Instance Expansion Algorithm for Micro-service with Prediction

Qing Lin, Jinhuan Wang, Jing Wang
School of Intelligence Science and Information Engineering
Xi'an Peihua University, Xi'an, China
530610232@qq.com, 360635476@qq.com, 369607673@qq.com

Keywords: Micro-service; Instance expansion; Prediction; Maximum correntropy criterion

Abstract: Instances expansion of micro-service consumes time for constructing new instances, it can't satisfy the requirement of low latency service, such as scientific calculation workflows. In order to reduces the instance expansion time, an instance expansion algorithm for micro-service with prediction is proposed in this letter, which sets correntry as cost function and uses MCC (Maximum Correntropy Criterion) to filter the burst service requirements to improve the accuracy of prediction, and use stochastic gradient descent algorithm to train the data set to predict the required micro-service instances in the next time. The performance of the proposed algorithm are analysed in real experiment telecom office with the compared ones using A/B test, and the experiment results show that the proposed algorithm has 70% less time of instance expansion than the compared ones, and the accuracy of the proposed algorithm is 20% more than the compared one which uses LMS as the cost function.

1 Introduction

Micro-service architecture is easier to deploy, update and expand than monolithic architecture[1]. It has been put into practice by Google, Facebook, Microsoft, Netflex. The micro-service architecture is capable of expanding horizontally by constructing new instances for the increasing service load. However, the existed micro-service expanding mechanisms use static algorithms to construct micro-service instances, they consume time for constructing new instances when new services come and can't satisfy the requirement of low latency services. Paper 2 provides a container-based resource assignment algorithm to initialize micro-service instances. Paper 3 provides a resource management algorithm for heterogeneous scientific calculation workflows. They are all static algorithms and consuming extra time when be used in the scenario of micro-service expansion. In order to reduce the time for constructing new micro-service instances in the scenario of micro-service expansion, an instance expansion algorithm with prediction is provided in this letter to pre-construct micro-service instances for eliminating the constructing time of new micro-service instances when service requirements come. As entropy of co-relation (correntropy) depicts the high-order moments between the prediction and real outputs[4], this algorithm sets correntropy as the cost function and uses MCC (Maximum Correntropy Criterion) to filter the burst service requirements, uses stochastic gradient descent algorithm to train the system and predict the future service requirement for micro-service instances. The performance of the proposed algorithm and compared ones are analysed in the real experimental telecom office to attest the advantages of the proposed algorithm in this letter. The contributions of this paper are described as follows: 1) This paper proposes prediction algorithm to reduce the time for instances expansion of micro-service for the first time; 2) This paper sets MCC as cost function to filter the burst service requirements, which could affect the accuracy of the prediction.

2 Instances Expansion Algorithm for Micro-service with Prediction

In order to predict the instance of micro-service with burst requirements, the cost function of the prediction system is defined as formula(1) to describe the correntropy of prediction output and prediction, which could filter the non-liner noise of system. $k(\cdot)$ is Gaussian kernel function and

$e = y - x$ depicts the error between real output and predicted. The historical record of micro-service instances is recorded as X_i, and the predicted result is marked as Y.

$$V(Y,X) = E\big(k(Y-X)\big) = \iint k(y-x) f_{YX}(y,x)dydx = \int k(e)\,de \tag{1}$$

$$k(\cdot) = G(\cdot) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\cdot)^2}{2\sigma^2}} \tag{2}$$

Parzen window estimation is used to get the approximate expression of (1), which is the defined cost function presented in (3), and gets the maximum value of it to get the MCC.

$$\widehat{v(y,x)} = \frac{1}{N}\sum_{i=1}^{N} G_\sigma(y_i - x_i) = \frac{1}{N}\frac{1}{\sqrt{2\pi}\sigma}\sum_{i=1}^{N} e^{-\frac{\left(y_i^R - y_i^E\right)^2}{2\sigma^2}} \tag{3}$$

The real output of and the depicted output of the proposed algorithm are marked as y_i^R and $y_i^E = W_n^T \cdot X_i$ respectively, in which W_n^T is the vector weight of the neural network in the proposed algorithm. In order to get the optimal vector weight for getting the maximum cost function, stochastic gradient descent algorithm is used to modify the vector weight as (4) to (6), in which η is the learning rate.

$$W(n+1) = W(n) + \eta \nabla \widehat{v(y,x)} \tag{4}$$

$$\nabla \widehat{v(y,x)} = \frac{1}{N}\frac{1}{\sqrt{2\pi}\sigma^3}\sum_{i=1}^{N} e^{-\frac{\left(y_i^R - W_n^T \cdot X_i\right)^2}{2\sigma^2}} (y_i^R - W_n^T \cdot X_i) X_i \tag{5}$$

$$W(n+1) = W(n) + \frac{\eta}{N}\frac{1}{\sqrt{2\pi}\sigma^3}\sum_{i=1}^{N} e^{-\frac{\left(y_i^R - W_n^T \cdot X_i\right)^2}{2\sigma^2}} (y_i^R - W_n^T \cdot X_i) X_i \tag{6}$$

