A Self-Adaptation Framework of Microservice Systems

Shuai Zhang, Xinjun Mao, Peini Liu and Fu Hou
College of Computer
National University of Defense Technology
Hunan, China 410073
{zhangshuai16a, xjmao, liupeini16, houfu}@nudt.edu.cn

***Abstract*—Microservice has been more and more applied to build software systems in industry field and research in academic field. And software systems are increasingly expected to dynamically self-adapt to accommodate resource variability, changing user needs, and system faults. Compared with traditional software systems, microservice systems have some characteristics, such as the highly self-contained components and the dynamic running instance, which pose challenges to traditional self-adaptation methods. Therefore, it needs to propose corresponding techniques and methods to cope with the characteristics of microservice systems. This paper analyzes the special self-adaptive requirements of microservice systems, proposes a microservice reference model, which describes basic elements and their relationships of microservice systems. Then we present a microservice system self-adaptation framework *MSSAF* to support the self-adaptation of microservice systems. We illustrate the feasibility and effectiveness of our approach in the context of a microservice system case.**

Keywords-self-adaptation; microservice; reference model; framework

I. INTRODUCTION

Microservices recently demonstrated to be an effective architectural paradigm to cope with software complexity and scalability [1]. Although microservice has received more and more pay attention, there is no generally accepted definition for microservice. A widely-recognized concept of microservice is proposed by Martin Fowler and James Lewis [2]. The success of the paradigm has been demonstrated in some domains, including mission-critical systems [3]. However, while microservice systems run in open and dynamic environment, there are still many uncertainties, e.g. changing user requirements and unpredictable system errors. To deal with the uncertainties, microservice systems need dynamically self-adapt to state changes of external environment and itself.

Microservice systems have some characteristics, e.g. platform-dependent and one microservice may have multiple instances. Since traditional software systems don't have these characteristics, the traditional self-adaptation methods have no specific solutions.

To realize the self-adaptation of microservice systems more efficient, this paper proposes a microservice reference model and a microservice system self-adaptation framework *MSSAF*. The microservice reference model describes the basic elements and their relationships of microservice systems. Microservice system self-adaptation framework *MSSAF* integrates some necessary tools to provide support for the self-adaptation of microservice systems. Based on these methods and tools, this paper has achieved better cost-effectiveness in self-adaptation of microservice systems.

The rest of this paper is structured as follows. Section II discusses some related works. Section III describes a microservice reference model. Section IV gives a detailed description of microservice system self-adaptation framework *MSSAF*. In section V, our works are illustrated by an intelligence system which is a microservice system. Finally, we summary this paper and discuss further works.

II. RELATED WORK

Nowadays microservice has become more and more popular in software engineering field. There are some summaries of existing studies on microservice find that most of the studies focused on microservice application, method and architecture [4]. Those studies mainly focus on maintainability, extensibility, complexity, flexibility and so on, while there are few studies related to the self-adaptivity of microservice. Nicola Dragoni et al. describe the origin, current and future of microservice in detail [1]. Franco Callegati et al. describe a service-oriented architecture that exploits the microservices orchestration paradigm to enable the creation of new services [5]. Massimo Villari et al. present an Orchestration Broker for each involved Fog computing node[6]. Despite these works consider the characteristics of microservice, there is no self-adaptation in their works. Sara Hassan and Rami Bahsoon refer to microservice and self-adaptation in their work [7]. But they consider the self-adaptation microservice from a design viewpoint without concrete implementation.

The aim of self-adaptation is to let the system monitor itself and based on its goals reconfigure or adjust itself to satisfy the changing conditions, or if necessary degrade gracefully [8]. In recent years, many researchers have proposed many methods to study the self-adaptation from different aspects. Shang-Wen Cheng and David Garlan propose a self-adaptation framework Rainbow which is based on software architecture [9]. Denaro

DOI reference number: 10.18293/SEKE2018-091

Giovanni et al. proposes an approach to design self-adaptive service-oriented architectures [10]. Marcello Thiry and Roger Anderson Schmidt present relevant approaches of self-adaptive systems driven by runtime models [11]. Besides, Luca Florio and Nitto focus on how to add autonomic capabilities to microservices without changing the way they are implemented but exploiting their containers [12]. The paper considers that microservice is a kind of distributed component, but they mainly discuss about autonomic capabilities which are different from self-adaptive capabilities.

