2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)

Energy Efficient Resource Management and Task Scheduling for IoT Services in Edge Computing Paradigm

Songyuan Li†, Jiwei Huang*‡

†*School of Computer Science*

Beijing University of Posts and Telecommunications, Beijing 100876, China

‡*State Key Laboratory of Networking and Switching Technology*

Beijing University of Posts and Telecommunications, Beijing 100876, China

Email: sylee1416@gmail.com, huangjw@bupt.edu.cn

***Abstract*—With the growing popularity of the Internet of Things (IoT), energy efficiency has been a critical concern during the design and development of IoT service systems. Meanwhile, edge computing has drawn significant attention as a burgeoning computing paradigm. This paper studies the energy efficiency issue of IoT systems by proposing a joint scheme of resource allocation and task scheduling under the edge computing paradigm. Specifically, dynamic processes of the IoT services and system are formulated by generalized queueing network models, based on which quantitative analyses of performance and energy consumption are conducted. The resource management and task scheduling are formulated by Markov Decision Process (MDP), which can balance the tradeoff between energy costs and QoS requirements. To attack the challenge of MDP search space explosion due to the large scale of IoT systems, Ordinal Optimization (OO) techniques are applied to the MDP algorithms, which are able to significantly narrow the search of MDP by slightly softening the optimization objective to a good enough subset. Finally, we conduct simulation experiments based on real-world IoT data. Evaluations and comparisons demonstrate that our approach is effective and efficient in practice.**

***Keywords*—Internet of Things (IoT); resource management; task scheduling; energy efficiency; Markov Decision Process; Ordinal Optimization.**

I. INTRODUCTION

The Internet of Things (IoT) is a burgeoning technique that connects interrelated computing devices over the network enabling their abilities to interact and cooperate with each other to create new applications and reach common goals [1]. With the rapid development of IoT, the huge computational workload for handling massive data generated by sensors and IoT devices has become a big challenge. Real-time data handling and information exchanges should be guaranteed, which nevertheless requires enhanced computational capacity at the edge of networks. For this case, traditional cloud-centric computational paradigm cannot accommodate itself to unpredictable sensing data explosion at the edge, thereby lacking the abilities of time-constrained IoT applications.

To attack this challenge, a novel computational paradigm namely edge computing has emerged, which extends the cloud resources to the edge of network and makes them being managed in a distributed way. It allows the computations to be performed at the edge of the network, from which sensors' data originates. Since many of the computational tasks can be completed near the data sources, the computation overload can be balanced in a distributed way and meanwhile the communication delay can be reduced [2]. Besides performance, energy efficiency is another hot topic in the globe. On the one hand, appropriate resource management according to dynamic workload has proved to be one of the most effective ways for improving energy efficiency in distributed computing systems [3]. On the other hand, it has been shown that the energy consumption brought by computations can be reduced if IoT devices with few computational resources upload their computation-intensive workload to the high-performance clouds [4]. Nevertheless, such uploading involves additional data transmission, consuming energy at the same time. Therefore, how to allocate the tasks between edges and clouds in order to attain optimal energy efficiency should be well studied accordingly.

There has already been a few research works to address the issue of task scheduling in IoT systems with edge computing paradigm. However, they only deal with the issue of task scheduling in edge-cloud systems, and do not simultaneously take into account resource management of servers [5-7]. A complete resource management framework in large scale computing systems exhibits high dimensions in state or action spaces. Deng at al. [8] proposed an analytical framework for edge-cloud systems to conduct the task scheduling balancing power consumption and delay, but the scheduling algorithm neglects significant scheduling times which high dimensions in action spaces bring about in large scale computing systems.

To fill these gaps, we make an attempt at improving the energy efficiency by jointly considering resource management and task scheduling in IoT systems which can be organized in edge computing paradigm. We theoretically model an edge-cloud system by queueing theory, and provide mathematical analyses of the performance and energy consumption. Based on the quantitative analyses, we design an algorithm of resource management and task scheduling based on Markov Decision

* Jiwei Huang is the corresponding author. Email: huangjw@bupt.edu.cn.

0-7695-6329-5/17/31.00 ©2017 IEEE
DOI 10.1109/ISPA/IUCC.2017.00129

Process (MDP) for obtaining tradeoff decisions between energy costs and performance metrics.. Fast scheduling is of much importance to guarantee user experience. To this end, we take advantage of Ordinal Optimization (OO) techniques in order to remarkably decrease the search space. The efficacy of our approach is validated by simulation experiments based on real-life data collected from IoT systems.

