一次 Keepalived 高可用的事故，让我重学了一遍它！

ITPUB 2022-07-03 18:00 发表于北京

以下文章来源于悟空聊架构 ，作者悟空聊架构

悟空聊架构
用故事讲解分布式、架构。《 JVM 性能调优实战》专栏作者，《Spring Cloud 实战...

前言

这次我又遇到了一个奇葩的问题：

> Keepalived 高可用组件的虚拟 IP 持续漂移，导致 MySQL 主从不断切换，进而导致 MySQL 主从数据同步失败。

虽然没能重现 Keepalived 的这个问题，但是我深入研究了下 Keepalived 的原理以及针对核心配置参数做了大量实验。悟空带着大家一起看下 Keepalived 到底是如何运转的，以及为什么它能做到高可用。

一、Keepalived 和 LVS 概述

1.1 Keepalived 概述

谈到 Keepalived，给人的印象就是用在高可用架构中，保证某个服务不故障，其实它还有很多其他的功能。Keepalived 是 Linux 系统下的一个比较轻量级的高可用解决方案，这个轻量级是相对于 Heartbeat 等组件的。虽然 Heartbeat 功能完善、专业性强，但是安装部署就没有 Keepalived 简单，Keepalived 只需要一个配置文件即可。企业中大多选择 Keepalived 作为高可用组件。

1.2 LVS 概述

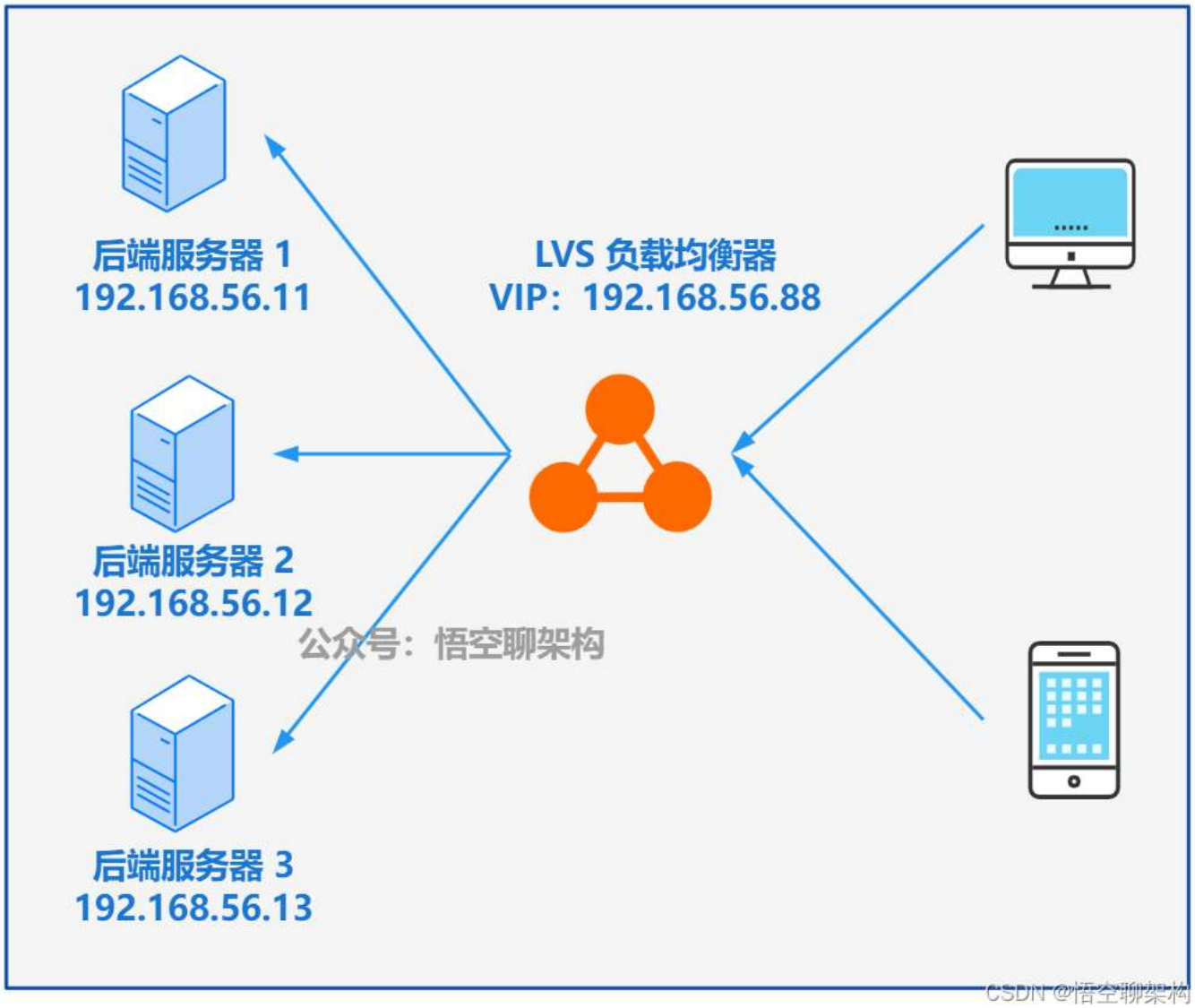
Keepalived 最开始是由 Alexandre Cassen 使用 C 语言编写的开源软件项目，项目的目的主要是简化 LVS 项目的配置并增强 LVS 的稳定性。简单来说，Keepalived 就是对 LVS 的扩展增强。

