFS control. On the one hand, we need to figure out the optimal initial CPU frequencies of the target system in the fast modulation. Given a maximum server-side latency (SLA), the optimal initial CPU frequencies should be as low as possible for power saving. A brute-force approach can be adopted with two heuristic criteria: travel the tiers from the one with most service time to the one with least service time; travel the CPU frequencies from the lowest to the highest. The number of travels is also really limited, since there are usually few service tiers and available CPU frequency levels. On the other hand, the step modulation loop needs the desired server-side latency *threshold zones* for the main patterns to be the reference inputs, which can be directly figured out from the functions $f_{L,i,j}$ given the load level, tier number, and main pattern number.

4.2.2 Controller Design

Similar to the control-theoretic terminology used in [27], we refer to the server cluster and deployed multi-tier services as *the target system*. As the available frequencies are usually very limited, we adopt a simplest feedback controller. Inspired by previous diagnosis of PowerTracer_NP, this controller is designed according to following principles: 1) Monitor the requirement differences among different patterns on different tiers. 2) Control the measured server-side latency according to the empirical performance model for each pattern. 3) Selectively deal with top N main patterns to make the majority of requests meet service-level agreement. In the controller,

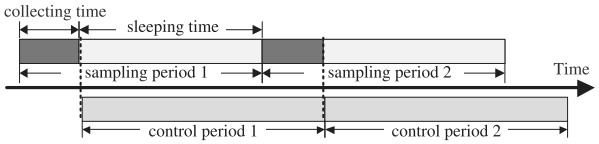


Fig. 7. The relationship of sampling periods and control periods.

the control inputs are the new clock frequency vector. The measured outputs are the server-side latencies of the top N main patterns, which are presented as a vector as well. The controller module makes the measured outputs fall in the latency threshold range by adjusting the values of the control inputs periodically.

The controller module consists of two main procedures: fast modulation and step modulation. In the fast modulation, PowerTracer can decide the current load level and compute the optimal CPU frequency of each server according to above empirical performance model. Once the fast DVFS modulation ends, a step modulation loop will immediately start. In our system, step modulation periods are composed of alternate *sampling* periods and *control* periods. For online request tracing, we use a sampling period which consists of collecting time and sleeping time. For the modulator, we get a control period in which the CPU frequencies are stable. The relationship is depicted in Fig. 7. A control period always has the same length as a sampling period. As the longer the collecting time window, the more logs the analysis algorithm has to process thus the higher overhead [23]. The collecting time window should be tuned to a relatively low value compared to the sleeping time. In the remainder of this paper, "period" represents "control period" if not specifically stated.

We model the performance transition of the multi-tier service as Equation (3). $\overrightarrow{D(t)}$ is the vector that represents the average server-side latencies of each pattern in the tth sampling period. $\overrightarrow{f(t)}$ is the vector that represents the CPU frequency levels of the nodes in the tth period. $\overrightarrow{F(t)}$ is the vector that represents the transition functions of each pattern from states of the tth period to that of the $(t+1)$th period

$$\overrightarrow{D(t+1)} = \overrightarrow{F(\overrightarrow{D(t)}, \overrightarrow{f(t)})}. \tag{3}$$

We design the step modulation procedure as defined in Equation (4).

$$\begin{cases} f_j(t+1) = f_j(t) + 1 & \exists i | (D_i(t) > UP * TH_i) \\ f_j(t+1) = f_j(t) - 1 & \forall i | (D_i(t) < LP * TH_i) \\ f_j(t+1) = f_j(t) & \text{otherwise.} \end{cases} \tag{4}$$

In the equation, $D_i(t)$ denotes the server-side latency of pattern i ($i = 1, \ldots, N$); j is the sequence number of the tier whose service time is the maximum among all patterns in the tth period. We define TH_i ($i = 1, \ldots, N$) as the server-side latency threshold of pattern i. The reference inputs are the desired server-side latency *threshold zones* for the main patterns. LP and UP denote the coefficients of the lower and upper thresholds. $\overrightarrow{f(t)}$ is the vector that represents the CPU frequency levels of the nodes in the tth period. For $f_j(t)$, $+1$ indicates stepping up the CPU frequency by one level, while -1 indicates stepping down the CPU frequency by one level.

In the tth sampling period, the request tracing module obtains a four-tuple performance data of the top N patterns. By comparing the measured server-side latencies of the top N patterns with the upper and lower latency thresholds, the controller module modulates the DVFS setting based on the following procedure.

For pattern i of each service instance, if $D_i(t)$ exceeds its upper threshold UP^*TH_i, the controller module chooses tier j that has the maximum service time to step up its CPU frequency and record the new frequency value into the frequency vector. The frequency values of the other tiers remain intact. For any pattern i ($i = 1, \ldots, N$), if $D_i(t)$ falls below its lower threshold LP^*TH_i, the controller module chooses tier j that has the minimum service time to step down its CPU frequency and record the new frequency into the frequency vector. The frequency values of the other tiers remain intact in the new period.

5 System Implementation

PowerTracer includes six components: ResourceTracer, Power metering, Analyzer, Power Model, Controller, and Scaler. The energy inefficiency diagnosis tool is composed of the former three components. The power saving system consists of ResouceTracer and the latter three components.

ResourceTracer records all service logs and periodically reports the resource status. Based on the resource logs, Analyzer unveils the top N patterns, which is responsible for diagnosing energy inefficiency. As for power saving, Power Model analyzes the performance data and then generates the power model into the file *Pre_model*. For different load levels, Controller runs the fast modulation based on *Pre_model*. After that, Controller executes step modulation loops, periodically calling Scaler to adjust CPU frequencies. Scaler sets the current frequency scaling governor of the Linux kernel to be *userspace* and records the new frequency into the *scaling_setspeed* file. The rest of the section details the implementation of ResourceTracer and the deployment of PowerTracer in virtualized environments.

