第 18 课
HA 集群软件 Keepalived

18.1 HA 集群中的相关术语

1．节点（node）

运行 HA 进程的一个独立主机，称为节点，节点是 HA 的核心组成部分，每个节点上运行着操作系统和高可用软件服务，在高可用集群中，节点有主次之分，分别称为主节点和备用/备份节点，每个节点拥有唯一的主机名，并且拥有属于自己的一组资源，例如，磁盘、文件系统、网络地址和应用服务等。主节点上一般运行着一个或多个应用服务。而备用节点一般处于监控状态。

2．资源（resource）

资源是一个节点可以控制的实体，并且当节点发生故障时，这些资源能够被其它节点接管，HA 集群软件中，可以当做资源的实体有：

- 磁盘分区、文件系统
- IP 地址 VIP
- 应用程序服务
- NFS 文件系统

3．事件（event）

也就是集群中可能发生的事情，例如节点系统故障、网络连通故障、网卡故障、应用程序故障等。这些事件都会导致节点的资源发生转移，HA 的测试也是基于这些事件来进行的。

4．动作（action）

事件发生时 HA 的响应方式，动作是由 shell 脚步控制的，例如，当某个节点发生故障后，备份节点将通过事先设定好的执行脚本进行服务的关闭或启动。进而接管故障节点的资源。

18.2 Keepalived 简介

Keepalived 是 Linux 下一个轻量级的高可用解决方案，它与 HACMP、RoseHA 实现的功能类似，都可以实现服务或者网络的高可用，但是又有差别：HACMP 是一个专业的、

功能完善的高可用软件，它提供了 HA 软件所需的基本功能，比如心跳检测和资源接管，监测集群中的系统服务，在群集节点间转移共享 IP 地址的所有者等，HACMP 功能强大，但是部署和使用相对比较麻烦，同时也是商业化软件；与 HACMP 相比，**Keepalived 主要是通过虚拟路由冗余来实现高可用功能**，虽然它没有 HACMP 功能强大，但 Keepalived 部署和使用非常简单，所有配置只需一个配置文件即可完成。这也是本课程重点介绍 Keepalived 的原因。

18.2.1 Keepalived 的用途

Keepalived 起初是为 LVS 设计的，专门用来监控集群系统中各个服务节点的状态。它根据 layer3, 4 & 5 交换机制检测每个服务节点的状态，如果某个服务节点出现异常，或工作出现故障，Keepalived 将检测到，并将出现故障的服务节点从集群系统中剔除，而在故障节点恢复正常后，Keepalived 又可以自动将此服务节点重新加入到服务器集群中，这些工作全部自动完成，不需要人工干涉，需要人工完成的只是修复出现故障的服务节点。

Keepalived 后来又加入了 VRRP 的功能，VRRP 是 Virtual Router Redundancy Protocol（虚拟路由器冗余协议）的缩写，它出现的目的是为了解决静态路由出现的单点故障问题，通过 VRRP 可以实现网络不间断地、稳定地运行。因此，Keepalived 一方面具有服务器状态检测和故障隔离功能，另一方面也具有 HA cluster 功能.下面详细介绍下 VRRP 协议的实现过程。

18.2.2 VRRP 协议与工作原理

在现实的网络环境中，主机之间的通信都是通过配置静态路由（默认网关）完成的，而主机之间的路由器一旦出现故障，通信就会失败，因此，在这种通信模式中，路由器就成了一个单点瓶颈，为了解决这个问题，就引入了 VRRP 协议。

熟悉网络的学员对 VRRP 协议应该并不陌生。它是一种主备模式的协议，通过 VRRP 可以在网络发生故障时透明地进行设备切换而不影响主机间的数据通信，这其中涉及两个概念：物理路由器和虚拟路由器。

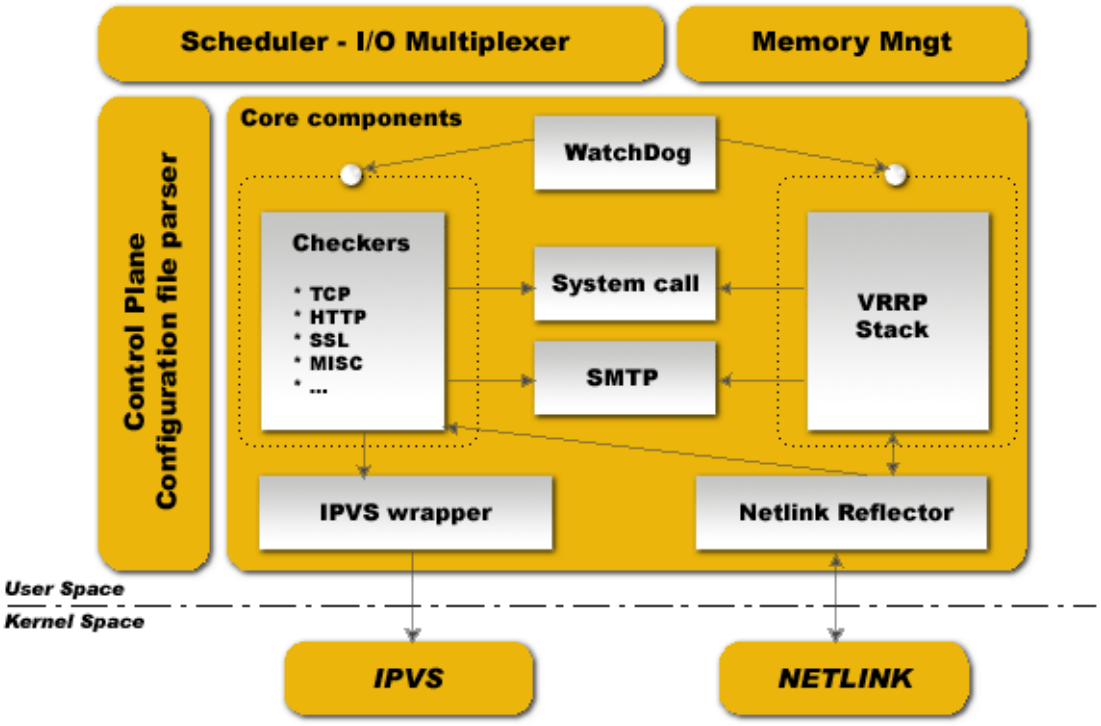
VRRP 可以将两台或多台物理路由器设备虚拟成一个虚拟路由器，这个虚拟路由器通过**虚拟 IP（一个或多个）对外提供服务，而在虚拟路由器内部，是多个物理路由器协同工作，**

