This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

Energy Efficient Scheduling and Management for Large-Scale Services Computing Systems

Ying Chen, *Student Member, IEEE*, Chuang Lin, *Senior Member, IEEE*, Jiwei Huang, *Member, IEEE*, Xudong Xiang, *Student Member, IEEE*, and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract—With the increasing popularity of services published online, energy consumption of services computing systems is growing dramatically. Besides Quality of Service (QoS), energy efficiency has become an important issue and drawn significant attention. However, energy efficient request scheduling and service management for large-scale services computing systems face challenges because of the high dynamics and unpredictability of request arrivals. In this paper, we jointly consider the conflicting metrics of performance, queue congestion and energy consumption. We propose a distributed online scheduling and management algorithm which does not require any priori statistical knowledge of request arrivals. Mathematical analysis is conducted which demonstrates that our algorithm can achieve arbitrary tradeoff between performance and energy efficiency. Numerical and real trace data based experiments are carried out to validate the effectiveness of our algorithm in optimizing energy efficiency while stabilizing the system.

Index Terms—services computing; request scheduling; service management; energy efficiency; QoS

1 Introduction

Services represent a type of relationships-based interactions between service providers and service consumers to achieve a certain business goal or solution objective [1]. Services Computing, as an emerging cross-discipline covering various aspects of business and IT services, is to create, operate, manage and optimize these services in a well-defined architecture for higher flexibility facing future business dynamics. With the increasing presence and adoption of services on the World Wide Web, there is a growing popularity of third-party commodity clusters and cloud environments providing services with different functionalities. As the number of functionally similar services available on the Internet increases rapidly, Quality of Service (QoS) is employed for describing non-functional characteristics of services [2]. Therefore, upon arrival of service requests, how to select the optimal execution plan that maximizes the end-to-end QoS has drawn significant attention, and there have been many research efforts with regard to such problem in services computing systems [3], [4].

The development and management of services computing systems has been focused on the improvement of performance, such as response time, throughput, reliability, etc. Recently, however, more and more attention has been paid to energy consumption, especially in large-scale computing infrastructures such as service systems, cloud systems and data centers [5], [6]. It is reported that the energy consumed by data centers in the US is more than 1.5 percent of the total energy consumption in the country, and the number is still increasing [7]. It is also estimated that the energy consumption of a Google search is 0.0003kWh on average, resulting in more than 32 million kWh energy consumed every year [8]. How to improve energy efficiency has become an important issue in service systems.

There are several approaches for reducing energy consumption in services computing systems: (1) the first approach is energy efficient request dispatch. It has been shown in [9] and [10] that, there is a significant opportunity promoting energy efficiency through effective request dispatch and allocation; (2) the second approach to reduce energy consumption is service management at service level, which monitors the workload and utilization information, and switches the service or its hosting virtual machine between active and idle states accordingly [11]; and (3) the third approach is power management at the hardware level, e.g., Dynamic Voltage and Frequency Scaling (DVFS). DVFS is one of the most popular

- *Ying Chen and Chuang Lin are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: chenying12@mails.tsinghua.edu.cn, chlin@tsinghua.edu.cn*
- *Jiwei Huang is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: huangjw05@gmail.com*
- *Xudong Xiang is with the Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing 100083, China. E-mail: xudong.xiang@csnet1.cs.tsinghua.edu.cn*
- *Xuemin (Sherman) Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

techniques for power management in computer systems [12]. In this paper, we jointly consider the three approaches to improve energy efficiency in services computing system, which is unexplored in previous literature.

Furthermore, existing research works generally promote QoS and energy efficiency based on assumption or prediction of request arrival or QoS properties [13], [14], [15]. In reality, the accuracy of such predictions can hardly be guaranteed. Because of the burstiness and fluctuations of request arrivals, it is extremely hard or even impossible to precisely predict the arrival statistics of service requests [11], [16]. Moreover, with the growing popularity of services on the Internet, traditional centralized optimization techniques such as combinatorial optimization and dynamic programming may not be applicable in large-scale service systems because of their high time-complexity.

To address these challenges, we introduce a distributed online energy efficient request scheduling and service management mechanism without requiring a priori knowledge of any statistical information of requests arrivals. In specific, we formulate the request scheduling and service management as an optimization problem, which can be widely applied to classical research problems, such as service selection. We develop a general model considering the decisions of request dispatch, service management and DVFS to improve energy efficiency from a systematical point of view. Based on Lyapunov optimization techniques, we propose a Distributed Online Scheduling and Management algorithm called *DOSM*. With an arbitrary parameter V to control the tradeoff between energy efficiency and queue length, the DOSM algorithm is proven to be $O(1/V)$-optimal with respect to the average energy efficiency while bounding the queue length by $O(V)$. Furthermore, we conduct both numerical experiments and real trace based simulations to demonstrate the effectiveness and efficiency of DOSM algorithm. Besides, sensitivity analysis is performed which can provide guidance for system management and optimization.

Our preliminary work was presented in the ICWS 2014 [17]. The main differences are as follows. (1) Our request scheduling and service management model is more generic than the conference version. The applicability of our model in services computing system is discussed in detail. Specifically, the relationship between our model and the traditional service selection is studied. (2) We combine dynamic voltage and frequency scaling which reduces the energy consumption at the hardware level to achieve a comprehensive optimization of energy efficiency in service systems. (3) The algorithm has been modified. In the worst case, the time complexity of the algorithm has been reduced from $O(n + 2^m)$ to $O(n + Nm)$, which is polynomial time. (4) We redo the experiments and conduct real trace based simulations to further validate the effectiveness of our model and algorithm in complex and real world situations. (5) Sensitivity analysis based on real trace is carried out which can find effective ways for system management and optimization.

The remainder of the paper is organized as follows. Section 2 introduces related work. In Section 3, we formulate the energy efficient request scheduling and service management problem. In Section 4, based on Lyapunov optimization techniques, we propose a distributed online algorithm. Optimality analysis of our algorithm is also provided. In Section 5, we conduct numerical experiments and real trace based simulations to verify its effectiveness. Sensitivity analysis is also performed. We conclude the paper in Section 6.

2 Related Work

With the increasing energy consumption associated with IT and service systems, energy efficiency has drawn significant attention in both the computing and mobile services [6], [9], [18]. There are several approaches to reduce energy consumption. In this paper, we consider three approaches, i.e., request dispatch, service management and DVFS [5], [10], [19].

Effective request dispatch is to make full use of services diversity, and distribute the requests to the appropriate services in order to meet the requirement and reduce energy consumption at the same time [20]. It has been shown that the energy efficiency can be improved significantly by effective request dispatch and allocation [9]. Gao et al. [21] proposed a flow optimization based framework for request dispatch and engineering, which monitored the request workload information and dynamically controlled the dispatch of user requests. Also, in our previous work [14], we proposed an energy efficient approach for service request dispatch, and proved that an effective request dispatch mechanism can significantly reduce energy consumption in large-scale service systems. Lin et al. [22] discussed environmental potential of workload dispatch and balancing based on the "receding horizon control" (RHC) algorithm, and studied how to introduce variants of RHC in the face of heterogeneity of delays, servers, and electricity prices. Adnan et al. [23] took advantage of geographical load balancing and combined the flexibility from the Service Level Agreements (SLAs) to differentiate among workloads for cost savings. They showed that significant cost savings can be achieved by dispatch of workload with future electricity price prediction. For our work, we do not require the prediction on electricity price, since the accuracy of prediction can hardly be guaranteed.

