Dubbo 基础教程

一、前言

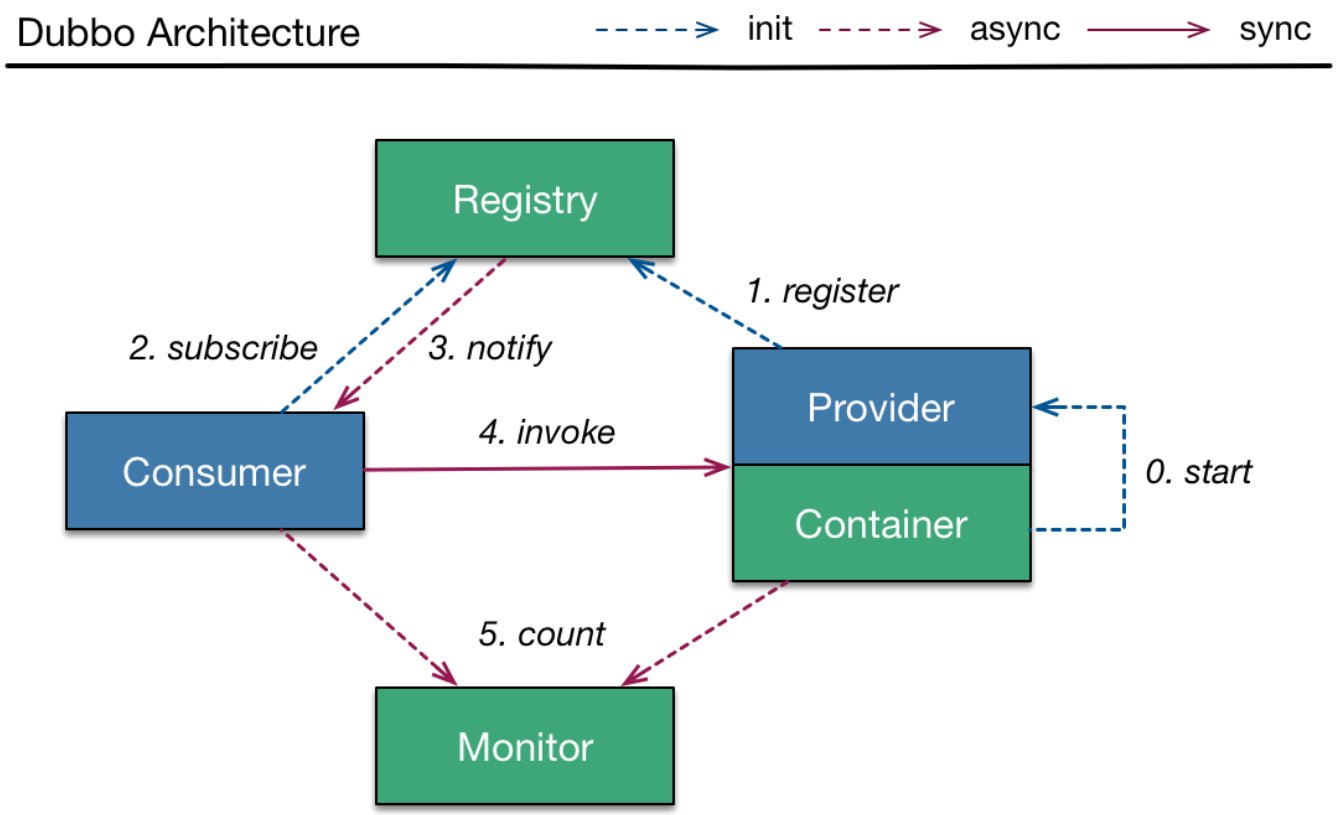
当服务越来越多时，容量的评估，小服务资源的浪费等问题逐渐显现，此时需要增加一个调度中心基于访问压力实时管理集群容量，提供集群利用率。其中，用于提高机器利用率的资源调度和治理中心是关键。

二、Dubbo 简介

2.1 概念

Dubbo 是阿里巴巴开源项目的一个分布式服务框架。其致力于提供高性能和透明化的 RPC 远程调用方案，以及 SOA 服务治理方案。

2.2 原理



调用关系说明：

> 1）服务容器启动、加载和运行服务提供者；
> 2）服务提供者在启动时，向注册中心注册自己提供的服务；
> 3）服务消费者在启动时，向注册中心订阅自己所需的服务；
> 4）注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更给消费者；
> 5）服务消费者从地址列表中，基于软负载均衡算法选一台服务提供者进行调用，如果调用失败再选另一台；
> 6）服务消费者和服务提供者在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

节点角色说明

节点	角色说明
Container	服务运行容器
Provider	暴露服务的服务提供者
Consumer	调用远程服务的服务消费者
Registry	服务注册与发现的注册中心
Monitor	统计服务的调用此处和调用时间的监控中心

