A Container Initial Deployment Method Based on Microservice Level of IoT

Yong-xiong ZHANG¹ and Liang-ming WANG²,*

¹Department of Economics and Trade, Guangzhou College of Technology and Business, China
²School of Software Engineering, South China University of Technology, China
*Corresponding author

Keywords: IoT, Microservice, Docker.

Abstract. The microservice architecture divides a single application into a set of independent and business-logic-specific services, and the services coordinate and cooperate with each other to provide the ultimate value to the users. Aiming at the high availability of IoT platform during the initial deployment phase, this paper designs and implements a strategy of container’s initial deployment based on microservice level of IoT platform. Through the model of microservice level on the IoT platform proposed in this paper and the calculation model of container redundancy based on the model mentioned above, the unified management of many microservices can be realized. To ensure the high availability of the IoT platform, the microservice will be deployed to the container cluster and the redundant backup of microservice will be carried out.

Introduction

With the development of network technology and widely use of smart devices, the importance of the platform in the ecology of the IoT industry has become increasingly prominent [1]. As a connection hub between devices and application services, the platform of IoT has to meet needs of connecting a large number of devices, while providing highly available application services and good user experiences. Therefore, high availability is the key demand for it. The common technologies with maintaining platform include redundancy backups, horizontal scaling, and so on.

As a loosely coupled evolutionary architecture [2], the microservice architecture can decouple the complex IoT. Also, Docker container isolates resources by the virtualization of the operating system layers [3]. On the one hand, it can well cope with the problem caused by the fact that the microservice architecture requires a high degree of autonomy for independent deployment and operation, on the other, it can make full use of the machine resources and helps to reduce time for the application deployment.

Compared to traditional virtual machines, the container technology, an operation system-level virtualization, can run multiple container examples on a single machine node to serve as application service nodes [4-6]. It helps to build a redundant backup environment, while reducing hardware cost by effectively using the resources of machine nodes. Over these years, the rapid development of Docker has made itself a typical representative of container technology. Docker can run on any platform such as physical machine, public cloud, and private cloud and so on. This compatibility allows users to easily deploy Docker images to any Docking-running environment, not having to worry about the differences in operating system or platform of the production environment.

Therefore, the microservice-based architecture of IoT can independently deploy each service in a loose service manner with its flexibility [7-10]. On the other hand, the initial deployment of the platform can be realized based on Docker container technology. And the redundant backup of the service is completed during the initial deployment, to ensure its high availability as a whole [11,12].

Container Initialization Deployment Architecture for the IoT Platform at Microservice Level

The overall architecture of the container initial deployment based on the IoT platform at microservice level in this paper is shown in Fig. 1. Under the loosely coupled microservice architecture, the IoT has
a large number of microservices of different methods in realizing business logic. There are differences in the guarantee requirements for availability. Therefore, the microservice level model in this paper determines the level of each microservice on the IoT platform. Then we configure the parsing model and analyze and process a series of data in terms of the level, the dependent relationship and the access port in microservice in YAML, which aims to carry out the next container deployment operation. Then, the container deployment model will calculate the redundancy of each microservice by the redundancy calculation model in this paper according to the configuration data of the microservice level and its dependency relationship. Based on the redundancy, the initial deployment of the microservice in the container cluster can be realized, and finally the highly available IoT platform function is provided to the user.

![Diagram showing the process of container deployment based on microservice level of IoT](image)

**Microservice Level Model of IoT Platform Based on Fuzzy Mathematics**

**Description for Microservice Level Model Method**

On the basis of the microservice architecture, there are a large number of microservices in the IoT platform, and they are diversified according to the different business logic. However, in the process of redundant backup of microservice, to ensure high availability of the platform, the operation is complicated and it is difficult to implement one-click deployment. Therefore, this paper innovatively presents the concept of microservice level. At present, there is relatively little research on how to classify the microservice level of IoT platform. One of the most important reasons is the uncertainty of the criteria for evaluating microservices level. The microservice level of IoT platform is ambiguous. There is no quantitative standard for the factors affecting the microservice level and it is impossible to obtain accurate data. Therefore, this paper uses fuzzy evaluation method to determine the level of microservice of the IoT platform.
Fuzzy comprehensive evaluation method is a widely used method in fuzzy mathematics. When evaluating a transaction, we will often encounter such kind of problems, where each and every one of the factors needs to be analyzed, as evaluation is determined by many factors. How to make a comprehensive evaluation with all factors on the basis of making individual comments on each factor? The basic idea of fuzzy evaluation is that the boundary of the transaction is not very obvious, and it is difficult to attribute it to a certain category when evaluating. So we should evaluate the affected factors one by one at first, and then implement a comprehensive fuzzy evaluation of all factors. This is to prevent the missing of any statistics and the midway loss of information. This is useful to resolve the problem of deviation from objective reality caused by such deterministic judgment and evaluation.

