IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 15, 2020, accepted February 9, 2020, date of publication February 13, 2020, date of current version February 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973758

Task Scheduling Algorithm Based on Improved Firework Algorithm in Fog Computing

SHUDONG WANG, TIANYU ZHAO, AND SHANCHEN PANG
School of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266555, China

Corresponding author: Shanchen Pang (pangsc@upc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572523, Grant 61572522, Grant 61873281, Grant 61672033, and Grant 61873280.

ABSTRACT As an emerging computing model close to the end-user, fog computing can move tasks from the cloud to the fog device to process, and make up for the lack of cloud computing in terms of mobile support, delay and location-aware. Due to the less computing resources and weak processing ability of fog devices, the lightweight computing has been the primary choice, higher requirements are imposed for time and load. Therefore, in the Cloud-fog computing architecture, a new task scheduling method(I-FASC) is proposed for the characteristics of tasks and resources, including an improved firework algorithm (I-FA) is proposed by introducing the explosion radius detection mechanism of fireworks. By two sets of experiments show this method can better reduce the processing time of the task and ensure better overall load balancing of fog devices in a cloud-fog computing system.

INDEX TERMS Cloud computing, fog computing, Internet of Things (IoT), task scheduling.

I. INTRODUCTION

Cloud computing is becoming more and more popular as a new distributed computing in the Internet era. By 2019, more than 70% of the calculations will be completed on the cloud server. Cloud computing provides users with efficient, secure computing and storage services, and it has been bringing huge economic benefits in many fields [1]–[4]. However, with the rapid development of smart devices, artificial intelligence and other Internet of things [5], the connectivity of objects are stronger and getting stronger, the number of devices connected to network is growing explosively, and lots of devices at the edge of the network need low latency and high reliability. Because the cloud data center is far away from the end user, the ultra-long distance transmission reduces the real-time performance and a large number of Internet of Things (IoT) applications are processed in the cloud data center, which increases the burden of the cloud [6]. In order to make up for the shortcomings of cloud computing in terms of delay, mobility support and enhance user experience. Cisco proposed fog computing in 2012 [7]. As an emerging computing model, fog computing extends the cloud computing paradigm from the network core to the network edge, providing compute, storage and network services between end-user and traditional clouds, and it can reduce latency and network congestion [8]–[10]. As a supplement to cloud computing, fog computing mainly deals with localized, real-time, and short-cycle data processing and analysis, for example: autopilot, telemedicine, mobile roaming, etc [11]–[13]. Therefore, combining fog computing with cloud computing is a hot topic in current technology research. As we all know, cloud computing, including hybrid cloud and public cloud is a computer and server connected by the Internet, and generally provide services to users in a large geographic area. Fog computing migrates data to a server near the terminal by managing certain applications or services at the edge of the network, usually providing services to users in a small local area network. When terminal data requires higher delay, fog computing can process data faster because of its location advantage. But the heterogeneity of the fog devices determines that different devices have different processing power, it is not as good as the data center, resulting in different time for the fog devices to handle tasks and the resource load will change frequently [14]–[17].

In the task scheduling of cloud computing, although there is a large-scale parallel processing architecture in cloud computing, there are still some phenomena such as low resource utilization and unbalanced load, which seriously affect the

The associate editor coordinating the review of this manuscript and approving it for publication was Guanjun Liu.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

work efficiency. On the one hand, large-scale data transmission will consume a lot of network bandwidth and increase the burden of cloud data center; On the other hand, due to the different terminal needs, the requirements for resources are also varied, and the long-distance data transmission will produce a large delay, unable to meet the time-delay sensitive user needs, at this time, the emergence of fog computing is very important. In the task scheduling of fog computing, the most basic problem is to match the requirements and resources. In iot scenarios, the user's needs are diverse, while fog computing service node performance is far inferior to cloud computing platform service node. Therefore, how to allocate the limited resources effectively in fog computing and allocate the resources according to users' needs to achieve the optimal scheduling goal is an urgent problem to be solved.

Therefore, to meet the needs of future Internet applications and avoid wasting resources, we need to ensure the low processing time of the fog devices and the relatively balance of the overall load level according to user needs, which is becoming an important issue for users and operators. So, based on this situation, the contributions of this paper are summarized as follows:

1) An improved firework algorithm (I-FA) is proposed, which introduce the explosion radius detection mechanism of fireworks to avoid optimal solution disappears.

2) A task scheduling method(I-FASC) is presented, this method take into account the characteristics of tasks and resources, including an improved firework algorithm.

II. RELATED WORK

In recent years, fog computing has become a trend. As a novel paradigm, the core idea of fog computing is that the computation should be performed near to data source. Many researchers from home and abroad have begun studying in this field [18]–[22].

In the theoretical study of fog computing, Sarkar and Misra [23] modeled the fog computing structure in theory, comparing traditional cloud computing architecture and fog computing architecture from the service delay and energy consumption. Research shows that the service delay and energy consumption have more advantages in fog computing than cloud computing, but this article is limited to theoretical analysis. Stojmenovic [24] gave the fog computing structure in the smart grids, and they are connected to the software-defined network scenario, but they only considered the application of fog computing. Fan and Ansari [21] proposed the cost aware cloudlet placement in mobile edge computing strategy where both the cloudlet cost and average E2E delay are considered in the cloudlet placement. Yi *et al.* [25] analyzed the objectives and challenges of the fog computing platform, and designed a low-latency, high-efficiency and versatile fog computing platform, which did not involve the computational service problem of fog computing. Yang *et al.* [26] devised CUE, an intelligent EC framework based on "cloud-user-edge" cooperation by taking UGC features and resource heterogeneity into account. Yigitoglu *et al.* [27] proposed a framework, called Foggy, that facilitates dynamic resource provisioning and automated application deployment in Fog Computing architectures.