3 Performance Analysis

In order to analyse the performance parameters of micro-services initialization supported in paper 5, the proposed algorithm and compared ones are performed in the experimental telecom office with A/B test, in which all the performed algorithms are loaded on the same ratio of service flows equally. The hardware environment of the experiment contains 16 blades of HP C7000 and software environment is OPS+ system of ZTE. The training data set contains the number of micro-service instances in the last 30 days.

3.1 Performance with Constant Flavour of Micro-service Instances

In order to analysis the performance of the proposed algorithm, in the same production test business parameters, we use the algorithm of [6], literature [2], literature [7] to replace the proposed algorithm, and verify the performance of different algorithms in the micro-service management framework. In order to facilitate the analysis, the literatures [6], the literature [2], the literature [7] and the algorithm mentioned in this paper are respectively described as MSV, MSH, MSC and MSA.

When the size of a micro-service instance is fixed, the maximum number of micro-service instances on each physical server are consistent and the remaining computing resources are consistent. In the experiment, T_v is set to 1, T_r is set to 6, is less than the one of the smallest micro-service instances computational resources.

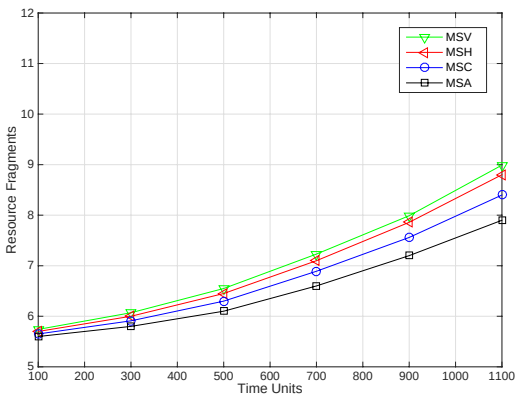

Figure 1. Analysis of resource fragments

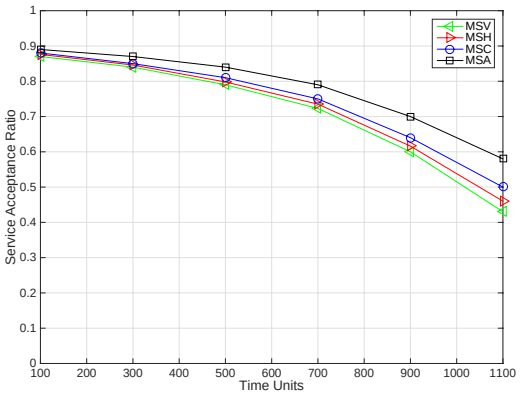

Figure 2. Services acceptance ratio

Fig.1 shows a curve of resource fragments along the time, at which point the specification of the micro-service instance is 2 VCPUs and 16GB memory. As time goes on, both the proposed algorithm and the contrast algorithms have the fragments of resources, but the proposed algorithm has fewer resource fragments, which shows that the proposed algorithm can select the most suitable micro-service instance to host the user's service requirement.

Fig.2 shows the user's service requirement acceptance rate. As can be seen from the graph, the acceptance rate of the proposed algorithm is higher than that of the comparison algorithm, because the acceptance rate of the service requirement is inversely proportional to the number of resource fragments. The more fragmented the resource, the more the total remaining resources of the physical server are, but any one fragment can not meet the resources of the service requirement, which causing the user's service requirement to be blocked.

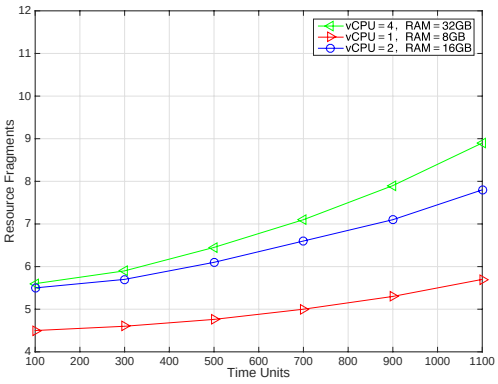

Figure 3. Resource fragments with different flavors of micro-services

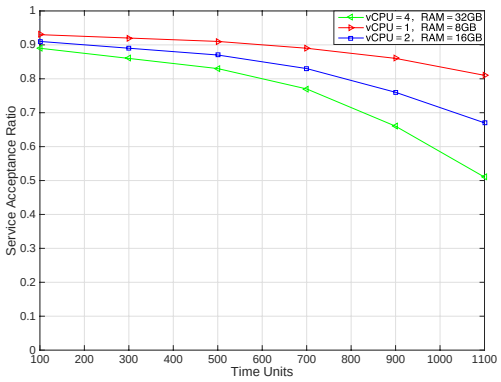

Figure 4. Service acceptance ration with different flavors of micro-services

Fig.3 shows the use of the same resource fragment thresholds and the resource fragments of the proposed algorithm when the micro-service instance specification is set differently.

