2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

An Improved Fast Search and Find of Density Peaks-based Fog Node Location of Fog Computing System

Xiaoqun Yuan*‡, Yan He*, Qing Fang* and Xiaowen Tong*, Changlai Du‡ and Yi Ding†,
*School of Information Management, Wuhan University, Wuhan, Hubei, China 430070 .
‡Department of Computer Science, Virginia Polytechnic Institute and State University, VA, USA 22043.
†School of Electronics and Information, Huazhong University of Science Technology, Wuhan, China, 430074

***Abstract*—As the most notably emerging wave of Internet deployments, Internet of Things (IoTs) requires mobility support, location awareness and low latency. Fog Computing, also termed edge computing, is a promising solution for IoTs by extending the Cloud Computing paradigm to the edge of Internet. But how to locate fog nodes' sites and determine the scale of each fog node is a main challenge of Fog Computing systems, especially for time sensitive Fog Computing systems. In this paper, we try to deal with this problem by proposing an improved Fast Search and Find of Density Peaks-based fog node location strategy to locate the fog nodes' sites and determine the resources for each located fog node. To this end, we firstly formulate the fog node location of Fog Computing systems as a clustering problem with multi-constraints. Then we propose an improved Fast Search and Find of Density Peaks-based fog node location algorithm, which introduces the time sensitive feature of IoT applications and improves the Fast Search and Find of Density Peaks clustering algorithm to make this clustering algorithm more robustness and adaptability. The experiment results show that our fog node location strategy not also can avoid the NP-hard problem of the traditional server placement strategies, but also has low time complexity.**

***Keywords*-Fog Computing; Internet of Things; Fog Node Location; Density Peaks clustering;**

I. Introduction

With the development of electronic communication technologies, more and more devices access to Internet directly and indirectly, which is called the Internet of Things (IoT). In the Internet of Things (IoT) paradigm, all physical items, also named as Things or devices, are connecting and talking to each other with machine-to-machine communications or person-to-computer communications[1][2]. These Internet connecting Things speed up awareness and response to events, leading to more output, high qualities of services and more faster emergency response. For example, in a factory, the imminent failure signaler on critical machines sent by their temperature sensors will avoid a costly shutdown with repair in time. In oil and gas exploration, pressure change information of oil pipelines generated by sensors on oil pipelines may slow down pumps automatically to avert a disaster. Thus, IoT has received attentions for years and is considered as the future of Internet. An estimated 50 billion Things will be connected to the Internet by 2020[3]. But due to the limited computation/storage capacities, most of Things can not process and store the data generated by them. To deal with this challenge, Fog Computing is proposed to extend the Cloud Computing paradigm to the edge of Internet by providing elastic resources and services to Things at the edge of network. Similar to Cloud Computing, Fog Computing integrates all resources such as computation, bandwidth and storage hosted on edge devices into geo-distributed service resource pools with Network Function Virtualization(NFV) and Software Defined Networks(SDN) technologies[4]. In Fog Computing systems, these gro-distributed resource pools at the edge of the network, which consist of resource-poor devices such as set-top-boxes, access points, routers, switches, base stations, and end devices, or resource-rich machines such as Cloudlet and IOx, are called fog nodes[5].

With these fog nodes, the data storage and the data processing happen outside of Things, reducing their storage and data processing overheads. In addition, fog nodes at the edge of the network may help to accelerate awareness and response to events by eliminating a round trip to the cloud for analysis, which may help to offload gigabytes of network traffic from the core network. Obviously, the performance of Fog Computing systems is significantly affected by the location selection and resources allocation for each fog node. However, due to the diversity of Things, how to locate fog nodes' sites and determine the scale of each fog node is a main challenge of Fog Computing systems. First of all, there are kinds of Things, such as sensors, mobiles and vehicles etc.. Each kind Things has its Internet accessing approach such as 4G-LTE and WiFi. Furthermore, each fog node consists of a wide variety of devices, which belong to different economic entities. For example, routers and switches are the basic elements of Internet and belong to their Internet Service Providers, while base stations are the basic elements of communication networks and belong to their communication companies. Each of these economic entities has their own interests, which makes service cost be the core factor of fog node construction. Thus, the diversity of Things, devices, Internet accessing approaches of Things as well as service cost and service performance make fog node location an important challenge of Fog Computing.