In terms of optimization strategy, Zeng *et al.* [28] researched a software-defined embedded system that supports fog computing to solve the problem of resource management in traditional embedded systems. They designed an efficient task scheduling and resource management strategy. For some nonlinear programming problems, they proposed an algorithm to minimize the task completion time. Pham and Euh [29] formulated the task scheduling problem in cloud-fog environment and then proposed a heuristic-based algorithm, whose major objective is achieving the balance between the makespan and the monetary cost of cloud resources. Bitam *et al.* [30] proposed a new bio-inspired optimization approach called Bees Life Algorithm (BLA) aimed at addressing the job scheduling problem in the fog computing environment, and this approach is based on the optimized distribution of a set of tasks among all the fog computing nodes. Sun *et al.* [31] proposed a two-level resource scheduling model and designed a resource scheduling scheme among fog nodes in the same fog cluster based on the theory of the improved non-dominated sorting genetic algorithm II (NSGA-II), which considers the diversity of different devices. Zhang and Zhou [32] proposed a method based on a two-stage strategy to maximize task scheduling performance and minimize nonreasonable task allocation in clouds, experimental results show that they can effectively improve the cloud's scheduling performance and achieve the load balancing of cloud resources. Li *et al.* [33] proposed a cloud task scheduling strategy based on clustering and improved SOS(Symbiotic Organisms Search) algorithm to solve the problems of some Quality of Service(QoS)-based scheduling algorithms in cloud computing environment.

Through careful study of the above studies, we find that they have advantages and limitations in many aspects: 1) many studies, e.g., [23]–[25], gave a sufficient introduction to the theoretical and platform of fog computing, and they were contributed to the research of optimization strategy. 2) various existing studies, e.g., [28], [30], [31], considered the optimization of time and energy consumption brought by fog computing, but they did not consider the problem between delay and device load according to tasks and resource types. It is easy to waste resources.

III. CLOUD-FOG COMPUTING WORKFLOW

A. CLOUD-FOG COMPUTING ARCHITECTURE

The architecture of Cloud-fog computing is divided into the terminal layer, fog computing layer, and core layer, as shown in Fig. 1.

The terminal layer is mainly composed of terminal resource requestors, including fixed devices and mobile devices, such as sensors on the road, mobile phones, smart watches, computers and others. There are three types of task

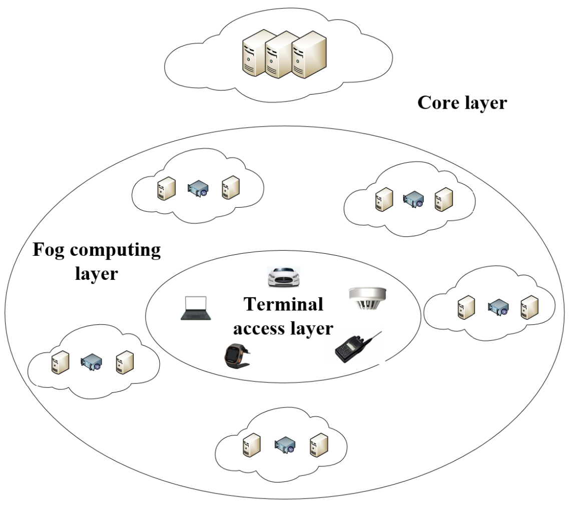

FIGURE 1. Cloud-Fog computing architecture.

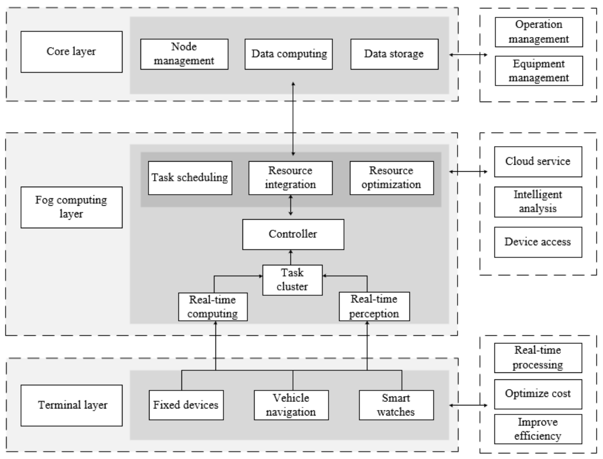

FIGURE 2. The workflow of the architecture.

requests in the Cloud-fog computing architecture in present, one for time-sensitive requests, for example: games, autopilot and others, and one for storage requests, for example: data comparison and medical storage and others, and one for bandwidth requests, for example: drone, AR and others. All tasks are generated at this layer, and the processed results will return to this layer eventually.

The fog computing layer is mainly composed of fog resource providers, including fog devices and fog processor. The fog devices are located at the edge of the network and composed of devices with computing, storage and transmit ability, such as base station and micro servers. The fog processor is a hub connecting the core devices, fog devices and terminal devices, and it can send different requests to different locations. This layer can sense terminal requests in real time and provide various services such as device access and data processing. And this can solve the processing delay problem and alleviate pressure for the core layer.

The core layer is mainly composed of cloud resource providers, including server clusters with large storage capacity and strong computing ability. Servers are interconnected in the core layer, and the virtual machines can be moved back and forth between servers to handle complex computing tasks. The emergence of the fog computing layer facilitates the efficient use of cloud resources by the core layer and makes this layer to focus on large-scale storage and big data processing.

B. CLOUD-FOG COMPUTING WORKFLOW

The workflow of this architecture is shown in Fig. 2. And the following is the specific working process.

1) The terminal device sends the data request to the processing module of the fog computing layer.

2) In the fog computing layer, we establish a task clustering model and a resource integration model and determine the execution location of the task according to resource integration.

3) The controller sends tasks to different servers for processing.

