Research Article

Service Composition Optimization Method Based on Parallel Particle Swarm Algorithm on Spark

Xing Guo, Shanshan Chen, Yiwen Zhang, and Wei Li

School of Computer Science and Technology, Anhui University, Anhui Province, China

Correspondence should be addressed to Yiwen Zhang; zywahu@qq.com

Received 2 September 2017; Accepted 6 November 2017; Published 19 December 2017

Academic Editor: Lianying Qi

Copyright © 2017 Xing Guo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Web service composition is one of the core technologies of realizing service-oriented computing. Web service composition satisfies the requirements of users to form new value-added services by composing existing services. As Cloud Computing develops, the emergence of Web services with different quality yet similar functionality has brought new challenges to service composition optimization problem. How to solve large-scale service composition in the Cloud Computing environment has become an urgent problem. To tackle this issue, this paper proposes a parallel optimization approach based on Spark distributed environment. Firstly, the parallel covering algorithm is used to cluster the Web services. Next, the multiple clustering centers obtained are used as the starting point of the particles to improve the diversity of the initial population. Then, according to the parallel data coding rules of resilient distributed dataset (RDD), the large-scale combination service is generated with the proposed algorithm named Spark Particle Swarm Optimization Algorithm (SPSO). Finally, the usage of particle elite selection strategy removes the inert particles to optimize the performance of the combination of service selection. This paper adopts real data set WS-Dream to prove the validity of the proposed method with a large number of experimental results.

1. Introduction

As big data develops, more and more users publish their resources in the form of Web services to promote the use of service. As a distributed computing model, which is self-contained, modular, and loosely coupled, Web services are characterized by being similar in functional attribute rather than nonfunctional attribute. Quality of Service (QoS) represents nonfunctional attribute in Web services, such as Availability, Price, and Reputation. With an ever-larger number of cloud services, selecting the optimal cloud service composition solution which satisfies user's requirement has become a matter of great interest in the field of service composition [1]. The existing service selection methods which obtain best service composition solution are based on their QoS information. In [2], the service composition model is studied under Cloud Computing. In [3], a service composition for service level agreement (SLA) is proposed, in which a vague semantic preference is used as per user preference to select optimal services with the help of a new method. Work [4] presents a service selection method based on fuzzy logic, in which intelligent cloud storage is used

and a lot of theoretical proof is given. In [5], a variety of hybrid services in heterogeneous clouds are used to perform service discovery and combination by Skyline operations. Work [6] presents a new approach, that is, to find a reliable dynamic service composition in two phases. Work [7] uses the weighted principal component analysis method to select the multimedia service.

These methods explore potential problems in the service composition and put forward solutions accordingly, though there are problems that need to be addressed like inefficiency in large-scale service composition selection when conducted in a Cloud Computing environment. Therefore, we propose a novel large-scale service selection method based on distributed computing environment, Spark [8], using the parallel particle swarm method to solve the service composition problem.

The contributions of this paper are summarized as follows.

- (1) Based on the service selection characteristics of big service, we propose SPSO service selection method. This method uses the combined potentials of Spark,

covering algorithm and particle swarm algorithm. Spark is used for parallelization, covering algorithm for reducing the initial search space, and particle swarm algorithm for optimization of service selection. These three techniques are combined to solve the problem of large-scale service selection.

- (2) In the service selection, SPSO is mainly divided into three phases; in the first an efficient parallel algorithm is proposed to cluster the Web candidate service set, combined with the covering algorithm in neural network, to reduce the search space of candidate service set. Then, based on RDD parallel computing strategy, we realized the storage and parallel search for large-scale composite service. Finally, we use the new elite selection strategy to optimize the service selection capacity of population particles.
- (3) In order to reflect the effectiveness of the proposed method, we have implemented the parallelization of the contrast algorithm. A large number of simulation experiments have been conducted on real data set WS-Dream to verify the feasibility of solving large-scale service composition.

The rest of this paper is organized as follows: Section 2 introduces related work; Section 3 presents Web service composition model; Section 4 introduces the improved particle swarm method; Section 5 verifies the effectiveness of our approaches through simulating experiments. Finally, summary and future work are presented.