Service management is an effective approach to reduce energy consumption at service level. It controls the states of services on servers by switching the services (or the virtual machines that host them) between active and idle states according to their workload

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

and utilization [16]. It has been demonstrated that service management can improve system utilization and achieve the desired tradeoff between energy cost and performance. Zhou et al. [24] took advantage of service management and presented an optimization framework for service platforms, which was able to reduce energy and achieve performance guarantees. Beloglazov et al. [25] proposed service management and allocation algorithm for energy-efficient management of service systems. The algorithm could reduce power consumption while satisfying QoS constraints. Jang et al. [26] proposed techniques for reducing energy consumption by tuning the services with low workload into standby state.

Another way to reduce energy consumption is DVFS at the hardware level [5]. Equipped by modern hardware computer components, dynamic speed scaling is helpful for improving energy efficiency, which adjusts the CPU frequencies of the server [12]. Another popular technique, i.e., dynamic voltage scaling, is commonly used in conjunction with frequency scaling [5]. It has been shown in [12] that power consumption can be dramatically cut down by decreasing the CPU voltage and frequency simultaneously. Such combined technique, called DVFS, has been widely used and supported by most modern CPUs. For example, most Intel CPUs have been equipped with the techniques of SpeedStep and TurboBoost, adjusting CPU's voltage and speed to its utilization [27]. The other famous CPU manufacturer, AMD, has also equipped their CPUs with the similar mechanisms, such as PowerNow! [28]. Guerra et al. [29] used dynamic voltage scaling techniques and Vary-On Vary-Off configuration to improve energy efficiency. Wierman et al. [15] discussed how to reduce power consumption by DVFS method. They assumed the request arrival as a Poisson process and modeled the system by an M/GI/1 queue. In our work, we do not require any statistical information of requests arrivals.

Although the request dispatch, service management and DVFS approaches have been well studied separately by the research community, our work jointly considers the three approaches in services computing systems and present a comprehensive optimization framework of energy efficiency in service systems. We design efficient algorithms providing reference value for system management and optimization. Extended from our previous work [17], we develop a more comprehensive model to study energy efficiency. Our model is applicable to the traditional service selection which is one of the most popular problems in the services computing community. Different from previous works which focused on local optimization at each time period without considering the variations across time [30], we guarantee the average performance and cost over a long-time period.

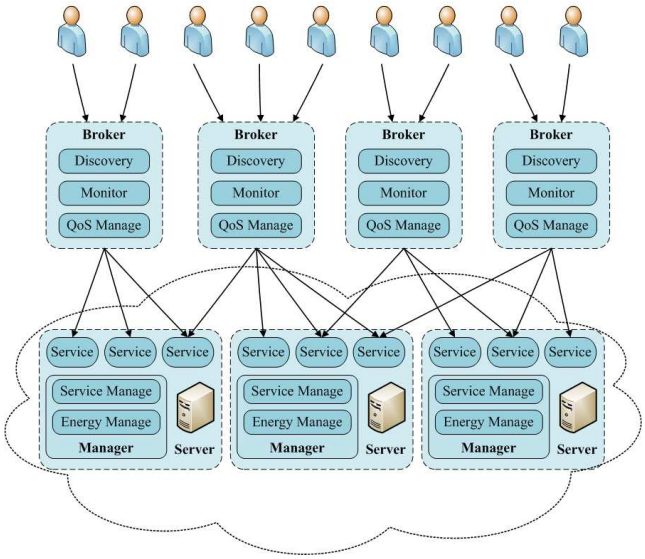

Fig. 1. Framework for energy efficient scheduling and management.

3 System Model

In this section, we formulate the request scheduling and service management problem and discuss the applicability of our formulation in services computing systems.

3.1 System Architecture

Fig. 1 shows a framework for energy efficient scheduling and management in a general services computing system. There are multiple services deployed on different servers, and many of them are duplicated for the following two reasons. First, some of the services with the same functionality are published by different service providers. Second, even one service provider may deploy multiple duplications of a service on different servers to guarantee the service availability to the users [31].

The brokers are responsible for service discovery, status monitor and QoS management. Upon arrival of user requests, the brokers search published services that can fulfill the functionalities and QoS requirements of these requests, and dispatch requests to the appropriate services [32]. There is a manager for each of the servers, managing the status of the services while determining the operational status of the servers (such as CPU voltage and server frequency using DVFS techniques).

3.2 Problem Formulation

Consider the services computing system with m categories of services denoted by a set I. Each category of services has candidates deployed on different servers and the set of all service candidates is denoted as C. The set of service candidates that are appropriate to accomplish service category $i \in I$ is expressed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

TABLE 1
Notations and Definitions.

Notation	Definition
I	set of services.
C	set of service candidates.
J	set of servers.
κ	service request.
$A_i(t)$	number of requests for service i that arrive in slot t.
$D_{ij}(t)$	number of requests dispatched to service candidate i on server j in slot t.
$y_{ij}(t)$	binary variable denoting the state of service candidate i on server j in slot t.
$u_j(t)$	running frequency of server j in slot t.
$QoS_\kappa^w(t)$	w-th QoS requirement of request κ in slot t.
$qos_\kappa^w(t)$	w-th QoS value of request κ in slot t.
$f(t)$	system profit in slot t.
$r_{ij}(t)$	reward of a request on service candidate i on server j in slot t.
$P_j(t)$	power consumption of server j in slot t.
$\varphi(t)$	price of unitary energy consumption in slot t.
τ	slot length.
$Q_{ij}(t)$	queue length of service candidate i on server j in slot t.

by C_i. Let $J = \{1, 2, \ldots, n\}$ denote the set of all physical servers in the services computing system. In order to make our approach more general, we assume the physical servers to be heterogeneous. The main notations in this paper are listed in Table 1.

3.2.1 Service Request Model

Each service request is characterized by a tuple $\kappa = \langle i, d_i, O_i \rangle$, where $i \in I$ denotes the category of service that the request belongs to, and d_i denotes the demand (i.e., job length or hardware resources) of such request. O_i is the set of servers where the services are deployed. K represents the set of all the requests.

We model the system as a discrete time-slotted system where the length of each time period τ can range from milliseconds to minutes. Let $A_i(t)$ denote the number of requests for service i arrived in time slot t.

We highlight the generality and practicability of our model that *it does not necessarily require a priori knowledge of any statistical information of* $A_i(t)$, which is generally unpredictable and difficult to obtain in practice.

3.2.2 Request Scheduling and Service Management

Let $B_{m\times n} = [b_{ij}]_{m\times n}$ represent the mapping matrix of services on servers, where $b_{ij} = 1$ represents that service i is deployed on server j, otherwise $b_{ij} = 0$. The problem of service deployment or placement is out of the scope of this paper, and the mapping matrix $B_{m\times n}$ is assumed to be predefined. In each time slot t, the service brokers make decisions to select the appropriate service candidates to serve the requests, and decide how many requests to be distributed to the services. Let $D_{ij}(t)$ denote the number of requests dispatched to service candidate i on server j. For those j with $b_{ij} = 0$, the variables $D_{ij}(t)$ are set to be zero.

Our approach also determines the state of service i (or its hosting virtual machine) on server j in each time slot t, and can switch it between active state and idle state in response to time-varying system workload, in order to arbitrate the performance-energy tradeoff. Let $y_{ij}(t)$ be the indicator variable of the state of service i on server j in slot t, where $y_{ij}(t) = 1$ means service i on server j is active, otherwise $y_{ij}(t) = 0$.

As the voltages of computer components are closely related to the server speed and dynamic voltage scaling is usually applied in conjunction with speed scaling [5], we let $u_j(t)$ be the DVFS decision variable, standing for the running frequency of server j in slot t. The set of all available frequencies for server j is represented by U_j.

