<https://mp.weixin.qq.com/s/jveqCUAiqKdKRoea9wESbw>

**长文图解七种负载均衡策略**

负载均衡、集群容错、服务降级这三个概念在DUBBO中非常重要，同理其它分布式框架也都有相同或者相近之概念。

从调用顺序角度分析，调用顺序依次是负载均衡、集群容错、服务降级。从解决问题角度分析，负载均衡解决了「选哪一个」问题，集群容错解决了「换哪一个」问题，服务降级解决了「全错怎么办」问题。

假设有1个服务消费者面对10个提供者，这时面临第一个问题就是「选哪一个」进行调用，所以负载均衡最先调用，假设选定了5号服务提供者进行服务调用。

假设消费者调用5号提供者发生了超时异常，这时面临第二个问题就是「换哪一个」进行调用：5号超时要不要换1号试一试，或者直接返回不进行重试，所以集群容错第二个调用。

假设已经重试了1号、3号、6号提供者全部超时，这时面临「全错怎么办」这第三个问题，这时可以直接返回一个固定值或者提示文案，所以服务降级第三个调用。

负载均衡作为整个链路第一个节点非常重要，本文结合DUBBO源码分析以下七种负载均衡策略：

* 简单随机
* 加权随机
* 简单轮询
* 简单加权轮询
* 平滑加权轮询
* 一致性哈希
* 最少活跃数

**2 简单随机**

简单随机含义是服务消费者每次会任意访问一个服务提供者，并且从概率角度看每个提供者被访问概率一致，可以通过指定范围随机数实现。

**3 加权随机**

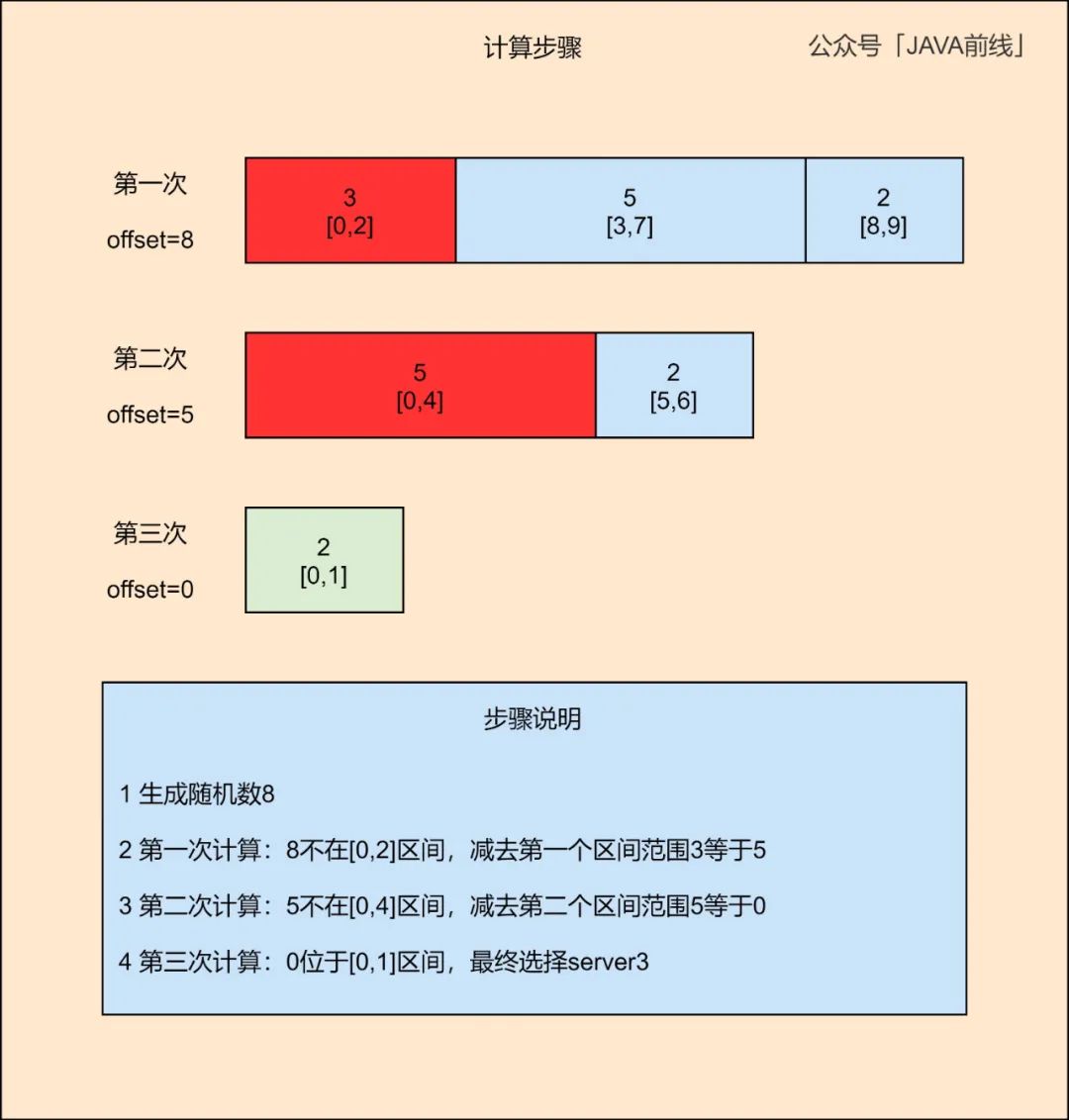
**3.1 设计思路**

加权随机新增了权重概念，假设服务器A权重等于1，服务器B权重等于5，从概率角度看B服务器被访问概率5倍于A服务器。实现按照权重访问有很多种方式，我们选择使用概率区间这个思路。

假设现在有三台服务器，服务器权重分别是3、5、2，那么三者构成如下图概率区间：



假设随机数等于8，其位于[8,9]区间，所以选择server3，下图说明了概率区间计算步骤：



# 4 简单轮询

简单轮询含义是服务消费者每次会依次访问一个服务提供者，并且从概率角度看每个提供者被访问概率一致，可以通过原子变量累加实现。

# 5 简单加权轮询

简单加权轮询新增了权重概念，假设服务器A权重等于1，服务器B权重等于5，从概率角度看B服务器被访问概率5倍于A服务器，我们还是使用概率区间这个思路，相较于加权随机会有一些变化。