5.1 ResouceTracer

ResourceTracer instruments the OS kernel activities using SystemTap [28] and utilizes the same correlation algorithm as PreciseTracer. It records resource consumption information, such as CPU time, memory page faults, disk R/Ws, and net I/Os. These resource profiling jobs are fulfilled through SystemTap probes, like ioblock, netdev, vm.pagefault probes. The probes are instrumented into kernel events we concern. ResourceTracer profiles resource consumption of certain processes or threads, ignoring the information that is unrelated to serving requests. Note that resource tracing for individual requests must satisfy the assumption that a single execution entity (a process or a kernel thread) of each component can only serve one request in a certain period. For serving each individual request, the execution entities of the components cooperate through sending or receiving messages via a reliable communication protocol, like TCP.

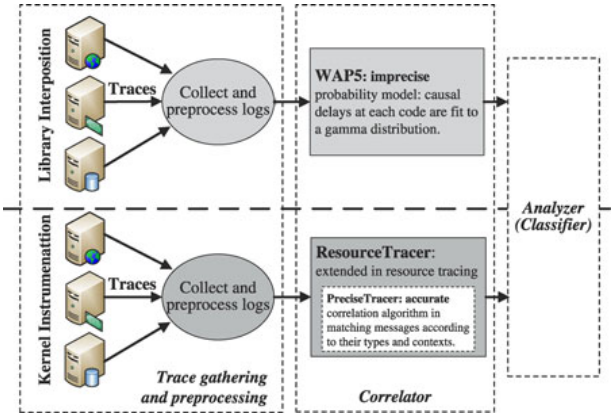


Fig. 8. Two tracer architectures.

In addition, ResourceTracer also monitors resource utilizations of each physical node by a coarse-grain resource accounting, like */proc* in Linux systems.

As shown in Fig. 8, we implement two instrumentation mechanisms for online request tracing: OS kernel instrumentation based on SystemTap and library interposition. We employ WAP5, which uses library interposition and accepts imprecision of a probabilistic correlation, as the counterpart of ResouceTracer in correlating interaction activities into causal paths.

5.2 PowerTracer in Virtualized Environments

Nathuji and Schwan [29] emphasized the limitation of many-core power management in virtualized systems because there is only one voltage rail into the CPU package. But for current quard-core or many-core processors, the cores inside a package begin to use individual Vcores (CPU Core Voltage) and Phase Locked Loop (PLLs). So this restriction can be mitigated with the change of CPU voltage supply. On the other hand, in virtualized environments, a virtual CPU (VCPU) is actually a task running on physical CPUs (PCPU). We usually tend to pin a VCPU to some certain PCPUs to prevent performance degradation caused by VCPU migration. In this case, scaling a PCPU's frequency via administration interfaces in domain 0 produces the ideal effects on a VCPU's running speed, as the PCPU is only running a single VCPU task. Taking Xen as an example, users in domain 0 can scale the frequencies of PCPUs through Xen power management interface (*xenpm* when using hypervisor based *cpufreq* driver). Therefore, we can apply the approaches proposed above by dynamically scaling the frequencies of the PCPUs to improve energy efficiency on the condition that multi-tier services are respectively deployed on different VMs, whose VCPUs are respectively bound to different PCPUs.

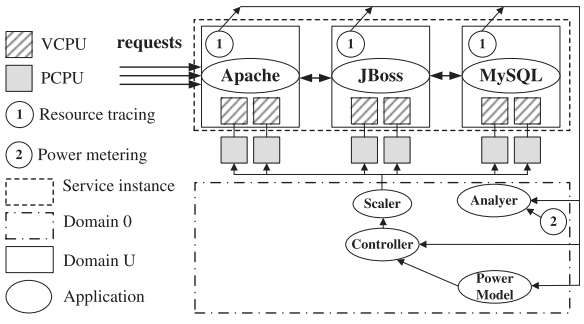


Fig. 9. The deployment of PowerTracer in virtualized environments.

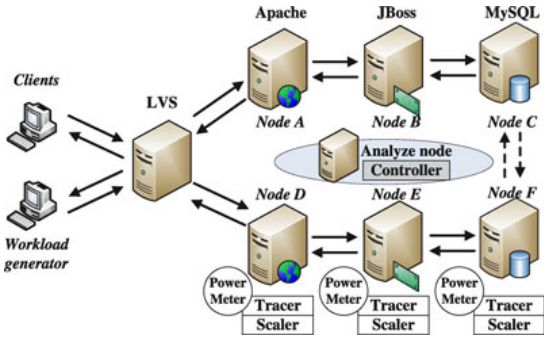


Fig. 10. The deployment diagram of three-tier platform.

The deployment of PowerTracer in virtualized environments is shown in Fig. 9. Unlike in physical environments, the Analyzer module, the Controller module, and the Scaler module are deployed in domain 0. The scaler module directly steps up or down frequencies of specific PCPUs, which is different from scaling PCPUs in physical nodes. Each VM has a request tracing module and a ResourceTracer module. Other modules are nearly the same as those in physical nodes.

6 Evaluation

We use three different kinds of three-tier web applications RUBiS [20], RUBBoS [21] and TPC-W [22] to evaluate the efficacy of our approach for both diagnosing energy inefficiency and saving power. RUBiS is a three-tier auction site prototype modeled after eBay.com, developed by Rice University. RUBBoS is a bulletin board benchmark modeled after an online news forum like Slashdot, also developed by Rice University. TPC-W is a transactional web e-commerce benchmark that simulates the activities of a business oriented transactional web server.

6.1 Experimental Setup

Our testbed is a heterogeneous ten-node platform composed of Linux-OS blade servers. In the following experiments, we deploy one or two service instances, each of which includes an Apache server, a JBOSS Server, and a MySQL database server on the testbed as shown in Fig. 10. Table 2 lists the available cores and frequencies. Tracer, Scaler, and Power Meter are deployed on all the three server tiers, but Controller and Analyzer are only deployed on the web server tier. Note that the service delay at the web server tier imposes the least impact on request performance. We adopt ResouceTracer as the default tracer.

