eCope: Workload-Aware Elastic Customization for Power Efficiency of High-End Servers

Bing Luo, *Student Member, IEEE*, Shinan Wang,
Weisong Shi, *Senior Member, IEEE*, and Yanfeng He

Abstract—Hardware components, especially CPU and memory, have made a lot of progress in terms of energy efficiency in the last decade. However, it is still far from the ideal energy proportional. Motivated by the recent observations that the energy efficiency of hardware components varies to a great extent depending on the workload characteristics, we propose eCope, workload-aware elastic customization for power efficiency of high-end servers, to reduce power consumption by workload aware and hardware customization for servers in datacenters. Our unique contribution is that eCope platform can take advantage of any configurable hardware that fits our assumption to improve the energy proportionality for various kinds of services without knowing the details of the target service. We illustrate three case studies to show how can we apply our idea to typical real-world back-end services (file system, database services and web-based services).

Index Terms—Energy-proportionality, sustainable computing, datacenter

1 Introduction

Energy management has now become a new focus in datacenters [1]. The state-of-the-art servers in datacenters are still far from being energy proportional [2]. Barraso and Hale report the CPU utilization of more than 5,000 servers during a six-month period, and they propose an energy proportional design for datacenter servers [2]. It means that "performance per watt" should be considered as the most important metric, particularly when the server is at the normal utilization level. After that, many approaches [3], [4], [5], [6], [7], [8] have been proposed to improve energy-proportionality in datacenters, using both software and hardware. However, they do not consider the different workloads of a server. Reiss et al. notice that workload characteristics are heterogeneous in resource types and their usage according to their analysis of the first publicly available trace data from a sizable multi-purpose cluster [9]. Furthermore, Voigt et al. find that workload characteristics are less steady and less predictable because applications are more agile and flexible [10]. This makes energy proportional design more difficult. Metri et al. try to understand how exactly the application type and the heterogeneity of servers and their configurations impact the energy efficiency of datacenters [11]. And they observe that each server has a different application specific energy efficiency values based on the type of application running, the size of the virtual machine, the application load, and the scalability factor.

- *B. Luo, S. Wang, and W. Shi are with the Department of Computer Science, Wayne State University, Detroit, MI 48202. E-mail: {bing.luo, shinan, weisong}@wayne.edu.*
- *Y. He is with the Alibaba Group, Hangzhou, China. E-mail: kuorui@taobao.com.*

Manuscript received 30 Nov. 2014; revised 7 June 2015; accepted 23 July 2015. Date of publication 4 Aug. 2015; date of current version 8 June 2016. Recommended for acceptance by C. Mastroianni, S. U. Khan, and R. Bianchini.
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/TCC.2015.2464802

Furthermore, even for the same server and same application running on it, Dean and Barroso notice that the latency variability is common, and the variability can be amplified by the scale [12]. In fact, variability is not only limited to the latency, it exists in all components of a server. Such dynamics and heterogeneity reduce the effectiveness of traditional energy proportional schema because traditional energy proportional schemas are usually optimized for a certain type of hardware or operating system or workload. So, it is better to design an elastic customization schema for servers.

There are many specific hardware customization approaches have been proposed to improve energy proportionality, including memory [13], [14], storage [15], [16], and multicore CPU [17], [18], [19], [20], [21], [22]. In this paper, we try to find a general workload-aware approach to achieve energy proportionality for servers in a datacenter. What we propose is called eCope, workload-aware **E**lastic **C**ustomization f**o**r **P**ower **E**fficiency of high-end servers, aiming to improve energy-proportionality by workload-aware hardware customization for servers in datacenters. More specifically, given a specific application and a workload range, we want to provide a framework that can find a way to achieve energy proportionality through hardware customization for a server. With eCope, we can find an optimized dynamic workload-power function and customize hardware according to both workload and the related optimized configuration.

The key contributions of eCope are summarized as following:

- A general workload-aware framework, eCope, is proposed to achieve energy proportionality for various kinds of services in datacenters.
- Energy proportionality is able to be achieved by eCope without knowing the details about the service by taking advantage of any configurable hardware that fits our assumption.

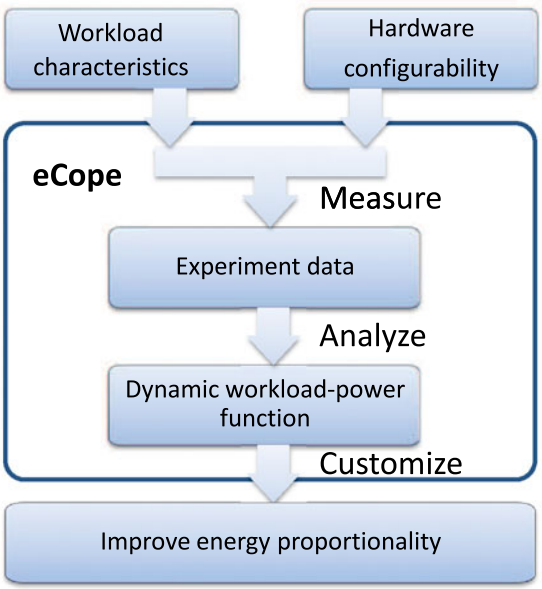


Fig. 1. The outline of eCope.

- The advantage of eCope has been demonstrated in three typical back-end services: file system, database service and web-based service.

We illustrate three case studies to show how can we apply our idea to typical real-world back-end services: Taobao File System (TFS) [23] (a distributed file system designed for small files), MySQL, PHP/Apache as case studies. Taobao is the 10th largest global site according to statics from Alexa [24]. Until 2009, there had been $2.86 * 10^{11}$ files stored on TFS, which occupies 1 PB (1024 TB) space. The number of files stored on TFS is still keeping increasing roughly two times every year. If each server can save power through the energy proportional design, it will be amplified by the scale significantly. As a result, eCope will have a huge impact on such a system. We argue that this type of elastic customization will be very useful for private datacenters, such as medium- and/or large-scale organizations, which have relative fixed types of workloads.

The rest of the paper is organized as following: Section 2 describes the design and methodology of eCope. Section 3 gives three case studies to show how can we apply our idea to typical real-world back-end services(file system, database services and web-based services). Section 4 discusses related works, and followed by conclusions and future works in Section 5.

2 eCope Design

Although Barraso and Hale propose energy proportional design for datacenter servers [2], there is no precise definition of how we can describe energy proportionality. The workload-power relation functions for current servers are still much higher than linear relation function [2], especially in the regular workload interval. To improve the energy proportionality, we want to reduce the power for the same workload. So, we use $\frac{workload}{power}$ to describe the energy proportionality. If the power is reduced for the same workload, this metric becomes larger. The aim of eCope is to find a general method and framework to improve energy proportionality

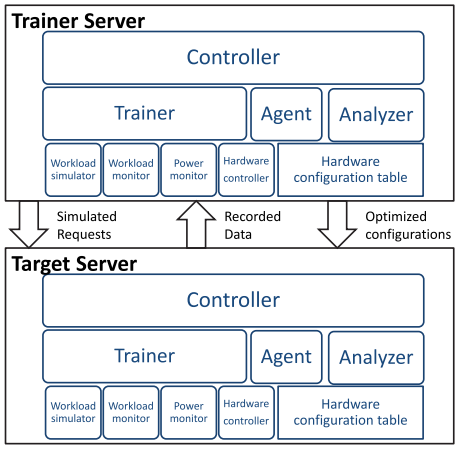


Fig. 2. The pair training process of eCope.

for servers within a datacenter. The servers that we focus on should satisfy the following assumptions:

- It is dedicated to run a particular application.
- Components of the server should be configurable to different states.
- For each configuration, the workload-power relation does not change over time.

There are two key observations behind our methodology. First, if we can fix the workload, different hardware configurations may result in different power behavior. There must be an optimal hardware configuration for this particular workload, so that we can do customization to improve energy proportionality. Second, for different workloads, the optimal hardware configuration may be different. Thus we need to have an elastic customization. In short, our goal is to identify the best hardware configuration under different workloads.