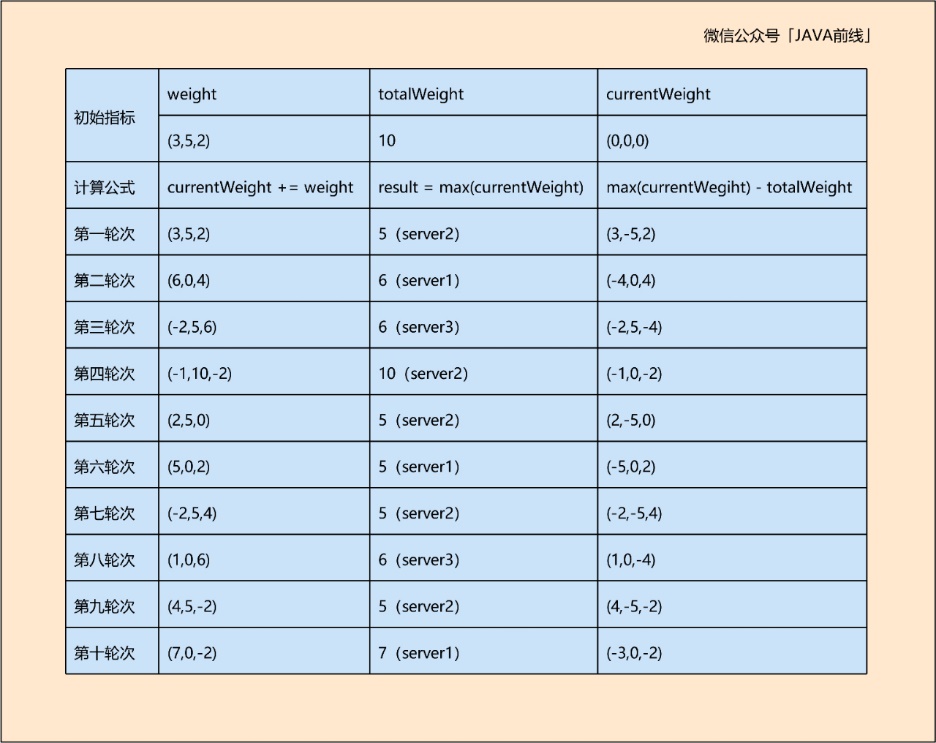


# 6 平滑加权轮询

## 6.1 设计思路

简单加权轮询有什么问题？我们分析其输出结果发现，连续3次访问服务器1，连续5次访问服务器2，连续2次访问服务器3，所以简单加权轮询策略会导致请求集中问题。

所以需要使用平滑加权轮询策略，将请求比较均匀地分散至各个服务器，下图说明了计算步骤：



