Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 115 – 124

doi:10.3906/elk-1704-206

A strategy for improving NetClust server placement for multicloud environments

Yuting ZHAO[1], Jinhong LIU[1], Qing FANG[2], Llifang XU[2], Changlai DU[3], Xiaoqun YUAN[2,3,*]
[1]School of Arts and Law, Wuhan University of Technology, Wuhan, P.R. China
[2]School of Information Management, Wuhan University, Wuhan, P.R. China
[3]Department of Computer Science, Virginia Tech, Blacksburg, VA, USA

Received: 17.04.2017 • **Accepted/Published Online:** 22.11.2017 • **Final Version:** 26.01.2018

Abstract: In this paper, we propose a fast server placement algorithm to improve the NetClust framework and make it more efficient and flexible. To this end, we introduce hierarchical clustering technologies to the NetClust framework and propose a flexible server placement algorithm, which integrates the agglomerative and divisive clustering technologies to reduce the time complexity and avoid the performance fluctuation affected by the initial node selection. The experiment results show that our server placement algorithm may reduce the time complexity of server selection of NetClust significantly and improve the flexibility and applicability of the NetClust.

Key words: Server placement, resource allocation, cloud-based service platform

1. Introduction

Server placement, which is always implemented to reduce service delay, balance network traffic, improve service reliability, and disperse flash crowds, is an important topic of distributed systems, such as P2P and CDN [1,2]. However, most works formulate it as an NP-hard problem, which suffers from several limitations of passive selection and poor scalability. To deal with these limitations, NetClust, a server placement framework, was proposed by Yin et al. [3]. In NetClust, the network coordination technologies provide the entire topological information for the server placement model with low measurement cost, and the K-means-based cluster technologies may select the suitable servers proactively. Thus, it is a practical server placement solution for large-scale applications such as datacenters or clouds. However, it is not suitable for some emerging technologies and applications, such as the Internet of things [4] and virtual reality. For example, micro and small companies lease the resources of geodistributed clouds to construct their service platforms for some online virtual reality games [5]. Due to the financial limitation of these micro and small companies and the pay-as-you-go model of the cloud, they may change their cloud service resource rental strategies, including the location of clouds and the resource capacities for end users. In this scenario, the server placement strategies not only should select the suitable servers from a large range of options, but they also must have good adaptability for the changes of end users. However, due to the time complexity $O(N^3)$ of the server selection algorithm, NetClust is only carried out offline and does not yield the excepted performance. Thus, it is necessary to develop a server placement algorithm to adapt to this scenario.

This paper aims to deal with this problem by proposing a flexible server placement algorithm (FSPA) for

*Correspondence: yuan20030308@whu.edu.cn

the NetClust framework, which can achieve good adaptability of server placement and is suitable for emerging applications. To this end, we first describe and investigate the NetClust framework, including the advantages and disadvantages. We then propose the FSPA, which is based on hierarchical clustering technologies and integrates the merits of the agglomerative and divisive clustering technologies to reduce the time complexity of server placement algorithm significantly. Finally, we evaluate and verify our server placement algorithm by comparing the server placement performance and service performance of our algorithm to the K-means-based algorithm with a carefully designed experiment. The results show that our server placement algorithm not only reduces the server selection time significantly but also avoids the performance fluctuation affected by the initial node selection. It is a flexible server placement and suitable for emerging applications.