同一时间只有一台物理路由器对外提供服务，这台物理路由器被称为主路由器（处于 MASTER 角色）。一般情况下 MASTER 由选举算法产生，它拥有对外服务的虚拟 IP，提供各种网络功能，如 ARP 请求、ICMP、数据转发等。而其他物理路由器不拥有对外的虚拟 IP，也不提供对外网络功能，仅仅接收 MASTER 的 VRRP 状态通告信息，这些路由器被统称为备份路由器（处于 BACKUP 角色）。当主路由器失效时，处于 BACKUP 角色的备份路由器将重新进行选举，产生一个新的主路由器进入 MASTER 角色继续提供对外服务，整个切换过程对用户来说完全透明。

在一个虚拟路由器中，只有处于 MASTER 角色的路由器会一直发送 VRRP 数据包，处于 BACKUP 角色的路由器只接收 MASTER 发过来的报文信息，用来监控 MASTER 运行状态，因此，不会发生 MASTER 抢占的现象，除非它的优先级更高。而当 MASTER 不可用时，BACKUP 也就无法收到 MASTER 发过来的报文信息，于是就认定 MASTER 出现故障，接着多台 BACKUP 就会进行选举，优先级最高的 BACKUP 将成为新的 MASTER，这种选举并进行角色切换的过程非常快，因而也就保证了服务的持续可用性。

18.2.3 Keepalived 的体系结构

Keepalived 是一个高度模块化的软件，结构简单，但扩展性很强，下图是官方给出的 Keepalived 体系结构拓扑图。



可以看出，Keepalived 的体系结构从整体上分为两层，分别是用户空间层（User Space）和内核空间层（Kernel Space）.下面介绍 Keepalived 两层结构的详细组成及实现的功能。

内核空间层处于最底层，它包括 IPVS 和 NETLINK 两个模块。IPVS 模块是 Keepalived 引入的一个第三方模块，通过 IPVS 可以实现基于 IP 的负载均衡集群。IPVS 默认包含在 LVS 集群软件中。

Keepalived 最初就是为 LVS 提供服务的，由于 Keepalived 可以实现对集群节点的状态检测，而 IPVS 可以实现负载均衡功能，因此，Keepalived 借助于第三方模块 IPVS 就可以很方便地搭建一套负载均衡系统。在这里有个误区，由于 Keepalived 可以和 IPVS 一起很好地工作，因此很多初学者都以为 Keepalived 就是一个负载均衡软件，这种理解是错误的。

在 Keepalived 中，IPVS 模块是可配置的，如果需要负载均衡功能，可以在编译 Keepalived 时打开负载均衡功能，反之，也可以通过配置编译参数关闭。

NETLINK 模块主要用于实现一些高级路由框架和一些相关的网络功能，完成用户空间层 Netlink Reflector 模块发来的各种网络请求。

用户空间层位于内核空间层之上，Keepalived 的所有具体功能都在这里实现.

18.3 Keepalived 安装与配置

18.3.1 Keepalived 的安装过程

keepalived 的安装非常简单，建议通过 yum 方式直接安装：

[root@233server ~]# yum install keepalived

如果需要 lvs 功能，还需要安装 ipvs 模块：

[root@233server ~]# yum install ipvsadm

18.3.2 Keepalived 的全局配置

在上节安装 Keepalived 的过程中，指定了 Keepalived 配置文件的路径为 /etc/Keepalived/Keepalived.conf，Keepalived 的所有配置均在这个配置文件中完成。

由于 Keepalived.conf 文件中可配置的选项比较多，这里根据配置文件所实现的功能，将 Keepalived 配置分为三类，分别是：

- 全局配置(Global Configuration)、
- VRRPD 配置
- LVS 配置。

下面将主要介绍下 Keepalived 配置文件中一些常用配置选项的含义和用法。

Keepalived 的配置文件都是以块（block）的形式组织的，每个块的内容都包含在{}中，以“#”和“!”开头的行都是注释。全局配置就是对整个 Keepalived 都生效的配置，基本内容如下：

```
! Configuration File for keepalived
global_defs {
    notification_email {
      dba.gao@gmail.com
      ixdba@163.com
    }
    notification_email_from Keepalived@localhost
    smtp_server 192.168.200.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}
```

