

An On-Site Elastic Autonomous Service Network with Efficient Task Assignment

Jiali You¹, Nannan Qiao¹, Jinlin Wang¹, Guoqiang Zhang², Yiqiang Sheng¹, Haojiang Deng¹, Xue Liu¹

¹National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, China

²School of Computer Science and Technology, Nanjing Normal University, Nanjing, China

¹{youjl, qiaonn, wangjl, shengyq, denghj, liux}@dsp.ac.cn, ²guoqiang@ict.ac.cn

Abstract—With the increasing demand of the big data processing and the personalized services in the Internet, the current service platforms such as cloud are becoming hard to meet the requirement of multiple-level quality of services (QoS) within a short execution time. In this paper, an on-site elastic autonomous service network named SEA is proposed to cope with the above problems by utilizing the available resources from edge devices. The proposed SEA is constructed by the devices at the edge of the Internet and provides multiple-level QoS service to users. The architecture of the SEA is firstly designed to maintain the edge devices efficiently and monitor the status of the available resources. Then, an efficient scheduling algorithm based on discrete particle swarm optimization (PSO) is proposed to assign the tasks to proper devices in the SEA. The evaluation shows that the average execution time based on PSO is reduced more than 20% in comparison with a state-of-the-art algorithm. The simulation also shows that the hybrid SEA-cloud service is superior to the cloud-based service.

Keywords—on-site, elastics, autonomous, service network, scheduling, particle swarm optimization

I. INTRODUCTION

With the development of the information system, especially the fusion of Internet, mobile Internet and Internet of things (IoT), there is a trend that the generated data size is too large to fit the transferring bandwidth. For this problem, current processing platforms, which are designed for the fast processing of complex services or problems, including parallel computing and cloud computing, have their limitations. With the rapid growth of various terminals, such as mobile phone and sensors, the need of the real-time processing of the data generated by the edge terminals brings large stress for the current cloud or process platforms. It mainly presents in the following aspects.

Cloud is too far away. Cloud computing is instead of the single point server by virtualization technologies to integrate the resources. It is seen as the fifth source besides water, electricity, gas and telephone [1]. However, in the users' point of view, most servers in the cloud have a long distance to users, which may not support the need of latency-sensitive services for various applications, e.g. IoT, security protection, and multimedia services.

It is hard to aggregate and process the big data with high efficiency. The architecture of the cloud is suitable for the global information collection, optimization, and the batch processing problems. If the huge generated data from edge terminals cannot be transferred to the cloud in time, some time-sensitive applications are hard to be executed completely on time.

The users' resource is almost waste. The service performance of the cloud is dependent on the deployed equipment. The current edge terminals, such as PCs and mobile phones, have strong computing ability and large storage space. This type of resource is not considered in the cloud-based service mode. If the edge resource can be seen as appended resources for Internet services, it may enhance the cloud service performance.

Based on the above analysis, we need a service platform, in which the users' resources at the edge of the network are used to process requests quickly, and the network and service characteristics are considered in the system design. For instance, in the online video service, if the users' requests of the file transcoding are processed in the vicinity, the finish time may be reduced a lot when compared with processing requests in the cloud. Besides, service devices at the edge can cache the content or mobile network information to help users near it obtain faster service. Therefore, we need an *on-site*, *elastic* and *autonomous* service network, where "on-site" means the requirement can be executed by geographically nearby devices and completed in time; "elastics" represents the resources can be assigned based on the service requirement; and "autonomous" supports the self-management of the large-scale network. We abbreviate this platform as SEA. In our work, we focus on how to design the architecture of the SEA service network, which can manage various types of devices at the edge of network, including the users' terminals. Besides, the amount of users' node is large in the Internet, and they are usually located disperse. To meet the latency and cost demand, how to assign the tasks in a user's request to proper execution node is the essential question for a SEA. Based on the above work, SEA can provide multiple-level QoS-based services, and can be seen as a complement of the cloud service. The contributions of our work include:

- We design an architecture of the SEA at the first time, in which the users' resources and deployed resources by the service providers are both considered and managed to provide the geographically nearby service for users.
- To find the proper execution nodes for the user's request in the SEA, a Discrete Particle Swarm Optimization (PSO) scheduling algorithm is proposed, which is suitable to improve the execution time and reduce the cost when compared with the algorithms usually used in the cloud.

In this paper, the related work is introduced in section II. The SEA architecture and function layers are described in section III. Section IV illustrates the scheduling model and the PSO based scheduling algorithm. The performance is evaluated

by the simulation through section V. Finally, the conclusions and the future work are presented in section VI.

II. RELATED WORK

About the SEA service network, many researchers make efforts on the related technologies. Fog Computing is a new concept proposed by Cisco [2,3]. The main idea about it is to expand the cloud to the edge of the network, make a new environment for the IoT, especially for wireless equipment connected network, and build a platform to support low latency and location awareness, mobility, large-scale management and real-time applications. By collaboration with the cloud, different time-sensitive works can be processed based on the different time limitation. For the equipment in the fog, it can be the weak ability facility, e.g. the access point, or the router; it also can be the high-performance equipment, like Cisco IOx[4]. Because of the confidential demand of the commercial production, most of the references about fog from Cisco focus on the characteristics, functions or application environment, but the architecture and related algorithms are not mentioned.