The main body of this paper is composed of all the services of the IoT platform based on the microservice architecture. According to the research of high availability of IoT platform, the level of microservice will be higher when the availability requirement is higher. From the availability requirement of using terminal of IoT platform, the following four elements are defined as the determining factors for service availability. Element 1: IoTDevice; Element 2: UserApplication; Element 3: ManagementPlatform; Element 4: ThirdPartyApplication.

The principle of evaluation:
(1) The level will be higher when the availability requirement of IoTDevice is higher.
(2) The level will be higher when the availability requirement of UserApplication is higher.
(3) The level will be higher when the availability requirement of ManagementPlatform is higher.
(4) The level will be higher when the availability requirement of ThirdPartyApplication is higher.

The step of determining the level of microservice in IoT platform is as following:
Step 1 Microservice P is supposed as the evaluation object.
Step 2 \( u_i \) \((i = 1, 2, \cdots, 4)\) is specified as the determination factor of Microservice P. \( u_i \) represents four elements as referred above. So, we can get the factor set \( U = \{u_1, u_2, \cdots, u_4\} \).
Step 3 \( V = \{v_1, v_2, v_3, v_4\} \) is supposed as the evaluation set of microservice P. \( v_1 \) represents grade 1, the lowest level of microservice, \( v_2 \) represent grade 2, and so on. Then grade 4 is the highest level of microservice.
Step 4 After making fuzzy evaluation with each factor in the set U, it will get the evaluation matrix \( R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{14} \\ r_{21} & r_{22} & \cdots & r_{24} \\ \vdots & \vdots & \ddots & \vdots \\ r_{41} & r_{42} & \cdots & r_{44} \end{bmatrix} \), \( r_{ij} \) represent the degree of membership of \( u_i \) about \( v_j \). A fuzzy comprehensive evaluation model is composed by \( (U, V, R) \).
Step 5 It is also called weight by making the importance index of each factor. It marks \( A = \{a_1, a_2, \cdots, a_n\} \), when \( \sum_{i=1}^{n} a_i = 1 \). In this paper, the weight sequence is calculated by statistical expert group scoring. The expert group is made from the related workers in IoT platform. Each expert in this group will give a mark on each factor. Then it will be got a weight sequence of each factor. And it get the average of the sequences after averaged.
Step 6 By combining the weights of various factors with the evaluation matrix, we get \( \bar{B} = A \cdot R = (\bar{b}_1, \bar{b}_2, \cdots, \bar{b}_m) \). After normalization, we get \( B = (b_1, b_2, \cdots, b_m) \).
Step 7 The maximum membership degree of B is used to determine the microservice level of P.

By analyzing and calculating the microservice in IoT platform by the referred model, we can get the initial rank sequence \( P = (p_1, p_2, \cdots, p_n) \). n represents the number of microservices.

**Deployment Policy for Container of Microservice Level**

**Redundancy Model of Container with Microservice**

From the level model of microservice, we know the grade set \( \text{Rank} = \{r_1, r_2, r_3, r_4\} \), which \( r_1 < r_2 < r_3 < r_4 \). The requirement of availability will be increasing with the level of microservice. It makes the high availability of the IoT platform by redundant backup in order to deploy multi-container instances of the IoT platform's microservices. Set \( \text{Redundancy} = \)
\{a_1, a_2, a_3, a_4\}, which \(a_1 < a_2 < a_3 < a_4\). \(a_4\) represents the highest redundancy. The redundancy will be increasing with the level of the microservice. We call this relationship as \(f_1\). According to this relationship, each element in Rank has its only partner in Redundancy, recorded \(f_1: \text{Rank} \rightarrow \text{Redundancy}\). Then, we can get \(R\) (Redundancy sequence of each microservice) = \(\{m_1, m_2, \ldots, m_i, \ldots m_n\}\), which \(i < n, n = \text{sum of microservice}, m_i \in \text{Redundancy}\). From the Fig. 2, we can know that there is a single and full mapping relationship between the microservice level and the redundancy of the container.

![Figure 2. Mapping Relationship between the Microservice Level and the Redundancy of the Container.](image)

However, in terms of defining levels of microservices by microservice level models, it is far from enough if we only consider from the perspective of using terminal as device, user application, and management platform. We should consider the dependencies between services within the IoT platform. So the redundancy will be not correct by using \(f_1\). For example, the level of microservice of Redis is \(r_3\), and its redundancy is \(a_3\), according to their relationship \(f_1\). But as the underlying layers on the platform, Redis is bound to be relied on other upper layers in the certain business process. So the redundancy of Redis should be \(a_4\).