II. System Model

A. Sensor Hub

Data fusion has been widely applied in edge computing especially for sensor-based applications [9]. It is highly required to integrate multiple data from different sensors representing the same real-world object into a consistent and accurate representation. To facilitate such process, sensor hubs are designed and deployed at the front-end portal of the edge-cloud system. Besides integrating sensor data synchronously, sensor hubs are able to reduce the energy consumption by powering themselves down when idle [10].

In most of the sensor hubs, data is firstly buffered and waiting for further processing by the microcontroller unit (MCU) until a certain amount of data has been accumulated in the buffer [11]. The sensing data generated by sensors is processed in batch by the sensor hub. Therefore, a sensor hub can be theoretically formulated by a specialized queueing model with batch service, illustrated by Figure 1.

It has been shown that the task arrival at application layer of a networked system can be approximately formulated by Poisson distribution [12], and the service time of each request is assumed to be exponentially distributed. We let λ_i denote the individual task arrival rate while $\hat{\mu}_i$ is the batch service rate. The arrival of the sensor request batches can be approximately identified as a Poisson arrival process [13], and the arrival rate of the sensor request batches can be expressed by $\hat{\lambda}_i = \lambda_i / b_i$. The utilization of the sensor hub can be calculated by $\rho_i = \hat{\lambda}_i / \hat{\mu}_i$. The average response time of a batched request (i.e., waiting time and service time included) can be calculated using Little's Law, expressed as (1),

$$T_i^{RS} = \frac{\mathrm{E}[q_i]}{\hat{\lambda}_i} = \frac{b_i \cdot \mathrm{E}[q_i]}{\lambda_i} \tag{1}$$

where $\mathrm{E}[q_i]$ is the average queue length expressed as (2).

$$\mathrm{E}[q_i] = \frac{\rho_i}{1 - \rho_i} \tag{2}$$

Referring to the literature [14], the power consumption of computer circuits is divided into static power and dynamic power. The former one is caused by leakage currents and is independent of clock frequency and occupancy rates, while the later is related to circuit activity and mainly depends on clock frequency, occupancy rates and I/O status. Hence, the power of an edge server *i* can be formulated as follows.

$$p_i = \sigma_i \left(p_i^{(static)} + \rho_i \cdot p_i^{(dyn)} \right) \tag{3}$$

where σ_i indicates the on/off state of the server (1 means the server is on and 0 means off); $p_i^{(static)}$, $p_i^{(dyn)}$ and ρ_i respectively denote the static power and the dynamic power and the utilization of the sensor hub.

B. Edge and Cloud Server Cluster

An edge-cloud computing system is well-organized in a hierarchy consisting of edge servers and cloud servers with different computational capacities, from the core cloud server to the geographically located edge servers. Both edge and cloud servers are clusters including several computers interconnected by LANs, and requests are scheduled among them.

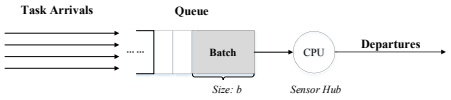

Fig. 1. Batch processing model for sensor hubs.

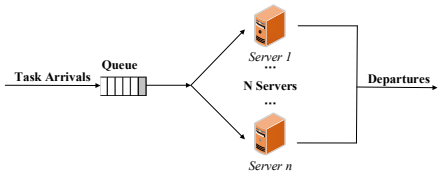

Fig. 2. Parallel computing model of an edge or cloud cluster.

The computing and scheduling amongst servers can be modeled by an M/M/n queuing system, demonstrated by Figure 2. It is assumed that the server cluster *j* consists of N_j physical servers, in which n_j ($0 \le n_j \le N_j$) servers are powered on.Let λ_j be the arrival rate of requests at the server cluster *j*, and μ_j be the service rate of a single server in the cluster j. Then, the utilization of each server in the cluster *j* is $\rho_j = \lambda_j / n_j \cdot \mu_j$. With Little's Law, the average response time of a request at the server cluster *j* can be formulated by (4),

$$T_j^{RS} = \frac{\mathrm{E}[q_j]}{\lambda_j} \tag{4}$$

where $\mathrm{E}[q_j]$ is the average queue length expressed as (5).

$$\mathrm{E}[q_j] = n_j \rho_j + \frac{\rho_j (n_j \rho_j)^{n_j}}{n_j! (1 - \rho_j)^2} \left[\sum_{k=0}^{n_j - 1} \frac{(n_j \rho_j)^k}{k!} + \frac{(n_j \rho_j)^{n_j}}{n_j! (1 - \rho_j)} \right]^{-1} \tag{5}$$

The power of the cluster *j* can be formulated as follows.

$$p_j = \sum_{k=1}^{N_j} \left(\sigma_k \left(p_k^{(static)} + \rho_k \cdot p_k^{(dyn)} \right) \right) \tag{6}$$

Here, σ_k denotes the on/off state of the k^{th} server in the cluster; $p_k^{(static)}$, $p_k^{(dyn)}$ and ρ_k respectively indicate the static power, the dynamic power and the utilization of the k^{th} server.