TABLE 2
Experiment Deployment

Nodes	Total cores	Available frequencies(GHz)
A & B	2	1.0, 1.8, 2.0, 2.2
C & D	8	0.8, 1.1, 1.6, 2.3
E & F	16	1.6, 1.7, 1.8, 2.0, 2.1, 2.2, 2.4

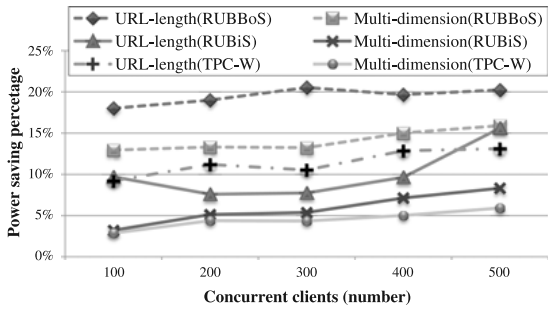


Fig. 11. The comparison of two classification methods.

6.1.1 Path Classification

PowerTracer provides two different ways to classify causal paths into different patterns. One uses a k-means clustering method [30] to classify causal paths based on the size of the first message sent by a client. Note that the size of the first packet, which includes different URL segments for different requests, has a linear relationship with the length of request URLs, and the length of the URL likely reflects the depth of the service and the complexity of the work to be performed. The other employs a multi-dimension k-means clustering algorithm. The distance metric is a function of the string-edit-distance of two paths, which denotes the differences between the graphs of two causal paths, and the resource consumption deltas of two paths, which include deltas of the CPU jiffies, memory page faults, disk R/W sizes, and net I/O sizes. We choose a commonly used clustering algorithm k-means, instead of other clustering algorithms like k-medoids, because it has a relatively lower time complexity, and it is more widely applied and easily implemented. The shortcoming of the super-polynomial worst case running time of k-means can be ignored as the input size is limited.

We first evaluate the two methods using the RUBiS read_only, RUBBoS, and TPC-W workloads with a group of clients, whose size varies from 100 to 500. The up ramp time, runtime session and down ramp time are set to 5,300 and 5 seconds, respectively. Fig. 11 shows that the first method saves more power than the other in all the workloads, implying that URL length is positively correlated with the service depth and the complexity of services. The multi-dimension clustering falls behind; a possible reason is that it is hard to decide the optimizing parameters for multi-dimension clustering and the number of patterns is usually much more than the former. Therefore, in the following experiments, we use the first method to classify paths into causal patterns.

6.2 Case Studies of Diagnosing Energy Inefficiency

We conduct two case studies of diagnosing energy inefficiency with PowerTracer. The experiments are run on the mixed workload of RUBiS with 500 clients. The three-tier services are deployed on Nodes A, B, and C in Fig. 10. Each workload includes three stages, of which the up ramp time, runtime session, and down ramp time are set to 5,300, and 5 seconds, respectively. The lengths of the control period and sampling period are set to 6 seconds. The collecting time during a sampling period is only 1 second.

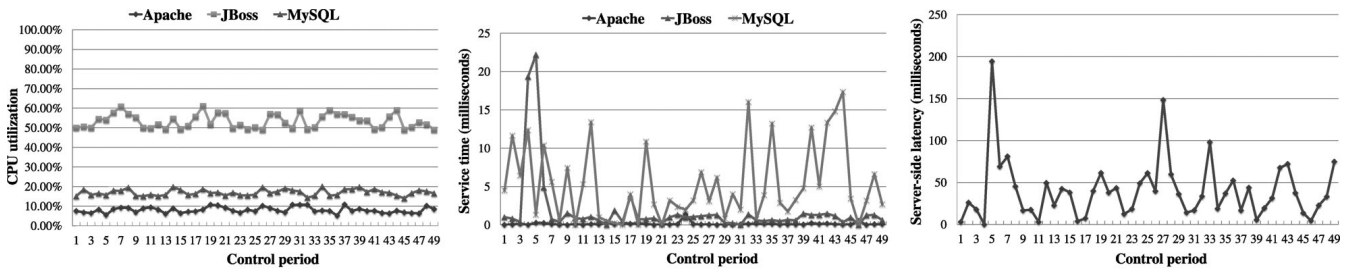
6.2.1 SimpleDVS

The SimpleDVS DVFS control algorithm presented by Horvath et al., [12] takes CPU utilization as the indicator in determining which server's clock frequency should be scaled, and implements a feedback controller based on the DVS mechanism. The scaling policy of SimpleDVS is that stepping down the clock frequency of the server with the minimum CPU utilization if the latency is below a lower threshold, and stepping up the clock frequency of the server with the maximum CPU utilization if the latency is above an upper threshold. For SimpleDVS, we set the latency threshold, UP (the upper latency threshold coefficient), and LP (the lower latency threshold coefficient) to 16.35 milliseconds, 1.2 and 0.8, respectively.

Through online resource tracing, we obtain performance results, which are shown in Fig. 12. We can see that at most of the time the CPU utilization of each tier has an approximate relationship of *JBoss* > *MySQL* > *Apache* and the service time of each tier has the relationship of *MySQL* > *JBoss* > *Apache*, indicating that the service time of each tier is not consistent with its CPU utilization. Thus, it is not optimal to use SimpleDVS by scaling clock frequency of the server which has the maximum or minimum CPU utilization to affect the server-side latency. Optimizing multi-node coordination of power governor is a feasible solution to mitigating energy inefficiency.

