



A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems

Lianyong Qi¹ · Yi Chen² · Yuan Yuan³ · Shucun Fu² · Xuyun Zhang⁴ · Xiaolong Xu^{2,5,6} 

Received: 29 January 2019 / Revised: 4 April 2019 / Accepted: 22 April 2019 /
Published online: 17 May 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Nowadays, with the development of cyber-physical systems (CPS), there are an increasing amount of applications deployed in the CPS to connect cyber space with physical world better and closer than ever. Furthermore, the cloud-based CPS bring massive computing and storage resource for CPS, which enables a wide range of applications. Meanwhile, due to the explosive expansion of applications deployed on the CPS, the energy consumption of the cloud-based CPS has received wide concern. To improve the energy efficiency in the cloud environment, the virtualized technology is employed to manage the resources, and the applications are generally hosted by virtual machines (VMs). However, it remains challenging to meet the Quality-of-Service (QoS) requirements. In view of this challenge, a QoS-aware VM scheduling method for energy conservation, named QVMS, in cloud-based CPS is designed. Technically, our scheduling problem is formalized as a standard multi-objective problem first. Then, the Non-dominated Sorting Genetic Algorithm III (NSGA-III) is adopted to search the optimal VM migration solutions. Besides, SAW (Simple Additive Weighting) and MCDM (Multiple Criteria Decision Making) are employed to select the most optimal scheduling strategy. Finally, simulations and experiments are conducted to verify the effectiveness of our proposed method.

Keywords Cyber-physical systems · QoS · Energy conservation · VM scheduling · Cloud

1 Introduction

In recent years, cyber-physical systems (CPS) which emerge as a novel computing system have gained a lot of popularity in many critical areas such as health care, manufacturing and

This article belongs to the Topical Collection: *Special Issue on Smart Computing and Cyber Technology for Cyberization*

Guest Editors: Xiaokang Zhou, Flavia C. Delicato, Kevin Wang, and Runhe Huang

✉ Xiaolong Xu
njuxlxu@gmail.com

Extended author information available on the last page of the article.

traffic control [29]. Plenty of enterprises utilize CPS to implement distributed computing resources. In CPS, physical systems work as sensors to collect information of real-world and send the sensory information to computation platforms for further analysis. Computation platforms analyze and process the information and then return a feedback or command to physical systems [16]. The real-time information brought by CPS is critical to make efficient decision. With the development of mobile devices, the integration of CPS and mobile devices bring more opportunities to attain more information. However, the complex applications in CPS (i.e. industrial applications and monitor applications) often require massive storages and computing resources to meet the performance requirements of users. Due to the storage and computing capacity limit of mobile devices, the performance of CPS applications is not capable of filling the bill. To meet the resource and storage requirements of applications in CPS, cloud computing emerges as a novel computing paradigm to provision rich computing resources [2, 20].

By virtue of the resource-rich cloud, cloud-based CPS improves the computing capacity of CPS to conduct resource-hungry applications. In order to provision the physical resources dynamically, virtualized technology is extensively used for resource management in the cloud platforms [13], which provides an effective manner to improve the resource efficiency of the cloud-based CPS. Running applications on the virtual machines (VMs) makes it possible for high resource utilization and low energy consumption. By virtue of integrating cloud with CPS, numerous systems like cloud-integrated vehicles which are unachievable due to the resources limit are able to be deployed efficiently [36].

With the aim of providing better service experience of cloud-based CPS, reasonable scheduling strategies are necessary to migrate applications to cloud more efficiently [3, 7]. Despite the advantages of VM migration, the VM migration operations also generate certain transmission delay, and the transmission of VM images leads to additional energy consumption of the switches in the datacenter [26]. Therefore, it is necessary to consider both the negative and the positive aspects of the VM migrations and determine the reasonable scheduling strategy due to the different requirements of users. Currently, the energy consumption of cloud-based CPS has received wide concerns since it not only augments the operating cost of the cloud providers, but also dramatically decreases battery life [4]. Thus, how to reduce energy consumption of the cloud datacenters becomes a key restrict for better user experience of resource intensive applications in cloud-based CPS [22, 27].

Meanwhile, owing to the rapid growth of multiple service demands, it remains challenging to meet the quality-of-service (QoS) requirements with the data traffic increasing [30]. It is troublesome to meet the global QoS requirement by virtue of the different users' preferences for a certain QoS. In view of this challenge, a QoS-aware VM scheduling method is proposed in this paper. By offloading various applications to other physical machines (PMs), the workloads on some under-load physical servers are migrated out, and these servers could be set to idle mode for energy saving [8, 14]. By reasonable offloading method, the requisite computing resources are provisioned and the QoS of applications in CPS can be ensured at the same time. Our work is to find an optimal VM scheduling strategy in cloud-based CPS with QoS enhancement. In view of this, a QoS-aware VM scheduling method for energy conservation, named QVMS, in cloud-based CPS is proposed. The main contributions are as follows.

- First, we define some basic concepts and present the system model. Then, we formalize the objective function and present the constraints.
- Besides, Non-dominated Sorting Genetic Algorithm (NSGA-III) is adopted to find the optimal VM scheduling strategies and achieve the goal of QoS enhancement including energy consumption, downtime and resource utilization.

- SAW (Simple Additive Weighting) and MCDM (Multiple Criteria Decision Making) are adopted to select the most optimal scheduling strategy.
- Finally, we conduct the extensive experimental evaluations to verify the effectiveness of our proposed VM scheduling method.

The rest of this paper is organized as follows. Section 2 presents the problem definition. Section 3 elaborates our proposed method. In Section 4, performance evaluation is illustrated. Section 5 summarizes the related work. Section 6 concludes the paper and presents the future work.

2 Problem definition

In this section, we introduce some formalized concepts in the QoS-aware VM scheduling method in cloud-based CPS. Quality of Service (QoS) includes a group of nonfunctional requirements like energy consumption, response time, availability, resource utilization, and throughput, etc. We analyze and quantify the energy consumption and resource utilization, and model the QoS-aware problem as a multi-objective optimization problem. Some key notations and descriptions used in the paper are listed in the Table 1.

2.1 Basic concepts

Suppose there are N PMs in the cloud environment used during the application execution, denoted as $P = \{p_1, p_2, \dots, p_N\}$. Besides, there are M CPS applications running on the PMs in P , denoted as $V = \{v_1, v_2, \dots, v_M\}$. We model the applications as special VMs which consist of multiple VM instances. Besides, we model the QoS indicators of the VM scheduling like energy consumption, downtime, resource utilization rate to quantify the QoS requirements. Let $X = \{x_1, x_2, \dots, x_M\}$ be the VM scheduling policy for the VMs in V , where $x_m \in P$ ($m = \{1, 2, \dots, M\}$) is the PM which the VM v_m is migrated to. Let $Y = \{y_1, y_2, \dots, y_M\}$ be the VM original deployment for the VMs in V , where $y_m \in P$ ($m = \{1, 2, \dots, M\}$) is the PM which the VM v_m is originally deployed on.

Table 1 Key notations and descriptions

| Notations | Descriptions |
|-----------|---|
| P | The PMs in cloud $P = \{p_1, p_2, \dots, p_N\}$ |
| V | The application in CPS $V = \{v_1, v_2, \dots, v_M\}$ |
| X | The VM scheduling policy for the VMs |
| Y | The VM original deployment |
| $E(X)$ | The total energy consumption |
| $AE(X)$ | The energy consumption of active VMs |
| $IE(X)$ | The energy consumption of idle VMs |
| $BE(X)$ | The PM basic energy consumption |
| $SE(X)$ | The total switch energy consumption |
| $D(X)$ | The downtime in the VM migration operation |
| RU | The resource utilization rate |

2.2 Energy consumption analysis

The VMs which are occupied by tasks are running, and let $AE(X)$ be the VM execution energy consumption caused by the task execution with the scheduling policy X . The energy consumption of active VMs with X for the m -th VM v_m in V is calculated by

$$AE(X) = \sum_{m=1}^M \theta_m \cdot a_m(X) \cdot T_m(X) \quad (1)$$

where θ_m is the requested number of VM instances for v_m , $a_m(X)$ is the energy consumption rate of running VM instances, and $T_m(X)$ is the running time of v_m .