Fog is a new concept proposed in recent years, and there is still blank in many aspects. Many researchers devote to the implementation and promotion of it. For the large-scale application on the IoT, authors in [5] propose a Paas based programming model, Mobile Fog, which is a simplified programming abstraction and supports applications dynamically scaling at runtime. For the resource management and scheduling, authors in [6] present an approach to map heterogeneous resources to the time resource, and assign the resources by maximizing the function of service-oriented utility. In [7], authors investigate the placement and migration methods, which can ensure the latency and reduce the network utilization. Researchers in [8] discuss the reliability of the fog computing in the communication of smart devices, and analyze the related problem in the cloud.

In existing works, we found that most works about Fog belong to the concept discussion and the function analysis. Some questions, such as how to make them practice, how to implement the system, and how to design the algorithms about scheduling or ensuring reliability, still have large research space.

Except Fog, some researchers also have the similar ideas. Mobile Cloud Computing (MCC) is an architecture that supports the moving of storage and process from mobile devices to the cloud [9], which can reduce the energy consumption of batteries, promote the storage space and enhance the service reliability. Besides the system of MCC, how to manage the wireless bandwidth and how to incorporate with other nodes are also investigated in [10,11]. Contrast with the MCC, Mobile-edge Computing (MEC) is driven by the market and proposed by ETSI [12]. MEC provides cloud-computing capabilities and an IT service environment at the edge of the mobile network, such as the mobile base station, which may bring ultra-low latency and high bandwidth for applications. Currently, some industrial companies participate in developing the products and related techniques of MEC, but there is still a long way to making a mature environment. If

MCC and MEC are combined and expanded to IoT, the basic idea of the combined one is consistent with the Fog.

SEA service is the abbreviation of on-site, elastics, and autonomous service. It is a user-centric service based architecture, which can make the users' requirement processed close to the users to enhance the service quality and accelerate the execution speed. SEA service is a universal model for Internet and IoT, and it can support multiple-level of QoS services by collaborating with the cloud. It gives new ways for the Internet applications, such as big data processing, latency-sensitive service and security protection. This concept is first proposed in 2010 [13], and there is a lot of work investigated in recent years. The author in [14] analyzes the need of handling over one Zettabyte of data and explains the concept of SEA computing and authors in [15] introduce the basic functions about SEA. Researchers in [16] propose a resource-oriented protocol called SeaHttp to extend the REST style, which can reduce the average energy consumption of group communication significantly. The work in [17] studies the task-scheduling framework for the cloud-assist sensor network about SEA to save the energy. [18,19] endeavor on the elastic architecture for processors and prevents from the occurrence of soft errors via architecture elasticity.

Our research work is also a part of the SEA computing. In the current work of the SEA and the Fog, most researchers discuss about problems in the wireless environment or processors, and the edge resources just focused on the deployed equipment, not the users' terminals. In addition, if large-scale terminals and devices join in the SEA, some issues, e.g., how to design the architecture of the system, how to implement the autonomous management of the resources at the edge of Internet, and how to schedule the complex tasks, still exist. Presently, there is no detailed design of SEA's architecture in the previous work. Therefore, we first propose the SEA architecture in our work. Then, the scheduling approach for task assignment in the SEA is investigated. In addition, a hybrid SEA-cloud collaboration system is discussed, which can boost the service performance in the Internet.

III. ARCHITECTURE OF THE SEA

In this section, the architecture of the SEA is introduced. Then, the functions in the SEA are described

A. The Architecture and Deployment of the SEA Service Network

SEA is constructed by the SEA space, SEA space management system, service dispatcher system and network management system, which is shown in figure 1. The component of the SEA space is the equipment at the edge of the network, which is named the SEA nodes, including deployed service devices, user terminals and other equipment. These nodes usually have processing ability and can provide service when they are idle. SEA space management system is used to manage the resource in the SEA space. The responsibility of the service dispatcher system is to locate the proper SEA node as the entrance for each request. Besides, network management system is to monitor the equipment and configure the parameters.

When a service request is proposed by a user, it will be dispatched to the proper service node, which can be seen as the entrance node. The node is selected based on the location of the user and the load-balance strategies. After the request is sent to the entrance node, the node should analyze the request and decompose it into the task sequence upon the service logical relationship. In our work, the task is the minimum granularity unit in a service, which is usually a step in the request execution procedure. For instance, in the speech recognition, if we would like to train an acoustic model, there are several steps, including data preprocessing, feature extracting, and acoustic model training. Moreover, there are several algorithms to extract the features, which can be seen as different tasks. Therefore, if three types of features are extracted, there are five tasks in this type of service. Generally, the logical relationship among tasks is pre-defined by different services or users, which are related to the type of the service. For each SEA node, it maintains a neighbor list to record the nodes that have low latency or high bandwidth to it. To obtain the fast process, SEA node selects the candidates that can provide the required resources to the tasks from the maintained neighbor list. Then, the scheduling algorithm will be calculated to find the proper execution nodes for all tasks.

