Energy Minimization for Cloud Services with Stochastic Requests

Shuang Wang[1,2](✉), Quan Z. Sheng[2], Xiaoping Li[1], Adnan Mahmood[2], and Yang Zhang[2,3]

[1] Southeast University, Nanjing, Jiangsu, China
{wangshuang,xpli}@seu.edu.cn
[2] Macquarie University, Sydney, Australia
{michael.sheng,adnan.mahmood}@mq.edu.au
[3] Wuhan University, Wuhan, Hubei, China
yangz10@whu.edu.cn

Abstract. Energy optimization for cloud computing services has gained a considerable momentum over the recent years. Unfortunately, minimizing energy consumption of cloud services has its own unique research problems and challenges. More specifically, it is difficult to select suitable servers for cloud service systems to minimize energy consumption due to the heterogeneity of servers in cloud centers. In this paper, the energy minimization problem is considered for cloud systems with stochastic service requests and system availability constraints where the stochastic cloud service requests are constrained by deadlines. An energy minimization algorithm is proposed to select the most suitable servers to achieve the energy efficiency of cloud services. Our intensive experimental studies based on both simulated and real cloud instances show the proposed algorithm is much more effective with acceptable CPU utilization, saving up to 61.95% energy consumption, than the existing algorithms.

Keywords: Energy minimization · Cloud service · Service request · Quality of Service · Rejection probability · System availability

1 Introduction

Energy optimization is not only important for protecting environments because it mitigates the carbon emission, but also indispensable for the cloud providers since it reduces the electricity consumption. It is estimated that 70 billion kilowatt-hours of electricity, i.e., about 1.8% of the total electricity consumption of the United States, is consumed in 2014 alone for cloud services [11]. According to the International Energy Agency's New Policies Scenario, which takes account of existing and planned government policies, the world primary energy demand is projected to be increased by 37% between 2012 and 2040 [23]. Energy consumption is directly proportional to power consumption. To reduce power consumption in computing systems, there are in general two approaches:

© Springer Nature Switzerland AG 2020
E. Kafeza et al. (Eds.): ICSOC 2020, LNCS 12571, pp. 133–148, 2020.
https://doi.org/10.1007/978-3-030-65310-1_11

(a) *thermal-aware hardware design* and (b) *power-aware software design* [20]. The thermal-aware hardware design approaches are related to hardware devices, while on the contrary, the power-aware software design approaches involve computing systems, including operating system-level power management, compiler-level power management, application-level power management, and cross-layer adaptations [20]. Our work falls largely as one of the power-aware software design methods.

In this paper, we take into consideration of the energy consumption minimization problem for stochastic requests with deadlines. Service requests arrive at the cloud centers both stochastically and dynamically while servers are heterogeneous in nature with different configurations. Dealing with stochastically arriving requests and heterogeneous servers in cloud service systems is a complex problem in its own essence [8,22]. The power consumption of servers and the response time of service requests are typically negatively correlated. If the service rate of servers is higher, more power is consumed and the service requests are processed quicker, thereby leading to a smaller response time. In addition, it is hard to evaluate the energy consumption with different service rates of servers in cloud systems. Server selection is a NP-hard problem. This is further complicated when selecting optimal number of servers from the cloud centers to minimize the energy consumption, with the needs to satisfy the deadlines of service requests and the system availability [15] (which defines the probability that requests can be processed) constraints of the selected servers. The major contributions of this work are as the following:

- We develop a novel cloud service system model that is based on the *queuing theory* to deal with the stochastic property of the cloud service requests. The model enables the efficient server selection by considering i) service request arrival rates, ii) service rates of the cloud servers, iii) deadlines of the service requests, and iv) the system availability constraints.
- The energy consumption of a cloud service system is measured by the power consumption of the servers based on their response time of the service requests, which is calculated by a proposed *energy evaluation* (EE) algorithm. We further develop an *energy minimization* (EM) algorithm to select suitable servers to minimize the energy consumption while meeting the service request deadlines and system availability constraints.
- We conduct extensive experimental studies using both simulated and real-life cloud instances and the results show the effectiveness of our proposed approach on energy optimization of cloud services.

The rest of the paper is organized as follows. In Sect. 2, we discuss the related work and in Sect. 3, we detail the model and formulate the research problem. The energy optimization algorithms are presented in Sect. 4. Finally, the experimental results are reported in Sect. 5, followed by conclusions and future research directions in Sect. 6.

2 Related Work

Energy consumption is closely related to power consumption. A stochastic activity network model was constructed to evaluate the power consumption and performance of servers in cloud computing [3]. By allocating the power to the servers, the overall quality of service (QoS) of the servers in the data center was optimized [7]. A heuristic algorithm was presented to minimize the energy consumption on the condition of location constraint of nodes [21]. A brownout-based approximate Markov Decision Process approach was proposed to improve trade-offs between energy saving and discount offered to service users [22]. The key novelty was to reduce the cumulative power by the dynamic voltage scaling [1]. A Constrained Markov Decision Process model was built for power management in web server clusters [18]. The energy consumption was minimized by optimizing the power consumption of different servers in [1,7,18] while the deadline constraint of service requests was not considered. In this paper, we consider the stochastic service requests with deadline constraint and the suitable servers are selected to minimize the energy consumption. The dynamic property of the queue capacity makes the energy consumption minimization problem different from the existing problems tackled in [1,7,18].