$I_m^n(X)$ is a binary variable to determine whether v_m is placed on p_n with policy X , and its calculation expression is

$$I_m^n(X) = \begin{cases} 1, & \text{if } x_m = p_n, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

According to the condition of v_m , we calculate the energy consumption of idle VMs by

$$IE(X) = \sum_{n=1}^N \sum_{m=1}^M I_m^n(X) \cdot \theta_m \cdot (\tau_n(X) - T_m(X)) \cdot \beta_m(X) \quad (3)$$

where $\beta_m(X)$ is the energy consumption rate of v_m in idle mode, $\tau_n(X)$ is the maximum of VMs running time.

The PM in operation has basic energy consumption regardless of whether it is idle or running. The calculation expression of PM basic energy consumption is

$$BE(X) = \sum_{n=1}^N \tau_n(X) \cdot \gamma_n(X) \quad (4)$$

where $\gamma_n(X)$ is the basic energy consumption rate of p_n .

Faced with the emergence of network hotspots and problem of overloading, the switches and hosts are arranged by the fat-tree topology in the cloud data center. Owing to the fat-tree topology, the loads that aggregate at the core layer are processed and diverted timely by the multiple links to the core layer.

The switches in the fat-tree topology are classified in three layers including core, aggregation and edge. The fat-tree topology is displayed in Figure 1. The topology rules of fat-tree are as follows: the number of pods contained in the topology is k , the number of connected server per pod is $(\frac{k}{2})^2$, the number of edge switches and aggregate switches within each pod is $\frac{k}{2}$, the number of core switches is $(\frac{k}{2})^2$, and the number of ports per switch in the network is k .

Let p_a be the source PM and p_b be the goal PM. Denote $l_{a,b}$ as a binary variable to judge whether p_a needs to access p_b , which is measured by

$$l_{a,b} = \begin{cases} 1, & \text{if } p_a \text{ needs to access } p_b, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Let $f_{a,b}$ be the frequency for p_a to access p_b , and the total access time is denoted by

$$K = \sum_{a=1}^N \sum_{b=1}^N l_{a,b} \cdot f_{a,b} \quad (6)$$

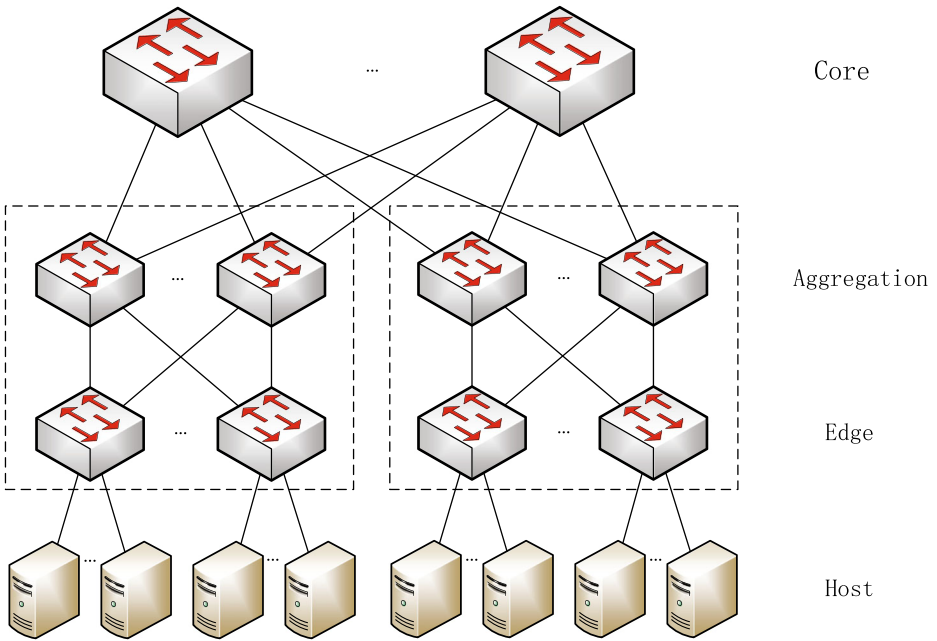


Figure 1 The fat-tree topology

In the fat-tree topology, the access time between PMs depends upon the distributed locations among the PMs. Let $ES(X)$, $AS(X)$ and $CS(X)$ be the corresponding switch that need to be accessed according to the scheduling strategy X in the edge layer, aggregation layer and core layer respectively. The accommodating situations in the fat-tree are classified into four conditions. First, The application is processed in the same PM due to the scheduling strategy, which is denoted as $y_a \neq x_a$. Then, x_a and y_a are linked to the same edge switch and different hosts, which is denoted as $y_a \neq x_a, ES(y_a) \neq ES(x_a)$. Besides, x_a and y_a are connected to the same aggregation instead of the same edge switch and host, which is denoted by $y_a \neq x_a, ES(y_a) \neq ES(x_a), AS(y_a) \neq AS(x_a)$. At last, x_a and y_a are linked to the same core switch instead of hosts, edge and aggregation switch, which is denoted by $y_a \neq x_a, ES(y_a) \neq ES(x_a), AS(y_a) \neq AS(x_a), CS(y_a) \neq CS(x_a)$. After analyzing all the possible accommodating situation, the access time for x_a to access y_b is calculated as

$$t_{a,b}(X) = \begin{cases} 0, & \text{if } x_a = y_a, \\ 2 \frac{D_m}{B_{SE}} \cdot l_{a,b}, & \text{if } ES(x_a) = ES(y_a), \\ 2 \left(\frac{D_m}{B_{SE}} + \frac{D_m}{B_{EA}} \right) \cdot l_{a,b}, & \text{if } AS(x_a) = AS(y_b), ES(x_a) \neq ES(y_a), \\ 2 \left(\frac{D_m}{B_{SE}} + \frac{D_m}{B_{EA}} + \frac{D_m}{B_{AC}} \right) \cdot l_{a,b}, & \text{if } CS(x_a) \neq CS(y_a), AS(x_a) \neq AS(y_a), \\ ES(x_a) = ES(y_a). \end{cases} \quad (7)$$

where D_m is the memory used by vm, and B_{SE} is the available network bandwidth between hosts and edge switches, B_{EA} is the available network bandwidth between edge switches and aggregation switches, and B_{AC} is the available network bandwidth between aggregation

switches and core switches. The number of switches for p_a accessing p_b in the topology is denoted by

$$t_{a,b}(X) = \begin{cases} 0, & \text{if } x_a = y_a, \\ 1, & \text{if } ES(x_a) = ES(y_a), \\ 3, & \text{if } AS(x_a) = AS(y_b), ES(x_a) = ES(y_a), \\ 5, & \text{if } CS(x_a) = CS(y_a), AS(x_a) \neq AS(y_a), ES(x_a) = ES(y_a). \end{cases} \tag{8}$$

During the scheduling process, each switch employed in the fat-tree has a base energy consumption SE_{base} which is calculated by

$$SE_{base}(X) = \frac{5}{4}k^2 \cdot \tau_n(X) \cdot \beta, \tag{9}$$

Then, the transmission energy consumption is calculated by accommodation situation in the above analysis. Based on the different scheduling strategy, the transmission energy SE_{ex} consumption is calculated by

$$SE_{ex}(X) = \sum_{a=1}^N \sum_{b=1}^N l_{a,b} \cdot f_{a,b} \cdot t_{a,b}(X) \cdot \delta_{a,b}(X) \cdot \gamma, \tag{10}$$

Then, the switch energy consumption is calculated by

$$SE(X) = SE_{base}(X) + SE_{ex}(X), \tag{11}$$

where β is the baseline energy consumption rate for each switch, and γ is the energy consumption rate for each port, $t_{a,b}(X)$ is the access time for p_a to access p_b , and $\delta_{a,b}(X)$ is the total number of switches in the topology.