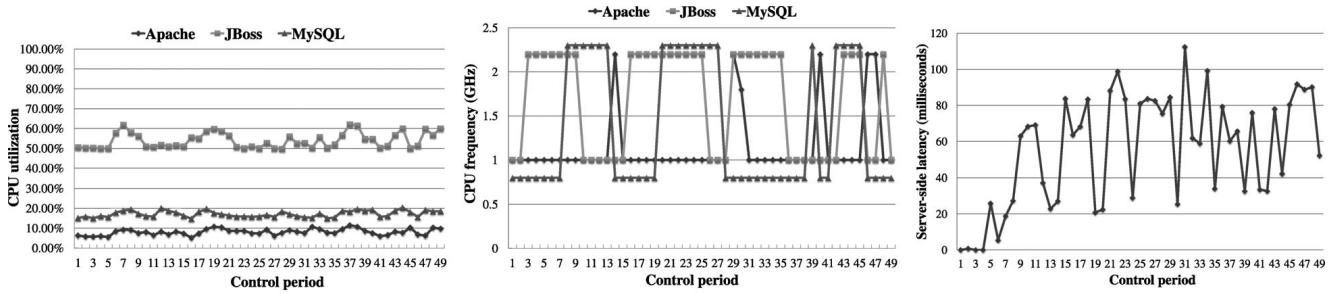
6.2.2 Ondemand Governor

The Ondemand governor, the most effective power management policy offered by Linux kernel, can vary CPU frequency based on CPU utilization. Fig. 13 shows that the CPU utilizations of each tier, the CPU frequencies of each node, and the server-side latencies are dynamically changed in different control periods, respectively, due to the use of Ondemand governor.



(a) CPU utilizations of each tier (b) The service time of each tier (c) Server-side latencies

Fig. 12. Experiment results of the target system equipped with simpleDVS.



(a) CPU utilizations of each tier (b) CPU frequencies of each tier (c) Server-side latencies

Fig. 13. Experiment results of the target system equipped with Ondemand.

Compared with that of SimpleDVS, Fig. 13c shows that the server-side latency of the service equipped with Ondemand is higher. At most of time, the server-side latency of the service equipped with Ondemand is higher than 50 milliseconds, while the server-side latency of the service equipped with SimpleDVS is lower than 50 milliseconds. This observation indicates that without coordination of DVFS control actions on each node, Ondemand performs poorly in controlling the server-side latencies. Meanwhile, the CPU utilization of each tier with Ondemand is higher than that of SimpleDVS at most of time. The CPU frequencies fluctuate significantly, which overreacts to the load fluctuation on each tier. Also the throughput decreases 19 percent in comparison to that of SimpleDVS. We can conclude the poor coordination of multi-node frequency scaling aggravates the performance degradation and power consumption. Therefore, the root cause of energy inefficiency in Ondemand for multi-tier services can be ascribed to the lack of the coordination of DVFS control actions on each node.

6.3 Effectiveness of Accurate DVFS Control

We conduct five groups of experiments to evaluate the effectiveness of accurate DVFS control of PowerTracer: static workload experiment, dynamic workload experiment, multi-service-instance experiment, application in virtual environments, and WAP5 versus ResourceTracer experiment.

We set the clock frequency of all servers to the maximum as *the baseline*. And three metrics are used to evaluate our system: the total system power savings compared to the baseline, the request deadline miss ratio, and the average server-side latency. Note that the power consumption under the baseline is not fixed for different load levels, and higher load will lead to higher power consumption even with the same clock frequency. We assume the server-side latency under the baseline is $\overrightarrow{SL}$, which is a vector representing the server-side latencies of the chosen main patterns. For the request deadline miss ratio, we compute the percentage of the requests whose server-side latencies exceed a predefined deadline under the SLA constraints. We compare the results of PowerTracer with those of the three other algorithms—SimpleDVS, Ondemand, and PowerTracer_NP. In the figures, we use *Performance* to denote the Linux Performance governor which forces the CPU to use the highest possible clock frequency. The energy consumption under the Performance governor is adopted as the baseline when we calculate the power saving ratio.

In the following experiments, the sampling period and the control period are set to 6 seconds. The upper latency threshold coefficient and the lower latency threshold coefficient are set to 1.2 and 0.8, respectively.

6.3.1 Static Workload

Whereas we have performed experiments on TPC-W, RUBBoS and the two workloads of RUBiS—read_only workload and read_write mixed workload, due to the page limit, we only present the experimental results of RUBBoS benchmark. The experimental results of other workloads are consistent with those of RUBBoS. We also investigate the effects of the server-side latency threshold and the number of main patterns upon the power consumption and other performance metrics.

In the experiments, we only deploy a service instance on a three-node platform composed of Nodes A, B and C, as shown in Fig. 10. For the RUBBoS workload, the number of clients ranges from 100 to 500. The up ramp time, runtime session, and down ramp time are set to 5 seconds, 300 seconds and 5 seconds, respectively. The deadline is set to 0.5 second. For PowerTracer, we set the server-side latency threshold to $2 \times \overrightarrow{SL}$, where $\overrightarrow{SL}$ is (6.31, 74.59, 112.50, 38.22, 27.83, 150.20, 72.19) milliseconds, and set the number of main patterns as 3 (the total number of patterns is 7).

Fig. 14 presents the total system power savings compared to the baseline, the average server-side latencies, and the request deadline miss ratios of the four algorithms, respectively, when the number of clients varies from 100 to 500. We find that PowerTracer gains the highest power saving, and can reduce the power consumption by 20.83 percent when the number of clients is 500. The SimpleDVS and Ondemand governor both have high request deadline miss ratios, and the SimpleDVS has the lowest power reduction. We also find that the DVFS control increases the server-side latency in most workloads.

6.3.2 Dynamic Workload

Based on RUBiS read_write mixed workload, we emulate a dynamic workload by modulating the number of clients as array [200, 400, 600, 800, 600, 400, 200]. The number of clients changes per each minute. In this experiment, we deploy a service instance on a three-node platform composed of Nodes A, B, and C, as shown in Fig. 10.