LVS（Linux Virtual Server）翻译过来就是 Linux 虚拟服务器，由章文嵩博士主导开发的开源负载项目，目前 LVS 已经被集成到 Linux 内核模块中。

LVS 主要用在负载均衡方面，比如 Web 客户端想要访问后端服务，Web 请求会先经过 LVS 调度器，调度器根据预设的算法决定如何分发给后端的所有服务器。

1.3 LVS 基本原理

LVS 的基本原理如下图所示：



LVS基本原理

LVS 的核心功能就是提供负载均衡，负载均衡技术有多种：

- 基于 DNS 域名轮流解析方案。
- 基于客户端调度访问方案。

- 基于应用层系统的调度方案。
- 基于 IP 地址的调度方案。

而效率最高的是基于 IP 地址的调度方案。其实就是将请求转发给对应的 IP 地址 + 端口号，它的效率是非常高的，LVS 的 IP 负载均衡技术是通过 IPVS 模块来实现的，IPVS 是 LVS 集群系统的核心软件。

LVS 负载均衡器会虚拟化一个IP（VIP），对于客户端来说，它事先只知道这个 VIP 的，客户端就将请求发送给 VIP，然后 LVS 负载均衡器会将请求转发给后端服务器中的一个，这些服务器都称为 Real Server（真实服务器）。转发的规则是通过设置 LVS 的负载均衡算法来的，比如随机分配、按照权重分配等。

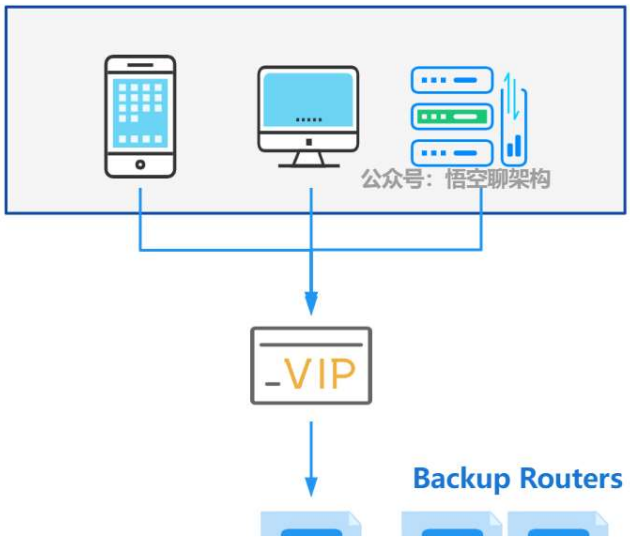
后端服务器的提供的功能要求是一致的，不论转发到哪台服务器，最终得到的结果是一致的，所以对于客户端来说，它并不关心有多少个后端服务器在提供服务，它只关心访问的 VIP 是多少。

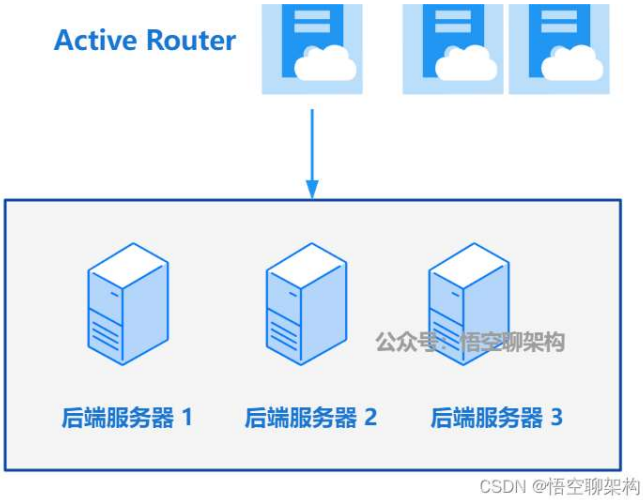
那么后端服务处理完请求后，如何将数据返回给客户端呢？根据 LVS 的不同模式，会选择不同的方式将数据返回给客户端。LVS 的工作模式有三种：NAT 模式、TUN 模式、DR 模式。这个后面讲到路由机制再来细说。

