Python开发之运维架构

讲师：王晓春

高可用性集群KEEPALIVED

本章内容

- 高可用集群
- KeepAlived 组成
- keepAlived 配置

集群Cluster

- 集群类型：
    - LB lvs/nginx（http/upstream, stream/upstream）
    - HA 高可用性
        - SPoF: Single Point of Failure
    - HPC
- 系统可用性的公式：A=MTBF/（MTBF+MTTR）
    - (0,1), 95%
    - 几个9（指标）: 99%, ..., 99.999%，99.9999%；

系统故障：

- 硬件故障：设计缺陷、wear out（损耗）、自然灾害......
- 软件故障：设计缺陷

集群Cluster

- 提升系统高用性的解决方案之降低MTTR：

  手段：冗余redundant

  active/passive　　主备

  active/active双主

  active --> HEARTBEAT --> passive

  active <--> HEARTBEAT <--> active

- 高可用的是“服务”　：

  HA nginx service：

  　　vip/nginx process[/shared storage]

  资源：组成一个高可用服务的“组件”

  (1) passive node的数量

  (2) 资源切换

集群Cluster

- shared storage：

    NAS：文件共享服务器；

    SAN：存储区域网络，块级别的共享

- Network partition：网络分区

    quorum：法定人数

        with quorum： > total/2

        without quorum: <= total/2

    隔离设备： fence

        node：STONITH = Shooting The Other Node In The Head，断电重启

        资源：断开存储的连接

集群Cluster

- TWO nodes Cluster
  - 辅助设备：ping node, quorum disk
- Failover：故障切换，即某资源的主节点故障时，将资源转移至其它节点的操作
- Failback：故障移回，即某资源的主节点故障后重新修改上线后，将之前已转移至其它节点的资源重新切回的过程
- HA Cluster实现方案:
  - ais：应用接口规范 完备复杂的HA集群
    - RHCS：Red Hat Cluster Suite红帽集群套件
    - heartbeat
    - corosync
  - vrrp协议实现：虚拟路由冗余协议
    - keepalived

KeepAlived

- keepalived：
  - vrrp协议：Virtual Router Redundancy Protocol
- 术语：
  - 虚拟路由器：Virtual Router
  - 虚拟路由器标识：VRID(0-255)，唯一标识虚拟路由器
  - 物理路由器：
    - master：主设备
    - backup：备用设备
    - priority：优先级
  - VIP：Virtual IP
  - VMAC：Virutal MAC (00-00-5e-00-01-VRID)

KeepAlived

- 通告：心跳，优先级等；周期性
- 工作方式：抢占式，非抢占式
- 安全工作：
  - 认证：
    - 无认证
    - 简单字符认证：预共享密钥
    - MD5
- 工作模式：
  - 主/备：单虚拟路径器
  - 主/主：主/备（虚拟路径器1），备/主（虚拟路径器2）

KeepAlived

- keepalived:

  vrrp协议的软件实现，原生设计目的为了高可用ipvs服务

- 功能：
  - vrrp协议完成地址流动
  - 为vip地址所在的节点生成ipvs规则(在配置文件中预先定义)
  - 为ipvs集群的各RS做健康状态检测
  - 基于脚本调用接口通过执行脚本完成脚本中定义的功能，进而影响集群事务，以此支持nginx、haproxy等服务