In the above researches, there are few works related to self-adaptation in the study of microservice. To cope with the characteristics of microservice, this paper analyzes the characteristics and proposes corresponding solution to realize the self-adaptation of microservice systems.

III. Microservice Reference Model

Microservice manages growing complexity by functionally decomposing large systems into a set of independent services. Despite microservice is an architectural pattern emerging out of Service-Oriented Architecture (SOA), it shows some important distinctive characteristics that are different from SOA:

Platform-dependent: There may be a lot of microservices in microservice software systems, communications are complexity and configurations are rather difficult and can also be prone to error. To make up these limitation, corresponding platforms must be developed to manage MSS.

Multi-instance: In microservice systems, instance is a running entity of a microservice, and a microservice can have multiple instances. At runtime, there are a lot of microservice instances that interact to realize the application logic of microservice systems.

The characteristics mentioned above propose some special needs for the self-adaptation of microservice systems. Since traditional self-adaptation methods don't have special solutions for these characteristics, the best effect cannot be achieved when realizing the self-adaptation of microservice systems. Therefore, it is necessary to propose some methods to deal with the characteristics. Based on descriptions of microservice architecture in some microservice literatures [13][14][15], we propose a microservice reference model (show in Fig. 1). The model consists of three layers, including system layer, service layer and instance layer.

System layer consists of three parts, microservice system realize the business logic, management platform provides management capabilities for microservice system, and limitations impose constraints on microservice system. **Service layer** consists of some microservices and corresponding protocols, microservice communicates with each other through protocols to compose microservice systems. **Instance layer** includes containers and microservice instances. Microservice instances run in containers. A set of microservice instances compose a microservice and they communicate with other microservice instances to realize concrete functions. The reason for such layering is as follow. System layer realizes business logic, mainly focuses on high-level goals and doesn't involve concrete implementation. Service layer focuses on how microservices cooperate to provide services for the system layer. Instance layer mainly related to the specific implementation. These three layers provide different observation dimension for microservice systems, and clearly express the characteristics which have mentioned earlier.

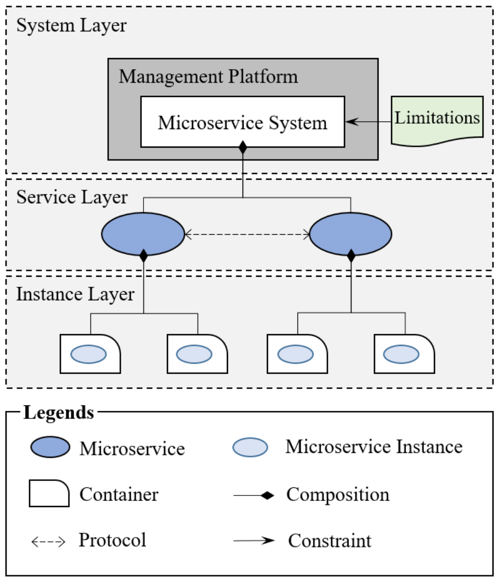


Fig. 1 Microservice reference model

IV. MSSAF: Self-Adaptation Microservice System Framework

The customizable self-adaptation framework has many advantages. For example, it reduces the cost of development by providing a substantial base of reusable infrastructure greatly. And it allows engineers to tailor the framework to different systems with relatively small increments of effort by providing separate customization methods.

To make microservice systems self-adaptation, we refer to the MAPE loop proposed by IBM [16]. For each part of the MAPE loop, we provide a corresponding tool, meanwhile we also provide some other necessary tools. Based on the microservice reference model, this paper proposes a Microservice System Self-Adaptation Framework (MSSAF) (as shown in Fig. 2). To automate system self-adaptation, we provide a self-adaptation strategy description language to represent self-adaptation scenario.