Low energy consumption and high service performance (e.g., response time, reliability and service level agreement) are usually negatively correlated. Higher performance implies less response time and faster service rates while lower energy implies lower allocated power which results in slower service rates and longer response time. A suitable queuing model was built to satisfy the conflicting objectives of high performance and low power consumption in [10]. Mobile devices were modeled as a semi-Markov decision process to achieve a good balance between the application execution time and power consumption [2]. The balance between higher performance and lower energy was studied in [2,10]. In [19], two novel adaptive energy-aware algorithms were proposed to achieve energy efficiency while minimizing SLA (service-level agreement) violation rate in cloud data centers. When balancing the performance and energy consumption, the queue capacity in [2,10] is determined while in our paper, the queue capacity changes dynamically with different servers.

Energy minimization with heterogeneous cloud servers has been studied in [6,8,14]. When the servers are heterogeneous, the server sequence has to be certain for performance analysis since different server sequences lead to different results. The heterogeneity of servers make problem more difficult. In this paper, heterogeneous servers in cloud service systems are considered with deadline constraint, which is not handled in the existing works [6,8,14].

3 Cloud Service System Model and Problem Description

In this section, the cloud service system model is constructed and then the research problem is formally defined.

3.1 Cloud Service System Model

When service requests arrive, cloud providers offer suitable servers to deal with the requests while minimizing the energy consumption. The arrival rate of the service requests is assumed to follow a Poisson distribution with parameter λ [9]. Let us assume that there are N heterogeneous servers in the cloud service system and the number of selected servers is n ($n \leq N$). According to our recent work [15], the service rates of the servers, which are the speed of requests processed by the servers, are assumed to follow exponential rates with $\mu_1, \cdots, \mu_N$. The deadline of service requests is denoted as D which implies that requests are processed before the arriving time plus D. All requests have the same D. The rejection probability is assumed to be P_R and the system availability is ξ.

The model of the cloud service system can be constructed, as illustrated in Fig. 1. Once the service requests arrive at the cloud system, a dedicated server is firstly selected to process these requests. If more requests arrive, requests wait in the queue. Due to the deadline constraint, the maximum number of service requests is computed by the selected server and if the system availability is satisfied, no more server will be selected. Requests are executed by the selected servers. Otherwise, more servers will be considered by iteration in order to meet the requirement of the system availability.

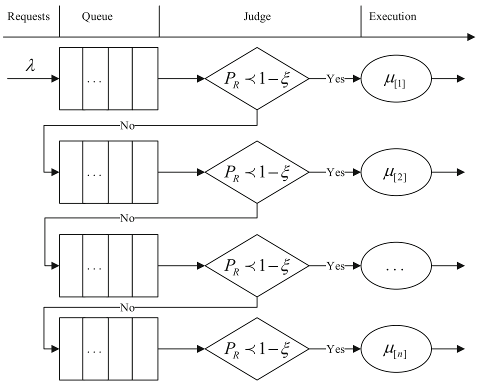

Fig. 1. Cloud service system model based on the queuing theory.

The queuing theory [13] is adopted in terms of the stochastic property of the cloud service requests. $R_{[i]} (i \in \{1, \cdots, n\})$ is the maximum queue length when the server with service rates $\mu_{[i]}$ is selected to process requests. $R_{[i]}$ is determined in terms of comparing the response time of requests processed by service rate with $\mu_{[i]}$ to the deadlines. $\{0, 1, \cdots, i, \cdots, \sum_{j=1}^{n} R_{[j]} + n\}$ is the state space where i is the number of service requests in the system. According to the system model in Fig. 1, states $\{0, 1, \cdots, 1 + R_{[1]}\}$ correspond to the first selected server with service rate $\mu_{[1]}$. States $\{\sum_{j=1}^{i-1} R_{[j]} + i, \cdots, \sum_{j=1}^{i} R_{[j]} + i\}$ correspond to the i^{th}

selected server with service rate $\mu_{[i]}$. When $1-\xi \le P_{\sum_{i=1}^{n} R_{[i]}+n}$, no new server will be selected. According to the stochastic property of requests and servers, the cloud service system is a Markov process. For the first selected server, the arrival rate of requests is λ while the arrival rate of next selected servers is the rejection probability that requests cannot be processed by the previous selected server by the arrival rate λ. The state transition process is shown in Fig. 2.

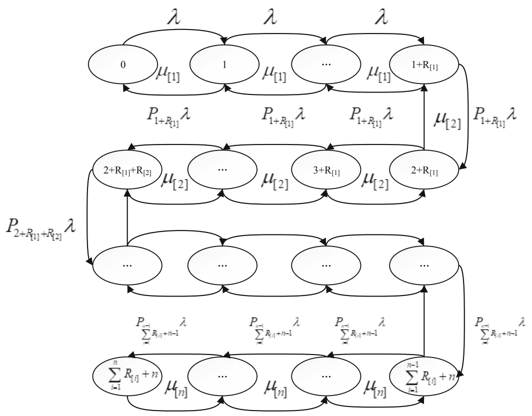

Fig. 2. State transition process.