全局配置以“global_defs”作为标识，在“global_defs”区域内的都是全局配置选项，其中：

- notification_email 用于设置报警邮件地址，可以设置多个，每行一个。注意，如果要开启邮件报警，需要开启本机的 Sendmail 服务。
- notification_email_from 用于设置邮件的发送地址。
- smtp_server 用于设置邮件的 smtp server 地址。
- smtp_connect_timeout 用于设置连接 smtp server 的超时时间。
- router_id 表示运行 Keepalived 服务器的一个标识，是发邮件时显示在邮件主题中的信息。

18.3.3 Keepalived 的 VRRPD 配置

VRRPD 配置是 Keepalived 所有配置的核心，主要用来实现 Keepalived 的高可用功能。下面进入 VRRP 实例的配置，也就是配置 Keepalived 的高可用功能。VRRP 实例段主要用来配置节点角色（主或从）、实例绑定的网络接口、节点间验证机制、集群服务 IP 等。下面是实例 VI_1 的一个配置样例。

```
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id   151
    priority 100
    advert_int 1
 track_interface {
    eth0
    eth1
    }
    authentication {
        auth_type PASS
        auth_pass qwaszx
    }
    virtual_ipaddress {
        192.168.200.16
        192.168.200.17 dev eth1
        192.168.200.18 dev eth2
    }
    nopreempt
    preemtp_delay   300

    notify_master "/etc/keepalived/master.sh "
```

notify_backup "/etc/keepalived/backup.sh"

notify_fault "/etc/keepalived/fault.sh"

}

以上 VRRP 配置以“vrrp_instance”作为标识，在这个实例中包含了若干配置选项，分别介绍如下：

- vrrp_instance 是 VRRP 实例开始的标识，后跟 VRRP 实例名称。
- state 用于指定 Keepalived 的角色，MASTER 表示此主机是主服务器，BACKUP 表示此主机是备用服务器。
- interface 用于指定 HA 监测网络的接口。
- virtual_router_id 是虚拟路由标识，这个标识是一个数字，同一个 vrrp 实例使用唯一的标识，即在同一个 vrrp_instance 下，MASTER 和 BACKUP 必须是一致的。
- priority 用于定义节点优先级，数字越大表示节点的优先级就越高。在一个 vrrp_instance 下，MASTER 的优先级必须大于 BACKUP 的优先级。
- advert_int 用于设定 MASTER 与 BACKUP 主机之间同步检查的时间间隔，单位是秒。
- track_interface 用于设置一些额外的网络监控接口，其中任何一个网络接口出现故障，Keepalived 都会进入 FAULT 状态。
- authentication 用于设定节点间通信验证类型和密码，验证类型主要有 PASS 和 AH 两种，在一个 vrrp_instance 下，MASTER 与 BACKUP 必须使用相同的密码才能正常通信。
- virtual_ipaddress 用于设置虚拟 IP 地址（VIP），又叫做漂移 IP 地址。可以设置多个虚拟 IP 地址，每行一个。之所以称为漂移 IP 地址，是因为 Keepalived 切换到 Master 状态时，这个 IP 地址会自动添加到系统中，而切换到 BACKUP 状态时，这些 IP 又会自动从系统中删除。Keepalived 通过“ip address add”命令的形式将 VIP 添加进系统中。要查看系统中添加的 VIP 地址，可以通过“ip add”命令实现。“virtual_ipaddress”段中添加的 IP 形式可以多种多样，例如可以写成“192.168.16.189/24 dev eth1”这样的形式，而 Keepalived 会使用 IP 命令“ip addr add 192.168.16.189/24 dev eth1”将 IP 信息添加到系统中。因此，这里的配置规则和 IP 命令的使用规则是一致的。
- nopreempt 设置的是高可用集群中的不抢占功能。在一个 HA Cluster 中，如果主节点死机了，备用节点会进行接管，主节点再次正常启动后一般会自动接管服务。这种来

回切换的操作，对于实时性和稳定性要求不高的业务系统来说，还是可以接受的，而对于稳定性和实时性要求很高的业务系统来说，不建议来回切换，毕竟服务的切换存在一定的风险和不稳定性，在这种情况下，就需要设置 nopreempt 这个选项了。设置 nopreempt 可以实现主节点故障恢复后不再切回到主节点，让服务一直在备用节点工作，直到备用节点出现故障才会进行切换。在使用不抢占时，只能在“state”状态为“BACKUP”的节点上设置，而且这个节点的优先级必须高于其他节点。

- preemtp_delay 用于设置切换的延时时间，单位是秒。有时候系统启动或重启之后网络需要经过一段时间才能正常工作，在这种情况下进行发生主备切换是没必要的，此选项就是用来设置这种情况发生的时间间隔。在此时间内发生的故障将不会进行切换，而如果超过“preemtp_delay”指定的时间，并且网络状态异常，那么才开始进行主备切换。
- **notify_master：指定当 Keepalived 进入 Master 状态时要执行的脚本，这个脚本可以是一个状态报警脚本，也可以是一个服务管理脚本。Keepalived 允许脚本传入参数，因此灵活性很强。**
- **notify_backup：指定当 Keepalived 进入 Backup 状态时要执行的脚本，同理，这个脚本可以是一个状态报警脚本，也可以是一个服务管理脚本。**
- **notify_fault：指定当 Keepalived 进入 Fault 状态时要执行的脚本，脚本功能与前两个类似。**
- **notify_stop：指定当 Keepalived 程序终止时需要执行的脚本。**

