Journal of Systems Architecture 127 (2022) 102512

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Multiserver configuration for cloud service profit maximization in the presence of soft errors based on grouped grey wolf optimizer☆

Peijin Cong [a], Xiangpeng Hou [a], Minhui Zou [a], Jiangshan Dong [b], Mingsong Chen [c], Junlong Zhou [a,d,e,*]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[b] Shanghai AI Laboratory, Shanghai 200062, China
[c] Ministry of Education Engineering Research Center of Software/Hardware Co-design Technology and Application, East China Normal University, Shanghai 200062, China
[d] State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[e] National Trusted Embedded Software Engineering Technology Research Center, East China Normal University, Shanghai 200062, China

ARTICLE INFO

Keywords:
Cloud computing
Multiserver configuration
Deadline miss rate
Soft error reliability
Profit

ABSTRACT

With the growing demand of cloud customers for computing resources, cloud computing has become more and more popular. As a pay-as-you-go model, cloud computing enables customers to use cloud services on demand anytime, anywhere over the Internet and it has become the backbone of modern economy. Obviously, profit maximization is especially important for cloud service providers (CSPs) in a competitive cloud service market. Extensive research papers have been conducted during the past few years for CSPs to optimize cloud service profit, whereas few of them considers the transient faults (resulting in soft errors) that may happen during service requests' execution and thus cause failed execution of these requests. In this paper, we study the multiserver configuration problem for cloud service profit maximization considering the deadline miss rate of service requests and the soft error reliability of the multiserver system. To solve the profit optimization problem, we first construct the models of multiserver system, deadline miss rate, and soft error reliability. Based on these models, we derive the models associated with cloud service revenue and cloud service costs. Then, we formulate the cloud service profit optimization problem and propose an effective grouped grey wolf optimizer (GWO)-based heuristic method that can determine the optimal multiserver configuration for a given customer demand to maximize cloud service profit. Experimental results show that the cloud service profit improvement achieved by our scheme can be up to 33.76% as compared with a state-of-the-art benchmark scheme.

1. Introduction

The cloud computing technology has revolutionized the computer science horizon and greatly promoted the development of IT business in the past decade [1,2]. It is also gradually changing people's daily lives by providing various cloud services to customers in a pay-as-you-go manner [3–7]. In the cloud service markets, there is a three-tier cloud service provisioning architecture that contains three major roles, i.e., customers, a CSP, and an infrastructure vendor. The CSP constructs its cloud service platform by renting the server resources from the infrastructure vendor and offers customers cloud services [8]. As a business model, cloud service profit maximization is a matter of particular concern to CSPs. In this paper, we study the multiserver configuration problem for a CSP to maximize its profit. The profit contains two parts, i.e., revenue and costs [2]. The revenue is usually influenced by factors

☆ This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB2101300, the National Natural Science Foundation of China under Grants 62172224, 61802185 and 61872147, in part by the Natural Science Foundation of Jiangsu Province, China under Grants BK20180470 and BK20190447, in part by the China Postdoctoral Science Foundation under Grants BX2021128, 2021T140327 and 2020M680068, in part by the Postdoctoral Science Foundation of Jiangsu Province, China under Grant 2021K066A, in part by the Open Research Fund of the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences under Grant CARCHA202105, in part by the Future Network Scientific Research Fund Project, China under Grant FNSRFP-2021-YB-6, and in part by the Open Research Fund of the National Trusted Embedded Software Engineering Technology Research Center (East China Normal University).

* Corresponding author at: School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.
E-mail address: jlzhou@njust.edu.cn (J. Zhou).

https://doi.org/10.1016/j.sysarc.2022.102512
Received 20 December 2021; Received in revised form 11 April 2022; Accepted 12 April 2022
Available online 21 April 2022

such as the amount of service requests and the service quality while the costs mainly come from the resource leasing expenditure paid to the hardware infrastructure provider and the electricity bill of energy consumed by the multiserver. Since the amount of service requests and the service quality are highly dependent on the configurations of the multiserver rented from the infrastructure provider, the cloud service profit is mainly determined by the multiserver configurations of the cloud service platform [9].

There are many papers focusing on multiserver configuration for cloud service profit maximization. For example, Cao et al. [10] studied the design of multiserver configurations to maximize profit. They modeled a multiserver as an M/M/m queuing system and used an analytical method to solve the profit optimization problem. Mei et al. [11] studied customer satisfaction-aware multiserver configuration to increase profit. They proposed a customer satisfaction measurement model and on top of that, they proposed a customer satisfaction-aware profit optimization scheme to determine the optimal multiserver configurations. The authors in [2,12] considered the fact of limited funding available in reality and solved the profit optimization problem under the funding constraint by proposing effective heuristic algorithms to optimize profit. Different from the above work, the authors in [9] adopted a dual-server renting mechanism to configure the multiserver system for profit maximization. Chiang and Ouyang [13] modeled the cloud service system as an M/M/R/K queuing system, in which all service requests exceeding the maximum capacity of the queue will be rejected. This work aims to find the optimal system configuration (i.e., server size and queue capacity) to maximize profit, whereas this strategy would lead to a decline in reputation and loss of future customers [14]. The authors in [8,15] introduced the customer perceived value (CPV) in psychology and designed CPV-based customer demand prediction models. Based on the models, they used the augmented Lagrange multiplier method to solve the multiserver configuration problem for profit optimization. Zhou et al. [7] studied the problem of minimizing makespan and monetary cost of scheduling workflows for CSPs, and proposed two efficient workflow scheduling schemes to address the problem.

Although the aforementioned methods are effective in optimizing cloud service profit, none of them considers the system reliability that ensures the correct execution of service requests and has non-negligible impacts on service quality. The service requests executing on cloud servers may suffer soft errors incurred by transient faults [16]. Transient faults appear in a short period of time and then disappear without damaging the hardware [17]. Generally, the power management techniques like dynamic voltage and frequency scaling for saving energy would cause an increased transient fault rate that adversely impacts reliability. Thus, to ensure a reliable execution of service requests submitted by customers, it is necessary to consider reliability in the optimization of energy consumption of cloud servers. Wu et al. [18] studied the task scheduling problem in dynamic voltage and frequency scaling-enabled cloud data centers considering the soft errors that may occur during task execution. To solve this, they presented a soft error-aware workflow scheduling approach to minimize the energy consumption of cloud data centers. However, only optimizing system energy consumption may not be able to achieve maximization of cloud service profit.

In this paper, we focus on the multiserver configuration problem that aims to maximize the cloud service profit considering both deadline miss rate and soft error reliability during the executions of service requests. The configuration problem involves the decision on the server size (i.e., the number of servers in the multiserver system) used to serve customers' service requests and the execution speed of the servers in the system. To solve the problem, we propose a grouped grey wolf optimizer (GWO)-based multiserver system configuration scheme. Our main contributions are summarized as follows.

- We model the deadline miss rate of service requests based on the probability distribution of requests' slack. Unlike the existing papers, our proposed deadline miss rate model considers the heterogeneity of cloud service requests and assumes the maximum waiting time of cloud service requests are different.
- We model the soft error reliability of service requests served on the multiserver system. Unlike the existing papers, our soft error reliability model is proposed for average-based scheduling that does not need real-time system information to schedule each service request and hence avoids high time and energy overheads.
- We consider the impact of deadline miss rate and soft error reliability on the revenue gained by processing service requests, and propose a new methodology to calculate the expected revenue based on the proposed deadline miss rate and soft error reliability models.
- We formulate the cloud service profit optimization problem considering the deadline miss rate of service requests and the system soft error reliability, and propose an effective grouped GWO-based multiserver system configuration scheme to solve the problem. Our scheme can avoid the local optimum as compared to the original grey wolf optimizer.

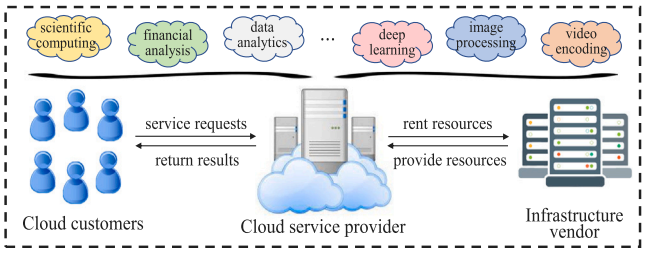


Fig. 1. Three-tire cloud service provisioning architecture.

Numerous simulations are carried out to verify the efficacy of our proposed method. Results show that our method is effective in configuring multiserver system for profit optimization as compared to the brute force search approach, and is superior to a state-of-the-art benchmark scheme in maximizing cloud service profit.

The rest of this paper is organized as follows. Section 2 gives the models and defines the studied problem. Section 3 introduces our proposed grouped grey wolf optimizer-based multiserver system configuration scheme. Section 4 validates our scheme through extensive experiments and analyzes the simulation results. Finally, Section 5 concludes the whole paper and gives the future work.

2. System models and problem definition

Consider a three-tier cloud service provisioning architecture that contains multiple customers, a CSP, and a infrastructure vendor. The architecture has been widely adopted in the existing literature [2,8,9]. As shown in Fig. 1, a CSP constructs its cloud service platform by renting the server resources from a infrastructure vendor, and provides customers with cloud services. Below, we first give the multiserver system model that has been widely adopted in the literature. We then construct the new models for evaluating the deadline miss rate of service requests and the soft error reliability of the multiserver system. Finally, the cloud service profit-related models are introduced. In particular, the methodology to calculate the revenue is developed based on our proposed deadline miss rate and soft error reliability models.