In this way, the total energy consumption denote as $E(X)$ is calculated by

$$E(X) = AE(X) + IE(X) + BE(X) + SE(X). \tag{12}$$

2.3 Downtime analysis

The downtime in the VM migration operation contains switch time and the access time of the log file. Suppose in the migration operation of v_m with X , the memory image is transmitted $I_m(X)$ times. Let the access time of all the remaining log files be $MT_m^{i_m}(X)$ when the log file transfers at $i_m(X)$ time. The downtime when v_m is migrated from p_n with X is calculated by

$$MT_m^{i_m}(X) = \sum_{n=1}^N \sum_{m=1}^M I_m^n(X) \cdot M_m^n(X) \cdot \frac{D_m^{i_m}(X)}{B_{m,n}}, \tag{13}$$

where $D_m^{i_m}(X)$ is the size of dirty page transferred by v_m , and $B_{m,n}$ is the bandwidth between y_m and p_n . In order to keep the consistency of VM memory condition before and after the migration, the dirty pages produced in the process of memory transmission are sent to the goal PM during the next transmission. Hence, the size of dirty page transferred by v_m could be calculated by

$$D_m^{i_m}(X) = \begin{cases} S_m(X), & \text{if } i = 0, \\ R_m(X) \cdot MT_m^{i_m-1}(X), & \text{otherwise.} \end{cases} \tag{14}$$

where $S_m(X)$ is the size of the mirror memory of v_m , and $R_m(X)$ is the producing rate of memory dirty page. The switch time of v_m is calculated by

$$OT_m(X) = \sum_{n=1}^N \sum_{m=1}^M I_m^n(X) \cdot M_m^n(X) \cdot 2\zeta_m(X), \quad (15)$$

Similarly, the computation latency is dismissed because of the same number and computing rate of occupied VMs. Then, the total downtime is calculated by

$$D(X) = \sum_{i_m=1}^{I_m} MT_m^{i_m}(X) + OT_m(X). \quad (16)$$

2.4 Resource utilization analysis

In the cloud data center, multiple VM instances are created to allocate resources. The resource requirements could be quantified by the number of VM instances. Let c_n be the capacity of n -th PM. The resource utilization with X $u_n(X)$ is calculated by

$$u_n(X) = \frac{1}{c_n} \cdot \sum_{m=1}^M \theta_M \cdot I_m^n(X). \quad (17)$$

Let K_n be the flag to judge whether the p_n is running, which is measured by

$$K_n = \begin{cases} 0, & \text{if } \sum_{m=1}^M I_m^n(X) \cdot L_m = 0, \\ 1, & \text{otherwise.} \end{cases} \quad (18)$$

where L_m is a binary variable to determine where v_m hosts a load, and its calculation expression is

$$L_m = \begin{cases} 1, & \text{if } v_m \text{ hosts a load,} \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

So, the total number of running PMs is calculated by

$$MP = \sum_{n=1}^N K_n. \quad (20)$$

The resource utilization rate is calculated by

$$RU(X) = \frac{1}{MP} \cdot \sum_{n=1}^N u_n(X). \quad (21)$$

2.5 Problem definition

In this paper, we focus on the QoS-aware VM scheduling method to reduce the energy consumption, downtime and the resource utilization, and the VM scheduling problem is defined by

$$\min(E(X)), \min(D(X)), \max(RU(X)). \quad (22)$$

$$s.t. x_m \in P, \tag{23}$$

$$\sum_{m=1}^M \theta_M \cdot I_m^n(X) \leq c_w, \tag{24}$$

$$\sum_{m=1}^M I_m^n(t) \cdot L_m \leq M. \tag{25}$$

3 A QoS-aware VM scheduling method for energy conservation in cloud-based CPS

As we discuss in the Section 2, the QoS-aware VM scheduling problem is a multi-objective optimization problem. NSGA-III is adopted for its efficient and accurate performance when solving optimization problem with multiple objectives ranging from three to fifteen. First in this section, the VM scheduling strategies are encoded and the fitness functions are given. Then, NSGA-III algorithm is adopted to find the optimal solution. Finally, a method overview is described.

3.1 Encoding

In this section, we encode for the VM scheduling strategies. In the genetic algorithm (GA), a gene represents a scheduling strategy of an application. A chromosome which represents a set of scheduling strategy of VMs is composed of a group of genes. The value of the scheduling strategy which is the location of PMs encodes as 1, 2, 3, ..., N. An encoding example of scheduling method for VMs in V with N PMs is displayed in Figure 2. As is shown in Figure 2, v₁ is migrated from p₁ to p₂, and v₂ is migrated from p₃ to p₁, and v₃ is migrated from p_N to p₄, and v_m is migrated from p₄ to p_N.

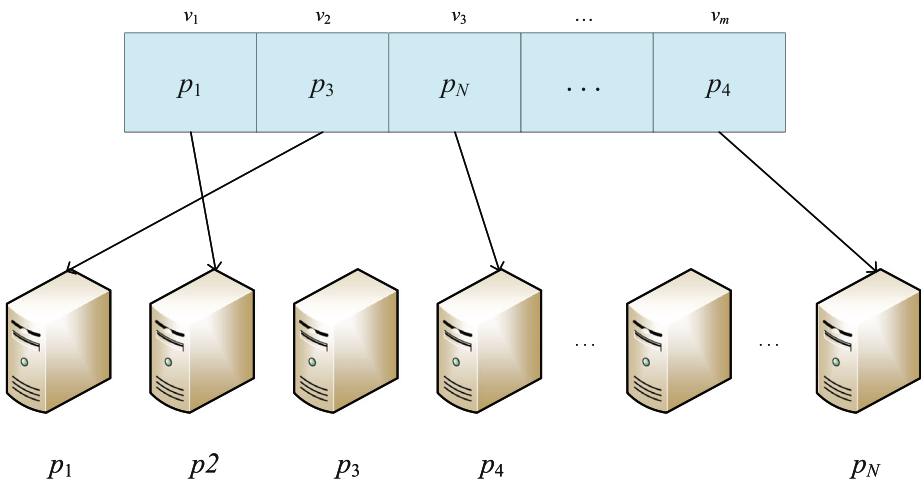


Figure 2 An instance of encoding

3.2 Fitness function

In GA, fitness function is used to determine whether a solution is efficient. A chromosome is the scheduling strategy of all the VMs in a schedule. In the paper, the NSGA-III algorithm utilizes the energy consumption and resource utilization as the fitness functions to find the optimal scheduling strategies. The fitness functions are given by (11), (15) and (21) respectively.

The energy consumption is the first fitness function, Algorithm 1 specifies the procedure of calculating the total energy consumption. In this algorithm, we input the fat-tree network topology, the number of VMs M , the number of PMs N , the encoding result of VM scheduling strategy c_s , the VM original deployment Y and the scheduling strategy X . We calculate the total energy consumption including the energy consumption of active VMs, the energy consumed by the idle VMs, the basic energy consumption of PMs, and the switch energy consumption in the algorithm. Finally, the outputs are the total energy consumption for the scheduling strategy.

Algorithm 1 Computing energy consumption.

Require: Fat-tree network topology, M , N , X , Y

Ensure: $E(X)$

```

1: for  $m = 1$  to  $M$  do
2:   Computing  $AE(X)$  by (1)
3:   for  $n = 1$  to  $N$  do
4:     if  $x_m = p_n$  then
5:       Computing  $IE(X)$  by (3)
6:     end if
7:   end for
8: end for
9: for  $n = 1$  to  $N$  do
10:  Computing  $BE(X)$  by (4)
11: end for
12: for  $a = 1$  to  $N$  do
13:  for  $b = 1$  to  $M$  do
14:    if  $p_a$  needs to access  $p_b$  then
15:      Computing  $t_{a,b}(X)$  by (7)
16:      Computing  $\delta_{a,b}(X)$  by (8)
17:      Computing  $SE(X)$  by (11)
18:    end if
19:  end for
20: end for
21: Computing  $E(X)$  by (12)
22: return  $E(X)$ 

```

The downtime of VM migrations is another fitness function, Algorithm 2 specifies the procedure of calculating the downtime of VM migration. In this algorithm, the inputs are the number of VMs M , the number of PMs N , the VM scheduling strategy X , and the VM original deployment Y . The downtime in the VM migration operation contains switch time and the access time of the log file. The access time of the log file is calculated (Lines 1-5), and the switch time is calculated (Lines 6-10). Then we get the total downtime (Line 10).

Algorithm 2 Computing downtime.