2. Related works

2.1. Server placement

There have been significant works on server placement and mirror replicas for distributed systems since Li et al. [1] proposed their placement algorithm for web proxies on the Internet. They formulated it as a facility location problem and proposed dynamic programming to find the optimal solution with computational complexity of $O(N^3M^2)$. Jamin et al. studied the performances of different placement strategies and found that increasing the number of mirror sites was effective in reducing client download time and server load [6]. Cameron et al. presented an approximate model for dense clients and servers and proposed a simple server allocation and placement algorithm based on high-rate vector quantization theory [7]. Xu et al. [8] focused on replication proxy and data replica placement in a network with the maximum number of proxies to minimize the total data transfer cost. Ahuja and Krunz formulated the server placement problem as a mixed-integer linear programming formulation and an efficient heuristic solution for the SP problem [9]. Yuan et al. [2] focused on the server placement of P2P for live streaming and proposed a server placement strategy for edge servers of CDN-P2P. Zhang and Tatipamula focused on intelligent server placement for social networks and proposed three scalable server placement strategies to select server locations among all the possible locations with less cost of interuser data sharing [10]. Chaisiri et al. formulated virtual machine placement as a stochastic integer programming problem and proposed an optimal virtual machine placement algorithm, which minimized the cost expenditure in each plan for hosting virtual machines in a multicloud provider environment under future demand and price uncertainty [11]. Zhang et al. presented a framework for dynamic service placement based on control- and game-theoretic models and proposed a coordination mechanism to maximize the social welfare of the system [12]. Jin et al. focused on the multidimensional stochastic VM placement problem and proposed a polynomial time algorithm to maximize the minimum utilization ratio of one server's resources [13]. Mijumbi et al. formulated the server placement as a binary integer linear program and proposed a greedy approximation [14]. Zhu et al. studied cloudlet placement and formulated the cloudlet placement as integer linear programming [15].

However, most works formulate it as an NP-hard problem, which suffers from several limitations of passive selection and poor scalability. To deal with these problems, Hao et al. introduced clustering technologies and proposed a cluster-based server placement framework, NetClust [3]. Xiang et al. then proposed an adaptive cloudlet placement, which is also based on K-means clustering and adapted mobile users [16]. However, since the clustering time of K-means clustering is significantly affected by the location of initial centroids, these K-means-based server placement methods cannot yield their results within their expected clustering time. Thus, it is necessary to develop a flexible server placement with less time complexity.

2.2. Clustering technologies

Clustering is a division of samples into groups of similar objects with some metrics such as distance or similarity and it is well investigated for a wide variety of fields including marketing, computers, biology, and libraries. Generally, most clustering algorithms can be classified as partitioning relocation clustering, hierarchical clustering, or other clustering techniques. Hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as a dendrogram, including agglomerative (bottom-up) and divisive (top-down) methods [17]. Partitioning algorithms divide data into several subsets with greedy heuristics in the form of iterative optimization, which consist of probabilistic clustering [18], K-medoids methods [19], and K-means methods [20]. Besides partitioning relocation clustering and hierarchical clustering, some other clustering algorithms have been proposed, such as the clustering algorithms based on fuzzy theory [21], or clustering algorithms based on graph theory [22] and density-based clustering algorithms [23]. The basic idea of fuzzy-based clustering algorithms was the continuous interval [0, 1] label to describe the belonging relationship among objects. The density-based algorithms classed the samples in regions with high density of data space into the same cluster [24,25].

3. NetClust overview

The NetClust framework was proposed to deal with the problem of traditional server placement, such as unscalability and high cost for information gathering. It consists of three components: a measurement engine, a network construction engine, and a placement engine, as shown in Figure 1. The measurement engine applied a lightweight ping method to the destination IP list and got the measurement delay data with little measurement overhead. The network construction engine uses the measurement data as input to construct a GNP-based network coordinate, which is used to predict the pairwise latency between all clients, including landmark selection and network coordinate construction. Based on this constructed network coordinate, the placement engine is used to determine the suitable server placement locations and allocate corresponding network resources for each selected server. To overcome the challenge of the unscalability of existing server placement solutions, NetClust applies a K-means-based clustering algorithm to classify users into different groups, which determines the locations for each server proactively. Then a stable-marriage algorithm is proposed to select the server for each user, which is applied to allocate network resources such as computation, storage, and/or bandwidth.