2.1. Multiserver system model

In a real-world cloud service platform (e.g., Amazon EC2 [19]), customers submit their service requests to a job queuing system (e.g., Sun Grid Engine [20] and Condor [21]), and the job queuing system adds

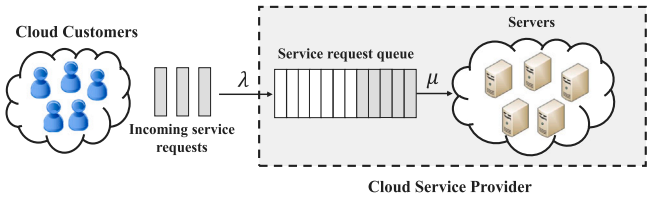


Fig. 2. The multiserver system model.

the requests into a queue, decide how to process the requests following a policy [22]. Thus, we generally abstract a cloud service platform as a multiserver model with a service request queue [9]. As illustrated in Fig. 2, the multiserver is modeled as an $M/M/m$ queuing system. The 1st M represents that the consecutive arrival intervals of customers' submitted service requests follow the memoryless Poisson process, i.e., the interarrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\lambda$ where λ refers to the service request arrival rate (i.e., the number of service requests arrived per second). The 2nd M indicates that the service time of a server follows a negative exponential distribution. The last m represents the number of servers which are able to handle service requests in parallel. In the figure, customers submit their service requests[1] to the CSP and the arriving service requests will be firstly admitted in the service request queue until they can be processed by any available server in the multiserver system. Like [8,9,12], the first-come-first-served (FCFS) policy is used and a queue with infinite capacity is maintained by the multiserver system.

The multiserver considered in this paper is composed of m homogeneous servers running at the same speed. The homogeneous multiserver model is simple yet effective and has been widely used in the literature [8–11,15]. It can be extended to heterogeneous multiserver systems using the configuration method proposed by Li et al. [12]. This method configures a heterogeneous cloud computing platform as multiple homogeneous multiserver systems. Each multiserver system is deployed with special software and is devoted to serve one type of service requests and applications. In such a multiserver system, the servers are homogeneous and execute at the same speed. Each multiserver can be treated as an $M/M/m$ queuing system which handles the requests following the FCFS discipline. Therefore, using the configuration method [12], our homogeneous multiserver model can be readily applied to heterogeneous cloud platforms. We leave the detailed discussion of this aspect to future work.

In the $M/M/m$ queuing system, the service requests' execution requirements, represented by r (Unit: billion instructions), are quantified by the number of instructions. The execution requirements r are i.i.d. exponential random variables with mean $\bar{r}$. Thus, the requests' service times t (Unit: second) are also i.i.d. exponential random variables, i.e., $t = r/s$ with mean $\bar{t} = \bar{r}/s$ where s (Unit: billion instructions/second) is the server speed. Let μ be a server's average service rate, i.e., the number of service requests that can be handled by a server per unit of time (UT), then μ can be obtained by $\mu = 1/\bar{t} = s/\bar{r}$. After obtaining μ, the server utilization ρ (represented by the ratio between the total number of service requests arrived per UT and the average service rate of a multiserver system per UT) can be calculated as $\rho = \lambda/m\mu = \lambda\bar{r}/(ms)$.

Based on the queuing system theory [23], let P_k represent the probability that k requests are waiting or processing in the queuing system. Then, we have [8,10–12]

$$P_k = \begin{cases} P_0 \cdot \frac{(m\rho)^k}{k!}, & k \leqslant m \\ P_0 \cdot \frac{m^m \rho^k}{m!}, & k \geqslant m \end{cases} \tag{1}$$

[1] Same as in [10,12], service requests with soft real-time requirements are considered in this paper.

where P_0 is the probability that there is no requests (waiting or being processed) in the multiserver system and it is expressed as [8,10–12]

$$P_0 = \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho}\right)^{-1} \tag{2}$$

Note that Eq. (1) holds only when $0 < \rho < 1$.

Customers are sensitive to request waiting time in the queue. However, some requests may wait for a long time before they can be processed due to the multiserver's limited computing capacity. Let $P_w(x)$ be the cumulative distribution function of the waiting time w of a request in the multiserver system, i.e., the probability that the request's waiting time w is less than x, then based on the queuing theory, $P_w(x)$ can be formulated as [9]

$$P_w(x) = 1 - \frac{P_m}{1-\rho} \cdot e^{-m\mu(1-\rho)x}, \tag{3}$$

where P_m represents the probability that there are m requests (waiting or being processed) in the queuing system. Based on Eqs. (1) and (2), P_m can be calculated as [9]

$$P_m = \frac{(m\rho)^m}{m!} \cdot \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho}\right)^{-1} \tag{4}$$

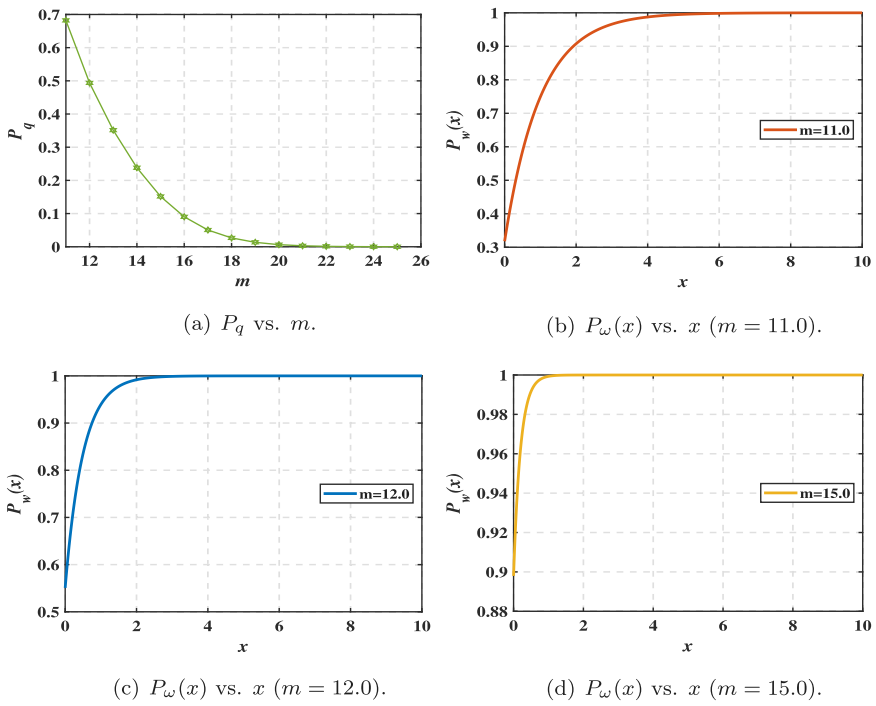
Here, we give an example to illustrate the above equations for better understanding. Let $\lambda = 10.0$, $\bar{r} = 1.0$, $s = 1.0$, we first analyze the relationship between the probability that a newly submitted service request must wait because all servers are busy (P_q) and the server size (m). Based on Eqs. (1) and (2), P_q can be derived as $P_q = \sum_{k=m}^{\infty} P_k = P_0 \frac{(m\rho)^m}{m!(1-\rho)}$. Fig. 3(a) shows the relationship between P_q and m when $\lambda = 10.0$, $\bar{r} = 1.0$, and $s = 1.0$. We can see that as m increases (i.e., more servers are available), the probability of queuing decreases gradually. The reason behind lies in that given customer demand (i.e., $\lambda\bar{r}$), more servers can serve more service requests per UT, thus the probability of service requests waiting to be executed will decrease accordingly.

Under the same parameter settings, Figs. 3(b)–3(d) plot the function $P_w(x)$ that represents the cumulative distribution function of the waiting time w of a service request in the multiserver system when $m = 11.0$, 12.0, and 15.0, respectively. From these figures, we can observe that the more servers are available in the multiserver system, the faster the system can process service requests. For example, when $m = 11.0$ (see Fig. 3(b)), the probability that a service request's waiting time is less than 1s is 0.7491 while when $m = 15.0$ (see Fig. 3(d)), the probability that the request's waiting time is less than 1s is 0.9993. Below, $P_w(x)$ will be used to model the service requests' deadline miss rate and the cloud service revenue.

2.2. Deadline miss rate and soft error reliability models

In this study, the service requests with soft real-time requirements are considered. That is to say, the service requests can tolerate occasional deadline misses during their executions [24]). As the mulitserver system's computing capacity is limited, some service requests may wait for a long time before they can be processed. That is to say, not all the arriving service requests can be handled in time by the CSP with limited computing capacity. Thus, given the arrived service requests, it is necessary to obtain the probability of the service requests to lose. Further, considering that the transient faults (resulting in soft error) may occur due to the effects of cosmic ray radiations or electromagnetic interference during the execution of service requests [17], which adversely impact quality of service (QoS), system reliability is also needed to be guaranteed. We model the deadline miss rate of service requests and the system soft error reliability as follows.

Deadline miss rate. Service requests with soft real-time requirements are usually characterized by slack [25] and deadline miss rate [26]. The slack is the maximal waiting time ($w_{\max}$) that a request can tolerate before it is processed while the deadline miss rate represents the ratio between the amount of requests missing deadline and the



(a) P_q vs. m. (b) $P_\omega(x)$ vs. x $(m = 11.0)$.

(c) $P_\omega(x)$ vs. x $(m = 12.0)$. (d) $P_\omega(x)$ vs. x $(m = 15.0)$.