In services computing systems, users usually have preferences and QoS requirements towards the services they are offered, and thus Service Level Agreement (SLA) should be carefully considered. Let W represent the set of indices referring to different QoS properties, including reliability, availability, integrity, etc. We use the superscript w to index QoS properties. Let $QoS_\kappa^w(t)$ represent the w-th QoS requirement of request κ in slot t. $qos_\kappa^w(t)$ stands for the w-th QoS value of request κ. We only consider the cases that larger QoS values represent better quality, since the other cases can be transformed by taking the opposite values. Thus it should be satisfied that $qos_\kappa^w(t) \geq QoS_\kappa^w(t), \forall \kappa \in K, \forall w \in W$.

We aim to maximize the system profit in the services computing system, which is a function of the decision variables $D_{ij}(t), y_{ij}(t)$ and $u_j(t)$, denoted by $f(D_{ij}(t), y_{ij}(t), u_j(t))$. Then the problem can be formulated as

$$\max_{D_{ij}(t), y_{ij}(t), u_j(t)} \quad f(D_{ij}(t), y_{ij}(t), u_j(t)); \tag{1}$$

subject to

$$\sum_{j\in J} D_{ij}(t) = A_i(t), \quad \forall i \in I, \forall t \in \mathbb{N}; \tag{2}$$

$$y_{ij}(t) \in \{0, 1\}, \forall i \in I, \forall j \in J, \forall t \in \mathbb{N}; \tag{3}$$

$$u_j(t) \in U_j, \forall j \in J, \forall t \in \mathbb{N}; \tag{4}$$

$$qos_\kappa^w(t) = qos_\kappa^w(D_{ij}(t), y_{ij}(t), u_j(t)) \geq QoS_\kappa^w(t), \forall \kappa \in K, \forall w \in W. \tag{5}$$

3.3 Model Applicability

Many service providers offer services that are functionally similar but differ in non-functional properties. When a request is submitted, how to choose the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

appropriate candidate service to provide the required functionality and satisfy the requirements has become an important problem. This issue, known as "service selection", has been widely studied by the services computing community [1]. Here, we discuss how service selection is related to our request scheduling and service management problem.

Consider a discrete time-slotted system and the total requests are classified into $|I|$ categories. Let $x_{ij}^{\kappa}(t)$ represent the service selection decision variable of request κ in time slot t, where $x_{ij}^{\kappa}(t) = 1$ means that service candidate i on server j is selected, otherwise $x_{ij}^{\kappa}(t) = 0$. The profit of request κ after service selection is denoted by $g(x_{ij}^{\kappa}(t))$ and the total requests' profit is $\sum_i \sum_j \sum_{\kappa} g(x_{ij}^{\kappa}(t))$. Then the service selection problem can be formulated as

$$\max_{x_{ij}^{\kappa}(t)} \sum_{i\in I} \sum_{j\in J} \sum_{\kappa\in K} g(x_{ij}^{\kappa}(t)); \tag{6}$$

subject to

$$x_{ij}^{\kappa}(t) \in \{0,1\}, \forall \kappa \in K, \forall i \in I, \forall j \in J, \forall t \in \mathbb{N}; \tag{7}$$

$$\sum_{j=1}^{n} x_{ij}^{\kappa}(t) = 1, \forall \kappa \in K, \forall i \in I, \forall t \in \mathbb{N}; \tag{8}$$

$$qos_{\kappa}^{w}(t) = qos_{\kappa}^{w}(x_{ij}^{\kappa}(t)) \geq QoS_{i}^{w}(t), \forall \kappa \in K, \forall w \in W. \tag{9}$$

We elucidate that service selection can be applicable in our model. When considering service selection without service management and DVFS, the objective is deduced to maximize the system profit $f(D_{ij}(t))$ with single argument $D_{ij}(t)$. Let $h(D_{ij}(t))$ denote the profit obtained by each service i on server j in time slot t, so we have $f(D_{ij}(t)) = \sum_i \sum_j h(D_{ij}(t))$. We present Theorem 1 which elucidates the connection of our model with service selection.

THEOREM 1: Our request scheduling model is equivalent to service selection if the functions $h(\cdot)$ and $g(\cdot)$ satisfy (10),

$$h(kx) = kg(x). \tag{10}$$

Therefore, it can be seen that our request scheduling and service management model is general and the traditional service selection can be regarded as part of the considered problem. Interested readers are referred to our detailed proof for Theorem 1 in the appendix.

3.4 Model Specification

In this subsection, we refine our request scheduling and service management problem in detail and develop a more specific model to describe the system properties. We try to maximize energy efficiency while bounding the queue length.

The reward of services is a critical concern in services computing systems. Let $r_{ij}(t)$ denote the reward obtained by a request served by service i on server j in time slot t, and thus we obtain $r_i(t) = \sum_j r_{ij}(t) D_{ij}(t)$ as the total reward of service i.

According to [5], the power consumption of a server consists of two parts, static power and dynamic power. For server j, the power consumed in time slot t can be expressed by (11), where P_{s_j} represents the static power, k and α ($\alpha \geq 1$) are constant, $\rho_j \in [0,1]$ and u_j represent the utilization and the frequency of the server, respectively.

$$P_j(t) = P_{s_j} + k\rho_j(t) u_j^{\alpha}(t). \tag{11}$$

Define $s_{ij} \in [0,1]$ as the ratio of the processing requirement from service i to the the total processing capacity of its hosting server j. It should be satisfied that $\sum_i s_{ij} = 1, \forall j \in J$. The utilization of server j in time slot t can be expressed as $\rho_j(t) = \sum_i s_{ij} y_{ij}(t)$.

Besides physical servers, other facilities such as cooling systems also consume a fraction of power in large-scale service systems. The ratio of power consumed by the entire service system to power consumed by servers is denoted by Power Usage Efficiency (PUE), which is commonly a constant in a certain large-scale computing system [33]. So the power consumption of the whole services computing system in time slot t can be expressed as $\text{PUE} \cdot \sum_j P_j(t)$. Define $\varphi(t)$ as the price of unitary energy consumption in time slot t, and the energy cost of the whole service system in slot t can be approximated as $\varphi(t) \cdot \text{PUE} \sum_j P_j(t)\tau$. The profit of the service system in slot t is expressed as

$$f(t) = \sum_i \sum_j r_{ij}(t) D_{ij}(t) - \varphi(t) \cdot \text{PUE} \sum_j P_j(t)\tau. \tag{12}$$

Unlike previous work focusing on the instantaneous system profit, we study the average property which can guarantee long-term profit over a large time horizon. The average profit of the services computing system over time slots $t = \{0, 1, 2, \ldots, T-1\}$ is expressed as below.

$$f = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\{f(t)\}.$$

Queueing delay is an important metric and can influence the degree of customer satisfaction. According to *Little's Law*, the queueing delay of a service is in proportion to the number of requests waiting in the queue, i.e., the queue length. Therefore, we aim to reduce the queue length and achieve low system congestion. Optimization of queue length has been considered in many previous works [11], [16], [19]. In this paper, we use $Q_{ij}(t)$ to represent the queue length of service i on server j in time slot t.

Let u_j^0 represent the base frequency of server j, which is assumed to be fixed for each server j. l_{ij} is defined as the number of requests that can be served in each slot by service i on server j running at base

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

frequency u_j^0, and l_{ij} is a constant. In each slot t, for server j running in frequency $u_j(t)$, assume the number of requests it can serve for service i can be calculated by $l_{ij} \cdot \frac{u_j(t)}{u_j^0}$. The queue length is updated as

$$Q_{ij}(t+1) = \max\left[Q_{ij}(t) - l_{ij}y_{ij}(t)\frac{u_j(t)}{u_j^0}, 0\right] + D_{ij}(t). \tag{13}$$

To reduce queueing delay, we try to push the queues under low congestion states and bound the average queue length q_{ij}. Let ξ be the bound of queue length, then

$$q_{ij} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{E}\{Q_{ij}(t)\} \le \xi, \forall i \in I, \forall j \in J, \exists\, \xi \in \mathbb{R}^+. \tag{14}$$

The optimization framework can be formulated as

$$\max_{D_{ij}(t), y_{ij}(t), u_j(t)} \quad \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{E}\{f(t)\}; \tag{15}$$

subject to (2), (3), (4), (5), (14).