2. Related Work

Service composition, mainly used in service-oriented architecture (SOA) and grid manufacturing, is a typical NP optimization problem. In literature previously mentioned, many scholars put forward various solutions to select appropriate services composition, such as improving the efficiency and quality of the service composition and reducing the size of the candidate service set. There are three ways to opt for a service composition: by local search, global search, or intelligent optimization algorithm. The local optimal method is to choose the best service in each candidate service set and then combine them [9]. However, the combination service may not be optimal. As for global search method, literature [10, 11] uses integer coding to solve the problem of service composition search, which is of great efficiency when the problem is small. In the cloud environment, however, the effectiveness of global search is weakened due to its poor scalability as the service composition business flow model becomes complex. To tackle scalability issue, the swarm intelligent optimization algorithm with high efficiency and fast speed is widely used in the service composition problem field. In literature [1], the bee colony algorithm is applied. The introduction of time enhancement function establishes a trusted service composition model, thus transforming service composition problem into a nonlinear integer coding problem. In [12], the correlation-aware service model is given, and the genetic algorithm is used to find the service composition in cloud manufacturing. In [13], a new gene coding as

well as the differential evolution algorithm is used to find the service composition, which improves the convergence of the algorithm. Work [14] combines the advantages of FOA algorithm and genetic algorithm to find the combined service. In [15], the particle swarm algorithm is applied to the service composition in the cloud manufacturing.

However, in the Cloud Computing environment, previous methods of service selection may not be effective. Many scholars have proposed parallel service selection method to deal with large-scale service composition. In [16], from the perspective of Pareto optimality, partial selection strategy is used to precede QoS awareness service composition selection. The Pareto set model proposed in this paper has been theoretically proved effective first and then evaluated by a large number of experiments. In [17], a new large-scale service composition selection method, that is, the Hadoop distributed computing platform, is introduced. The discrete particle swarm optimization algorithm is combined with the Hadoop platform to select the service composition. In [18], parallel K -means algorithm and particle swarm algorithm are used to select the service composition on the Hadoop platform. Despite the full use of its computational advantage, Hadoop parallel computing platform features inefficiency in reading data. As a memory-based cluster computing platform, Spark, widely used in distributed data processing, has the characteristic of Hadoop and optimizes it to abstract the distributed data into a flexible distributed data set RDD [19].

3. Model of Service Composition

These three important components of cloud service calculation, namely, Cloud Service Providers (CSPs), Cloud Broker (CB), and Cloud Consumers, serve different purposes. The process of combining service is mainly divided into the following steps:

- (1) Cloud Consumer publishes the requirements to CB, and CB receives the preference weight of QoS attribute $W = (W_1, W_2, \dots, W_i, \dots, W_r)$, where r represents the number of QoS properties.
- (2) CB divided task into multiple subtasks $T = (T_1, T_2, \dots, T_i, \dots, T_n)$.
- (3) To fulfill subtask, CB selects a service from candidate services $WS^i = (WS_1^i, WS_2^i, \dots, WS_j^i, \dots, WS_m^i)$, in which $j \in \{1, 2, \dots, m\}$. The services which address the same atomic task are classified as a set of candidate service. Services selected from each candidate service set constitute a composite service $CS = (WS^1, WS^2, \dots, WS^i, \dots, WS^n)$. The nonfunctional attribute of Web service can be represented as $Q_{WS_j^i}^k = \{q_1, q_2, \dots, q_k\}$.
- (4) The service quality $Q_{cs}^i = \{q_1, q_2, \dots, q_k\}$ of the selected service is calculated based on workflow model.
- (5) Calculate the fitness of the composite service. Select the optimal service and give feedback to Cloud Consumer.

4. SPSO

The standard particle swarm algorithm, proposed by Eberhart and Kennedy in 1995, is a kind of evolutionary computation which originated from the study of bird predation [20]. In the process of searching, we start from a set of random solution, finding and updating the optimum solution in each iteration. In the search space, each particle represents a solution. The population migrates in parallel when moving. It is, therefore, viable to solve large-scale service composition problems by the parallelization of particle in distributed computing environment. The specific method is shown as in Figure 1.

When the population position is initialized, parallel covering algorithm is used to obtain multiple clustering centers as the starting point. Then, subpopulation migration is completed in Spark distributed computing environment; inert particles are removed through particle elite selection strategy. In the end, relatively optimal service composition is selected.