According to the state transition process described in Fig. 2, the input rate of requests is equal to the output rate of requests for each state. Denote $P_i(i \in \{0, 1, \cdots, 1+R_{[1]}\})$ as the steady state probability for state i. Therefore, the balance equations for the first server are:

$$\lambda P_0 = \mu_{[1]} P_1 \tag{1}$$

$$(\lambda + \mu_{[1]}) P_1 = \lambda P_0 + \mu_{[1]} P_2 \tag{2}$$

$$\cdots$$

$$\lambda P_{R_{[1]}} = \mu_{[1]} P_{1+R_{[1]}} \tag{3}$$

In addition, the steady state probabilities $P_0, P_1, \cdots, P_{1+R_{[1]}}$ satisfy

$$\sum_{i=1}^{1+R_{[1]}} P_i = 1 \tag{4}$$

$\rho_{[1]}$ is denoted as $\frac{\lambda}{\mu_{[1]}}$. According to Eq. (1), Eq. (2) and Eq. (3), it is obtained

$$P_1 = \rho_{[1]} P_0 \tag{5}$$

$$P_2 = \rho_{[1]}^2 P_0 \tag{6}$$

$$\cdots$$

$$P_{1+R_{[1]}} = \rho_{[1]}^{1+R_{[1]}} P_0 \tag{7}$$

P_0 is calculated in terms of Eq. (4), (5), (6) and (7):

$$P_0 = \begin{cases} \dfrac{1-\rho_{[1]}}{1-\rho_{[1]}^{R_{[1]}+2}} & \rho_{[1]} \neq 1 \\ \dfrac{1}{2+R_{[1]}} & \rho_{[1]} = 1 \end{cases} \tag{8}$$

The steady state probabilities in different states are calculated according to Eq. (5), (6), (7) and (8). Based on the steady state probabilities, the rejection probability $P_{r_{[1]}}$ is calculated using:

$$P_{r_{[1]}} = P_{1+R_{[1]}} \tag{9}$$

The rejection probability $P_{r_{[1]}}$ is related to the system availability ξ. When the n^{th} server is selected, it should meet the requirement of the system availability ξ, which implies that $P_{r_{[n]}} \leqslant 1-\xi$. For minimizing the energy consumption, when a server is selected, the rejection probability $P_{r_{[1]}}$ should be as small as possible so that more service requests can be processed. The relationship between $P_{r_{[1]}}$ and $R_{[1]}$ can been proved in Theorem 1.

Theorem 1. *$P_{r_{[1]}}$ decreases with the increase of $R_{[1]}$.*

Proof. According to (8), $P_{r_{[1]}} = \frac{(1-\rho_{[1]})\rho_{[1]}^{1+R_{[1]}}}{1-\rho_{[1]}^{2+R_{[1]}}}$. The derivative of $P_{r_{[1]}}$ is calculated as

$$\begin{aligned} \frac{DP_{r_{[1]}}}{DR_{[1]}} &= \frac{((1-\rho_{[1]})\rho_{[1]}^{1+R_{[1]}})'(1-\rho_{[1]}^{2+R_{[1]}}) - (1-\rho_{[1]}^{2+R_{[1]}})'((1-\rho_{[1]})\rho_{[1]}^{1+R_{[1]}})}{(1-\rho_{[1]}^{2+R_{[1]}})^2} \\ &= \frac{((1-\rho_{[1]})\rho_{[1]}^{1+R_{[1]}})\ln\rho_{[1]}(1-\rho_{[1]}^{2+R_{[1]}}) + \rho_{[1]}^{2+R_{[1]}}\ln\rho_{[1]}(1-\rho_{[1]})\rho_{[1]}^{1+R_{[1]}}}{(1-\rho_{[1]}^{2+R_{[1]}})^2} \\ &= \frac{((1-\rho_{[1]})\rho_{[1]}^{1+R_{[1]}})\ln\rho_{[1]}}{(1-\rho_{[1]}^{2+R_{[1]}})^2} \end{aligned} \tag{10}$$

When $0 < \rho_{[1]} < 1$, it is easy to calculate that only $\ln\rho_{[1]} < 0$ which results in $\frac{DP_{r_{[1]}}}{DR_{[1]}} < 0$. $P_{r_{[1]}}$ decreases with the increase of $R_{[1]}$. When $\rho_{[1]} = 1$, according to Eq. (5), (6), (7) and (8), the steady state probabilities are the same for all states. $P_{r_{[1]}}$ decreases with the increase of the number of states which results from the increase of $R_{[1]}$. When $\rho_{[1]} > 1$, only $1-\rho_{[1]} < 0$. $\frac{DP_{r_{[1]}}}{DR_{[1]}} < 0$. Therefore, $P_{r_{[1]}}$ decreases with the increase of $R_{[1]}$. ■

Based on the steady state probabilities, it is easy to calculate the number of requests in the cloud service system. The number of service requests $L_{[1]}$ in the system is

$$L_{[1]} = \sum_{i=0}^{1+R_{[1]}} i \times P_i \tag{11}$$

Based on $L_{[1]}$, the response time of requests is determined. The response time of requests $T_{r_{[1]}}$ is determined based on the Little theorem [5]:

$$T_{r_{[1]}} = \frac{L_{[1]}}{\lambda(1 - P_{r_{[1]}})} \tag{12}$$

If $T_{r_{[1]}} \leq D$, compared $P_{r_{[1]}}$ to $1 - \xi$, $R_{[1]}$ is calculated. With the ξ (system availability) and D (service requests deadline) constraints, servers are selected to minimize the energy consumption in a cloud system. When the next server is selected, the arrival rate of requests changes with the current rejection probability $P_{r_{[i]}}$. Similar to the first selected server, the balance equations for the i^{th} selected server are obtained. The steady state probabilities are computed in terms of the balance equations. The rejection probability $P_{r_{[i]}}$ and the expected response time of requests are calculated by using Eq. (9) and (12).