4) The fog computing layer and core layer send the processed results to the terminal.

IV. PRETREATMENT

A. TASK CLUSTERING

With the continuous development of artificial intelligence and big data, service experience has become more and more important. In the resource service of fog computing, the requirements of various tasks have different requirements for various resources in fog computing.

We can research tasks at the task level and use reasonable strategies to process the tasks. This way can facilitate the selection of different fog computing servers and reduce the size of candidate tasks during scheduling.

Assuming that K_i represents the size of task T_i (i = 1, 2, ..., n), S_i represents the expected storage space of task, D_i represents the expected completion time of task, Y_i represents the expected bandwidth of task.

We cluster the tasks according to expected storage space, expected time and expected bandwidth. In order to facilitate data processing, normalization processing is performed separately as follows:

$$S_i^* = \frac{S_i}{\sum_{i=1}^{n} S_i} \tag{1}$$

$$D_i^* = \frac{D_i}{\sum_{i=1}^{n} D_i} \tag{2}$$

$$Y_i^* = \frac{Y_i}{\sum_{i=1}^{n} Y_i} \tag{3}$$

All tasks are classified into a cluster C, then we calculate distance between the different tasks:

$$d_{ij} = \sqrt{(S_i^* - S_j^*)^2 + (D_i^* - D_j^*)^2 + (Y_i^* - Y_j^*)^2} \tag{4}$$

The two task T_i, T_j with the largest d_{ij} values are classified into different clusters C_1 and C_2. Then we calculate the Euclidean distance between other resources and these two center points, and select the task with the largest distance value as the center point of the cluster C_3. And we determine the type of task by comparing the value of S_i, D_i and Y_i, including time-sensitive requests, storage requests and bandwidth requests. Then we calculate the distance between the remaining tasks and this two tasks, they will be labeled as $dist_x$, $dist_y$ and $dist_z$. And we classify them according to distance, for example: if $dist_x > dist_y > dist_z$, the task is assigned to cluster C_1, otherwise it is assigned to others until these tasks divided into three different clusters. We can target resource scheduling based on task classification.

B. RESOURCE INTEGRATION

At present, the fog computing platform can be regarded as a virtualized resource pool, which is a cluster composed of many computers with different processing capabilities. Therefore, we integrate the different resources according to different index. In this article, we divide resources into three types: computing, storage, and bandwidth.

We assume that C_j represents the processing power of fog device N_j(i = 1, 2, ..., m), M_j represents the storage space of fog device, B_j represents the bandwidth of fog device. Then we allocate resources based on processing power, bandwidth capacity and storage capacity, as shown in the following formula:

$$C_j^* = \frac{C_j - C_{min}}{C_{max} - C_{min} + 0.1} \quad (5)$$

$$M_j^* = \frac{M_j - M_{min}}{M_{max} - M_{min} + 0.1} \quad (6)$$

$$B_j^* = \frac{B_j - M_{min}}{B_{max} - B_{min} + 0.1} \quad (7)$$

C_{min} and C_{max} represent the maximum and minimum processing power of fog device, respectively, M_{min} and M_{max} represent the maximum and minimum bandwidth capacity of fog device respectively, B_{min} and B_{max} represent the maximum and minimum storage capacity of fog device respectively. And we determine the type of resource by comparing the value of this index, for example: if $C_j^* > M_j^* > B_j^*$, the resource belong to the type of computing resource. Then, we can perform task scheduling according to three task types.

C. CONSTRAINT

As we all know, the task data needs to be transmitted by the user to the edge server or cloud server for processing through the base station. When multiple tasks need to be offloaded to the fog server, the total transmission demand may exceed the transmission capacity of the base station. At this time, the edge controller is required to obtain the current resource occupation of the server and determine the offload location of the task.

We assume that $Max_{connect_i}$ represents the maximum number of connections per device N_j, and $connect_i$ represents the connections of the current fog server N_j, at this time, we must ensure that the number of server connections cannot exceed the maximum number of current server connections, that is:

$$connect_i \leq Max_{connect_i} \quad (8)$$

At the same time, the following four constraints need to be met:

1) There is no priority constraint between tasks.

2) Tasks are not allowed to be preempted during execution. Each task can only be processed in one device, and one device can handle multiple tasks.

3) The completion time of a single task shall not exceed the expected completion time.

4) The number of tasks is greater than the number of devices.

V. OBJECTIVE MODEL

After clustering tasks and integrating resources, we begin targeted resource allocation. In fog computing, the fog devices are composed of heterogeneous network devices, so the latency of the service must be fully considered when scheduling tasks and the resource load often change. Therefore, we propose an objective model for time and load.

A. TIME MODEL

The fog computing is proposed to solve the problem of delay in cloud computing [30]. The processing speed of fog devices is different for different tasks, and the execution time will affect the service quantity for users. Therefore, the main goal is to efficiently reduce the completion time of task.

Assuming that K_i represents the size of task T_i, the time of fog device N_j to execute task T_i can be defined as follows:

$$Time_{ij} = \frac{K_i}{C_j} \quad (9)$$

Since the resource requestors request n tasks, the time required for the last fog device to complete the task can be calculated as follows:

$$Totaltime = \max_{j=1}^{m} \sum_{i=1}^{n} Time_{ij} \quad (10)$$

When *Totaltime* is smaller, the entire devices have the least amount of completion time.