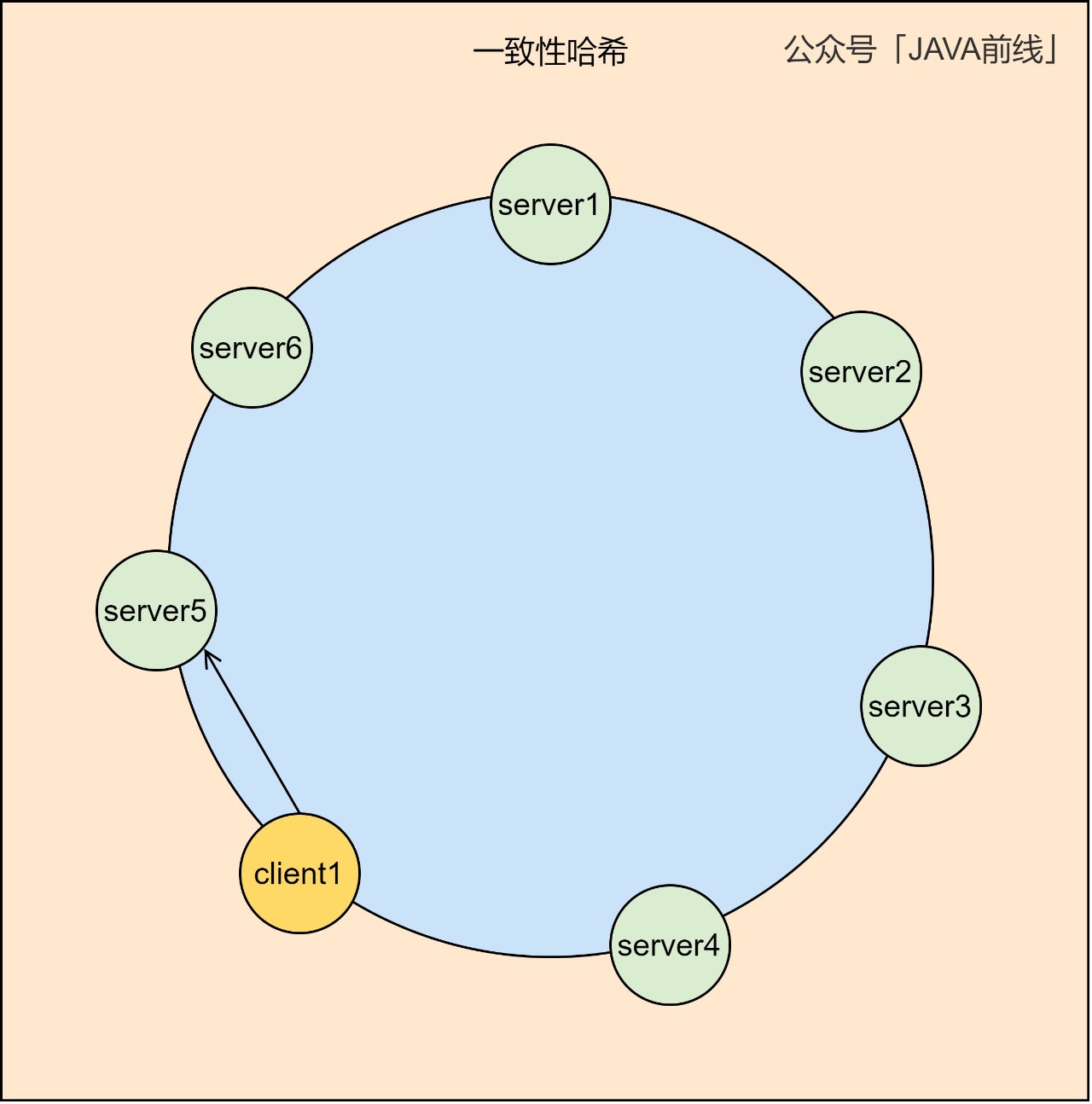
**7 一致性哈希**

一致性哈希策略具有三个显著特性：第一是在不新增或者删除提供者情况下，同一个客户端总是可以访问到同一个提供者。第二是一致性哈希可以有效分散新增或者删除服务提供者带来的波动性。第三是一致性哈希虚拟节点可以更加有效分散特性二之波动性。