MSSAF consists of three parts, including translation tools, self-adaptation engine and microservice systems. Translation tools is used to translate the self-adaptation strategies, the results of translation are used to support self-adaptation engine to complete the self-adaptation logic. Self-adaptation engine communicates with other two parts to implement self-adaptation. Management platform and microservice systems correspond to the system layer of microservice reference model.

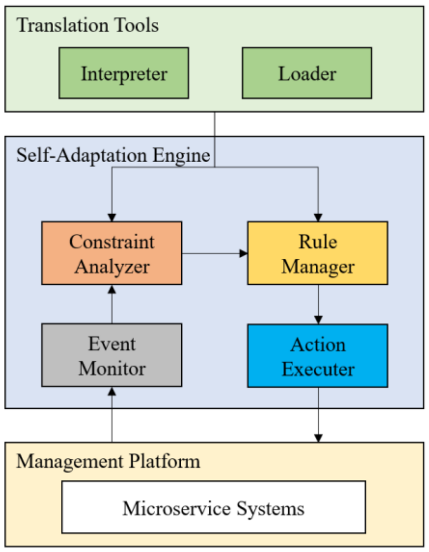


Fig. 2 Self-adaptation microservice system framework

The reason why MSSAF has no the service layer and instance layer of microservice reference model is that we mainly focus on the self-adaptation process of microservice systems in MSSAF. The actual adjustment objects are the microservice and microservice instance which are the parts of microservice systems.

Self-adaptation engine consists of several components that provide the monitoring, detection, decision, and action capabilities of self-adaptation. *Event Monitor* obtains the monitoring data of microservice systems from the management platform. *Constraint Analyzer* uses informations which come from Event Monitor to judge whether there is a constraint violation. There are some alternative constraints, and the thresholds are specified in self-adaptation strategies by users. *Rule Manager* selects corresponding rules after triggering by Constraint Analyzer. A self-adaptation rule consists of two parts, including condition and action. The condition includes a signal which is sent by Constraint Analyzer for indicating if the rule is triggered, and it may include some essential constraints which are specified at design time. The actions of self-adaptation rules are the abstract operations of different objects. *Action Executer* performs the actual operations for microservice systems according to the actions of self-adaptation rules which are managed by Rule Manager.

Translation tools include an Interpreter and a Loader. Strategies cannot be directly executed and needs to be interpreted before running. Since we provide some common self-adaptation strategies, it needs a tool for static translation. And when there is a need for adding new self-adaptation strategies at runtime, it needs to provide a tool for dynamic loading. Our translation tools can be customized for different program languages and platforms.

V. Illustration of MSSAF with Intelligence System

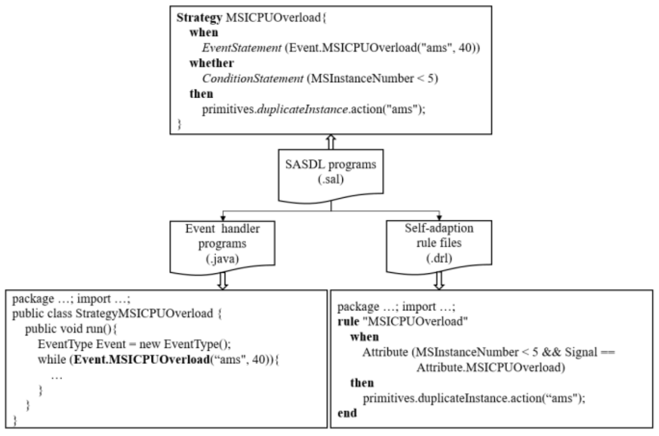
Since there are few related studies as similar as our works, it cannot compare our works with others. This paper adopts an intelligence system to illustrate the feasibility and availability of MSSAF.

The intelligence system is used to collect and manage intelligence from different sources and provides the functions of search, analysis, distribution for these intelligences. Besides, the intelligence system also provides a frontend service and a user service for access and management. When there are many users accessing the system in a short time, it will need expand to reduce CPU load. Correspondingly, when user access is too small, it will need shrink to be cost saving. To meet these requirements, we use MSSAF to design and realize the self-adaptation of the intelligence system.