Require: M VMs allocated on N PMs with X
VM original deployment Y

Ensure: Downtime

```

1: for  $n = 1$  to  $N$  do
2:   for  $m = 1$  to  $M$  do
3:     Computing  $MT_m^{i_m}(X)$  by (13)
4:   end for
5: end for
6: for  $n = 1$  to  $N$  do
7:   for  $m = 1$  to  $M$  do
8:     Computing  $OT_m(X)$  by (15)
9:   end for
10: end for
11: for  $i_m = 1$  to  $I_m$  do
12:   Computing  $D(X)$  by (16)
13: end for
14: return  $D(X)$ 

```

The resource utilization of PMs is also a fitness function, algorithm 3 specifies how to calculate the average resources utilization of PMs used. The inputs are M VMs to be scheduled, N PMs, the VM scheduling strategy X , and the VM original deployment Y . In this algorithm, we first calculate the total resource utilization, then calculate the resource utilization rate of PMs used.

Algorithm 3 Computing resources utilization.

Require: M, N, X, Y

Ensure: $RU(X)$

```

1:  $u = 0$ 
2: for  $m = 1$  to  $N$  do
3:   for  $n = 1$  to  $M$  do
4:     if  $x_m = p_n$  then
5:       calculate  $u_n(X)$  by (17)
6:        $u = u + u_w(X)$ 
7:     end if
8:   end for
9: end for
10: for  $m = 1$  to  $N$  do
11:   for  $n = 1$  to  $M$  do
12:     Computing  $MP$  by (20)
13:   end for
14: end for
15: Computing  $RU(X)$  by (21)
16: return  $RU(X)$ 

```

According to what we discuss in the Section 2, in order to achieve QoS enhancement, we are supposed to reduce the energy consumption and downtime and optimize the resource

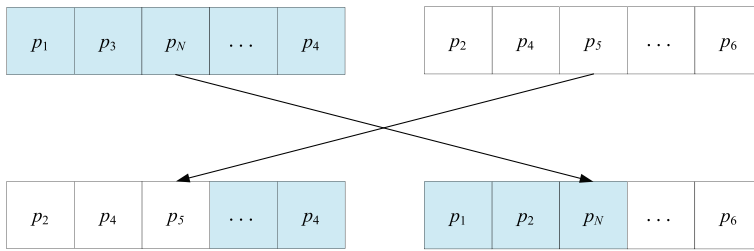


Figure 3 An instance of crossover

utilization with the constraints given by (23), (24) and (25). The fitness function optimization problem is a multi-objective optimization problem, and NSGA-III algorithm is used to select the best scheduling policy in the paper.

3.3 Optimize the VM scheduling strategy using NSGA-III algorithm

First, the parameters of the GA are initialized, such as the size of the population. The crossover is to combine the two parental chromosomes in the population, trying to get better offspring chromosomes. The number of iterations IT , the possibility of crossover and mutation are denoted by P_c , P_d . Each chromosome consists of the VM scheduling policies which are denoted by c_s . The chromosome in the s -th schedule is denoted as $C_{s,e} = \{c_{s,e} \mid 1 \leq s \leq N_{pop}, 1 \leq e \leq M\}$.

The parent population P_t of size N_{pop} is randomly initialized in the specified value range, and then an offspring population Q_t which has a size of N_{pop} is generated by crossover and mutation operators. The crossover is to combine the two chromosomes in the parent population to generate new pair of chromosomes. As is shown in Figures 2 and 3, the crossover point is determined firstly, and the genes around the crossover points are changed.

The mutation is to modify some genes of chromosome and generate new individuals with the aim of preventing early convergence. Figure 4 shows a instance of mutation operation in one schedule strategy.

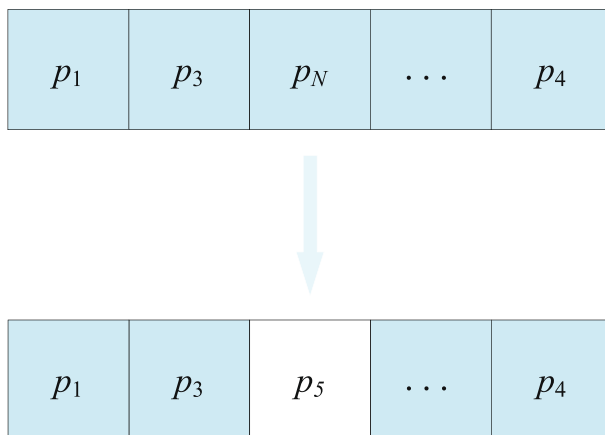


Figure 4 An instance of mutation

Then, populations P_t and Q_t are combined as population R_t of size $2N_{pop}$. Similar to the basic framework of the NSGA-II algorithm, the non-dominated sorting is used to divide the population R_t by non-domination levels. Then, each chromosome is added to a new population S_t , until the size of S_t is equal to N_{pop} . For further selection, the ideal points and extreme points are determined first to normalize the objective value, and compute the reference points accordingly. The ideal points are calculated by the minimum for three fitness functions (E_{min} , D_{min} , RU_{min}) in S_t . The objective value is translated by subtracting the minimum from the fitness function and then the ideal points of S_t become a zero vector. The process could be denoted by

$$E^a(X_i) = E(X_i) - E_{\min}(X_i \in S_t) \quad (26)$$

$$D^a(X_i) = D(X_i) - D_{\min}(X_i \in S_t) \quad (27)$$

$$RU^a(X_i) = RU(X_i) - RU_{\min}(X_i \in S_t) \quad (28)$$

The extreme points are identified by the solutions that make the achievement scalarizing function minimum. Let ϵ^E , ϵ^D and ϵ^R be the extreme values of energy consumption, downtime and resource utilization. The extreme values are calculated by

$$\epsilon^E = \max \frac{E^a(X_i)}{W_E} (X_i \in S_t) \quad (29)$$

$$\epsilon^D = \max \frac{D^a(X_i)}{W_D} (X_i \in S_t) \quad (30)$$

$$\epsilon^R = \max \frac{RU^a(X_i)}{W_{RU}} (X_i \in S_t) \quad (31)$$

where W_E , W_D and W_{RU} are the weight factors of the three fitness functions. For each objective function, we could get an extreme vector to construct a hyper-plane. Hence, the intercept of each objective axis could then be computed as a_E , a_D , a_{RU} . The objective functions are normalized as

$$E^n(X_i) = \frac{E(x) - E_{\min}}{a_E} (X_i \in S_t) \quad (32)$$

$$D^n(X_i) = \frac{D(x) - D_{\min}}{a_D} (X_i \in S_t) \quad (33)$$

$$RU^n(X_i) = \frac{RU(x) - RU_{\min}}{a_{RU}} (X_i \in S_t) \quad (34)$$

According to the new constructed hyper-plane, the reference points are simply placed onto the normalized hyper-plane. Each axis has an intercept of 1 and is divided into g parts along each objective. The total number of reference points is C_{2M+g-1}^g .

After normalizing each objective function adaptively, we need to associate population individuals with reference points. We count the number of population members are associated with reference points. The solutions in the non-dominated front F_l are sorted by the number of reference points associated with solutions. A solution is selected randomly from the solutions that associated with the maximum reference points each time. The selection procedure is repeated until the size of S_t is equal to N_{pop} for the first time. Finally, the N_{pop} strategies for the child population are selected.

3.4 Selecting scheduling strategy using MCDM and SAW

Our method is aimed at realizing the trade-off among energy consumption, downtime and resource utilization. Each child population selected above contains N_{pop} chromosomes which represent scheduling strategies of VMs. To select the relatively optimal scheduling strategy, MCDM and SAW are employed. The higher energy consumption of scheduling strategy means the solution is worse. Therefore, energy consumption is a negative criterion for our scheduling strategy. We normalize the energy consumption as

$$F(E(X_i)) = \begin{cases} \frac{E^{\max} - E(X_i)}{E^{\max} - E^{\min}}, & E^{\max} - E^{\min} \neq 0, \\ 1, & E^{\max} - E^{\min} = 0, \end{cases} \tag{35}$$

where E^{\max} and E^{\min} are the maximum and minimum of energy consumption of X_i scheduling strategy. Similarly, the downtime and resource utilization can be normalized by

$$F(D(X_i)) = \begin{cases} \frac{D^{\max} - D(X_i)}{D^{\max} - D^{\min}}, & D^{\max} - D^{\min} \neq 0, \\ 1, & D^{\max} - D^{\min} = 0, \end{cases} \tag{36}$$

$$F(RU(X_i)) = \begin{cases} \frac{RU^{\max} - RU(X_i)}{RU^{\max} - RU^{\min}}, & RU^{\max} - RU^{\min} \neq 0, \\ 1, & RU^{\max} - RU^{\min} = 0, \end{cases} \tag{37}$$

In addition, to calculate the utility value of each solution, the weight of each objective function requires determination. The utility value in the scheduling strategy X_i is calculated by

$$V(X_i) = w_E \cdot F(E(X_i)) + w_D \cdot F(D(X_i)) + w_{RU} \cdot F(RU(X_i)), \tag{38}$$

where $V(X)$ represents the utility value of the X scheduling strategy and w_E, w_D, w_{RU} are the weight of three objectives relatively. After calculating all the utility values of N_{pop} solutions, the maximum utility value is calculated by

$$V_{op} = \max_{i=1}^{N_{pop}} V(X_i). \tag{39}$$

The solution X_{op} with the maximum utility value V_{op} is selected as the most optimal scheduling strategy.