Energy consumption is measured by the power consumption of servers based on the response time of requests. According to [8], the power consumption of a server is determined by $W = wCV^2\eta$, where w is the switching activity, C the electrical capacitance, V the supply voltage and η the clock frequency. For any physical server with a service rate $\mu_{[i]}$, $\mu_{[i]} \propto \eta$ and $\eta \propto V^{\phi}$ with $0 < \phi \leq 1$. $\eta \propto V^{\phi}$ implies $V \propto \eta^{1/\phi}$. According to [17], $\mu_{[i]} \propto \eta$ and $V \propto \eta$ imply $W_{[i]} \propto \mu_{[i]}^{\alpha}$ where $\alpha = 1 + 2/\phi \geq 3$, i.e., P can be represented by $\kappa\mu_{[i]}^{\alpha}$ where κ is a constant:

$$W_{[i]} = \kappa\mu_{[i]}^{\alpha} + W^*. \tag{13}$$

W^* is the static power consumption.

3.2 Problem Description

Since the number of servers are dynamic, the expected energy consumption is calculated in terms of the expected response time of service requests, the power consumption of servers, and the probability of states calculated by the corresponding servers. The total number of service rates should be not less than λ to balance the system. Otherwise, the system could break down. The energy consumption minimization problem is therefore formally described as follows:

$$\min E = (1 - P_{r_{[1]}})W_{[1]}T_{r_{[1]}} + \sum_{i=2}^{n} \prod_{j=1}^{i-1} P_{r_{[j]}}(1 - P_{r_{[i]}})W_{[i]}T_{r_{[i]}} \tag{14}$$

$$\lambda \leq \sum_{i=1}^{n} \mu_{[i]} \tag{15}$$

$$1 - \xi \leq P_{\sum_{i=1}^{n} R_{[i]}+n} \tag{16}$$

$$T_{r_{[i]}} \leq D \quad i \in \{1, 2, \cdots, n\} \tag{17}$$

$$n \leq N \tag{18}$$

According to Eq. (12), we can have $T_{r_{[i]}} = \frac{L_{[i]}}{\lambda(1-P_{r_{[i]}})}$. Therefore, the following equation is obtained:

$$\begin{aligned}
E &= (1 - P_{r_{[1]}})W_{[1]}T_{r_{[1]}} + \sum_{i=2}^{n}\prod_{j=1}^{i-1} P_{r_{[j]}}(1 - P_{r_{[i]}})W_{[i]}T_{r_{[i]}} \\
&= (1 - P_{r_{[1]}})W_{[1]}\frac{L_{[1]}}{\lambda(1 - P_{r_{[1]}})} + \sum_{i=2}^{n}\prod_{j=1}^{i-1} P_{r_{[j]}}(1 - P_{r_{[i]}})W_{[i]}\frac{L_{[j]}}{P_{r_{[j]}}\lambda(1 - P_{r_{[i]}})} \\
&= \sum_{i=1}^{n}\frac{W_{[i]}L_{[i]}}{\lambda} \qquad (19)
\end{aligned}$$

Therefore, Eq. (14) is transformed equivalently into $E = \sum_{i=1}^{n} \frac{W_{[i]}L_{[i]}}{\lambda}$.

4 Energy Minimization Algorithm

Energy minimization is closely related to the response time of service requests, the number of servers and the power consumption in a cloud service system. To solve the problem, different servers are selected to satisfy the deadline constraint and minimize the energy consumption. The number of service requests waiting in the cloud system is determined by the selected servers. The queue capacity is determined firstly by considering the deadline constraint D and the system availability ξ by Algorithm 1 according to which the energy consumption is evaluated. After energy evaluation, selected servers are determined to minimize the energy consumption in Algorithm 2.

The Energy Evaluation (EE) algorithm (see Algorithm 1) is proposed to evaluate the energy consumption of a selected server. μ is assumed as the service rate of the selected server. With the increase of R, the response time of service increases while the rejection probability decreases gradually. With the deadline D and the system availability ξ constraints, the queue capacity R is determined (lines 3–6). The number of requests in the cloud service system is computed (line 7) and the power consumption for the selected server is calculated (line 8). The energy consumption E is evaluated (line 9). The time complexity of Algorithm 1 is determined by the queue length R.