Fig. 1 shows the outline of eCope. The main input of eCope is workload characteristics and hardware configurability. Workload characteristics refer to the metric of instant performance such as network throughput, request per second, CPU utilization and so on. It can be measured by monitoring the NIC, CPU or the service. User needs to choose a suitable metric to describe the workload for their service. In our case studies, we choose the network throughput. Hardware configurability means what and how components can be configured (i.e., CPU can be switched into different frequencies, the hard disk can be set to different modes and so on). Although every configurable hardware can be included, it's better to choose the ones that can affect power effectively.

The basic eCope process consists of three phrases:

1) Pair training. We do training to get the relationship between the workload and the power for a given environment.
2) Analyzing. We then fit measuring data to get an optimized dynamic workload-power function.
3) Application. We apply the customization to improve energy proportionality.

2.1 Pair Training

The first phase is to do the training. Fig. 2 shows the process of training as well as the structure of the eCope framework.

Two servers are paired to train each other sequentially. The server that we want to optimize energy proportionality is the target server, and the other one that simulate requests and do analysis is the trainer server. eCope is deployed on both trainer server and target server.

The trainer component on both trainer sever and target server are active in this phase. They cooperate to call the hardware controller on target server so as to set the hardware to every possible configuration. And for each configuration eCope will do the following: 1) The power monitor on the target server measures the idle power. 2) The workload simulator on trainer server trigger requests to target server at different levels to generate necessary workload. Meanwhile, the power monitor and the workload monitor on target server record the real workload and related power dissipation on target server.

The workload simulator, power monitor, and workload monitor and hardware controller involved in these processes are the basic components of eCope and may varies for different hardware or services. For example, we can use NodeManager to monitor system power, or we can use wattsup to monitor system power. It depends on what kind of devices are available. So, in our design, we use plugin mechanism to makes it flexible. Each of these components has a selector to determine which one to use at run time. So that we can implement both NodeManager based monitor and wattsup based monitor as plug-ins and the power monitor selector will choose the right one to use according to the user input.

2.2 Analyzing

After all the data described are collected, the target server sends them to trainer server. In the second phase, the Analyzer on the trainer server is responsible to analyze the measured data and sends the optimized configuration table to target server. To do so, we first need to find a proper function to do curve fitting for workload-power relation. After the curve fitting for each configuration, we can get a set of static workload-power functions $\{f_1, f_2, \ldots, f_n\}$.

The static workload-power function is the workload-power relation function associated with just one configuration. If a workload-power relation is achieved by using more than one configuration (which means in different workload intervals the power may be related to different configurations), then we refer it as the dynamic workload-power relation. Its function is called the dynamic workload-power function. We denote dynamic workload-power function as $(\{f_1, f_2, \ldots, f_n\}, \{x_1, x_2, \ldots, x_n\})$, where f_1 to f_n are static workload-power functions and x_1 to x_n are intervals that f_1 to f_n are effective on respectively. The union of all x_i should be the whole possible workload interval, and x_i should be pairwise disjoint. Formally, $(\{f_1, f_2, \ldots, f_n\}, \{x_1, x_2, \ldots, x_n\})$ means:

$$f(x) = \begin{cases} f_1(x), & if \quad x \in x_1 \\ f_2(x), & if \quad x \in x_2 \\ \quad \ldots & \\ f_n(x), & if \quad x \in x_n. \end{cases} \tag{1}$$

In this way, we can mix different configurations on one graph. For simplicity, we can treat static workload-power function as a special dynamic workload-power function that has only one function and one interval. Among all possible dynamic workload-power functions that can be constructed by a certain set of static workload-power functions, there must exist an optimal dynamic workload-power function that achieves the best energy proportionality under every possible configuration and also meets the performance requirement and energy condition(which we will discuss in detail in Section 2.4). We refer it as the optimized dynamic workload-power function. So the aim of this phase is to find the optimized dynamic workload-power function.

Generally, we can obtain all the intersection points to separate the workload interval and find the functions that have the lowest power in each interval and meet the performance limitation. Then combine these functions together with the interval that is between two neighboring intersection points. In this way, the complexity is $O(n^3)$.

Since obtaining optimized dynamic workload-power function only need to be done once, and there are not too much hardware configurations on current servers, such complexity is acceptable. In fact, in our case study, the calculation spends less than 1 second. Even though, For most particular fitting functions, we may have better ways to get the optimized dynamic workload-power function. These methods are not mainly for improving performance, but for easier programming. We will see an instance of how to do so in the case study part.

Here, the analyzing process is done on the trainer server. However, since both trainer server and target server have an analyzer component, the analyzing process can be done on target server as well. User can choose which one to use for their convenience.

2.3 Application

After the optimized dynamic workload-power function for both servers are obtained, these servers can just work on its own(not paired), and customization can be achieved according to this function. In other words, when the service is running, the agent component monitors the workload and applies the configuration related to the interval where the current workload is. For example, if the optimized dynamic workload-power function is ($\{f_1,f_2,f_3\}$, {(10,30],[0,10),[30,50]}) and the current workload is 20, then configuration 1 will be applied.

2.4 Discussing

Our methodology can be applied to any applications running on the server that meet our assumptions. We do not limit the type of hardware or application in our method. Currently, we can modify CPU frequency, network speed, hard disk mode. In the future, we may be able to change the memory frequency. User can choose any hardware that satisfy our assumption.

According to our definition, we try to increase $energy\ proportionality = \frac{workload}{power}$. And

$$\begin{aligned} Energy &= \int_0^t \text{power}\,\mathrm{d}t \\ &= \int_0^t \frac{\text{workload}}{(\frac{workload}{power})}\,\mathrm{d}t \\ &= \int_0^t \frac{workload}{Energy \quad proportionality}\,\mathrm{d}t. \end{aligned}$$

We can see that if the workload can be fixed or the workload does not change too much, then only when energy proportionality increases, the energy decreases. And the workload is determined by the user, which means it is independent to the configuration. Thus to improve energy proportionality is equivalent to reducing the energy. On the other hand, if the workload is allowed to change with in a performance requirement, we need to know how much energy proportionality improvement is required to ensure energy saving.

To do so, we need to have a performance limitation to prevent too much performance loss. Assume the maximum performance loss could be α(percentage), after we change the configuration, denote the new workload as $workload'$, and denote the new execution time as t'. Then

$$\overline{workload'} \geq (1-\alpha) * \overline{workload}$$

$$So\ that$$

$$t' = \frac{1}{\overline{workload'}} \leq \frac{1}{(1-\alpha) * \overline{workload}} = \frac{1}{1-\alpha} * t$$

$$Then,$$

$$\begin{aligned} Energy &= \int_0^{t'} \frac{workload'}{Energy \quad proportionality}\,\mathrm{d}t \\ &\leq \int_0^{\frac{1}{1-\alpha}*t} \frac{workload}{Energy \quad proportionality}\,\mathrm{d}t \\ &\approx \frac{1}{1-\alpha} * \int_0^{t} \frac{workload}{Energy \quad proportionality}\,\mathrm{d}t. \end{aligned}$$

This means that if energy proportionality increase more than ($\frac{1}{1-\alpha}$) times, it can ensure energy saving, otherwise, the configuration should not be considered. This is called energy condition in our paper. Since it is deduced by performance requirement, when we say performance requirement in this paper, it also includes energy condition.

We find that the dynamic workload-power relation is a good tool to show the effect of hardware configuration. Not only because we can easily compare energy proportionality under different workload-power relations on the graph, but also because it provides a uniform method to calculate the optimized configuration. We can also use it to do customization. Therefore, the dynamic workload-power relation is the core data structure of eCope.

3 Case Study

As we described in Section 2, we are interested in certain applications running on the dedicated servers in datacenters. There are three particular services: file system, database services and web-based services.

We take TFS, MySQL, and PHP/Apache as our case studies since they fit our assumption in Section 2 very well, and they are also typical types of back-end services running in real-world. TFS is a Linux-based distributed file system which provides high reliability and concurrent access by redundancy, backup, and load balance technology. TFS is mainly designed for small files less than 1 MB in size and adopts a flat structure instead of the traditional directory structure. The open source TFS project is developed and maintained by Taobao, a part of Alibaba Group.