3.5 Method overview

Our goal is to minimize the downtime and the energy consumption in this paper. The VM scheduling problem is defined as a multi-objective optimization problem and NSGA-III is utilized to find the optimal scheduling strategy. The overview of our method is displayed in Algorithm 4. The input of the algorithm 4 are the initial population P_t and the number of iterations IT . The algorithm outputs the optimal VM scheduling strategy. We generate the population R_t of size $2N_{pop}$ by combining P_t and Q_t . (Lines 2 and 3). Then, we primarily select the child population by non-dominated sorting (Lines 4 to 9). Moreover, we select the strategies further by reference points. (Lines 10 to 15) Finally, we obtain the optimal scheduling strategy X_n .

Algorithm 4 Finding the optimal VM scheduling strategy**Require:** H, P_t, IT **Ensure:** V_{op}

```

1: for  $it = 1$  to  $IT$  do
2:    $Q_t = \emptyset, i = 1$ 
3:    $Q_t = \text{Crossover and mutation}(P_t)$ 
4:    $R_t = P_t \cup Q_t$ 
5:    $(F_1, F_2, \dots) = \text{Non-dominated sorting}(R_t)$ 
6:    $Q_t = \emptyset, i = 1$ 
7:   while  $P_{t+1}$  has enough space to include  $F_i$  do
8:      $S_t = S_t \cup F_i$ 
9:      $i = i + 1$ 
10:  end while
11:  if  $P_{t+1}$  has enough space to include  $F_i$  then
12:    Normalize objectives and generate reference points
13:    Associate each population member of  $S_t$  with a reference point
14:    Choose remaining  $K$  members one at a time from  $F_t$  to construct  $P_{t+1}$ 
15:  end if
16:   $it = it + 1$ 
17: end for
18: Pick out the optimal scheduling strategy  $X_{op}$  using MCDM and SAW
19: return  $X_{op}$ 

```

4 Experimental evaluation

4.1 Simulation setup

In our simulation, four datasets with different scales of the applications are applied, and the number of applications is set to 50, 100, 150 or 200. The specified parameter settings in this experiment are illustrated in Table 2.

4.2 Performance analysis of QVMS

As displayed in Figure 5, when the number of applications is 50, 100, 150, or 200, the number of generated solutions by QVMS is 3, 3, 3 and 4 respectively. After comparing the utility value given in (36), we obtain the most optimal schedule strategy among all the generated solutions. The solution with the maximum utility value is regarded as the most

Table 2 Parameter settings

| Parameter description | Value |
|---|-------|
| The number of applications M | 40 |
| The number of PMs N | 400 |
| The number of pods in fat-tree topology k | 24 |
| The baseline energy consumption rate of n -th PM $\gamma_n (X)$ | 342W |
| The active energy consumption rate of m -th VM $\alpha_m (X)$ | 23W |

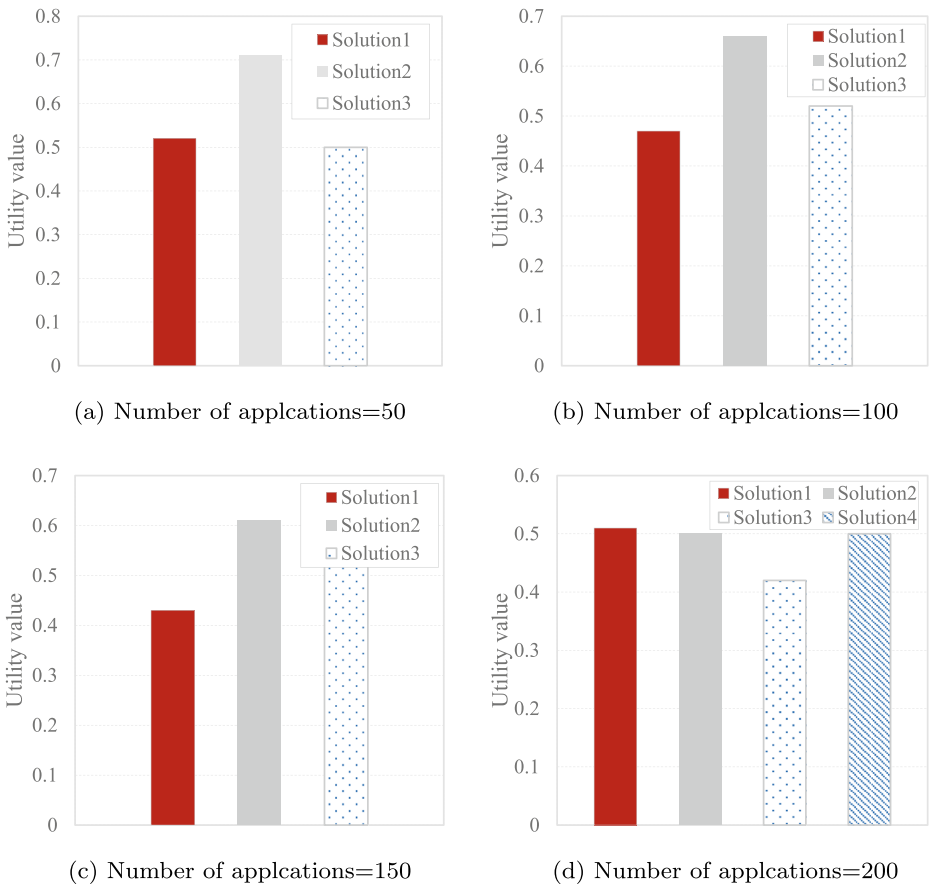


Figure 5 Comparison of utility value of the solutions generated by QVMS with different application sets

optimal schedule strategy. For example, the solution 2 in Figure 5b has a obviously higher utility value than the other two solutions and is the optimal solution in all 3 solutions.

4.3 Comparison analysis

In this subsection, the comparisons before and after employing our proposed QVMS method with the same experimental context are analyzed in detail. The resource utilization, the downtime and the energy consumption are the main metrics for evaluating the performance of the VM scheduling method. To analyze the advantage of QVMS, several scheduling methods are employed to contrast with. The contrast methods are introduced as follows.

- **Benchmark:** The applications are scheduled to the nearest PM by the shortest path algorithm until the current PM cannot satisfy the resource requirement of the application to be scheduled. Then, the remaining applications are offloaded to the next close PM. The process is repeated until all the applications are offloaded.
- **ESM (Energy-aware VM scheduling method):** ESM is an energy-aware dynamic VM scheduling method in clouds presented by Dou et al. [8]. The method includes two VM

offloading phases where applications are offloaded with lower energy consumption or higher performance.

4.3.1 Comparison of resource utilization

Figure 6 shows the comparison of the resource utilization of the PMs among Benchmark, ESM and QVMS with different application scales. It is intuitive from Figure 6 that our proposed strategy QVMS achieves high and stable resource utilization. That is, QVMS reduces the number of idle VMs and wastes less resources.

4.3.2 Comparison of energy consumption

In the experiment, the energy consumption consists of four parts, active energy consumption of VMs, the idle energy consumption of VMs, the basic energy consumption of PMs and the energy consumption of switches. The energy consumption of VMs, PMs and switches are displayed in Figures 7, 8 and 10 relatively. As is shown Figure 7, the energy consumption of VMs of all the three method is nearly the same. Besides, QVMS has an obvious advantage in the energy consumption of switches. Figure 8 shows that QVMS consume less energy than Benchmark and ESM. Figure 9 shows the comparison of running PMs, our QVMS only has fewer than half of PMs in Benchmark, which means QVMS save more energy consumption and the basic energy consumption of PM is illustrated in Figure 10. The difference between QVMS and ESM in energy consumption of PMs is nearly the same as the number of running PMs. Finally, Figure 11 displays the total energy consumption. With the increase of the application scale, the gap between with or without the employment of QVMS is enlarged, which may because the QVMS employs less PMs and reduce the basic energy consumption of PMs, which save the total energy consumption to a great extent. ESM takes the better performance into consideration at times, which lead to a little bit higher energy consumption than QVMS on some conditions.