18.3.4 Keepalived 的 LVS 配置

由于 Keepalived 属于 LVS 的扩展项目，因此， Keepalived 可以与 LVS 无缝整合，轻松搭建一套高性能的负载均衡集群系统。下面介绍下 Keepalived 配置文件中关于 LVS 配置段的配置方法。

LVS 段的配置以“virtual_server”作为开始标识，此段内容有两部分组成，分别是 real_server 段和健康检测段。下面是 virtual_server 段常用选项的一个配置示例：

```
virtual_server 192.168.12.200 80 {
    delay_loop 6
    lb_algo rr
```

lb_kind DR

persistence_timeout 50

persistence_granularity <NETMASK>

protocol TCP

sorry_server <IPADDR> <PORT>

下面介绍每个选项的含义。

- virtual_server：设置虚拟服务器的开始，后面跟虚拟 IP 地址和服务端口，IP 与端口之间用空格隔开。
- delay_loop：设置健康检查的时间间隔，单位是秒。
- lb_algo：设置负载调度算法，可用的调度算法有 rr、wrr、lc、wlc、lblc、sh、dh 等，常用的算法有 rr 和 wlc。
- lb_kind：设置 LVS 实现负载均衡的机制，有 NAT、TUN 和 DR 三个模式可选。
- persistence_timeout：会话保持时间，单位是秒。这个选项对动态网页是非常有用的，为集群系统中的 session 共享提供了一个很好的解决方案。有了这个会话保持功能，用户的请求会一直分发到某个服务节点，直到超过这个会话的保持时间。需要注意的是，这个会话保持时间是最大无响应超时时间，也就是说，用户在操作动态页面时，如果在 50 秒内没有执行任何操作，那么接下来的操作会被分发到另外的节点，但是如果用户一直在操作动态页面，则不受 50 秒的时间限制。
- persistence_granularity：此选项是配合 persistence_timeout 的，后面跟的值是子网掩码，表示持久连接的粒度。默认是 255.255.255.255，也就是一个单独的客户端 IP。如果将掩码修改为 255.255.255.0，那么客户端 IP 所在的整个网段的请求都会分配到同一个 real server 上。
- protocol：指定转发协议类型，有 TCP 和 UDP 两种可选。
- ha_suspend：节点状态从 Master 到 Backup 切换时，暂不启用 real server 节点的健康检查。
- sorry_server：相当于一个备用节点，在所有 real server 失效后，这个备用节点会启用。

下面是 real_server 段的一个配置示例：

real_server 192.168.12.132 80 {

weight 3

inhibit_on_failure

notify_up <STRING> | <QUOTED-STRING>

notify_down <STRING> | <QUOTED-STRING>

}

下面介绍每个选项的含义。

- real_server：是 real_server 段开始的标识，用来指定 real server 节点，后面跟的是 real server 的真实 IP 地址和端口，IP 与端口之间用空格隔开。
- weight：用来配置 real server 节点的权值。权值大小用数字表示，数字越大，权值越高。设置权值的大小可以为不同性能的服务器分配不同的负载，为性能高的服务器设置较高的权值，而为性能较低的服务器设置相对较低的权值，这样才能合理地利用和分配了系统资源。
- inhibit_on_failure：表示在检测到 real server 节点失效后，把它的“weight”值设置为 0，而不是从 IPVS 中删除。
- notify_up：此选项与上面介绍过的 notify_maser 有相同的功能，后跟一个脚本，表示在检测到 real server 节点服务处于 UP 状态后执行的脚本。
- notify_down：表示在检测到 real server 节点服务处于 DOWN 状态后执行的脚本。

健康检测段允许多种检查方式，常见的有 HTTP_GET、SSL_GET、TCP_CHECK、SMTP_CHECK、MISC_CHECK。首先看 TCP_CHECK 检测方式示例：

```
TCP_CHECK {
        connect_port 80
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
```

下面介绍每个选项的含义介。

- connect_port：健康检查的端口，如果无指定，默认是 real_server 指定的端口。
- connect_timeout：表示无响应超时时间，单位是秒，这里是 3 秒超时。

- nb_get_retry：表示重试次数，这里是 3 次。
- delay_before_retry：表示重试间隔，这里是间隔 3 秒。

下面是 HTTP_GET 和 SSL_GET 检测方式的示例：

```
HTTP_GET | SSL_GET
    {
        url  {
        path   /index.html
        digest   e6c271eb5f017f280cf97ec2f51b02d3
        status_code    200
        }
    connect_port 80
    bindto   192.168.12.80
    connect_timeout   3
    nb_get_retry   3
    delay_before_retry   2
    }
```

下面介绍每个选项的含义。

- url：用来指定 HTTP/SSL 检查的 URL 信息，可以指定多个 URL。
- path：后跟详细的 URL 路径。
- **digest：SSL 检查后的摘要信息**，这些摘要信息可以通过 genhash 命令工具获取。
  例如：genhash -s 192.168.12.80 -p 80 -u /index.html。
- status_code：指定 HTTP 检查返回正常状态码的类型，一般是 200。
- bindto：表示通过此地址来发送请求对服务器进行健康检查。

下面是 MISC_CHECK 检测方式的示例：

```
MISC_CHECK
{
misc_path   /usr/local/bin/script.sh
misc_timeout   5
! misc_dynamic
```