Solving the problem offline is problematic since it involves unpredictable future information, e.g., service request arrivals. Besides, as the scale of the service system increases, it is computationally complex and challenging using centralized solutions. Therefore, we propose a distributed and online algorithm which can solve the problem efficiently and effectively.

4 Optimizing Energy Efficiency

By taking advantage of Lyapunov optimization techniques [34], we propose a Distributed Online Scheduling and Management algorithm called *DOSM*, which makes control decisions in each time slot to optimize the long-term average system profit.

4.1 Problem Transformation

Let $\Theta(t) = (Q_{ij}(t))$ denote the queue matrix of services on servers. We define the Lyapunov function as

$$L(\Theta(t)) = \frac{1}{2}\sum_{i\in I}\sum_{j\in J}Q_{ij}^2(t), \tag{16}$$

where $L(\Theta(t))$ reflects the queue congestion state of the services computing system at the beginning of time slot t. To maintain system stability and reduce queueing delay, we try to reduce $L(\Theta(t))$ and push the services computing system towards a low congestion state.

We define the *conditional Lyapunov drift* as follows.

$$\Delta(\Theta(t)) = \mathbf{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\}. \tag{17}$$

Minimizing the drift $\Delta(\Theta(t))$ alone would tend to push the queues towards a low congestion state and reduce the queue length, but the resulting average power expenditure might be unnecessarily large, which would incur a large cost and reduce the total profit [11]. Thus there exists tradeoff between these factors [34]. Following the Lyapunov framework, we define the *drift minus profit* expressed as $\Delta(\Theta(t)) - V\mathbf{E}\{f(t)|\Theta(t)\}$. The parameter $V > 0$ represents the tradeoff between queueing delay and system profit, and can be regarded as a weight placed on the profit. It can be determined by service providers or users according to their requirements in real applications.

In Theorem 2, we show that the *drift minus profit* can be upper bounded.

THEOREM 2: (Bounding Drift Minus Profit) In each slot t, under any algorithm, for all possible values of $\Theta(t)$ and any parameter value of V, if there exits a peak value A_i^{max} that upper bounds the number of requests arrived in each slot, the *drift minus profit* can be upper bounded by

$$\begin{aligned}&\Delta(\Theta(t)) - V\mathbf{E}\{f(t)|\Theta(t)\}\\ \le& B - \sum_i\sum_j \mathbf{E}\{Vr_{ij}(t)D_{ij}(t) - Q_{ij}(t)D_{ij}(t)|\Theta(t)\}\\ &- \sum_j \mathbf{E}\left\{\left(\begin{array}{l}\sum_i Q_{ij}(t)l_{ij}y_{ij}(t)\frac{u_j(t)}{u_j^0}\\ -V\varphi(t)\mathrm{PUE}P_j(t)\tau\end{array}\right)\middle|\,\Theta(t)\right\},\end{aligned} \tag{18}$$

where $B = \frac{1}{2}[\sum_i\sum_j(l_{ij}\frac{\hat{u}_j}{u_j^0})^2 + \sum_i(A_i^{max})^2]$ is a constant.

The proof of Theorem 2 is provided in the appendix.

4.2 Distributed Online Algorithm for Energy Efficient request scheduling and Service management

We aim to minimize the upper bound of drift minus profit, as shown by the right-hand-side (R.H.S.) of (18). We propose an online algorithm DOSM, which can decompose the problem into subproblems, and solve the problem in a distributed way. Furthermore, it can be proven that DOSM can approach the optimal profit while still bounding the queue length in Section 4.3.

In each slot t, DOSM observes the current queue length $\Theta(t)$ and makes the following control decisions to minimize the upper bound of *drift minus profit*: (1) request dispatch, which is to decide how many requests to dispatch to each service candidate, and (2) service management and DVFS, which switch services between active and idle states as well as adjust the server frequencies. After the decisions are made, the queue length evolves as (13).

(1) *Request Dispatch*. In each slot t, DOSM performs request dispatch to minimize the second term of the R.H.S of (18), which is equivalent to maximize the opposite of it. Furthermore, the dispatch decisions of requests for different categories of services are independent from each other, and the subproblem can

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

be reduced to the following problem for each $i \in I$, which can be solved concurrently.

$$\max_{D_{ij}(t)} \sum_j (Vr_{ij}(t) - Q_{ij}(t))D_{ij}(t); \tag{19}$$

subject to (2), (5).

(19) can be viewed as a generalized max-weight problem, where in each time slot t the number of requests dispatched to service is weighted by $Vr_{ij}(t) - Q_{ij}(t)$. User preferences and QoS requirements are also considered, and let $O'_i(t)$ be the set of candidates that satisfy users requirements. $O'_i(t)$ can be obtained by comparing each candidate's QoS values with user's QoS requirements. We call the candidates that belong to $O'_i(t)$ the *potential candidates*. So the optimal solution is to dispatch all the requests to the one with the maximum value of $Vr_{ij}(t) - Q_{ij}(t)$ in O'_i, which is given by

$$D_{ij}(t) = \begin{cases} A_i(t), & j = j_i^*; \\ 0, & otherwise, \end{cases} \tag{20}$$

where $j_i^* = \text{argmax}_{j \in O'_i(t)}(Vr_{ij}(t) - Q_{ij}(t))$ denotes the index of the server with the maximum *profit minus queue*.

The result elucidates that the optimal policy is to distribute all the requests to a single candidate service i on server j_i^*, which indicates that only one of the candidate services is selected to handle all the requests. Therefore, at service layer viewpoint, our scheduling problem is equivalent to service selection, which also proves the applicability of our model.

(2) *Service Management and DVFS*. DOSM makes the service management decision $y_{ij}(t)$ and DVFS decision $u_j(t)$ in each slot t in order to minimize the third term of the R.H.S. of (18). Similarly, this can be transformed to maximizing its opposite. The variables $y_{ij}(t)$ and $u_j(t)$ are independent among different servers, and this subproblem can be reduced to solving problem (21) concurrently for each server $j \in J$.

$$\max_{y_{ij}(t), u_j(t)} \sum_i Q_{ij}(t) l_{ij} y_{ij}(t) \frac{u_j(t)}{u_j^0} - V\varphi(t)\text{PUE}P_j(t)\tau; \tag{21}$$

subject to (3) and (4).

Problem (21) involves integer variables $y_{ij}(t)$ and it is a generalized mixed integer programming (MIP) problem. In the following, we explore the characteristics of the problem and solve it in polynomial time, which is efficient and can guarantee to find the optimal solutions.