C. Edge-Cloud System

Figure 3 describes a hierarchical infrastructure of an edge-cloud system. In the outmost layer are the sensor hubs together with sensors, which are the origin of data. Relevant pieces of primitive data from multiple sensors are combined into a single one which provides a more precise description at the sensor hub. Blow is the edge layer, where the edge cluster connects the corresponding sensor hubs and sensors, and process some basic computational tasks. The services departing from sensor hubs are separately input into a set *M* of edge cluster through high speed networking. We label the j^{th} edge cluster by ε_j , while $H_i^{(\varepsilon_j)} (1 \le i \le I_j)$ denotes the i^{th} sensor hub subordinating to ε_j. A maximum of N_j servers can perform simultaneously for the edge cluster ε_j, and the actual amount of performing servers within *M* edge clusters is specified as an M-tuple $\boldsymbol{n} = (n_1, \ldots, n_j, \ldots, n_M)$. The central cloud cluster (labeled by *c*) is positioned in the core layer of the edge-cloud system. There are

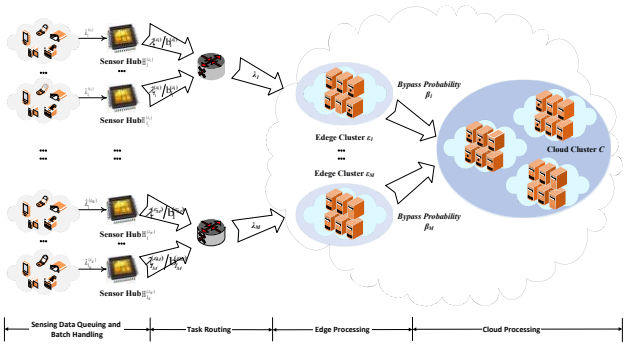

Fig. 3. Overall architecture for an edge-cloud system.

totally N_c servers deployed at the cloud cluster, and n_c indicates the number of power-on servers at the cloud.

Tasks are performed in alliance with edge and cloud servers. The bypass probability $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_j, \ldots, \beta_M)$ characterizes the task allocation between the edge and cloud layer, and the communication distance between the edge cluster ε_j and cloud cluster is D_j. Task scheduling is applied to determine whether the task is proceeded at the edge layer or bypass to the cloud layer. An M-tuple $\boldsymbol{q} = (q_1, \ldots, q_j, \ldots, q_M)$ represents the queue length within M edge clusters, which will be applied to the state formulation in the MDP problems defined later. Notice that $1 \leq q_j \leq Q_j$, and Q_j is the maximum queue length in the edge cluster ε_j.

As mentioned in the previous subsections, sensor hubs are modeled by M/M/1 queuing systems, and the edge and cloud clusters care formulated by M/M/n queuing models. Therefore, a queuing network model for the edge-cloud system is established. With Burke's Theorem, it has been widely accepted that the departure process of an M/M/1 or M/M/n system is also Poisson with the same departure rate as the arrival one. Since the arrivals of batched services approximately conform to Poisson distribution [13] and the service times are exponentially distributed, it is reasonable to assume the departures of the services from the sensor hubs are also Poisson.

Let $\lambda_i^{(\varepsilon_j)}$ and $\hat{\mu}_i^{(\varepsilon_j)}$ respectively indicate the individual task arrival and the batch service rate at the senor hub $H_i^{(\varepsilon_j)}$, and $b_i^{(\varepsilon_j)}$ is the batch size. The arrival rate of the sensor request batches can be calculated with $\hat{\lambda}_i^{(\varepsilon_j)} = \lambda_i^{(\varepsilon_j)} / b_i^{(\varepsilon_j)}$. The arrival rate λ_j of the edge cluster E_j can be expressed as (7), and the service rate of each edge server is labeled as μ_j. The service rate of cloud server is μ_c; the arrival rate λ_c can be calculated by (7).

$$\lambda_j = (1 - \beta_j) \cdot \sum_{i=1}^{I_j} \hat{\lambda}_i^{(\varepsilon_j)} \qquad \lambda_c = \beta_j \cdot \sum_{i=1}^{I_j} \hat{\lambda}_i^{(\varepsilon_j)} \tag{7}$$