KeepAlived

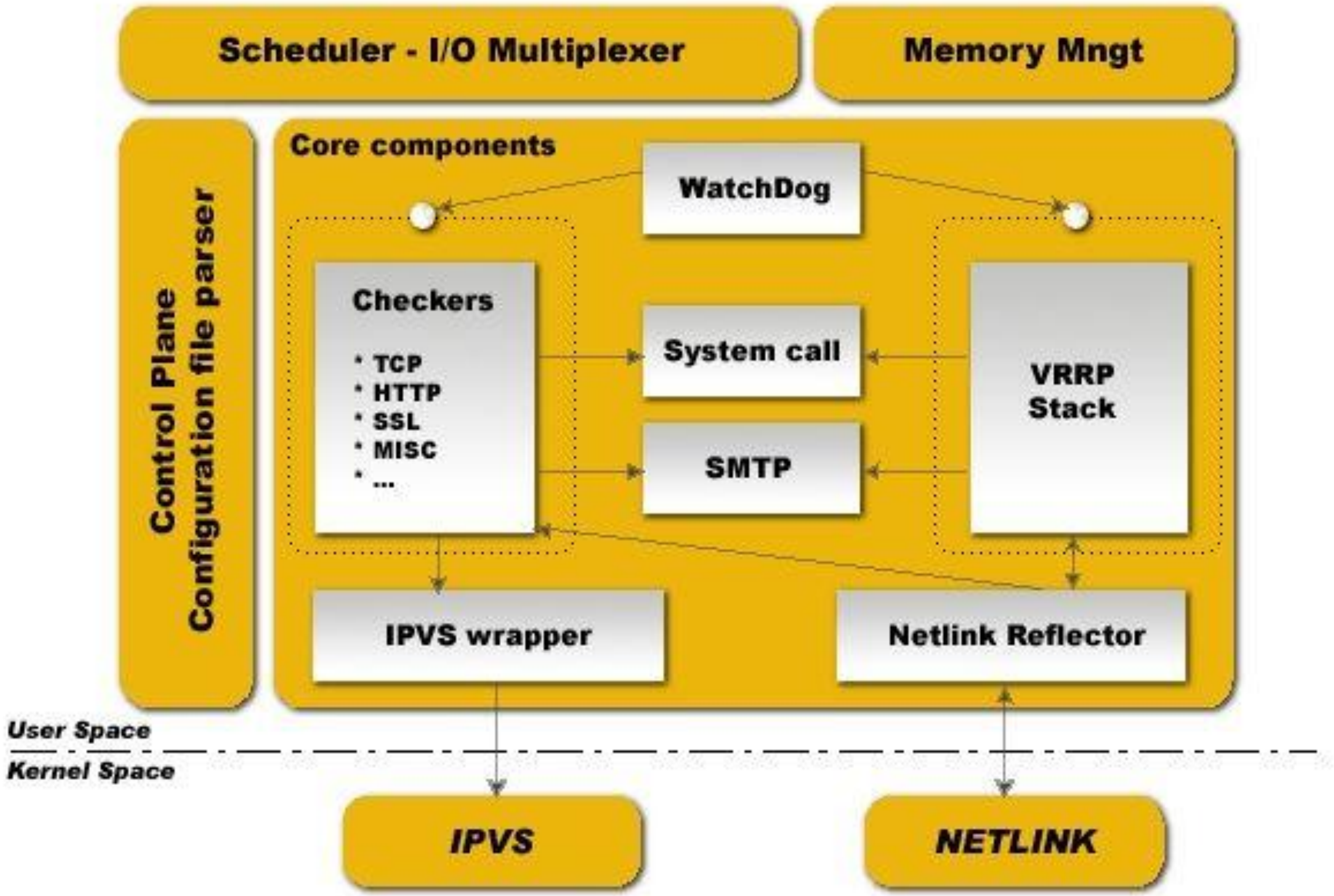
- 组件：
  - 核心组件：

    vrrp stack

    ipvs wrapper

    checkers
  - 控制组件：配置文件分析器
  - IO复用器
  - 内存管理组件

KeepAlived组成



KeepAlived实现

- HA Cluster 配置准备：
  - (1) 各节点时间必须同步

    ntp, chrony
  - (2) 确保iptables及selinux不会成为阻碍
  - (3) 各节点之间可通过主机名互相通信（对KA并非必须）

    建议使用/etc/hosts文件实现
  - (4) 各节点之间的root用户可以基于密钥认证的ssh服务完成互相通信（对KA并非必须）

KeepAlived实现

- keepalived安装配置：

  CentOS 6.4+ Base源

- 程序环境：
  - 主配置文件：/etc/keepalived/keepalived.conf
  - 主程序文件：/usr/sbin/keepalived
  - Unit File：/usr/lib/systemd/system/keepalived.service
  - Unit File的环境配置文件：/etc/sysconfig/keepalived

KeepAlived配置

- 配置文件组件部分：
- TOP HIERACHY

    GLOBAL CONFIGURATION

        Global definitions

        Static routes/addresses

    VRRPD CONFIGURATION

        VRRP synchronization group(s)：vrrp同步组

        VRRP instance(s)：即一个vrrp虚拟 路由器

    LVS CONFIGURATION

        Virtual server group(s)

        Virtual server(s)：ipvs集群的vs和rs

KeepAlived配置

- 配置语法：
- 配置虚拟路由器：

```
vrrp_instance <STRING> {
        ....
}
```

- 专用参数：

state MASTER|BACKUP：当前节点在此虚拟路由器上的初始状态；只能有一个是MASTER，余下的都应该为BACKUP

interface IFACE_NAME：绑定为当前虚拟路由器使用的物理接口

virtual_router_id VRID：当前虚拟路由器惟一标识，范围是0-255

priority 100：当前物理节点在此虚拟路由器中的优先级；范围1-254

advert_int 1：vrrp通告的时间间隔，默认1s

KeepAlived配置

```
authentication { #认证机制
        auth_type AH|PASS
        auth_pass <PASSWORD>  仅前8位有效
}
virtual_ipaddress { #虚拟IP
        <IPADDR>/<MASK> brd <IPADDR> dev <STRING> scope <SCOPE> label <LABEL>
        192.168.200.17/24 dev eth1
        192.168.200.18/24 dev eth2 label eth2:1
}
track_interface { #配置监控网络接口，一旦出现故障，则转为FAULT状态实现地址转移
        eth0
        eth1
        …
}
```