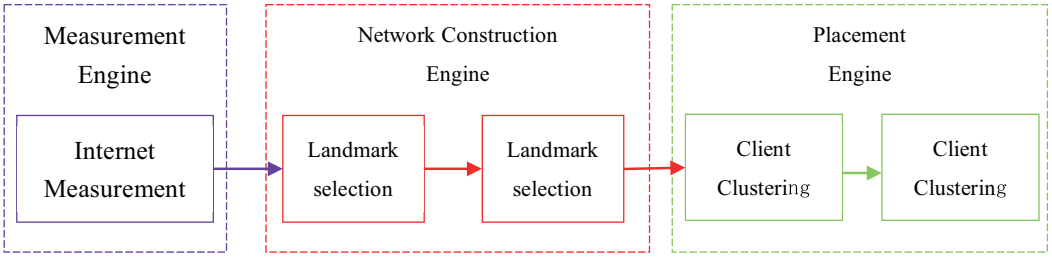

Figure 1. The framework of NCBSP.

Obviously, K-means takes advantage of the idea of partitioning to get the clusters with the iteration, which can find the server deployment location proactively while avoiding the NP-hard problem of traditional server placement solutions. Thus, by applying the technologies of the GNP-based network coordinate, K-means clustering, and stable marriage, NetClust not only can gather the global Internet topology information with low measurement cost but can also select the suitable server deployment sites globally and proactively. It is an

effective solution for a large range of Internet application systems, such as datacenter or cloud constructions, but it is not suitable for some emerging applications such as virtual reality games, data analysis, and processing. By applying a pay-as-you-go business model and VM technologies, the service platforms for these applications can be constructed with multicloud resources, but most of these emerging applications are time-sensitive, which means that their server placement strategies should be flexible enough to meet the financial and application requirements. Unfortunately, although NetClust can achieve the "optimal" server placement strategy for large-scale systems globally and proactively, the time complexity $O(N^3)$ of its server selection algorithm makes this static deployment strategy work only offline. Furthermore, its user-matching and resource allocation strategies are based on the interaction between users and servers, which is also not suitable for the service of online virtual reality games. Thus, it is necessary to improve the server selection algorithm to adapt the randomness and mobility of mobile terminals.

4. Improving the server selection strategy

In this section, we propose our strategy for improving server selection based on NetClust. To this end, we first investigate the server selection algorithm and a strategy improving server selection is designed to overcome the shortcoming of NetClust.

4.1. Server placement algorithm of NetClust

The server placement framework of NetClust consists of the server site selection algorithm and resource allocation algorithm, as shown in Algorithm 1 and Algorithm 2, respectively. In Algorithm 1, the classical K-means clustering algorithm is applied to cluster the points N in network coordinates and achieve the centroid for each cluster, as shown in Steps 2–5. Based on these selected servers, Algorithm 2 is proposed to map each end user to its corresponding server and allocated resources for each selected server based on a stable match algorithm.

Algorithm 1 A cluster-based server placement algorithm

Start;

1: *Input:* Coordinates of N points, the number of clusters K, and the number of candidate server location J;

2: Initialize the cluster centroids μ_1, μ_2, μ_3, $\ldots$, μ_n;

3: Assign each of the N points to a cluster whose centroid is closest to the point;

4: Update the new cluster's centroids μ_i, i = 1, 2, 3,$\ldots$, K;

5: Repeat Steps 3 and 4 until the membership of each cluster does not change;

6: For each cluster i, generate the set of J points that have the lowest placement cost; denote this set by S_i;

7: For each cluster j, select a location from set S_i, which is closest to its centroid μ_i; the selected location is denoted by $\mu \mathrm{p}_i$;

8: *Output:* Pareto optimal server locations $\mu \mathrm{p}_i$, i = 1, 2, 3 $\ldots$, K;

The K-means clustering algorithm is one of the simplest clustering algorithms and can be carried out easily, but its time complexity of *N(NKI)* is affected by the selection of the initial centroids, which means that the computing consumption of server selection varies with the start selection of the centroids significantly. Thus, this algorithm is not suitable for scenarios in which beginners start their businesses by constructing their service platforms with the resources of geodistributed clouds. Algorithm 2 takes the quality of service of users and

Algorithm 2 Client/server matching algorithm

Start;

1: *Input:* server locations x_i, i = 1, 2, 3,..., K, the capacity of each server;

