

SERVICE MANAGEMENT METHOD AND THE DEVICE

Technical Field

The present invention relates to the field of network technologies, and in particular to a method and a device for service management.

Background

A distributed system is a computer system consisting of a plurality of interconnected processing resources, which implement the same task in a collaborative way as controlled by the entire system. The implementation of a task may involve different service units in a plurality of application systems, and information interaction is carried out among these small service units mostly through remote invocation. However, remote invocation will result in significant time consumption, and when a distributed system grows to a certain scale, this part of consumption will severely lower response efficiency of the system. According to relevant technologies, to save time consumed by remote invocation, an application analysis may be conducted and application systems having a relationship of service dependence can be deployed in a physically concentrated manner; however, such a way has relatively poor flexibility and adaptability, and analysis and deployment need to be performed again when there are changes to the applications, which increases the cost and affects the performance of services.

Summary

In light of the above, the present invention provides a method and a device for service management, so as to improve the response efficiency and adaptive capability of a distributed system.

Specifically, the present invention is implemented through the following technical solutions:

According to a first aspect, a method for service management is provided, comprising:
detecting popularity of use of a remote target service by an application;
when the popularity reaches a preset first threshold value, loading the target service to the application locally.

According to a second aspect, a device for service management is provided, comprising:

a popularity detection module configured to detect popularity of use of a remote target service by an application;

a migration control module configured to, when the popularity detection module detects that the popularity reaches a preset first threshold value, load the target service to the application locally.

With the method and the device for service management according to the present invention, when the popularity of a service reaches a preset threshold value, the service is downloaded to the application locally, which improves the response efficiency and adaptive capability of a distributed system relative to remote invocation.

Brief Description of the Drawings

Fig. 1 illustrates a distributed system architecture according to an embodiment of the present invention;

Fig. 2 is a flow chart of a method for service management according to an embodiment of the present invention;

Fig. 3 illustrates a structural diagram of a device for service management according to an embodiment of the present invention;

Fig. 4 illustrates a structural diagram of another device for service management according to an embodiment of the present invention.

Detailed Description of the Invention

Fig. 1 illustrates an optional distributed system architecture. The system comprises a plurality of processing resources, which may be, for example, a data storage server, a JAVA virtual machine, etc. Various application system services and data are distributed in the above processing resources. Several concepts used in the method for service management according to the present embodiment will be briefly described as follows:

Service and application: using a user opening an account at a bank as an example (it should be noted that this application is only an example), it is assumed that three steps are required for the entire account-opening process: 1. Verifying the user's I.D. information; 2. Verifying whether there are records of bad credit; 3. Setting up new account information. Then, each of "verifying I.D. information", "verifying credit records" and "setting up new account information" may be referred to as a "service"; the service of "verifying I.D. information" may probably be provided by an "application" system of a public security agency, while the services of "verifying credit records" and "setting up new account

information” are provided by an “application” system of the bank. With respect to the account opening request by the user, it can be simply described that an application system of the bank accepts the request and then invokes, remotely, the service provided by an application system of the public security agency to perform the first step of verification; subsequently, the second step and the third step are invoked locally to ultimately complete the entire account opening process. According to this solution, we can simply interpret “application” as a way to divide services and deployments of “services”, one “application” may provide one or more “services” to a user or other “applications”. In the process to implement a user request, the service dependence among application systems is referred to as “an application using service” or “an application invoking service” in the present embodiment.

Service accessing a data source: in the logic implementation process of a service, it typically involves data access; for example, the verifying I.D. information in the example above is assumed to use an I.D. database to query if the database stores the I.D. information to be verified, which involves the access to the data source, the “I.D. database”, by the service, “verifying I.D. information”. The service can be performed normally only when the data access is normal. As shown in Fig. 1, the data source may be from a database “Data Source”, a file system “File System”, or a high-speed cache “Cache”.

Local service and remote service: regarding local and remote, the two concepts may be interpreted in terms of their physical meanings, namely a service invoked by an application is a remote service when the service is not on the resource where the application is deployed, otherwise it is a local service; in the above example, if the application system of the bank is deployed on a processing resource A, while the application system of the public security agency is deployed on a processing resource B, then the service, “verifying I.D. information”, will be a remote service for the application system of the bank, and the other two services are local services as they belong to the application system of the bank. In other words, when the application system of the bank invokes the service, “setting up new account information”, the service logic is deployed on the processing resource A and can be directly invoked and executed on the resource A, making it a local service. On the other hand, the public security application system to which the invoked service, “verifying I.D. information”, belongs is deployed on another processing resource B, and consequently the application system of the bank needs to send, via a network, a service invocation request to the application on the processing resource B and wait for the return of the execution result by the service, “verifying I.D. information”, which is remote service invocation.