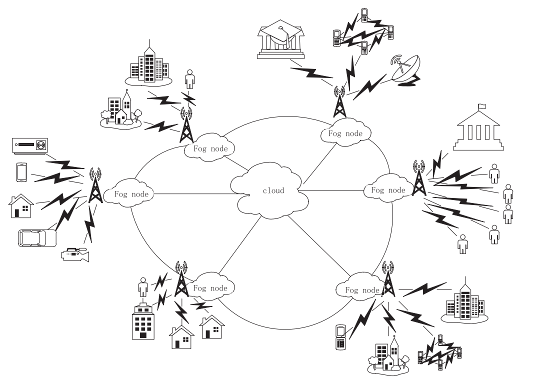

Figure 1. the Fog Computing architecture

In this paper. we try to deal with this challenge by proposing an improved Fast Search and Find of Density Peaks-based fog node location strategy to locate the fog node sites and determine the resources for each located fog node. To this end, we first investigate the Fog Computing architecture and analyze the influencing factors of fog nodes' location. Then we formulate the fog node location of Fog Computing as a clustering problem with multi-constraints. Based on our fog node location model, we introduce the MISE theory to the Fast Search and Find of Density Peaks clustering algorithm[6] and design an improved Fast Search and Find of Density Peaks-based fog node location algorithm to locate each fog node site and allocate the corresponding resource for each located fog node for our Fog Computing systems. Experimental results validate the effectiveness of the proposed fog node location strategy at last.

The rest of this paper is organized as follows. The background knowledge is presented in Section II. Section III describes the proposed fog node location model and the corresponding algorithm by using an improved Fast Search and Find of Density Peaks method in detail. Experimental results are presented and discussed in Section IV, and finally, the concluding remarks are presented in Section V.

II. Background Knowledge and Motivation

In this section, we present a brief introduction of our Fog Computing architecture and formulate the fog node location of our Fog Computing architecture.

A. Fog Computing

Fog Computing is a term proposed by Cisco to extent the cloud computing paradigm from the core of Internet to the edge of Internet to provide time sensitive services for IoT applications, such as vehicle, smart grid and wireless sensor and actuator networks[7][8]. Similar to Cloud Computing, it is also a highly visualization service platform that all kinds of resources such CPU, bandwidth and storage of different network devices are integrated into fog nodes to provide computation, storage, and networking services between end devices(Things) and traditional cloud servers[9]. Customers can develop, manage, and run software applications on fog nodes, as shown in Fig. 1. Fig. 1 presents a fog-computing reference architecture, which be taken as a three tier network structure. Tier 1 is the bottom-most layer encompassing all Things and Thing Networks, which sense a multitude of events and transmit the sensed data to its upper layer. Tier 2 is the Fog Computing layer, consists of geo-distributed fog nodes, including servers and devices, such as routers, gateways, switches, and access points etc., which are responsible for processing, computing, and storing the sensed data temporarily. Tier 3, also called as the cloud computing layer, is the upper-most layer in this architecture, which consists of datacentre-based cloud, processes and stores an enormous amount of data.

Obviously, in Fig. 1, Things and Thing Networks generate all kinds of raw data and transmit this data to its nearest fog node. Then this data will be processed by real-time or latency-sensitive applications implemented in these fog nodes. All necessary and long time storage data processed by fog nodes is gathered and transmitted to cloud computing layer to process for business decision and store for long time, as shown in Fig. 2. Fig. 2 shows the data processing framework of Fog Computing, covering the whole ranges of time scales from milliseconds to months. Similar to Fig. 1, this framework is also consists of three level. The bottom level is also the Things layer, which aims to generate raw data and little actionable data analysis, consisting of Things and Thing Networks and covering time ranges from millisecond-sub seconds to sub-seconds. At this level, sensed data is generated and exchanged among Things with machine-to-machine (M2M) interactions. Besides, little real-time actionable data is analyzed according to the capacities of Things and most of the sensed data is transmitted to the second level, the fog computing layer. The second level (seconds to sub-hours) consists of geo-distributed fog nodes and aims to provide most real-time and time sensitive services for Things and Thing Networks. Besides, some local and time sensitive decisions, such as local traffic management strategies, are determined based on the information of different fog nodes for stabilization of local Thing Networks. Based on the data processing of fog nodes, important but nonsensitive business information and history data are transmitted to the upper layer, the cloud computing layer. At this level, information covering a wide geographical region, such as a whole city or even a whole country and beyond, is gathered to achieve business decisions for enterprisers or management policies for administrators. Obviously, by deploying a geo-distributed fog computing layer between Things and Cloud Computing, most of sensed data are processed in fog nodes at the edge of Internet, which not only can reduce service latency, but also off-loads most of Internet traffic.