For each given $u_j(t) \in U_j$, the original objective of (21) can be rewritten as

$$Obj = \sum_i (Q_{ij}(t) l_{ij} \frac{u_j(t)}{u_j^0} - V\varphi(t)\text{PUE}\tau k u_j^\alpha(t) s_{ij}) y_{ij}(t) - V\varphi(t)\text{PUE}\tau P_{sj}.$$

Algorithm 1 Distributed Online Scheduling and Management (DOSM)

1: In the beginning of each time slot t, observe the current queue length $Q_{ij}(t)$.
2: Solve the Max-Weight problem (19) for each category of service $i \in I$ in parallel
3: Solve the MIP problem (21) for each server $j \in J$ in parallel
4: $t = t + 1$
5: Repeat steps 1-4

Algorithm 2 Max-Weight Problem Solving for Service i

1: **for all** Potential candidates **do**
2: Calculate $Vr_{ij}(t) - Q_{ij}(t)$
3: Search for the candidate on server j^* with the maximum value of $Vr_{ij}(t) - Q_{ij}(t)$
4: Set $D_{ij}(t)$ according to (20)

Algorithm 3 MIP Problem Solving for Server j

1: $maxObj = -\infty$
2: **for all** $u_j(t) \in U_j$ **do**
3: **for all** candidates i on the server **do**
4: **if** $\gamma_{ij}(t) > 0$ **then**
5: $y_{ij}(t) = 1$
6: **else**
7: $y_{ij}(t) = 0$
8: **if** $Obj > maxObj$ **then**
9: $maxObj = Obj$
10: $maxu_j = u_j(t)$
11: **for all** candidates i on the server **do**
12: $maxy_{ij}(t) = y_{ij}(t)$
13: $u_j(t) = maxu_j$
14: **for all** candidates i on the server **do**
15: $y_{ij}(t) = maxy_{ij}(t)$

Let $\gamma_{ij}(t) = Q_{ij}(t) l_{ij} \frac{u_j(t)}{u_j^0} - V\varphi(t)\text{PUE}\tau k u_j^\alpha(t) s_{ij}$. Considering that the expression of $-V\varphi(t)\text{PUE}\tau P_{sj}$ is a constant to the decision variables, so for each given $u_j(t) \in U_j$, (21) can be reduced to solving (22) for each server $j \in J$.

$$\max_{y_{ij}(t)} \sum_i \gamma_{ij}(t) y_{ij}(t); \tag{22}$$

subject to (3).

(22) is a binary integer programming (BIP) problem. The decision variables $y_{ij}(t)$ are set to 1 when $\gamma_{ij}(t) > 0$, and 0 otherwise. So (21) can be solved by enumerating each $u_j(t) \in U_j$ and comparing each optimal result of (22).

The detailed algorithm is shown in Algorithm 1-3.

The time complexity of our DOSM algorithm for the worst case can be divided into the following two parts: (1) request dispatch, and (2) service management and DVFS. In the request dispatch part,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

since our algorithm can solve it concurrently for each service $i \in I$, and the max-weight problem can be solved by exploring every value $(Vr_{ij}(t) - Q_{ij}(t))$ once, the time complexity of this part is $O(n)$. For solving the MIP problem in the service management and DVFS part, similarly, our DOSM algorithm can solve it concurrently. Let N be the maximum number of possible solutions of $u_j(t)$ for all $j \in J$. For each given $u_j(t)$, it needs $O(m)$ operations to solve the reduced BIP problem, so the time complexity of this part is $O(Nm)$. The overall time complexity of our DOSM algorithm is $O(n + Nm)$, which is polynomial time.

4.3 Optimality Analysis

We show that DOSM can also obtain arbitrary tradeoffs between system profit and queue length, approaching the optimal profit while still bounding the queue length. Note that there exists upper bound $\hat{f}$ and lower bound $\check{f}$ of the objective f under the assumption that the number of requests arrived in each slot can be bounded by A_i^{max}. We prove that the average system profit and queue length under our DOSM algorithm are bounded as in Theorem 3.

THEOREM 3: If there exists ε satisfying $\lambda + \varepsilon \in \Lambda$, where Λ is the capacity region of the system, then under the DOSM algorithm, for any value of the parameter V, the average queue length is bounded by (23).

$$\begin{aligned} \bar{Q} &= \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i} \sum_{j} \mathbf{E}\{Q_{ij}(t)\} \\ &\leq \frac{B + V(\hat{f} - \check{f})}{\varepsilon}. \end{aligned} \tag{23}$$

Furthermore, the average system profit can be bounded by (24), which indicates the profit derived by our algorithm can approach the optimal value by increasing the parameter V. Here B is the constant defined in Theorem 2.

$$f^{DOSM} \geq f^* - \frac{B}{V}. \tag{24}$$

Theorem 3 indicates a $[O(1/V), O(V)]$ tradeoff between the average system profit and queue length under the DOSM algorithm. In addition, it can reach to the optimal system profit by increasing the parameter V. Although this would also increase the average queue length, the queue length q_{ij} of each service can still be bounded, since letting $\xi = \frac{B+V(\hat{f}-\check{f})}{\varepsilon}$, (14) can be satisfied. In practice, service operators can choose the parameter V according to actual demands and make the desired choice. The detailed proof of Theorem 3 can be found in the appendix.

5 Evaluation

In this section, we conduct numerical experiments and real trace based simulation to validate the effectiveness of our DOSM algorithm. We also present sensitivity analysis to help identify the bottleneck of the services computing system and provide guidance for system optimization.

5.1 Numerical Experiments

5.1.1 Experiment Setup

In the numerical experiments, there are $n = 400$ servers and $m = 400$ categories of services in the system. Each service has 10 candidates that are functionally similar but differ in QoS properties. Each server holds 10 candidates. The reward of each request is generated randomly between 0.1 and 0.2. Two QoS properties are considered, i.e., reliability and availability. The reliability and availability values of each candidate are generated randomly between 0.4 and 1. Users' reliability and availability requirements are randomly set between 0.5 and 0.9.

The slot length τ is set as 300 seconds. We will discuss the details on τ later. The service rate of each service $i \in I$, i.e., the number of requests that can be served in each time slot is set randomly between 60 and 300. The arrival rate λ_i is set to be the service rate multiplied by a factor $\beta \leq 1$. The requests arriving processes are generated according to Poisson distribution [15]. Note that our approach is more general and is able to deal with any type of requests arrivals.

We use power-measuring equipments to collect power consumption data from servers with Intel Nehalem Bloomfield processor. The average static power consumption is $\bar{p_s} = 140$ watt and the coefficient k is set to be 100 based on the data. The parameter α is set to be 3 according to [5]. For each server $j \in J$, the base frequency is 133MHz, and there are four multipliers, i.e., 0, 12, 20, 22. PUE is set to be 1.2 based on [24].

5.1.2 Parameter Analysis

1) Impact of tradeoff parameter V.

Fig. 2 shows the average system profit and queue length with different values of V. It can be seen that as V increases, the average system profit improves under our DOSM algorithm, and tends to the optimum when V becomes sufficiently large. However, the improvement of profit starts to diminish with far more increase of V. This is consistent with (24) in Theorem 3, which shows that our DOSM algorithm can approach the maximum profit with a gap of $O(1/V)$. The queue length also grows with the increase of V. This is consistent with (23) in Theorem 3 that the queue length is upper bounded by $O(V)$. Along with system profit, it shows the tradeoff between average system profit and queue congestion.

2) Impact of unitary energy price parameter φ.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

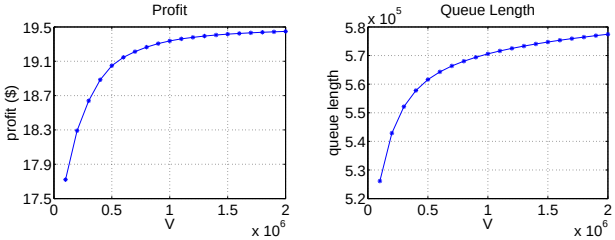

Fig. 2. Average system profit and queue length with different V.

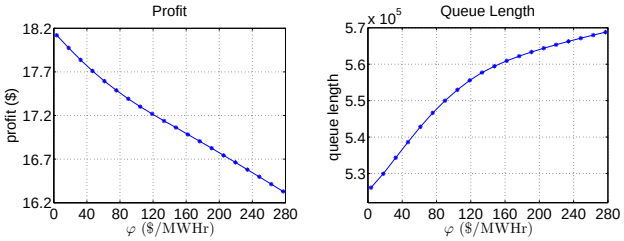

Fig. 3. Average system profit and queue length with different φ.

