Dubbo

Not Given

2022 年 7 月 24 日

目录

第一章：分布式系统中的相关概念 **1**
一、传统项目与互联网项目 1
二、集群与分布式 1
三、架构的演变 2

第二章：Dubbo 的简介与基本使用 **3**
一、简介 3
二、ZooKeeper 3
三、基本使用方式 3

第三章：Dubbo 的高级特性 **5**
一、Dubbo-admin 5
二、序列化 5
三、地址缓存 6
四、超时与重试 6
五、多版本 7
六、负载均衡 7
七、集群容错 8
八、服务降级 8

第一章：分布式系统中的相关概念

一、传统项目与互联网项目

1. 传统项目：公司内部系统，主要使用群体为公司员工
2. 互联网项目：微信、百度等，使用群体为网民
3. 互联网项目面对的挑战：用户基数大，高并发，用户忍耐力低
4. 互联网项目的设计目标

- **高性能**：提供快速的访问体验。
- **高可用**：网站服务一直可以正常访问。
- **可伸缩**：通过硬件增加/减少，提高/降低处理能力。
- **高可扩展**：系统间耦合低，方便的通过新增/移除方式，增加/减少新的功能/模块。
- **安全性**：提供网站安全访问和数据加密，安全存储等策略。
- **敏捷性**：随需应变，快速响应。

5. 衡量网站的性能指标

衡量网站的性能指标：

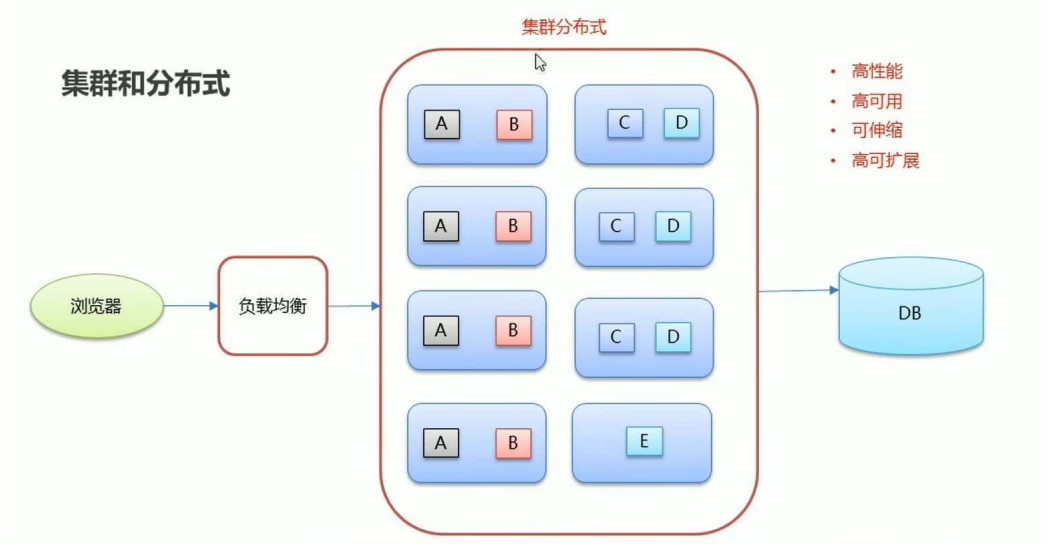
- **响应时间**：指执行一个请求从开始到最后收到响应数据所花费的总体时间。
- **并发数**：指系统同时能处理的请求数量。
  - **并发连接数**：指的是客户端向服务器发起请求，并建立了TCP连接。每秒钟服务器连接的总TCP数量
  - **请求数**：也称为QPS(Query Per Second) 指每秒多少请求.
  - **并发用户数**：单位时间内有多少用户
- **吞吐量**：指单位时间内系统能处理的请求数量。
  - **QPS**：Query Per Second 每秒查询数。
  - **TPS**：Transactions Per Second 每秒事务数。
  - 一个事务是指一个客户机向服务器发送请求然后服务器做出反应的过程。客户机在发送请求时开始计时，收到服务器响应后结束计时，以此来计算使用的时间和完成的事务个数。