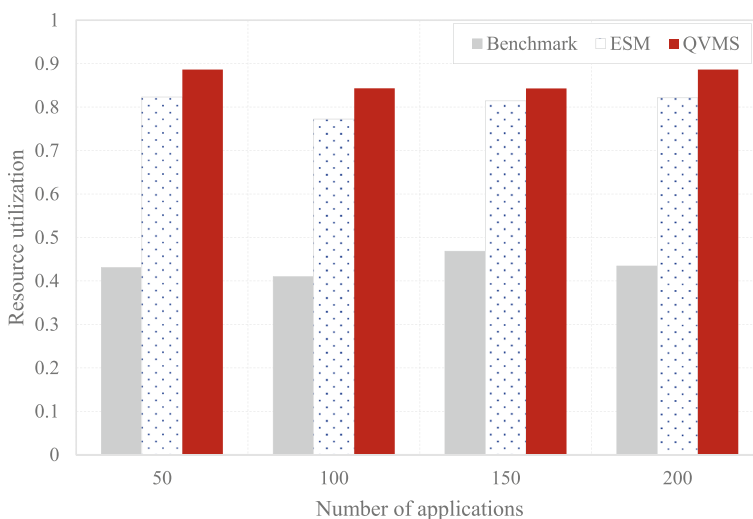


Figure 6 Comparison of average resource utilization with Benchmark and ESM

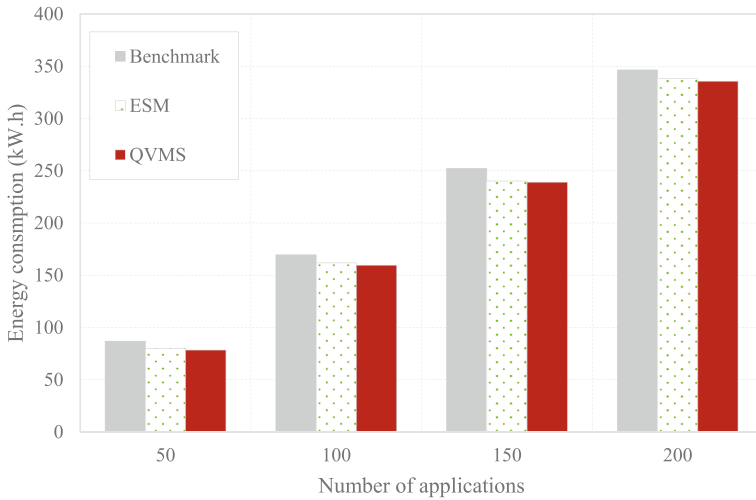


Figure 7 Comparison of the energy consumption of running VMs with Benchmark and ESM

4.3.3 Comparison of downtime

To improve the resource utilization, VM migrations are necessary between PMs, which results in the cost of downtime. According to the model, network topological distance is the key factor affecting the cost of downtime. We calculate the downtime consumption of all the migration compositions, based on the four application scales. From Figure 12, simulation results show that the downtime fluctuates around 35 seconds in the simulation environment and the downtime of ESM and Benchmark is a little longer than QVMS.

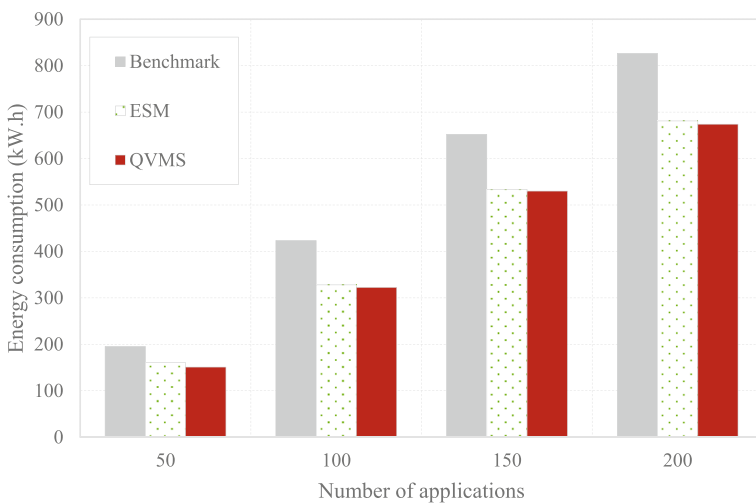


Figure 8 Comparison of the energy consumption of switches with Benchmark and ESM

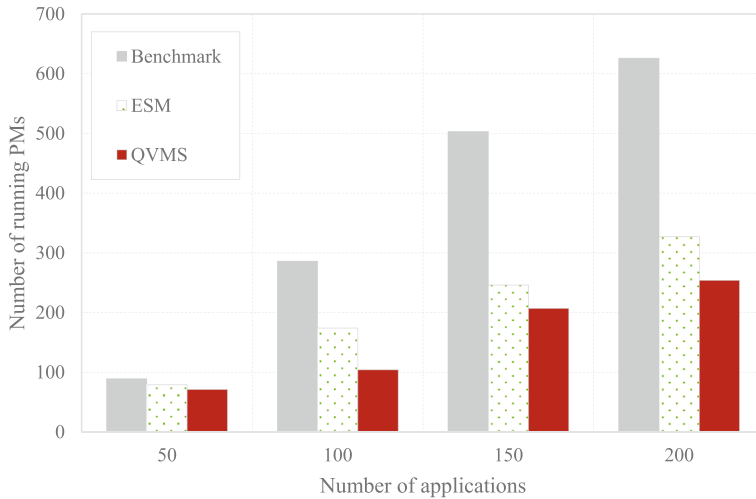


Figure 9 Comparison of the number of running PMs with Benchmark and ESM

5 Related work

With the development of manufacturing industry, CPS are systematically utilized to monitor and process the information between the physical factory layer and the cyber computing layer [1, 19, 21, 34]. In [19], a CPS architecture is proposed as an implement of CPS to improve efficiency, reliability and quality. The proposed CPS structure consists of two components including advanced connectivity and intelligent data management. Besides, to address the complexity, heterogeneity and multidisciplinary nature of industrial CPS, Akkaya et al. using aspect-oriented modeling to segregate aspects of expertise and manage the complexity [1]. In [34], Yu et al. analyze the potential of CPS that makes

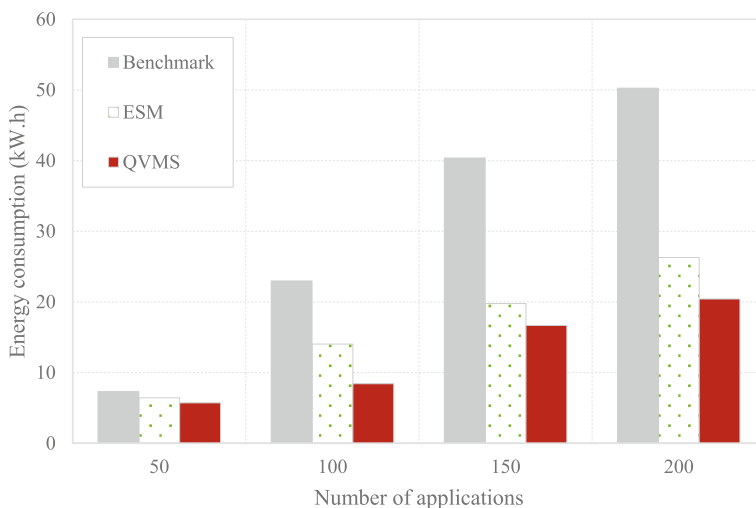


Figure 10 Comparison of the energy consumption of running PMs with Benchmark and ESM