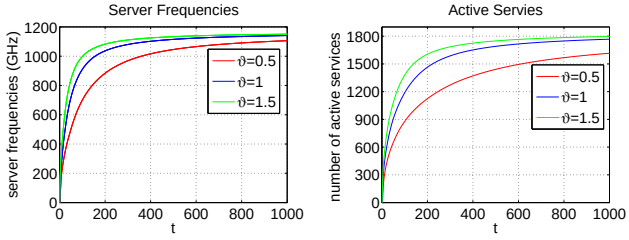

Fig. 4. Average server frequencies and number of active services under different ϑ.

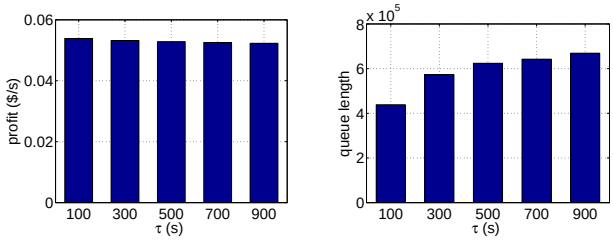

Fig. 5. Average profit and queue length under different τ.

We evaluate the impact of parameter φ on system profit and queue length. Fig. 3 shows the average system profit and queue length with different values of φ. It can be seen that the system profit decreases as φ increases, due to the rising of energy cost. The queue length also increases for the same reason. Our DOSM algorithm trys to reduce the energy consumption by adjusting the servers to run at low speed and switching some services to idle state, thus increasing the total queue length. However, the rise of queue length will start to diminish with far more increase of φ, which shows that the queue length will stabilize gradually.

3) Impact of arrival rate λ_i.

To evaluate the impact of arrival rate on the system, we scale the arrival rate up or down to $\vartheta \cdot \lambda_i$. Three experiments with ϑ= 0.5, 1 and 1.5 are discussed. The experiment is carried out for 1,000 time slots for each setting. Fig. 4 shows the average sum of all servers' frequencies and number of active services under different ϑ. As it can be seen, both servers speed and number of active services increase when ϑ becomes larger. This shows that our DOSM algorithm can adapt to different arrival rates and adjust the decisions accordingly. Besides, both the servers speed and number of active services will become stable as time goes by.

4) Impact of slot length τ.

To evaluate the impact of slot length τ on average system profit (in second) and queue length, we vary τ from 100 seconds to 900 seconds. The experiment is carried out for 5,000 slots for each setting. Fig. 5 shows that when τ becomes larger, the average profit in each second drops slightly, and the queue length increases. This is because a larger τ makes it hard to adapt to the system dynamics as the system variables would vary during a long time slot. So a smaller τ can help to adapt to changes and achieve better results. However, when τ is too small, it would lead to high overheads of estimating or obtaining the system variables. One typical way is to set τ corresponding to the price updating frequencies in electricity markets [11], [16].

5.2 Simulation Experiments Based on Real Trace Data

5.2.1 Simulation Setup

Workload Data: In the simulation experiments, we use two real world workload traces to verify our approach under different services computing contexts.

The first workload trace is from LHC Computing Grid (LCG). The project is a global collaboration of more than 170 computing centres in 40 countries, and provides SaaS services for data storage and analysis [35]. The LCG data was provided by the e-Science Group of HEP at Imperial College London. The time period of the whole data lasts for 11 days from Nov 20th to 30th, 2005. The submission time, duration time and resource demand of the service requests can be found in the trace [36]. We classify the requests into 20 categories of services according to their duration time and resource demand. Each service has 50 candidates and the whole candidates are deployed on 200 servers in the system.

The second workload trace is from LPC of Blaise Pascal University of Clermont-Ferrand. LPC is part of the EGEE project (Enabling Grids for E-science in Europe), which develops an infrastructure and provides IaaS services to users [37]. We choose 11 days of data from the LPC trace, in accordance with LCG trace. The requests are also classified into 20 categories of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

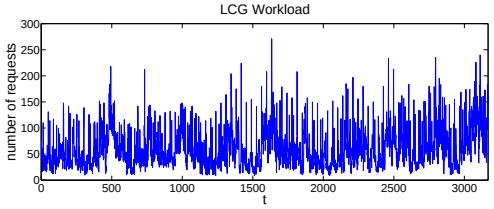

Fig. 6. LCG Workload trace.

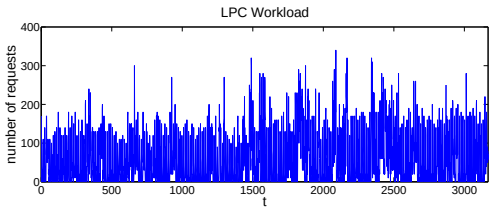

Fig. 7. LPC Workload trace.

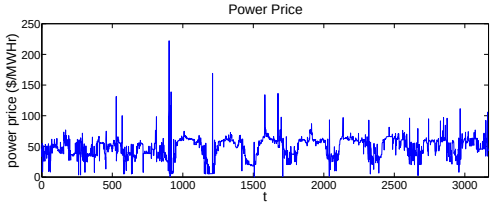

Fig. 8. Power price trace.

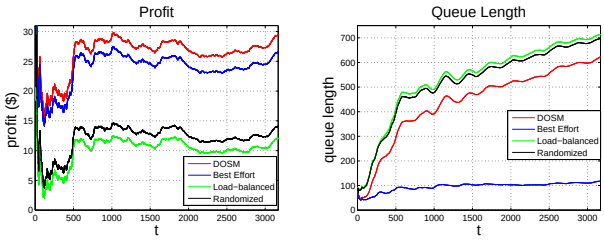

Fig. 9. Average system profit and queue length of the LCG system under different algorithms.

services according to the duration time and resource demand.

The period τ of each time slot t is set to be 300 seconds corresponding to the power price data updating frequency, which will be introduced later. Thus, there are 3,168 time slots in our simulation. Based on the submission time logs, we can obtain the number of requests for each service that arrived in each time slot t. Fig. 6 shows the number of requests that arrived in the LCG system over time slots. It can be seen that the workload fluctuates over time. Fig. 7 shows the workload trace of LPC system (scaled up by a factor 10). The service rates are set inversely proportional to the duration times.

Price Data: We collect the real world locational marginal price (LMP, in unit of $/MWHr) from the website of New York independent system operator (ISO) [38], which publishes the updated power price data every 300 seconds. We choose 3,168 price data in accordance to the workload data. Fig. 8 shows the power price trace over the 3,168 time slots. It can be seen that the price fluctuations periodically and the time of each period is about 250 time slots, roughly equal to one day. The reward data are set to be the same as that in the numerical experiments.

QoS Data: We adopt the updated QWS Dataset which includes 2,507 real Web services. The QoS parameters of these services were measured using the Web Service Broker (WSB) framework over a six-day period [39]. We choose 5 QoS properties from the dataset, including availability, successability, reliability, compliance and best practices. For all these QoS properties, a higher value stands for a better service. We randomly select 1,000 services from the dataset. Users' QoS preferences and requirements are generated based on the value ranges of the corresponding QoS property in the dataset.

5.2.2 Comparative Analysis

Based on the trace data, we compare our DOSM algorithm with three other algorithms: (1) *Best Effort* algorithm, where the server runs at the maximum speed and the service is switched active as long as the corresponding queue is not empty, while we use the same method as our DOSM algorithm for request dispatch; (2) *Load-balanced* algorithm, where the requests are dispatched to each service based on the serving capacity, while the service management and DVFS are the same as our DOSM algorithm; and (3) *Randomized* algorithm, where the requests are dispatched to each service randomly, while the rest is the same as DOSM algorithm.