二、Keepalived 流量转发原理

Keepalived 为 Linux 系统提供了负载均衡和高可用能力。负载均衡的能力来自 Linux 内核中的 LVS 项目模块 IPVS（IP Virtual Server）。

Keepalived 运行在 Linux 系统中，它会启动内核中的 LVS 服务来创建虚拟服务器。比如我们在两台服务器上都启动了一个 Keepalived 服务，然后 LVS 会虚拟化出来一个 IP（VIP），但是只有一个 Keepalived 会接管这个 VIP，就是说客户端的请求只会到 Master Keepalived 节点上。这样流量就只会到一台 keepalived 上了，然后 keepalived 可以配置几台真实的服务 IP 地址和端口，通过负载调度算法将流量分摊到这些服务上。对于另外一台 Backup Keepalived 节点，它是待机状态，没有流量接入的。





三、Keepalived 如何进行选主的

那么上面的两个 Keepalived 服务是如何选出其中一个作为 Master 节点的呢?

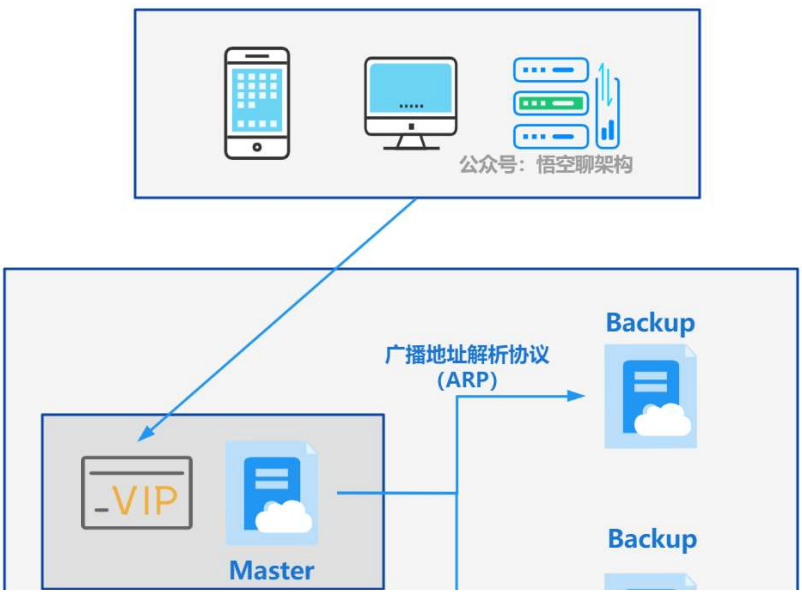
我们一般都是运行在两台主备服务器或一主多备的服务器上。而这多台服务器都是遵循 VRRP 的。

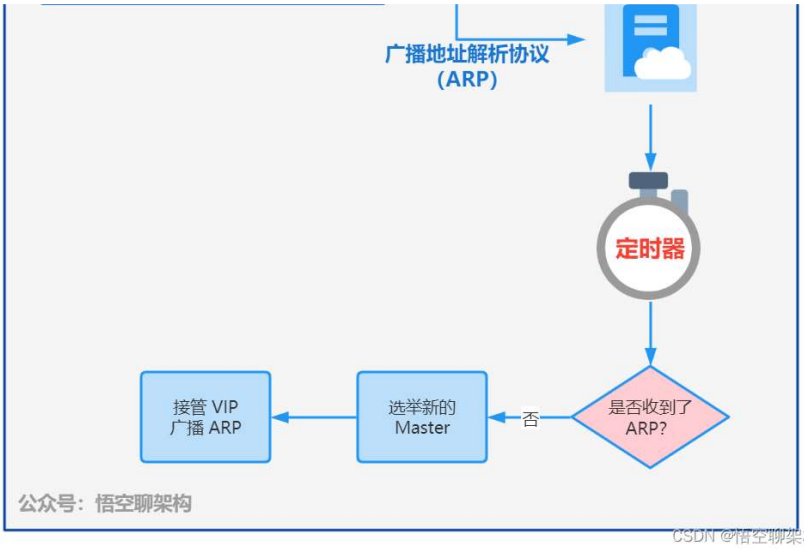
3.1 VRRP 协议

VRRP 的全称为 Virtual Router Redundancy Protoco，虚拟路由冗余协议。它是一种容错协议，为了解决局域网中单点路由故障的问题。比如之前我们都是一个路由器进行路由转发，如果这个路由器故障了，那么整个路由转发的链路就断了，服务就不可用了。

VRRP 协议主要的功能:

- 虚拟路由器和虚拟 IP。
- Master 广播 ARP 报文。
- Backup 选举新的 Master。





现在我们配置多台路由器（一主多备），每台路由器都有一个自己的 IP 地址，它们组成一个路由器组，其中有一个作为 Master，其他作为 Backup。然后这些路由器会虚拟出单个路由，拥有自己的 IP 地址，也就是 Virtual IP，简称 VIP。

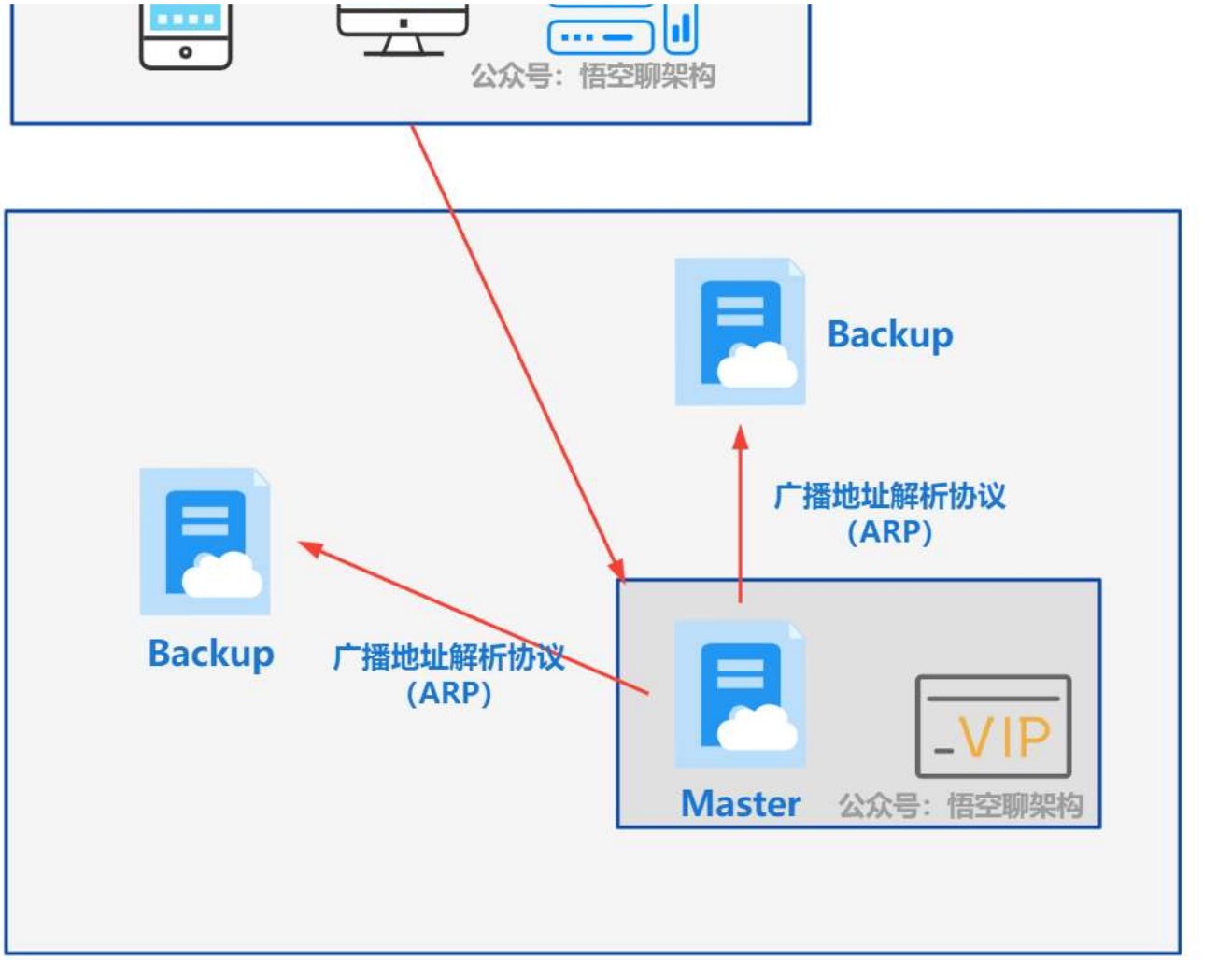
客户端访问这个虚拟的 IP 地址就可以了，当主路由器故障了，备份路由器通过选举机制选出一个新的主路由器，继续向客户端提供路由服务，实现了路由功能的高可用。