Fig. 3. Performance analysis of the $M/M/m$ queuing system.

amount of requests arrived. Considering the heterogeneity of cloud service requests, we assume that the maximum waiting time of cloud service requests are different and follow a uniform distribution. Referring to [25], $w_{\max}$ is uniformly distributed in the interval $[\mathcal{T}_{\min}, \mathcal{T}_{\max}]$ where $\mathcal{T}_{\min}$ and $\mathcal{T}_{\max}$ are the minimal and the maximal values of the slack respectively. Thus, the probability density function (PDF) of $w_{\max}$ is expressed as

$$f(w_{\max}) = \begin{cases} 0, & w_{\max} < \mathcal{T}_{\min} \\ \frac{1}{\mathcal{T}_{\max} - \mathcal{T}_{\min}}, & \mathcal{T}_{\min} \leqslant w_{\max} \leqslant \mathcal{T}_{\max} \\ 0, & w_{\max} > \mathcal{T}_{\max} \end{cases} \tag{5}$$

Based on Eqs. (3) and (5), the service requests' deadline miss rate (DMR) can be derived as

$$\begin{aligned} DMR &= \int_{\mathcal{T}_{\min}}^{\mathcal{T}_{\max}} f(w_{\max}) \cdot (1 - P_w(w_{\max}))dw_{\max} \\ &= \frac{m\lambda^m(e^{-(\frac{ms}{\bar{r}}-\lambda)\mathcal{T}_{\min}} - e^{-(\frac{ms}{\bar{r}}-\lambda)\mathcal{T}_{\max}})}{(\frac{s}{\bar{r}})^{m-1}(\frac{ms}{\bar{r}} - \lambda)^2(\mathcal{T}_{\max} - \mathcal{T}_{\min})(m!\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{1-\rho})} \end{aligned} \tag{6}$$

Clearly, the smaller the DMR, the better the system performance and hence the higher the customer satisfaction. Based on Eq. (6), we can calculate the number of service requests that would be successfully executed by the system among all the arrival service requests.

Soft error reliability. Soft errors can be modeled by an exponential distribution with an average soft error arrival rate which represents the expected number of failures occurring per second [27]. According to the reliability model widely used in [27,28], the soft error rate ς_{se} of a server is formulated as

$$\varsigma_{\text{se}}(s) = \varsigma_0 \cdot e^{\frac{d(s_{\max}-s)}{s_{\max}-s_{\min}}}, \tag{7}$$

where ς_0 denotes the average error rate when the server is running at the maximal speed (i.e., $s = s_{\max}$) and d is a coefficient indicating the sensitivity of error rates to frequency scaling [27]. From Eq. (7) we can deduce that to decrease the soft error rate of a server, the CSP needs to increase server speed. But it leads to increased power consumption and costs.

The reliability of a service request is estimated as the probability of its successful execution without suffering from soft errors. Based on Eq. (7), the reliability of a service request is expressed as

$$R(m, s, r) = e^{-\varsigma_{\text{se}}(s)\cdot\frac{r}{ms}}, \tag{8}$$

where m and s represent the server size and server speed respectively, and r denotes the execution requirements of service requests. We use re-execution to tolerate transient faults for service requests and assume each service request can tolerate at most one fault [29]. Under this assumption, a service request is deemed as failed only if itself and its re-execution both suffer a failure. Thus, the soft error reliability of a service request can be calculated as

$$R_{\text{re}}(m, s, r) = 1 - (1 - R(m, s, r))^2. \tag{9}$$

The execution requirements r of a request with mean $\bar{r}$ is a random variable following a exponential distribution. Thus, the PDF of r is derived as $f_r(z) = \frac{1}{\bar{r}}e^{-z/\bar{r}}$. Using Eqs. (8)–(9) and $f_r(z)$, the expected soft error reliability of a request for average-based scheduling is derived as

$$\begin{aligned} R_{\text{exp}} &= \int_0^\infty (f_r(z) \cdot R_{\text{re}}(m, s, r))d_z \\ &= ms(\frac{2}{ms + \varsigma_{\text{se}}(s)\bar{r}} - \frac{1}{ms + 2\varsigma_{\text{se}}(s)\bar{r}}). \end{aligned} \tag{10}$$

2.3. Cloud service profit model

As aforementioned, the cloud service profit is decided by revenue and cost which are specified as follows.

Revenue. The cloud service revenue is mainly affected by service price, QoS, and customer demand. In particular, QoS is mainly affected by the multiserver system configuration. Generally, better system configuration brings higher QoS to customers, but it inevitably increase service costs. In cloud service markets, service-level agreement (SLA) has been commonly used by many actual CSPs (e.g., Rackspace [30], Joyent [31], Microsoft Azure [32]) to guarantee service quality. SLA is a commitment between a CSP and customers on the service price and QoS [9]. For example, according to SLA, if a CSP serves a customer with a low QoS, then the provider will provide the customer with free

service as a penalty. In this paper, QoS is reflected by service requests' waiting time. Let $w_{\max}$ be the maximal service waiting time specified by the CSP in the SLA, then the charge function for a service request with execution requirement r and waiting time w is [9]

$$C(r,w)=\begin{cases} p\cdot r & ,\ 0\leqslant w\leqslant w_{\max} \\ 0 & ,\ w>w_{\max} \end{cases} \tag{11}$$

where p refers to the price per one billion instructions (Unit: cents/billion instructions). According to Eq. (11), if a request is handled within the maximum service waiting time $w_{\max}$, then the service provider will charge the request based on p and r. Otherwise, the request is free as a penalty. Therefore, considering deadline miss rate (see Eq. (6)) and soft error reliability (see Eq. (10)), we propose a formula to calculate the expected revenue gained by processing requests per UT as

$$Rev=\lambda\cdot p\cdot\bar{r}\cdot(1-DMR)\cdot R_{\exp}, \tag{12}$$

where $p\cdot\bar{r}$ (see Eq. (11)) denotes the expected charge to a request, and details on $p\cdot\bar{r}$ can be found in [9].

Cost. Cloud service cost is the sum of the capital expenditure (CAPEX) [33] for leasing infrastructure (i.e., servers) to serve customers' service requests and the electricity bills (i.e., operating expenditure (OPEX) [34]) for maintaining the normal operation of the infrastructure. The lease cost of a multiserver system per UT depends on the lease price of servers and the number of servers in the system. Suppose θ is the lease price of a server per UT (Unit: cents/second), then the lease cost of a multiserver having m servers per UT is

$$Cost_{\text{lease}}=m\cdot\theta. \tag{13}$$

The electricity cost of energy consumed by the multiserver system per UT depends on the electricity price and the power consumption [12]. Concentrating on the server's processor, the power consumption is composed of dynamic power P_{dyn} and static power P_{sta}. Thus, the multiserver's total power consumption P_{tot} is formulated as

$$P_{\text{tot}}=P_{\text{dyn}}+P_{\text{sta}}, \tag{14}$$

where P_{dyn} dominates P_{tot} [10]. Referring to the dynamic power model adopted in [8–10,12], we formulate P_{dyn} as $P_{\text{dyn}}=\xi\cdot s^{\phi}$ where ξ and ϕ are hardware-dependent constants and s is the server speed. Since servers in the system are not always fully utilized, the energy consumed by a multiserver per UT can be derived as

$$\begin{aligned} E_{\text{tot}}&=m\cdot P_{\text{tot}}=m\cdot(P_{\text{dyn}}+P_{\text{sta}}) \\ &=m\cdot(\rho\cdot\xi\cdot s^{\phi}+P_{\text{sta}}), \end{aligned} \tag{15}$$

where m is the number of servers. Assuming that the energy's electricity price (Unit: cents/Watt) is ζ, the electricity cost of a multiserver per UT is expressed as

$$Cost_{\text{electricity}}=\zeta\cdot E_{\text{tot}}=\zeta\cdot m\cdot(\rho\cdot\xi\cdot s^{\phi}+P_{\text{sta}}). \tag{16}$$

Profit. The profit of cloud services is the difference between revenue and cost. Thus, the profit of cloud services per UT is obtained as

$$\begin{aligned} Pro&=Rev-Cost_{\text{lease}}-Cost_{\text{electricity}} \\ &=\lambda\bar{p}\bar{r}(1-DMR)R_{\exp}-m(\theta+\zeta(\rho\xi s^{\phi}+P_{\text{sta}})). \end{aligned} \tag{17}$$

2.4. Problem definition

The goal of this paper is to maximize the cloud service profit by configuring the multiserver system. The cloud service profit maximization problem is formulated as

$$\text{Maximize:}\quad Pro \tag{18}$$

$$\text{Subject to:}\quad \rho<1 \tag{19}$$

$$s_l\leqslant s\leqslant s_u \tag{20}$$

Fig. 4. Leadership hierarchy of the grey wolves in GWO.

$$m_l\leqslant m\leqslant m_u. \tag{21}$$

In the above formulation, ρ refers to the server utilization, Eq. (19) ensures that server utilization is less than 1 [35]. In Eqs. (20) and (21), s_l and s_u denote the lower and upper bound of server speed s, m_l and m_u represent the lower and upper bound of server size m. These two equations indicate that the server configurations (i.e., m and s) are within the effective domains.