### 7.1 特性分析

#### (1) 特性一

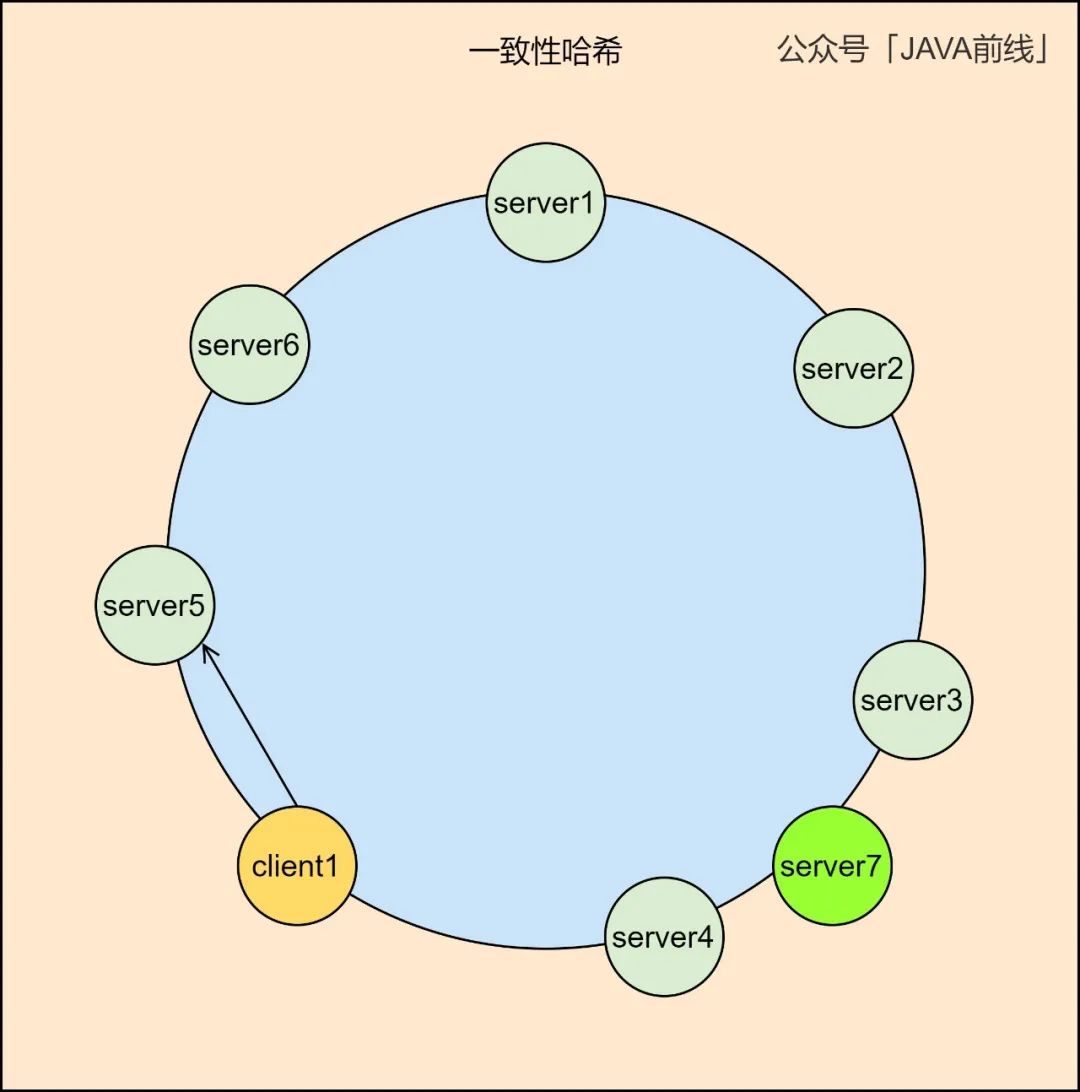
在不新增或者删除提供者情况下，同一个客户端总是可以访问同一个提供者。第一步6个提供者分布在哈希环中，第二步client1发起访问请求，此时计算客户端哈希值位于哈希环位置，第三步沿着哈希环顺时针旋转，找到与客户端哈希值最近的提供者server5。当哈希环结构不发生改变时，client1总是路由到server5。