KeepAlived配置

- nopreempt：定义工作模式为非抢占模式
- preempt_delay 300：抢占式模式，节点上线后触发新选举操作的延迟时长，默认模式
- 定义通知脚本：

  notify_master <STRING>|<QUOTED-STRING>：

  当前节点成为主节点时触发的脚本

  notify_backup <STRING>|<QUOTED-STRING>：

  当前节点转为备节点时触发的脚本

  notify_fault <STRING>|<QUOTED-STRING>：

  当前节点转为“失败”状态时触发的脚本

  notify <STRING>|<QUOTED-STRING>：

  通用格式的通知触发机制，一个脚本可完成以上三种状态的转换时的通知

KeepAlived单主配置示例

- 单主配置示例：

```
! Configuration File for keepalived
global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from keepalived@localhost
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id node1        #主机名，在另一结点为node2
    vrrp_mcast_group4 224.0.100.100
}
```

KeepAlived单主配置示例

```
vrrp_instance VI_1 {
        state MASTER    #在另一个结点上为BACKUP
        interface eth0
        virtual_router_id 6   #多个节点必须相同
        priority 100      #在另一个结点上为90
        advert_int 1            #通告间隔1s
        authentication {
                auth_type PASS     #预共享密钥认证
                auth_pass 571f97b2
        }
        virtual_ipaddress {
                172.18.100.66/16 dev eth0 label eth0:0
        }
        track_interface {
                eth0
        }
}
```

KeepAlived双主配置

双主模型示例：

```
! Configuration File for keepalived
global_defs {
        notification_email {
                root@localhost
        }
        notification_email_from keepalived@localhost
        smtp_server 127.0.0.1
        smtp_connect_timeout 30
        router_id node1
        vrrp_mcast_group4 224.0.100.100
}
```

KeepAlived双主配置

```
vrrp_instance VI_1 {
        state MASTER
        interface eth0
        virtual_router_id 6
        priority 100
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 571f97b2
        }
        virtual_ipaddress {
                172.16.0.10/16 dev eth0
        }
}
```

KeepAlived双主配置

```
vrrp_instance VI_2 {
        state BACKUP
        interface eth0
        virtual_router_id 8
        priority 98
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 578f07b2
        }
        virtual_ipaddress {
                172.16.0.11/16 dev eth0
        }
}
```

示例通知脚本

```
#!/bin/bash
#
contact='root@localhost'
notify() {
        mailsubject="$(hostname) to be $1, vip floating"
        mailbody="$(date +'%F %T'): vrrp transition, $(hostname) changed to be $1"
        echo "$mailbody" | mail -s "$mailsubject" $contact
}
case $1 in
master)
        notify master
        ;;
backup)
        notify backup
        ;;
fault)
        notify fault
        ;;
*)
        echo "Usage: $(basename $0) {master|backup|fault}"
        exit 1
        ;;
esac
```

KeepAlived双主配置

- 脚本的调用方法：

```
notify_master "/etc/keepalived/notify.sh master"
notify_backup "/etc/keepalived/notify.sh backup"
notify_fault "/etc/keepalived/notify.sh fault"
```

KeepAlived支持IPVS

- 虚拟服务器：
- 配置参数：

```
virtual_server IP port |
virtual_server fwmark int
{
        …
        real_server {
        …
}
        …
}
```

常用参数

- delay_loop <INT>：检查后端服务器的时间间隔
- lb_algo rr|wrr|lc|wlc|lblc|sh|dh：定义调度方法
- lb_kind NAT|DR|TUN：集群的类型
- persistence_timeout <INT>：持久连接时长
- protocol TCP：服务协议，仅支持TCP
- sorry_server <IPADDR> <PORT>：所有RS故障时，备用服务器地址
- real_server <IPADDR> <PORT>

```
{
        weight <INT>     RS权重
        notify_up <STRING>|<QUOTED-STRING>    RS上线通知脚本
        notify_down <STRING>|<QUOTED-STRING> RS下线通知脚本
        HTTP_GET|SSL_GET|TCP_CHECK|SMTP_CHECK|MISC_CHEC K { ... }：定义当前主机的健康状态检测方法
}
```