路由器开启 VRRP 功能后，根据优先级配置进行选举，优先级高的会成为主（Master）路由器，另外的则会成为备（Backup）路由器。

> Master 路由器定期发送 VRRP 通知报文给 Backup 路由器，告诉它们我是在正常工作的，你们不用竞选新的 Master 路由器。

关于 Master 和 Backup 通信的原理其实很简单，就是一个**心跳机制**，不过这个和 Eureka 的心跳机制不一样，Eureka 是客户端定期向 Eureka 注册中心发送心跳，而 Keepalived 则是 Master 定期向 Backup 发送心跳机制，而 Backup 路由器它有一个定时监测通知的任务，如果在这个时间段内未收到通知，则认为 Mater 故障了，然后通过优先级进行选举，选举出新的 Master 后，就定期发送 VRRP 通知报文给 Backup 路由器。（Eureka 心跳机制：唐太宗把微服务的“心跳机制”玩到了极致！）

通过这个 VRRP 协议，可以提高系统的可用性，避免因单点故障导致的服务不可用问题，同时在路由器故障时，无需手动修改网络连接信息以访问新的 Master 路由器。如下图所示，Backup 切换为了 Master。



CSDN @悟空聊架构

关于选举的配置主要依赖 vrrp_instance 和 vrrp_script 字段。

3.2 vrrp_instance 配置

对于 Keepalived 的选主有三个重要参数：

- state：可选值为 MASTER、BACKUP。
- priority：节点的优先级，可选值为 [1-255]。
- nopreempt：不抢占模式，如果配置，则当优先级高时，会将自己设置为 Master。

```
vrrp_instance VI_1 {
    # 节点为 BACKUP
    state BACKUP
    # 优先级为 100
    priority 100
    # 不抢占模式
    nopreempt
}
```

```
}
```

当一台设置为 master，另外一台设置为 BACKUP，当 MASTER 故障后，BACKUP 会成为新的 MASTER，而当老的 MASTER 恢复后，又会抢占成为新的 MASTER，接管 VIP 的流量，导致不必要的主备切换。为了避免这种主备切换，我们可以将两台 Keepalived 都设置为 BACKUP，且高优先级的那台 Keepalived 设置为不抢占 nopreempt。

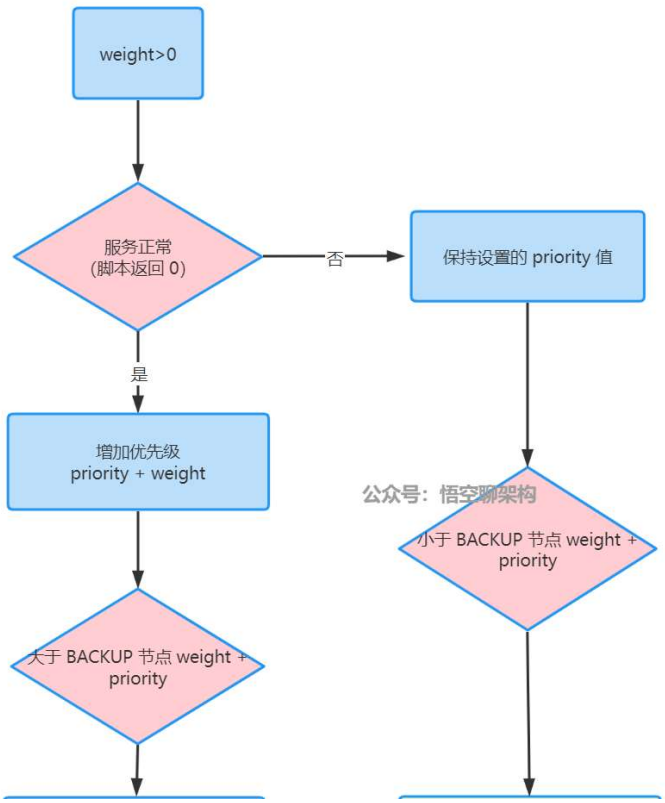
3.2 vrrp_script 配置

而优先级 priority 它是可以增减的，通过 vrrp_script 来配置：

```
vrrp_script restart_mysql {
    # 监测和重启 mysql 容器，如果 MySQL 服务正常或 MySQL 失败
    script "/usr/local/keepalived/restart_mysql.sh"
    interval 5
    weight -20
}
```

这个是定时执行脚本的配置，script 配置会监测 mysql 服务是否不正常。这是一个自定义的脚本，可以自己写返回值。这里我写的逻辑是如果 MySQL 服务正常则返回 0，不正常则返回 1。