In order to carry on the analysis and optimization, a discrete-time Markov chain (DTMC) is embedded into the CTMC. Suppose that each epoch has the length of time period τ equally, and the state transition cannot occur simultaneously. The embedded DTMC has the transition probability,

$$p_{q_j \to q_j+1} = \frac{\lambda_j \left(1 - e^{-\sum_{k=1}^{M}(\lambda_k + q_k\mu_k)\tau}\right)(1 - \beta_j)}{\sum_{k=1}^{M}(\lambda_k + q_k\mu_k)} \tag{8}$$

$$p_{q_j \to q_j-1} = \frac{q_j\mu_j \left(1 - e^{-\sum_{k=1}^{M}(\lambda_k + q_k\mu_k)\tau}\right)}{\sum_{k=1}^{M}(\lambda_k + q_k\mu_k)} \tag{9}$$

$$p_{loop} = e^{-\sum_{k=1}^{M}(\lambda_k + q_k\mu_k)\tau} + \frac{\lambda_j\beta_j \left(1 - e^{-\sum_{k=1}^{M}(\lambda_k + q_k\mu_k)\tau}\right)}{\sum_{k=1}^{M}(\lambda_k + q_k\mu_k)} \tag{10}$$

where $p_{q_j \to q_j+1}$ and $p_{q_j \to q_j-1}$ denote the one-step transition probability, and p_{loop} presents the probability maintaining state invariance.

Stationary computation time at the sensor hub and edge or cloud server cluster, which are separately labeled as $T_i^{(\varepsilon_j)}$ and T_j or T_c, can be calculated by (1) and (4). Then, the data transfer time from the edge cluster ε_j to the cloud cluster c can be formulated by $T_j^{TR} = D_j \cdot t$, where t is the data transfer time per unit communication distance.

Power consumed by the sensor hub and the server cluster, which are respectively indexed by $P_i^{(\varepsilon_j)}$ and P_j, can be calculated according to (3) and (6). Besides, the power consumption due to data transfer can be split into two portions. One is a fixed cost for channel acquirement and MAC layer control negotiation, and the other is packet size-dependent energy consumption [15]. Because the number of served tasks is approximately proportional to the number of packages being sent out, the energy consumed by data transmission between the i^{th} layer and j^{th} layer can be estimated by (11), where $p^{(dyn)}$ and $p^{(static)}$ are constant coefficients.

$$p_j^{TR} = p^{(dyn)} \cdot \lambda_j \cdot \beta_j \cdot \tau + p^{(static)} \tag{11}$$

III. Approach

Since the scale of edge-cloud system is extremely huge and speed of schedule is finite, it is essential to purpose an efficient scheduling strategy. Thus, we propose a task scheduling scheme with the allocation of resources by applying the thoughts of OO.

A. Reward Model

In order to put forward the scheduling scheme, we firstly present a reward model on the edge-cloud system. Since the senor hub usually performs in a stable periodic working state while is typically related to the specialized task, power consumption and response time at the sensor hub are kept as constant. Thus, our scheduling scheme mainly focuses on the task allocation between the edge and cloud layer. The first objective is to minimize the power consumption from resources consumption. As mentioned earlier in Section II, the power consumption is divided into two parts, one of which is from computation at the sensor hub and the edge or cloud cluster while another is power consumed by the data transfer. Therefore, the total power consumption can be defined as follows.

$$P^{sys} = \sum_{j=1}^{M} \left(P_j + P_j^{TR}\right) + P_c \tag{12}$$

The second objective is to meet the upper bound of the response time T_{SLA}. At the edge layer, we assume that the reward is closely related to the transient response time. At the cloud site, however, since the cloud is usually equipped with adequate resource and thus much more stable on performance with various workload, stationary response time is treated as the variable to estimate the computing time at the cloud layers. Considering the time taken by computation at the edge and cloud

servers and bypass transmission, the total time spent for a task can be formulated as (13). The conditions where the response time exceeds T_{SLA} should be eliminated, so the response time is treated as the pruning indicator in the reward model. Note that tasks are regarded as homogenous with the same T_{SLA}.

$$T^{sys} = \sum_{j=1}^{M} \lambda_j \cdot \left(\frac{q_j \cdot (1-\beta_j)}{n_j \cdot \mu_j} + \beta_j \cdot \left(T_j^{TR} + T_c \right) \right) \Big/ \sum_{j=1}^{M} \lambda_j \quad (13)$$

Thus, the reward model is set up based on the optimization of energy consumption and the constraint of response time. We conduct task scheduling with the resource allocation between the edge layer and the cloud layer, where the probability of bypass transmission and the number of power-on machines are decision variables. In the reward function, both time and energy elements are normalized, formulated as follows.

$$R = \frac{T_{SLA} - T^{sys}}{T_{SLA}} \Big/ \frac{P^{sys}}{P_{max}} \quad (14)$$

B. MDP Formulation

A primary scheduling strategy using Markov Decision Process (MDP) is given in this part. A standard MDP problem is defined as the following 5 ingredients.