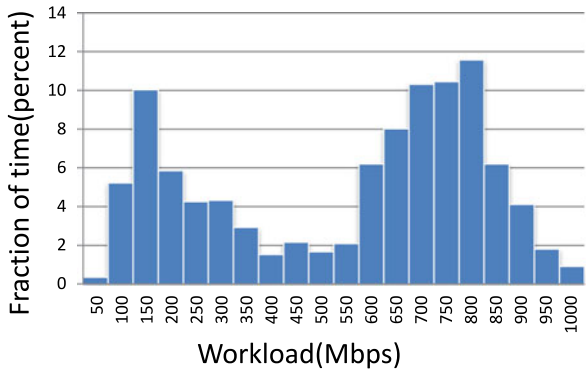


Fig. 3. The workload distribution of TFS.

In our case studies, the throughput of the network transfer rate is a good metric for workload. It is obvious for TFS and MySQL. For PHP/Apache case study, although request per second is also a good metric, network throughput can equivalently describe the workload since the page sizes are the same in our experiment. In addition, network throughput is service independent, which means network throughput monitor can be also used for a wide range of services. We would like to emphasis that user can choose any other metrics that are able to describe workload. To avoid confusion, however, all 'workload' in the case study section refer to the network throughput (measured by Mbps).

We conduct TFS case studies in the same environment as TFS production environment in Alibaba. MySQL and PHP/Apache case studies are conducted on our lab servers.

The performance limitation for three case studies is maximum 5 percent of performance loss. The hardware configurations include the combination of 16 CPU frequencies (from 1.2 to 2.5 GHz, 2.7 GHz, and Turbo boost mode), three kinds of network interface controller (NIC) speeds (10, 100, and 1,000 Mbps), and three kinds of disk modes (Normal, Standby, and Sleep). So, each configuration should include these three components, e.g., (1.2 GHz, 100 Mbps, Normal).

The network switches we used in all our case studies are 1 G network switches, which means that the maximum network throughput is 1,000 Mbps, so the range of wokload is 0 to 1,000 Mbps. TFS and MySQl are IO intensive. Fig. 3 shows a typical normal workday workload distribution of TFS on one server provided by Taobao Crop. We can see that the server is rarely idle, and in most time, the workload is around 100 to 20 Mbps and 600 to 800 Mbps, while the CPU utilization is always lower than 20 percent. For PHP/Apache case study, we implement a simple service that dynamically computes π and return it through the web interface. Our experiment data shows that when the CPU utilization is 100 percent, the network throughput is under 60 Mbps, which is far below the maximum network throughput, so PHP/Apache case study is CPU-bounded.

3.1 Basic Components Implementation

For TFS, it has its own interfaces to access the files, so the workload simulator was implemented by using a TFS client API. Before the experiment starts, we store amount of files to TFS, and save the filenames of all these files to a filename list. When we launch the workload simulator, we pass a desired number of files and number of processes to it. Each

workload simulator process first reads all filenames from the list. Then, randomly picks a desired number of files to read from TFS. To read a file, the workload simulator connects with the nameserver first, and then it sends the block id to main nameserver to get the address of desired dataserver. Later, the workload simulator uses that address to connect with the dataserver, send both the block ID and file ID to it, and get the file data from it directly. Thus, the entire workload can be controlled by passing different numbers of files and processes. Most of the energy is consumed by the dataservers throughout the entire process. In our experiment, we only optimize the dataserver. The nameservers and heart agents run on separate servers.

We use SQL workbench as the MySQL workload simulator. Before the experiment starts, we store a dataset on a MySQL server. When launching the workload simulator, we pass the number of records and the number of processes to it. Each workload simulator process queries the same number of records by generating a SQL statement.

We use Apache Bench as the PHP/Apache workload simulator. We implement Chudnovsky algorithm in a php page that dynamically computes π using BCMath arbitrary precision mathematics functions in PHP. Then, we invoke Apache bench with a desired number of requests and number of concurrency to access the page.

We use the same power monitor and workload monitor for our case studies. The workload monitor records NIC throughput and the power monitor reads power by using Intel Node Manager and Watts up. Node Manager is a set of hardware and software to optimize and manage power and cooling resources in the data center. This server power management technology extends component instrumentation to the framework level and can be used to get power information from sensors integrated on motherboard chips.

Our power monitor can read power information from both Wattsup and Node Manager. We connected the Wattsup device to our dataserver, but the power data can be read from the USB interface connected to the Wattsup device. Node Manager is supported on our dataserver, we can read information locally. We can also read power information from node manager through the network interface by using IPMI protocal. In order to limit the overhead on the dataserver, we read all power data from another server. Since the data transferred by node manager is quite small (less than 1 kbps each time) compared to the workload of TFS or MySQL (measured by Mbps), we can neglect it and consider all network data to be generated by TFS or MySQL.

Fig. 4 shows the structure of our power monitor. When it is launched, we must first specify which drive to use. Both drivers implement the same interfaces. For Wattsup, it can only monitor the whole system power. We construct a serial device manager to communicate with the USB interface. It sends the command packets and receives data packets (called WUPacket) by using Wattsup protocol, but it does not know the meaning of these packets. The WUPacket parser mainly processes the data packet, and WUlog component is used to construct a command packet to control the Wattsup device. Wattsup drive asks WUlog to initialize the Wattsup device on start. When receiving the packet from the serial device manager, WUPacket parser extracts each field in the packet and returns it to the Wattsup drive.

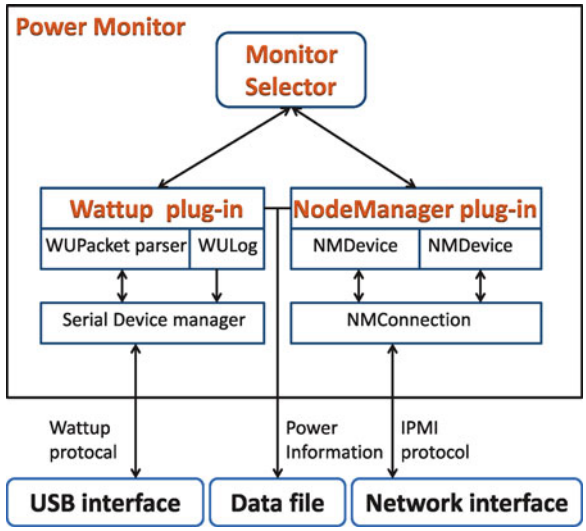


Fig. 4. The structure of power monitor.

However, in some cases, the Wattsup device does not send out data for a long time.

By default, we ask Wattsup device to collect data for every one second. If Wattsup drive finds that the serial device manger has failed to read data from USB for a certain time, then it will ask WUlog to reset the device. For Node Manager, we have two ways to connect to the network. We choose the intelligent platform management interface (IPMI) approach because it can be used on other servers that do not support Node Manager but still support IPMI. Node Manager is different from Wattsup. It can monitor not only total system power, but also component level power, such as CPU and memory. So there are several node manager device classes that are responsible for each component. All Node Manager device classes use the same node manager connection class to communicate with node manager on the target server. Both the Wattsup driver and node manager driver can save the power information to a specific data file.

3.2 Process

In the measuring phase, the trainer component collects the power and workload information under various situations, including the following: (1) when the system is idle. (2) when turning on the TFS or MySQL but not putting any workload on it. (3) when there are workload on TFS or MySQL server, but no hardware control. (4) when there are workload on TFS or MySQL server, and TFS or MySQL is running under a certain hardware configuration. We can analyze these data to identify the optimal configuration for a particular workload.

Before we start analyzing, we need to determine a fitting function for workload-power relation. This function is related to the environment. User can choose the best one fits their training data. We tried different types of functions like linear, polynomial, power, exponential function and so on. We decide to use power function because its coefficient of determination(or R-square) shows the best fitting result among all these functions, which means the power function is the best one to describe the relation between workload and power for our experiment platform. The power function has the form:

$$DynamicPower = a * workload^b. \quad (2)$$

Although the optimized dynamic workload-power function can be calculated by the method described in the Section 2, we find a better way to do this for our case studies. Suppose two configurations (denoted as A and B) have the workload-power relation $f = a_1 * x^{b_1}$ and $g = a_2 * x^{b_2}$, where $a_1, c_2 > 0$ and $b_1, b_2 > 0$, there is only one positive intersection point:

$$x = \left(\frac{a_2}{a_1}\right)^{\frac{1}{b_1 - b_2}}. \quad (3)$$

It means that this point is a turning point. If configuration A consumes less power when the workload is lower than this point, then configuration B consumes less power when the workload is higher than this point. Of course, mathematically, this point can be any value even higher than the maximum possible workload, so we need to check whether the point is in the range (in our case 0 to 1,000).