B. LOAD MODEL

In fog computing, there are many load factors to affect fog devices [34].We assume that C_j^l represents the CPU utilization of fog device N_j, M_j^l represents the memory utilization of fog device N_j, B_j^l represents the bandwidth utilization of fog device N_j. In order to achieve the rationality of resource allocation, it is assumed that each fog device can normally receive the request of resource requestors and the load of each fog device is defined as follows:

$$Load_j = \alpha_1 * C_j^l + \alpha_2 * M_j^l + \alpha_3 * B_j^l \quad (11)$$

$$\alpha_1 = \frac{C_j^*}{C_j^* + M_j^* + B_j^*} \tag{12}$$

$$\alpha_2 = \frac{M_j^*}{C_j^* + M_j^* + B_j^*} \tag{13}$$

$$\alpha_3 = \frac{B_j^*}{C_j^* + M_j^* + B_j^*} \tag{14}$$

α represents the weight of the corresponding resource and $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

The average load of all fog devices is defined as $\overline{L}$, and the formula can be expressed as follows:

$$\overline{L} = \frac{\sum_{j=1}^{m} Load_j}{m} \tag{15}$$

The load of each fog devices is expressed by the standard deviation by Eq.(6), and the calculation formula can be expressed as follows:

$$Load = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(Load_j - \overline{L})} \tag{16}$$

When *Load* is smaller, the load of the entire device is more balanced.

VI. IMPROVED FIREWORK ALGORITHMS

A. FIREWORK ALGORITHMS

The firework algorithm (FWA) is a new swarm intelligence optimization algorithm proposed by Tan and Zhu [35] and others in 2010. The algorithm is inspired by the phenomenon of fireworks exploding in the night sky and generating sparks. The fireworks and the sparks generated by the explosion together form the whole body.

The basic principle of the firework algorithm: the explosion radius and the number of explosion sparks of each firework are not similar, the poor fitness value of the firework has a large explosion radius and few sparks. So, it has global search capability; while the good fitness value of the firework has a small explosion radius and a large number of sparks. So, it has local search capability. Fireworks performs resource allocation and information interaction in accordance with the values of each adaptive degree, so that the entire population can achieve a balance between global search capabilities and local search capabilities [36], [37].

The typical firework algorithm is as follows:

1) Generate fireworks of the specified population size, and calculating the fitness value of each firework according to the fitness function.

2) Fireworks start to explode and calculate the explosion radius and number of sparks of each firework.

3) Generate explosion spark and gaussian mutation spark according to the explosion radius and the number of sparks.

4) Fireworks, explosive sparks and gaussian mutant sparks will constitute a new candidate population, and the current optimal value can be found out from this population.

5) Determine whether the termination condition is met. If the output optimal value is met, stop iteration, otherwise enter step (6).

6) Find new fireworks according to the selection rules, and enter step (2).

For the research of firework algorithm, we analyze the explosion radius formula of the firework algorithm, and proposes an explosion radius detection mechanism in view of the possibility that the optimal spark radius may approach zero.

B. IMPROVED FIREWORK ALGORITHMS

1) ENCODING

In the swarm intelligence algorithm, the individual coding scheme is set according to the actual problem and the best solution is obtained by iterating to the optimal solution [38].

In this article, chromosome X_j consisting of m genes represents the fog devices, each of gene represents a fog device, and each gene corresponds to the corresponding task code, as shown in Table 1:

TABLE 1. Chromosome X_j.

Device	1	2	3	4	...	m
Request	3	0	1	2	...	4

The chromosome is encoded as $\{'3','0','1,5','2',..,'4'\}$, indicating that task 3 is assigned to device 1, tasks 1 are assigned to device 3, task 2 is assigned to device 4, task 4 is assigned to device m, and device 2 is not assigned to task.

Each task can and can only be executed by one device, and each device can execute multiple tasks.

2) FITNESS FUNCTION

The fitness value is an important indicator for evaluating the individual's pros and cons. In the firework algorithm, the fitness value determines the number of sparks that are produced by each firework. Fireworks with good fitness values generate more sparks, otherwise generate fewer sparks.

We assume that the spark of the lth iteration of fireworks j is the solution. Since the magnitudes of time and load are quite different, it will produce large errors in directly processing. So, it is necessary for two goals to be numerically normalized separately.

$$F_1(X_j^l) = e^{-Totaltime_j} \tag{17}$$

$$F_2(X_j^l) = e^{-Load_j} \tag{18}$$

Based on the normalization of the above goals, the fitness function is finally obtained by the summation method. The formula can be calculated by Eq.(19):

$$f(x_j^l) = \alpha * F_1(X_j^l) + \beta * F_2(X_j^l) \tag{19}$$

α and β represent the weights of time and load respectively, its value reflects the emphasis of the optimization goal, and the center of gravity can be changed by adjusting the value, where $0 \le \alpha,\beta \le 1$, and $\alpha + \beta = 1$.

3) EXPLOSION DETECTION MECHANISM

The formula for calculating the number of explosion sparks and the radius of a firework explosion is:

$$S_i = S\frac{Y_{max} - f(x_i^l) + C}{\sum_{i=1}^{N}(Y_{max} - f(x_i^l)) + C} \tag{20}$$

$$A_i = A_{max}\frac{f(x_i^l) - Y_{min} + C}{\sum_{i=1}^{N}(f(x_i^l) - Y_{min}) + C} \tag{21}$$

S_i indicates the number of subgeneration sparks in the ith fireworks. A_i represents the explosion radius of the ith fireworks. A_{max} represents constant, which indicates the largest explosion radius. S represents the maximum number of sparks in the fireworks subgeneration. Y_{max} indicates the worst fitness value of this generation. Y_{min} indicates the fitness value of the fireworks with the best fitness value of the current population. $f(x_i^l)$ represents the fitness value of the ith firework. C represents the minimum number of machines.

The fitness value of the best fireworks is better, but it may cause $f(x_i^l) - Y_{min} = 0$, and $\sum_{i=1}^{N}(f(x_i^l) - Y_{min}) \gg C$, therefore, formula (21) approximates to zero. Since the best fireworks can generate more sparks, if the radius of the sparks is zero, it is easy to be abandoned when the fireworks are finally selected, which waste computing time.