Decision Epoch. A decision timeslot is indexed by $n \in \mathbb{N}^+$. Each decision timeslot takes τ time units. In correspondence, the decision is executed at $t = \tau, 2\tau, 3\tau, 4\tau, \ldots \ldots$

State and State Space. We assume that the cloud is stable on performance with various workload, and thus decision-making is conducted at the edge layer. The state $S(n)$ is defined by the M-tuple $q = (q_1, \ldots, q_j, \ldots, q_M)$, which represents the workload in M edge clusters.

Action and Action Space. Energy efficiency is achieved by allocating the computational workload between the edge and cloud layer and charging the number of power-on servers. Interlayer workload allocation is characterized as the probability of bypass transmission $\boldsymbol{\beta}$. The number of power-on servers at the edge layer is formulated by $\boldsymbol{n}$ while the number of power-on servers at the cloud is n_c. Thus, actions a_n can be expressed by $\boldsymbol{\beta} \times \boldsymbol{n} \times n_c$, and all candidate actions form the action space A.

Rewards Function. Let the reward function $r(S(n), a_n)$ defined by the reward at each timeslot, as $r(S(n), a_n) = r_n$, where r_n is the reward derived from (14) at the timeslot of n.

Objective Value Function. Let π denote the policy specifying the determined actions at a decision timeslot. Then our objective is to find out the best policy $\pi^* \in \Pi$ with the largest expected total discounted reward formulated as (15).

$$V^{\pi^*} = \max E^{\pi} \left\{ \lim_{N \to \infty} \sum_{n=1}^{N} \gamma^{n-1} \cdot r(S(n), a_n) \right\} \quad (15)$$

According to the definition of the Bellman Equations, the objective equation can be also expressed recursively as (16).

$$V_n(S(n)) = \max_{a_n \in A} \left\{ \begin{matrix} r(S(n), a_n) + \\ \gamma \sum_{s' \in S} p(s'|S(n), a_n) V_{n+1}(s') \end{matrix} \right\} \quad (16)$$

where $\gamma \in (0, 1)$ is a discount factor. The probability of state transition is defined in (8)-(10).

In spite of the high computational complexity to find the best actions for MDP problems using value iteration or policy iteration, it will take immense computation prices to work out the exact solution in the large scale edge-cloud system. Therefore, it is necessary to reduce the selection range of the action space in MDP.

C. An OO-Based Practical Solution

Borrowing the ideas of Ordinal Optimization (OO) [16], a scheme for tasks and resources scheduling is proposed as follows. In order to reduce time and space significantly, a coarse but efficient model is usually applied, which estimates the performance criteria of candidate solutions in the decision-making space $\bar{S}$. The decision-making space $\bar{S}$ here is defined by the action space A in our MDP problem. Then the number s of selected candidates is derived, given OPC type and error level. Our coarse model and how to select the required solutions will be introduced in detail subsequently.

Coarse Model. Differing from the refined model (i.e., MDP optimization), our coarse model ignores the future earnings and simply focuses on the current reward. In other words, we use the reward function (14) at each timeslot to estimate the performance criteria of scheduling scenarios. If $r_n < 0$, then the estimated value is set as 0.

Selecting the Scheduling Scenarios. Let us assume that there are s scheduling scenarios selected as the action space of MDP optimization. Here, s will be determined by the TSECS algorithm below. Empirically, the alignment probability P_A is set as 0.95, which is actually a high value. Based on the theory of OO, s generally depends on the nature of specific problems (i.e., Ordered Performance Curves (OPCs) and the error level).

Shapes of OPCs demonstrate the distribution of solutions in the solution domain $\bar{S}$, which indicate the quality of solutions (i.e., whether the good enough solutions are easy to obtain in the optimization problem). Based on the equation of (14), we can apply r_e to estimate the type of OPCs. Assuming that the size of $\bar{S}$ is $\bar{N}$, all the scenarios in $\bar{S}$ is pre-evaluated using the coarse model, and the estimated values of these scenarios are sorted in ascending order $r_e^{[1]}, r_e^{[2]}, \ldots \ldots, r_e^{[\bar{N}]}$. The ordered estimated value can be normalized into the range $[0,1]$.

$$r(x_i) = \frac{r_e^{[i]} - r_e^{[1]}}{r_e^{[N]} - r_e^{[1]}} \quad , \quad x_i = \frac{i-1}{\bar{N}-1} \quad for\ i = 1 \ldots \ldots \bar{N} \quad (17)$$

There are generally three kinds, which are flat, neutral and flat. If the OPC is flat, few good solutions are in the solution domain and the good enough scenarios are hard to obtain. To the contrary, if the OPC is steep, there are many good solutions totally and a smaller s is sufficient to cover the desired solutions.