Fig. 9 shows the average system profit and queue length of the LCG system under different algorithms. It can be seen that our DOSM algorithm achieves the highest profit among the four algorithms, which shows the effectiveness of DOSM. The Best Effort algorithm achieves higher profit than the Load-balanced and Randomized algorithms, since it can serve more requests in each time slot, thus reduce energy cost. The queue length of the Best Effort algorithm is the lowest, since the the actual serving rate of the Best Effort algorithm is the largest. The queue length of our DOSM algorithm is smaller than those of Load-balanced and Randomized algorithms. Together with system profit, it is shown that DOSM algorithm outperforms the Load-balanced and Randomized algorithms in both system profit and queue congestion, and DOSM achieves the highest system profit with a little sacrifice of queue congestion.

Fig. 10 shows the average system profit and queue length of the LPC system under different algorithms. It can be seen that the profit under DOSM algorithm is the highest among all algorithms. The queue length under DOSM algorithm is slightly larger than that of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

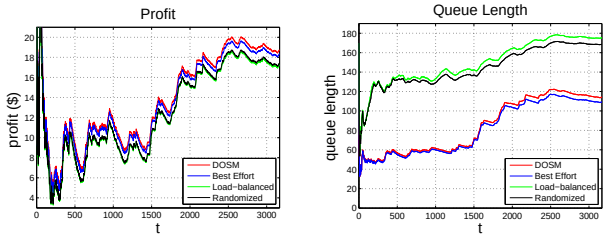

Fig. 10. Average system profit and queue length of the LPC system under different algorithms.

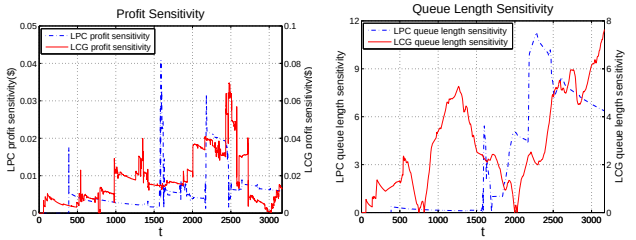

Fig. 11. Profit sensitivity and queue length sensitivity with V in the LCG and LPC systems.

TABLE 2
Execution Time of Different Algorithms

System	Algorithms Execution Time (s)			
	DOSM	Best Effort	Load-balanced	Randomized
LCG	0.953	0.907	0.947	0.949
LPC	0.955	0.901	0.946	0.948

the Best Effort algorithm, but smaller than those of Load-balanced and Randomized algorithms. Similar results have also been obtained for the LCG system.

Table 2 gives the execution times (in second) of the four algorithms in the LCG and LPC systems. For each experiment, we run 20 times to calculate the average result. It can be seen that the execution time differences among our DOSM algorithm, Load-balanced and Randomized algorithms are small. The execution time of the Best Effort algorithm is slightly smaller than the other three algorithms.

5.3 Sensitivity Analysis

Sensitivity analysis is often performed in the process of system optimization, which can help to find bottlenecks and provide guidance for system design and management [40]. We conduct parametric sensitivity analysis and discuss the sensitivity of system profit and queue length based on the real trace data presented in Section 5.2.

Let F be the objective function and η be one of the parameters. In practise, the sensitivity can be obtained by the definition of partial differential coefficient described in (25).

$$S_\eta(F) = \left| \frac{F(\eta + \Delta\eta) - F(\eta)}{\Delta\eta} \right|. \quad (25)$$

Fig. 11 shows system profit sensitivity and queue length sensitivity of the LCG and LPC systems in terms of V over the 3,168 time slots. The parameter V is set to be 60 and ΔV is set as $0.001 \times V = 0.06$. It can be seen that the profit sensitivities of both systems fluctuate over time. The profit sensitivity of the LCG system reaches to the maximum value at $t_1 = 2{,}500$, i.e., the profit can reach to the maximum gain when changing V at time slot t_1, which provides guidance for profit optimization. The queue length sensitivity of the LCG system increases sustainably during the last 200 time slots. The profit sensitivity of the LPC system

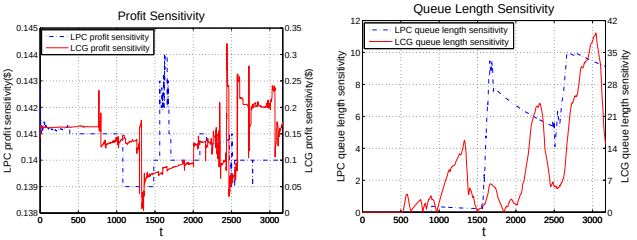

Fig. 12. Profit sensitivity and queue length sensitivity with φ in the LCG and LPC systems.

reaches to the maximum value at the 1,600 slot and the queue length sensitivity of the LPC system reaches to the highest value at the 2,300 slot.

Fig. 12 shows system profit sensitivity and queue length sensitivity of the LCG and LPC systems with φ over time slots. The parameter φ is set to be the same value as the trace in Section 5.2 and $\Delta\varphi = 0.3\$/MWHr$. It can be seen that the profit sensitivity of the LCG system reaches to the maximum value at $t_2 = 2490$, which indicates that the profit optimization by changing φ is the most effective at time slot t_2. The queue length sensitivity of the LCG system becomes the maximum at $t_3 = 3050$. It can be seen that t_3 is a good point to adjust φ to optimize both system profit and queue length of the LCG system. The profit sensitivity of the LPC system reaches to the maximum value at $t_4 = 1700$. The queue length sensitivity of the LPC system is also large at t_4. Thus t_4 is a good point to adjust φ to optimize both system profit and queue length of the LPC system.

6 Conclusion

In this paper, we study energy efficient scheduling and management for large-scale services computing systems. We jointly consider the metrics of energy efficiency, performance and queue congestion. We combine the three approaches of request dispatch, service management and DVFS to improve energy efficiency from a systematical point of view. A distributed online algorithm based on Lyapunov optimization techniques is proposed, which can achieve near optimal system profit while bounding the queue length. Our algorithm does not require any arrival statistics or any prediction on future information. Mathematical

analysis is provided to evaluate the effectiveness of our algorithm, which is further validated by both numerical experiments and real trace based simulation.

ACKNOWLEDGMENTS

This work is supported by the National Grand Fundamental Research 973 Program of China (No. 2010CB328105, No. 2013CB329105), the National Natural Science Foundation of China (No. 61020106002), and Tsinghua University Initiative Scientific Research Program (No. 20121087999).

REFERENCES

[1] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer and Tsinghua University Press, 2007.

[2] P. Xiong, Y. Fan, and M. Zhou, "QoS-aware web service configuration," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 4, pp. 888–895, 2008.

[3] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria workflow tasks allocation and scheduling in cloud computing environments," in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, June 2012, pp. 638–645.

[4] T. He, S. Chen, H. Kim, L. Tong, and K.-W. Lee, "Scheduling parallel tasks onto opportunistically available cloud resources," in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, June 2012, pp. 180–187.

[5] A. Beloglazov, R. Buyya, Y. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[6] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "Energy-aware autonomic resource allocation in multitier virtualized environments," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 2–19, 2012.

[7] W.-c. Feng, X. Feng, and R. Ge, "Green supercomputing comes of age," *IT professional*, vol. 10, no. 1, pp. 17–23, 2008.

[8] J. G. Koomey, "Estimating regional power consumption by servers: A technical note," *Lawrence Berkeley National Laboratory, Oakland, CA*, 2007.