二、集群与分布式

1. 基本概念

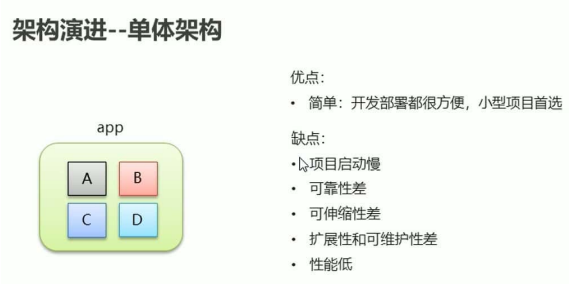
- 集群：很多“人”一起，干一样的事。
- 分布式：很多“人”一起，干不一样的事。这些不一样的事，合起来是一件大事。

2. 实例

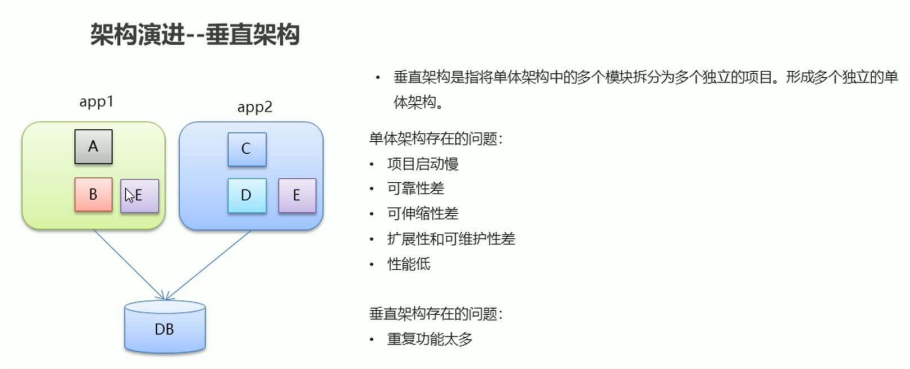


三、架构的演变

1. 单体架构



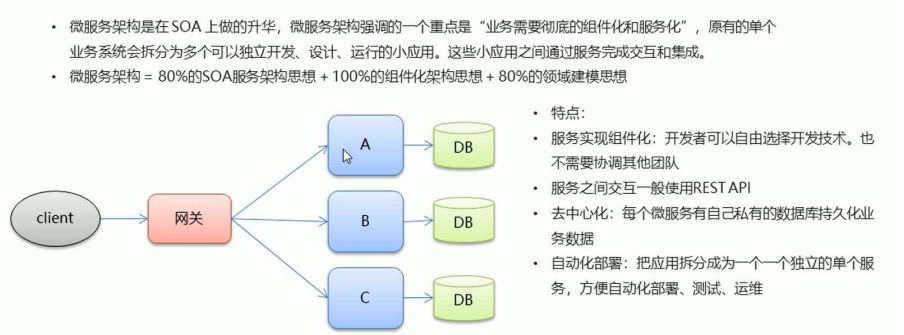
2. 垂直架构



3. 分布式架构

4. SOA 架构

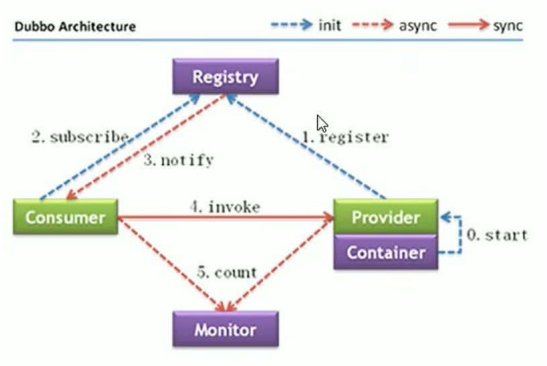
5. 微服务架构



第二章：Dubbo 的简介与基本使用

一、简介

1. 概念：一款高性能、轻量级的 RPC 开发框架
2. 基本架构

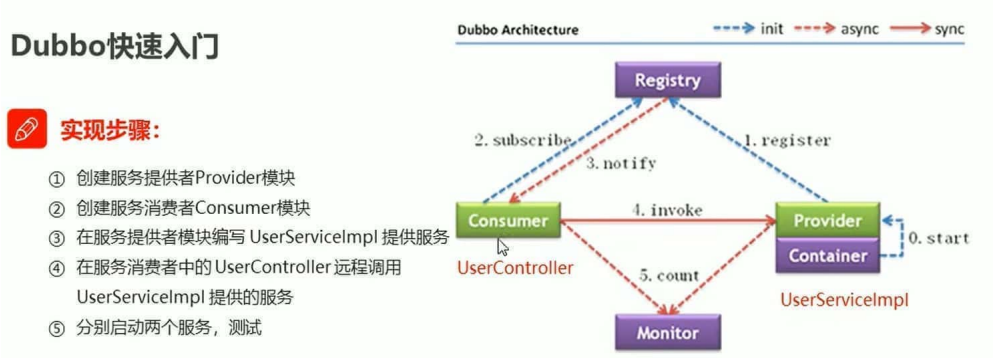


二、ZooKeeper

1. 概念：一种供 Dubbo 使用的注册中心
2. 安装方式：`https://www.bilibili.com/video/BV1VE411q7dX?p=7`