3. Our approach

Meta-heuristics have been commonly used in solving optimization problems due to its characteristics of simplicity, flexibility and local optima avoidance [36]. One interesting branch of the population-based meta-heuristics is swarm intelligence. Recently, a new swarm intelligence technique with inspiration from the social hierarchy and hunting behavior of grey wolf packs, i.e., grey wolf optimizer (GWO), has been proposed [36]. The GWO mimics the leadership hierarchy and hunting mechanism of gray wolves in nature. Based on GWO, we propose an effective multiserver configuration algorithm to solve the profit optimization problem. Below, we first present the modeling process of our proposed grouped GWO and then describe our grouped GWO-based multiserver configuration algorithm in detail.

3.1. Mathematical model of the grouped GWO

The original GWO uses a single group of grey wolves for hunting. The group contains four kinds of grey wolves: alpha (α), beta (β), delta (δ) and omega (ω) wolves [37]. These wolves have a very strict social dominant hierarchy as shown in Fig. 4. In the figure, the α wolf is regarded as the most dominating member in the group and is responsible for making decisions about hunting. The β wolves are subordinate wolves that help α wolves in decision-making, thus they are probably the best candidates to be alpha wolves in case the α wolves pass away or become very old. δ wolves obey α and β wolves but dominate ω wolves, and they are responsible for scouting. The ω wolves are of the lowest ranking in the hierarchy. The existence of ω wolves is benefit to maintain the entire pack and the dominance structure [36,37].

In this paper, we divide the original grey wolves into two groups, i.e., a cooperative hunting group and a random scout group. As shown in Fig. 5, the cooperative hunting group still obeys the social dominant hierarchy and it contains three types of grey wolves (i.e., α, β, and ω wolves) in which α and β wolves are responsible for hunting, and ω wolves follows them. The random scout group is responsible for scouting and searching and it only contains δ wolves. Note that the δ wolves do not follow α and β wolves, but concentrate on searching and exploring unknown areas where the prey may appear. To achieve a deep exploitation in the cooperative hunting group, we assume that there is one alpha wolf and three beta wolves, i.e., β_1, β_2, and β_3. In the grouped GWO, each grey wolf corresponds to a solution for server configuration, and the fitness of each grey wolf refers to the profit that the CSP can obtain under the server configuration corresponding to this grey wolf. Thus, in the hunting process, the four wolves with the best fitness are defined as α wolf and β (i.e., β_1, β_2, and β_3) wolves, and the

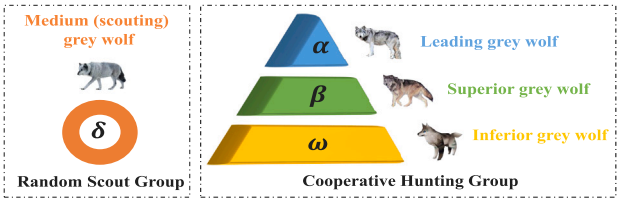


Fig. 5. Grey wolves groups in the grouped GWO.

rest wolves are ω wolves, where the best solution is deemed as the α wolf while the second, third and fourth best solutions are defined as the β_1, β_2, and β_3 wolves respectively. Below, we introduce the main phases of grey wolf hunting.

Encircling prey. The grey wolves encircle the prey during the hunt. Referring to [36–38], we model the encircling behavior of grey wolves by

$$\overrightarrow{D} = |\overrightarrow{C} \cdot \overrightarrow{X_p}(\eta) - \overrightarrow{X}(\eta)|, \tag{22}$$

where $\overrightarrow{D}$ denotes the distance between the grey wolf and the prey. η denotes the current number of iterations and $\overrightarrow{C}$ is a coefficient vector. $\overrightarrow{X_p}(\eta)$ and $\overrightarrow{X}(\eta)$ are the position vector of the prey and the grey wolf, respectively. Based on Eq. (22), the grey wolf's position is updated by

$$\overrightarrow{X}(\eta+1) = \overrightarrow{X_p}(\eta) - \overrightarrow{A} \cdot \overrightarrow{D}, \tag{23}$$

where $\overrightarrow{A}$ is also a coefficient vector. The coefficient vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ are calculated as

$$\overrightarrow{A} = \vec{a} \cdot \overrightarrow{\sigma_1}, \overrightarrow{C} = 2 \cdot \overrightarrow{\sigma_2}, \tag{24}$$

where $\vec{a}$ is the encircling coefficient vector. Its components are linearly decreased from 2 to 0 during iterations [37], i.e., $\vec{a} = 2(1 - (\eta/\eta_{\max})^2)$ where $\eta_{\max}$ denotes the maximum number of iterations. $\overrightarrow{\sigma_1}$ and $\overrightarrow{\sigma_2}$ are random vectors uniformly spanned in $[-1, 1]$ and $[0, 1]$, respectively.

Hunting. The grey wolves are able to recognize the location of prey and encircle them. In our scheme, the hunting of prey is guided by α and β wolves. In an abstract search space, the location of the optimum (prey) is unknown in advance. To simulate the hunting behavior of grey wolves, α and β wolves are assumed to have better knowledge about the potential location of prey. Thus, we save the first four best solutions (i.e., positions of α and β wolves) obtained so far and oblige the ω wolves to update their positions according to the position of α and β wolves. Based on Eqs. (22)–(24), the updating laws of their locations can be written as

$$\begin{aligned}
\overrightarrow{D_\alpha} &= |\overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \overrightarrow{X}|, \\
\overrightarrow{D_{\beta_1}} &= |\overrightarrow{C_2} \cdot \overrightarrow{X_{\beta_1}} - \overrightarrow{X}|, \\
\overrightarrow{D_{\beta_2}} &= |\overrightarrow{C_3} \cdot \overrightarrow{X_{\beta_2}} - \overrightarrow{X}|, \\
\overrightarrow{D_{\beta_3}} &= |\overrightarrow{C_4} \cdot \overrightarrow{X_{\beta_3}} - \overrightarrow{X}|,
\end{aligned} \tag{25}$$

where $\overrightarrow{D_\alpha}$, $\overrightarrow{D_{\beta_1}}$, $\overrightarrow{D_{\beta_2}}$, and $\overrightarrow{D_{\beta_3}}$ denote the distance between a grey wolf and α, β_1, β_2, and β_3 wolves, respectively. $\overrightarrow{C_1}$, $\overrightarrow{C_2}$, $\overrightarrow{C_3}$, and $\overrightarrow{C_4}$ are coefficient vectors. $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, and $\overrightarrow{X_{\beta_3}}$ indicate the position of α, β_1, β_2, and β_3 wolves, respectively. $\overrightarrow{X}$ denotes the position of the grey wolf. Based on Eq. (25), the position of the grey wolf is updated as

$$\begin{aligned}
\overrightarrow{X_1} &= \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot (\overrightarrow{D_\alpha}), \\
\overrightarrow{X_2} &= \overrightarrow{X_{\beta_1}} - \overrightarrow{A_2} \cdot (\overrightarrow{D_{\beta_1}}), \\
\overrightarrow{X_3} &= \overrightarrow{X_{\beta_2}} - \overrightarrow{A_3} \cdot (\overrightarrow{D_{\beta_2}}), \\
\overrightarrow{X_4} &= \overrightarrow{X_{\beta_3}} - \overrightarrow{A_4} \cdot (\overrightarrow{D_{\beta_3}}),
\end{aligned} \tag{26}$$

where $\overrightarrow{A_1}$, $\overrightarrow{A_2}$, $\overrightarrow{A_3}$, and $\overrightarrow{A_4}$ are coefficient vectors. Eq. (26) gives four update mechanisms of the gray wolf's position, thus $\overrightarrow{X_1}$, $\overrightarrow{X_2}$, $\overrightarrow{X_3}$, and $\overrightarrow{X_4}$ denote the updated position of the gray wolf, respectively. Based on Eqs. (23) and (26), the grey wolf's position in the next iteration is

$$\overrightarrow{X}(\eta+1) = k_\alpha \cdot \overrightarrow{X_1} + k_\beta \cdot (\frac{\overrightarrow{X_2} + \overrightarrow{X_3} + \overrightarrow{X_4}}{3}), \tag{27}$$

where k_α and k_β ($0 < k_\alpha, k_\beta < 1$ and $k_\alpha + k_\beta = 1$) denote the guiding coefficients of α and β wolves, respectively.

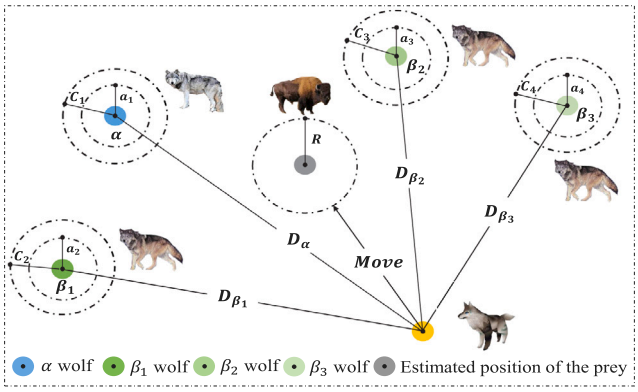


Fig. 6. The position update mechanism of ω wolves.

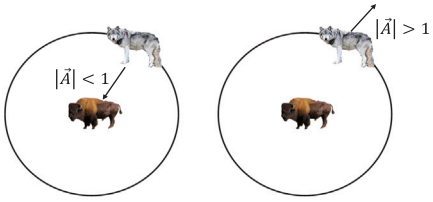


Fig. 7. Attacking prey (left) vs. searching for prey (right).