当 weight 为正数

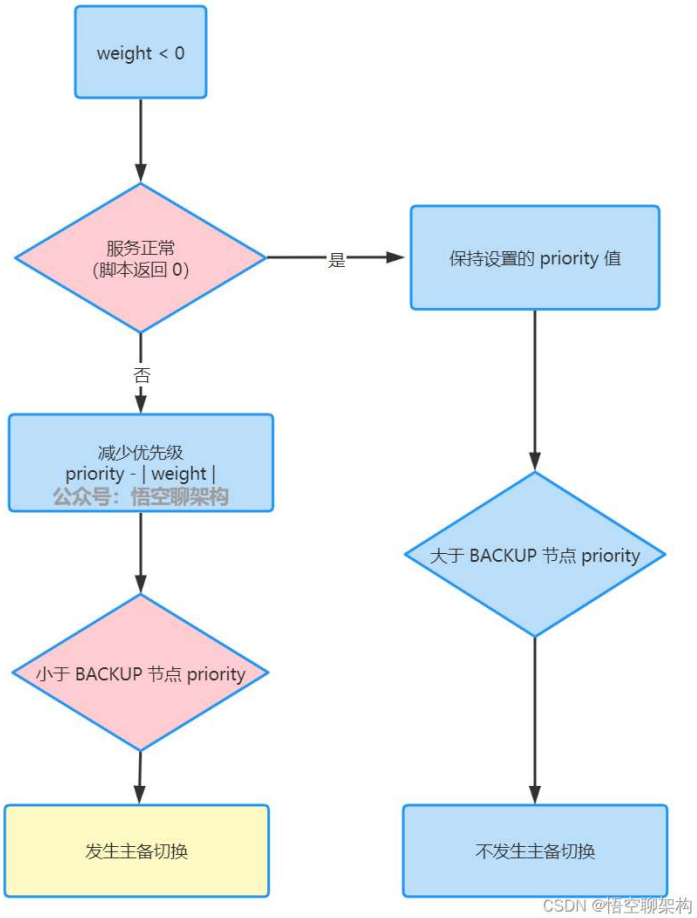


当脚本返回 0 时（服务正常），则增加优先级=priority + weight；否则，保持设置的 priority 值。

切换策略：

- 如果 MASTER 节点的 vrrp_script 脚本检测失败时，如果 MASTER 节点的 priority 值小于 BACKUP 节点 weight + priority，则发生主备切换。
- 如果 MASTER 节点的 vrrp_script 脚本检测成功时，如果 MASTER 节点的 priority 值大于 BACKUP 节点 weight + priority，则不发生主备切换。

当 weight 为负数



当脚本返回非 0 时（服务异常），则优先级=priority - |weight|；否则，保持设置的 priority 值。

切换策略：

- 如果 MASTER 节点的 vrrp_script 脚本检测失败时，如果 MASTER 节点的 priority -|weight| 值小于 BACKUP 节点 priority 值，则发生主备切换。
- 如果 MASTER 节点的 vrrp_script 脚本检测成功时，如果 MASTER 节点的 priority 值大于 BACKUP 节点 priority 值，则不发生主备切换。

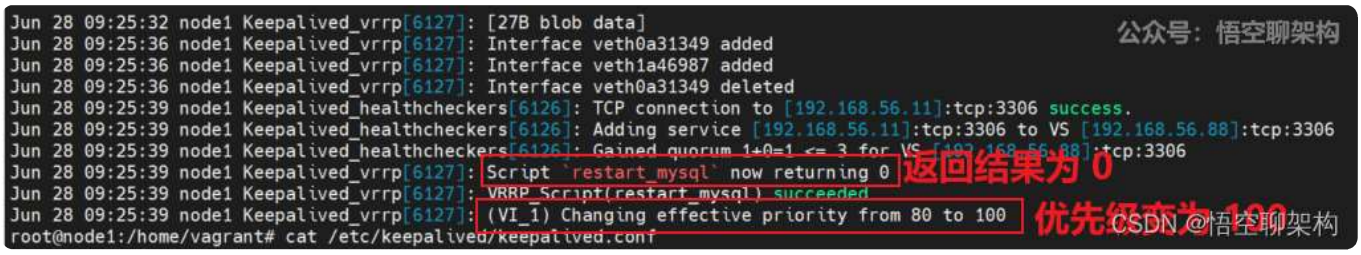
注意：增加或减少优先级的范围为 [1,254]。

举例说明：

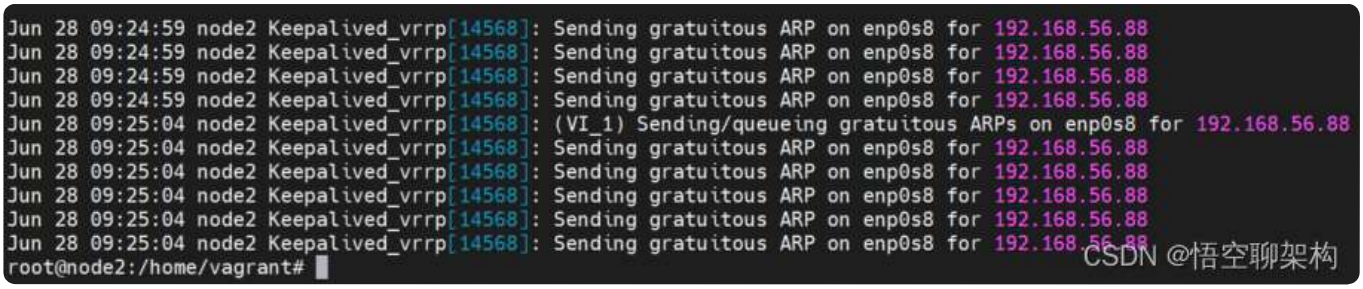
两台 Keepalived 的 state 都配置成 BACKUP，其中一台服务器 node1 的 Keepalived 的优先级设置为 100，不抢占模式，另外一台 node2 的优先级设置为 90，抢占模式。