SEA service can quickly process the user requirement by the edge equipment that is close to the user. It suits for the local computing. Cloud service is based on the virtualization technologies, which particularly emphasize on the batch

processing. These two types of computing modes focus on different points. If they are combined, the service performance of the cloud can be improved. Usually, we can use the cloud as a super SEA node that has strong capability, and the scheduling process will assign the tasks to proper SEA nodes or the cloud.

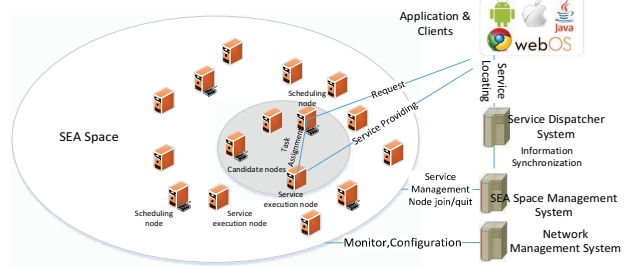


Fig. 1. The deployment architecture of the SEA Service Network.

B. Function Layers of the SEA Service Network

The system of SEA is a universal platform for different types of services. We develop a SEA system software that should be running on each SEA node. Through this system, all nodes in the SEA space can be autonomously managed, the data should be transferred among nodes, and resources and the network can be monitored. This system is constructed by several layers that are illustrated in figure 2. The description of different functions is as follows.

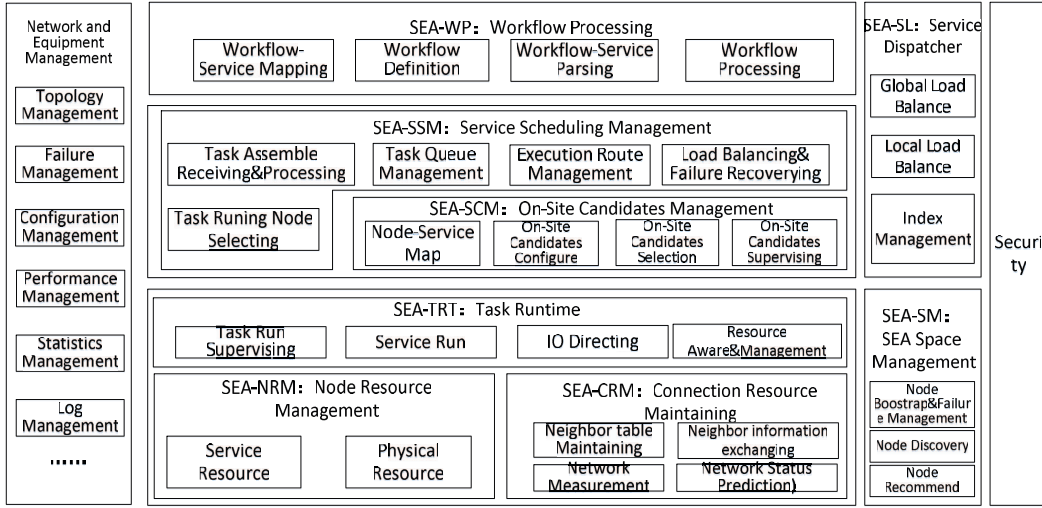


Fig. 2. The functions in the system of SEA Service Network

SEA Space Management is to manage all nodes in the SEA space, such as the node joining and quitting.

Service Dispatcher is to assign the user's request to the proper SEA node as the request entrance based on the location and load-balance strategies.

Workflow Processing is the external interface of SEA, which can receive the external request from the user, decompose it into logically dependent tasks, and export them to the service scheduling layers based on the logically dependent relationship.

Service Scheduling Management is responsible for the task scheduling, which should organize candidate execution

nodes for each task, choose the proper execution node, and then load the task to the selected node. In this process, the scheduling algorithm is the key point to guarantee the performance of the SEA service.

Task Runtime receives the task assignment from the service scheduling management, executes the task and monitors the running state of the task.

Node Resource Management takes charges to maintain the information about resources in the node.

Connection Resource Maintaining is a function to measure the network connection condition, exchange the information between nodes and maintain the neighbors.

In addition, **Network and Equipment Management** is a layer to monitor the devices in the system, which is a typical function in various network systems; the function of **Security** is responsible for security protection, attack detection, etc.

Based on the above system, all SEA nodes can be managed and monitored. The organization approach of nodes is out of the scope of this paper, and we will introduce it in another one. In this paper, we focus on the problem about the task scheduling in the SEA, which is the essential question to get the satisfying performance, all it will be discussed in the next section.