Fig. 15 presents the total system power savings compared to the baseline, the average server-side latencies, and the request deadline miss ratios of the four algorithms,

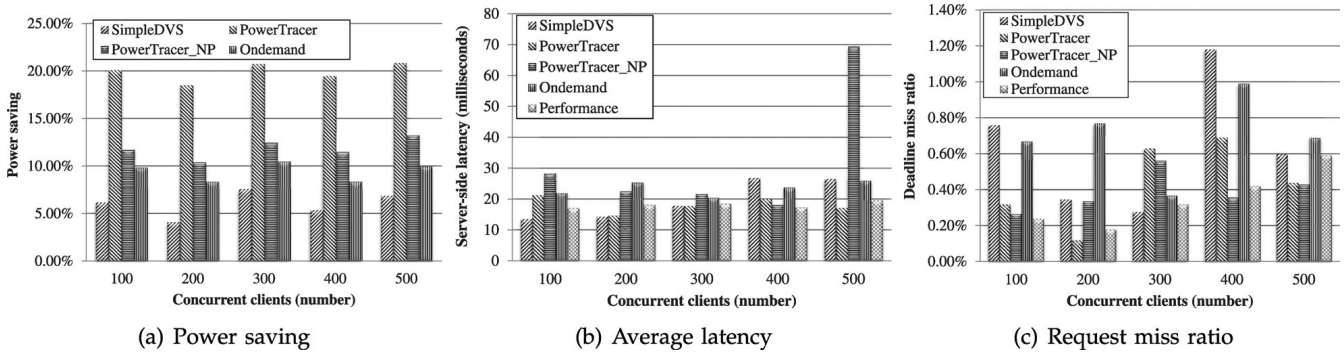


Fig. 14. A comparison of four power management algorithms in static workload.

respectively. In these experiments, we set the number of the main patterns to 3 and the latency threshold to $5 \times \overrightarrow{SL}$. The deadline is set to 0.2 second. For different workload level, $\overrightarrow{SL}$ is different. We can see that PowerTracer and PowerTracer_NP saves much more power than SimpleDVS and Ondemand. Though PowerTracer_NP gains 1.1 percent higher power saving than PowerTracer, we can see from Fig. 15c that PowerTracer_NP also has higher request deadline miss ratio than PowerTracer.

6.3.3 Multi-Service-Instance Experiment

To demonstrate that PowerTracer is applicable on multi-service instances, we deploy two service instances on the nine-node platform. The synchronization of databases is based on master-master replication, which guarantees database consistency. We use RUBiS read_only workload and set the number of clients to 500. The parameters are set as the same as those in Section 6.3.1. The deadline is set to 0.5 second. For PowerTracer, we set the server-side latency threshold to $3 \times \overrightarrow{SL}$. The number of main patterns is set to 5.

Fig. 17 shows that PowerTracer reduces power consumption by 13.98 percent, the highest among all the four algorithms. Meanwhile, PowerTracer achieves the very low request deadline miss ratio, which is below 0.80 percent.

Note that PowerTracer is based on PreciseTracer [23], in which we have demonstrated how to improve the scalability of the online tracing approach through two mechanisms: tracing on demand and sampling. In addition, our experimental results show that PreciseTracer achieves fast responsiveness, and imposes negligible impacts on the throughput and the average response time of services, like RUBiS. Therefore, PowerTracer is promising in scalability.

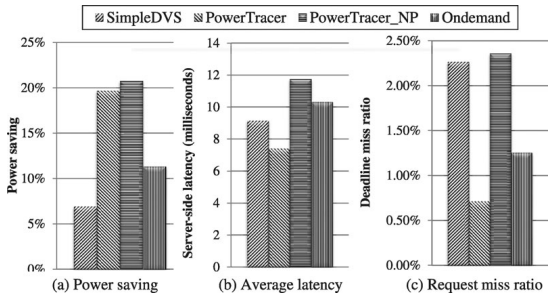


Fig. 15. A comparison of four power management algorithms in dynamic workload.

6.3.4 PowerTracer in Virtualized Environments

We apply PowerTracer on a virtualized platform, of which the physical node has eight PCPUs with available frequencies of 0.8, 1.1, 1.6 and 2.3 GHz (Quad-Core AMD Opteron (tm) Processor 2376) and 8 GB memory. The virtual machine (VM) hypervisor is Xen 3.1.4. Upon domain 0, there are three VMs where the web server, the application server, and the database server are deployed. Each VM is bound to two VCPUs and has maximum 2,048 MB memory capacity. The deployment is almost the same as shown in Fig. 9. We use RUBBoS read_write workload with the number of clients varying from 100 to 500. For all the algorithms, the frequency scaling of PCPUs are conducted through *xenpm* interfaces.

In Fig. 16, the experiment results show that PowerTracer can reduce power consumption by nearly 7 percent on average, which is almost twice as the Ondemand governor does. The reason why the power-saving percentage is not as high as those in physical target systems probably lies in the virtual mechanism of the three servers. Sharing resources among all domains improves resource utilization, but the power consumption of physical components does not change much. We can also observe that the server-side latencies and request miss ratios are higher than those in physical environments. The main reason is that although the VCPUs are bound to PCPUs, the performance interference among virtual machines still exists in other components, especially disks. A feasible solution is to improve performance isolation or use a full virtualization hypervisor instead. Jung et al. [31] proposed a self-aware search algorithm and addressed the tradeoff among power consumption, performance, and transient costs to improve energy efficiency and resource utilization in virtual environments. Integrating this method with our approaches could further improve energy efficiency.

6.3.5 WAP5 versus ResourceTracer

We further compare two tracers: WAP5 and ResourceTracer. Performance data are gained from each tracer to perform the DVFS controller. We use a three-tier platform composed of Nodes D, E, and F, as shown in Fig. 10. We run the same workload as used in Section 6.3.1. For PowerTracer, the server-side latency threshold and the number of main patterns are also the same. From the comparison listed in Table 3, we can see PowerTracer equipped with ResourceTracer saves power by 15.01 percent, which is 1.51 times of that equipped with WAP5. Moreover, it has much lower request deadline miss ratio and server-side latency. Since