In response to the above problem, this article introduces a radius detection mechanism for fireworks. If the explosion radius of a firework is less than a certain threshold, we set its radius to this threshold:

$$A_i = \begin{cases} A_{min} & A_i < A_{min} \\ A_i & A_i \geq A_{min} \end{cases} \tag{22}$$

The threshold A_{min} changes continuously with the number of iterations, the calculation formula is as follows:

$$A_{min} = A_{max} - A_{max}\frac{l}{L} + C \tag{23}$$

l represents the number of explosive searches, L represents the number of total searches.

4) GAUSSIAN MUTATION

In the fireworks algorithm, the diversity of the population is further improved due to mutation operation. The calculation method is shown in Eq. (23).

$$x_{ik} = x_{ik} * g \tag{24}$$

g obeys the Gauss distribution of the mean value of 1 and the variance of 1, x_{ik} represents the value of the ith individual in the kth dimension.

5) SELECTION RULE

The individuals with the best fitness values are retained to the next generation from fireworks, explosive sparks and gaussian variation Sparks. And the rest $N - 1$ individuals are selected according to the following formula:

$$p(x_i) = \frac{D(x_i)}{\sum_{i=1}^{k} D(x_i)} \tag{25}$$

$$D(x_i) = \sum_{j=1}^{k} d(x_i, x_j) \tag{26}$$

k represents a set of fireworks, explosive sparks and gaussian variation Sparks. $D(x_i)$ represents the Euclidean distance between fireworks i and fireworks j, if the firework is farther away from x_i, the probability of being retained to the next generation is greater.

6) TIME COMPLEXITY

According to the steps of firework algorithm, the time complexity of population initialization is $O(N)$, the time of generating explosion radius is $O(N)$, the time complexity of generating sparks is $O(N)$, the time complexity of explosion detection mechanism is O(1), the time complexity of boundary processing is $O(N)$, the time complexity of the Euclidean distance is $O(N * M)$, and the time complexity of selecting the next generation of roulette is $O(N + N * M)$, where, N is the number of fireworks population and M is the number of sparks generated by fireworks, and all steps go through Tth iterations. So the total time complexity is as follows:

$$\begin{aligned} T_n &= O(T(N + N + N + 1 + N + N * M + N + N * M)) \\ &= O(5N * T + T + 2T * N * M) \\ &= O(5n^2 + 2n^3 + n) \end{aligned} \tag{27}$$

C. TASK SCHEDULING METHOD

The task scheduling process of this article(I-FASC) is as follows:

1): Initialize parameters, including metrics for tasks, virtual machines, and algorithms.

2): Task clustering and resource integration, divided into three different types.

3): Initialize the number of fireworks N, and select specific resources to optimize, each firework represents a solution.

4): Calculate fitness function value for fireworks according to Eq.(19) and the number of sparks and explosion radius of fireworks.

5): Introducing explosion radius detection mechanism and determining the radius of the current spark.

6): Selecte a firework to gaussian mutation randomly according to Eq.(23).

7): Selecte N fireworks as the next generation firework population.

8): If the maximum number of iterations is not reached, turn step 4, otherwise ends.

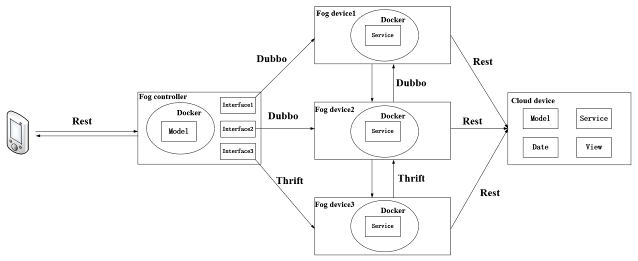

FIGURE 3. Simulation architecture.

VII. SIMULATION

A. EXPERIMENTAL ENVIRONMENT

This experiment uses Alibaba Cloud Server as the cloud server, a Core i5, 4G memory device is used as the fog controller, an 8-core smartphone with 4G memory is used as the edge device, and ten devices with different configurations are used as the fog server.

Task parameter settings: The range of the number of tasks is [10 − 1000], the range of the size of task K_i is [10000 − 50000], the expected storage space range of task S_i is [500 − 4000], the expected completion time range of task D_i is [0.1 − 50], and the expected bandwidth range of task Y_i is [1000 − 5000]. Fog server parameter settings: the range of processing speed C_j is [500 − 1000], the range of bandwidth range B_j is [500 − 6000], the range of storage space M_j is [521 − 4096], the range of CPU utilization C_j^l is [20 − 85], the range of memory utilization M_j^l is [10 − 90], the range of bandwidth utilization B_j^l is [10 − 90], the maximum server connections $Max_{connect_i}$ is [100 − 1000]. Firework algorithm parameter settings: one Gaussian spark, the total number of fireworks N is 200, the maximum number of sparks S is 80, the maximum explosion radiance A_{max} is 200, the maximum number of iterations L is 1000, the minimum number of machines C is 0.0001.

In this experiment, we deploy microservice architectures in different fog devices and containerize them in order to facilitate migration, and we use Dubbo and Thrift that is a transfer protocol to communication between services. Fog controller includes task cluster model, resource integration model and many algorithm model, it communicates with other servers through Dubbo, fog devices includes many services that deal with many task processing, fog device connects with the fog controller through Rest, the specific simulation architecture is shown in the following figure.

B. EXPERIMENTAL RESULTS

After all the parameters are configured, this paper has conducted two sets of experiments respectively for scheduling method and improved firework algorithm.

1) EXPERIMENT OF SCHEDULING METHOD

The experiments of task scheduling method include three sets of experiments: task clustering, task completion time, and edge server load.

When task scheduling started, we performed cluster experiments on 500 tasks. The experimental result is shown in the following figure 4, it can be seen from the figure that given tasks are divided into three different categories by our clustering strategy, which is convenient for scheduling in the next step.