[9] C. Wang, M. de Groot, and P. Marendy, "A service-oriented system for optimizing residential energy use," in *2009 IEEE 7th International Conference on Web Services (ICWS '09)*, 2009, pp. 735–742.

[10] P. Bartalos and M. Blake, "Green web services: Modeling and estimating power consumption of web services," in *2012 IEEE 19th International Conference on Web Services (ICWS '12)*, Jun. 2012, pp. 178–185.

[11] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," in *2013 IEEE INFOCOM*, 2013, pp. 872–880.

[12] L. Minas and B. Ellison, *Energy efficiency for information technology: How to reduce power consumption in servers and data centers*. Intel Press, 2009.

[13] Z. Chen, C. Liang-Tien, B. Silverajan, and L. Bu-Sung, "Ux-an architecture providing QoS-aware and federated support for UDDI," in *2003 IEEE 1st International Conference on Web Services (ICWS '03)*, 2003.

[14] J. Huang and C. Lin, "Improving energy efficiency in web services: An agent-based approach for service selection and dynamic speed scaling," *International Journal of Web Services Research (IJWSR)*, vol. 10, no. 1, pp. 29–52, 2013.

[15] A. Wierman, L. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *2009 IEEE INFOCOM*, April 2009, pp. 2007–2015.

[16] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS '12)*, 2012, pp. 22–31.

[17] Y. Chen, J. Huang, X. Xiang, and C. Lin, "Energy efficient dynamic service selection for large-scale web service systems," in *2014 IEEE 21st International Conference on Web Services (ICWS '14)*, Jun. 2014, pp. 558–565.

[18] X. Zhang, X. Shen, and L.-L. Xie, "Joint subcarrier and power allocation for cooperative communications in lte-advanced networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 2, pp. 658–668, February 2014.

[19] Z. Zhou, F. Liu, Y. Xu, R. Zou, H. Xu, J. Lui, and H. Jin, "Carbon-aware load balancing for geo-distributed cloud services," in *2013 IEEE 21st International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS '13)*, Aug 2013, pp. 232–241.

[20] S. Liu, G. Quan, and S. Ren, "On-line real-time service allocation and scheduling for distributed data centers," in *2011 IEEE International Conference on Services Computing (SCC '11)*, July 2011, pp. 528–535.

[21] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 211–222, Aug. 2012.

[22] M. Lin, Z. Liu, A. Wierman, and L. Andrew, "Online algorithms for geographical load balancing," in *2012 International Green Computing Conference (IGCC)*, June 2012, pp. 1–10.

[23] M. Adnan, R. Sugihara, and R. Gupta, "Energy efficient geographical load balancing via dynamic deferral of workload," in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, June 2012, pp. 188–195.

[24] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in saas clouds," in *2013 IEEE INFOCOM*, 2013, pp. 872–880.

[25] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012.

[26] 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 Transactions on Computers*, vol. 60, no. 4, pp. 552–564, April 2011.

[27] Intel turbo boost technology - on-demand processor performance. [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html?wapkw=Turbo

[28] AMD PowerNow! Technology. [Online]. Available: http://www.amd.com/us/products/technologies/amd-powernow-technology/Pages/amd-powernow-technology.aspx

[29] R. Guerra, J. Leite, and G. Fohler, "Attaining soft real-time constraint and energy-efficiency in web servers," in *2008 ACM symposium on Applied computing*. ACM, 2008, pp. 2085–2089.

[30] D. Lin, C. Shi, and T. Ishida, "Dynamic service selection based on context-aware QoS," in *2012 IEEE 9th International Conference on Services Computing (SCC '12)*, 2012, pp. 641–648.

[31] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining local optimization and enumeration for qos-aware web service composition," in *2010 IEEE 8th International Conference on Web Services (ICWS '10)*, July 2010, pp. 34–41.

[32] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *2010 Seventh International Conference on Information Technology: New Generations (ITNG)*, April 2010, pp. 684–689.

[33] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[34] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[35] Worldwide LHC Computing Grid. [Online]. Available: http://wlcg.web.cern.ch/search/node/LCG

[36] H. Li, M. Muskulus, and L. Wolters, "Modeling job arrivals in a data-intensive grid," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, E. Frachtenberg and U. Schwiegelshohn, Eds. Springer Berlin Heidelberg, 2007, vol. 4376, pp. 210–231.

[37] K. Deng, J. Song, K. Ren, and A. Iosup, "Exploring portfolio scheduling for long-term execution of scientific workloads in iaas clouds," in *Proceedings of SC13: International Conference for*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2444845, IEEE Transactions on Services Computing

High Performance Computing, Networking, Storage and Analysis. ACM, 2013, pp. 55–66.

[38] New York ISO Real-time Market LBMP. [Online]. Available: http://www.nyiso.com/public/markets_operations/market_data/pricing_data/index.jsp

[39] E. Al-Masri and Q. H. Mahmoud, "Qos-based discovery and ranking of web services," in *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, Aug 2007, pp. 529–534.

[40] J. T. Blake, A. L. Reibman, and K. S. Trivedi, "Sensitivity analysis of reliability and performability measures for multiprocessor systems," in *Proceedings of the 1988 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '88, 1988, pp. 177–186.

Ying Chen received the B.Eng. degree in School of Computer Science from Beijing University of Posts and Telecommunications, Beijing, China, in 2012. She is currently working towards the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University. Her research interests are modeling, performance evaluation and optimization of web services and services computing. She is a student member of the IEEE.

Chuang Lin is a professor of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree in Computer Science from the Tsinghua University in 1994. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He has published more than 300 papers in research journals and IEEE conference proceedings and has published five books. He is a member of ACM Council, a senior member of the IEEE and the Chinese Delegate in TC6 of IFIP. He serves as the Associate Editor of IEEE Transactions on Vehicular Technology, the Area Editor of Journal of Computer Networks and the Area Editor of Journal of Parallel and Distributed Computing.

Jiwei Huang is an assistant professor in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunications. He received the Ph.D. degree and B.Eng. degree both in computer science and technology from Tsinghua University in 2014 and 2009, respectively. He was a visiting scholar at Georgia Institute of Technology. His research interests are in services computing and performance evaluation. He has published more than 15 papers in international journals and conference proceedings, e.g., IEEE Transactions on Services Computing, SIGMETRICS, ICWS, SCC, etc. He is a member of the IEEE.

Xudong Xiang received the B.Sc. in information management and information systems and M.Sc. degree in management science and engineering from Beijing Information Science and Technology University, Beijing, China, in 2008 and 2011, respectively. He is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, University of Science and Technology Beijing. His current research focuses on the stochastic modeling, optimal control and performance evaluation of computer networks and systems. He is a student member of the IEEE.

Xuemin (Sherman) Shen received the B.Sc. degree from Dalian Maritime University, Dalian, China, in 1982, and the M.Sc. and Ph.D. degrees from Rutgers University, New Brunswick, NJ, USA, in 1987 and 1990, respectively, all in electrical engineering. He is a Professor and University Research Chair with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloon, ON, Canada. He was the Associate Chair for Graduate Studies from 2004 to 2008. His research focuses on resource management in interconnected wireless/wired networks, wireless network security, wireless body area networks, and vehicular ad hoc and sensor networks.He is a Distinguished Lecturer of the IEEE Communications Society. He received the Excellent Graduate Supervision Award in 2006 and the Outstanding Performance Award in 2004 and 2008 from the University of Waterloo, the Premiers Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada, and the Distinguished Performance Award in 2002 and 2007 from the Faculty of Engineering, University of Waterloo. He is a registered Professional Engineer of Ontario, Canada, a fellow of the IEEE, a fellow of Canadian Academy of Engineering and a fellow of Engineering Institute of Canada.