}

MISC 健康检查方式可以通过执行一个外部程序来判断 real server 节点的服务状态，使用非常灵活。以下是常用的几个选项的含义。

- misc_path：用来指定一个外部程序或者一个脚本路径。
- misc_timeout：设定执行脚本的超时时间。
- misc_dynamic：表示是否启用动态调整 real server 节点权重，“!misc_dynamic”表示不启用，相反则表示启用。在启用这功能后，Keepalived 的 healthchecker 进程将通过退出状态码来动态调整 real server 节点的“weight”值，如果返回状态码为 0，表示健康检查正常，real server 节点权重保持不变；如果返回状态码为 1，表示健康检查失败，那么就将 real server 节点权重设置为 0；如果返回状态码为 2~255 之间任意数值，表示健康检查正常，但 real server 节点的权重将被设置为返回状态码减 2，例如返回状态码为 10，real server 节点权重将被设置为 8（10-2）。

到这里为止，Keepalived 配置文件中常用的选项已经介绍完毕，在默认情况下，Keepalived 在启动时会查找/etc/Keepalived/Keepalived.conf 配置文件，如果配置文件放在其他路径下，通过“Keepalived -f”参数指定配置文件的路径即可。

在配置 Keepalived.conf 时，需要特别注意配置文件的语法格式，因为 Keepalived 在启动时并不检测配置文件的正确性，即使没有配置文件，Keepalived 也照样能够启动，所以一定要保证配置文件正确。

18.4 Keepalived 基础功能应用实例

Keepalived 提供了 vrrp_script、notify_master、notify_backup 等多个功能模块，通过这些模块可以实现对集群资源的托管以及集群服务的监控。

18.4.1 Keepalived 基础 HA 功能演示

在默认情况下，Keepalived 可以实现对系统死机、网络异常及 Keepalived 本身进行监控，也就是说当系统出现死机、网络出现故障或 Keepalived 进程异常时，Keepalived 会进行主备节点的切换。但这些还是不够的，因为集群中运行的服务也随时可能出现问题，

因此，还需要对集群中运行服务的状态进行监控，当服务出现问题时也进行主备切换。Keepalived 作为一个优秀的高可用集群软件，也考虑到了这一点，它提供了一个 vrrp_script 模块专门用来对集群中服务资源进行监控。

1. 配置 Keepalived

下面将通过配置一套 Keepalived 集群系统来实际演示一下 Keepalived 高可用集群的实现过程。这里以操作系统 CentOS release 6.5、keepalived-1.2.12 版本为例，更具体的集群部署环境如下表所示。

主机名	主机 IP 地址	集群角色	集群服务	虚拟 IP 地址
keepalived-master	192.168.66.11	Master（主节点）	HTTPD	192.168.66.80
keepalived-backup	192.168.66.12	Backup（备用节点）	HTTPD	

通过此表可以看出，这里要部署一套基于 HTTPD 的高可用集群系统。

关于 Keepalived 的安装，之前已经做过详细介绍，这里不再多说，下面给出 keepalived-master 节点的 keepalived.conf 文件的内容。

```
global_defs {
    notification_email {
      acassen@firewall.loc
      failover@firewall.loc
      sysadmin@firewall.loc
    }
    notification_email_from Alexandre.Cassen@firewall.loc
    smtp_server 192.168.200.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}

vrrp_script check_httpd {
```

```
    script "killall -0 httpd"
    interval 2
    }

vrrp_instance HA_1 {
    state MASTER
    interface eth0
    virtual_router_id 80
    priority 100
    advert_int 2
    authentication {
        auth_type PASS
        auth_pass qwaszx
    }
    notify_master "/etc/keepalived/master.sh "
    notify_backup "/etc/keepalived/backup.sh"
    notify_fault "/etc/keepalived/fault.sh"

    track_script {
    check_httpd
    }

    virtual_ipaddress {
        192.168.66.80/24 dev eth0
    }
}
```

其中，master.sh 文件的内容为：

```
#!/bin/bash
```

```
LOGFILE=/var/log/keepalived-mysql-state.log
echo "[Master]" >> $LOGFILE
date >> $LOGFILE
```

backup.sh 文件的内容为：

```
#!/bin/bash
LOGFILE=/var/log/keepalived-mysql-state.log
echo "[Backup]" >> $LOGFILE
date >> $LOGFILE
```

fault.sh 文件的内容为：

```
#!/bin/bash
LOGFILE=/var/log/keepalived-mysql-state.log
echo "[Fault]" >> $LOGFILE
date >> $LOGFILE
```

这三个脚本的作用是监控 Keepalived 角色的切换过程，从而帮助读者理解 notify 参数的执行过程。

keepalived-backup 节点上的 keepalived.conf 配置文件内容与 keepalived-master 节点上的基本相同，需要修改的地方有两个：

- 将“state MASTER”更改为“state BACKUP”。
- 将 priority 100 更改为一个较小的值，这里改为“priority 80”。

2. Keepalived 启动过程分析

将配置好的 keepalived.conf 文件及 master.sh、backup.sh、fault.sh 三个文件一起复制到 keepalived-backup 备用节点对应的路径下，然后在两个节点启动 http 服务，最后启动 Keepalived 服务。下面介绍具体的操作过程。

首先在 keepalived-master 节点启动 keepalived 服务，执行如下操作：

```
[root@keepalived-master keepalived]# chkconfig --level 35 httpd on
[root@keepalived-master keepalived]# /etc/init.d/httpd  start
```