4.1. Coding Scheme. Firstly, the parallel particle swarm in the Spark cluster is encoded. The population in the RDD is encoded as shown in Figure 2, where the population is $P = (P_1, P_2, \dots, P_m)$ and m is the number of particles. Each particle has recorded its information including current position, velocity, and historical optimum position. The task is divided into subtask $T = (T_1, T_2, \dots, T_i, \dots, T_n)$, where n represents the number of divided abstract subtasks and also indicates that the search space of the particles has n dimensions. The specific coding mode is shown in Figure 2.

4.2. Initialization. The initial location of the population particles is a critical factor when it comes to population diversity. To randomly initialize the position of the particle, the use of particle swarm algorithm is highly apt to generate search inefficiency. This paper, therefore, uses the parallel covering algorithm [21] to cluster multiple candidate services based on their QoS properties and ensure that the population particles are randomly distributed in these initial starting points.

As a kind of clustering algorithm, covering algorithm, proposed by L. Zhang and B. Zhang on the basis of the neural network model, is developed from the idea that separates samples with less similarities for a set of fields. The QoS properties set of the Web service is $Q_{WS_j}^k = \{q_1, q_2, \dots, q_k\}$, where k is the number of properties. Each candidate service set is seen as a k -dimensional point set. The main steps are as follows:

- (1) The center of gravity of all points unclustered is calculated by Euclidean distances. Select the nearest point from the center of gravity as the initial center.
- (2) Calculate the distance between the remaining points and the center. The average distance is applied as a radius, and distances which are less than the radius of the service clustered as a cover.
- (3) Calculate the distance between all unclustered points with the center. Select the farthest point as the new

center and then recalculate the distance and take the average distance as the radius.

- (4) For the remaining unclustered points, the points whose distance with the center distance is less than the radius are screened as a new cover.
- (5) If there is any unclustered point left, repeat steps (3)-(4).

As shown in Figure 3, $S^i = (S_1^i, S_2^i, S_3^i, S_4^i, S_5^i)$ is the circular coverage after being clustered, where the red dot is the cluster center. The size of each circular coverage is proportional to the number of services which it contains. Clustering centers $C^i = (C_1^i, C_2^i, C_3^i, C_4^i, C_5^i)$ of candidate service set can be obtained by applying covering algorithm.

Based on the Spark distributed computing environment, this paper uses the parallel covering algorithm to cluster the services in each candidate service set (Algorithm 1).

The candidate service to be selected is seen as an RDD. After covering the clustering of the Web service $WS^i = (WS_1^i, WS_2^i, \dots, WS_j^i, \dots, WS_m^i)$ in each candidate service set, there will be n -dimensional coverage $S^i = (S_1^i, S_2^i, \dots, S_j^i, \dots, S_k^i)$, where k is the number of spherical coverage and n is the number of QoS properties. Each n -dimensional covering will have multiple clustering centers, and clustering centers $C^i = (C_1^i, C_2^i, \dots, C_j^i, \dots, C_k^i)$. After clustering analysis, we can obtain multidimensional circular covering $S = (S^1, S^2, \dots, S^i, \dots, S^n)$ and clustering center.

4.3. Fitness Evaluation. The fitness value of each composite service has to be calculated. In the selection of Web services, the overall QoS of the composite service has a great impact on the service evaluation. The fitness is used as the evaluation of the Web combination service. The smaller the fitness, the better. The fitness function applied in this paper is

$$\text{fitness} = \sum_{r=1}^n w_r * Q_{cs}^r. \quad (1)$$

w_r represents the preference of Cloud Consumer for the r th QoS attribute of the composite service; n is the total number of service QoS attributes; Q_{cs}^r represents the r th QoS attribute value of the composite service.