Algorithm 1. Obtain Optimized Dynamic Workload-Power Function

$i = 0$; $f_1 = g_1$; $w_1 = Maxworkload$;
$y = g_1(Maxworkload)$;
$Candidate = \{g_v | v = 1 \quad to \quad n\}$;
for all $i = 2 \quad to \quad Maxworkload$ **do**
  if g_i *violate the perf. limitation* **then**
    $Candidate = Candidate - g_i$;
  else
    if $g_i(Maxworkload) < y$ **then**
      $f_1 = g_i$;
      $y = g_i(Maxworkload)$;
    end if
  end if
end for
$Candidate = Candidate - \{f_1\}$;
$i = 2$; $tmp = 0$; $Over = \emptyset$; $f_{i+1} = NULL$;
while $f_{i-1} \neq NULL$ **do**
  $w = 0$;
  for all $g \in Candidate$ **do**
    $tmp = (\frac{aValue(g)}{aValue(f_{i-1})})^{\frac{1}{bValue(f_{i-1}) - bValue(g)}}$;
    if $tmp >= w_{i-1}$ **then**
      $Over = Over \cup \{g\}$;
    else
      if $tmp > w$ **then**
        $w = tmp$;
        $w_i = w$;
        $f_i = g$
      end if
    end if
  end for
  $Candidate = Candidate - Over - f_i$;
  $i = i + 1$;
end while
return $(\{f_1, f_2, \ldots, f_{i-1}\}, \{[0, w_{i-1}), \ldots, [w_3, w_2), [w_2, w_1]\})$;

Assuming that the set of static workload-power functions is $G = \{g_v | v = 1 \quad to \quad n\}$. Using Equation (3), algorithm 1 gives a better way to obtain the optimized dynamic workload-power function.

In the worst case, we can find one function during each iteration and the set $Over$ is always empty. This results in a running time complexity of $O(n^2)$. Given the fact that the configurations on current servers are not too much, and the training just need to be done once, the performance is not an issue. We finish the calculation less than one second for both TFS and SQL case studies. The benefit of this algorithm is to make programming easier.

At last, eCope applies customization. The agent component lookup the optimized dynamic workload-power function periodically and change the hardware configuration if needed.

3.3 Evaluation

To evaluate eCope, we first measure system idle power, TFS idle power, MySQL idle power, and PHP/Apache idle power under all possible hardware configurations. We also measure the static workload-power relation without any optimization to get a baseline. Then we launch the simulator to supply workload on TFS, MySQL and PHP/Apache, and proceed to measure the system power under different hardware configurations. Next, these data are fit into the power function in order to get the static workload-power function, and an optimized dynamic workload-power function is calculated by using algorithm 1. Lastly, we apply the customization and compared the power saving.

When we do the baseline measurement, the DVFS function is turned off in BIOS setting so that no governor is activated, the NIC speed is 1,000 Mbps, and the disk mode is normal. Otherwise, userspace governor is used, so that the CPU frequencies are controlled by eCope completely.

3.3.1 Experimental Environment

MySQL 5.1.52, PHP 5.5.22, Apache 2.4.12 are set up on an Intel R2000GZ family server in our lab as target server with Intel Xeon CPU E5-2680 0 @ 2.70 GHz and DDR3 1333 MHz 8*8 GB Memory. Our workload simulators are deployed on a Dell 01V648 server. The Intel server is the target server and the Dell server is the trainer server. The operating system of the Intel server and the Dell server are RedHat 6 $\times$ 86_64 and CentOS 4 $\times$ 86_64 respectively. The Intel server and the Dell server are connected by a 1 G network switch. The Intel server support node manager that enables reading the system power, CPU power, and memory power information. In addition, Wattsup is set up to compare system power to the data collected from Node Manager. TFS 2.1.13 is set up on the same type of server as the production TFS server in Alibaba Group for our experiment. The TFS server is equipped with Xeon CPU E5-2400 0 @ 2.20 GHz and 10*10 TB disks. Also servers are connected by a 1 G network switch, so that the range of workload is 0 to 1,000 Mbps.

3.3.2 Base Line

Fig. 5 shows some results of system idle power under three kinds of hardware configurations. We observe that in this period, the difference between the average power of the highest CPU frequency (2.71 GHz) and the average power of the lowest CPU frequency(1.2 GHz) is still less than 1 Watt. Therefore, the average idle power under different configurations is almost the same. In addition, we notice that the

Fig. 5. The system idle power over three periods.

power changes periodically. Thus, we define idle power as the average power over integral times of periods. As a result, when we calculate the idle power, we always take the same number of periods of data to avoid errors caused by such periodic phenomena. For each configuration, we collect the system idle power for one day. Table 1 shows the average system idle powers (si power), which indicates that the system idle power is almost the same under different hardware configurations although the power vibrates over time. Thus, we can treat the system power the same under different configurations. Tables 1, 2, and 3 also show the average idle power when the service is on but no workload. When TFS is running but has no workload on it, the power increases when the CPU frequency increases. The difference between maximum and minimum power is about 1.4 percent. On the other hand, When the idle power of MySQL and PHP/Apache does not change so much.

Next, we measure the power under different workloads without any optimization. Since the idle power changes periodically, the dynamic power should be calculated carefully. Fig. 6 shows the total system power when the simulator launched four jobs sequentially with the same workload. Number 1 to 4 in the figure shows when these four jobs are launched, and the red line roughly shows the idle power. We can see that when the first two jobs are working, the idle power is about 117 Watts. When the last two jobs are executing, the idle power is about 125 Watts. The total system powers for all those four jobs are, however, almost the same. It is surprising that when the idle power increased about 7 watts, the execution time and total system power are almost the same.

We also check it for those low workloads that consume 130 Watts total system power, and observed the total system powers are almost the same while the idle power changes periodically. So, it is not capped at a single server level. We repeated the experiment under different configurations and different workloads, and found that this phenomenon

TABLE 1
The Average System Idle Power and the Average TFS Idle Power

Freq. (GHz)	Sys idle power (Watts)	TFS idle power (Watts)
1.2	164.53	173.91
1.3	164.51	173.93
1.4	163.49	173.88
1.5	164.52	173.97
1.6	164.54	173.89
1.7	164.50	173.96
1.8	164.54	173.91
1.9	164.51	173.87
2.0	164.50	173.93
2.1	164.54	173.90
2.2	164.55	173.96
Turbo Boost	117.27	121.19

TABLE 2
The Average System Idle Power and the Average MySQL Idle Power

Freq. (GHz)	Sys idle power (Watts)	MySQL idle power (Watts)
1.2	117.46	117.51
1.3	117.50	117.73
1.4	117.40	117.47
1.5	117.40	117.54
1.6	117.34	117.63
1.7	117.41	117.74
1.8	117.31	117.88
1.9	117.23	117.79
2.0	117.71	117.96
2.1	117.66	117.79
2.2	117.51	117.63
2.3	117.59	117.90
2.4	117.45	117.65
2.5	117.35	117.69
2.7	117.33	117.35
Turbo Boost	117.27	117.46

TABLE 3
The Average System Idle Power and the Average PHP/Apache Idle Power

Freq. (GHz)	Sys idle power (Watts)	PHP/Apache idle power (Watts)
1.2	117.33	117.43
1.3	117.43	117.47
1.4	117.38	117.43
1.5	117.40	117.51
1.6	117.51	117.59
1.7	117.42	117.53
1.8	117.57	117.63
1.9	117.39	117.54
2.0	117.41	117.47
2.1	117.54	117.61
2.2	117.63	117.81
2.3	117.39	117.55
2.4	117.67	117.79
2.5	117.42	117.61
2.7	117.52	117.55
Turbo Boost	117.73	117.81

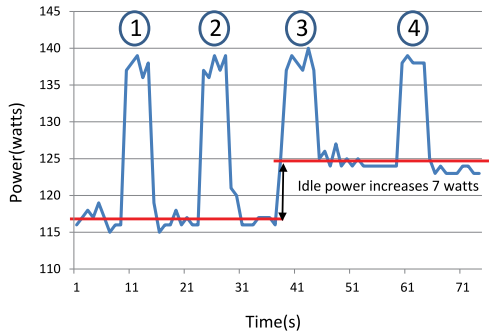


Fig. 6. Total system power when four jobs are executed sequentially with same workload but different idle power.

is common in our experimental environment. This means that the total power might not always equal to the idle power plus dynamic power. This may be caused by uncore power, but we haven't yet identified why this happens. We will continue to explore the reasons. Since it is not related to this paper, in our evaluation, we define the dynamic power as the average total power minus the average service idle power so that on average, the total power is still equal to the idle power plus the dynamic power. We used the power function, shown in Equation (2), to fit the workload-power relation. For TFS, the function is:

$$DynamicPower = 0.1140 * workload^{0.8449}. \quad (4)$$

For MySQL, the function is:

$$DynamicPower = 0.1310 * workload^{0.8313}. \quad (5)$$

For PHP/Apache, the function is:

$$DynamicPower = 24.8839 * workload^{0.5087}. \quad (6)$$

These functions are used as the base line to make comparisons with our optimization.