IV. THE TASK ASSIGNMENT IN THE SEA

In practice, the request proposed by a user can be seen as a workflow of the service, which can be decomposed into several tasks. Therefore, assigning the tasks to proper execution nodes is our purpose. In this section, the task-scheduling model is introduced first; then, a PSO based scheduling algorithm is applied to find the proper assignment schema for all tasks.

A. The Task Scheduling Model

The typical logical structure of the tasks in a workflow is the sequential, parallel and hybrid structure that has the mixture structure of sequential and parallel tasks. If a request contains K tasks $s = \{v_1, v_2, \dots, v_k, \dots, v_K\}$ and the node p_{rev} receives the request, the node should find the candidates from the neighbor list for each task. The selection criterion can be defined in advance or measured online. In our work, we define the selection criterion is that available resources should meet the

task requirement, and a candidate set P'_k is obtained for v_k . This process is done for all tasks, and a node set $P = \{p_{rev}\} \cup P'_1 \cup P'_2 \cup \dots \cup P'_K = \{p_1, p_2, \dots, p_i, \dots, p_N\}$ is generated as the candidate set of the request. The cost of each node in a time unit, which is usually seen as the price of renting the devices and VMs, or the price of the bandwidth, is $C = \{c_1, c_2, \dots, c_i, \dots, c_N\}$, and the cost limitation of the request is c_{up} . The target of the scheduling is to assign the tasks to the nodes that can minimize the execution makespan and guarantee the total cost no more than the cost constraint. In our work, makespan is the execution time of all tasks in a request, which is the main metric in our work. Moreover, the data transition time and the waiting time should also be considered in calculating the makespan. For the different structures of the tasks in a request, the objective function is shown in Table I. In this table, for the sequential task structure, the target is to find the assignment scheme that minimizes the sum of the execution time of all tasks. For the parallel task structure, the object is to find the assignment that minimizes the maximum execution time of any task. For the hybrid task structure, it can be seen as the combination of Q sequential paths. All paths compose the path set Φ and the q^{th} path is $s_q = \{v_1, \dots, v_k, \dots, v_{K'}\}$. The object of the hybrid structure is to find the scheduling plan that can minimize the makespan of the longest path.

Thereby, when the tasks relationship in a request is complex, the complexity of the optimization is very high, which may affect the efficiency seriously. We should design the heuristic algorithm to calculate the scheduling assignment.

TABLE I. OBJECTIVE FUNCTIONS OF DIFFERENT STRUCTURES IN A SERVICE WORKFLOW
(In the table, $t_{i,k}$ means the total time that task v_k executed by node p_i .)

Type of Structure	Sequential Tasks	Parallel Tasks	Hybrid Tasks
Objective Function	$\arg \min_{\{s_{i,k}\}} \sum_{k=1}^K \sum_{i=1}^N s_{i,k} t_{i,k}$ Subject to $\sum_{k=1}^K \sum_{i=1}^N s_{i,k} t_{i,k} c_i \leq c_{up}$ $\sum_{i=1}^N s_{i,k} = 1, \quad s_{i,k} \in \{0,1\}$ $\sum_{i=1}^N s_{i,k} t_{i,k} \leq t_{rev,k}$	$\arg \min_{\{s_{i,k}\}} \max_{\{v_k \in V\}} \left\{ \sum_{i=1}^N s_{i,k} t_{i,k} \right\}$ Subject to $\sum_{k=1}^K \sum_{i=1}^N s_{i,k} t_{i,k} c_i \leq c_{up}$ $\sum_{i=1}^N s_{i,k} = 1, \quad s_{i,k} \in \{0,1\}$ $\sum_{i=1}^N s_{i,k} t_{i,k} \leq t_{rev,k}$	$\arg \min_{\{s_{i,k}\}} \max_{\{s_q \in \Phi\}} \left\{ \sum_{k=1}^{K'} \sum_{i=1}^N s_{i,k} t_{i,k} \right\}$ Subject to $\sum_{k=1}^K \sum_{i=1}^N s_{i,k} t_{i,k} c_i \leq c_{up}$ $\sum_{i=1}^N s_{i,k} = 1, \quad s_{i,k} \in \{0,1\}$ $\sum_{i=1}^N s_{i,k} t_{i,k} \leq t_{rev,k}$

B. Discrete Particle Swarm Optimization based Task Scheduling Algorithm

Particle Swarm Optimization (PSO)[20,21] is a stochastic algorithm, which shows the obvious higher efficiency than the other stochastic algorithms, such as the Genetic algorithm. The basic idea of it is to mimic the social behavior as the particle and the target is to find the proper particle that satisfies the optimum conditions. In the SEA, we proposed a discrete data based PSO algorithm for task scheduling. Through defining two fitness functions to make the searching direction diversity, the algorithm may avoid falling into local solution to some extent. Moreover, the pruning function is applied in the searching process, which can reduce the search space significantly and improve the calculation efficiency.