Error level is another main factor which determines the value of s. It refers to the rate of divergence between estimated values and true values. In optimization for MDP, optimized policy depends more on the order of rewards while the concrete reward values are of little importance in the optimum iterative procedure. Therefore, the error derives from the partial disorder of reward values.

The estimated reward values of $\bar{N}$ candidates can be formulated by a vector $\boldsymbol{R_e} = \left(r_e^{(1)}, r_e^{(2)}, \ldots \ldots, r_e^{(\bar{N})} \right)$ while the true ones obtained by Bellman objective equation (16) are given as another vector $\boldsymbol{R} = \left(r^{(1)}, r^{(2)}, \ldots \ldots, r^{(\bar{N})} \right)$. Here, error level can be measured as the vectorial angle of $\boldsymbol{R_e}$ and $\boldsymbol{R}$. Let

$$diff_k = 1 - \cos\langle \boldsymbol{R_e^{(k)}}, \boldsymbol{R^{(k)}} \rangle = 1 - \frac{\boldsymbol{R_e^{(k)}}^T \cdot \boldsymbol{R^{(k)}}}{\left\| \boldsymbol{R_e^{(k)}} \right\|_2 \times \left\| \boldsymbol{R^{(k)}} \right\|_2} \quad (18)$$

represent the normalized difference for the k^{th} scheduling scenario, where $\boldsymbol{R_e^{(k)}}$ and $\boldsymbol{R^{(k)}}$ respectively denote the estimated and true values of the k^{th} scheduling scenario. The error level can be gained by the maximum of $diff_k$ among the $\bar{N}$ candidates, which can be calculated as

$$\max_{1\le k\le \bar{N}}\left\{1-\frac{\boldsymbol{R_e^{(k)^T}}\cdot\boldsymbol{R^{(k)}}}{\left\|\boldsymbol{R_e^{(k)}}\right\|_2\times\left\|\boldsymbol{R^{(k)}}\right\|_2}\right\} \quad (19)$$

If the error level is less than 0.5, it is considered to be a small one with s much smaller. It indicates medium one when $0.5 \le$ error level < 1, while error level ≥ 1 represents a large one while a larger s is required to cover the good solutions

With OPC type and error level defined, the number s of selected scheduling scenarios can be obtained based on OO techniques. Alignment level k and the size of good enough subset g can be freely regulated, while the alignment probability P_A is normally set as 0.95. The number s of selected scheduling scenarios can be calculated as (20)-(23), where Z_o, ρ, γ and η are correspondingly defined [16]. We apply (20), (21) and (23) to map our parameters to the parameters in [16] to eliminate the inconsistency caused by differences between the scale of solution space. The main steps of our TSECS algorithm are described in Algorithm 1.

$$k' = max\{1, \lceil 10000\cdot k/\bar{N}\rceil\} \quad (20)$$
$$g' = max\{1, \lceil 10000\cdot g/\bar{N}\rceil\} \quad (21)$$
$$s'(k', g') = e^{Z_o}(k')^{\rho}(g')^{\gamma} + \eta \quad (22)$$
$$s = \lceil \bar{N}\cdot s'/1000\rceil \quad (23)$$

Algorithm 1 Task Scheduling of Edge-Cloud System (TSECS)

Input: Decision-making space of scheduling scenarios $\bar{\boldsymbol{S}}$, the number of good enough solutions $\boldsymbol{g}$, alignment level $\boldsymbol{k}$
Output: Determined action $\boldsymbol{a_n}$.
1: Calculate the reward values of all the scheduling scenarios in $\bar{\boldsymbol{S}}$ using coarse model
2: Estimate the OPC type based on (17)
3: Estimate the normalized error level based on (19)
4: Calculate the number $\boldsymbol{s}$ of selected scenarios based on (20)-(23), and the theory of OO ensures that $\boldsymbol{s}$ scheduling scenarios contains at least $\boldsymbol{k}$ good enough scheduling scenarios with probability no less than 0.95
5: Use the iterative algorithm for MDP optimization to obtain the determined action $\boldsymbol{a_n}$ within the selected $\boldsymbol{s}$ scheduling scenarios
6: Return the determined action $\boldsymbol{a_n}$.