Apparently, remote service invocation will result in time consumption for network transmission and waiting. When an application frequently invokes a remote service, in particular, the execution efficiency of the application will be significantly affected; the method for service management in the present embodiment will be used to improve the response efficiency of a distributed system and reduce the time consumption of service invocation. In a conventional way, moreover, it is necessary to manually analyze services on a critical service pathway and services to be depended on, to deploy the application systems, in which these services are located physically nearby. As a result, the requirements for service analysis personnel are relatively high, and the complexity is increased for the deployment, operation, and maintenance of the entire system; when the critical service pathway changes or the dependence relationship changes, re-deployment is required, which will affect the service stability. Moreover, it is very inflexible and has poor adaptability. The method for service management in the present embodiment will further improve the adaptive capability of a distributed system to flexibly adapt to service invocation by applications. Relative to physically nearby deployment, moreover, the present embodiment lowers the granularity of migration to “service”, which will be favorable for lowering the equipment cost.

As shown in Fig. 1, the present embodiment adds a device for service management in the distributed system, the device comprising: a popularity detection module and a migration control module, wherein the device for service management can execute the flow of the method for service management shown in Fig. 2, comprising:

201. Detecting popularity of use of a remote target service by an application;

For example, in this step, it could be that the popularity detection module detects popularity of use of a remote target service by an application. It should be noted that the device for service management can be considered as a global management device, which can summarize and record a lot of information in actual implementation. Table 1 below illustratively lists some optional record information:

Table 1 Information of services invoked by applications

Application	Service used	Popularity of use	First (loading) threshold	Second (unloading) threshold	Can it be loaded	Loading version (loading	Remarks

			value	value		situation of device in this application	
Y1	S3	682	500	200	Yes	{192.168.10.1: loaded 1.0; 192.168.10.2: not loaded}	
Y1	S5	2315	2000	1000	No		Local data source does not meet requirements
Y2	S1	282	100	10	No		Service does not support migration
Y2	S2	150	200	100	No		Invocation popularity is not enough

Table 2 Services information

Service	Application to which it belongs	Can it be migrated	Dependence description (Dependence description of data sources, memory resources, CPU, etc.)	Current version
S1	Y1	No	None	1.0
S2	Y1	Yes	None	1.1
S3	Y2	Yes	None	1.0
S5	Y2	Yes	{MySQL: "192.168.12.1: 3306"; Memcache: "192.168.13.1: 12001"}	1.6

The information in Table 1 above is explained as follows: application and service have been described above and will not be described in detail again. The execution of a service may be dependent on a plurality of services for assistance. The method for service

management in the present embodiment involves service migration, but not every service can be migrated.

Migration of services: with reference to the example in Fig. 1, the application Y1 on the processing resource A needs to use the service S3, while the logic of the service S3 is stored on the processing resource B. If the application Y1 invokes the service S3 on the processing resource B, it is a remote invocation of the service. The service S3 may be migrated from the processing resource B to the processing resource A, wherein the migration refers to the migration of the service logic. As mentioned above, the execution of a service may probably involve the access to a data source. Then it can be considered that a complete execution of a service comprises two parts, “logic” and “data”. The service migration in the present embodiment refers to the migration of the logic, and a copy of a segment of the logic of a target service is migrated. Namely, the logic of the service S3 is stored on the processing resource A at the same time, such that the processing resource A also stores a copy of the logic of the service S3. Consequently, the logic flow of the service S3 may be executed locally at the processing resource A. In addition, it should be noted that the service S3 belongs to Y2 and is merely used by Y1. Even if migration (migration is a process of copying the service logic, and a copy of the service logic is migrated) takes place, the service S3 still belongs to Y2, which does not change.

Migratable service: as mentioned above, the migration of a service refers to the migration of logic only, i.e. the logic of the processing flow defined by the service. If the service is migrated to an application (migrated to the processing resource corresponding to the application), therefore, it is required that the application can access resources on which normal execution of the service is dependent. For example, the normal execution of the service S5 needs to access a data source (MySQL and Memcache), but the application Y1 cannot access that data source. Then, the service S5 cannot be executed normally after the service S5 is migrated to the application Y1, and the service S5 is not migratable with respect to the application Y1.