Different servers are selected to minimize the energy consumption in the cloud service system by Algorithm 2. $\boldsymbol{E}^o$ is the energy consumption vector for the selected servers and $\boldsymbol{P_r^o}$ is the rejection probability vector for the selected servers. $U = \{\mu_1, \cdots, \mu_N\}$ is the server set. $\boldsymbol{E}$ is the energy consumption vector for servers. E_{sum} is the total energy for the selected servers. The arrival rates of service requests for the next server is determined by the current rejection probability in terms of Fig. 2. During the server selection procedure, N servers are evaluated firstly (lines 5–6). Since the rejection probability in the cloud service system and the energy consumption have different ranges and units, we employ a min-max normalization. We use $\boldsymbol{E}'$ and $\boldsymbol{P_r}'$ to denote the values of $\boldsymbol{E}^o$ and $\boldsymbol{P_r^o}$ after min-max normalization, respectively (line 9). When selecting

Algorithm 1: Energy Evaluation (EE) Algorithm

```
  Input: $\mu, \xi$
1 begin
2     $R \leftarrow 0$, $T_r \leftarrow 0$,$P_R \leftarrow 1$;
3     while $T_r < D$& $P_R > 1 - \xi$ do
4         $R \leftarrow R + 1$;
5         Calculate $T_r$ by Eq. (12) and $\mu$;   /* Response time calculation*/
          Calculate $P_R$ by Eq. (9);   /* Rejection probability calculation*/
6     Calculate $L$ by $R$ and $\mu$;   /*The number of requests calculation*/
      Calculate $W$ by Eq. (13);   /*Power consumption calculation*/
      $E \leftarrow \frac{LW}{\lambda}$;   /*Energy consumption evaluation*/
7 return $R, P_R, E$.
```

more servers, the rejection probability of the system decreases while the energy consumption increases. We denote $\boldsymbol{r} = \frac{\boldsymbol{P_r}'}{\boldsymbol{E}'}$ as the selection metric (line 9). A server with the $\min(\boldsymbol{r})$ value is selected to minimize energy consumption (lines 10–11). With the constraint of system availability ξ, different servers are selected and the number of servers n is determined (lines 3–13). The n^{th} server is compared with the rest servers to minimize the energy consumption further because one of the rest servers may be better than the last selected server with the system availability ξ and deadline D constraints (lines 15–24). The time complexity of Algorithm 2 is $O(NR)$.

5 Experiments

In the proposed EM algorithm, there are five system parameters and we will first present our experiments to calibrate the parameters using simulated cloud instances. We then present experimental results to compare the proposed EM algorithm against three existing algorithms using both simulated and real-life instances. All compared algorithms are coded in MATLAB and un on an Intel Core i7-4770 PC (CPU@3.20 GHz, RAM@8 GBytes).

5.1 Parameter Calibration

For energy minimization in a cloud service system, the commonly tested parameters are: the total number of servers in a cloud service system N, the service rates of the N heterogeneous servers, the service request arrival rate λ, the request deadline D, and the system availability ξ. To statistically analyze the effects of the system parameters on the proposed algorithm framework, we calibrate these parameters over randomly generated test instances.

Since the values of the calibrated parameters should be as close as possible to real scenarios, we set the parameter configurations according to the Alicloud[1]

[1] https://github.com/alibaba/clusterdata.

Algorithm 2: Energy Minimization (EM) Algorithm

```
Input: $U, \lambda, \xi$
begin
    $\boldsymbol{E}^o \leftarrow \mathbf{0}$, $\boldsymbol{P}_r^o \leftarrow \mathbf{0}$, $U_s \leftarrow \emptyset$;
    while $P_R > 1 - \xi$ do
        $\boldsymbol{E} \leftarrow \mathbf{0}$, $\boldsymbol{P}_r \leftarrow \mathbf{0}$, $\boldsymbol{r} \leftarrow \mathbf{0}$, $E_{sum} \leftarrow 0$;
        for $i = 1$ to $N$ do
            $[R, \boldsymbol{E}_i, \boldsymbol{P}_{r\,i}] \leftarrow EE(\mu_i, \xi)$;   /*Energy evaluation*/
        $r_{min} \leftarrow 10$, $\boldsymbol{P}_r{}' \leftarrow \mathbf{0}$, $\boldsymbol{E}' \leftarrow \mathbf{0}$;
        for $i = 1$ to $N$ do
            $\boldsymbol{E}_i' \leftarrow \frac{\boldsymbol{E}_i - \min(\boldsymbol{E})}{\max(\boldsymbol{E}) - \min(\boldsymbol{E})}$, $\boldsymbol{P}_r{}'_i \leftarrow \frac{\boldsymbol{P}_{r\,i} - \min(\boldsymbol{P}_r)}{\max(\boldsymbol{P}_r) - \min(\boldsymbol{P}_r)}$,
             $\boldsymbol{r}_i \leftarrow \frac{\boldsymbol{P}_r{}'_i}{\boldsymbol{E}_i'}$;   /*Normalization*/
            if $\boldsymbol{r}_i \leqslant r_{min}$ then
                $r_{min} \leftarrow \boldsymbol{r}_i$, $k \leftarrow i$;   /*the $i^{th}$ server selection*/
        $U \leftarrow U - \mu_k$, $U_S \leftarrow U_S \cup \mu_k$, $n \leftarrow n + 1$, $N \leftarrow N - 1$;
        $P_R \leftarrow \boldsymbol{P}_{r\,k}$, $P_{ri}^o \leftarrow \boldsymbol{P}_{r\,k}$, $E_i^o \leftarrow \boldsymbol{E}_k$, $E_{sum} \leftarrow E_{sum} + \boldsymbol{E}_k$;
    /*Last server determination*/;
    for $i = 1$ to $N$ do
        $R \leftarrow 1$, $T_R \leftarrow 0$;
        if $n > 1$ then
            $P_R \leftarrow P_{r(n-1)}^o$;
        else
            $P_{ri}^o \leftarrow 1$;
        $[R, E_1, P_R] \leftarrow EE(\mu_i, \xi)$;   /*Energy evaluation*/
    $E' \leftarrow \boldsymbol{E}_n^o$;
    if $E_1 < E'$ then
        $E_{sum} \leftarrow E_{sum} - E' + E_1$, $E' \leftarrow E_1$, $\boldsymbol{P_{rn}^o} \leftarrow P_R$, $U_{Sn} \leftarrow \mu_i$;
return $R$, $E_{sum}$.
```