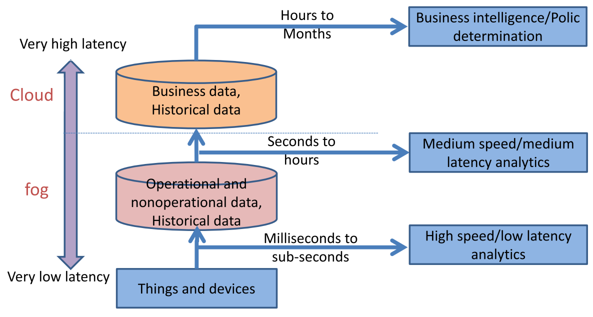
Very high latency
Hours to Months
Business intelligence/Polic determination
Cloud
Business data, Historical data
Seconds to hours
Medium speed/medium latency analytics
fog
Operational and nonoperational data, Historical data
Milliseconds to sub-seconds
High speed/low latency analytics
Very low latency
Things and devices

Figure 2. data analytic framework of Fog Computing

B. Fog Node Location of Fog Computing

In our Fog Computing architecture, both fog computing layer and Thing layer are heterogeneous. At Thing layer, there are kinds of Things, such as sensors, mobiles and vehicles, who access Internet with different approaches. For example, cameras, vehicles, traffic lights and sensors in parking lots etc. are the Things of smart transportation. Sensors form Thing networks and then access Internet, vehicles may access Internet through WiFi or base station, even some vehicles form Thing networks to access Internet while cameras and traffic lights may access to Internet directly. At Fog Computing layer, in order to provide low latency and location awareness services for all kinds of Things and Things Networks, various kinds of Internet devices and servers used to provide more storage and computation capacities are involved. Obviously, managing networks of billions of heterogeneous devices, which run one or more services and serve Things and Things Networks flexibly and effectively, is incredibly challenging and complex.

A potential and intuitive solution for this challenge is to divide Things into Thing clusters and integrate devices into virtual fog nodes according to Thing clusters. To this end, Things and Thing Networks are classed into geo-distributed clusters with clustering algorithms, then all resource of the service devices and servers for each Thing cluster are integrated into one virtual fog node with emerging techniques, such as SDN and NFV. At last, P2P technologies are applied to these geo-distributed fog nodes to form a fog computing network, which allows fog nodes cooperating better and fog computing networks more scalable. However, fog nodes consist of various kinds of Internet devices. The owners of these devices are independent from each other and pursue their own profits. Thus, service cost is an important factor for fog node location of Fog Computing. Besides, there are some applications running on each fig node, which means fog nodes should provide unique quality of service to Things for each application. Thus, Fog Node location of Fog Computing can be formulated as a clustering-based multi-constrained optimization problem.

III. Fog Node Location Strategy of Fog Computing

In this Section, we describe our fog node location strategy of Fog Computing in detail. To this end, we first model fog node location, then an improved Fast Search and Find of Density Peaks-based fog node locating algorithm is proposed.

A. Fog Node Location Model

As discussed in Section II.B, the fog node location is a clustering-based multi-constrained optimization problem. Considering one Fog Computing system with N Things, Thing i requires storage resources rs_i and computation resourcesrc_i to satisfy its service. To simplify our model, we assume that all Things have the same maximum response latency $MDel$ and can be served by P virtual fog nodes. Fog node Nod_p has M kind of devices $KDev$, in which the number of kind of devices $KDev_pm$ is L_pm. Let device Dev_pmj denote the jth device of kind m, del_{pmji} mean the service delay that the jth device serves Thing i and its service resources with the unit resource cost $C_pmj = \{Cs_pmj, Cc_pmj\}$ is $R_pmj = \{Rs_pmj, Rc_pmj\}$, where Rs means storage resources, Rc denotes computation resource, Cs presents the unit resource cost of Rs and Cc presents the unit resource cost of Rc. Then the fog node location can be formulated as follows:

$$\min: \ k_1 * \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{j=1}^{L_{pm}}\sum_{i=1}^{N} del_{pmji}x_i + k_2 * \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{j=1}^{L_{pm}} C_{pmj}R_{pmj}$$
$$\texttt{s.t.} \quad \begin{cases} x_i = 1, if\, Thing\, \mathrm{i}\, is\, served, otherwise, x_i = 0 \\ del_{pmji} \le MDel \\ \left[\sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{j=1}^{L_{pm}} C_{pmj}R_{pmj}\right] \ge \sum_{i=1}^{N}(rs_i + rc_i) \end{cases} \tag{1}$$

where $\sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{j=1}^{L_{pm}}\sum_{i=1}^{N} del_{pmji}x_i$ denotes the total resources TR of Fog Computing and $\sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{j=1}^{L_{pm}} C_{pmj}R_{pmj}$ denotes the total cost TC for these resources. In fact, unit resources' cost of one kind device is different from that of other kinds, even the cost of the same kind device at one location is different from that of other locations. But at the same location, the resource capacities and unit resources cost of each kind is really the same. Then, the total resources TR and resource cost TC of this fog node can be described as $\sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{i=1}^{N} L_m del_{pmi}x_i$ and $\sum_{p=1}^{P}\sum_{m=1}^{M} L_m C_{pm} R_{pm} \sum_{i=1}^{M} L_i * C_i * R_i$ respectively.