2: Initialize the clients' preference lists and servers' acceptance lists;

3: For each client, assign the client to the server, which is at the head of the preference list of the client;

4: For each server, sort the clients in its request list, in the same order as that of the acceptance list of the server;

5: For each overloaded server, remove the clients that are located at the tail of the request list of the server until its workload is not greater than its capacity;

6: Repeat Step 3 for the clients that are not assigned to any server, Step 4 for the servers with newly assigned clients, and Step 5 for overloaded servers, until each of the clients is assigned to a specific server;

7: *Output:* the client/server matchings listed the request lists of all servers;

the quality of service of servers into account and provides a solution for beginners to construct their service platform with a server placement strategy. However, due to financial restrictions and profit pursuing, these beginners may opt to apply a flexible server placement strategy with low time complexity to adapt the change of users dynamically. The time complexity $O(N^3)$ of Algorithm 2 makes the resource allocation algorithm of NetClust run only once a day or even longer. Furthermore, the server placement strategy is based on the fact that each server's capabilities are fixed and limited after deployment. This is not suitable for our server placement scenarios due to the pay-as-you-go business model of clouds. With this business model, beginners can get enough resources from their selected clouds, so the resource allocations strategy of our server placement should be based on the resource requirement of users with low time complexity.

5. Improving the NetClust server placement algorithm

To overcome the shortcomings of NetClust, we design an algorithm for improving NetClust placement to reduce the computation complexity, as shown in Algorithm 3. Different from K-means clustering, our algorithm implements hierarchical clustering to class datasets, which combines geometric and nongeometric properties, along with a cluster dissimilarity function, into binary cluster trees. Then we integrate the divisive clustering into agglomerative clustering based on merging dissimilarity functions, which obeys a nondecreasing property:

$$d(i,j) \leq d(A \cup C, B), \tag{1}$$

where d is the dissimilarity function and A, B, C are all the cluster names. This property denotes that if two clusters, A and B, agree that they are each other's best match among all the current clusters, then it is impossible for any future grouping of the other clusters to create a better match for either of them. It means that it is possible to build the same tree with a different order of individual nodes, so it is reasonable to achieve subclusters by applying Ward's clustering and the nondecreasing property. To this end, we first divide the whole clustering space V into M subclusters with the constraint of $N >> M >> K$ (where $M = p^d$ and d is the dimensions of the cluster space; $>>$ means far larger, for example $c >> g$ means c is more than 10 times the value of g) and we calculate the centroid for each subcluster. Based on the selected centroids, the clustering applies a Ward's hierarchical clustering algorithm to locate the appropriate servers, which consists of Step 2,

Algorithm 3 Algorithm for approving NetClust server placement

Start;

1: *Input:* V is the space composed by network coordinates; N = the number of end users;
K = the number of the placement site;
W = the number of candidate servers;
M = the number of the initialization subspace;

2: Divide V into M subspace based on $M = p^d$ (d is the dimensions of the cluster space) uniformly, calculate the centroid coordinate Cen_i of each subspace, and let it be as the centroids of corresponding subspaces;

3: For each subcluster pair, calculate its distance by applying Eq. (2) and construct the distance matrix of subclusters;

4: Find out the nearest subcluster pairs from the distance matrix by applying the nondecreasing property, merge each subcluster pair into one new subcluster, and calculate the centroid of the new subcluster with $\frac{n_i x_i + n_j x_j}{n_i + n_j}$.

5: Repeat Step 3 and Step 4 until our hierarchical clustering tree is constructed;

6: Determine the clustering number K and the centroid of each cluster based on the hierarchical clustering tree according to the deployment cost and (or) the user experience metrics;

7: Determine the K suitable servers from W candidate servers based on the deployment cost and (or) the user experience metrics;

8: Allocate the service resources among these servers based on their service sample number and achieve the logical server placement strategy;

9: Map the coordinate values of these logical servers to the IP, respectively. Then parse each IP to its corresponding physical deployment site and get the server placement strategy for distributed cloud computing.