as: $N \in$\{10, 20, 30, 40\}, $\lambda \in$\{\{1, ..., 20\}, \{21, ..., 40\}, \{41, ..., 60\}, \{61, ..., 80\}\}(per second), $D \in$\{0.2, 0.4, 0.6, 0.8, 1\}(second), $\xi \in$\{0.55, 0.65, .75, 0.85, 0.95\}. The maximum service rate of cloud servers is assumed to be $\mu \in$\{10, 20, 30, 40\}(per second). The service rates of heterogeneous servers are determined by $\mu_i = \frac{i\mu}{N}(i \in \{1, \cdots, N\})$. Therefore, there are $4 \times 4 \times 5 \times 5 \times 4 = 1,600$ parameter combinations in total. Five instances are generated randomly for each arrival rate λ, i.e., five instances are generated for each combination, that is, $1,600 \times 5$=8,000 instances in total are tested for calibrating the parameters combinations.

Th experimental results are analyzed by using the multi-factor analysis of variance (ANOVA) statistical technique [15]. Three main hypotheses (normality, homoscedasticity, and independence of the residuals) are checked from the residuals of the experiments. All three hypotheses can be accepted by considering the well-known robustness of the ANOVA technique. The resulting p-values

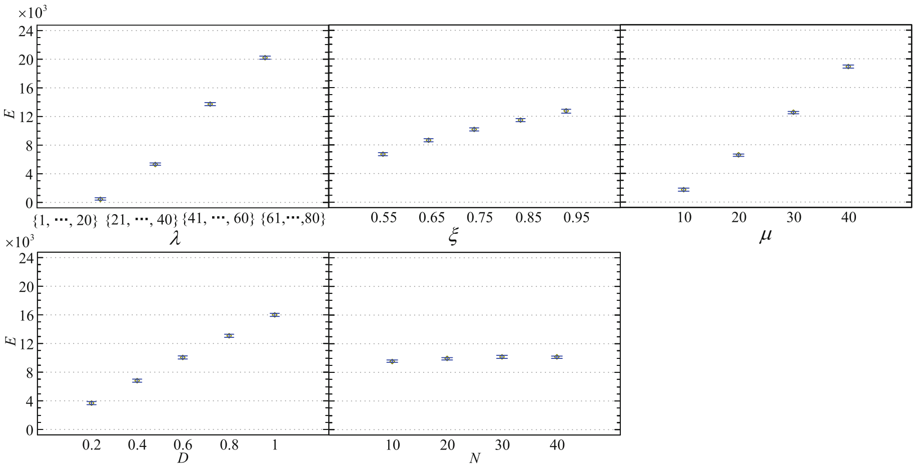

Fig. 3. Means plot of the five studied parameters with 95% confidence level Tukey HSD intervals.

are less than 0.05, meaning that all studied factors have a significant impact on the response variables at the 95% confidence level within ANOVA.

The means plot of the five studied factors on the total energy consumption E with 95% HSD (Tukey's Honest Significance Differences) intervals is shown in Fig. 3 and we can observe that:

- λ has a great influence on energy consumption E. With an increase in the upper bound of λ, E increases with statistically significant differences. E becomes minimum when λ takes values from $\{1, \ldots, 20\}$. The reason lies in that fewer arriving service requests can be processed by servers with small service rates, which consumes few energy.
- ξ greatly influences energy consumption. E increases with the increase of ξ. The differences are statistically significant. E becomes the minimum when ξ=0.55. The reason lies in that a higher ξ implies less rejection probability which requires more servers.
- Similarly, μ has a great impact on E. E takes the minimum value when $\mu = 10$ because a bigger μ implies more power consumption.
- D has a great influence on E. With an increase in D, the energy E increases. A bigger D results in an increase in the response time of requests for servers.
- Though the statistical differences of N on E are insignificant, which implies that the number of servers are satisfied with the stochastic requests, E becomes maximum when N equals 30.

5.2 Algorithm Comparison

During server selection, the FS (Fastest Server) policy always selects the fastest server while the Random policy selects servers randomly when the system availability is not satisfied. To minimize the energy consumption, the rejected requests