To illustrate the feasibility of intelligence system self-adaptation by using MSSAF, we use the CPU usage rate of microservice instance as the quality attribute that self-adaptation concerns. We monitor the CPU usage of a microservice instance for a period of time, and take the average as CPU usage rate of this instance, namely "MSICPUUsage".

Self-adaptation scenario: If there is a swift growth of MSICPUUsage in a short period of time, and there is a threshold, such as 40%. Once the MSICPUUsage exceeds this threshold, the self-adaptation rule is triggered. When the constraints in the self-adaptation rule are satisfied, this self-adaptation rule will be executed. Finally, the specific modification is mapped to the platform by a series of self-adaptation actions.

Experiment: The experimental object is analysis microservice (ams), the experimental setup is that the initial instance number is 2, the refresh interval is 120s, the CPU usage rate threshold is set to 40%, and the number of microservice instance is limited to no more than 5. The translation of strategy StrategyMSICPUOverload is shown in Fig. 3.


```
Strategy MSICPUOverload{
  when
    EventStatement (Event.MSICPUOverload("ams", 40))
  whether
    ConditionStatement (MSInstanceNumber < 5)
  then
    primitives.duplicateInstance.action("ams");
}
```

```
package …; import …;
public class StrategyMSICPUOverload {
  public void run(){
    EventType Event = new EventType();
    while (Event.MSICPUOverload("ams", 40)){
      …
    }
  }
}
```
```
package …; import …;
rule "MSICPUOverload"
  when
    Attribute (MSInstanceNumber < 5 && Signal ==
                Attribute.MSICPUOverload)
  then
    primitives.duplicateInstance.action("ams");
end
```

Fig. 3 The translation of self-adaptation strategy StrategyMSICPUOverload

In the experiment, we use the Locust performance testing tool to simulate the stress test, it sets as 500 users access 1000 times per second, and it stops after 10 minutes. Fig. 4 indicates the changes of the number and CPU usage rate of microservice instance during the experimental process.

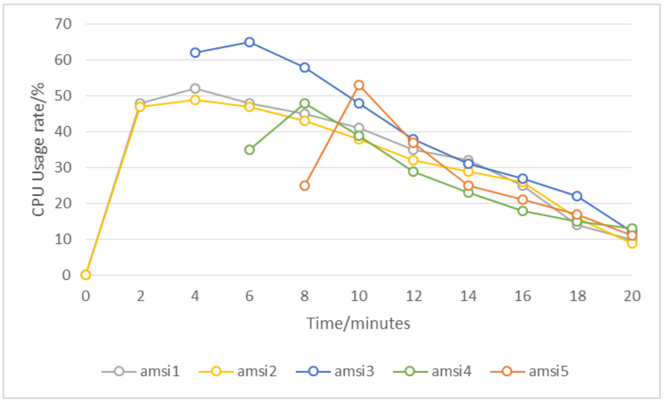


Fig. 4 The diagram of experiment process

As shown in Fig. 4, analysis microservice have two instances amsi1 and amsi2 at beginning. There are large numbers of user accesses in 2 minutes, the CPU usage rate of microservice instance increases rapidly, then the overloading strategy is triggered. The increased amsi3 handles large numbers of accesses in 4 minutes, CPU usage rate is too high, and the overloading strategy will be triggered. The increased amsi4 begins to handle user requests in 6 minutes, the first 3 instances' CPU usage rates still exceed the threshold 40%, and the overloading strategy continues to be triggered. The instance amsi5 was added in 8 minutes, while other instances' CPU usage rates exceed the threshold, the strategy won't be triggered now because the self-adaptation rule set the number of microservice instance to be no more than 5. The stress test has stopped after 10 minutes, the CPU usage rates of the five instances tend to get closer because of the load balancing.