Step 3, and Step 4 of Algorithm 3. In our algorithm, the distance between each subcluster pair can be achieved based on Ward's linkage, which is described as:

$$d(i,j) = \sqrt{\frac{2 * n\,i\,n\,j}{n\,i + n\,j}\,\|\overline{x\,i} - \overline{x\,j}\|}, \tag{2}$$

where $\|\|$ is Euclidean distance, $\overline{x\,i}$ and $\overline{x\,j}$ are the centroids of clusters i and j respectively, and n_i and n_j are the number of elements in clusters i and j.

Based on these distances, we construct the distance matrix of subclusters and apply the nondecreasing property to choose the subcluster pairs with minimal distances from the constructed distance matrix. Then each selected subcluster pair is merged into one new subcluster in parallel. After that, the new distance of each subcluster pair is calculated and the distance matrix with newly merged subclusters is updated according to the distance calculating metric of Eq. (2). We repeat these steps until we get our default value, such as the clustering number or the minimum service time. Then we determine the location of the selected servers. After server locations are determined, all users are assigned to the nearest selected servers and then the resource requirement for each server is calculated based on its service users, as shown in Algorithm 3.

Our algorithm takes the client distribution density, server deployment cost, and user-experienced latency requirements into account. By calculating the cancroid coordinate value of each set, the algorithm guarantees the network traffic localization. Furthermore, our solution is flexible to optimize either deployment cost, service performance, or both by choosing different metrics, as shown in Step 9 in Algorithm 3. Obviously, the cluster-based server placement algorithm achieves the logical server placement strategy, but the servers need to be deployed in physical sites. Thus, we should transform our logical placement locations into physical deployment locations, which is achieved based on the mapping relationship between the delay and coordinate value, as shown in Step 11 in Algorithm 3. As for the time complexity of Algorithm 3, Step 2 and Step 6 have the complexity of $O(N)$. Steps 7, 8, and 9 have the time complexity of $O(KW)$, $O(KN)$, and $O(N+W)$, respectively. Step 3, Step 4, and Step 5 apply a parallel subcluster merging algorithm to construct our hierarchical clustering tree through merging each nearest subcluster pair to one new subcluster, which has time complexity similar to Ward's hierarchical cluster of $O(N)$.

6. Experimental design and performance analysis

To evaluate the performance of the proposed algorithm under similar network environments, we designed our input and simulation environment to investigate the algorithm efficiency and service performance differences of our server placement strategy, the adaptive cloudlet placement method (ACPM) [16], and the NetClust server placement strategy (NetClust) [3].

Figure 2 plots the time efficiency of different server selection algorithms. Figure 2a compares the time efficiency of different server selection algorithms with fixed server number of 5 and Figure 2b shows the server selection time of different algorithms with 10,000 end users. In Figure 2, the upper subgraph plots the server selection time of our proposed algorithm, FSPA, while the lower subgraph denotes the server selection time of NetClust. From Figure 2, we see that our selection algorithm is much better than NetClust when the inputs are the same. When the server number or the end user number is fixed, the server selection time of FSPA is much lower than that of NetClust. When the server number is fixed to 5, the maximum selection time of FSPA is 0.22 s, while the minimal selection time of NetClust is 8.07 s with the end user number increasing from 5000 to 25,000, as shown as in Figure 2. In Figure 2, this value of FSPA is between 0.05 and 0.06, while this value increases from 9.1 to 71.8 for NetClust when the end user number is 10,000 and the servers are changed from 1 to 20. Figure 2 also shows that the selection time of NetClust is positively correlated with the end user number and the server number, while this value of our proposed algorithm is positively correlated with the end user number. With the server number fixed to 5, the selection time of NetClust is increased from 8.07 s to 149.27 s when end user number is increased from 5000 to 25,000. This value increases from 9.18 s to 71.76 s when the servers increase from 1 to 20 while the end user number is fixed to 10,000. Figure 3 denotes the service performance of different selection algorithms. In Figure 3, the solid and dashed curves represent the service performances of NetClust and FSPA, respectively, where these performances are obtained when the end user number is 5 in Figure 3a and the server number is fixed to 5000 in Figure 3b. Interestingly, we observed that the service delay of both selection algorithms decreases with the server number increasing, following the distribution of the negative exponent, and the convergence point can be achieved at about 10, as shown in Figure 3b. Figure 3 also shows there is no significant difference between these two curves, which means these two server selection algorithms have the same service performance. Figure 3a shows that both these two curves are fluctuating between 72 ms and 78 ms.