三、基本使用方式

1. 实现步骤



2. 创建项目：用 Spring 和 SpringMVC 创建两个项目，分别作为 Service 模块和 Web 模块，其中 Service 可以调用 Web 模块。但是这时项目还不算分布式的，因为 Service 模块不能独立运行
3. 配置 Service 模块

   (1) 导入坐标

```
<!--Dubbo的起步依赖，版本2.7之后统一为rg.apache.dubb -->
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo</artifactId>
    <version>${dubbo.version}</version>
</dependency>
<!--ZooKeeper客户端实现 -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
    <version>${zookeeper.version}</version>
</dependency>
<!--ZooKeeper客户端实现 -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-recipes</artifactId>
    <version>${zookeeper.version}</version>
</dependency>
```

   (2) 设置方法注解：目的为将提供服务的类对外发布

```
@Service//将这个类提供的方法（服务）对外发布。将访问的地址 ip，端口，路径注册到注册中心中
public class UserServiceImpl implements UserService {

    public String sayHello() { return "hello dubbo!~"; }
}
```

(3) Dubbo 配置

```
<!--dubbo的配置-->
<!--1.配置项目的名称,唯一-->
<dubbo:application name="dubbo-service"/>
<!--2.配置注册中心的地址-->
<dubbo:registry address="zookeeper://192.168.149.135:2181"/>
<!--3.配置dubbo包扫描-->
<dubbo:annotation package="com.itheima.service.impl" />
```

4. 配置 Web 模块

(1) 导入坐标

```
<!--Dubbo的起步依赖，版本2.7之后统一为rg.apache.dubb -->
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo</artifactId>
    <version>${dubbo.version}</version>
</dependency>
<!--ZooKeeper客户端实现 -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
    <version>${zookeeper.version}</version>
</dependency>
<!--ZooKeeper客户端实现 -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-recipes</artifactId>
    <version>${zookeeper.version}</version>
</dependency>
```

(2) 设置方法注解：目的为进行远程调用

```
//注入Service
//@Autowired//本地注入

/*
    1. 从zookeeper注册中心获取userService的访问url
    2. 进行远程调用RPC
    3. 将结果封装为一个代理对象。给变量赋值

 */

@Reference//远程注入
private UserService userService;


@RequestMapping("/sayHello")
public String sayHello(){
    return userService.sayHello();
}
```

(3) Dubbo 配置

```
<!--dubbo的配置-->
<!--1.配置项目的名称,唯一-->
<dubbo:application name="dubbo-service"/>
<!--2.配置注册中心的地址-->
<dubbo:registry address="zookeeper://192.168.149.135:2181"/>
<!--3.配置dubbo包扫描-->
<dubbo:annotation package="com.itheima.service.impl" />
```