For a service to be a migratable service with respect to an application, it is required that the application can access a data source needed by the service, and certainly, other conditions may be imposed. For example, the service is an independent service with a clear logic boundary and does not have complex mutual dependence with other services, and moreover, the service is complete and small, such that it can be easily migrated. It may be further required that the service is a no-state service with no state of mutual dependence. All these are to make a service easy to be migrated. Furthermore, whether the accessibility of a

service is affected after the service is migrated to an application could be taken into account. For example, a service has high efficiency when accessing a data source before migration, but the execution efficiency of the service becomes lower, due to various reasons after the migration, to such a degree that the service requirement of the application is no longer met. As a result, the service would be deemed not suitable for migration. Furthermore, it should be noted that a migratable service is usually a remote service to an application, as migration is not necessary for local services.

A distributed system may comprise many applications. In this step, the device for service management can globally manage what services are migratable to what applications, manage data sources that each application can access, and unify the names of the data sources to facilitate determining whether an application comprises a data source required by a service. In addition, the device for service management can further summarize information like the popularity of use of a migratable service by applications, whether the service has been loaded to an application, and whether the service has been updated.

Popularity of use of a migratable service by applications: the “popularity” is mainly used to indicate the frequency of use of the service by applications. For example, the service is invoked by applications 50 times within one hour; certainly, what type of metric is specifically used to indicate the popularity may be flexibly determined according to actual situation. In the present embodiment, a first threshold value and a second threshold value may be set for the popularity, the first threshold value is greater than the second threshold value, and the first threshold value is mainly used to indicate, when the popularity value reaches or is higher than the threshold value, that the use of the service by the application is very frequent and has reached the tolerance limit at which the execution efficiency of the application is affected. Service migration could be performed, namely, the service is loaded to the application; the second threshold value is mainly used to indicate, when the popularity value reaches or is lower than the threshold value, that the application currently rarely uses the service, and service unloading may be performed to save processing resources.

Furthermore, to avoid frequent loading and unloading, an intermediate value may be set between the first threshold value and the second threshold value above. For example, the first threshold value is 100 times, the second threshold value is 20 times, and an intermediate value of 80 times is set between these two threshold values. Such a practice will prevent overly frequent loading or unloading of the service that causes the system unstable.

Loading of a service to an application: the loading of a service to an application refers to that a remote service has been migrated to an application locally. For example, as shown in

Fig. 1, the service S3 in the processing resource B is migrated to the processing resource A where the application Y1 is located, which indicates that the service S3 has been loaded to the application Y1.

Service update: the logic of a service in a distributed system may produce, for example, different versions of the service; the device for service management in the present embodiment may monitor whether a service has been updated; furthermore, to reduce the resource consumption by the device for service management, it may only monitor remote services that have been loaded to applications. That is because the goal of monitor service update is as follows: if a service loaded to an application has been updated, then the current service on the application is an old service that cannot be used, and it is necessary to promptly unload the old service and re-load the new service. As a result, the device for service management may only monitor the state of services that have been loaded to applications.

In a specific implementation, a state monitoring module may be added to the device for service management for monitoring whether a target service loaded to an application has been updated. Referring to Fig. 3, in such a case, the device for service management may comprise: a popularity detection module 31, a migration control module 32, and a state monitoring module 33.

In this step, the popularity detection module of the device for service management will detect popularity of use of a remote target service by an application, and specifically, monitor popularity of use of a remote service by each application, and proceed to 202 according to the detected popularity.

202. When the popularity reaches a preset first threshold value, loading the target service to the application locally.

For example, in this step, when the popularity detection module detects that the popularity of use of the target service by the application reaches a preset first threshold value, the migration control module loads the target service to the application locally. Optionally, it is assumed that the popularity of use of the service S3 on the processing resource B by the application Y1 on the processing resource A has reached the first threshold value, and then the migration control module can migrate the service S3 from the processing resource B to the processing resource A. Moreover, the migration control module may be located on the processing resource A, or may be a module dedicated to loading or unloading remote services. In other words, two modules may be provided on the processing resource A, one of which is a module dedicated to managing local services, and the other one is a module dedicated to

loading or unloading remote services (e.g. the remote class loader on the JAVA virtual machine JVM). Such arrangement makes it easier to manage remote services. When there is a need to load or unload a remote service, the migration control module just needs to be notified by the popularity detection module.