Combining Figures 2 and 3, it is clear that our algorithm can significantly reduce the computation consumption without too much service performance degradation. When the server number is fixed to 5 while

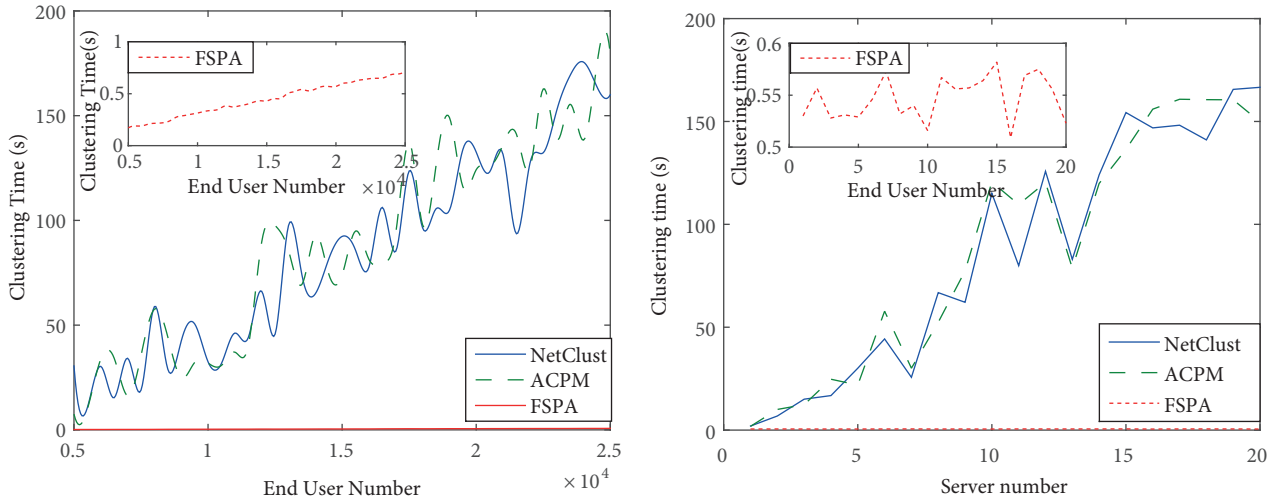

Figure 2. Server selection time of different algorithms: a) server selection time with fixed server number; b) server selection time with fixed end user number.