4.4. Parallel Particle Migration. After initialization, the cluster particles are encapsulated into an RDD. Suppose that there are N particles in the population, and the search space is D dimension which indicates that there are D subtasks divided; the position of the i th ($i < N$) particles can be expressed as $X_i^t = (X_{i1}^t, X_{i2}^t, \dots, X_{ij}^t, \dots, X_{iD}^t)$, in the t th generation, and each dimension of the position represents the selected Web service. $V_i^t = (V_{i1}^t, V_{i2}^t, \dots, V_{ij}^t, \dots, V_{iD}^t)$ is the flying speed of the particle. In the t th generation, the individual recorded optimal position of the i th particle search is $P_i^t = (P_{i1}, P_{i2}, \dots, P_{ij}, P_{iD})$; the current optimal position of population is $P_g = (P_{g1}, P_{g2}, \dots, P_{gj}, \dots, P_{gD})$. In the $(t+1)$ th

FIGURE 1: The flow chart of SPSO algorithm.

```

Input: WS
Output:  $S = (S^1, S^2, \dots, S^i, \dots, S^n)$ 
(1)  $i = j = 1$ 
(2) For  $i < n$  do
(3)   While exist unclustered points
(4)     If (without clustered center)
(5)       Compute distance and generate center
(6)       Compute Euclidean distance
(7)       Generate  $S_1^i$ 
(8)     End if
(9)     If (exist clustered center)
(10)       $j = j + 1$ 
(11)      Compute distance
(12)      Take remote point as new center
(13)      Compute Euclidean distance
(14)      Generate  $S_j^i$ 
(15)    End if
(16)  End while
(17) End for

```

ALGORITHM 1: Particle initialization based on parallel covering algorithm.

FIGURE 2: Encoding of composition service.

generation, the update formula for the j th dimension and position of particle is shown as follows:

$$V_{ij}^{t+1} = W * V_{ij}^t + c_1 * r_1 * (P_{ij}^t - X_{ij}^t) + c_2 * r_2 * (P_{gj}^t - X_{ij}^t) \quad (2)$$

$$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^t, \quad (3)$$

where c_1 and c_2 are learning factors. r_1 and r_2 are random variables evenly distributed over the interval $[0, 1]$. W is the inertia weight which measures the effect of the velocity of the migration on the next movement. The formula is

$$W = W_{\min} + (W_{\max} - W_{\min}) * \frac{t}{T}, \quad (4)$$

where $W_{\max}$ is the maximum inertia weight value and $W_{\min}$ is the minimum inertia weight value. t is the current evolutionary generation and T is the total evolutionary generation. Generally, take $W_{\max} = 0.9$, $W_{\min} = 0.3$.

Particle population migration can be seen as the transformation of RDD, and the operation of selecting global

FIGURE 3: Covering clustering in candidate service set.

optimal particle as the action during each iteration. The fitness of the best particle is broadcasted to population, and the population particles migrate to the next subpopulation until the migration ends.

4.5. Elite Selection Strategy. When using the Spark cluster to search the service composition, the diversity of the particles has a great influence on finding the optimal particle. After several searches, according to the search strategy, if the particle activity range is small, the optimization effect has limited effect on the whole population. This paper introduces the mechanism of particle elite selection, increasing the diversity of the population by removing the inert particles.

The specific idea of the mechanism is that when encoding particles, add parameters of the historical optimal position without changing the number of each particle. If the particle is not the optimal, and the historical optimal position of the

Input: $S = (S^1, S^2, \dots, S^i, \dots, S^n)$, N, m, T, L
Output: P_g
(1) Initiate particle swarm and compute fitness
(2) $t = 0$
(3) **For** $i < N$ **do**
(4) Update position and speed
(5) Compute fitness
(6) Update history information
(7) **If** ($P_i < L$)
(8) Elite selection strategy
(9) **End if**
(10) **End for**
(11) $T = t + 1$
(12) Generate the best particle

ALGORITHM 2: SPSO.

particle is not updated to a certain threshold value L , and the particle migration range is small, then the particle can be considered an inert particle. The historical best solution during the multiple migration process remains the same.

4.6. Algorithm Procedure. Based on the above analysis and design, the service composition optimization algorithm can be demonstrated as in Algorithm 2.

In Algorithm 2, $S = (S^1, S^2, \dots, S^i, \dots, S^n)$ is the covering of multiple candidate service set including multiple clustering centers in each candidate service set. The population selects the random initial starting point according to these clustering centers. When the particles are reinitialized, the particle initialization is performed through the particle elite selection strategy.

5. Experiments

In this paper, we evaluated the efficiency of the improved particle swarm algorithm by comparing with the PSO algorithm [15]. The experimental value is the average of the 20 times of experiment.