In the SEA, the workflow of a service request is $S = \{v_1, v_2, \dots, v_k, \dots, v_K\}$, which contains K tasks. For the node p_{rev} which receives S , its scheduling candidates are $P = \{p_1, p_2, \dots, p_i, \dots, p_N\}$. Particle $X = \{x_1, \dots, x_k, \dots, x_K\}$ means the scheduling assignment schema of all tasks, where x_k represents the node index of task v_k assigned to. Meanwhile, two fitness functions are defined, $f_1(x) = \text{makespan}(X)$ and $f_2(x) = \text{cost}(X)$, which are the makespan and the cost of each assignment schema. In addition, to describe the calculation process, three operations are defined:

Operation 1: $Y = X \odot Z$ means the difference of two particles in the search space, where $X = \{x_1, x_2, \dots, x_k, \dots, x_K\}$,

$Z = \{z_1, z_2, \dots, z_K\}$ and $Y = \{y_1, y_2, \dots, y_K\}$. In this operation, if $x_k = z_k$, $y_k = 1$; otherwise $y_k = 0$.

Operation 2: $Y = P_x X \oplus \dots \oplus P_z Z$ means the value of each dimension in particle X is remained its own value with probability P_x and updated to the value in the same position of Z with probability P_z , and $\sum P = 1$.

Operation 3: $Y = X \otimes Z$ represents that if $x_k \neq z_k$, x_k is updated by z_k ; otherwise, the value remains unchanged.

Based on the above definition, the scheduling process is as follows.

a) Initialize M particles, and put them into $PSOList$. The initialized particles can be generated randomly.

b) Calculate $f_1(X)$ and $f_2(X)$ for all particles, and rank the $PSOList$ in ascending based on the $f_1(X)$ and $f_2(X)$ individually, by which two ranked lists are gotten. We select Top- L particles from two lists and put them in set_1 and set_2 separately. For all particles, the optimal value b_m will be calculated by $b_m = \min f_1(X_m)$, where X_m is any particle in the $PSOList$.

c) For each particle X_m in the $PSOList$, we calculate the new speed and new particles based on formula (1) and (2), where X_{ser1} is any particle in set_1 and X_{ser2} is the particle in set_2 , and X_m^* is a new particle.

$$V_m = P_1 X_m \oplus P_2 (X_m \odot X_{ser1}) \oplus P_3 (X_m \odot X_{ser2}) \quad (1)$$

$$X_m^* = X_m \otimes V_m \quad (2)$$

If $f_1(X_m^*) < b_m$, $b_m = f_1(X_m^*)$ and put X_m^* into $PSOList$.

d) Remove the particles that the cost is larger than c_{up} from $PSOList$. If the terminal condition is not reached, go to step b); else, stop and output b_m .

Based on the above process, all tasks in a request can be assigned to nodes by the PSO algorithm.

V. EVALUATION AND ANALYSIS

In this section, we design the simulation to mimic the requests processed by the SEA, and different metrics are studied. In addition, a SEA-cloud hybrid structure is simulated and analyzed.

A. Simulation Setup

In our simulation, we set 18 SEA nodes belonging to three ISPs or clusters. The bandwidth between nodes is different. In a cloud, the cost of execution is including the fee of renting the VMs and the fee of the bandwidth of VMs. It is usually calculated based on the renting time. In our simulation, the bandwidth and the cost between different ISPs are listed in Table II. The computation capability is usually described by the number of CPUs or the size of the memory. For a specific service type, the resources occupation can be estimated with the different data size for processing to some extent. To make the simulation convenient, we transfer the computation capability to the data processing speed. For the service type in our simulation, the computation capability of each node for the service is from 0.25-1M data/second, and it is following the

uniform distribution. The cost is about 0.3-1.5cent/s with the same distribution. We assume that users propose requests from nodes in ISP2 network and ISP3 network. The users in each ISP network propose 20 requests following the Poisson distribution with $\lambda=30s$; the data size of each request follows a Gaussian distribution with $N(\mu=700M, \sigma^2=400)$. Because in lots of applications about the big data processing, such as MapReduce, the parallel task structure is very common. Therefore, we analyze and compare the performance of the parallel-task structure in our simulation. The similar work can also be applied to the sequential structure and hybrid structure.

TABLE II. THE BANDWIDTH AND TRANSFERRING COST BETWEEN DIFFERENT ISP NETWORKS.

	Bandwidth(Mbps)			Cost(cent/s)		
	ISP1	ISP2	ISP3	ISP1	ISP2	ISP3
ISP1	50	15	30	0.1	1	5
ISP2	---	100	30	---	0	3
ISP3	---	---	80	---	---	0.2

The purpose of the scheduling is to assign tasks to proper nodes. To compare with other algorithms, we simulate Max-min, Min-min and Random scheduling algorithms as the baselines. These algorithms are easy to implemented and have good performance in clouds, which are usually used for comparison or for initializing the parameters in some stochastic algorithms[22]. Meanwhile, the execution time, waiting time and the transferring time should be considered in calculating the scheduling solution. The definition of the baseline algorithms are as follows.