Since both devices and servers of one fog node and Things served by this fog node are at the edge of Internet, it is reasonable to consider that the service scale of this fog node is homogeneous. As a result, the bandwidth resources between each fog node and its serving Things are unlimited, the response latency of one fog node is the linear function of the distance between this node and its Thing. Furthermore, all resources of devices and servers, which serve one Thing cluster are integrated into a virtual fog node. In addition, fog nodes always use the most "suitable" device to response one Thing's request and each kind of devices has the same unit resource cost. Therefore, the resource cost of devices can be replaced by Things. That is to say, the property of service cost for one Thing is applied to substitute its service resource cost, including storage service cost and computation service cost. In our paper, we let a virtual fog node with all kinds of resources replace these geo-distributed devices and servers to serve Things and Thing Networks. The service latency can be described as the distance between this fog node and its serving Thing. After normalizing service resource and resource cost of all Things, fog node location can be redescribed as follows:

Considering a Euclid space with N points, each point denotes one Thing. The resource and resource cost, which are used to serve this Thing, are considered as the properties of this point. Then these points are clustered as several clusters to minimize the weighted distance between these points and their clusters centroid. Besides, all clusters should meet the constraint that the distance of each point to its cluster centroid is less than the threshold, shown as follows:

$$\min: \; k_1 \sum_{i=1}^{M}\sum_{j=1}^{M} d_j y_{ij} C_i x_i + k_2 \sum_{i=1}^{M}\sum_{j=1}^{M} L_{ij} y_{ij}$$

$$\texttt{s.t.} \quad \begin{cases} x_i, y_{ij} \in \{0,1\} \\ L_{ij} = d_j D_{ij} \\ \left[L_{ij}\right] \geq DThr \end{cases} \tag{2}$$

where k_1 and k_2 are weights of object and $k_1 + k_2 = 1$, x_i denotes the point selected as a virtual fog node, y_{ij} means the point j served by virtual fog node i, L_{ij}, which should satisfy the service quality of point, that is, $L_{ij} \geq DThr$, is the service latency when fog node i serves point j. Obviously, Eq. (2) is a NP hard problem. Thus, we propose an improved Fast Search and Find of Density Peaks-based fog node location algorithm to deal with this problem.

B. Improved Fast Search and Find of Density Peaks-based Fog Node Location algorithm

Essentially, Fast Search and Find of Density Peaks algorithm is a fast clustering algorithm. Clustering is a division of samples into groups of objects similar in some metrics such as distance or similarity, and well investigated for its wide variety flied including Marketing, Computer, Biology and Libraries etc.. Generally, clustering algorithms can be classified as Partitioning Relocation Clustering, Hierarchical Clustering and Other Clustering Techniques. Hierarchical clustering builds a cluster hierarchy, also known as a dendrogram or a tree of clusters, including agglomerative (bottom-up) and divisive (top-down)[10][11]. Partitioning algorithms divide data into several subsets with greedy heuristics, which consists of Probabilistic Clustering[14], K-Medoids Methods[12] and K-Means Methods[13]. Besides Partitioning Relocation Clustering and Hierarchical Clustering, there are some other clustering algorithms proposed, such as the cluster algorithms based on Fuzzy Theory[15], clustering algorithms based on Graph Theory[17] and Density-based clustering algorithms[16] etc.. Recently, Clustering was introduced to avoid the NP hard problem of traditional server placement strategies [18][20], such as NetClust[21]. NetClust proposed a K-means-based clustering server placement sites determination algorithm to select the server placement sites for large scale service platform, such as Cloud computing and datacenters. It clustered the points N into M clusters and achieved the centroid for each cluster as the server placement sites. Although K-means cluster algorithm is simple and carried out easily, it has the time complexity of $N(NKI)$ affected by the initial centroid selection. To overcome this problem, we introduce the Fast Search and Find of Density Peaks clustering algorithm and propose an improved Fast Search and Find of Density Peaks-based fog node location algorithm.

The Fast Search and Find of Density Peaks clustering algorithm was proposed by A. Rodriguez and A. Laio in 2013, which was based on the idea that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities[6]. According to these two characters, two properties of data points: 1) ρ_i and 2) δ_i were defined and clustering algorithm is designed. In this algorithms, ρ_i means the local density of point i, which can be calculated as follow:

$$\rho_{\mathrm{i}} = \sum_{l=1}^{N} X(d_{ij} - d_c) \tag{3}$$

where d_{ij} is the distance between point p_i and p_j, d_c is the cutoff distance.