The microservices architecture is an architectural model that advocates the division of a single application into a set of independent, business-logic-specific services that coordinate and work together to provide the ultimate value to the user. Therefore, the microservice architecture is essentially a distributed system. In terms of the communication between service nodes, it is essentially a manifestation of service dependency no matter the communication method, synchronous or asynchronous. It can select the services with dependency from the microservice in IoT platform. A directed graph \(P\) is constructed with this relationship. It is a directed acyclic graph, because there is no ring in the case of the dependency between services. This graph takes the microservice as node. The in-degree of the node indicates the dependencies of the service. It means that the greater the in-degree of the node, the more dependent by other microservices.

**Container Initialization Deployment Method**

The orchestration deployment technology of docker containers and docker Swarm is used to build container clusters and microservices deployment environment. In the Swarm cluster, the selection of working nodes and the distribution of container deployment requests are all completed by the management node.

At first, the management node receives the request of container deployment. Docker container implements resource control by technology of control group, so the request of container deployment can contain restrictions such as the specification of the resources available to the containers. The selected mode chooses the right working node by these restrictions. Then the management node gets the requested node from the spread policy. Spread policy is to get the working node of smallest numbers in containers cluster and distribute container deployment requests to that node, so as to achieve the goal of container distribution in container cluster as evenly as possible. The management node first establishes a heap structure for the information of the working node, and then constructs a small top heap according to the number of containers on the current node. In this way, the
management node can quickly obtain the required work node and distribute the container deployment request to this node to implement the deployment of the microservice.

However, the redundant backup is not considered in the work of Swarm cluster. Each container is a single body in Swarm cluster. There is not any relationship of redundant backup with them. So there will be a problem that multiple copies of the containers belonging to the same microservice are deployed on the same working node during the process of redundant backup. When the working node is unavailable, the microservice is not available too.

Therefore, in order to ensure the high availability of the IoT platform through the deployment of container redundant backup and improve the shortcomings of deployment strategy in Swarm cluster, this paper optimizes the module of management node. To solve the problem that Swarm cluster lacks support for container redundancy scenarios, an optimal solution and a suboptimal solution can be obtained. The optimal solution is that for a container deployment request, the final working node should not contain a container instance belonging to the same microservice as the container to be deployed. Meanwhile, the node is the least number of deployed containers in the node that meets this condition. The suboptimal solution is that for a container deployment request, all the nodes that satisfy the constraints such as the custom resource limit already have instances belonging to the same microservice as the container to be deployed. The nodes with the least number of deployed containers can be obtained only from these working nodes.

Experiment Analysis

Experimental Environment Configuration

Three machine nodes are used to build Docker Swarm cluster, which are named as worker1, worker2 and worker3. The worker3 is indicated as the working node.

High Availability Test for Container Deployment Strategy Based on Microservice Level of IoT

On the container cluster of 5.1, SoapUI is used as a stress testing tool to verify the effectiveness of container initialization deployment strategy based on microservice level of IoT platform for maintaining high availability of IoT platform by comparing the availability of data forwarding service under normal and invalid container conditions.

SoaUI is used to test the pressure of 100 requests from concurrent threads within 30 seconds for data forwarding service. The average response time of the request is 2.07 milliseconds and the number of errors is 0, which means that the data forwarding service is in a high availability state under this stress test conditions.

In order to simulate the state of failure, a container of data forwarding service is stopped on purpose. Then, the container copy information of the data forwarding service is reduced from the original four container copies to three.

Again, SoapUI is used to test the pressure of 100 requests from concurrent threads within 30 seconds for data forwarding service. The average response time of the request is 2.36 milliseconds and the number of errors is 0, which means that the data forwarding service is still in a high availability state under this stress test conditions.

From those results, the data forwarding service can still quickly respond to the request in the same stress testing condition, although the original container is not available. And there is no problem that the service is not available. So it is verified that the container initialization deployment strategy based on microservice level of the IoT platform can effectively guarantee its high availability.

Conclusions

In order to ensure its high availability, this paper designs and implements a container initial deployment strategy of microservice level in IoT platform. The platform of microservice architecture is composed of a large number of microservices. It is difficult to manage the microservice and implement redundant backup in the deployment process. Therefore, in this paper, the microservice level model of the platform is proposed to manage the microservice hierarchically. And the
A calculation model of container redundancy is also proposed on the base of it. Different microservices are deployed to the container cluster with corresponding redundancy at a time, which effectively guarantees the high availability of the IoT platform.

Acknowledgement
This research was financially supported by the Characteristic Innovation Project (Natural Science) of Department of Education of Guangdong Province in 2017 (Subject No.: 2017KTSCX202); 2018 Annual Subject of the '13th Five-Year' Plan of Guangzhou Philosophical and Social Sciences (Subject No.: 2018GZGJ147).

References