[root@keepalived-master keepalived]# /etc/init.d/keepalived start

Keepalived 正常运行后共启动了 3 个进程，其中一个进程是父进程，负责监控其余两个子进程（分别是 vrrp 子进程和 healthcheckers 子进程）。然后观察 keepalived-master 上 Keepalived 的运行日志，信息如下图所示。

```
Mar  4 17:23:19 keepalived-master Keepalived_vrrp[24348]: VRRP_Script(check_httpd) succeeded
Mar  4 17:23:21 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Transition to MASTER STATE
Mar  4 17:23:23 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Entering MASTER STATE
Mar  4 17:23:23 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) setting protocol VIPs.
Mar  4 17:23:23 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Sending gratuitous ARPs on eth0 for
192.168.66.80
Mar  4 17:23:23 keepalived-master Keepalived_healthcheckers[24347]: Netlink reflector reports IP 192.168.66.80 added
Mar  4 17:23:23 keepalived-master avahi-daemon[1315]: Registering new address record for 192.168.66.80 on
eth0.IPv4.
Mar  4 17:23:28 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Sending gratuitous ARPs on eth0 for
192.168.66.80
```

从日志可以看出，在 keepalived-master 主节点启动后，VRRP_Script 模块首先运行了 check_httpd 的检查，发现 httpd 服务运行正常，然后进入 Master 角色，如果检查 httpd 服务异常，将进入 Fault 状态，最后将虚拟 IP 地址添加到系统中，完成 Keepalived 在主节点的启动。此时在主节点通过命令“ip add”就能查看到已经添加到系统中的虚拟 IP 地址。

再查看/var/log/keepalived-mysql-state.log 日志文件，内容如下：

[root@keepalived-master keepalived]#tail -f /var/log/keepalived-mysql-state.log

[Master]

Tue Mar 4 17:23:23 CST 2014

通过上面给出的三个脚本的内容可知，Keepalived 在切换到 Master 角色后，执行了/etc/keepalived/master.sh 这哥脚本，从这里也可以看出 notify_master 的作用。

接着在备用节点 keepalived-backup 上也启动 keepalived 服务，执行如下操作：

[root@keepalived-backup keepalived]# chkconfig --level 35 httpd on

[root@keepalived-backup keepalived]# /etc/init.d/httpd start

[root@keepalived-backup keepalived]# /etc/init.d/keepalived start

然后观察 keepalived-backup 上 Keepalived 的运行日志，信息如下图所示。

```
Mar  4 17:27:15 keepalived-backup Keepalived_healthcheckers[25912]: Opening file '/etc/keepalived/keepalived.conf'.
Mar  4 17:27:15 keepalived-backup Keepalived_healthcheckers[25912]: Configuration is using : 7500 Bytes
Mar  4 17:27:15 keepalived-backup Keepalived_vrrp[25913]: VRRP_Instance(HA_1) Entering BACKUP STATE
Mar  4 17:27:15 keepalived-backup Keepalived_vrrp[25913]: VRRP sockpool: [ifindex(2), proto(112), unicast(0), fd(10,11)]
Mar  4 17:27:15 keepalived-backup Keepalived_healthcheckers[25912]: Using LinkWatch kernel netlink reflector...
Mar  4 17:27:15 keepalived-backup Keepalived_vrrp[25913]: VRRP_Script(check_httpd) succeeded
```

从日志输出可以看出，keepalived-backup 备用节点在启动 Keepalived 服务后，由于自身角色为 Backup，所以会首先进入 Backup 状态，接着也会运行 VRRP_Script 模块检查 httpd 服务的运行状态，如果 httpd 服务正常，将输出“succeeded”。

在备用节点查看下/var/log/keepalived-mysql-state.log 日志文件，内容如下：

[root@keepalived-backup keepalived]#tail -f /var/log/keepalived-mysql-state.log

[Backup]

Tue Mar 4 17:27:15 CST 2014

由此可知，备用节点在切换到 Backup 状态后，执行了/etc/keepalived/backup.sh 这脚本。

3. Keepalived 的故障切换过程分析

下面开始测试一下 Keepalived 的故障切换（failover）功能，首先在 keepalived-master 节点关闭 httpd 服务，然后看看 Keepalived 是如何实现故障切换的。

在 keepalived-master 节点关闭 httpd 服务后，紧接着查看 Keepalived 运行日志，操作如下图所示。

```
[root@keepalived-master keepalived]# killall -9 httpd
[root@keepalived-master keepalived]# tail -f /var/log/messages
Mar  4 18:22:17 keepalived-master Keepalived_vrrp[24348]: VRRP_Script(check_httpd) failed
Mar  4 18:22:19 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Entering FAULT STATE
Mar  4 18:22:19 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) removing protocol VIPs.
Mar  4 18:22:19 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Now in FAULT state
Mar  4 18:22:19 keepalived-master avahi-daemon[1315]: Withdrawing address record for 192.168.66.80 on eth0.
Mar  4 18:22:19 keepalived-master Keepalived_healthcheckers[24347]: Netlink reflector reports IP 192.168.66.80
removed
```

从日志可以看出，在 keepalived-master 节点的 httpd 服务被关闭后，VRRP_Script 模块很快就能检测到，然后进入了 Fault 状态，最后将虚拟 IP 地址从 eth0 上移除。