5. 配置公共接口：在两个模块的 pom.xml 文件中分别进行配置，让它们都依赖于公共接口模块，这样可以避免两个项目引入的接口不一致

```
<!--依赖公共的接口模块-->
<dependency>
    <groupId>com.itheima</groupId>
    <artifactId>dubbo-interface</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

第三章：Dubbo 的高级特性

一、Dubbo-admin

1. 概念：一种图形化的服务管理平台

   - dubbo-admin管理平台，是图形化的服务管理页面
   - 从注册中心中获取到所有的提供者/消费者进行配置管理
   - 路由规则、动态配置、服务降级、访问控制、权重调整、负载均衡等管理功能
   - dubbo-admin是一个前后端分离的项目。前端使用vue，后端使用springboot
   - 安装 dubbo-admin其实就是部署该项目

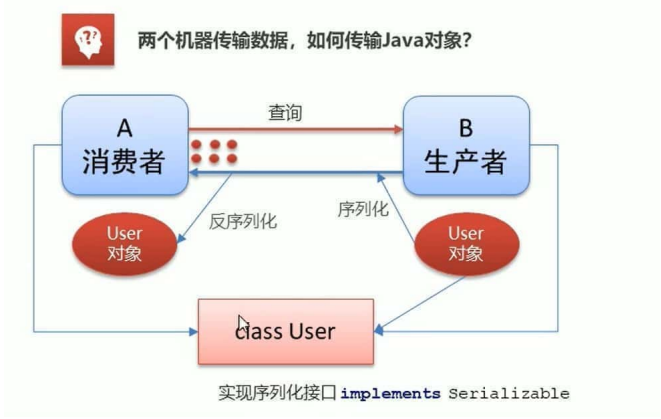
2. 安装方式：`https://www.bilibili.com/video/BV1VE411q7dX?p=11`

3. 基本使用：目前只有查询功能

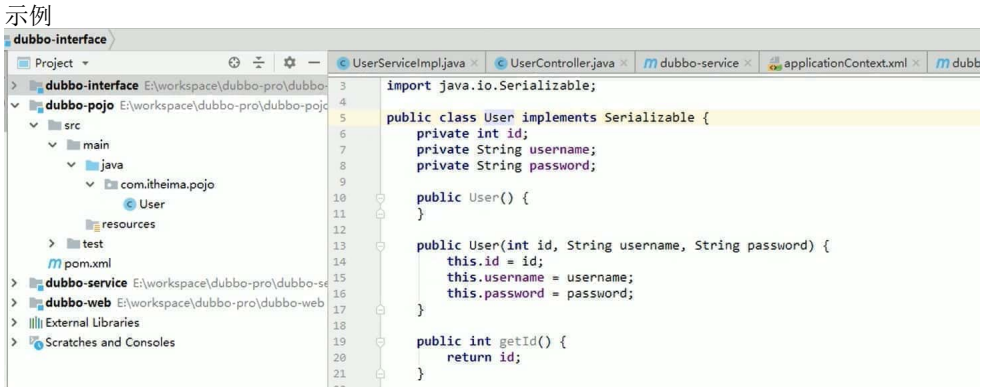
二、序列化

1. 概念：将 java 对象转化为流的数据，然后进行传输

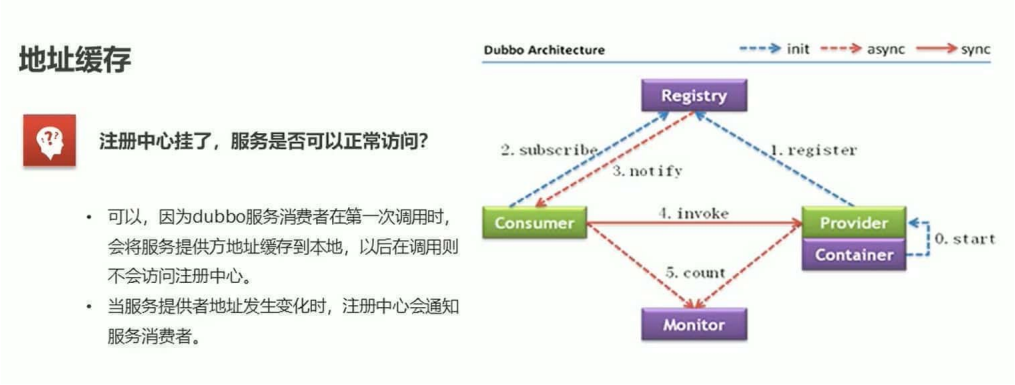
2. Dubbo 中的序列化：用于将实体类的数据在不同模块间传输



3. 示例

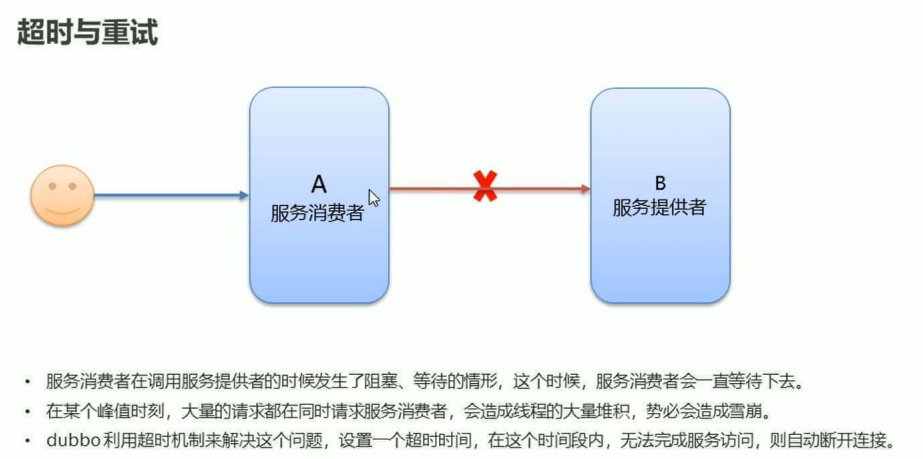


三、地址缓存

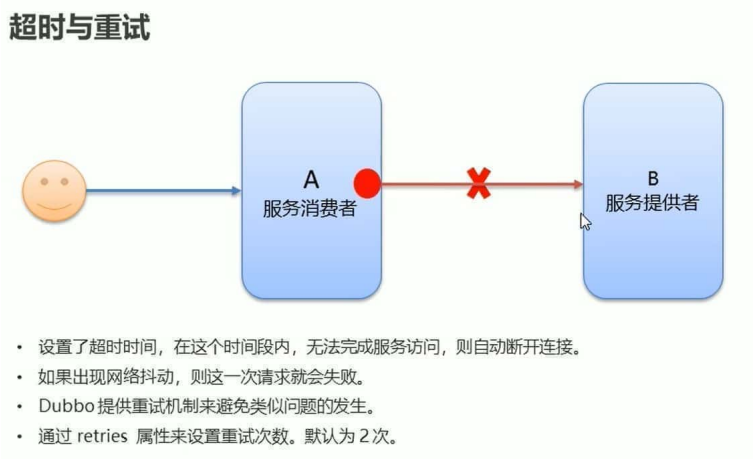


四、超时与重试

1. 超时机制：在一定时间内请求不到服务，端口会被释放。一般在 Service 中配置



2. 重试机制：在网络抖动或超时后重新发送请求

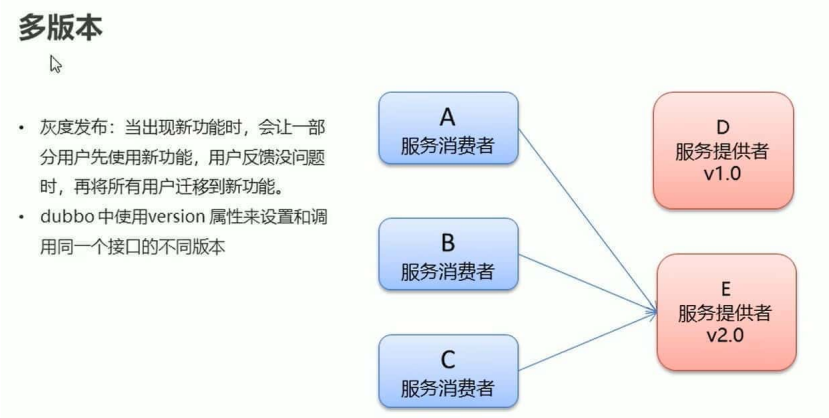


3. 示例