- Max-min Scheduling: assign the largest task to the node which has minimum expected completion time.
- Min-min Scheduling: assign the smallest task to the node which has minimum expected completion time.
- Local Random Scheduling: assign the tasks to the nodes in the same ISP randomly.

B. Performance of the Task-Scheduling Algorithm

1) Comparing the performance without the cost constraint

In order to compare the scheduling performance of different algorithms, we set the maximum PSO iteration 8 and no cost constraint is imposed. The value in the different dimension of a particle is changing towards 3 directions, including remaining itself, changing to the particles in the set_1 and set_2 with probability p_1 , p_2 , and p_3 . In our simulation, $p_1=0.3$, $p_2=0.6$, $p_3=0.1$ which are pre-set by the experience. Figure 3 illustrates the points of makespan and cost of all requests under different algorithms. It clearly shows that the makespan of each request is better than the result of baseline algorithms. When we compare the relative performance of the makespan and the cost between PSO and other algorithms, as presented in table III, we found that the average makespan is improved significantly. The cost is even improved if compared with Max-min and Min-min algorithms.

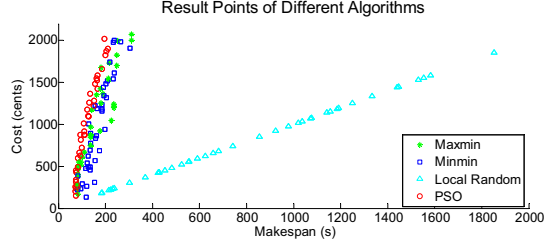


Fig. 3. Scattering points of makespan and cost of all requests under different algorithms.

TABLE III. AVERAGE RELATIVE PERFORMANCE OF PSO VS BASELINE ALGORITHMS.

	Makespan (%)	Cost (%)
PSO vs. Max-min	20.6	5.1
PSO vs. Min-min	27.6	0.8
PSO vs. Local_Random	72.1	-13.9

2) Comparing the performance with the cost constraint

When the cost is constrained, if the up-limit of the cost is low, the cost of serving a request may reach the up-limit before all tasks are executed completely. In this situation, the request may be declined. Therefore, we use the request declined ratio as another evaluation metric. The total cost of Max-min algorithm $C_{\max\min}$ is as the benchmark, and the up-limit of the cost is $C_{up} = \alpha C_{\max\min}$, where α is the weight. With different values of α , the declined ratio is shown in figure 4. We can see that the declined ratio of PSO is lower than Max-min and Min-min algorithms under different weight, which implies that the scheduling assignment from PSO is more reasonable than other algorithms. Although the declined ratio of Local Random

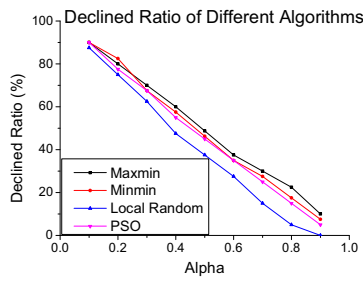


Fig. 4. Requests declined ratio of different algorithms.

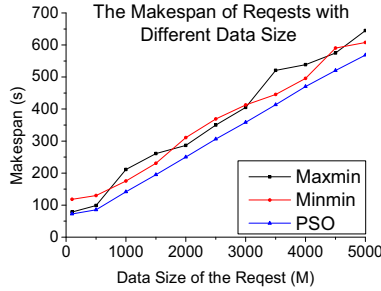


Fig. 5. The performance of request with different data size.

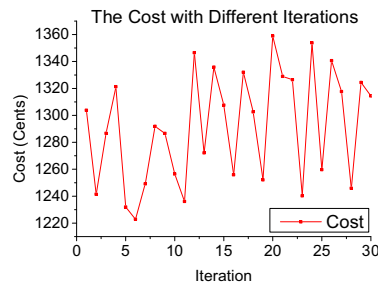
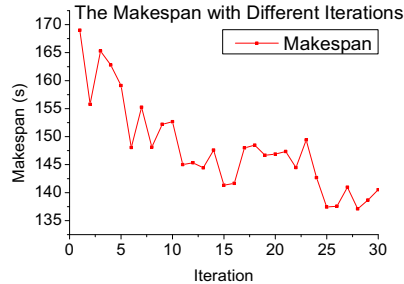
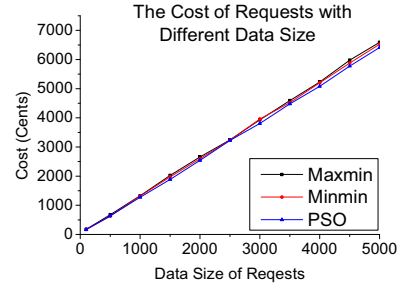


Fig. 6. The performance of PSO with different training iterations.

is the lowest, which is caused by the low transferring cost, the makespan of this algorithm for each request is quite high (Shown in figure3). Therefore, considering the tradeoff between the makespan and the cost, PSO offers the most satisfactory performance among all algorithms.