Based on ρ_i, δ_i also defined as :

$$\delta_i = \begin{cases} min(d_ij) & \text{if } \ni j, \text{ s.t: } \rho_j > \rho_i \\ max(d_ij) & \text{if } \ni j, \text{ s.t: } \rho_j < \rho_i \end{cases} \tag{4}$$

Eq. (4) shows that δ_i denotes the distance of data point i to closest higher dense point.

Obviously, ρ_i is the number of points closer than d_c to point i, which shows the probability of point i as the cluster center from a local perspective, while δ_i means minimum distance of point i and a nearest high-density point, which describes the probability of point i as the cluster center globally. Then ρ_i and δ_i are plotted to select points as the cluster centers through manual manners. After achieving clusters centers, other points are assigned to the nearest cluster in a single round accurately. However, due to cluster centers determined by manual, this algorithm is not carried out automatically. Besides, there is no reasonable decision strategies for the cutoff distance d_c, which affects the efficiency of this algorithm significantly. On the other hand, Eqs. (3) and (4) show that the Fast Search and Find of Density Peaks clustering algorithm also belongs to the k-ernel density estimation approach, which always implements Mean integrated squared error (MISE) as quality estimation criteria for optimal choice of smoothing parameter. In fact, Mean integrated squared error (MISE) has been investigated for independent and identically distributed random variables by Z. I. BOTEV[19]. Due to the huge amount of Things and random access to the mobile Things, it is reasonable to conclude that these N points are independent and identically distributed in the Euclid space. Therefore, we implement Mean integrated squared error (MISE) to improve the Fast Search and Find of Density Peaks clustering algorithm, which is given below:

$$MISE\hat{f}(t) = E_f \int [\hat{f}(x;t) - f(x)]^2 dx \tag{5}$$

In fact, Eq. (5) is conveniently decomposed into integrated squared bias and integrated variance components, which can be described as follows respectively:

$$MISE\hat{f}(t) = \int (E_f[\hat{f}(x;t) - f(x)])^2 dx + \int var_f[\hat{f}(x;t)]dx \tag{6}$$

Eq. (6) shows that the integrated squared bias of Mean integrated squared error (MISE) can be described as the expectation of random sample and the integrated variance can be denoted by the standard deviation of random sample. On the other hand, the Fast Search and Find of Density Peaks clustering algorithm considers that a cluster center has higher ρ and large δ than non-center data-points. However, there always exists phenomena that a single cluster contains more one density peak. In this case, each different density peak can be considered as a potential cluster center, which makes it difficult to select the exact number of clusters, so the decision graph is plotted to assist the appropriate cluster center selection. Therefore, we introduce Mean integrated squared error (MISE) to Fast Search and Find of Density Peaks clustering algorithm to improve its efficiency. To this end, we firstly calculate the mean $E(d)$ of all distance and let $d_c = 0.2 * E(d)$. Then Eq. (3) can be transferred as:

$$\rho_\mathrm{i} = \sum_{l=1}^{N} X(d_{ij} - 0.2 * E(d)) \tag{7}$$

Based on Eq. (7), the mean $E(\rho)$ and the standard deviation $\sigma(\delta)$ can be calculated, which denote the integrated squared bias and the integrated variance of the cluster center respectively. Then $E(\rho)$ and $\sigma(\delta)$ can be implied to select the appropriate cluster centers with the follows inequalities.

$$\mathrm{EC_i} \geq E(\rho_i) \tag{8}$$

$$\mathrm{Del_i} \geq \sigma(\delta_i) \tag{9}$$

Eqs. (8) and (9) show that local cluster centers are the data-points with larger distance and higher densities as compared to the neighbors data-points. Obviously, Eqs. (7), (8) and (9) can be implemented to determine the cluster centers, which is based on the properties and location of points without the specific distance constraint. But each Thing of fog computing systems has its tolerable response delay. Thus we design our improved Fast Search and Find of Density Peaks-based fog node location algorithm based on Eqs. (7), (8) and (9) and the specific distance constraint, shown in Algorithm 1.

Algorithm 1 Improved Fast Search and Find of Density Peaks-based fog node location algorithm.

```
Input:
    The Number of Things, $N$;
    The service resource and unjoint resource cost for Thing $i$, $r_i$ and $c_i$;
    The location of Thing $i$, $p_i$;
    The minimum maximum latency of Things served by fog nodes, $dmin$;
Output:
    The number of Clusters, $M$;
    The location of Cluster $i$, $CP_i$;
    The service Things' number of Cluster $i$, $L_i$;
    The total service resources of Cluster $i$, $TS_i$;
    The total service cost of Cluster $i$, $TC_i$.
1: Calculate the distance of each pair points $d_ij$;
2: Calculate the mean $E(d)$ of distance $d_{ij}$ and get $d_c = 0.2 * E(d)$;
3: Calculate $\rho_i$ and $\delta_i$ with Eqs. (7) and (4) respectively;
4: Calculate $E(\rho)$ and $\sigma(\delta)$;
5: Achieve cluster number and their centers by implementing Eqs. (8) and (9);
6: Achieve cluster number and their centers by implementing Eqs. (8) and (9);
7: Assign remaining points to each cluster center and calculate their service distances;
8: Repeat Step 3 to Step 7 till all service distances below $dmin$;
9: return $M$, $CP_i$, $L_i$, $TS_i$ and $TC_i$;
```