As can be seen from the diagram, if the resource fragment threshold does not change, when the micro-service instance specification becomes larger, the probability of the remaining computing resources on the physical server cannot create new micro-service instances increases, and more resource fragments occurs. Conversely, when the micro-service instance specification becomes small, the granularity of distributed micro-service instances on the physical server becomes smaller, which can accommodate more micro-service instances and less resource fragmentation.

Fig.4 shows the acceptance rate of service requirement under different Micro service instance specifications, because the acceptance rate of the business request is inversely proportional to the fragments of the resources, the number of resource fragments is higher when the micro-service instance is larger, resulting in a lower acceptance rate of the service request.

3.2 Performance with Variable Flavour of Micro-service Instances

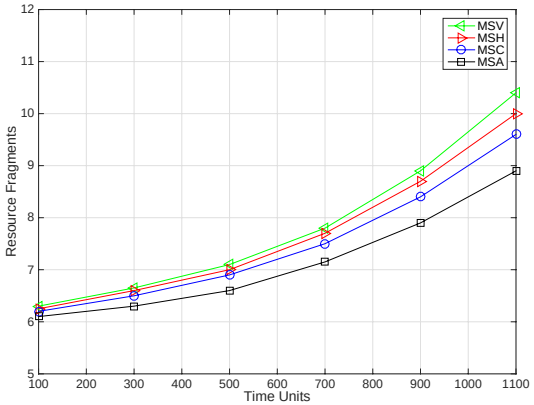

Figure 5. Resource fragments along with different flavors of micro-services

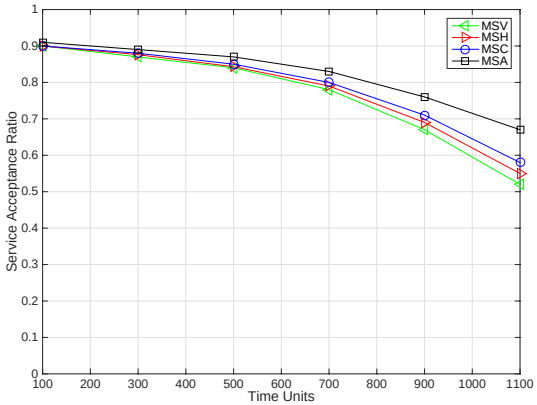

Figure 6. Service acceptance ration along with different flavors of micro-services

Fig.5 shows a curve of resource fragments along the time, as you can see from the figure, as the micro-service instance specification changes over time, more resource fragments occurs on the physical server than the fixed micro-service instance specification. Compared with the comparison algorithm, because the optimization performance of the proposed algorithm is better than the comparison algorithm, the physical server appears less resource fragmentation.

As shown in Fig.6, when the specification of a micro-service instance is changed, the service request acceptance rate of the proposed algorithm is higher than the comparison algorithm because the acceptance rate of the service request is inversely proportional to the resource fragments. However, along with the time, the resource fragments on the physical server increases, and the acceptance rate of the proposed algorithm and the comparison algorithm will decrease.

3.3 Running Time Analysis

The time of instance expansion of the proposed algorithm and [2], [3] is presented in Fig.7, which are marked as IEA-MCC, IA-CB and IA-SA. As the figure presents, the time for instance expansion of the proposed algorithm is almost 70% less than the compared algorithms. The time for instances expansion of the compared algorithm in [2] and [3] have two parts, which are the construction time and discovery time of new instances. The reduction time of the proposed algorithm than the compared algorithms is just the construction time of new instances in the compared algorithms.