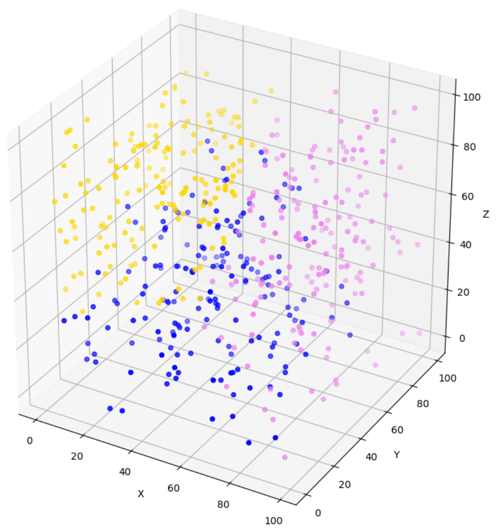

FIGURE 4. Cluster with 500 tasks.

X, Y, and Z coordinates represent the expected storage of the task, the expected completion time of the task and the expected bandwidth of the task, respectively, the value is calculated as a percentage. The yellow task point is the bandwidth requests task, the blue task point is the storage requests task and the purple task point is the time-sensitive requests task.

For the task completion time, we compare the scheduling method in this paper(I-FASC) with first come first service(FSFC), an aco-based algorithm(Rank-ACO) [39] and a double-fitness genetic algorithm(DFGA) [40] under different optimization weights.

We set two different weights separately, one of which is 0.8 and 0.2 for α and β, and the other is 0.2 and 0.8 for α and β. Figure 5, 6 show the algorithm execution time line

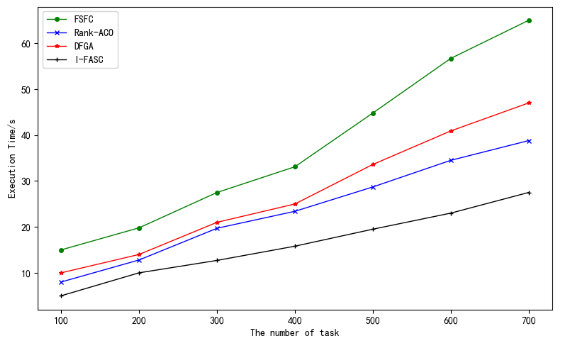

FIGURE 5. Comparison of execution time under different number of request tasks ($\alpha = 0.8$, $\beta = 0.2$).

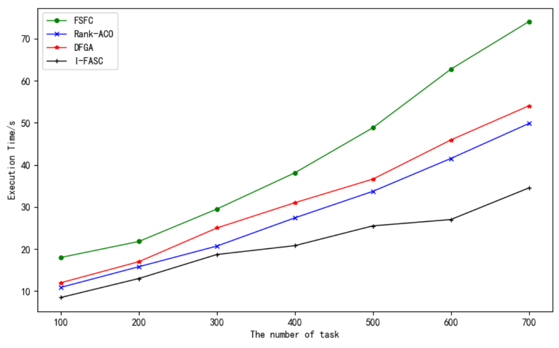

FIGURE 6. **Comparison of execution time under different number of request tasks ($\alpha = 0.2$, $\beta = 0.8$).**

chart in different task numbers, and figure 7, 8 show the load ratio line chart in different task numbers.

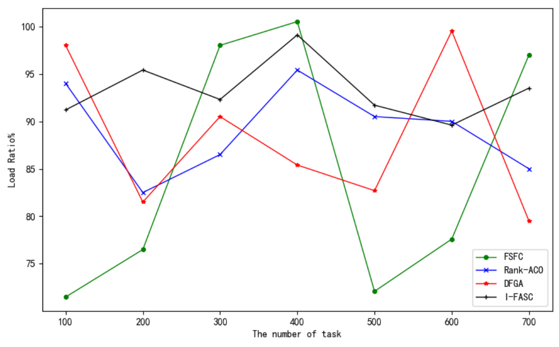

FIGURE 7. **Comparison of load ratio under different number of request tasks (α=0.8, β=0.2).**

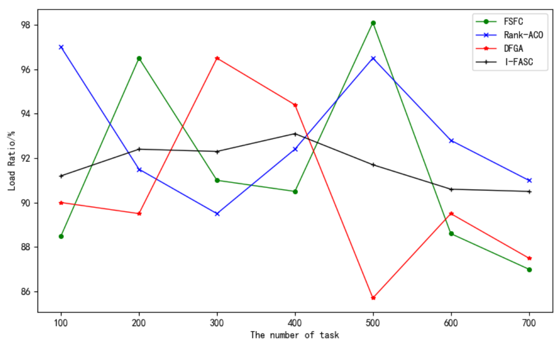

FIGURE 8. **Comparison of load ratio under different number of request tasks (α=0.2, β=0.8).**

Figure 5, 6 shows that with the increased number of tasks, I-FASC maintains lower task execution time compared with other algorithms in different weights. This is because we take into account the characteristics of tasks and resources, select different devices with improved firework algorithm according to the characteristics of the task, this can greatly speed up processing efficiency. DFGA only considers the average completion time of the task, and does not preprocess the task scheduling, resulting in the overall efficiency reduction. Rank-ACO although utilizes an upward rank value along with an insertion-based policy to further guide the ants toward quality solutions, but, with the increase of data, the performance of the algorithm is still limited.

It can be seen from the figure 7, 8 that with the different number of tasks, when using different scheduling method to schedule tasks in different weights, the load of each method varies greatly. It can make the device's resource allocation more balanced by using I-FASC to schedule. It is beneficial to give full play to the performance of each device, thereby improving the performance of the entire system.