```
//@Service//将该类的对象创建出来，放到Spring的IOC容器中   bean定义
//将这个类提供的方法（服务）对外发布。将访问的地址 ip，端口，路径注册到注册中心中
@Service(timeout = 3000,retries = 0)//当前服务3秒超时
public class UserServiceImpl implements UserService {

    public String sayHello() { return "hello dubbo hello!~"; }


    public User findUserById(int id) {
        //查询User对象
        User user = new User( id: 1, username: "zhangsan", password: "123");

        return user;
    }
}
```

五、多版本

1. 概念：设置调用统一接口的不同版本



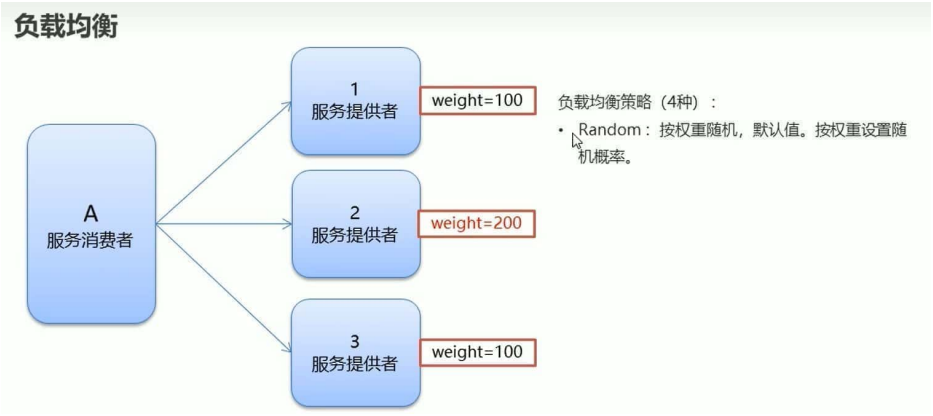
2. 示例