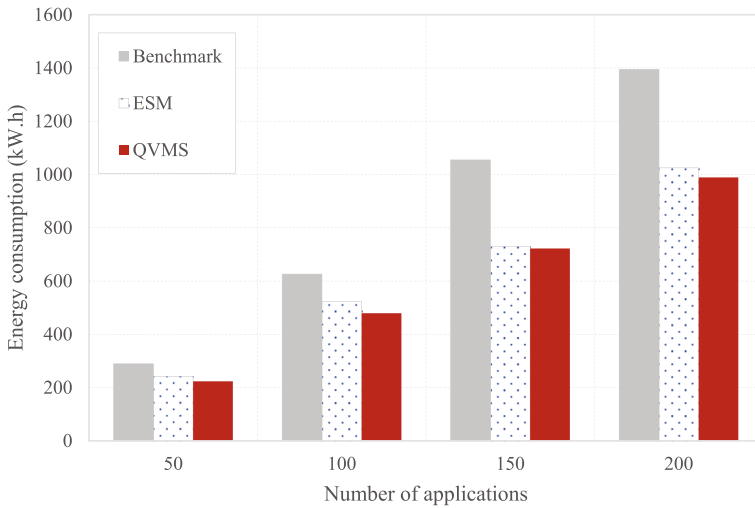


Figure 11 Comparison of the total energy consumption with Benchmark and ESM

contributions to the development of smart grid. By virtue of CPS, the efficiency of monitoring, controlling and communication is advanced. However, CPS are lack in real-time performance and dynamic conditions which pose great challenges to monitor industrial applications successfully. Lu et al. surveyed advances in real-time wireless network for industrial CPS including real-time scheduling algorithm, wireless cyber-physical simulation and cyber-physical co-design of wireless control systems [21].

Furthermore, offloading resource intensive applications to cloud has been a popular method to overcome the capacity limit of mobile devices [31, 33]. When offloading the applications, the characteristics of applications and quality of network should be taken

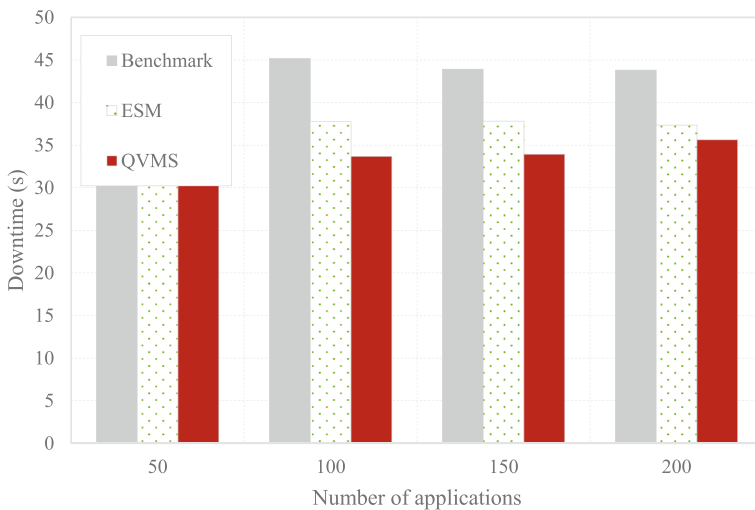


Figure 12 Comparison of the downtime with Benchmark and ESM

into consideration. In [38], Zhou et al. presented an offloading framework which is called mCloud consisting of nearby cloudlets and remote public cloud. By offloading to the mCloud, the context of mobile devices is used to provide an adaptive code offloading decision. However, when many users offloading their tasks to the cloud, they could produce interference to each other and cause long latency. Zheng et al. present a multi-user dynamic computation offloading method using game theory to make efficient and reasonable offloading decision [37]. Similarly, Hong et al. improved the Quality of Experience (QoE) from the perspective of users' context. To solve the problem of scheduling the offloaded data and selecting offloading services, a energy-latency-price trade-off is presented [15].

Combining CPS with cloud computing is an alternative approach for enhancing performance of CPS applications [9, 11, 18, 25, 35]. In [9], to address the challenge of resource management in cloud system, Gai et al. proposed a smart cloud-based optimizing workload model to assign tasks in heterogeneous clouds taken the sustainable factors into consideration. To reply to the emerging "Industry 4.0", the integration of cloud computing and CPS becomes significant. In [35], Yue et al. present a service-oriented industrial cyber-physical model. With the support of cloud and service application, the CPS enable a sustainable system and environmental friendly business. However, as the number of applications increasing, there are operational risks that CPS being attacked. Rahman et al. proposed a forensic-by-design framework for cloud-based CPS to protect CPS cloud system from being attacked [25]. Moreover, with the propose of addressing the lack of applications supporting monitoring and analysis of dynamic human activity in cloud-based CPS, Gravina et al. proposed activity as a service to support human activity recognition [11]. Kumar et al. proposed an intelligent and energy-efficient scheme in smart grid CPS by cloud-based control using game theory to realize efficient energy management [18].

In recent years, many researches focus on efficient approaches to reducing cloud data-centers' energy consumption [7, 23, 32, 39]. To address this problem, Chiang et al. proposed an efficient green control (EGC) algorithm which solves constrained optimization problems to improve the performance [7]. Zhu et al. proposed an energy-aware scheduling method named EARH for real-time and independent tasks [39]. EARH utilizes a rolling-horizon and is able to be integrated with other scheduling method. In [23], an energy and monetary cost-aware task scheduling model is proposed to offload multiple tasks to cloud. The scheduling model first find the optimal scheduling strategy for tasks, and then offer a reduction in the cost.

Dynamic VM integration is also a way to reduce energy consumption. A feasible way proposed by Chen et al. to handle this problem is reducing search space by selecting a suitable VM composition in the first step, and then abandoning on worse compositions. What's more, it can ensure optimality of VM integration [5, 20, 24].

In many cases, cloud data centers need service composition mechanisms because single VM can hardly meet all users' needs [6, 10, 12, 17, 28]. Jiang et al. proposed to choose the top k service composition because it can avoid the emergency such as the unavailability of best service composition [17]. Chen et al. think that QoS-aware service composition need to be formulated as a multi-objective optimization problem due to the requirements of QoS attributes. To find the optimal composition solutions, Chen et al. also proposed an approach based on Pareto set model which adjust the weight of different QoS attributes [6]. In [28], Shah et al. analyze the challenge of realizing QoS of health care service, and using CPS to combine real-time patient monitoring with data processing to enable QoS requirement.

To the best of our knowledge, there are few investigations, focusing on preserving QoS requirement as well as optimizing energy consumption when combining CPS with cloud

computing. The efficient VM scheduling method is needed to meet the QoS requirement when offloading CPS applications to the cloud.

6 Conclusion and future work

In this paper, a QoS-aware VM scheduling method for energy conservation in cloud-based CPS is proposed. First, we construct a systematic model with parameters such as downtime, resource utilization and energy consumption. Then, we see this model as a multi-objective optimization and solve it based on NSGA-III. Furthermore, the most optimal scheduling strategy is selected by MCDM and SAW. The method can be used to minimize downtime, energy consumption and maximize resource utilization. Through experimental evaluation, our method is proved to be effective. In the future, more QoS standards will be taken into consideration to meet multiple users' requirements. Furthermore, our proposed method will be adjusted ulteriorly to make it more practical for real life scenario.


Acknowledgements This research is supported by the National Science Foundation of China under grant no. 61702277 and no. 61872219.