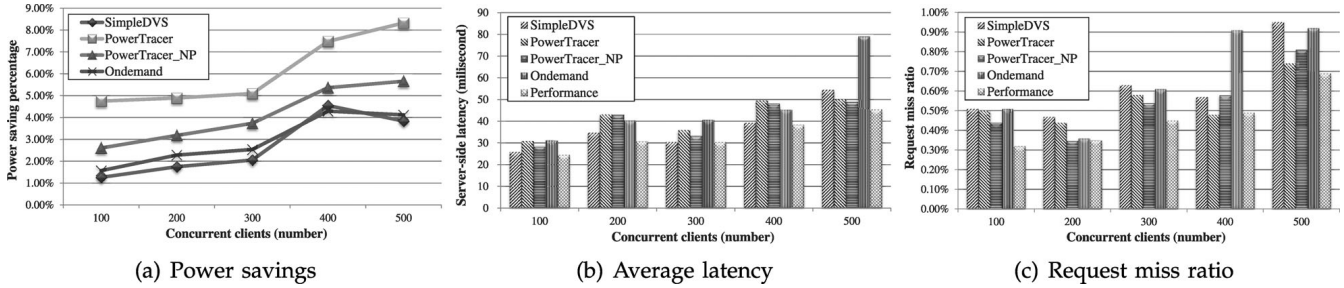


Fig. 16. A comparison of four power management algorithms in virtual environments.

WAP5 accepts imprecision of probabilistic correlations and uses the imprecise performance information, PowerTracer equipped with WAP5 suffers lower power efficiency.

6.4 Discussion

For three different workloads of RUBiS, RUBBoS and TPC-W, our experimental results show that PowerTracer or PowerTracer_NP outperforms its peers [12], [19], indicating that request tracing can improve the accuracy of DVFS control. At most of time, PowerTracer outperforms PowerTracer_NP. This implies that monitoring the main patterns of performance data, rather than the average data, can also improve the accuracy of DVFS control. However, the optimal number of main patterns varies with different workloads. We also observe that setting a higher latency threshold under a certain limit in PowerTracer improves power savings, the details of which can be found in [25].

Responsiveness and CPU frequency fluctuation. Responsiveness is not the focus of this paper. We notice that the responsiveness of this framework to the workload fluctuation is limited by the control period of the step modulation. Sharp fluctuations may be responded more than one control period if several step modulations are needed. Gong and Xu [32] proposed a responsive gray-box controller, which is helpful for our further work, but they focused on responsiveness and performance degradation rather than power consumption. Besides, the frequency fluctuation also becomes an issue when it comes to fine-grained DVFS. As described in [33], the ad-hoc controller oscillates between two neighboring performance states whose power consumption levels just bound the set point power. In this paper, the DVFS controller is not stimulated by a set point power actuating the modulation, but guided by the performance model that tries to meet the service-level agreement of main patterns rather than an overall power cap.

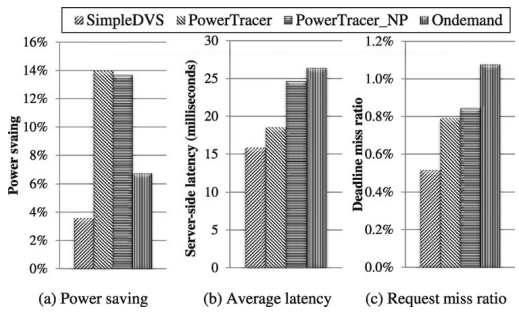


Fig. 17. A comparison of four power management algorithms with multiple service instances.

Potential for power saving. So far, in most of commercial servers, CPU is the only energy proportional component, and our work is also confined by this limitation. Barroso and Holzle [34] showed that four components, including CPU, DRAM, disk, and network switches, are the main sources of power consumption in a data center. Therefore, we believe that our accurate DVFS control will play a more important role in power saving when the concept of DVFS is extended to the other system components. [35] unreferenced paper

Potential extension. In this paper, we only provide a prototype implementation of leveraging online request tracing for accurate DVFS control. However, our approach can be applied for another power saving approach—dynamic cluster reconfiguration [12], [14], [36], since it will also benefit from accurate performance behavior monitoring. Besides, the recent shift to multi-core or many-core processors makes it applicable to run many multi-tier services in a single server, bringing new challenges for operating the cores, memory, internal network. Though some work already exists in this field [37], [38], [39], we believe it is still worth implementing PowerTracer in multi-core or many-core environment and adopt either online request tracing or DVFS control to improve the servers' overall energy efficiency. Moreover, with the recent release of big data benchmarks [40], [41], we will evaluate our systems in the context of big data.

7 Related Work

We summarize the related work in the following four perspectives.

DVFS in server clusters. Horvath et al. [12] proposed a coordinated distributed DVS policy based on feedback controller for three-tier web server systems. However, their work fails to devise accurate DVFS control algorithms for reasons explained in Section 6.2. They also proposed a service prioritization scheme for multi-tier web server clusters [11], where clients are assigned different priorities based on performance requirements. Chen et al. [13] developed a simple metric called *frequency gradient* to predict the impact of processor frequency changes on the end-to-end response times of multi-tier applications.

TABLE 3
ResourceTracer versus WAP5

	Power saving	*Average latency (milliseconds)*	*Request miss ratio*
ResourceTracer	15.01%	4.30	0.02%
WAP5	9.93%	10.52	1.08%

Virtual machine based server consolidation. Dhiman et al. [42] revealed that co-scheduling VMs with heterogeneous characteristics on the same physical node is beneficial from both energy efficiency and performance. Nathuji and Schwan [29] proposed the VirtualPower approach which utilizes VPM states to map VM-level updates made to soft power states to actual changes in the states and in the allocation of underlying virtualized hardware. Wang et al. [43] proposed *virtual batching*, a novel request batching solution for virtualized servers with primarily light workloads. Wang and Wang [15] proposed Co-Con, a cluster-level control architecture that coordinates individual power and performance control loops for virtualized server clusters. Gong and Xu propsed vPnP [44], a robust feedback control-based coordination system which directs the resource allocation (VCPU time slice) amongst virtual machines. Padala et al. [45] developed an adaptive resource control system that dynamically adjusts the resource shares to individual tiers in order to meet application-level QoS goals. Malkowski et al. [46] proposed a multi-model controller which deduces adaptation decisions from several models. Using measured performance data from previous runs of the application, the controller also supports an empirical model that provides assured adaptation decisions.