5.1. Parameter Setting of Algorithm. Our experiments are initiated by a real-world service quality set WS-Dream [22], where more than 30 million Web services data as well as their quality values are collected. We chose the two properties as the QoS evaluation index, namely, response time (RT) and throughput (T). QoS preference weight is (0.5, 0.5).

Experimental Environment. Spark cluster consists of 9 nodes. We adopted Spark 1.4. The number of cores that can be used in the cluster is 72.

5.2. Effects of Spark Parameter. This experiment was carried out to test the effect of parallelism and core on algorithm in Spark cluster.

Firstly, by setting different parallelism, the effect of parallelism on the time consumption of two algorithms is investigated. In this paper, five subtasks under the service selection scenarios were taken as an example. Each subtask

FIGURE 4: The runtime for the different parallelism.

corresponding to the candidate service set entails 100000 services. The number of particles is 5000, and that of iterations is 500, and the total number of cores is 30. The results are shown in Figure 4.

In Figure 4, as the degree of parallelism increases, the time consumption of the two algorithms for service selection increases. When the degree of parallelism is set between 10 and 30, the time consumption is significantly less than that of between 40 and 60. The reason is that the parallelization of population particles is related to the idle cluster resources. When the degree of parallelism is 10–30, the population of particles is divided into 10–30 subpopulations, and the particles are migrated in parallel. When the degree of parallelism is 40–60, the particle migration can not be carried out in parallel because of the lack of available auditing resources, resulting in more time consumption.

Then, we examine the effect of the total number of cores on the consumption of the two algorithms. The candidate service has 100000 services, the number of iterations is 500, the degree of parallelism is 30, the number of particles is 5000, and the different number of cores is set. The results are shown in Figure 5.

As shown in Figure 5, it can be seen that, with the increase in the number of cores, the time for service selection is gradually reduced and stabilized. In the 10–30 stages of core, as the number of cores increases, the number of subpopulations that can migrate simultaneously increases while the time consumed decreases. When the core is 30–60, the cluster resources are sufficient; the time consumption tends to be stable.

5.3. Effectiveness. We tested the effect of the number of particles on the fitness value. This set of experiments tested 20 subtasks, in which candidate services are all set as 100000. In the Spark cluster environment, the parallelism is set to 20, the number of iterations is 500, and the total number of cores is 20. The results are shown in Figure 6.

FIGURE 5: The runtime for the different number of cores.

FIGURE 7: The fitness values for different number of iterations.

FIGURE 6: The fitness value for the different number of particles.

FIGURE 8: The runtime of different number of particles.

From Figure 6, with the number of particles increasing, the selected combination of service improves while the fitness value of the overall trend was declining.

This experiment is investigated to evaluate the effect of the iteration number on the fitness value. The group of experiments tested 20 subtasks. The number of candidate services is set to 200000, the number of parallelism is 30, the number of particles is 5000, and the total number of cores is 30. The experimental results are shown in Figure 7.

It can be seen that, from Figure 7, SPSO is superior to PSO in terms of the ability of finding optimal solutions. The average fitness generally declined with the increase in the number of iterations.

5.4. Efficiency. In this experiment, we examine the efficiency of the SPSO algorithm. We tested the effect of the number of particles on the time consumption of the SPSO algorithm

under different subtask. The number of different particles is selected in the experiment, and the number of candidate service sets is 100,000. Set the parallel number to 20 and the iterations number to 500. The experimental results are shown as in Figure 8.

Figure 8 shows that, as the number of particles increases, the more time it takes to complete the parallelization of the particles and the more time it takes for the service selection. Meanwhile, for the same number of population particles, time consumption increases as the number of subtasks increases.

6. Conclusion

In this paper, we proposed an improved particle swarm optimization algorithm in Spark cluster to solve the problem

of Web service composition optimization in a big data environment. In the simulation experiment, we have studied the effects of Spark parameter and effectiveness of the improved algorithm. Experimental results show that the improved particle swarm optimization algorithm proposed in this paper outperforms other algorithms in Web service composition. For future works, we will examine the impact of service reliability on service selection issue in the context of large-scale service composition optimization.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Key Technology R&D Program (no. 2015BAK24B01), the General Research for Humanities and Social Sciences Project of Chinese Ministry of Education (no. 15YJAZH112), and the Educational Commission of Anhui Province of China (no. KJ2016A038).