In addition, the loading or unloading of services in the present embodiment may be performed asynchronously with the application. For example, an application is currently in the process of remote invocation of a service, but it has been detected that the popularity of use of the service by the application reaches the first threshold value. Then, the migration control module performs migration of the service, but the migration does not affect the current execution by the application. The application is still in the remote service invocation mode for this time, and will only use the service loaded locally in the next execution by the application.

Furthermore, as mentioned above, the popularity detection module continues detection of the popularity of use of the target service by the application after the target service has been loaded to the application locally. If it is detected that the popularity reaches the second threshold value, namely the application subsequently has a relatively low frequency of use of the target service, the migration control module may unload the target service from the application locally, so as to save the processing resources.

Furthermore, after the target service has been loaded to the application locally, the state monitoring module of the device for service management may notify the migration control module upon detection of an update to the target service, such that the migration control module unloads the target service on the application locally, and loads the updated target service.

After a remote service has been loaded to the application locally, the device for service management may further comprise a service invocation module 34. As shown in Fig. 4, when used to execute an application, the service invocation module 34 determines whether the application locally stores the loaded target service; if the determination result is that the target service is stored locally, it is preferable to use the local target service to execute the application to reduce time consumption by remote invocation; if the determination result is that the target service is not stored locally, regular remote invocation is used.

It can be seen from the embodiment above that, by adding a device for service management into a distributed system, the device can implement the method for service management in the embodiment above. When the popularity of use of a remote target service by an application reaches a frequency threshold value, the remote service is migrated to the

application locally. Relative to conventional ways, such arrangement allows the device for service management to automatically determine the use frequency of a service and perform automatic service migration, which saves the cost for manual analysis and deployment, and improves the service deployment efficiency. Such arrangement of automatic service determination and migration has better flexibility and can quickly adapt to the actual situation of application execution. By migrating a remote service to an application locally, it also avoids the time consumption by remote invocation and improves the response efficiency of the distributed system. In addition, the method in the present embodiment uses service as the granularity of analysis, which achieves more precise migration granularity and further improves the processing efficiency relative to the conventional centralized deployment with an application granularity.

The above description is only a preferred embodiment of the present invention, which is not used to limit the present invention. Any modification, equivalent substitution, or improvement made within the spirit and principle of the present invention shall be encompassed by the present invention.

Claims

1. A method for service management, wherein the method comprises:
 - detecting popularity of use of a remote target service by an application;
 - when the popularity reaches a preset first threshold value, loading the target service to the application locally;
 - wherein, before the loading the target service to the application locally, the method further comprises:
 - determining that the target service meets migration conditions, the migration conditions comprising: the application comprises a data source required by the target service, and the target service is an independent service.
2. The method according to claim 1, wherein, after the loading the target service to the application locally, the method further comprises:
 - unloading the target service from the application locally when the popularity reaches a preset second threshold value.
3. The method according to claim 1, wherein, after the loading the target service to the application locally, the method further comprises:
 - unloading the target service from the application locally when it is determined that the target service is updated.
4. The method according to claim 1, wherein, after the loading the target service to the application locally, the method further comprises:
 - when executing the application, determining whether the application locally stores the loaded target service;
 - if the determination result is that the target service is stored locally, using the target service to execute the application.
5. A device for service management, wherein the device comprises:
 - a popularity detection module configured to detect popularity of use of a remote target service by an application;
 - a migration control module configured to, when the popularity detection module detects that the popularity reaches a preset first threshold value, load the target service to the application locally;
 - wherein the popularity detection module is further configured to determine that

the target service meets migration conditions, the migration conditions comprising: the application comprises a data source required by the target service, and the target service is an independent service.

6. The device according to claim 5, wherein
the migration control module is further configured to unload the target service from the application locally when the popularity detection module detects that the popularity reaches a preset second threshold value.

7. The device according to claim 5, wherein the device further comprises a state monitoring module;
the state monitoring module is further configured to monitor updates to the target service; and
the migration control module is further configured to unload the target service from the application locally when the state monitoring module determines that the target service is updated.

8. The device according to claim 5, wherein the device further comprises:
a service invocation module configured to determine, when executing the application, whether the application locally stores the loaded target service; if the determination result is that the target service is stored locally, use the target service to execute the application.