紧接着查看 keepalived-backup 节点上 Keepalived 运行日志，信息如下图所示。

```
Mar 4 18:24:00 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) Transition to MASTER STATE
Mar 4 18:24:02 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) Entering MASTER STATE
Mar 4 18:24:02 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) setting protocol VIPs.
Mar 4 18:24:02 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) Sending gratuitous ARPs on eth0 for 192.168.66.80
Mar 4 18:24:02 keepalived-backup avahi-daemon[1207]: Registering new address record for 192.168.66.80 on eth0.IPv4.
Mar 4 18:24:02 keepalived-backup Keepalived_healthcheckers[27792]: Netlink reflector reports IP 192.168.66.80 added
Mar 4 18:24:07 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) Sending gratuitous ARPs on eth0 for 192.168.66.80
```

从日志可以看出，在 keepalived-master 节点出现故障后，备用节点 keepalived-backup 立刻检测到，此时备用机变为 Master 角色，并且接管了 keepalived-master 主机的虚拟 IP 资源，最后将虚拟 IP 绑定在 eth0 设备上。

Keepalived 在发生故障时进行切换的速度是非常快的，只有几秒钟的时间，如果在切换过程中，持续 ping 虚拟 IP 地址，几乎没有延时等待时间。

4. 故障恢复切换分析

由于设置了集群中的主、备节点角色，因此，主节点在恢复正常后会自动再次从备用节点夺取集群资源，这是常见高可用集群系统的运行原理。下面继续演示下故障恢复后 Keepalived 的切换过程。

首先在 keepalived-master 节点上启动 httpd 服务：

```
[root@keepalived-master ~]# /etc/init.d/httpd  start
```

紧接着查看 Keepalived 运行日志，信息如下图所示。

```
Mar 4 20:11:58 keepalived-master Keepalived_vrrp[24348]: VRRP_Script(check_httpd) succeeded
Mar 4 20:11:58 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) prio is higher than received advert
Mar 4 20:11:58 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Transition to MASTER STATE
Mar 4 20:11:58 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Received lower prio advert, forcing new election
Mar 4 20:12:00 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Entering MASTER STATE
Mar 4 20:12:00 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) setting protocol VIPs.
Mar 4 20:12:00 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Sending gratuitous ARPs on eth0 for 192.168.66.80
Mar 4 20:12:00 keepalived-master Keepalived_healthcheckers[24347]: Netlink reflector reports IP 192.168.66.80 added
Mar 4 20:12:00 keepalived-master avahi-daemon[1315]: Registering new address record for 192.168.66.80 on eth0.IPv4.
Mar 4 20:12:05 keepalived-master Keepalived_vrrp[24348]: VRRP_Instance(HA_1) Sending gratuitous ARPs on eth0 for 192.168.66.80
```

从日志可知，keepalived-master 节点通过 vrrp_script 模块检测到 httpd 服务已经恢复正常，然后自动切换到 Master 状态，同时也夺回了集群资源，将虚拟 IP 地址再次绑定在 eth0 设备上。

继续查看 keepalived-backup 节点 Keepalived 的运行日志信息，如下图所示。

```
Mar 4 20:13:51 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) Received higher prio advert
Mar 4 20:13:51 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) Entering BACKUP STATE
Mar 4 20:13:51 keepalived-backup Keepalived_vrrp[27793]: VRRP_Instance(HA_1) removing protocol VIPs.
Mar 4 20:13:51 keepalived-backup Keepalived_healthcheckers[27792]: Netlink reflector reports IP 192.168.66.80
removed
Mar 4 20:13:51 keepalived-backup avahi-daemon[1207]: Withdrawing address record for 192.168.66.80 on eth0
```

从上面图中可以看出，keepalived-backup 节点在发现主节点恢复正常后，释放了集群资源，重新进入了 Backup 状态，于是整个集群系统恢复了正常的主、备运行状态。

纵观 Keepalived 的整个运行过程和切换过程，看似合理，事实上并非如此：在一个高负载、高并发、追求稳定的业务系统中，执行一次主、备切换对业务系统影响很大，因此，不到万不得已，尽量不要进行主、备角色的切换，也就是说，在主节点发生过程后，必须要切换到备用节点，而在主节点恢复后，不希望再次切回主节点，直到备用节点发生故障时才进行切换，这就是前面绍过的不抢占功能，可以通过 Keepalived 的“nopreempt”选项来实现。

18.4.2 通过 vrrp_script 实现对集群资源的监控

在上节介绍 Keepalived 基础 HA 功能时用到了 vrrp_script 这个模块，此模块专门用于对集群中服务资源进行监控。与此模块一起使用的还有 track_script 模块，在此模块中可以引入监控脚本、命令组合、shell 语句等，以实现对服务、端口多方面的监控。track_script 模块主要用来调用“vrrp_script”模块使 Keepalived 执行对集群服务资源的检测。

此外在 vrrp_script 模块中还可以定义对服务资源检测的时间间隔、权重等参数，通过 vrrp_script 和 track_script 组合，可以实现对集群资源的监控并改变集群优先级，进而实现 Keepalived 的主、备节点切换。