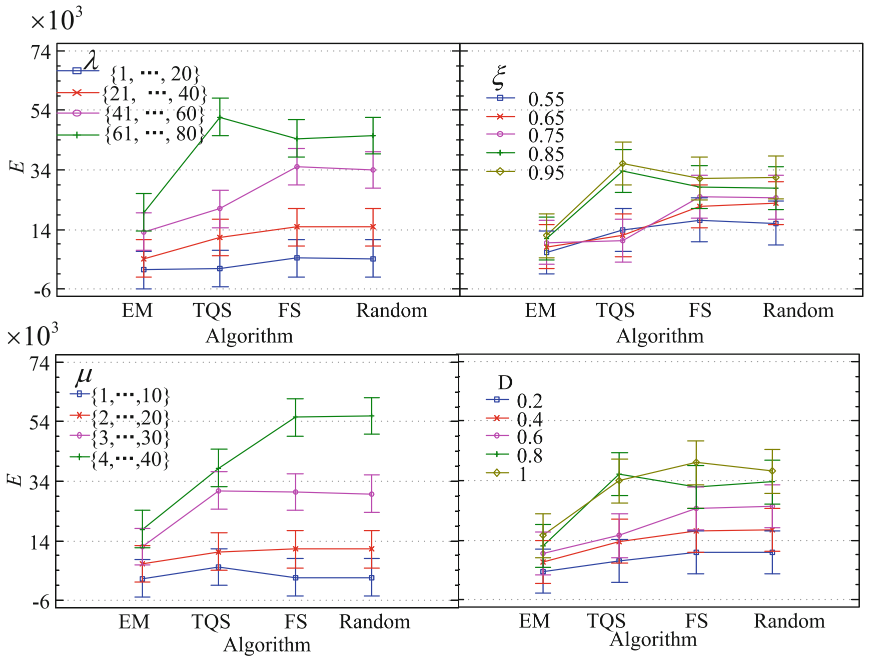

Fig. 4. The mean plots of the interactions between each parameter and the four compared algorithms.

that current server cannot process will be executed by the next selected servers while in the traditional queuing system (TQS) $M/M/N/N+R$ [13], it implies that service requests are processed immediately when there are idle servers. In our experiments, we compare EM with these three methods.

Algorithm Comparison over Simulated Instances. Since there are no benchmark instances for the considered problem, we first compare the four algorithms across simulated instances. Based on the calibrated results in Sect. 5.1, the system parameters N=10 and other parameters are the same as Sect. 5.1. Five instances are randomly generated for each of the 400 combinations, i.e., 2,000 instances are tested on each of the four algorithms. The results are shown in Fig. 4.

From Fig. 4, we can see that when the arrival rate λ takes a value from $\{61, \ldots, 80\}$, EM obtains the smallest E while TQS obtains the largest. FS and Random perform similarly. In other words, with an increase in service requests arrival rate λ, EM becomes more effective than the other three algorithms.

With an increase in ξ from 0.55 to 0.95, EM always results in the smallest E whereas TQS becomes the largest. FS is always worse than Random with a larger E. The higher values of ξ demonstrate the superiority of E. Random has the largest E which is similar to FS. The energy on TQS is smaller than FS. EM

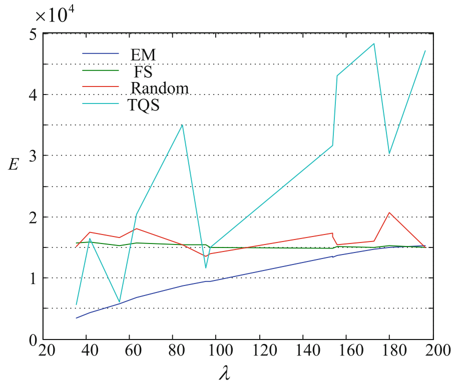

Fig. 5. Algorithm comparison over real instances.

is much more robust than the other three algorithms, i.e., with an increase in μ, the performance of EM fluctuates less than the other three. Similarly, with an increase of D, FS obtains the largest E, while EM gets the smallest E. Random is a littler larger than TQS.

Table 1. Algorithm comparison

	EM	TQS	FS	Random
E	9,398.28	21,220.6	24,705.0	24,659.6
CPU time	0.0043	0.0003	0.0023	0.0021

To further compare the algorithms, the average performance on effectiveness (the average energy consumption) and efficiency (CPU time) are shown in Table 1. According to Table 1, we can observe that EM obtains the smallest E, 9,398.28, followed by 21,220.6 of TQS. FS obtains the largest E 24,705 and for Random, it is 24,659.6. Comparing to TQS, FS, and Random, EM can save up to 61.95% energy consumption. Although EM has the longest CPU time of 0.0043 s, it is still comparable to the other three algorithms and acceptable in practice.

Algorithm Comparison over Real Instances. To evaluate the performance in real systems, the real production Cluster-trace-v2018[2] published by the Alibaba Group is analyzed, which contains eight-day sample data from one of the production clusters. By analyzing the *start_time*[3] of requests [15], the arriving time interval is obtained which implies that the arrival rates are Poisson distributed. According to *start_time* and *end_time*[4], the execution times of

[2] https://www.aliyun.com.
[3] http://clusterdata2018pubcn.oss-cn-beijing.aliyuncs.com/batch_task.tar.gz.
[4] http://clusterdata2018pubcn.oss-cn-beijing.aliyuncs.com/batch_instance.tar.gz.

all servers are calculated and the execution time of each server is exponential with different arrival rates and service rates. The arrival rate λ is analyzed by different types of service requests. The service rates $\mu_{[i]}$ $(i \in \{1, \ldots, N\})$ are evaluated by different types of servers. Similar to the simulated instances, D, ξ are set to 0.2, 0.95 respectively because the information on these parameters is not available in the real instances. The service rates are obtained with {14.5, 15.4, 16.9, 17.4, 18.5, 19.4, 20.4, 21.3, 22.8, 23.9} [15] and the number of servers is already known, i.e., $N = 10$.