Through the experiment above, we illustrate that MSSAF can provide some preliminary self-adaptation capabilities for microservice systems. From the experimental results, the self-adaptation implementation of MSSAF can utilize the characteristics of microservice systems very well and conduct elastic expansion on more fine-grained. In short, MSSAF has the potential to satisfy the self-adaptation and customization requirements of microservice systems.

VI. Conclusion

To deal with the problems that microservice systems need to be self-adaptation and traditional self-adaptation methods cannot cope the characteristics of microservice systems, we analyze the characteristics of microservice systems and propose a microservice reference model. To implement the self-adaptation of microservice systems and provide reuse infrastructures, we propose a self-adaptation microservice system framework *MSSAF*, which can implement the self-adaptation of microservice systems. Through an example of an intelligence system, we illustrate that *MSSAF* can provide self-adaptation capabilities for microservice systems.

In future work, we would like to improve our approach by enhancing these tools, such as adding strategy conflict detection for Constraint Analyzer, adding strategy selection algorithm for Rule Manager, etc. These may be our future research directions. In addition, each component of self-adaptation engine can be published as a microservice, then it will be possible to realize the self-adaptation of self-adaptation logic via multiple MAPE loops to expand the self-adaptation capabilities.

Acknowledgment

This research is supported by research grants from Natural Science Foundation of China under Grant No. 61532004 and 61379051.

References

[1] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," *Present and Ulterior Software Engineering*, Springer, Cham, 2017.

[2] J. Lewis and M. Fowler, *Microservices*, http://martinfowler.com/articles /microservices.html.

[3] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How To Make Your Application Scale," *arXiv preprint* arXiv:1702.07149, 2017.

[4] C. Pahl, and P. Jamshidi, "Microservices: A Systematic Mapping Study," in *International Conference on Cloud Computing & Services Science*, 2016, pp. 137-146.

[5] F. Callegati, G. Delnevo, A. Melis, S. Mirri, M. Prandini, and P. Salomoni, "I want to ride my bicycle: A microservice-based use case for a MaaS architecture," in *IEEE Symposium on Computers and Communications*, 2017, pp. 18-22.

[6] M. Villari, A. Celesti, G. Tricomi, A. Galletta, and M. Fazio, "Deployment orchestration of microservices with geographical constraints for Edge computing," in *IEEE Symposium on Computers and Communications*, 2017, pp. 633-638.

[7] S. Hassan, and R. Bahsoon, "Microservices and Their Design Trade-Offs: A Self-Adaptive Roadmap," in *IEEE International Conference on Services Computing*, 2016, pp. 813-818.

[8] D. Weyns, "Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges," *Handbook of Software Engineering*, Springer, 2017.

[9] S.-W. Cheng, "Rainbow: Cost-Effective Software Architecture-Based Self-Adaptation," *Dissertations & Theses - Gradworks*, 2008.

[10] G. Denaro, D. Tosi, and D. Schilling, "Towards self-adaptive service-oriented architectures," in *Workshop on Testing*, 2006, pp. 10-16.

[11] M. Thiry, and R. A. Schmidt, "Self-adaptive Systems Driven by Runtime Models," in *The International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2017, pp. 248-253.

[12] L. Florio, and E. D. Nitto, "Gru: An Approach to Introduce Decentralized Autonomic Behavior in Microservices Architectures," in *IEEE International Conference on Autonomic Computing*, 2016, pp. 357-362.

[13] D. Namiot, and M. Sneps-Sneppe, "On Micro-services Architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, pp. 24-27, 2015.

[14] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 113-116, 2015.

[15] M. Mazzara, K. Khanda, R. Mustafin, V. Rivera, L. Safina, and A. Sillitti, "Microservices Science and Engineering," in *International Conference in Software Engineering for Defence Applications*, Springer, Cham, 2016, pp. 11-20.

[16] G. Agarwal, R. Shah, J. Walrand, H. A. Alzoubi, S. S. Lee, M. Rabinovich, O. Spatscheck, V. D. M. Jacobus, I. Avramopoulos, and M. Suchara, "An Architectural Blueprint for Autonomic Computing," *IBM White Paper*, 2006, pp. 31.