node1 节点配置的优先级高，它成为 Master 节点，当 Master 节点监控的 MySQL 服务发生故障后，会降低优先级，从 100 降低到 80。另外一台优先级为 90，收到优先级比自己低的 ARP 广播时，就会变成新的 Master 节点。而 node1 节点会成为 BACKUP 节点，当 node1 监控到 MySQL 服务恢复后，优先级变为配置的 priority 100，但是也不会抢占。

如下图所示：虽然 node1 上的 keepalived 重启 mysql 成功了，优先级也恢复成了 100，但是并没有变为 master，还是维持 backup 状态。



而 node2 还是 master 节点，定时向 node 1 发送 vrrp 通知，如下图所示：



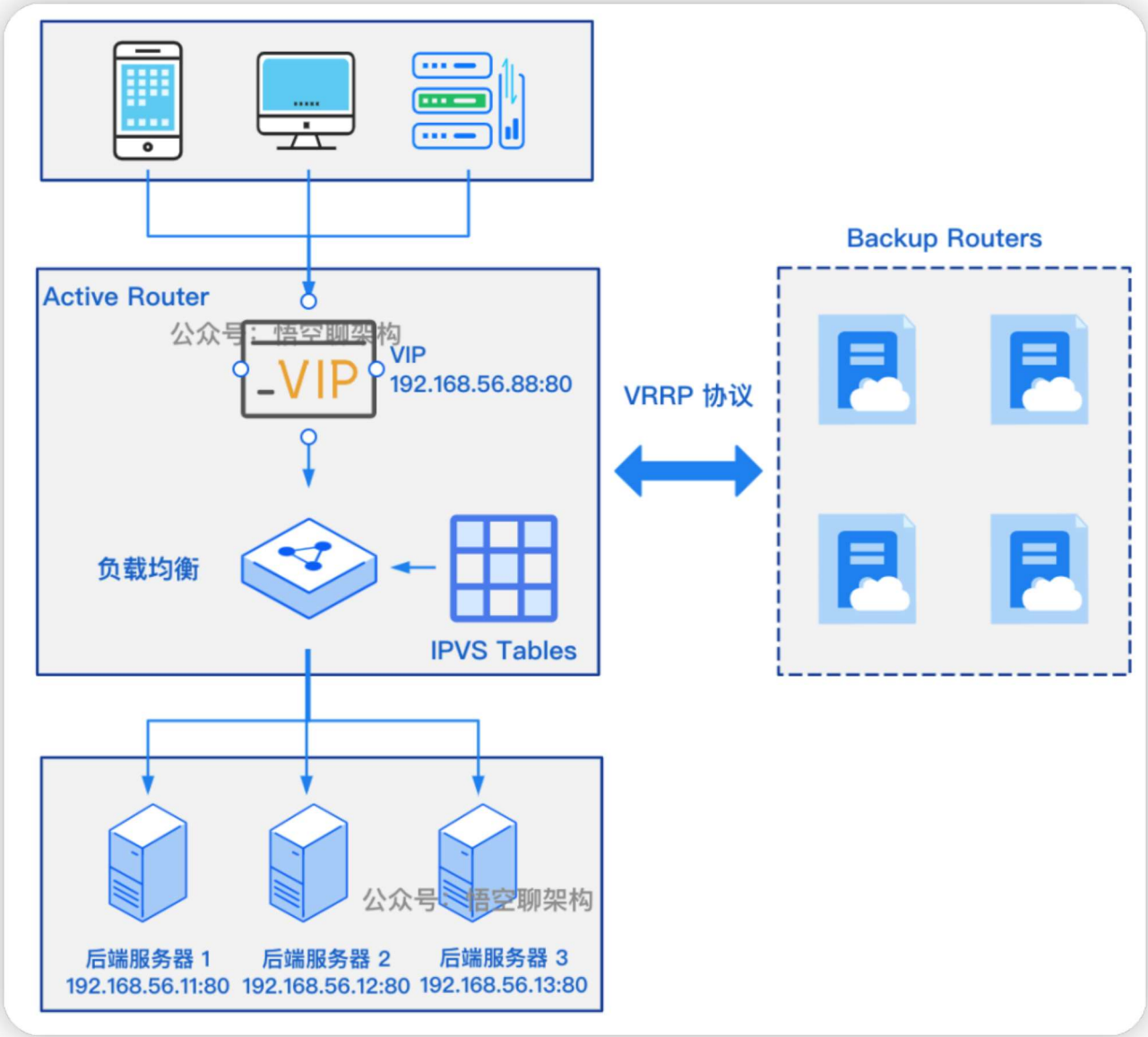
如果 node2 的 mysql 宕机了，那么它的优先级会从 90 降低到 70，即使这样，也不会出现主备切换，因为我们配置的策略就是 node1 不会抢占。如果要在这种情况下切换到 node1，就只能将 node2 的 keepalived 主动停掉，故障转移中篇会讲到。