References

1. Akkaya, I., Derler, P., Emoto, S., Lee, E.: Systems engineering for industrial cyber-physical systems using aspects. *Proc. IEEE* **104**(5), 997–1012 (2016)
2. Alam, K., Saddik, A.: C2ps: a digital twin architecture reference model for the cloud-based cyber-physical systems. *IEEE Access* **5**, 2050–2062 (2016)
3. Canali, C., Chiaraviglio, L., Lancellotti, R., Shojafar, M.: Joint minimization of the energy costs from computing, data transmission, and migrations in cloud data centers. *IEEE Trans. Green Commun. Netw.* **2**(2), 580–595 (2018)
4. Chen, X.: Decentralized computation offloading game for mobile cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **26**(4), 974–983 (2015)
5. Chen, Y., Huang, J., Lin, C., Hu, J.: A partial selection methodology for efficient QoS-aware service composition. *IEEE Trans. Serv. Comput.* **8**(3), 384–397 (2015)
6. Chen, Y., Huang, J., Lin, C., Shen, X.: Multi-objective service composition with QoS dependencies. *IEEE Trans. Cloud Comput.* (2016)
7. Chiang, Y., Ouyang, Y., Hsu, C.: An efficient green control algorithm in cloud computing for cost optimization. *IEEE Trans. Cloud Comput.* **3**(2), 249–262 (2015)
8. Dou, W., Xu, X., Meng, S., Zhang, X., Hu, C., Yu, S., Yang, J.: An energy-aware virtual machine scheduling method for service QoS enhancement in clouds over big data. *Concurrency and Computation: Practice and Experience*, 29(14), e3909 (2017)
9. Gai, K., Qiu, M., Zhao, H., Sun, X.: Resource management in sustainable cyber-physical systems using heterogeneous cloud computing. *IEEE Transactions on Sustainable Computing* **3**(2), 60–72 (2018)
10. Garcia-Valls, M., Bellavista, P., Gokhale, A.: Reliable software technologies and communication middleware: a perspective and evolution directions for cyber-physical systems, mobility, and cloud computing. *Futur. Gener. Comput. Syst.* **71**, 171–176 (2017)
11. Gravina, R., Ma, C., Pace, P., Aloï, G., Russo, W., Li, W., Fortino, G.: Cloud-based activity-aaservice cyber-physical framework for human activity. *Futur. Gener. Comput. Syst.* **75**, 158–171 (2017)
12. Gu, L., Zeng, D., Guo, S., Barnawi, A., Xiang, Y.: Cost efficient resource management in fog computing supported medical cyber-physical system. *IEEE Trans. Emerg. Top. Comput.* **5**(1), 108–119 (2017)
13. Hasan, M., Kouki, Y., Ledoux, T., Pazat, J.: When Green SLA becomes a possible reality in cloud computing. *IEEE Trans. Cloud Comput.* **5**(2), 249–262 (2017)
14. Hasan, M., Kouki, Y., Ledoux, T., Pazat, J.: When Green SLA becomes a possible reality in cloud computing. *IEEE Trans. Cloud Comput.* **5**(2), 249–262 (2017)
15. Hong, H., El-Ganainy, T., Hsu, C., Harras, K., Hefeeda, M.: Disseminating multilayer multimedia content over challenged networks. *IEEE Trans. Multimedia* **20**(2), 345–360 (2018)

16. Hossain, M., Malhotra, J.: Cloud-supported cyber–physical localization framework for patients monitoring. *IEEE Syst. J.* **11**(1), 118–127 (2017)
17. Jiang, W., Hu, S., Liu, Z.: Top K query for QoS-aware automatic service composition. *IEEE Trans. Serv. Comput.* **7**(4), 681–695 (2014)
18. Kumar, N., Zeadally, S., Misra, S.: Mobile cloud networking for efficient energy management in smart grid cyber-physical systems. *IEEE Wirel. Commun.* **23**(5), 100–108 (2016)
19. Lee, J., Bagheri, B., Kao, H.: A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters* **3**, 18–23 (2015)
20. Liu, Y., Liu, A., Guo, S., Li, Z., Choi, Y., Sekiya, H.: Context-aware collect data with energy efficient in Cyber–physical cloud systems, *Futur. Gener. Comput. Syst.* (2017)
21. Lu, C., Saifullah, A., Li, B., Sha, M., Gonzalez, H., Gunatilaka, D., Wu, C., Nie, L., Chen, Y.: Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proc. IEEE* **104**(5), 1013–1024 (2016)
22. Nir, M., Matrawy, A., St-Hilaire, M.: Economic and energy considerations for resource augmentation in mobile cloud computing. *IEEE Trans. Cloud Comput.* **6**(1), 99–113 (2018)
23. Nir, M., Matrawy, A., St-Hilaire, M.: Economic and energy considerations for resource augmentation in mobile cloud computing. *IEEE Trans. Cloud Comput.* **6**(1), 99–113 (2018)
24. Qi, L., Meng, S., Zhang, X., Wang, R., Xu, X., Zhou, Z., Dou, W.: An exception handling approach for privacy-preserving service recommendation failure in a cloud environment. *Sensors* **18**(7), 2037 (2018)
25. Rahman, N., Glisson, W., Yang, Y., Choo, K.: Forensic-by-design framework for cyber-physical cloud systems. *IEEE Cloud Computing* **3**(1), 50–59 (2016)
26. Rodriguez-Mier, P., Mucientes, M., Lama, M.: Hybrid optimization algorithm for large-scale QoS-aware service composition. *IEEE Trans. Serv. Comput.* **10**(4), 547–559 (2017)
27. Sadooghi, I., Martin, J., Li, T., Brandstatter, K., Maheshwari, K., Ruivo, T., Garzoglio, G., Timm, S., Zhao, Y., Raicu, I.: Understanding the performance and potential of cloud computing for scientific applications. *IEEE Trans. Cloud Comput.* **5**(2), 358–371 (2017)
28. Shah1, T., Yavari, A., Mitra, K., Saguna, S., Jayaraman, P., Rabhi, F., Ranjan, R.: Remote health care cyber-physical system: quality of service (QoS) challenges and opportunities. *IET Cyber-Physical Systems* **1**(1), 40–48 (2016)
29. Shu, Z., Wan, J., Zhang, D., Li, D.: Cloud-integrated cyber-physical systems for complex industrial applications. *Mobile Netw. Appl.* **21**(5), 865–878 (2016)
30. Wang, S., Lei, T., Zhang, L., Hsu, C., Yang, F.: Offloading mobile data traffic for QoS-aware service provision in vehicular cyber-physical systems. *Futur. Gener. Comput. Syst.* **61**, 118–127 (2016)
31. Xu, X., Dou, W., Zhang, X., Chen, J.: Enreal: an energy-aware resource allocation method for scientific workflow executions in cloud environment. *IEEE Trans. Cloud Comput.* **4**(2), 166–179 (2016)
32. Xu, X., Dou, W., Zhang, X., Hu, C., Chen, J.: A traffic hotline discovery method over cloud of things using big taxi GPS data. *Software: Practice and Experience* **47**(3), 361–377 (2017)
33. Xu, X., Zhao, X., Ruan, F., Zhang, J., Tian, W., Dou, W., Liu, A.: Data placement for privacy-aware applications over big data in hybrid clouds. *Secur. Commun. Netw.* **2017**, 1–15 (2017)
34. Yu, X., Xue, Y.: Smart grids: a cyber–physical systems perspective. *Proc. IEEE* **104**(5), 1058–1070 (2016)
35. Yue, X., Cai, H., Yan, H., Zou, C., Zhou, K.: Cloud-assisted industrial cyber-physical systems: an insight. *Microprocess. Microsyst.* **39**(8), 1262–1270 (2015)
36. Zhang, Y., Qiu, M., Tsai, C., Mehedi Hassan, M., Alamri, A.: Health-CPS: healthcare cyber-physical system assisted by cloud and big data. *IEEE Syst. J.* **11**(1), 88–95 (2017)
37. Zheng, J., Cai, Y., Wu, Y., Shen, X.: Dynamic computation offloading for mobile cloud computing, A stochastic game-theoretic approach. *IEEE Trans. Mobile Comput.* **18**(4), 771–786 (2018)
38. Zhou, B., Dastjerdi, A., Calheiros, R., Srirama, S., Buyya, R.: A context-aware offloading framework for heterogeneous mobile cloud. *IEEE Trans. Serv. Comput.* **10**(5), 797–810 (2017)
39. Zhu, X., Yang, L., Chen, H., Wang, J., Yin, S.hu., Liu, X.: Real-time tasks oriented energy-aware scheduling in virtualized clouds. *IEEE Trans. Cloud Comput.* **2**(2), 168–180 (2014)

Affiliations

Lianyong Qi¹ · Yi Chen² · Yuan Yuan³ · Shucun Fu² · Xuyun Zhang⁴ · Xiaolong Xu^{2,5,6} 

Lianyong Qi
lianyongqi@gmail.com

Yi Chen
yzyzchenyi@gmail.com

Yuan Yuan
yyuan@msu.edu

Shucun Fu
shucunfu@gmail.com

Xuyun Zhang
xuyun.zhang@auckland.ac.nz

- ¹ School of Information Science and Engineering, Qufu Normal University, Jining, China
- ² School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China
- ³ Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
- ⁴ Department of Electrical and Computer Engineering, University of Auckland, Auckland, New Zealand
- ⁵ Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China
- ⁶ State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China