In Algorithm 1, Step 1 aims to obtain the weighted distances of each pair points in the given Euclid space, in which the service resources and the corresponding service cost of each point are normalized, then they are attached to its distance as the weights. Step 2 calculates the cutoff distance which determines the data set size of the potential cluster centers and affects the efficiency of our location algorithm. Step 3 to Step 8 is the core of our location algorithm, which determines the location, service resources and service cost of each fog node through iteration. Step 3 to

Step 7 choose the candidate locations based on Eqs. (4), (7), (8) and (9), then these candidate locations are tested with $dmin$, which denotes that all Things should be served with satisfied service quality, by the Step 8. If the delay constraint is not satisfied, the candidate locations can't be taken as the point set of the potential cluster centers and Step 3 to Step 8 will be carried out again. This process should be carried out again and again till the delay constraint is satisfied. Based on the selected cluster centers, Step 9 returns the location of each selected fog node as well as its service resources and cost for these resources.

IV. EXPERIMENTS

A. Experiment Establish

To evaluate the efficiency and performance of our proposed method, clusters created by Improved Fast Search and Find of Density Peaks-based fog node location algorithm(IFSFDPFNL) are compared to K-means and Integrated Optimization Problems(IOP) described as Eq. (1). Both K-means and IOP are applied to server placement and investigated well. K-means-based server placement strategies is introduced for large scale applications such as cloud and datacenters[21] while Integrated Optimization is always applied to model traditional server placements, which formulate the traditional server placements as an optimization problem with different metrics, for example, service delay and deployment cost etc.. In our experiment, these three algorithms are implemented in the same Euclidean sample space with the same points, which are denoted as Things of Fog Computing systems. Each Thing is attached two values between 0 and 1. One value aims to simulate thing's required resources, which are generated randomly in our experiment. The other value is used to denote the service cost for unit resource if this point is selected as the fog node. According to the results of the previous work, there is no significant difference of the service costs for unit resource in certain area[22]. Therefore, we divide the sample space into 25 subspace evenly and generate a value between 0 and 1 as the unit resource service cost for one each subspace. In addition, since fog node is virtual service node consisting of devices and servers, any location in the sample space selected by fog node location algorithm can be considered as a fog node. Thus, for each fog node, the unit resource cost is determined by its location. Due to the features of independence and randomness of Things, each of them can be considered as a random point. Thus, we use a set of pseudorandom samples to imitate them, which are generated with mean 1 and standard deviation 100. The sample number increases from 5000 to 10000 with step of 100. Since the algorithm time complexity of K-means is greatly influenced by the initial centroid selection, we set a time threshold of successful clustering with $200s$. That is to say, when the server selection time is less than $200s$, we consider this k-means clustering is successful, vice versa. To eliminate the impact of the initial centroid selection as possible, we repeat these two experiments 10 times and get the average clustering time of successful server selection.

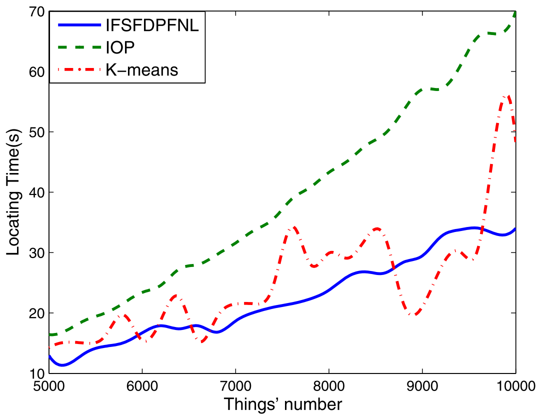

Figure 3. The time efficiency of three algorithms

B. Performance Evaluation

Fig. 3 plots the time efficiency of different fog node location algorithms with 5 fog nodes. In Fig. 3, the blue solid curve denotes the fog node locating time of our proposed algorithm of IFSFDPFNL while the green dotted and red point curves show the fog node locating time of Integrated Optimization Problems(IOP) and K-means cluster algorithm respectively. From Fig. 3, we can find out that our selection algorithm achieves the best time efficiency while the Integrated Optimization Problems(IOP) has the worst time efficiency among these three algorithms. The time efficiency of K-means is far more efficiency than IOP and similar to our IFSFDPFNL algorithm at sometimes. But this does not mean that its time efficiency is similar to our IFSFDPFNL algorithm because some unsuccessful clustering data is excluded in our experiment results. Due to the inappropriate initial cluster center selection, the time efficiency of K-means fluctuates intensely, which is shown clearly in Fig. 3. In fact, Fig. 3 also shows that the time efficiency of K-means is the most fluctuating among these three algorithms.