3) Analyzing the Performance of Different Parameters

A request with high or low workload may affect the performance of the scheduling. Thereby, we test the requests with different data size, and the same request is proposed by different nodes 10 times to get the reliable average result. The impact of the different data size of requests is compared in figure 5. Because the makespan of Local Random is much worse than other algorithms, we just compare PSO, Max-min and Min-min algorithms. In this figure, we can see that the makespan and cost increase almost linearly with the increasing of request's data size. The makespan of PSO in different points are much lower than Max-min and Min-min algorithms, but the difference in the cost is minor. Consequently, regardless of the request data size, PSO always shows outstanding performance in the scheduling.

Figure 6 shows the comparison of the performance with different iterations in the training process. It is obviously showing that with the increasing of the iterations, the makespan is reducing, although it may fluctuate in some points. However, there is no increasing trend in the cost curve. Meanwhile, we found that the training time almost increases linearly with the increasing of iterations. When the iteration equals 3, the performance of PSO is much better than the baseline algorithms, and the training process is quite efficient. Thereby, we can select the proper parameters based on the size of the request to get the balance of the efficiency and the precision.

4) The Impact of the Size of Neighbor List

In the SEA, more neighbors may bring more opportunity to select proper nodes in scheduling process, but it will increase the workload of maintaining the network obviously. To evaluate how many neighbors the SEA node should maintain, we test the PSO algorithm by a request that is proposed by different nodes several times, and the size of the data is 1000M. With the different number of neighbors, the scheduling performance is illustrated in figure 7, and the performance of the Max-min algorithm over all nodes is compared. It is clearly shown that the SEA can obtain the same performance if it just maintains 9 neighbors compared with Max-min algorithm. When it maintains more neighbors, the makespan can be reduced significantly. No matter how many neighbors a SEA node maintains, the cost from PSO is lower than the Max-min algorithm. Therefore, the PSO algorithm can get better performance, although the nodes just maintain a small size of the neighbor list.

C. Evaluate the Performance of Combining the SEA and Cloud

SEA can execute the tasks at the edge of network, which may be close to the users. Based on this structure, the resources at the edge of network can be utilized well, and the extra

resources can enhance the performance of the cloud service. Moreover, if the users' terminals join in the SEA service, they are usually free, such as P2P network, which can also reduce the service cost. To test the performance of the SEA architecture, we use devices in ISP3 as the nodes in the cloud, and users' nodes in ISP1 and ISP2 as the SEA nodes. If just cloud service is considered, all tasks should be processed just by the cloud nodes. If the SEA is appended, all tasks should be scheduled over nodes in SEA and Cloud. Figure 8 presents the makespan and cost of all requests based on these two structures. From the figure, we can see that if users nodes are organized as SEA and collaborated with the cloud, the makespan and cost can be improved significantly.

Besides, the SEA is suitable for the problem that uses the local information or needs to train the models quickly. If the SEA collaborate with the cloud, it can provide multiple-QoS services. For instance, in video services, the SEA service can be used to predict the local content trend or the user behavior, while the cloud is used to collect the global information, analyze the content distribution of the whole network, and help ISPs decide service strategies. Therefore, the collaboration of them may bring benefits in the network processing and the big data analyzing. How to use the SEA-cloud in applications is our future's work.

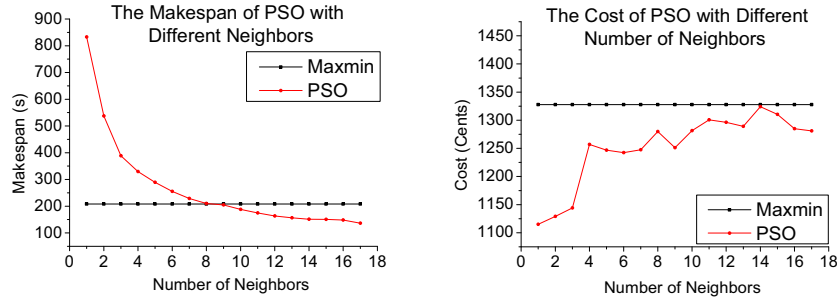


Fig. 7. The Performance of PSO with the different number of neighbors.

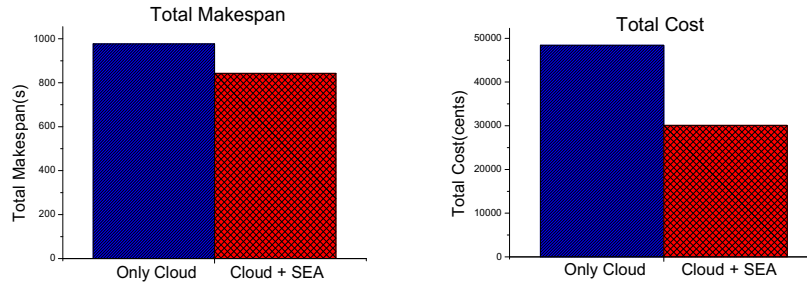


Fig. 8. The performance comparison of the cloud service and the cloud & SEA service.