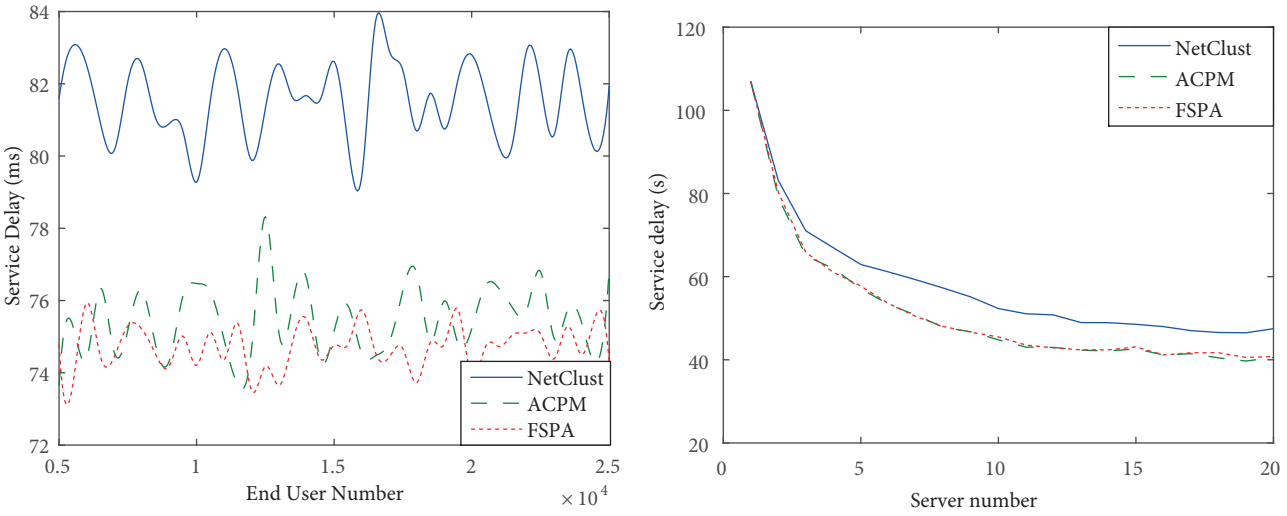

Figure 3. Service performance of different algorithms: a) service delay with fixed server number; b) service delay with fixed end user number.

the end user number increases from 5000 to 25,000, the selection time of NetClust increases by about 140 s (from 8.07 s to 149.27 s), while this value for FSPA is just 0.144 s (from 0.0654 s to 0.209 s). When the end user number is fixed to 5000 while the servers increase from 1 to 20, the selection time of NetClust increases by about 62.52 s (from 9.18 s to 71.7 s), while this value of FSPA is just 0.0066 s (from 0.0516 s to 0.0582 s). On the other hand, whenever server number and end user number are fixed, the service performances of NetClust and FSPA have no obvious differences. When server number is fixed, the average service times of both NetClust and FSPA fluctuate between 72 ms and 78 ms and the maximum difference between the service time of these two algorithms is 3.2664 ms. When end user number is 5000, the value is 2.5283 ms even if the server number increases from 1 to 20.

7. Conclusion

As a promised solution to construct large-scale service systems, NetClust presents a server placement framework to select suitable server deployment sites and allocates resources for each server. However, due to the high time complexity of its server placement algorithm, this framework is not suitable for scenarios in which micro and small beginners construct their service platforms. In this paper, we deal with this problem by proposing a flexible server placement algorithm. Our algorithm integrates agglomerative clustering and divisive clustering technologies to select the server deployment sites and allocate resources for each selected server quickly and accurately. The experimental results show that our algorithm can significantly reduce the computation consumption without too much service performance degradation. Furthermore, our server placement algorithm can avoid the inestimable problem of server selection computation consumption, which NetClust faces during the period of its server selection.

Acknowledgment

This paper is partially supported by the National Social Science Foundation of China under the agreement of 09CTQ024. It is also supported by the Top Discipline of Library, Information and Data Science in Ministry of Education of the Peoples' Republic of China.

References

[1] Li B, Golin MJ, Italiano GF, Deng X, Sohraby K. On the optimal placement of web proxies in the Internet. Int Fed Info Proc 1998; 8: 485-495.

[2] Yuan X, Yin H, Min G, Liu X, Hui W, Zhu G. A suitable server placement for peer-to-peer live streaming. J Supercomput 2013; 64: 1092-1107.

[3] Yin H, Zhang X, Zhan T, Zhang Y, Min G, Wu DO. NetClust: A framework for scalable and pareto-optimal media server placement. IEEE T Multimedia 2013; 8: 2114-2124.

[4] Lee I, Lee K. The Internet of things (IoT): applications, investments, and challenges for enterprises. Bus Horizons 2015; 58: 431-440.

[5] Gupta P, Seetharaman A, Raj JR. The usage and adoption of cloud computing by small and medium businesses. Int J Inform Manage 2013; 5: 861-874.

[6] Cronin E, Jamin S, Jin C, Kurc AR, Raz D, Shavitt Y. Constrained mirror placement on the Internet. In: IEEE 2001 International Conference on Computer Communication; 22–26 April 2001; Anchorage, AK, USA. New York, NY, USA: IEEE. pp. 1369-1382.