Fig.4 shows the service performance of different fog node location algorithms. Similar to Fig. 3, the blue solid curve of Fig. 4 denotes the average service delay of our proposed algorithm of IFSFDPFNL while the green dotted curve shows the average service delay of Integrated Optimization Problems(IOP) and the red point line presents the service performance varying with the Things' number increasing with K-means cluster algorithm. It is clear that our selection algorithm achieves the best service performance among three algorithms while the Integrated Optimization Problems(IOP) have the worst service performance among

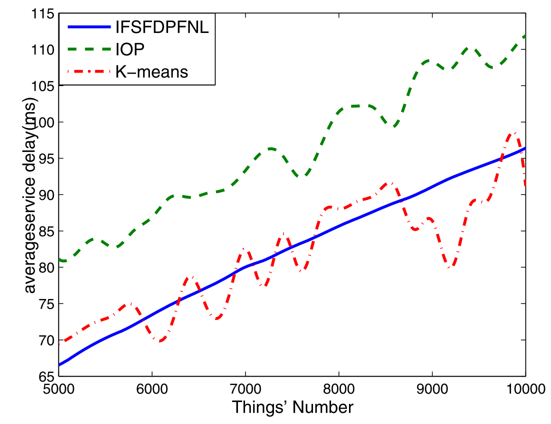

Figure 4. The service performance of three algorithms

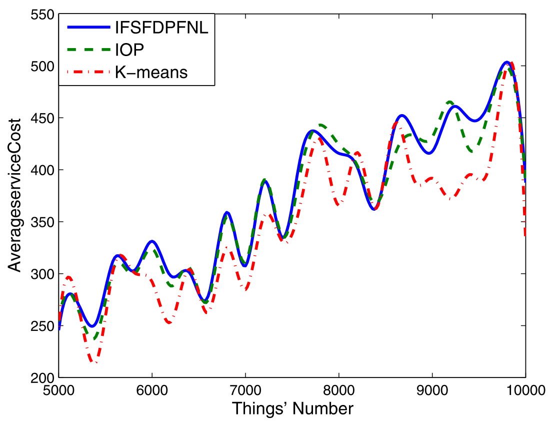

Figure 5. The service cost of three algorithms

these three algorithms in Fig. 4. Different from Fig. 3, the service performance curve of our IFSFDPFNL algorithm is smoother than the other service performance curves while the time efficiency curve of IOP is smoothest among these three time efficiency curves in Fig. 3.

Fig.5 shows the average service cost difference of selected fog nodes for different fog node location algorithms. In Fig. 5, we can find out that although the average service cost of K-means is slightly smaller than our IFSFDPFNL algorithm and IOP wholly, there is no significant service cost difference among these three algorithms. What's more, due to the fluctuating of these three curves, it is unfeasible to claim that K-means clustering algorithm can achieve the less average service cost than the other algorithms.

Comparing with Fig. 3, Fig. 4 and Fig.5, we can find that 1) K-meams clustering may achieve the worst robustness results among these three fog node location algorithms. Both Fig. 3 and Fig. 4 show that the curves of the other algorithms is smoother than the corresponding curve of K-means and Fig. 5 shows that there is no significant fluctuation difference among these three curves; 2) our proposed algorithm achieves the best service performance and best time efficiency while IOP algorithm get the worst service performance and time efficiency among these three algorithms. Obviously, comparing with the other algorithms, our proposed algorithm can achieve the high service performance and time efficiency while the average service cost achieved by it is not significant increasing than the other algorithms. Therefore, it is a effective algorithm to locate the fog nodes for Fog Computing systems.

V. Conclusion

This paper addresses the fog node location of Fog Computing. To this end, we first formulates it as a clustering-based multi-constrained optimization problem. Then an Improved Fast Search and Find of Density Peaks-based fog node location algorithm is proposed to achieve suitable fog node sites and allocate the resources and service cost for each located fog node for our fog node location model. Our proposed fog node location not only integrates the metrics of service quality and service costs, but also guarantees the service quality for all Things. The experiment results show that comparing with the traditional K-means clustering and Integrated Optimization Problems server placement algorithms, our proposed fog node location algorithm can achieve the best service performance and best algorithm time efficiency without too much service cost increasing, it is a effective fog node location algorithm for Fog Computing systems.