Figure 5 shows the performance of the four compared algorithms. It can be observed that our proposed EM algorithm always obtains the smallest values as λ increases. FS, Random and TQS fluctuate as λ is less than 100. TQS is worse than Random followed by FS when λ is bigger than 100.

From the experiments on both simulated and real instances, we can observe the similar results in terms of algorithm performance. Our proposed EM algorithm always achieves the best performance on energy consumption.

6 Conclusion

Energy optimization of cloud computing services has been a key challenge due to their unique characteristics such as dynamic and stochastic service requests and heterogeneous cloud servers. In this paper, we present a novel approach to minimize the energy consumption by selecting appropriate cloud servers with the consideration of the service request deadline and system availability constraints. In particular, we develop a new cloud service system model based on the queuing theory to deal with stochastic service requests. An energy minimization (EM) algorithm is proposed to select the most suitable servers to achieve the energy efficiency while satisfying service request deadlines and system availability constraints. Our experimental studies show that the EM algorithm saves up to 61.95% energy consumption than the other algorithms. Ongoing work will focus on further improving our model by relaxing some constraints, e.g., the Poisson distribution of service requests.

References

1. Bilal, K., Fayyaz, A., Khan, S.U., Usman, S.: Power-aware resource allocation in computer clusters using dynamic threshold voltage scaling and dynamic voltage scaling: comparison and analysis. Cluster Comput. **18**(2), 865–888 (2015). https://doi.org/10.1007/s10586-015-0437-9
2. Chen, S., Wang, Y., Pedram, M.: A semi-Markovian decision process based control method for offloading tasks from mobile devices to the cloud. In: 2013 IEEE Global Communications Conference (GLOBECOM), pp. 2885–2890. IEEE (2013)
3. Entezari-Maleki, R., Sousa, L., Movaghar, A.: Performance and power modeling and evaluation of virtualized servers in IaaS clouds. Inf. Sci. **394–395**, 106–122 (2017)

4. Gerards, M.E.T., Hurink, J.L., Hölzenspies, P.K.F.: A survey of offline algorithms for energy minimization under deadline constraints. J. Sched. **19**(1), 3–19 (2016). https://doi.org/10.1007/s10951-015-0463-8
5. Gross, D., Harris, C.M.: Fundamentals of Queueing Theory. Wiley, New York (2008)
6. Khazaei, H., Ic, J., Ic, V.B., Mohammadi, N.B.: Modeling the performance of heterogeneous IaaS cloud centers. In: IEEE International Conference on Distributed Computing Systems Workshops, pp. 232–237. Philadelphia, PA, USA (2013)
7. Li, K.: Optimal power allocation among multiple heterogeneous servers in a data center. Sustain. Comput. Inf. Syst. **2**, 13–22 (2012)
8. Li, K.: Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management. IEEE Trans. Cloud Comput. **4**(2), 122–137 (2016)
9. Mei, J., Li, K., Li, K.: Customer-satisfaction-aware optimal multiserver configuration for profit maximization in cloud computing. IEEE Trans. Sustain. Comput. **2**(1), 17–29 (2017)
10. Mitrani, I.: Managing performance and power consumption in a server farm. Ann. Oper. Res. **202**(1), 121–134 (2013)
11. Shehabi, A., et al.: United states data center energy usage report. Technical report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States) (2016)
12. Tarello, A., Sun, J., Zafer, M., Modiano, E.: Minimum energy transmission scheduling subject to deadline constraints. In: Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2005), pp. 67–76. IEEE (2005)
13. Tijms, H.C.: A First Course in Stochastic Models. Wiley, Amsterdam (2004)
14. Tirdad, A., Grassmann, W.K., Tavakoli, J.: Optimal policies of M(t)/M/C/C queues with two different levels of servers. Eur. J. Oper. Res. **249**(3), 1124–1130 (2016)
15. Wang, S., Li, X., Ruiz, R.: Performance analysis for heterogeneous cloud servers using queueing theory. IEEE Trans. Comput. **69**(4), 563–576 (2020)
16. Wu, C.M., Chang, R.S., Chan, H.Y.: A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. Future Gener. Comput. Syst. **37**, 141–147 (2014)
17. Zhai, B., Blaauw, D., Sylvester, D., Flautner, K.: Theoretical and practical limits of dynamic voltage scaling. In: Design Automation Conference. Proceedings, pp. 868–873 (2004)
18. Zheng, X., Yu, C.: Markov model based power management in server clusters. In: IEEE/ACM Intl Conference on Green Computing and Communications and International Conference on Cyber, Physical and Social Computing (2010)
19. Zhou, Z., et al.: Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms. Future Gener. Comput. Syst. **86**, 836–850 (2018)
20. Zomaya, A.Y., Lee, Y.C.: Energy-Efficient Distributed Computing Systems. Wiley, New York (2012)
21. Cong, X., Zi, L., Shuang, K.: Energy-aware and location-constrained virtual network embedding in enterprise network. In: Liu, X., et al. (eds.) ICSOC 2018. LNCS, vol. 11434, pp. 41–52. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17642-6_4

22. Xu, M., Buyya, R.: Energy efficient scheduling of application components via brownout and approximate Markov decision process. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 206–220. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_14
23. Sieminski, A., et al.: International energy outlook. Energy Inf. Admin. (EIA) **18** (2014)