TABLE 4
Fitting Results for TFS

frequency (GHz)	a	b	R-square
1.2	2.7495	0.2161	0.9341
1.3	2.5566	0.2336	0.9362
1.4	1.4905	0.3185	0.9276
1.5	2.2639	0.2610	0.9207
1.6	1.9094	0.2946	0.8932
1.7	0.7858	0.4613	0.9311
1.8	0.4405	0.5751	0.9445
1.9	1.5794	0.3956	0.9376
2.0	2.0666	0.3631	0.9575
2.1	2.2627	0.3504	0.9554
2.2	2.0027	0.3796	0.9569
Turbo Boost	2.6411	0.3820	0.9123

3.3.3 *Analyzing*

In this part, we obtain the optimized workload-power functions. For TFS, Different workloads are achieved by using different numbers of workload simulator processes. All the sizes of test files were 100 KB, and each thread operates on 1,000 files. Next, we calculate the dynamic power and used power function to do least squares fitting on these data. Table 4 shows the fitting result when the NIC speed is 1,000 Mbps and hard disk mode is normal. Most b values in the table are smaller than 0.7, which means that the power function fits better than a linear function because the set of linear functions is a subset of power functions. Fig. 7 shows part of the related figure of static workload-power functions. Using the algorithm 1, we obtain the optimized workload-power function for TFS as:

$$f(x) = \begin{cases} f_{1.8}(x), & if\ x \in [0, \quad 115.6) \\ f_{1.4}(x), & if\ x \in [115.6, 395.3) \\ f_{1.2}(x), & if\ x \in [395.3, 1000]. \end{cases} \quad (7)$$

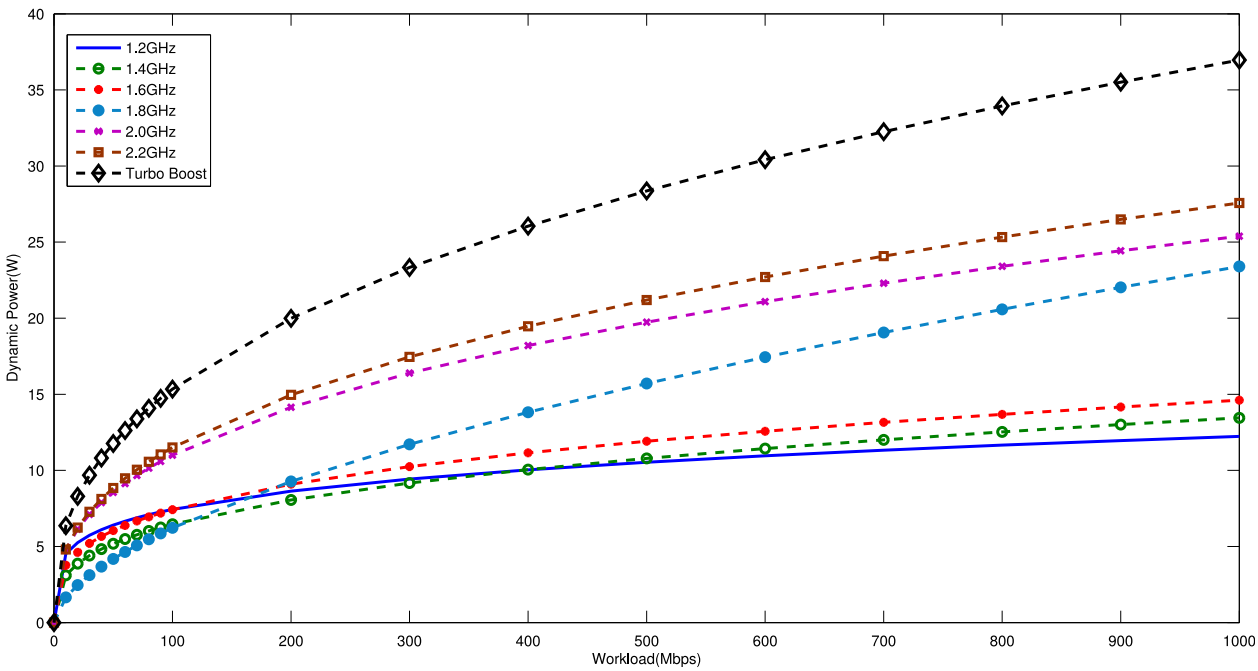


Fig. 7. TFS static workload-power functions (when NIC is 1,000 Mbps and disk is in normal mode).

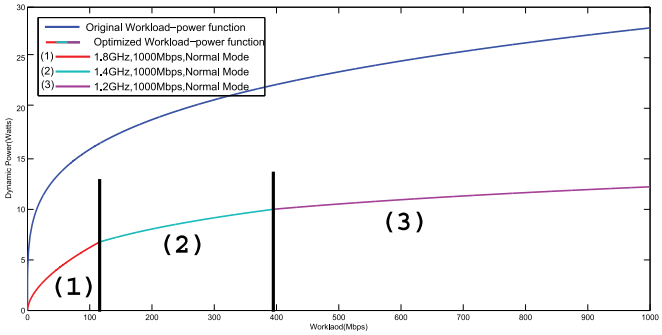


Fig. 8. The original workload-power relation and The optimized workload-power relation for TFS.

The curve with four colors in Fig. 8 shows the graph of the function. It contains three configurations that related to different CPU frequencies.

For MySQL, Different workloads are achieved by using different numbers of workload simulator processes. Each process selects half of the data in the database. Table 5 shows the fitting result when the NIC speed is 1,000 Mbps and hard disk mode is normal. We can see that b values for MySQL are larger than those for TFS. Some b value even reach 0.9994, which means that it is almost linear. Fig. 9 shows part of the related figure of static workload-power functions. Using the algorithm 1, we obtain the optimized workload-power function for MySQL as:

$$f(x) = \begin{cases} f_{1.2\,GHz/100\,Mbps}(x), & if\ x \in [0, \quad 70.9) \\ f_{1.2\,GHz/1000\,Mbps}(x), & if\ x \in [70.9, 1000]. \end{cases} \quad (8)$$

The curve with two colors in Fig. 10 shows the graph of the function. It contains two configurations that have the same CPU frequencies, but different network speed. Fig. 10 also shows the base line of MySQL that we obtained in Section 3.3.2.