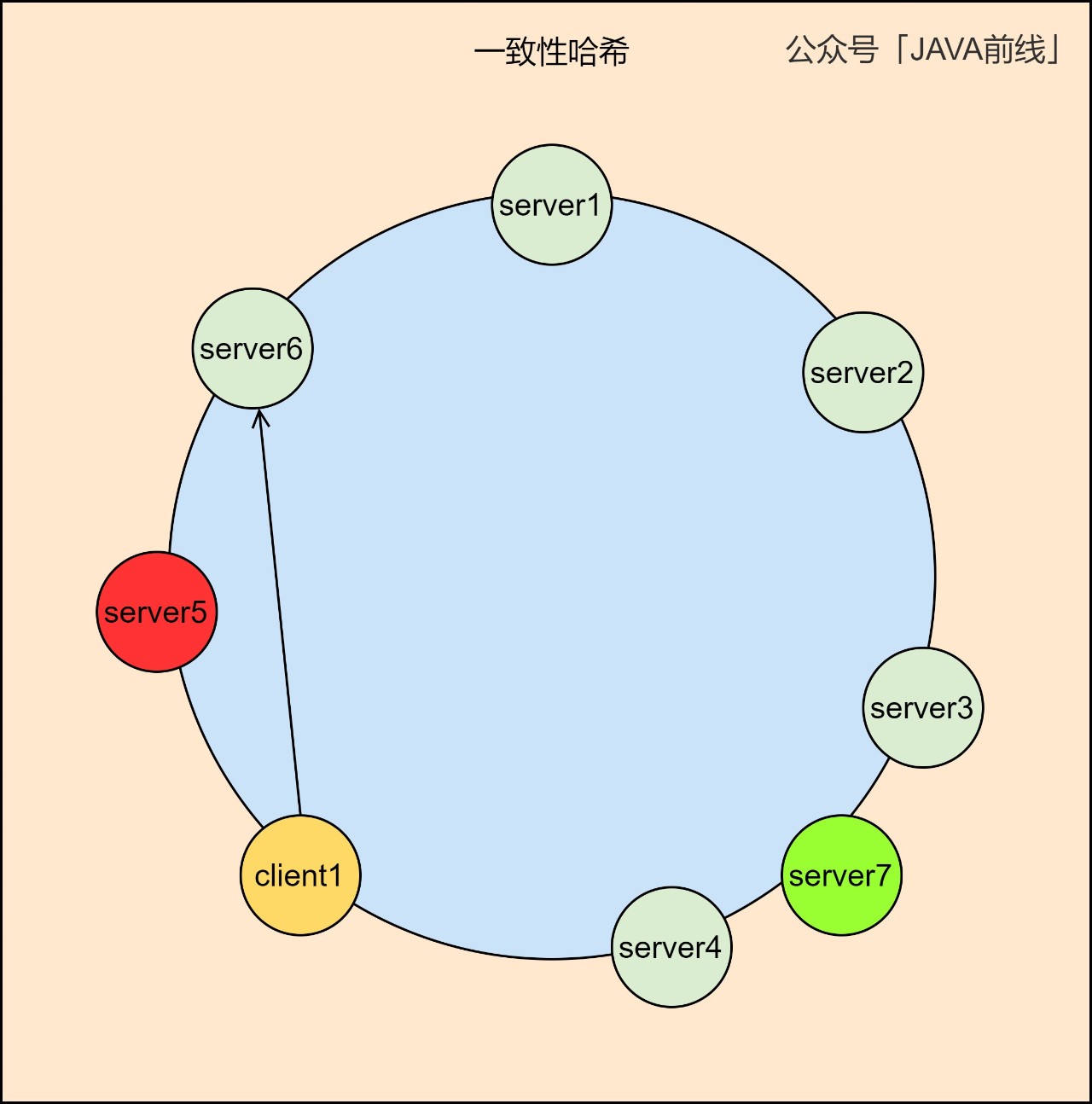
哈希环应该选择什么数据结构呢？我们可以选择TreeMap构建哈希环，其底层使用了红黑树。