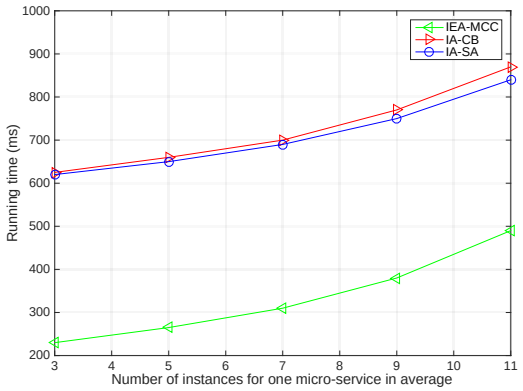

Figure 7. The time of instances expansion for the proposed algorithm and compared algorithms

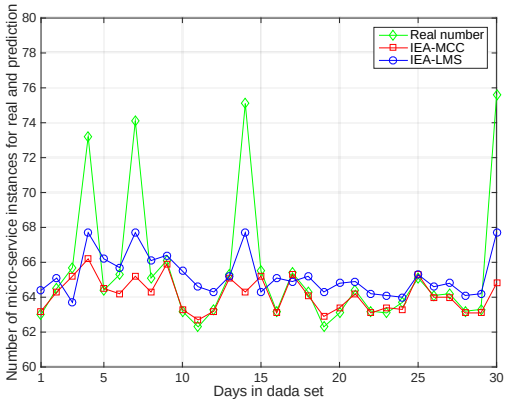

Figure 8. The number of micro-service instances for real and prediction

The real records and the prediction results of the proposed algorithm and the compared one that uses LMS (Least Mean Square) are presented in Fig.8. The standard deviation between the prediction results and real records of the proposed algorithm and the compared one is presented in Fig.9 to analyse the accuracy of the proposed algorithm. The proposed algorithm and the compared one are marked as IEA-MCC and IEA-LMS respectively.

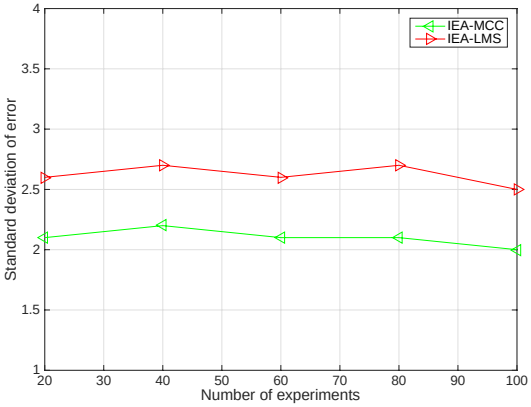

Figure 9. Standard deviation of the proposed algorithm and compared one

Form Fig.8 and Fig.9, the proposed algorithm in this letter has almost 20% lower standard deviation of error than the compared one. The reason is that the training data set contains burst service requirements and they could affect the accuracy of prediction. The algorithm proposed in this letter uses MCC as the cost function, in which the burst service requirement is recognized as non-Gaussian noise, to filter the burst service requirement and reduce the influence of them.

4 Conclusions

This letter proposes an instance expansion algorithm with prediction, which uses prediction algorithm to predict how many micro-service instances need to be pre-constructed to satisfy the service requirement to eliminate the time of constructing new micro-service instances. Furthermore, in order to increase the accuracy of prediction, the prediction algorithm sets correntroy as the cost function and uses MCC to filter the burst service requirements. The time of instance expansion of the proposed algorithm is compared with the ones without prediction mechanism, and the prediction accuracy of the proposed algorithm is compare with the one with LMS cost function in the experimental telecom office with A/B test. The experiment results show that the proposed algorithm reduces the time of instances expansion almost by 70% than algorithms without prediction, and the standard deviation of error of the proposed algorithm is 20% less than the one of prediction algorithm with LMS cost function in average with the real service requirement. The proposed algorithm in this letter introduces prediction mechanism into instances expansion micro-service for the first time, and this letter gives references to design the instances expansion algorithm with prediction.

References

[1] Alshuqayran N, Ali N, Evans R. 2016. A Systematic Mapping Study in Microservice Architecture, International Conference on Service-Oriented Computing and Applications, 10, 44-51.

[2] Kan C. 2016. DoCloud: An elastic cloud platform for Web applications based on Docker, International Conference on Advanced Communication Technology, 7, 478-483.

[3] Nguyen P, Nahrstedt K. 2017. MONAD: Self-Adaptive Micro-Service Infrastructure for Heterogeneous Scientific Workflows, International Conf. on Autonomic Computing, 9, 187-196.

[4] Hua Q, Wentao M, Jihong Z, et al. 2013. Kernel Least Mean Kurtosis Based Online Chaotic Time Series Prediction, Chinese Physics Letters, 30(11), 468-477.

[5] Zhu X, Gong X, Tsang D. 2017. The optimal macro control strategies of service providers and micro service selection of users: quantification model based on synergetics, Wireless Networks, 1-14.

[6] Guo D, Wang W, Zeng G, et al. 2016. Microservices Architecture Based Cloudware Deployment Platform for Service Computing, IEEE Service-Oriented System Engineering,358-363.

[7] Nguyen P, Nahrstedt K. 2017. MONAD: Self-Adaptive Micro-Service Infrastructure for Heterogeneous Scientific Workflows, IEEE International Conference on Autonomic Computing, 9:187-196.