In addition, we will send all tasks to the cloud device to process and use the same algorithm to compare with I-FASC. Figure 9 show that I-FASC has lower task execution time in different fog devices compared with cloud device. The effect of data transmission time on the fog computing model can be ignored in total time, therefore, fog computing is much better in real-time than cloud computing models. On the contrary, when the data processing time is much longer than the data transmission time, cloud computing can make better use of its computing advantages.

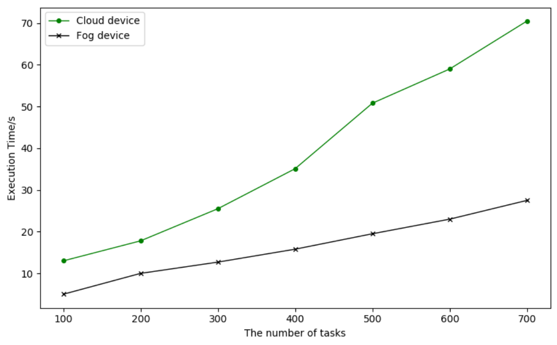

FIGURE 9. **Comparison under different devices.**

2) EXPERIMENT OF ALGORITHM

In order to verify the optimization abilities of the improved firework algorithm in Cloud-fog computing architecture, the experiments of the improved firework algorithm include two sets of experiments: explosion radius, algorithm performance.

Figure 10 shows the average of explosion radius of FW and I-FW after 200 tasks are executed, by comparison, as the number of iterations increases, the average explosion radius of fireworks of I-FA is smaller than FA, which means that each iteration of algorithm leaves more individuals with better fitness.

We compare different algorithm models under the same scheduling method and algorithm parameters, including DFGA, Rank-ACO and firework algorithm(FA), as shown in Fig.11

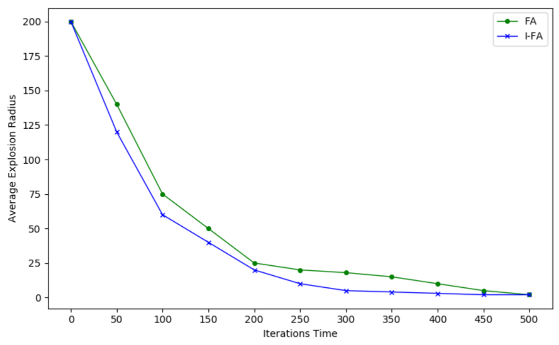

FIGURE 10. Comparison of average explosion radius.

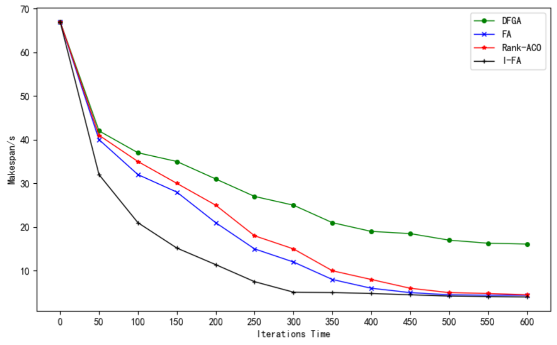

FIGURE 11. Performance comparison results of the four algorithms.

Fig. 11. shows the average performance curve of all task completion time-iteration times after 200 tasks are executed on ten fog devices. Among the three algorithms, I-FA has the least number of iterations and the fastest convergence speed. As the number of iterations increases, the task completion time of I-FA is consistently shorter than that of other algorithms. This can be seen that I-FA is more accurate than DFGA, Rank-ACO and FA. This is because of the complexity of the Double-Fitness genetic algorithm, such as crossovers and mutations. Due to the lack of initial pheromone, Rank-ACO needs a long time to search, resulting in its slow convergence speed. While I-FA has accelerated the emergence of optimal solutions by introducing an explosion detection mechanism.

VIII. CONCLUSION

Fog computing is a kind of lightweight computing, a new task scheduling method(I-FASC) is proposed for the characteristics of tasks and resources in this article, and then an improved genetic algorithm(I-FA) is proposed by introduce the explosion radius detection mechanism of fire-works. Then, we simulate a cloud-fog computing system, and this method can achieve better execution time and ensure better load in a short time by two sets of experiments in this system.

With the widespread application of fog computing, more people will be attracted to study this field. In this article, the task scheduling is studied in the fog computing, and some research results have been achieved. However, in the actual application development, there are still many shortcomings. For example, in the modeling of the task processing time and load optimization, the energy consumption of the fog devices processing task is not considered and there are many service indicators in fog computing. So modeling the energy consumption and selecting other service indicators as optimization goals can be considered in the next step.

REFERENCES

[1] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the Internet of Things: Security and privacy issues," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 34–42, Mar. 2017.

[2] W. Li, K. Liao, Q. He, and Y. Xia, "Performance-aware cost-effective resource provisioning for future grid IoT-cloud system," *J. Energy Eng.*, vol. 145, no. 5, Oct. 2019, Art. no. 04019016.

[3] S. Pang, T. Ding, A. Rodriguez-Paton, T. Song, and Z. Pan, "A parallel bioinspired framework for numerical calculations using enzymatic p system with an enzymatic environment," *IEEE Access*, vol. 6, pp. 65548–65556, 2018.

[4] S. Wang, D. You, and M. Zhou, "A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S 4PR," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4173–4179, Aug. 2017.

[5] X. Zeng, G. Xu, X. Zheng, Y. Xiang, and W. Zhou, "E-AUA: An efficient anonymous user authentication protocol for mobile IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1506–1519, Apr. 2019.

[6] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[7] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[8] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," in *Mobile Networks and Applications*. 2019, pp. 1–11.

[9] H. Gao, S. Mao, W. Huang, and X. Yang, "Applying probabilistic model checking to financial production risk evaluation and control: A case study of Alibaba's Yu'e Bao," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 3, pp. 785–795, Sep. 2018.