Energy efficiency of specific systems. Leverich and Kozyrakis[47] modified Hadoop to allow scale-down of operational clusters (i.e., operating at reduced capacity). Tsirogiannis et al. [48] experimented with several classes of database systems and storage managers, varying parameters that span from different query plans to compression algorithms and from physical layout to CPU frequency and operating system scheduling, and they found out that within a single node intended for use in scale-out (shared-nothing) architectures, the most energy-efficient configuration is typically the one with the highest performance.

Request tracing. There are two types of request tracing: black-box [2], [3], [6], [49], [50] or white-box [7], [8], [9] approaches, on a basis of which we develop our online request tracing systems.

8 Conclusion

In this paper, we proposed a generalized framework of applying a request tracing approach for energy inefficiency diagnosis and power saving in multi-tier service systems. Our resource tracing tool can characterize main causal path patterns in serving different requests and record server-side latency, especially the service time of each tier in different patterns, and resource consumption information, providing guidelines for diagnosing energy inefficiency. Based on the online tracing approach, we also presented a hybrid DVFS control algorithm that combines an empirical performance model for fast modulation at different load levels and a simpler controller for adaptation. To validate the efficacy of the proposed framework called PowerTracer, we developed a prototype and conducted extensive experiments on a three-tier platform. Experimental results demonstrate PowerTracer can uncovers existing energy inefficiencies and outperforms its peers [12], [19] in power saving, including in virtualized environments. Moreover, PowerTracer equipped with ResourceTracer outperforms that equipped with WAP5, indicating that higher accuracy of request tracing leads to more power saving.

Acknowledgments

The authors very grateful to anonymous reviewers. This work was supported by the NSFC project (Grant No. 60933003), the Chinese 973 project (Grant No. 2011CB302500), and HuaWei funding. Jianfeng Zhan is the corresponding author.

References

[1] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, pp. 291–302, Jun. 2005.

[2] B. C. Tak, C. Tang, C. Zhang, S. Govindan, B. Urgaonkar, and R. N. Chang, "vPath: Precise discovery of request processing paths from black-box observations of thread and network activities," in *USENIX*, 2009, pp. 19–19.

[3] P. Reynolds, J. L. Wiener, J. C. Mogul, M. K. Aguilera, and A. Vahdat, "WAP5: Black-box performance debugging for wide-area systems," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 347–356.

[4] Z. Zhang, J. Zhan, Y. Li, L. Wang, D. Meng, and B. Sang, "Precise request tracing and performance debugging for multi-tier services of black boxes," in *Proc. Dependable Syst. Netw.*, 2009, pp. 337–346.

[5] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier, "Using magpie for request extraction and workload modelling," in *Proc. 6th USENIX Symp. Operating Syst. Des. Implementation*, 2004, pp. 259–272.

[6] S. Agarwala, F. Alegre, K. Schwan, and J. Mehalingham, "E2EProf: Automated end-to-end performance management for enterprise systems," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2007, pp. 749–758.

[7] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem determination in large, dynamic internet services," in *Proc. Int. Conf. Dependable Syst. Netw.*, 2002, pp. 595–604.

[8] B. H. Sigelman, L. Andr, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a large-scale distributed systems tracing infrastructure," *Google Technical Report dapper-2010-1*, Apr. 2010.

[9] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica, "X-trace: A pervasive network tracing framework," in *Proc. 4th USENIX Symp. Netw. Syst. Des. Implementation*, 2007, pp. 271–284.

[10] T. Horvath and K. Skadron, "Multi-mode energy management for multi-tier server clusters," in *Proc. 17th Int. Conf. Parallel Archit. Compilation Techn.*, 2008, pp. 270–279.

[11] T. Horvath, K. Skadron, and T. Abdelzaher, "Enhancing energy efficiency in multi-tier web server clusters via prioritization," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2007.

[12] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu, "Dynamic voltage scaling in multitier web servers with end-to-end delay control," *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 444–458, Apr. 2007.

[13] S. Chen, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and W. H. Sanders, "Blackbox prediction of the impact of DVFS on end-to-end performance of multitier systems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, pp. 59–63, Mar. 2010.

[14] K. Rajamani and C. Lefurgy, "On evaluating request-distribution schemes for saving energy in server clusters," in *ISPASS*, 2003, pp. 111–122.

[15] X. Wang and Y. Wang, "Coordinating power control and performance management for virtualized server clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 2, pp. 245–259, Feb. 2011.

[16] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," *ACM SIGPLAN Notices*, vol. 44, pp. 205–216, Mar. 2009.

[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 *Proc. 4th ACM Eur. Conf. Comput. Syst.*, 2009, pp. 13–26.

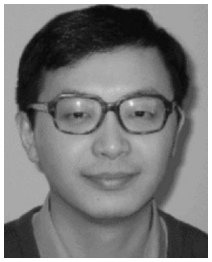
[18] G. Lu, J. Zhan, H. Wang, Y. Lin, and C. Chuliang, "PowerTracer: Tracing requests in multi-tier services to diagnose energy inefficiency," in *ICAC*, 2012, pp. 97–102.

[19] V. Pallipadi and A. Starikovskiy, "The ondemand governor: Past, present and future," in *OLS*, 2006, pp. 223–238.