TABLE 5
Fitting Results for MySQL

frequency(GHz)	a	b	R-square
1.2	0.03771	0.8835	0.9926
1.3	0.04482	0.9636	0.9997
1.4	0.05423	0.8647	0.9937
1.5	0.1608	0.7411	0.9606
1.6	0.09302	0.7986	0.9923
1.7	0.1774	0.7287	0.9806
1.8	0.1841	0.7534	0.9591
1.9	0.03406	0.9994	0.9620
2.0	0.0475	0.9978	0.9901
2.1	0.1932	0.7234	0.9560
2.2	0.0951	0.8502	0.9881
2.3	0.1921	0.7534	0.9842
2.4	0.0947	0.8573	0.9996
2.5	0.0832	0.9330	0.9873
2.7	0.1214	0.8010	0.9376
Turbo Boost	0.1394	0.8482	0.9220

For PHP/Apache, Different workloads are achieved by using different concurrency level. Table 6 shows the fitting result when NIC speed is 100 Mbps and hard disk mode is normal. Fig. 11 shows all related figures of static workload-power functions that meets the performance requirement and energy condition, when NIC speed is 100 Mbps and hard disk mode is normal. Notice that configurations with lower frequency are not shown in the Fig. 11 because they either causes more performance loss than performance requirement(which is maximum 5 percent performance loss) or they violate the energy condition. Using the algorithm 1, we obtain the optimized workload-power function for PHP/Apache as:

$$f(x) = f_{2.4\,GHz/100\,Mbps}(x), \quad if\ x \in [0, \quad 60]. \quad (9)$$

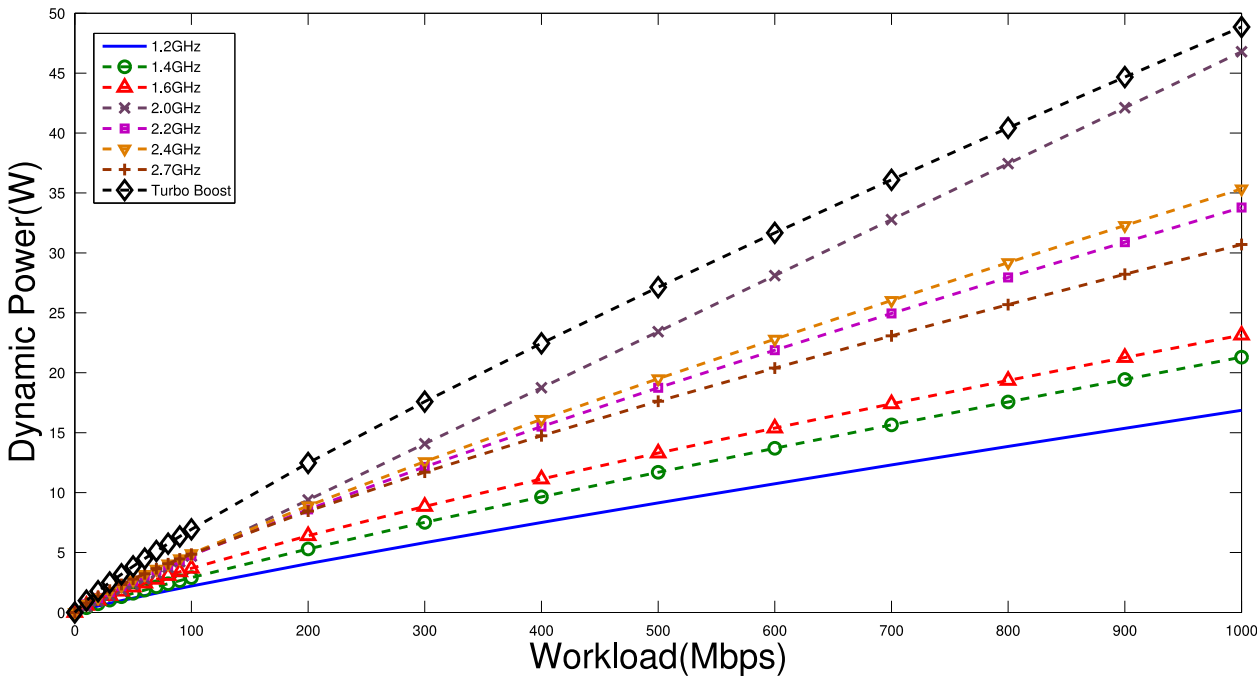


Fig. 9. MySQL static workload-power functions(when NIC is 1,000 Mbps and disk is in normal mode).

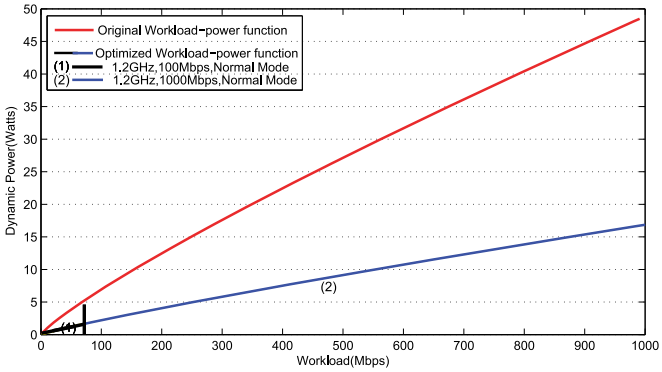


Fig. 10. The original workload-power relation and the optimized workload-power relation for MySQL.

The blue curve in Fig. 12 shows the graph of the function. It contains only one static workload-power relation function which is easier to apply.

3.3.4 Customization Results

After we apply the customization to TFS, MySQL, and PHP/Apache, we measure the power under our control and find that compared to the original behavior, TFS can save up to 51.1 percent of dynamic power and 41.5 percent dynamic power on average (up to 12.0 percent total system power, and 7.0 percent total system power on average) with average 0.57 percent performance loss. For MySQL, it can save up to 65.5 percent of dynamic power and 65.3 percent dynamic power on average (up to 19.3 percent total system power, and 12.2 percent system power on average) with average 0.98 percent performance loss. For PHP/Apache, it can save up to 19.6 percent of dynamic power and 18.37 percent dynamic power on average (up to 11.6 percent total system power, and 9.8 percent system power on average) with average 4.7 percent performance loss.

TABLE 6
Fitting Results for PHP/Apache

frequency(GHz)	a	b	R-square
1.2	12.8805	0.5651	0.9991
1.3	14.7596	0.5255	0.9978
1.4	14.0645	0.5499	0.9993
1.5	14.7331	0.5431	0.9986
1.6	15.8533	0.5277	0.9989
1.7	16.3604	0.5226	0.9976
1.8	17.2294	0.5140	0.9983
1.9	17.8611	0.5094	0.9988
2.0	17.3629	0.5242	0.9990
2.1	19.1891	0.5043	0.9997
2.2	20.0848	0.5001	0.9988
2.3	20.0652	0.5078	0.9979
2.4	20.4294	0.5070	0.9983
2.5	20.8062	0.5121	0.9997
2.7	24.6902	0.4876	0.9689
Turbo Boost	30.7797	0.5095	0.9983

4 Related Work

Researchers usually try to achieve energy-proportionality from two different aspects of view: the service itself or the available hardware. So previous studies in this area generally fall into two ways: One way is to understand how a specific service is running and the using these information to do optimization. The other way is to utilize new features of hardware or design new hardware or device.

Characteristics of a service or application is helpful to do fine-grained optimization. Xu et al. [25] propose an energy-aware query optimization framework, PET, enables the database system to run under a DBA-specified energy/performance tradeoff level via its power cost estimation module and plan evaluation model. They also introduce a