[7] Cameron CW, Low SH, Wei DX. High-density model for server allocation and placement. Perf E R SI 2002; 30: 152-159.

[8] Xu J, Li B, Lee DL. Placement problems for transparent data replication proxy services. IEEE J Sel Area Comm 2002; 20: 1383-1398.

[9] Ahuja S, Krunz M. Algorithms for server placement in multiple description-based media streaming. IEEE T Multimedia 2008; 7: 1382-1392.

[10] Zhang Y, Tatipamula M. The freshman handbook: a hint for the server placement of social networks. In: IEEE 2012 International Conference on Parallel and Distributed Systems; 17–19 December 2012; Singapore. New York, NY, USA: IEEE. pp. 173-174.

[11] Chaisiri S, Lee BS, Niyato D. Optimal virtual machine placement across multiple cloud providers. In: IEEE 2009 Asia-Pacific Services Computing Conference; 7–11 December 2009; Singapore. New York, NY, USA: IEEE. pp. 103-110.

[12] Zhang Q, Zhu Q, Zhani MF. Dynamic service placement in geographically distributed clouds. IEEE J Sel Area Comm 2013; 12: 762-772.

[13] Jin H, Pan D, Jing X. Efficient VM placement with multiple deterministic and stochastic resources in data centers. In: IEEE 2012 Global Communication Conference; 3–7 December 2012; Anaheim, CA, USA. New York, NY, USA: IEEE. pp. 2505-2510.

[14] Mijumbi R, Serrat J, Gorricho JL, Rubio-Loyola J, Davy S. Server placement and assignment in virtualized radio access networks. In: IEEE 2015 International Conference on Network and Service Management; 9–13 November 2015; Barcelona, Spain. New York, NY, USA: IEEE. pp. 398-401.

[15] Xu Z, Liang W, Xu W, Jia M, Guo S. Efficient algorithms for capacitated cloudlet placements. IEEE T Parall Distr 2016; 27: 2866-2880.

[16] Xiang H, Xu X, Zheng H, Li S, Wu T, Dou W, Yu S. An adaptive cloudlet placement method for mobile applications over GPS big data. In: IEEE 2016 Global Communication Conference; 4–8 December 2016; Washington, DC, USA. New York, NY, USA: IEEE. pp. 1-6.

[17] Bouguettaya A, Yu Q, Liu X, Zhou X, Song A. Efficient agglomerative hierarchical clustering. Expert Syst Appl 2015; 42: 2785-2797.

[18] Bin J, Jian P, Yufei T, Xuemin L. Clustering uncertain data based on probability distribution similarity. IEEE T Knowl Data En 2013; 4: 751-763.

[19] Zadegan SMR, Mirzaie M, Sadoughi F. Ranked k-medoids: a fast and accurate rank-based partitioning algorithm for clustering large datasets. Knowl-Based Syst 2013; 39: 133-143.

[20] Xie J, Jiang S. A simple and fast algorithm for global K-means clustering. In: 2010 2nd International Workshop on Education Technology and Computer Science; 6–7 March 2010; Wuhan, China. New York, NY, USA: IEEE. pp. 36-40.

[21] Aliahmadipour L, Torra V, Eslami E. On Hesitant Fuzzy Clustering and Clustering of Hesitant Fuzzy Data. Fuzzy Sets. New York, NY, USA: Springer, 2017.

[22] Hartuv E, Shamir R. A clustering algorithm based on graph connectivity. Inform Process Lett 2000; 76: 175-181.

[23] Kriegel H, Kröger P, Sander J, Zimek A. Density-based clustering. WIRES Data Min Knowl 2011; 1: 231-240.

[24] Sharan R, Shamir R. Center click: a clustering algorithm with applications to gene expression analysis. In: International Conference on Intelligent Systems for Molecular Biology; 19–23 August 1999; San Diego, CA, USA. Cambridge, MA, USA: MIT Press. pp. 307-316.

[25] Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput Surv 1999; 31: 264-323.