[10] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, "Toward service selection for workflow reconfiguration: An interface-based computing solution," *Future Gener. Comput. Syst.*, vol. 87, pp. 298–311, Oct. 2018.

[11] G. Xu, J. Liu, Y. Lu, X. Zeng, Y. Zhang, and X. Li, "A novel efficient MAKA protocol with desynchronization for anonymous roaming service in Global Mobility Networks," *J. Netw. Comput. Appl.*, vol. 107, pp. 83–92, Apr. 2018.

[12] N. Ran, J. Hao, Z. Dong, Z. He, Z. Liu, Y. Ruan, and S. Wang, "K-codiagnosability verification of labeled petri nets," *IEEE Access*, vol. 7, pp. 185055–185062, 2019.

[13] W. Duo, X. Jiang, O. Karoui, X. Guo, D. You, S. Wang, and Y. Ruan, "A deadlock prevention policy for a class of multithreaded software," *IEEE Access*, vol. 8, pp. 16676–16688, 2020.

[14] T. Song, Z. Wang, P. Xie, N. Han, J. Jiang, and D. Xu, "A novel dual path gated recurrent unit model for sea surface salinity prediction," *J. Atmos. Ocean. Technol.*, to be published.

[15] S. Pang, J. Yao, X. Wang, T. Ding, and L. Zhang, "Transmission control of MPTCP Incast based on buffer balance factor allocation in data center networks," *IEEE Access*, vol. 7, pp. 183428–183434, 2019.

[16] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing," *IEEE Access*, vol. 7, pp. 146379–146389, 2019.

[17] T. Song, Y. Wang, G. Li, and S. Pang, "Server consolidation energy-saving algorithm based on resource reservation and resource allocation strategy," *IEEE Access*, vol. 7, pp. 171452–171460, 2019.

[18] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 4, pp. 904–916, Jul. 2019.

[19] Y. Hou, N. Wu, M. Zhou, and Z. Li, "Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.

[20] H. Yuan, J. Bi, M. Zhou, and A. C. Ammari, "Time-aware multi-application task scheduling with guaranteed delay constraints in green data center," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1138–1151, Jul. 2018.
[21] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 4, pp. 926–937, Jul. 2019.
[22] S. Wang, Aorigele, G. Liu, and S. Gao, "A hybrid discrete imperialist competition algorithm for fuzzy job-shop scheduling problems," *IEEE Access*, vol. 4, pp. 9320–9331, 2016.
[23] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: A green computing paradigm to support IoT applications," *IET Netw.*, vol. 5, no. 2, pp. 23–29, Mar. 2016.
[24] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. Australas. Telecommun. Netw. Appl. Conf. (ATNAC)*, Nov. 2014, pp. 117–122.
[25] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. technol. (HotWeb)*, Nov. 2015, pp. 73–78.
[26] B. Yang, D. Wu, and R. Wang, "CUE: An intelligent edge computing framework," *IEEE Netw.*, vol. 33, no. 3, pp. 18–25, May 2019.
[27] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig, "Foggy: A framework for continuous automated IoT application deployment in fog computing," in *Proc. IEEE Int. Conf. AI Mobile Services (AIMS)*, Jun. 2017, pp. 38–45.
[28] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.
[29] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *Proc. 18th Asia-Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Oct. 2016, pp. 1–4.
[30] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Inf. Syst.*, vol. 12, no. 4, pp. 373–397, Apr. 2018.
[31] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II," *Wireless Pers. Commun.*, vol. 102, no. 2, pp. 1369–1385, Sep. 2018.
[32] P. Zhang and M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 772–783, Apr. 2018.
[33] K. Li, L. Guan, and C. Guo, "Cloud task scheduling strategy based on clustering and improved symbiotic organisms search algorithm," *Comput. Appl.*, vol. 38, no. 3, pp. 707–714, 2018.
[34] A. Rayes and S. Salam, "Fog computing," in *Internet Things From Hype to Reality*. Springer, 2019, pp. 155–180.
[35] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Proc. Int. Conf. Swarm Intell.* Springer, 2010, pp. 355–364.
[36] T. Si and R. Ghosh, "Explosion sparks generation using adaptive transfer function in firework algorithm," in *Proc. 3rd Int. Conf. Signal Process., Commun. Netw. (ICSCN)*, Mar. 2015, pp. 1–9.
[37] B. Zhang, Y.-J. Zheng, M.-X. Zhang, and S.-Y. Chen, "Fireworks algorithm with enhanced fireworks interaction," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 1, pp. 42–55, Jan. 2017.
[38] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, Sep. 2017.
[39] N. Edward and J. Elcock, "An efficient task scheduling algorithm for heterogeneous multiprocessing environments," in *Proc. Int. Conf. Inf. Comput. Technol. (ICICT)*, Mar. 2018, pp. 101–106.
[40] J.-F. Li and J. Peng, "Task scheduling algorithm based on improved genetic algorithm in cloud computing environment," *J. Comput. Appl.*, vol. 31, no. 1, pp. 184–186, Mar. 2011.

SHUDONG WANG received the degree from the Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2004. She is currently a Professor with the China University of Petroleum, Qingdao, China. Her current research interests include biostatistics, systems biology, and theory and application of DNA computational models.

TIANYU ZHAO received the B.S. degree from the Qingdao University of Technology, Qingdao, in 2017. He is currently a Graduate Student with the China University of Petroleum, Qingdao, China. His current research interests include cloud computing, fog computing, and artificial intelligence.

SHANCHEN PANG received the degree in computer software and theory from Tongji University, Shanghai, China, in 2008. He is currently a Professor with the China University of Petroleum, Qingdao, China. His current research interests include theory and application of Petri net, service computing, trusted computing, and artificial intelligence.