Fig. 6 illustrates the position update mechanism of ω wolves tracking its prey in a two-dimensional (2D) search space. We can see that ω wolves would be guided and they update their locations based on the discovered locations of α and β wolves. It also can be found that the final position of the prey is a random position within a circle defined by the positions of α and β wolves. In other words, α and β wolves estimate the position of the prey and ω wolves randomly updates their position around the prey.

Attacking prey. As aforementioned, when the position of the prey no longer changes, the grey wolves will begin to attack the prey. The process of approaching the prey is simulating by lowering the value of the encircling coefficient vector $\vec{a}$. According to Eq. (24), when $\vec{a}$ decreases, the fluctuation range of the coefficient vector $\overrightarrow{A}$ also becomes smaller. It can be readily deduced that $\overrightarrow{A}$ is a random value in the range of $[-a, a]$ where a would be decreased from 2 to 0 over the iterations. When the values of $\overrightarrow{A}$ are in the range of $[-1, 1]$, the next position of a search grey wolf could be anywhere between its current position and the position of the prey. In particular, when $|\overrightarrow{A}| < 1$, the grey wolves attack towards the prey, as illustrated in Fig. 7.

Search prey. Grey wolves mostly estimate and predict the position of the prey according to the positions of α and β wolves. They diverge from each other to search for prey and converge to attack prey. As shown in Fig. 7, when $|\overrightarrow{A}| > 1$, grey wolves will be far away from the currently inferred prey position (corresponding to the local optimal solution), so as to find a better prey position (corresponding to the optimal solution). Another component of the grouped GWO that favors exploration is the coefficient vector $\overrightarrow{C}$. According to Eq. (24), the $\overrightarrow{C}$ vector contains random values in the range of $[0, 2]$ which offers random weights for prey to stochastically emphasize or deemphasize the effect of prey in defining the distance of Eq. (22). Introducing $\overrightarrow{C}$

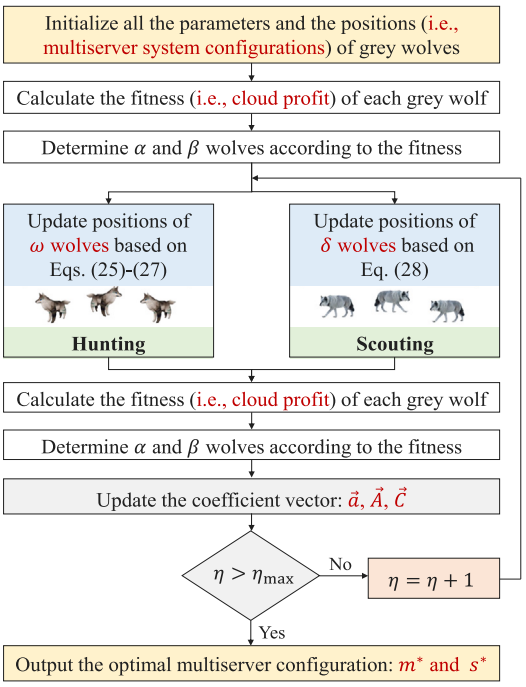


Fig. 8. Workflow of the proposed scheme.

helps avoid falling into local optimality as much as possible during the optimization process.

As mentioned above, the δ wolves in the random scout group are responsible for scouting and searching. Considering the widespread nature of the searching behavior of δ wolves, their locations can be updated as

$$\overrightarrow{X}(\eta+1) = \overrightarrow{X}(\eta) + \vec{\vartheta}, \tag{28}$$

where ϑ is a random variable within a limited range. Note that in the process of scouting the prey, δ wolves do not follow α and β wolves, but concentrate on continuously searching and exploring unknown areas where the prey may appear within the feasible areas. To better simulate the communication and cooperation between the grey wolves in the cooperative hunting group and the random scout group, we assume that when δ wolves find a position with better fitness during the search, it will notify α and β wolves this position and exchange positions with α and β wolves. Specifically, δ wolves will compare their fitness with the fitness of α and β wolves to ensure that α and β wolves always occupy the positions with the best fitness.

3.2. Grouped GWO-based multiserver configuration

We propose a grouped GWO-based multiserver configuration scheme to solve the cloud service profit optimization problem given in Eqs. (18)–(21). Fig. 8 shows the workflow of our grouped GWO-based multiserver configuration scheme. The scheme firstly initializes the parameters and the positions (i.e., the multiserver configurations) of all grey wolves. Then, according to Eq. (17), it calculates the fitness (i.e., cloud service profit) of each grey wolf. Based on the fitness, the scheme can determine the α and β wolves, i.e., the four grey wolves with the best fitness are regarded as α, β_1, β_2 and β_3 wolves, respectively.

In our scheme, a grey wolf is regarded as a point in a 2D search space. The grey wolf's position coordinates (m, s) and the multiserver configuration (i.e., server size and server speed) are in an one-to-one mapping relationship. For example, let (m, s) denote the position coordinates of a grey wolf, where the abscissa (m) represents the server size and the ordinate (s) represents the server speed. The fitness of the grey wolf is the profit gained by the CSP under the multiserver configuration, which can be calculated by Eq. (17). Algorithm 1 describes our grouped GWO-based multiserver configuration scheme. Inputs to the algorithm include the service request arrival rate λ, the average service execution requirements $\bar{r}$, the server rental price θ, the electricity price ζ, the server related parameters ξ and ϕ, the static power consumption P_{sta}, the numbers of β wolves and ω wolves $\mathcal{N}_\beta$ and $\mathcal{N}_\omega$, and the total number of grey wolves $\mathcal{N}_t$. Outputs are the optimal multiserver configuration (m^* and s^*) and the optimal cloud service profit (Pro^*).

Algorithm 1: Grouped GWO-based multiserver configuration scheme.

Input: Service request arrival rate λ, average service execution requirements $\bar{r}$, server rental price θ, electricity price ζ, server related parameters: ξ and ϕ, static power consumption P_{sta}, numbers of β wolves and ω wolves: $\mathcal{N}_\beta$ and $\mathcal{N}_\omega$, and the total number of grey wolves $\mathcal{N}_t$;

Output: Optimal multiserver system configurations: m^* and s^* and the optimal cloud service profit: Pro^*;

1 Initialize $\mathcal{N}_t$ groups of multiserver configurations (i.e., the positions of α, β, ω, and δ wolves): $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}} \sim \overrightarrow{X_{\beta_3}}$, $\overrightarrow{X_{\omega_1}} \sim \overrightarrow{X_{\omega_{30}}}$, $\overrightarrow{X_{\delta_1}} \sim \overrightarrow{X_{\delta_5}}$;
2 Initialize the coefficient vectors: $\vec{a}$, $\vec{A}$, and $\vec{C}$;
3 Calculate the cloud service profit under each group of multiserver configuration using Eq. (17);
4 Call Algorithm 2 to assign the four groups of optimal multiserver configurations to $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, $\overrightarrow{X_{\beta_3}}$;
5 **while** $\eta \leqslant \eta_{max}$ **do**
6 **for** $i = 1$ to $\mathcal{N}_\omega$ **do**
7 Update the positions of ω_i wolf using Eqs. (25)-(27);
8 Compute the cloud service profit under the multiserver configuration of ω_i wolf using Eq. (17);
9 Call Algorithm 2 to assign the four groups of optimal multiserver configurations to $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, $\overrightarrow{X_{\beta_3}}$;
10 Update the positions of δ wolves using Eq. (28);
11 Call Algorithm 2 to assign the four groups of optimal multiserver configurations to $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, $\overrightarrow{X_{\beta_3}}$;
12 Update the coefficient vectors: $\vec{a}$, $\vec{A}$, and $\vec{C}$;
13 $\eta = \eta + 1$;
14 $(m^*, s^*) \leftarrow \overrightarrow{X_\alpha}$;
15 Calculate the optimal cloud service profit (Pro^*) under (m^*, s^*) using Eq. (17);
16 **return** m^*, s^*, Pro^*;

The algorithm works as follows. First, the positions of all the grey wolves are initialized (line 1). The total number of the grey wolves is denoted by $\mathcal{N}_t$. Here, we assume that there are 39 groups of multiserver system configurations, i.e., $\mathcal{N}_t = 39$. In the cooperative hunting group, the numbers of α, β, and ω wolves denoted by $\mathcal{N}_\alpha$, $\mathcal{N}_\beta$ and $\mathcal{N}_\omega$ are set to 1, 3, and 30, respectively, while in the random scout group, the number of δ wolves denoted by $\mathcal{N}_\delta$ is set to 5. Note that the 39 groups of multiserver configurations are randomly generated within the effective domain of $[s_l, s_u]$ and $[m_l, m_u]$. The coefficient vectors $\vec{a}$, $\vec{A}$, and $\vec{C}$ are initialized on line 2. Based on the initialized 39 groups of multiserver configurations, the algorithm calculates the cloud service profit under each group of multiserver configuration based on Eq. (17) (line 3). Once the profits (i.e., fitness) of all the multiserver configurations are obtained, line 4 calls Algorithm 2 to assign the four groups of the optimal multiserver configurations to $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, and $\overrightarrow{X_{\beta_3}}$. Algorithm 2 uses the selective sorting method to sort the fitness of all grey wolves in descending order, so as to obtain four groups of optimal multiserver configurations. Subsequently, the algorithm iteratively solves the optimal multiserver configuration on lines 5–13 and calculates the optimal profit on lines 14–15.

Specifically, based on the positions of α and β wolves, Algorithm 1 first updates the positions of ω wolves based on Eqs. (25)–(27) and

Table 1
Parameter settings.