```
//@Service//将该类的对象创建出来，放到Spring的IOC容器中   bean定义
//将这个类提供的方法（服务）对外发布。将访问的地址 ip，端口，路径注册到注册中心中
@Service(version = "v2.0")
public class UserServiceImpl2 implements UserService {

    public String sayHello() { return "hello dubbo hello!~"; }


    public User findUserById(int id) {
        System.out.println("new....");
        //查询User对象
        User user = new User( id: 1, username: "zhangsan", password: "123");

        return user;
    }
}
```

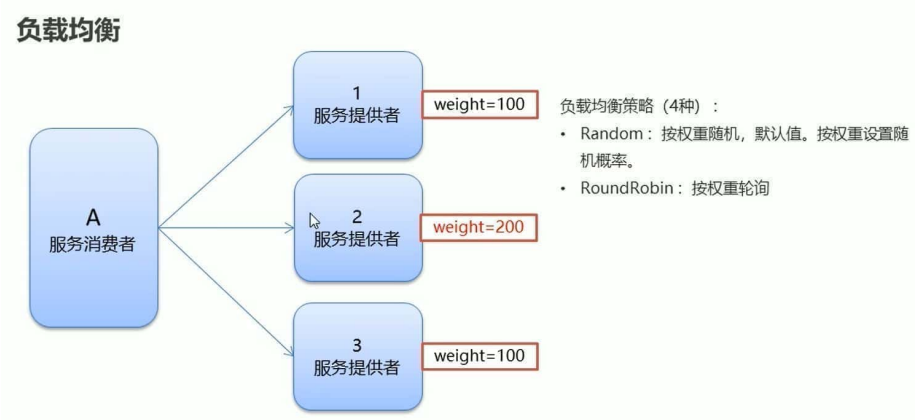
```
@Reference(version = "v2.0")//远程注入
private UserService userService;
```

六、负载均衡

1. 概念：将服务请求进行分流
2. Random



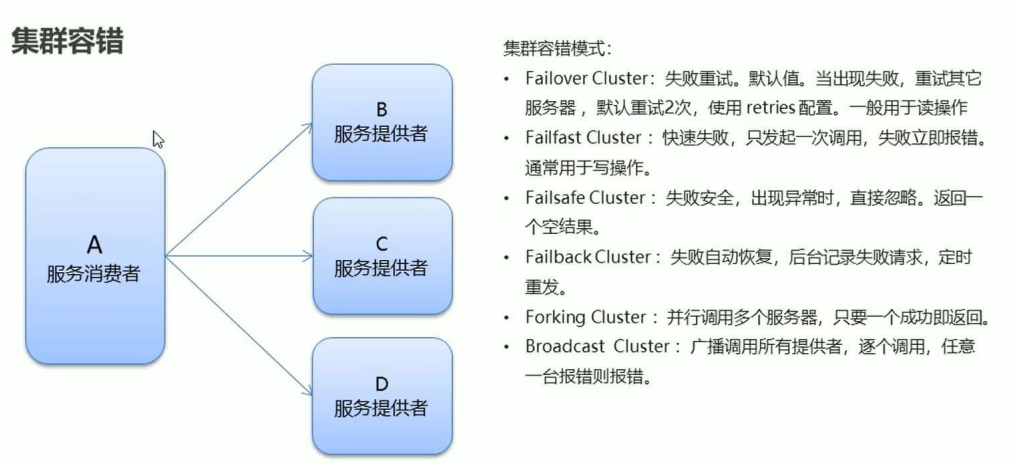
3. RoundRobin



七、集群容错

1. 概念：出错后的应对策略

2. 分类



八、服务降级

1. 概念：当负载过度时，将不重要的服务关掉，释放资源以保证主要服务可以正常运行

2. 分类