IV. Evaluation

A. Experimental Setup

In order to evaluate the validity and efficacy of our TSECS algorithm, we conduct experiments simulating an edge-cloud system where two edge clusters affiliate with one cloud cluster and each edge cluster receives task requests from senor hubs. Task requests are generated according to the workload trace from the T-Drive trajectory dataset. The T-Drive dataset collects the GPS trajectories of 10,357 taxies within the city of Beijing during a period of one week in 2008 [17,18]. We randomly select 10 taxis and aggregate their arrivals, which simulates the edge server gathering data from subordinate sensor hubs. With the intensive study of the arrival patterns, the

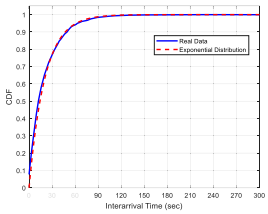

Fig. 4. CDF of interarrival times for an edge cluster.

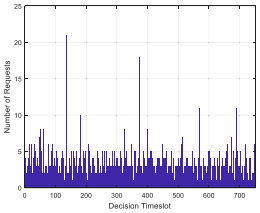

Fig. 5. T-Dive workload trace.

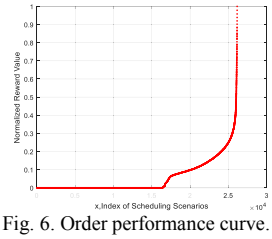

Fig. 6. Order performance curve.

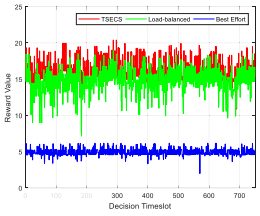

Fig. 7. Reward values under different algorithms.

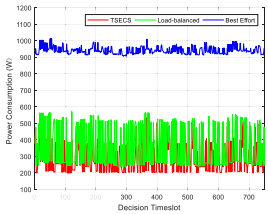

Fig. 8. Power consumption under different algorithms.

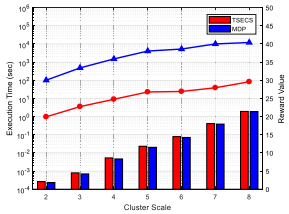

Fig. 9. Comparison of execution time and reward values.

TABLE I. Average Response Time of Different Algorithms

T_{SLA} (sec)	Algorithms Response Time (sec)		
	TSECS	Load-balanced	Best Effort
8.00	7.67	7.16	7.59

arrival is approximate to a procedure of Poisson arrival. The cumulative distribution function (CDF) of the aggregated arrivals at one of edge clusters is demonstrated as Figure 4, and the arrival process at another edge cluster is conducted with the similar manner. Therefore, the T-Drive dataset can be applied in our analyses presented in the previous sections, in order to make evaluations in reality. The period τ of each decision timeslot is set to be 20 seconds, and the workload trace is selected as the segment for 15,000 secs. Therefore, there are 750 decision timeslots in our simulation. The number of requests arriving in the edge-cloud system over timeslots is shown in Figure 5.

B. Experimental Results

Figure 6 shows the OPC, from which it can be seen that the OPC type is steep. It implies that good enough scheduling scenarios are easier to obtain, and fewer scheduling scenarios should be selected within the universal set of candidate solutions. We also estimate the error level of scheduling scenarios and obtain the normalized error as 0.5789 according to (19), which corresponds to a medium level. The OPC and the error level indicate that our TSECS algorithm can be effective and efficient.

We evaluate the effectiveness of our approach by comparing our TSECS algorithm with two other algorithms: (1) *Load-balanced* algorithm, where the requests are dispatched to edge or cloud servers based on the serving capacity, while the

resource management is the same as our TSECS algorithm; and (2) *Best Effort* algorithm, where all of servers are switched on and keep constantly running as long as the corresponding queue is not empty, while we use the same method as our TSECS algorithm for task scheduling.

Figure 7 demonstrates the reward value of the edge-cloud system under different algorithms. It can be seen that our TSECS algorithm achieves the highest reward value amongst the three algorithms, which verifies the effectiveness of our algorithm. The Load-balanced algorithm achieves higher reward value than those of the Best Effort algorithm, since the Best Effort algorithm consumes the maximum power energy whereas the response time differences amongst the three algorithms are minimal. Power consumption and response time costs will be introduced subsequently. As regards the energy costs, Figure 8 compares the power consumption amongst the three algorithms. Our TSECS algorithm is the most energy efficient. The Best Effort algorithm switches all of servers on, which consequently produces the largest power consumption. In terms of the response time, all of three algorithms meet the demand of SLA (i.e., T_{SLA}). Only slight differences exist in response time. Table I lists the average response time under the three algorithms amongst which our TSECS algorithm has the longest response time. Notwithstanding a little sacrifice of response time, our TSECS algorithms achieves the most energy efficient.