[20] (2009) RUBiS: Rice university bidding system. http://rubis. ow2.org/
[21] (2005) RUBBoS: Bulletin board benchmark. http://jmob.ow2.org/rubbos.html
[22] (2013) TPC-W: a transactional web e-commerce benchmark. http://www.tpc.org/tpcw/
[23] B. Sang, J. Zhan, G. Lu, H. Wang, D. Xu, L. Wang, and Z. Zhang, "Precise, scalable, and online request tracing for multi-tier services of black boxes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 1159–1167, Jun. 2012.
[24] (2012) Linux virtual server. http://www.linux-vs.org/
[25] Y. Lin, G. Lu, J. Zhan, H. Wang, and L. Wang, "PowerTracer: Tracing requests in multi-tier services to diagnose energy inefficiency," ICT Tech. Rep., (2012) [Online]. Available: http://prof.ncic.ac.cn/jfzhan/
[26] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai, "SLA decomposition: Translating service level objectives to system level thresholds," in *Proc. 4th Int. Conf. Autonomic Comput.*, 2007, p. 3.
[27] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin, "What does control theory bring to systems research?" *SIGOPS Oper. Syst. Rev.*, vol. 43, pp. 62–69, Jan. 2009.
[28] (2013) SystemTap: An instrumentation tool which simplifies the gathering of information about the running linux system. [Online]. Available: http://sourceware.org/systemtap
[29] R. Nathuji and K. Schwan, "Virtualpower: Coordinated power management in virtualized enterprise systems," in *Proc. 21st ACM SIGOPS Symp. Operating Syst. Principles*, 2007, pp. 265–278.
[30] J. Hartigan and M. Wong, "A k-means clustering algorithm: Algorithm as 1366," *Appl. Stat.*, vol. 28, pp. 126–130, 1979.
[31] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, 2010, pp. 62–73.
[32] J. Gong and C.-Z. Xu, "A gray-box feedback control approach for system-level peak power management," in *Proc. 39th Int. Conf. Parallel Process.*, 2010, pp. 555–564.
[33] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," in *ICAC*, 2007, pp. 4–4.
[34] L. Barroso and U. Holzle, "The case for energy-proportional computing," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 33–37, Dec. 2007.
[35] J. Zhan, L. Wang, X. Li, W. Shi, C. Weng, W. Zhang, and X. Zang, "Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2155–2168, Nov. 2013.
[36] J. Yang, K. Zeng, H. Hu, and H. Xi, "Dynamic cluster reconfiguration for energy conservation in computation intensive service," *IEEE Trans. Comput.*, vol. 61, no. 10, pp. 1401–1416, Oct. 2012.
[37] V. Hanumaiah and S. Vrudhula, "Energy-efficient operation of multi-core processors by DVFS, task migration and active cooling," *IEEE Trans. Comput.*, vol. 63, no. 2, pp. 349–360, Feb. 2014.
[38] J.-W. Jang, M. Jeon, H.-S. Kim, H. Jo, J.-S. Kim, and S. Maeng, "Energy reduction in consolidated servers through memory-aware virtual machine scheduling," *IEEE Trans. Comput.*, vol. 60, no. 4, pp. 552–564, Apr. 2011.
[39] S. Liu and J.-L. Gaudiot, "Potential impact of value prediction on communication in many-core architectures," *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 759–769, Jun. 2009.
[40] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zhen, G. Lu, K. Zhan, and B. Qiu, "BigDataBench: A big data benchmark suite from internet services," in *Proc. High Perform. Comput. Archit.*, 2014, pp. 488–499.
[41] Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, L. Wang, and J. Zhan, "BDGS: A scalable big data generator suite in big data benchmarking," in *CoRR*, 2014.
[42] G. Dhiman, G. Marchetti, and T. Rosing, "vgreen: A system for energy efficient computing in virtualized environments," in *Proc. 14th ACM/IEEE Int. Symp. Low Power electron. Des.*, 2009, pp. 243–248.
[43] Y. Wang, R. Deaver, and X. Wang, "Virtual batching: Request batching for energ conservation in virtualized servers," in *Proc. 18th Int. Workshop Quality Serv.*, 2010, pp. 1–9.
[44] J. Gong and C.-Z. Xu, "vPnP: Automated coordination of power and performance in virtualized datacenters," in *Proc. 18th Int. Workshop Quality Service*, 2010, pp. 1–9.
[45] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 289–302, Mar. 2007.
[46] S. J. Malkowski, M. Hedwig, J. Li, C. Pu, and D. Neumann, "Automated control for elastic n-tier workloads based on empirical modeling," in *Proc. 8th ACM Int. Conf. Autonomic Comput.*, 2011, pp. 131–140.
[47] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of hadoop clusters," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 61–65, Mar. 2010.
[48] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the energy efficiency of a database server," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 231–242.
[49] E. Koskinen and J. Jannotti, "BorderPatrol: Isolating events for black-box tracing," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 191–203, Apr. 2008.
[50] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen, "Performance debugging for distributed systems of black boxes," *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 74–89, Oct. 2003.

Gang Lu received the BS degree in computer science in 2006 from the Huazhong University of Science and Technology, China, and is working toward the PhD degree in computer science at the Institute of Computing Technology, Chinese Academy of Sciences. His research focuses on parallel and distributed computing.

Jianfeng Zhan received the PhD degree in computer engineering from the Chinese Academy of Sciences, Beijing, China, in 2002. He is currently a professor of computer science with the Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include distributed and parallel systems. He has authored more than 40 peer-reviewed journal and conference papers in the aforementioned areas. He is one of the core members of the petaflops HPC system and data center computing projects at Institute of Computing Technology, Chinese Academy of Sciences. He received the Second-class Chinese National Technology Promotion Prize in 2006, and the Distinguished Achievement Award of the Chinese Academy of Sciences in 2005. He is a member of IEEE.



Haining Wang received the PhD degree in computer science and engineering from the University of Michigan at Ann Arbor in 2003. He is an associate professor of Computer Science at the College of William and Mary, Williamsburg, VA. His research interests lie in the area of networking systems, security, and distributed computing. He is a senior member of IEEE.

Chuliang Weng is a principal researcher at Huawei Shannon Lab. In 2013, he joined Huawei, and as a technical director, started a research team focusing on building storage systems for virtualized datacenters. Before coming to Huawei, he was an associate professor of computer science at Shanghai Jiao Tong University. From 2011 to 2012, he was a visiting scientist with the Department of Computer Science at Columbia University in the City of New York. He received his Ph.D. from Shanghai Jiao Tong University in 2004, and his research interests include parallel and distributed systems, operating systems and virtualization, and storage systems. He is a member of the IEEE, ACM and CCF.