Parameter	Definition	Value
ς_0	Average soft error rate when the server is running at the maximum speed	10^{-5} [28]
d	Hardware-related coefficient in dynamic voltage scaling	2 [39]
p	Service request charge per one billion instructions	15 cents per billion instructions
θ	Rental price of a server per UT	15 cents per second [8–10,12]
ξ	Processor-related coefficient	9.4192 [8–10,12]
P_{sta}	Static power consumption	2 watts per second [8,10,12]
ζ	Electricity price of energy	0.1 cents per Watt × second [8,10,12]
$\mathcal{T}_{\min}$	Minimum value of the slack during a UT	0.5 s [24]
$\mathcal{T}_{\max}$	Maximum value of the slack during a UT	5 s [24]

Algorithm 2: Sorting-based selection of four optimal multiserver configurations.

Input: $\mathcal{N}_t$ groups of multiserver configurations: $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}} \sim \overrightarrow{X_{\beta_3}}$, $\overrightarrow{X_{\omega_1}} \sim \overrightarrow{X_{\omega_{30}}}$, $\overrightarrow{X_{\delta_1}} \sim \overrightarrow{X_{\delta_5}}$, and corresponding cloud service profit: Pro_α, $Pro_{\beta_1} \sim Pro_{\beta_3}$, $Pro_{\omega_1} \sim Pro_{\omega_{30}}$, $Pro_{\delta_1} \sim Pro_{\delta_5}$;
Output: Four best multiserver configurations: $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, and $\overrightarrow{X_{\beta_3}}$;
1 Initialize S_{con} and S_{pro};
2 **for** $i = 1$ to $\mathcal{N}_t$ **do**
3 $temp \leftarrow$the subscript index with the largest value in S_{pro} ;
4 Exchange the values of $S_{\text{pro}}[i]$ and $S_{\text{pro}}[temp]$;
5 Exchange the values of $S_{\text{con}}[i]$ and $S_{\text{con}}[temp]$;
6 $\overrightarrow{X_\alpha} \leftarrow S_{\text{con}}[0]$;
7 **for** $i = 1$ to $\mathcal{N}_\beta$ **do**
8 $\overrightarrow{X_{\beta_i}} \leftarrow S_{\text{con}}[i]$;
9 **return** $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, $\overrightarrow{X_{\beta_3}}$;

calculates the cloud service profit under the configuration of ω wolves based on Eq. (17) (lines 6–8). Once the fitness of all ω wolves are obtained, it then calls Algorithm 2 to assign the four groups of the optimal multiserver configurations to $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, and $\overrightarrow{X_{\beta_3}}$ (line 9). Subsequently, the positions of δ wolves are updated based on Eq. (28) on line 10 and Algorithm 2 is called again to assign the four groups of the optimal multiserver configurations to $\overrightarrow{X_\alpha}$, $\overrightarrow{X_{\beta_1}}$, $\overrightarrow{X_{\beta_2}}$, and $\overrightarrow{X_{\beta_3}}$ on line 11. Line 12 updates the coefficient vectors of $\vec{a}$, $\vec{A}$, and $\vec{C}$. The number of iterations is updated on line 13. The algorithm stops until it reaches the maximum number of iterations $\eta_{\max}$. Finally, the optimal position $\overrightarrow{X_\alpha}$ owned by the α wolf is the optimal multiserver configuration (i.e., m^* and s^*) (line 14). Based on m^* and s^*, line 15 calculates the optimal cloud service profit (Pro^*) based on Eq. (17) and line 16 returns the optimal multiserver configuration (m^* and s^*) and cloud service profit (Pro^*).

4. Experimental results and analysis

We conduct three sets of experiments to validate the efficacy of our proposed grouped GWO-based multiserver configuration scheme. The first set of simulations is to test the optimal solution exploration capacity of our scheme. The second set of simulations is to study the impact of multiserver configurations on the cloud service profit under varying service request rates. The third set of simulations is to compare the cloud service profit achieved by our scheme and two benchmark methods. The parameter settings used in the simulations are presented in Table 1.

4.1. The optimal solution exploration capacity

We test the capacity of our proposed grouped GWO-based multiserver configuration scheme in exploring the optimal solution. In the simulations, the average service execution requirement $\bar{r}$ is set to 1, the service request arrival rate λ is set to 30.0, the domain ranges of the system configurations (i.e., m and s) are the same to the settings in Section 4.2. In addition, the numbers of α, β, ω, and δ wolves are set to 1, 3, 20, and 5, respectively. The maximum number of iterations $\eta_{\max}$ is set to 100. The guiding coefficients of α and β wolves, i.e., k_α and k_β given in Eq. (27) are set to 0.4 and 0.6, respectively. The settings of other parameters such as p, θ, ξ, P_{sta}, ζ, $\mathcal{T}_{\min}$, and $\mathcal{T}_{\max}$ are the same as in Table 1.

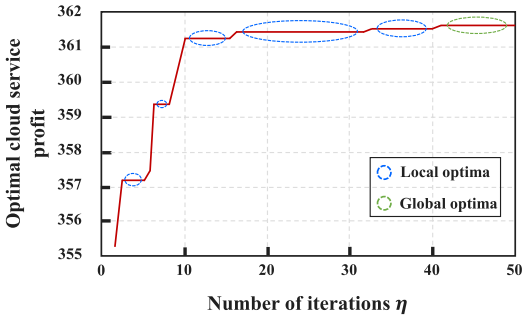


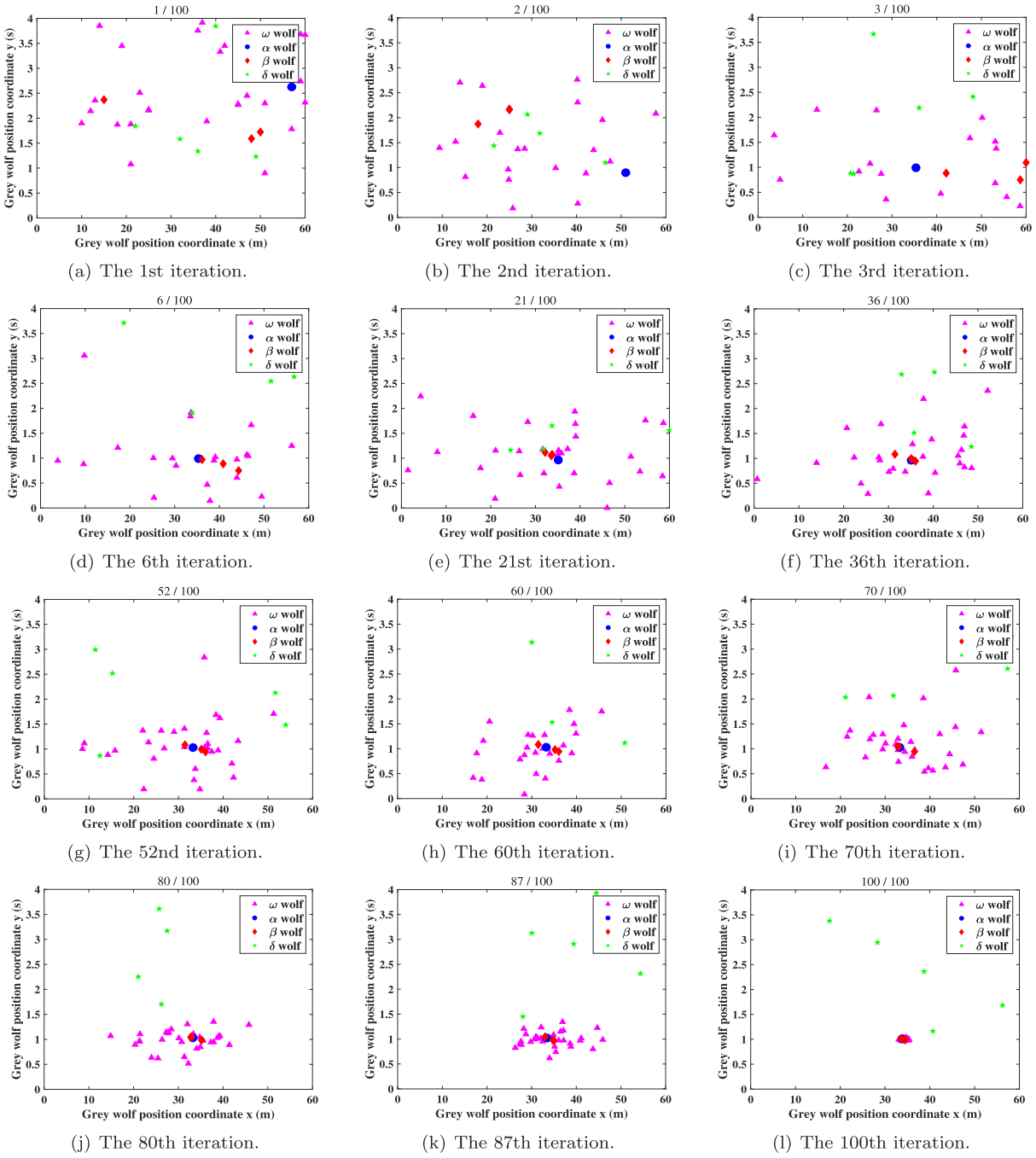
Fig. 9. Optimal cloud service profit vs. η.