四、Keepalived 的负载均衡机制

4.1 转发机制

要理解 Keepalived 的负载均衡机制，必须了解 IPVS，也就是 IP Virtual Server，IP 虚拟服务器。

IPVS 模块是 Keepalived 引入的一个第三方模块，目的是解决单 IP 多服务器的工作环境，通过 IPVS 可以实现基于 IP 的负载均衡集群。IPVS 默认包含在 LVS 软件中，而 LVS 又是包含在 Linux 系统中。所以 Keepalived 在 Linux 系统上可以直接利用 LVS 的功能。LVS 的作用就是虚拟出一个 IP，也就是 VIP，客户端请求先到达 VIP，然后从服务器集群中选择一个服务器节点，将流量转发给这个节点，由这个节点处理请求。



如图所示：

- Keepalived 是运行在用户空间的 LVS 路由（LVS Router）进程，作为 MASTER 角色 Keepalived 称为 Active Router，BACKUP 角色的 Keepalived 称为 SLAVE Router。

只有 Active Router 是工作的，其他 Router 是 Stand By （待机状态）。

- Active Router 和 Backup Router 之间是通过 VRRP 协议进行主备切换的。
- Active Router 会启动内核中 LVS 服务以创建虚拟服务器，虚拟服务器有一个虚拟 IP（VIP），比如下图中的 VIP 为 192.168.56.88。
- Active Router 还会设置 IPVS TABLES（服务器列表），记录了后端服务器的地址及服务运行状态。负载均衡就从服务器列表选择一个可用的服务进行转发。
- 这些后端服务是配置在 Keepalived 的 virtual_server 配置项里面的，如下所示，配置了三个 real_server，分别对应了三台后端服务器。

```
virtual_server 192.168.56.88 80 {
    delay_loop 6
    lb_algo rr
    lb kind NAT
    protocol tcp
    # 服务器 1
    real_server 192.168.56.11 80 {
        TCP_CHECK {
        connect timeout 10
    }
    # 服务器 2
    real_server 192.168.56.12 80 {
        TCP_CHECK {
        connect timeout 10
    }
    # 服务器 3
    real_server 192.168.56.13 80 {
        TCP_CHECK {
        connect timeout 10
    }
```

4.2 负载调度算法

配置中有一个字段 lb_algo，这个就是负载调度算法，可以配置成 rr、wrr、lc、wlc、sh、dh 等。常用的是 rr 和 wrr。

- rr，就是 Round-Robin，轮询算法， 每个服务器平等的，依次被调度。
- wrr，就是 Weighted Round-Robin，加权轮询调度算法，加权值较大的，会被转发更多的请求。比如有的服务器硬件能力较弱，则可以将加权值配置得低一点。
- lc，就是 Least-Connection，最少连接算法。请求被转发到活动连接较少的服务器上。连接数是通过 IPVS Table 来动态跟踪的。

- wlc，加权最少连接。根据权重 + 连接数 分配请求。
- sh，目标地址哈希算法，通过在静态 Hash 表中查询目的 IP 地址来确定请求要转发的服务器，这类算法主要用于缓存代理服务器中。
- dh，源地址哈希算法，通过在静态 Hash 表中查询源 IP 地址来确定请求要转发的服务器，这类算法主要用于防火墙的 LVS Router 中。

五、总结

Keepalived 作为高可用、高性能组件，在集群环境中用得还是挺多的，所以去理解 Keepalived 的底层原理，也可以学到很多高可用和负载均衡的通用原理。

本篇介绍了 Keepalived 的 IPVS 功能，启动了一个虚拟服务器，虚拟化了一个 VIP，用来接收客户端的请求，然后通过负载调度算法将流量转发给真实服务器。

Keepalived 一般用在都是一主一备或一主多备的场景，而对于主的选举是通过配置 state、privority、nopreemt、weight 字段来达到的。

阅读原文