三、快速入门

Dubbo 采用全 Spring 配置方式，透明化接入应用，对应用没有任何 API 入侵，只需用 Spring 加载 Dubbo 配置即可。

3.1 安装注册中心

官方推荐使用 **Zookeeper** 作为注册中心，因此本次测试使用 **Zookeeper**，将其放置在 **ip** 为 **192.168.2.14** 的虚拟机上。

```
# 解压和转移目录
tar -zxvf zookeeper-3.4.8.tar.gz -C /usr/
cd /usr
mv zookeeper-3.4.8 zookeeper

# 设置配置文件
cd /usr/zookeeper/conf
cp zoo_sample.cfg zoo.cfg

# 启动 zookeeper
/usr/zookeeper/bin/zkServer.sh start

# 查看 zookeeper 运行状态，如果出现 Mode: standalone 说明运行成功
/usr/zookeeper/bin/zkServer.sh status
```

3.2 服务提供者

创建一个 Maven 项目（名为 dubbo-service 的 web 项目）。

pom.xml 配置：

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.light</groupId>
  <artifactId>dubbo-service</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>4.3.10.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>dubbo</artifactId>
        <version>2.6.0</version>
    </dependency>
    <dependency>
        <groupId>com.101tec</groupId>
        <artifactId>zkclient</artifactId>
        <version>0.9</version>
    </dependency>
  </dependencies>
</project>
```

web.xml 配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">

    <display-name>dubbo-service</display-name>

    <!-- spring容器 start -->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext-dubbo.xml</param-value>
    </context-param>
    <!-- spring容器 end -->

</web-app>
```

接口：

```
public interface HelloService {

    String sayHello(String name);
}
```

实现类：

```
public class HelloServiceImpl implements HelloService {

    @Override
    public String sayHello(String name) {
        return "Hello," + name;
    }

}
```

applicationContext-dubbo.xml 配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://code.alibabatech.com/schema/dubbo
       http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

    <dubbo:application name="hello-demo"/>

    <dubbo:registry address="zookeeper://192.168.2.14:2181"/>

    <dubbo:protocol name="dubbo" port="20880"/>

    <dubbo:service interface="com.light.dubbo.service.HelloService"
ref="helloService"/>

    <bean id="helloService"
class="com.light.dubbo.service.impl.HelloServiceImpl"/>

</beans>
```

3.3 服务消费者

创建一个 Maven 项目（名为 dubbo-consumer 的 web 项目）。

pom.xml 配置：

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.light</groupId>
    <artifactId>dubbo-consumer</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>4.3.10.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>4.3.10.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.9.3</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>dubbo</artifactId>
            <version>2.6.0</version>
        </dependency>
        <dependency>
            <groupId>com.101tec</groupId>
            <artifactId>zkclient</artifactId>
            <version>0.9</version>
        </dependency>
    </dependencies>
</project>
```

web.xml 配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">

    <display-name>dubbo-consumer</display-name>

    <!-- spring容器 start -->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext-dubbo.xml</param-value>
    </context-param>
    <!-- spring容器 end -->

    <!-- springmvc容器 start -->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:springmvc.xml</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
    <!-- springmvc容器 end -->

</web-app>
```

将 **dubbo-service** 项目中的 **HelloService** 接口复制到该项目（**dubbo-consumer**）中。

控制层：

```
@Controller
public class HelloController {

    @Autowired
    private HelloService helloService;

    @RequestMapping("hello")
    @ResponseBody
    public String hello(String name) {
        return this.helloService.sayHello(name);
    }
}
```

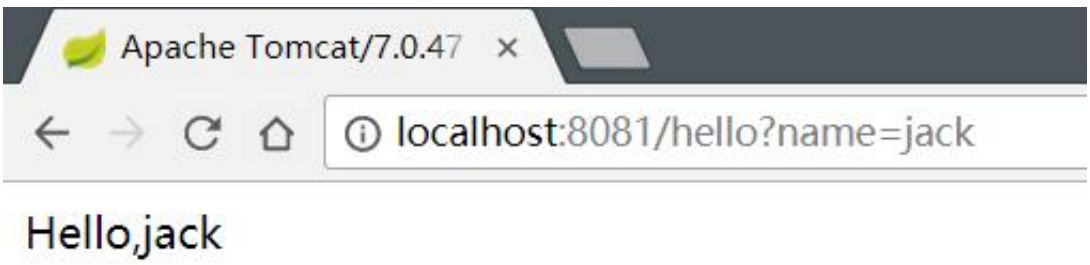
applicationContext-dubbo.xml 配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://code.alibabatech.com/schema/dubbo
       http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

    <dubbo:application name="hello-demo"/>

    <dubbo:registry address="zookeeper://192.168.2.14:2181"/>

    <dubbo:protocol name="dubbo" port="20880"/>

    <dubbo:reference interface="com.light.dubbo.service.HelloService"/>


</beans>
```

springmvc.xml 配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
                        http://www.springframework.org/schema/mvc
                        http://www.springframework.org/schema/mvc/spring-
mvc-4.0.xsd
                        http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-4.0.xsd">