#### (2) 特性二

一致性哈希可以有效分散新增或者删除提供者带来的波动性，例如新增服务器server7，但是并不影响client1路由结果：

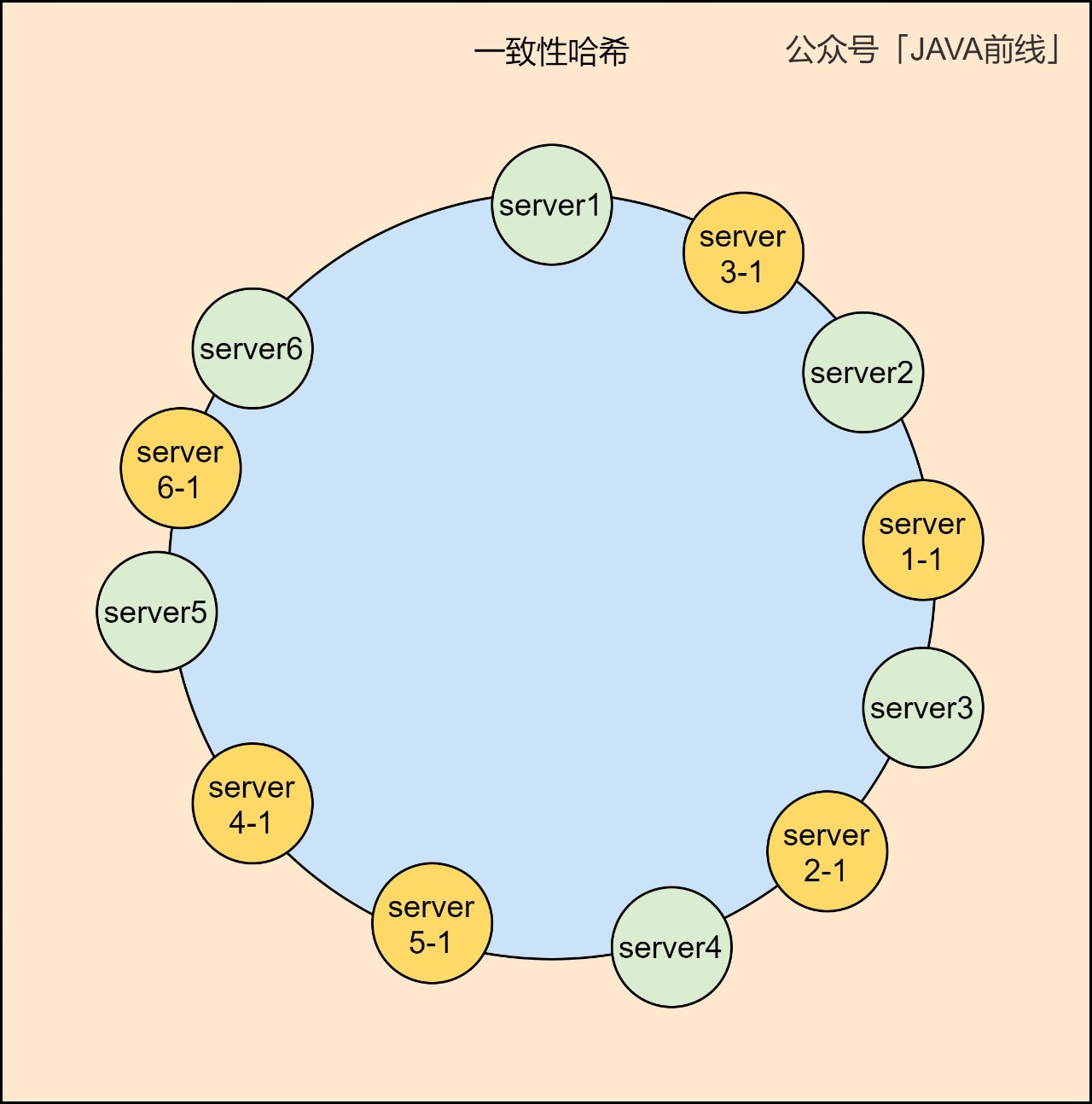


服务器server5发生宕机只会影响client1路由结果，并不会影响其它客户端路由结果：



#### (3) 特性三

一致性哈希虚拟节点可以更加有效分散特性二之波动性，例如我们可以为每个服务器节点新增一个虚拟节点，使得服务器分散得更加均匀：



# 8 最少活跃数

每个提供者维护并发处理的任务个数，任务个数越大活跃度越高。在服务消费者进行负载均衡时，第一查询提供者负载情况，第二选择活跃度最低的提供者

# 9 文章总结

第一本文首先分析了负载均衡、集群容错、服务降级这三组概念，第二结合代码分析了简单随机，加权随机，简单轮询，简单加权轮询，平滑加权轮询，一致性哈希，最少活跃数七种负载均衡策略，其中权重计算、平滑加权轮询，一致性哈希算法值得注意，希望本文对大家有所帮助。