Fig. 9 presents the relationship between the optimal cloud service profit and the number of iterations (η) in Algorithm 1. As can be seen that in the iterative process, α wolf has fallen into the local optimal position many times, but with the extensive exploration of δ wolves, α wolf can effectively deviate from the local optimal position to search the global optimal position that has the maximum profit. For better intuition, Fig. 10 takes 100 iterations as an example that demonstrates the visualization process of searching the optimal solution using our grouped GWO-based scheme. Clearly, in the early iterations, the positions of grey wolves were randomly scattered in the feasible region. Then, with the number of iterations increases, α and β wolves gradually move closer, and ω wolves has always been following the α and β wolves. The δ wolves, which are responsible for searching and scouting the prey, have been performing random scout in the feasible region to avoid falling into local optima. When the iterations are finished, α, β, and ω wolves have gathered together and captured the prey, that is, the optimal solution has been obtained. In addition, from Fig. 10 we can also observe that the grey wolves spend about 70% of their time on extensive exploration of the prey, and about 30% of their time on in-depth exploitation and attacking the prey. That is to say, after the number of iterations (η) reaches 70 times, the grey wolves begin to attack the prey.

4.2. The optimal cloud service profit

We study the effect of multiserver configurations on the cloud service profit under varying service request arrival rates (λ). In the simulations, the average service execution requirement $\bar{r}$ is set to 1, the domain range of the server size is set to $[10, 60]$, and domain range of the server speed is set to $[0.75, 4.00]$.

Fig. 11 shows the optimal cloud service profit and multiserver configurations when λ is 20.0, 25.0, 30.0, and 35.0, respectively. Fig. 11(a) plots the relationship between the optimal cloud service profit and



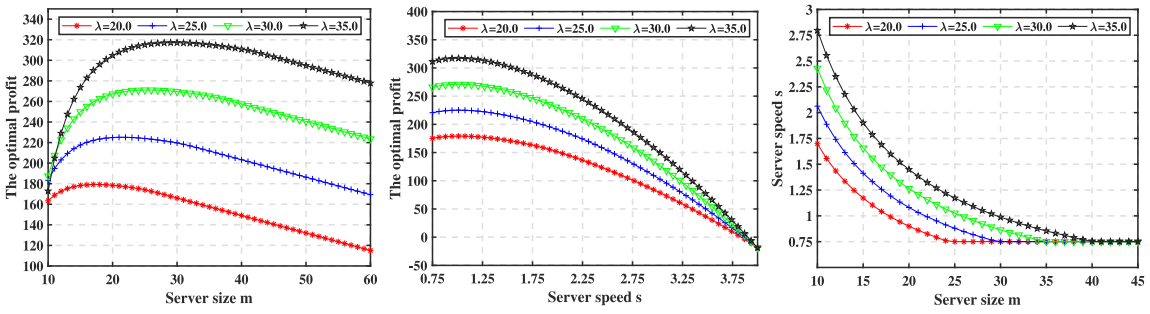
(a) The 1st iteration. (b) The 2nd iteration. (c) The 3rd iteration.

(d) The 6th iteration. (e) The 21st iteration. (f) The 36th iteration.

(g) The 52nd iteration. (h) The 60th iteration. (i) The 70th iteration.

(j) The 80th iteration. (k) The 87th iteration. (l) The 100th iteration.

Fig. 10. The visualization process of searching the optimal solution using our scheme.



(a) Optimal cloud service profit vs. m. (b) Optimal cloud service profit vs. s. (c) s vs. m.

Fig. 11. Optimal cloud service profit and multiserver configurations (i.e., m and s) under different λ.

Table 2
Comparison of the optimal cloud service profit between our scheme and the TSPPM scheme [10] under different λ (Pro_Imp: Profit improvements compared with the TSPPM scheme).

Parameters		TSPPM scheme			Our scheme			Pro_Imp
λ	$\bar{r}$	m^*_{TSPPM}	s^*_{TSPPM}	Pro^*_{TSPPM}	m^*_{Our}	s^*_{Our}	Pro^*_{Our}	$(Pro_{\text{Our}} - Pro_{\text{TSPPM}})/Pro_{\text{TSPPM}}$
5.0	1.0	6.05	1.66	42.04	10.00	0.77	54.67	30.04%
10.0	1.0	11.67	1.01	87.65	12.34	1.04	117.24	33.76%
15.0	1.0	17.23	0.99	134.75	17.79	1.03	178.02	32.11%
20.0	1.0	22.76	0.98	181.44	23.24	1.02	239.06	31.75%
25.0	1.0	28.27	0.98	229.49	28.64	1.01	300.25	30.83%
30.0	1.0	33.78	0.97	276.38	33.96	1.01	361.54	30.81%
35.0	1.0	29.27	0.97	324.65	39.11	1.01	422.90	30.24%

Table 3
Comparison of the optimal cloud service profit between our scheme and the BFS scheme [40] under different λ (Opt_Dev: The degree of deviation from the optimal solution obtained by the BFS scheme).

Parameters		BFS scheme			Our scheme			Opt_Dev
λ	$\bar{r}$	m^*_{BFS}	s^*_{BFS}	Pro^*_{BFS}	m^*_{Our}	s^*_{Our}	Pro^*_{Our}	$\lvert(Pro_{\text{Our}} - Pro_{\text{BFS}})/Pro_{\text{BFS}}\rvert$
5.0	1.0	10.00	0.7692	54.6704402	10.00	0.7691	54.6704000	0.000074%
10.0	1.0	12.30	1.0474	117.2427911	12.34	1.0442	117.2425000	0.000248%
15.0	1.0	17.80	1.0310	178.0219731	17.79	1.0321	178.0219000	0.000041%
20.0	1.0	23.22	1.0209	239.0574055	23.24	1.0204	239.0573000	0.000044%
25.0	1.0	28.59	1.0140	300.2464292	28.64	1.0118	300.2460000	0.000143%
30.0	1.0	33.93	1.0088	361.5382338	33.96	1.0074	361.5378000	0.000120%
35.0	1.0	39.25	1.0048	422.9037336	39.11	1.0084	422.9025000	0.000292%

the server size m under different λ. We can see that the cloud service profit increases with the growth of server size m. But once m exceeds a certain threshold, the cloud service profit will decrease gradually. This indicates that there exists an optimal m with the maximum cloud service profit for a given customer demand. More or less servers would lead to a loss of cloud service profit. Similarly, we can observe from Fig. 11(b) that the cloud service profit increases at the beginning but drops once s exceeds a certain threshold, with the increase in server speed s. This is because that better multiserver configuration would cause more energy consumption and cloud service cost, and hence less profit. In addition, the results reveal that a higher service request arrival rate λ generally requires a better multiserver configuration for profit maximization.

Fig. 11(c) demonstrates the relationship between the server speed s and the server size m under varying λ. From the figure, we can observe that when λ is fixed, s and m exhibit opposite changing tendencies. The reason is that given the service request arrival rate λ, there exists an upper bound on the processing capability required by the arrival service requests. Therefore, with the increased server size m, our scheme would lower the server speed s to save energy for maximizing profit.

4.3. Comparison of our scheme with benchmark schemes

Our scheme is also compared with a state-of-the-art benchmark scheme TSPPM [10] in terms of maximizing the cloud service profit under different service request arrival rates. The TSPPM scheme considers two server speed and power consumption models in the cloud service cost calculation and uses an analytical method to decide the optimal multiserver configurations for maximizing profit. Note that TSPPM does not consider the occurrence of soft errors during the execution of service requests. Below, we compare our scheme with TSPPM in maximizing the cloud service profit under the same parameter settings.

Table 2 compares the optimal cloud service profits obtained by our scheme and TSPPM [10] under different service request arrival rates λ. From the table, we can see that our scheme is superior to TSPPM in increasing cloud service profit. For example, compared to TSPPM, our scheme has an average increase of 31.36% and a maximum increase of 33.76% with respect to the profit. The reason is that our scheme considers the impact of deadline miss rate and soft error reliability on the cloud service profit when optimizing multiserver configurations.

Furthermore, to verify the efficacy of our scheme, we compare our scheme with the brute force search (BFS) method [40] that can find the optimal solution in the feasible domain space.

Table 3 compares the solutions obtained by our scheme and BFS under varying service request arrival rates λ. In the simulations, BFS iterates 1.625 billion times for each set of parameters. As can be seen that the magnitude error of the solutions obtained by our scheme and BFS is on the order of 10^{-6}, indicating that our solution is very close to the optimal solution.

5. Conclusions and future work

In this paper, we studied the cloud service profit optimization problem that specially considers the deadline miss rate and soft error reliability of service requests. To solve this problem, we first constructed the models of multiserver system, deadline miss rate, and soft error reliability, and on the top of these models, we built the cloud service profit model from the aspects of revenue and cost. Then, we formulated the profit optimization problem under the constraints of server utilization and configuration. Finally, we proposed a grouped GWO-based multiserver configuration scheme that can maximize cloud service profit by determining the optimal server size and server speed under a given customer demand. Experimental results show that our scheme has an average increase of 31.36% and a maximum increase of 33.76% in terms of cloud service profit when compared to TSPPM. Moreover, the solution generated by our scheme is very close to the optimal solution obtained by the brute force search.

In the near future, we plan to combine our soft error rate model with neural network techniques to achieve proactive prediction of the occurrence of soft errors in the server. In addition, we will extend our multiserver model to heterogeneous cloud computing platforms for handling various types of service requests or application domains.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Buyya, S.N. Srirama, G. Casale, et al., A manifesto for future generation cloud computing: Research directions for the next decade, ACM Comput. Surv. 51 (5) (2018) 1–38.