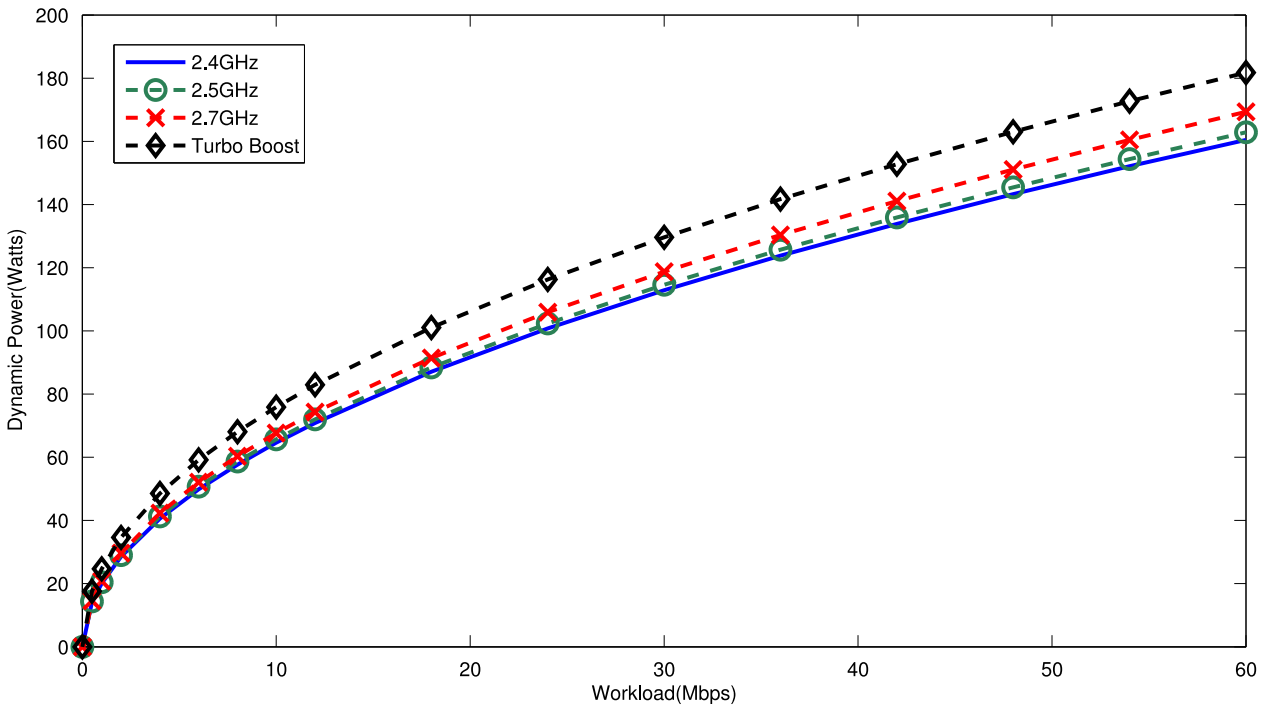


Fig. 11. PHP/Apache static workload-power functions(when NIC is 100 Mbps and disk is in normal mode).

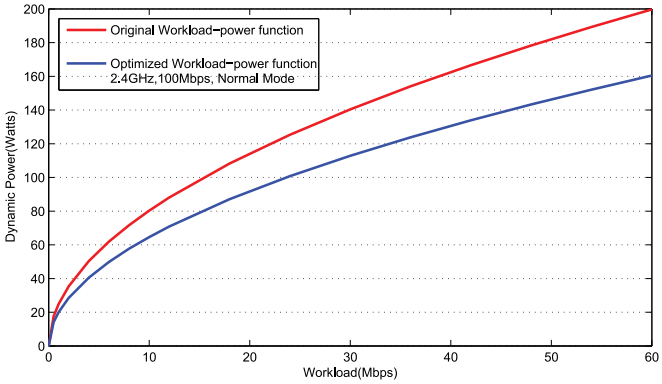


Fig. 12. The original workload-power relation and the optimized workload-power relation for PHP/Apache.

power-aware online feedback control framework for energy conservation at the DBMS level based on rigorous control-theoretic analysis for guaranteed control accuracy and system stability [26]. Both work are built as a part of the PostgreSQL kernel. Zheng et al. [16] notice that storage servers consume significant amounts of energy and are highly non-energy-proportional. So they propose a storage system, called LogStore, that enables two-speed disks to achieve substantially increased energy proportionality and, consequently, lower energy consumption. Psaroudakis et al. [27] argue that databases should employ a fine-grained approach by dynamically scheduling tasks using precise hardware models and so they propose a dynamic fine-grained scheduling for DBMS memory accessing. Lang et al. [28] focus on designing an energy-efficient clusters for database analytic query processing. They explore the cluster design space using empirical results and propose a model that considers the key bottlenecks to energy efficiency in a parallel DBMS. Amur et al. [15] focuses on large-scale cluster-based storage and data-intensive computing platforms that are increasingly built on and co-mingled with such storage. They propose Rabbit, which is a distributed file system that arranges its data-layout to provide ideal power-proportionality down to very low minimum number of powered-up nodes.

On the other hand, using new hardware design can also reduce the power directly for different kinds of services. Malladi et al. [29] observed that currently DDR3 memory in servers is designed for high bandwidth but not for energy proportionality. Mobile-class memory, however, addresses the energy efficiency challenges of server-class memory by forgoing more expensive interface circuitry. Therefore they take advantage of mobile DRAM devices, trading peak bandwidth for lower energy consumption per bit and more efficient idle modes. Zhang et al. [30] believes that current fine-grained DRAM architecture incurs significant performance degradation or introduces large area overhead. So they propose a novel memory architecture called Half-DRAM. In this architecture, the DRAM array is reorganized that only half of a row can be activated. Hu et al. [31] focused on how energy-saving mechanisms through the design of Internet transmission equipment e.g., routers, and green reconfigurable router (GRecRouter). they mainly contribute to the design and manufacture of some core components of a green Internet like energy-efficient routers. Lo et al. [32] present PEGASUS, a feedback-based controller that using new feature of current available CPU, called running average power limit (RAPL) to improves the energy proportionality of WSC systems.

Fu et al. [33] present a practical and scalable solution, Cloud- PowerCap, for power cap management in a virtualized cluster. It is closely integrated with a cloud resource management system, and dynamically adjusts the per-host power caps for hosts in the cluster. Chen et al. [34] try to address challenges of reliability and energy efficiency of resource-intensive applications in an integrated manner for both data storage and processing in mobile cloud using k-out-of-n computing. Kazandjieva et al. [35] take the advantages of different classes of devices and put the application running on the best location. Their implementation, called Anyware, provides desktop-class performance while reducing energy consumption through a combination of lightweight clients and a small number of servers. Recently, it is suggested that we can halt the system when the it is idle, and using a static rate when it is busy. This strategy performs almost as good as an optimal speed scaling mechanism [36]. Wong and Annavaram [37] present Knight Shift that presents an active low power mode. By the addition of a tightly-coupled compute node, their system enables two energy-efficient operating regions. Liu et al. [38] present a runtime power management tool called SleepScale, which is designed to efficiently exploit existing power control mechanisms. Pillai and Shin [39] present real-time DVS algorithms that modify the OS's real-time scheduler to provide energy savings while maintaining real-time deadline guarantees.

Compared with previous work, our work is looking for a general workload-aware framework that can improve energy proportionality without knowing the details about the target service. So that we do not need to modify the current service, and thus can support various kinds of services. On the other hand, although we use DVFS as an example of the configurable hardware in our case study, our framework can take advantage of any configurable hardware (even for future hardware) that fits our assumption. For example, in MySQL and PHP/Apache case studies, the optimized configuration includes the NIC configuration.

5 Conclusions and Future Work

In this paper, we propose a general approach, eCope, to improve energy-proportionality by workload-aware and hardware customization for servers in datacenters, and we obtain the optimized configuration by using our optimized dynamic workload-power function. We implemented eCope for TFS and MySQL, and applied a more specific and effective method to get the optimized dynamic workload-power function. In next step, we will try to extend our eCope to rack or cluster level. Since we know the workload-power relation for each server, we can do workload schedule according to these workload-power relations to make the entire rack or cluster energy proportional. We notice that currently there is not much configurable hardware available on the market. This may be a limitation when applying the approach. However, our eCope methodology is easy to extend to software customizations because in fact, we only assume that there are different configurations that can affect system power. So we will also explore software customizations.

ACKNOWLEDGMENTS

The authors would like to thank for Wensong Zhang, Zheng Li, and Fei Duan, from Alibaba Group, Stanley Wang from Intel for their early discussion of this work. We also thank for Intel China for their contribution of hardware for the performance evaluation. This work was in part supported by NSF grant CNS-1205338, the Introduction of Innovative R&D team program of Guangdong Province (NO.201001D0104726115), and Wayne State University Office of Vice President for Research. This material is based upon work supporting while serving at the National Science Foundation.

REFERENCES

[1] 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.

[2] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[3] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 157–168, Jun. 2009.

[4] K. Shen, A. Shriraman, S. Dwarkadas, X. Zhang, and Z. Chen, "Power containers: An os facility for fine-grained power and energy management on multicore servers," *SIGPLAN Not.*, vol. 48, no. 4, pp. 65–76, Mar. 2013.

[5] D. Shin, J. Kim, N. Chang, J. Choi, S. W. Chung, and E.-Y. Chung, "Energy-optimal dynamic thermal management for green computing," in *Proc. Int. Conf. Comput.-Aided Design*, 2009, pp. 652–657.