Acknowledgment

This paper is partially supported by National Social Science Foundation of China under the agreement of 71673208.

References

[1] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelffle, *Vision and Challenges for Realizing the Internet of Things*. Luxembourg, Germany: European Union, 2010, ISBN 9789279150883.

[2] S. Lien, K. Chen, Y, Lin, *Toward ubiquitous massive accesses in 3GPP machine-to-machine communications*. IEEE Communications Magazine, Vol. 49, No. 4, pp. 66 - 74, 2011.

[3] CISCO, *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. white paper.

[4] F.Bonomi, R.Milito, J.Zhu, S.Addepalli, *Fog Computing and its role in the Internet of Things*. ACM SIGCOMM Workshop on Mobile cloud Computing, Helsinki, Finland, pp. 13 - 16, 2012.

[5] S. Yi, C. Li, Q. Li, *A Survey of Fog Computing: Concepts, Applications and Issues*. Proceedings of the 2015 Workshop on Mobile Big Data, Hangzhou, China, pp. 37 - 42, 2015.

[6] A. Rodriguez and A. Laio, *Clustering by fast search and find of density peaks*. SCIENCE, Vol. 344, No. 6191, pp. 1942 - 1946.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, *Fog Computing and its role in the Internet of Things*. ACM SIGCOMM Workshop on Mobile cloud Computing, Helsinki, Finland, pp. 13 - 16, 2012.

[8] I. Stojmenovic and S. Wen, *The Fog Computing Paradigm: Scenarios and Security Issues*. Proc. of the 2014 Federated Conf. on Computer Science and Information Systems, pp. 1 - 8, 2014.

[9] S. Singh, Y. Chiu, Y. Tsai and J. Yang, *Fog Computing, Mobile Edge Computing, Cloudlets - which one*. Proc. of Computer Symposium (ICS), 2016 International, pp. 731 - 735, 2016.

[10] S. C. Johnson, *Hierarchical clustering schemes*. Psychometrika, Vol. 32, No. 3, pp. 241 - 254, 1967.

[11] A. Bouguettay, Q. Yu, X. Liu, X. Zhou, A. Song, *Efficient agglomerative hierarchical clustering*. Expert Systems with Applications, Vol. 42, No. 5, pp. 2785 - 2797, 2015.

[12] S. M. R. Zadegan, M. Mirzaie, F. Sadoughi, *Ranked k-medoids: A fast and accurate rank-based partitioning algorithm for clustering large datasets*. Knowledge-Based Systems, Vol. 39, pp. 133 - 143, 2013.

[13] J. Xie, S. Jiang, *A Simple and Fast Algorithm for Global K-means Clustering*. 2010 Second International Workshop on Education Technology and Computer Science (ETCS), 2010.

[14] B. Jiang, J. Pei, Y. Tao, X. Lin, *Clustering uncertain data based on probability distribution similarity*. IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 4, pp. 751 - 763, 2013.

[15] R. Zhang, M. Shan, X. Liu, L. Zhang, *A novel fuzzy hybrid quantum artificial immune clustering algorithm based on cloud model*. Engineering Applications of Artificial Intelligence, Vol. 35, pp. 1 - 13, 2014.

[16] H. P. Kriegel, P. Kroger, J. Sander, A. Zimek, *Density-based clustering*. Data Mining and Knowledge Discovery, Vol. 1, No. 3, pp. 231 - 240, 2011.

[17] E. Hartuv and R. Shamir, *A clustering algorithm based on graph connectivity*. Information Processing Letters, Vol. 76, No. 4 - 6, pp. 175 - 181, 2000.

[18] M. Charikar and S. Guha, *Improved combinatorial algorithms for the facility location and k-median problems*. Proceedings of the 40th Annual Symposium on Foundations of Computer Science, pp. 31 - 40, 1999.

[19] Z. I. BOTEV, J. F. GROTOWSKI, D. P. KROESE, *KERNEL DENSITY ESTIMATION VIA DIFFUSION*. The Annals of Statistics, Vol. 38, No. 5, 2916 - 957, 2010

[20] K. Jain, V. V. Vazirani, *Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation*. Journal of the ACM, pp. 274 - 296, 2001.

[21] H. Yin, X. Zhang, T. Zhan, Y. Zhang, G. Min, and D. O. Wu, *NetClust: A Framework for Scalable and Pareto-Optimal Media Server Placement*. IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 15, NO. 8, pp. 2114 - 2124, 2013.

[22] H. Xu and B. Li, *A General and Practical Datacenter Selection Framework for Cloud Services*. 2012 IEEE Fifth International Conference on Cloud Computing, pp. 9 - 16, 2012.