[2] P. Cong, Z. Zhang, J. Zhou, X. Liu, Y. Liu, T. Wei, Customer adaptive resource provisioning for long-term cloud profit maximization under constrained budget, IEEE Trans. Parallel Distrib. Syst. 33 (6) (2022) 1373–1392.

[3] P. Han, C. Du, J. Chen, F. Ling, X. Du, Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique, J. Syst. Archit. 112 (2021) 101837.

[4] K.K. Chakravarthi, L. Shyamala, TOPSIS inspired budget and deadline aware multi-workflow scheduling for cloud computing, J. Syst. Archit. 114 (2021) 101916.

[5] B. Liu, S. Meng, X. Jiang, X. Xu, L. Qi, W. Dou, A QoS-guaranteed online user data deployment method in edge cloud computing environment, J. Syst. Archit. 118 (2021) 102185.

[6] H. Feng, Y. Deng, J. Li, A global-energy-aware virtual machine placement strategy for cloud data centers, J. Syst. Archit. 116 (2021) 102048.

[7] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, M. Chen, Cost and makespan-aware workflow scheduling in hybrid clouds, J. Syst. Archit. 100 (2019).

[8] T. Wang, J. Zhou, G. Zhang, T. Wei, S. Hu, Customer perceived value- and risk-aware multiserver configuration for profit maximization, IEEE Trans. Parallel Distrib. Syst. 31 (5) (2020) 1074–1088.

[9] J. Mei, K. Li, A. Ouyang, K. Li, A profit maximization scheme with guaranteed quality of service in cloud computing, IEEE Trans. Comput. 64 (11) (2015) 3064–3078.

[10] J. Cao, K. Hwang, K. Li, A. Zomaya, Optimal multiserver configuration for profit maximization in cloud computing, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1087–1096.

[11] J. Mei, K. Li, K. Li, Customer-satisfaction-aware optimal multiserver configuration for profit maximization in cloud computing, IEEE Trans. Sustain. Comput. 2 (1) (2017) 17–29.

[12] K. Li, J. Mei, K. Li, A fund-constrained investment scheme for profit maximization in cloud computing, IEEE Trans. Serv. Comput. 11 (6) (2018) 893–907.

[13] Y. Chiang, Y. Ouyang, Profit optimization in sla-aware cloud services with a finite capacity queuing model, Math. Probl. Eng. 2014 (2014) 1–11.

[14] A. Fox, R. Griffith, A. Joseph, et al., Above the clouds: A berkeley view of cloud computing, Dep. Electr. Eng. Comput. Sci. 28 (13) (2009) 1–23.

[15] P. Cong, L. Li, J. Zhou, K. Cao, T. Wei, M. Chen, S. Hu, Developing user perceived value based pricing models for cloud markets, IEEE Trans. Parallel Distrib. Syst. 29 (12) (2018) 2742–2756.

[16] J. Zhou, J. Sun, M. Zhang, Y. Ma, Dependable scheduling for real-time workflows on cyberCphysical cloud systems, IEEE Trans. Ind. Inf. 17 (11) (2021) 7820–7829.

[17] D. Zhu, H. Aydin, Energy management for real-time embedded systems with reliability requirements, in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, 2006, pp. 528–534.

[18] T. Wu, H. Gu, J. Zhou, T. Wei, X. Liu, M. Chen, Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud, J. Syst. Archit. 84 (2018) 12–27.

[19] Amazon EC2. [Online]. Available: http://aws.amazon.com.

[20] Y. Yang, Y. Chen, Sun grid engine (SGE) and its application, in: Proceedings of International Symposium on Computers & Informatics, 2015, pp. 975–982.

[21] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: The condor experience, Concurr. Comput.: Pract. Exper. 17 (2–4) (2005) 323–356.

[22] T. Tannenbaum, D. Wright, K. Miller, M. Livny, Condor: a distributed job scheduler, in: Beowulf Cluster Computing with Linux, MIT Press, 2002, pp. 307–350.

[23] L. Kleinrock, Queueing Systems: Theory, Vol. 1, John Wiley and Sons, 1975, [Online]. Available.

[24] L. He, S.A. Jarvis, D.P. Spooner, H. Jiang, D.N. Dillenberger, G.R. Nudd, Allocating non-real-time and soft real-time jobs in multiclusters, IEEE Trans. Parallel Distrib. Syst. 17 (2) (2006) 99–112.

[25] W. Zhu, B.D. Fleisch, Performance evaluation of soft realtime scheduling for multicomputer cluster, in: Proceedings of the IEEE International Conference on Distributed Computing Systems, 2000, pp. 610–617.

[26] L. He, S.A. Jarvis, D.P. Spooner, X. Chen, G.R. Nudd, Dynamic scheduling of parallel jobs with QoS demands in multiclusters and grids, in: Proceedings of the IEEE/ACM International Conference on Grid Computing, 2004, pp. 402–409.

[27] J. Zhou, M. Zhang, J. Sun, T. Wang, X. Zhou, S. Hu, DRHEFT: Deadline-constrained reliability-aware HEFT algorithm for real-time heterogeneous MPSoC systems, IEEE Trans. Reliab. 71 (1) (2022) 178–189.

[28] G. Aupy, A. Benoit, Y. Robert, Energy-aware scheduling under reliability and makespan constraints, in: Proceedings of the International Conference on High Performance Computing, 2012, pp. 1–10.

[29] J. Zhou, X.S. Hu, Y. Ma, J. Sun, T. Wei, S. Hu, Improving availability of multicore real-time systems suffering both permanent and transient faults, IEEE Trans. Comput. 68 (12) (2019) 1785–1801.

[30] Rackspace, 2021, [Online]. Available: http://www.rackspace.com/information/legal/cloud/sla.

[31] Joyent, 2021, [Online]. Available: http://www.joyent.com/company/policies/cloud-hosting-service-level-agreement.

[32] Microsoft azure, 2021, [Online]. Available: http://azure.microsoft.com/en-us/support/legal/sla/.

[33] Capital expenditure. [Online]. Available: https://en.wikipedia.org/wiki/Capitalexpenditure.

[34] Operating expense. [Online]. Available: https://en.wikipedia.org/wiki/Operatingexpense.

[35] K. Li, C. Liu, K. Li, A.Y. Zomaya, A framework of price bidding configurations for resource usage in cloud computing, IEEE Trans. Parallel Distrib. Syst. 27 (8) (2016) 2168–2181.

[36] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.

[37] B. Yang, X. Zhang, T. Yu, H. Shu, Z. Fang, Grouped grey wolf optimizer for maximum power point tracking of doubly-fed induction generator based wind turbine, Energy Convers. Manage. 133 (2017) 427–443.

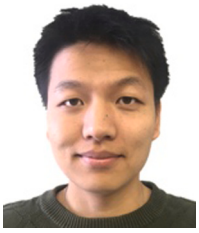
[38] N. Mittal, U. Singh, B.S. Sohi, Modified grey wolf optimizer for global engineering optimization, Appl. Comput. Intell. Soft Comput. 2016 (2016).

[39] K.V. Vishwanath, N. Nagappan, Characterizing cloud computing hardware reliability, in: Proceedings of the ACM Symposium on Cloud Computing, 2010, pp. 193–204.

[40] Brute-force search. [Online]. Available: https://en.wikipedia.org/wiki/Brute-force_search.

Peijin Cong received her B.S. and Ph.D. degrees in Computer Science from East China Normal University, Shanghai, China, in 2016 and 2021, respectively. She is currently with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. Her research interests are in the area of Cloud Computing and Internet of Things. She has published 20 refereed papers, including 10 papers in premier IEEE Transactions.

XiangPeng Hou received the B.S. degree from the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China, in 2021. He is currently pursuing the master degree with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests are in the areas of Cloud Computing and Internet of Things.



Minhui Zou received the B.S. degree and Ph.D. degree in computer science and technology from Chongqing University, China, in 2013 and 2018, respectively. He is currently a lecturer with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His current research interests include Hardware Security, Edge Computing, and Memory Computing.

Jiangshan Dong is a researcher of Shanghai AI Laboratory, Shanghai, China. He received the BS degree from Wuhan University, China, in 2003, the MS degree from Nanjing University, China, in 2006, and the Ph.D. degree from Shanghai University, China, in 2015. Dr. Dong has published more than 10 refereed papers. His research interests include Big Data and Cloud Computing.

Mingsong Chen received the B.S. and M.E. degrees from Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006 respectively, and the Ph.D. degree in Computer Engineering from the University of Florida, Gainesville, in 2010. He is currently a Professor with the Software Engineering Institute at East China Normal University. His research interests are in the area of Cloud Computing and Cyber-Physical Systems, where he has published more than 100 refereed papers. He is an Associate Editor of IET Computers & Digital Techniques and Journal of Circuits, Systems, and Computers.

Junlong Zhou received the Ph.D. degree in Computer Science from East China Normal University, Shanghai, China, in 2017. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, during 2014–2015. He is currently an Associate Professor with the Nanjing University of Science and Technology, China. His research interests are Embedded Systems, Cloud-Edge Computing, and Internet of Things, where he has published 90 refereed papers, including 30 in premier IEEE/ACM Transactions. He has been an Associate Editor for the Journal of Circuits, Systems, and Computers, the IET Cyber-Physical Systems: Theory & Applications, a Subject Area Editor for the Journal of Systems Architecture, and a Guest Editor for 6 ACM/IET/Elsevier/Wiley Journals such as ACM Transactions on Cyber-Physical Systems. He has held chair positions in many IEEE/ACM conferences. He received the Best Paper Award from IEEE iThings 2020.