VI. CONCLUSIONS

To meet the increasing need of multiple-level quality of services (QoS) within a short makespan in Internet or IoT, a new architecture, SEA service network is proposed, which composes by the deployed and users' devices at the edge of the network and provide geographically nearby service to users. In our work, the architecture of SEA is designed and explained. And the core problem in the SEA, how to schedule the tasks

from the user's request to get the low makespan and cost in the SEA, is discussed. To solve it, a discrete particle swarm optimization based scheduling algorithm is proposed, which can assign the tasks to proper execution nodes to achieve fast processing with the cost constraint. Our simulation presents that the proposed PSO scheduling algorithm has lower makespan, less declined requests, and lighter communication

workload than baseline algorithms. Moreover, a SEA-cloud hybrid structure is analyzed, which can improve the performance of the cloud service, especially for the applications that amount of users participating in. In our work, the collaboration of SEA and cloud is discussed simply, and we will focus on this question in the next work.

VII. ACKNOWLEDGEMENT

This work is supported by “Research of smart TV platform and service support environment”(XDA06040501) and “Youth innovation promotion association CAS”(Y529111601).

REFERENCES

- [1] R. Buyya , C. S. Yeo , S. Venugopal, et al, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” *Future Generation Computer Systems*, 2009, 25(6): 599-616.
- [2] F. Bonomi, “Cloud and fog computing: trade-offs and applications,” in *ISCA, EON Workshop*, 2011.
- [3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *ACM SIGCOMM MCC*, 2012
- [4] <https://developer.cisco.com/site/iox/technical-overview/>
- [5] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, “Mobile fog: a programming model for large-scale applications on the Internet of Things,” *SIGCOMM workshop MCC*, 2013, pp. 15-20.
- [6] T. Nishio, R. Shinkuma, T. Takahashi and N. B. Mandayam, “Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud,” *MobileCloud*, 2013, pp.19-26.
- [7] B. Ottenwalder, B. Klodehofe, K. Rothermel, and U. Ramachandran, “MigCEP: operator migration for mobility driven distributed complex event processing,” *DEBS*, 2013, pp. 183-194.
- [8] H. Madsen, G. Albeanu, B. Burtschy and F. L. Popentiu-Vladicescu, “Reliability in the utility computing era: Towards reliable fog computing,” *IWSSIP*, 2013, pp. 43-46.
- [9] H. D. Thai, C. Lee, D. Niyato, P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches,” *Wireless Communications and Mobile Computing*, 13(18), 2013, pp. 1587–1611.
- [10] G. Huerta-Canepa, D. Lee, “A virtual cloud computing provider for mobile devices,” *MCS*, 2010.
- [11] L. Zhang, X. Ding, Z. Wan, M. Gu, X. Li, “WiFace: a secure geosocial networking system using WiFi-based multi-hop MANET,” *MCS*, 2010.
- [12] <http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing>
- [13] M. Jiang, “Urbanization and information——The developing importunity of China,” *JieFang Daily*, 2010.
- [14] Z. Xu, “Cloud-Sea computing systems: towards thousand-fold improvement in performance per watt for the coming zettabyte era,” *Journal of Computer Science and Technology* 2014, 29(2):177-181.
- [15] J. Wang, J. Tian, J. You, etc., “Research and design of an on-site, elastic, autonomous network-SEA service system,” *Science China Information Sciences*, 2015, 45(10):1237–1248.
- [16] C. Hou, D. Li, J. Qiu, et al, “SeaHttp: a resource-oriented protocol to extend REST style for Web of things,” *Journal of Computer Science and Technology* 2014, 29(2):205-215.
- [17] H. Shi, D. Li, J. Qiu, et al, “A task execution framework for cloud-assisted sensor networks,” *Journal of Computer Science and Technology* 2014, 29(2):216-22.
- [18] Y. Chen, T. Chen, Q. Guo, et al, “Elastic architecture adaptable to millions of application scenarios,” *Network and Parallel Computing Lecture Notes in Computer Science*, 2012, pp. 188-195.
- [19] Y. Yin, Y. Chen, Q. Guo, et al, “Prevention from soft errors via architecture elasticity,” *Journal of Computer Science and Technology*, 2014, 29(2):247-254.
- [20] J. Kennedy, R. Eberhart, “Particle swarm optimization,” *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp.1942–1948.
- [21] Y. Shi, R. Eberhart, “A modified particle swarm optimizer,” *Proceedings of IEEE International Conference on Evolutionary Computation*, 1998, pp.69–73.
- [22] K. Kaur, A. Chhabra, and G. Singh, “Heuristics based genetic algorithm for scheduling static tasks in homogeneous parallel system,” *International Journal of Computer Science and Security*, 2010.