下面就详细介绍下 vrrp_script 模块常见的几种监测机制，至于选择哪种监控方面，视实际应用环境而定。

1. 通过 killall 命令探测服务运行状态

这种监控集群服务的方式主要是通过 killall 命令实现的。killall 会发送一个信号到正在运行的指定命令的进程。如果没指定信号名，则发送 SIGTERM。SIGTERM 也是信号名的一种，代号为 15，它表示以正常的方式结束程序的运行。其实 killall 可用的信号名有很多，可通过“killall -l”命令显示所有信号名列表，其中每个信号名代表对进程的不同执行方式，例如，代号为 9 的信号表示将强制中断一个程序的运行。**这里要用到的信号为 0，代号为 0 的信号并不表示要关闭某个程序，而表示对程序（进程）的运行状态进行监控，如果发现进程关闭或其他异常，将返回状态码 1，反之，如果发现进程运行正常，将返回状态码 0。**vrrp_script 模块正是利用了 killall 命令的这个特性，变相的实现了对服务运行状态的监控。下面看一个实例：

```
vrrp_script check_mysqld {
    script "killall -0 mysqld"
    interval 2
    }
track_script {
    check_mysqld
    }
```

在这个例子中，定义了一个服务监控模块 check_mysqld，其采用的监控的方式是通过“killall -0 mysqld”的方式，其中“interval”选项检查的时间间隔，即 2 秒钟执行一次检测。

在 mysql 服务运行正常情况下，通过 killall 命令检测结果如下：

```
[root@keepalived-master ~]# killall -0 mysqld
[root@keepalived-master ~]# echo $?
0
```

这里通过“echo $?”方式显示了上个命令的返回状态码，mysql 服务运行正常，因此返回的状态码为 0，此时 check_mysqld 模块将返回服务检测正常的提示。接着将 mysql 服务关闭，再次执行检测，结果如下：

```
[root@keepalived-master ~]#     killall -0 mysqld
```

```
mysqld: no process killed
[root@keepalived-master ~]# echo $?
1
```

由于 mysql 服务被关闭，因此返回的状态码为 1，此时 check_mysqld 模块将返回服务检测失败的提示。然后根据 vrrp_script 模块中设定的 “weight” 值重新设置 Keepalived 主、备节点的优先级，进而引发主、备节点发生切换。

从这个过程可以看到，vrrp_script 模块其实并不关注监控脚本或监控命令是如何实现的，它仅仅通过监控脚本的返回状态码来识别集群服务是否正常，如果返回状态码为 0，那么就认为服务正常，如果返回状态码为 1，则认为服务故障。明白了这个原理之后，在进行自定义监控脚本的时候，只需按照这个原则来编写即可。

2. 检测测端口运行状态

检测端口的运行状态，也是最常见的服务监控方式，在 Keepalived 的 vrrp_script 模块中可以通过如下方式对本机的端口进行检测：

```
vrrp_script check_httpd {
    script "</dev/tcp/127.0.0.1/80"
    interval 2
    fall    2
    rise    1
    }
track_script {
     check_httpd
     }
```

在这个例子中，通过 “</dev/tcp/127.0.0.1/80” 这样的方式定义了一个对本机 80 端口的状态检测，其中，“fall” 选项表示检测到失败的最大次数，也就是说，如果请求失败两次，就认为此节点资源发生故障，将进行切换操作；“rise” 表示如果请求一次成功，就认为此节点资源恢复正常。

3. 通过 shell 语句进行状态监控

在 Keepalived 的 vrrp_script 模块中甚至可以直接引用 shell 语句进行状态监控，例如下面这个示例：

```
vrrp_script chk_httpd {
    script "if [ -f /var/run/httpd/httpd.pid ]; then exit 0; else exit 1; fi"
    interval 2
    fall    1
    rise    1
    }
track_script {
    chk_httpd
    }
```

在这个例子中，通过一个 shell 判断语句，检测 httpd.pid 文件是否存在，如果存在，就认为状态正常，否则认为状态异常，这种监测方式对于一些简单的应用监控或者流程监控非常有用。从这里也可以得知，vrrp_script 模块支持的监控方式十分灵活。

4. 通过脚本进行服务状态监控

这是最常见的监控方式，其监控过程类似于 Nagios 的执行方式，不同的是，这里只有 0、1 两种返回状态，例如下面这个示例：

```
vrrp_script chk_mysqld {
    script "/etc/keepalived/check_mysqld.sh"
    interval 2
    }
    track_script {
    chk_mysqld
    }
```

其中，check_mysqld.sh 的内容为：

```
#!/bin/bash
```

```
MYSQL=/usr/bin/mysql
MYSQL_HOST=localhost
MYSQL_USER=root
MYSQL_PASSWORD='xxxxxx'

$MYSQL -h $MYSQL_HOST -u $MYSQL_USER -p$MYSQL_PASSWORD -e "show status;" > /dev/null 2>&1
if [ $? = 0 ] ;then
        MYSQL_STATUS=0
else
        MYSQL_STATUS=1
fi
        exit $MYSQL_STATUS
```

这个一个最简单的实现 mysql 服务状态检测的 shell 脚本，它通过登录 mysql 数据库后执行查询操作来检测 mysql 运行是否正常，如果检测正常，将返回状态码 0，否则返回状态码 1。其实很多在 Nagios 下运行的脚本，只要稍作修改，即可在这里使用，非常方便。

18.5 keepalived 问题排查技巧

检测两个 keepalived 主机之间是否能通信的办法有

1、停掉一个 keepalived，看另外一个 keepalived 的日志/var/log/messages 里是否有新的日志

2、用嗅探器抓包，例如：

```
tcpdump -v -i eth1 host 224.0.0.18
tcpdump -vvv -n -i eth1 host 224.0.0.18
```

其中，224.0.0.18 是 VRRP 组播使用的目的地址，默认为组播每秒发送一次。