Furthermore, we also validate the efficiency of our TSECS algorithm by measuring the execution time and comparing reward values with the traditional MDP iterative algorithm. Figure 9 compares the execution time and reward values between the two approaches. With the scale expansion of server cluster, the growth of execution time outperforms the exponential increase. It also evidently shows that there are significant quantitative differences between these two approaches, where the execution time of our TSECS algorithm is within 100 seconds, while the traditional MDP iterative methodology takes much more time beyond orders of magnitude. The difference between the estimated reward values and the accurate reward values is extremely small.

V. CONCLUSION

This paper studies the performance and energy issue simultaneously in IoT systems. We firstly put forward a modeling approach of energy-aware performance evaluation for an edge-cloud IoT system. Then, we propose the TSECS algorithm for task scheduling and resource allocations, making tradeoff decisions between energy costs and performance metrics. In order to make our approach more effective, we introduce OO techniques to significantly promote the efficiency especially in large scale systems. Finally, the efficacy of our scheme is validated by experimental results from simulations based on real-life IoT data.

There are several avenues for future work. The models can be further specified according to different types of real-life IoT systems. Besides, detailed mathematical analysis of the TSECS algorithm should be provided to evaluate the efficiency of our algorithm theoretically. Moreover, the scheduling algorithm will be designed more carefully to be accustomed to various task arrivals other than an established distribution.

ACKNOWLEDGMENT

This work is supported by Beijing Natural Science Foundation (No. 4162042), National Natural Science Foundation of China (No. 61502043), and BeiJing Talents Fund (No. 2015000020124G082).

REFERENCES

[1] O. Vermesan and P. Friess, Internet of Things – From Research and Innovation to Market Deployment, River Press, 2014.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," IEEE Internet Of Things Journal, vol. 3, no. 5, pp. 637-646, 2016.

[3] S. Liu, G. Quan, and S. Ren, "On-line real-time service allocation and scheduling for distributed data centers," IEEE International Conference on Services Computing, 2011, pp. 528–535.

[4] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar and J. H. Abawajy, Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study, IEEE Access, vol. 5, pp. 9882-9910, 2017.

[5] P. Borylo, A. Lason, J. Rzasa, A. Szymanski and A. Jajszczyk, "Energy-aware fog and cloud interplay supported by wide area software defined networking," IEEE International Conference on Communications, 2016, pp. 1-7.

[6] X. Tao, K. Ota, M. Dong, H. Qi and K. Li, Performance Guaranteed Computation Offloading for Mobile-Edge Cloud Computing, IEEE Wireless Communications Letters, doi: 10.1109/LWC.2017. 2740927.

[7] Y. Mao, J. Zhang and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," IEEE Wireless Communications and Networking Conference, 2017, pp. 1-6.

[8] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption, IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171-1181, 2016.

[9] X. Yang, R.-P. Yang, and D.-D. Wu, "A resilient data fusion algorithm in wireless sensor networks," International Conference on Information System and Artificial Intelligence, 2016, pp. 192-195.

[10] M.Hayashikoshi, H.Kawai, H.Ueki, and T. Shimizu, "Normally-off MCU architecture and power management method for low-power sensor network," Internatiinal SoC Design Conference, 2015, pp. 151-152.

[11] T.-K. Chien, L.-Y. Chiou, S.-S. Sheu, J.-C. Lin, C.-C.Lee, T.-K. Ku, M.-J. Tsai, and C.-I. Wu, "Low-power MCU with embedded ReRAM buffers as sensor hub for IoT applications," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 6, no. 2, pp. 247-257, 2016.

[12] E. Chlebus and J. Brazier, "Nonstationary poisson modeling of web browsing session arrivals," Information Processing Letters, vol. 102, no. 5, pp. 187-190, 2007.

[13] B. Gunter, G. Stefan, d. M. Hermann, and S. T. Kishor, Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications, Hoboken, John Wiley, 2005.

[14] 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.

[15] S. Lindsey, K. Sivalingam, and C. S. Raghavendra, Handbook on Wireless Networks and Mobile Computing, Hoboken, John Wiley, 2001.

[16] T. W. Edward Lau and Y. C. Ho, "Universal alignment probabilities and subset selection for ordinal optimization," Journal of Optimization Theory and Applications, vol. 93, no. 3, pp. 455-489, 1997.

[17] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie et al.,"T-drive: driving directions based on taxi trajectories," SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2010, pp. 99-108.

[18] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 316-324.