[6] J. Sun, R. Zheng, J. Velamala, Y. Cao, R. Lysecky, K. Shankar, and J. Roveda, "A self-tuning design methodology for power-efficient multi-core systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 1, pp. 4:1–4:24, Jan. 2013.

[7] I. Takouna, W. Dawoud, and C. Meinel, "Dynamic configuration of virtual machine for power-proportional resource provisioning," in *Proc. 2nd Int. Workshop Green Comput. Middleware*, 2011, pp. 4:1–4:6.

[8] T. Minartz, T. Ludwig, M. Knobloch, and B. Mohr, "Managing hardware power saving modes for high performance computing," in *Proc. Int. Green Comput. Conf. Workshops*, 2011, pp. 1–8.

[9] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. 3rd ACM Symp. Cloud Comput.*, 2012, pp. 7:1–7:13.

[10] H. Voigt, T. Kissinger, and W. Lehner, "Smix: Self-managing indexes for dynamic workloads," in *Proc. 25th Int. Conf. Sci. Statist. Database Manage.*, 2013, pp. 24:1–24:12.

[11] G. Metri, S. Srinivasaraghavan, W. Shi, and M. Brockmeyer, "Experimental analysis of application specific energy efficiency of data centers with heterogeneous servers," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, 2012, pp. 786–793.

[12] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.

[13] K. Kumar, K. Doshi, M. Dimitrov, and Y.-H. Lu, "Memory energy management for an enterprise decision support system," in *Proc. 17th IEEE/ACM Int. Symp. Low-Power Electron. Design*, 2011, pp. 277–282.

[14] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proc. 8th ACM Int. Conf. Autonomic Comput.*, 2011, pp. 31–40.

[15] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, "Robust and flexible power-proportional storage," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 217–228.

[16] W. Zheng, A. P. Centeno, F. Chong, and R. Bianchini, "Logstore: Toward energy-proportional storage servers," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2012, pp. 273–278.

[17] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," *SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 449–460, Jun. 2011.

[18] V. Lari, S. Muddasani, S. Boppu, F. Hannig, M. Schmid, and J. Teich, "Hierarchical power management for adaptive tightly-coupled processor arrays," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 1, pp. 2:1–2:25, Jan. 2013.

[19] G. Kornaros and D. Pnevmatikatos, "Hardware-assisted dynamic power and thermal management in multi-core SoCs," in *Proc. 21st Ed. Great Lakes Symp. Great Lakes Symp. VLSI*, 2011, pp. 115–120.

[20] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch, "Power management of online data-intensive services," *SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 319–330, 2011.

[21] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proc. Int. Symp. Low Power Electron. Design*, 2007, pp. 38–43.

[22] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram, "A case for guarded power gating for multi-core processors," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, 2011, pp. 291–300.

[23] T. F. System. (2014) [Online]. Available: http://tfs.taobao.org/

[24] Alexa. (2013) [Online]. Available: http://www.alexa.com/topsites

[25] Z. Xu, Y.-C. Tu, and X. Wang, "Pet: Reducing database energy cost via query optimization," *Proc. VLDB Endowment*, vol. 5, no. 12, pp. 1954–1957, Aug. 2012.

[26] Z. Xu, X. Wang, and Y.-C. Tu. (2013). Power-aware throughput control for database management systems. in *Proc. 10th Int. Conf. Auton. Comput.*, pp. 315–324. [Online]. Available: https://www.usenix.org/conference/icac13/technical-sessions/presentation/xu_zichen

[27] I. Psaroudakis, T. Kissinger, D. Porobic, T. Ilsche, E. Liarou, P. Tözün, A. Ailamaki, and W. Lehner, "Dynamic fine-grained scheduling for energy-efficient main-memory queries," in *Proc. 10th Int. Workshop Data Manage. New Hardware*, 2014, pp. 1:1–1:7.

[28] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis, "Towards energy-efficient database cluster design," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1684–1695, Jul. 2012.

[29] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile dram," in *Proc. 39th Annu. Int. Symp. Comput. Archit.*, 2012, pp. 37–48.

[30] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie, "Half-dram: A high-bandwidth and low-power dram architecture from the rethinking of fine-grained activation," in *Proc. 41st Annu. Int. Symp. Comput. Archit.*, 2014, pp. 349–360.

[31] C. Hu, C. Wu, W. Xiong, B. Wang, J. Wu, and M. Jiang, "On the design of green reconfigurable router toward energy efficient internet," *IEEE Commun. Mag.*, vol. 49, no. 6, pp. 83–87, Jun. 2011.

[32] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis, "Towards energy proportionality for large-scale latency-critical workloads," in *Proc. 41st Annu. Int. Symp. Comput. Archit.*, 2014, pp. 301–312.

[33] Y. Fu, A. Holler, and C. Lu. (2014, Jun.). Cloudpowercap: Integrating power budget and resource management across a virtualized server cluster. in *Proc. 11th Int. Conf. ton. Comput.*, pp. 221–231. [Online]. Available: https://www.usenix.org/conference/icac14/technical-sessions/presentation/fu

[34] C. Chen, M. Won, R. Stoleru, and G. Xie, "Energy-efficient fault-tolerant data storage and processing in mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 28–41, Jan. 2015.

[35] M. Kazandjieva, C. Shah, E. Cheslack-Postava, B. Mistree, and P. Levis, "System architecture support for green enterprise computing," in *Proc. 5th Int. Green Comput. Conf.*, Nov. 2014, pp. 1–10.

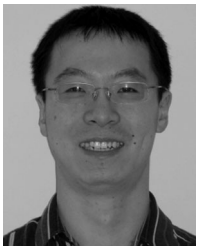
[36] A. Wierman, L. L. H. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems: Optimality and robustness," *Perform. Eval.*, vol. 69, no. 12, pp. 601–622, Dec. 2012.

[37] D. Wong and M. Annavaram, "Knightshift: Scaling the energy proportionality wall through server-level heterogeneity," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2012, pp. 119–130.

[38] Y. Liu, S. C. Draper, and N. S. Kim, "Sleepscale: Runtime joint speed scaling and sleep states management for power efficient data centers," *SIGARCH Comput. Archit. News*, vol. 42, no. 3, pp. 313–324, Jun. 2014.

[39] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 89–102, Oct. 2001.

Bing Luo received the BSc degree in information and computing science from the East China University of Science and Technology, Shanghai, China. He is currently working toward the PhD degree in computer science at Wayne State University, Detroit, MI. His research interests include operating systems, artificial intelligence, and energy efficient computing system. He received the ISCA 2013 Student's Travel Grants. He is a student member of the IEEE.



Shinan Wang received the PhD degree in computer science from Wayne State University, in 2014. His research interests include computer system power profiling and management. His research results have been published in Sustainable Computing: Informatics and Systems, ICEAC, WEED, and several other conferences and journals. He received the IPSN 2009 and IGCC 2013 Student's Travel Grants. He is also awarded with Wayne State University Summer Dissertation Fellowship.

Weisong Shi received the BE degree from Xidian University in 1995, and the PhD degree from the Chinese Academy of Sciences in 2000, both in computer engineering. He is a professor of computer science at Wayne State University, where he leads the Mobile and Internet Systems Laboratory. His research interests include parallel and internet computing, energy-efficient computer systems, mobile computing, and smart health. He has published more than 140 peer-reviewed journal and conference papers and has an H-index of 30. He is the chair of the IEEE CS Technical Committee on the Internet, and serves on the editorial board of *IEEE Internet Computing*, *Elsevier Sustainable Computing*, *Journal of Computer Science and Technology* (JCST) and *International Journal of Sensor Networks*. He received the National Outstanding PhD dissertation Award of China in 2002 and the National Science Foundation (NSF) Career Award in 2007, Wayne State University Career Development Chair Award in 2009, and the Best Paper Award of ICWE04, IEEE IPDPS05, HPCChina'12 and IEEE IISWC'12. He is a senior member of the IEEE and ACM, a member of the USENIX.

Yanfeng He received the ME degree from the Beijing University of Posts & Telecommunications in 2003. After employed by Potevio, Siemenz, and ChinaCache, he joined into Alibaba Group in 2009 and takes charge of the development of systems related to cloud computing, such as CDN, distributed storage, server customization with green computing, etc.