      <!-- 只扫描含有@Controller注解的类 -->
      <context:component-scan base-package="com.light.dubbo.controller">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller" />
      </context:component-scan>

      <!-- 加载解析 @rquestMapping等注解的解析器 -->
      <mvc:annotation-driven/>

</beans>
```

先启动服务提供者的项目（8080），再启动服务消费者的项目（8081）。打开浏览器访问http://localhost:8081/hello?name=jack,结果如下图：



四、监控

4.1 获取源码

```
git clone --branch dubbo-2.6.0 https://github.com/alibaba/dubbo.git
```

下载完成后使用 IDE 工具引入其子项目 dubbo-sample\dubbo-monitor-sample 进行编译和打包。打包后会在项目的 target 目录下生成 dubbo-monitor-simple-2.6.0-assembly.tar.gz 压缩文件。

4.2 修改配置

1. 解压 dubbo-monitor-simple-2.6.0-assembly.tar.gz 压缩包，修改 dubbo-monitor-simple-2.6.0\conf\dubbo.properties：

```
dubbo.registry.address=zookeeper://192.168.2.14:2181
```

1. 在服务提供者的配置文件中添加：

```
<!-- 注册中心自动查找监控服务 -->
<dubbo:monitor protocol="registry"/>
```

最后启动 dubbo-monitor-simple-2.6.0\bin\start.bat。打开浏览器访问 http://localhost:8080/，效果图如下：

!\](http://images.extlight.com/dubbo-03.jpg)

五、管理控制台

Dubbo 提供了一套在线管理服务的管理控制台，该管理控制台为阿里巴巴内部裁减版本，开源部分主要包含：路由规则、动态配置、服务降级、访问控制、权重调整和负载均衡。

5.1 获取运行项目

在第四节下载的 duboo 源码中，通过 IDE 工具引入其子项目 dubbo-admin 进行编译和打包。 打包后会在项目的 target 目录下生成 dubbo-admin-2.6.0.war 压缩文件。

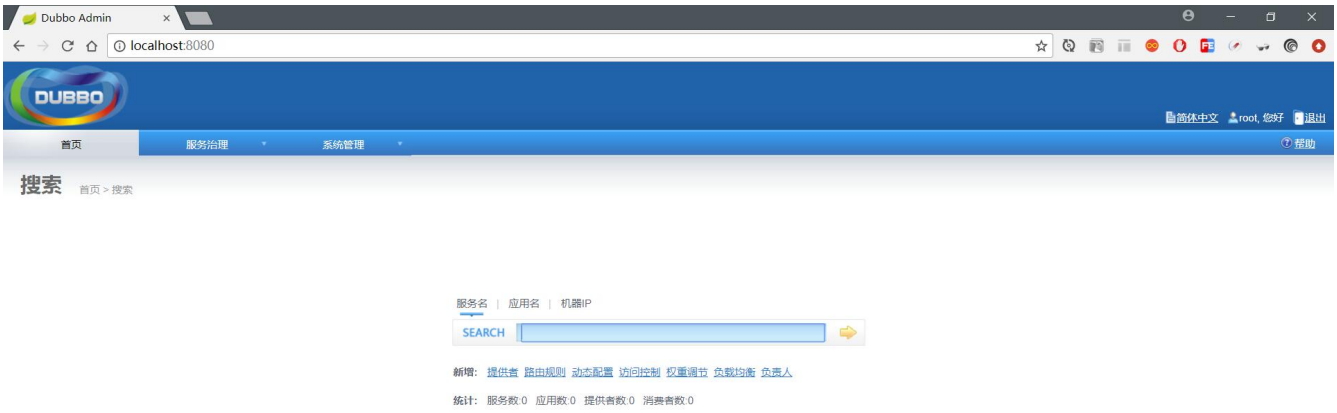
5.2 修改配置

将 dubbo-admin-2.6.0.war 里边的文件和文件夹复制粘贴到 tomcat 的 ROOT 目录中并修改 webapps\ROOT\WEB-INF\dubbo.properties 文件内容：

```
dubbo.registry.address=zookeeper://192.168.2.14:2181
dubbo.admin.root.password=root
dubbo.admin.guest.password=guest
```

其中，配置中设置 2 个用户：root 和 guest。

最后启动 tomcat 容器，打开浏览器访问http://localhost:8080/，页面要求输入账号和密码，登录后效果图如下：



六、参考资料

Dubbo 官网