References

- [1] Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue, "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," *Applied Intelligence*, vol. 42, no. 4, pp. 661–678, 2015.
- [2] Y. Zhao, Y. Li, I. Raicu et al., "A Service Framework for Scientific Workflow Management in the Cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 930–944, 2015.
- [3] X. Zhao, L. Shen, X. Peng, and W. Zhao, "Toward SLA-constrained service composition: An approach based on a fuzzy linguistic preference model and an evolutionary algorithm," *Information Sciences*, vol. 316, pp. 370–396, 2015.
- [4] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart Cloud Storage Service Selection Based on Fuzzy Logic, Theory of Evidence and Game Theory," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2348–2362, 2016.
- [5] F. Zhang, K. Hwang, S. U. Khan, and Q. M. Malluhi, "Skyline Discovery and Composition of Multi-Cloud Mashup Services," *IEEE Transactions on Services Computing*, vol. 9, no. 1, pp. 72–83, 2016.
- [6] Z.-Z. Liu, D.-H. Chu, Z.-P. Jia, J.-Q. Shen, and L. Wang, "Two-stage approach for reliable dynamic Web service composition," *Knowledge-Based Systems*, vol. 97, pp. 123–143, 2016.
- [7] L. Qi, W. Dou, and J. Chen, "Weighted principal component analysis-based service selection method for multimedia services in cloud," *Computing*, vol. 98, no. 1-2, pp. 195–214, 2016.
- [8] M. Zaharia, M. Chowdhury, and M. J. Franklin, "cluster computing with working sets," in *Proceedings of the Usenix Conference on Hot Topics in Cloud Computing*, p. 10, 2010.
- [9] J. E. Hadad, M. Manouvrier, and M. Rukoz, "TQoS: transactional and qos-aware selection algorithm for automatic web service composition," *IEEE Transactions on Services Computing*, vol. 3, no. 1, pp. 73–85, 2010.
- [10] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in *Proceedings of the 18th International World Wide Web Conference (WWW '09)*, pp. 881–890, ACM, Madrid, Spain, April 2009.
- [11] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient web service composition with end-to-end QoS constraints," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, article no. 7, 2012.
- [12] H. Jin, X. Yao, and Y. Chen, "Correlation-aware QoS modeling and manufacturing cloud service composition," *Journal of Intelligent Manufacturing*, 2015.
- [13] F.-C. Pop, D. Pallez, M. Cremene, A. G. B. Tettamanzi, M. Suciu, and M. Vaida, "QoS-based service optimization using differential evolution," in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11)*, pp. 1891–1898, Ireland, July 2011.
- [14] F. Seghir and A. Khababa, "A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition," *Journal of Intelligent Manufacturing*, pp. 1–20, 2016.
- [15] F. Tao, D. Zhao, Y. Hu, and Z. Zhou, "Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 4, pp. 315–327, 2008.
- [16] Y. Chen, J. Huang, C. Lin, and J. Hu, "A partial selection methodology for efficient QoS-aware service composition," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 384–397, 2015.
- [17] Y. Zhang, Z. Jing, and Y. Zhang, "MR-IDPSO: A novel algorithm for large-scale dynamic service composition," *Tsinghua Science and Technology*, vol. 20, no. 6, Article ID 7349932, pp. 602–612, 2015.
- [18] M. S. Hossain, M. Moniruzzaman, G. Muhammad, A. Ghoneim, and A. Alamri, "Big Data-Driven Service Composition Using Parallel Clustered Particle Swarm Optimization in Mobile Environment," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 806–817, 2016.
- [19] M. Zaharia, M. Chowdhury, T. Das et al., "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the Usenix Conference on Networked Systems Design and Implementation*, p. 2, USENIX Association, 2012.
- [20] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micromachine and Human Science*, pp. 39–43, Nagoya, Japan, October 1995.
- [21] L. Zhang and B. Zhang, "A geometrical representation of McCulloch-Pitts neural model and its applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 10, no. 4, pp. 925–929, 1999.
- [22] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2014.