KeepAlived配置检测

- HTTP_GET|SSL_GET：应用层检测

```
HTTP_GET|SSL_GET {
        url {
            path <URL_PATH>：定义要监控的URL
            status_code <INT>：判断上述检测机制为健康状态的响应码
            digest <STRING>：判断为健康状态的响应的内容的校验码
        }
connect_timeout <INTEGER>：连接请求的超时时长
nb_get_retry <INT>：重试次数
delay_before_retry <INT>：重试之前的延迟时长
connect_ip <IP ADDRESS>：向当前RS哪个IP地址发起健康状态检测请求
connect_port <PORT>：向当前RS的哪个PORT发起健康状态检测请求
bindto <IP ADDRESS>：发出健康状态检测请求时使用的源地址
bind_port <PORT>：发出健康状态检测请求时使用的源端口
}
```

KeepAlived配置检测

- TCP_CHECK {

  connect_ip <IP ADDRESS>：向当前RS的哪个IP地址发起健康状态检测请求

  connect_port <PORT>：向当前RS的哪个PORT发起健康状态检测请求

  bindto <IP ADDRESS>：发出健康状态检测请求时使用的源地址

  bind_port <PORT>：发出健康状态检测请求时使用的源端口

  connect_timeout <INTEGER>：连接请求的超时时长

}

单主模型IPVS示例

- 高可用的ipvs集群示例：

```
! Configuration File for keepalived
global_defs {
        notification_email {
                root@localhost
        }
        notification_email_from keepalived@localhost
        smtp_server 127.0.0.1
        smtp_connect_timeout 30
        router_id node1
        vrrp_mcast_group4 224.0.100.10
}
```

单主模型IPVS示例

```
vrrp_instance VI_1 {
        state MASTER
        interface eth0
        virtual_router_id 6
        priority 100
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 571f97b2
        }
        virtual_ipaddress {
                172.16.0.10/16 dev eth0
        }
        notify_master "/etc/keepalived/notify.sh master"
        notify_backup "/etc/keepalived/notify.sh backup"
        notify_fault "/etc/keepalived/notify.sh fault"
}
```

单主模型IPVS示例

```
virtual_server 172.16.0.10 80 {
    delay_loop 3
    lb_algo rr
    lb_kind DR
    protocol TCP
    sorry_server 127.0.0.1 80
    real_server 172.16.0.11 80 {
        weight 1
        HTTP_GET {
            url {
                path /
                status_code 200
            }
        connect_timeout 1
        nb_get_retry 3
        delay_before_retry 1
        }
    }
```

单主模型IPVS示例

```
real_server 172.16.0.12 80 {
        weight 1
        HTTP_GET {
                url {
                        path /
                        status_code 200
                }
        connect_timeout 1
        nb_get_retry 3
        delay_before_retry 1
        }
}
```

双主模式的lvs集群

双主模式的lvs集群，拓扑、实现过程；

配置示例（一个节点）：

```
! Configuration File for keepalived
global_defs {
notification_email {
root@localhost
}
notification_email_from kaadmin@localhost
smtp_server 127.0.0.1
smtp_connect_timeout 30
router_id node1
vrrp_mcast_group4 224.0.100.100
}
}
```

双主模式的lvs集群

```
vrrp_instance VI_1 {
state MASTER
interface eth0
virtual_router_id 6
priority 100
advert_int 1
authentication {
        auth_type PASS
        auth_pass f1bf7fde
}
virtual_ipaddress {
        172.16.0.80/16 dev eth0 label eth0:0
}
```

双主模式的lvs集群

```
track_interface {
          eth0
}
notify_master "/etc/keepalived/notify.sh master"
notify_backup "/etc/keepalived/notify.sh backup"
notify_fault "/etc/keepalived/notify.sh fault"
}
```

双主模式的lvs集群

```
vrrp_instance VI_2 {
        state BACKUP
        interface eth0
        virtual_router_id 8
        priority 98
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass f2bf7ade
        }
```

双主模式的lvs集群

```
virtual_ipaddress {
        172.16.0.90/16 dev eth0 label eth0:1
}
track_interface {
        eth0
}
notify_master "/etc/keepalived/notify.sh master"
notify_backup "/etc/keepalived/notify.sh backup"
notify_fault "/etc/keepalived/notify.sh fault"
}
```

双主模式的lvs集群

```
virtual_server fwmark 3 {
delay_loop 2
lb_algo rr
lb_kind DR
nat_mask 255.255.0.0
protocol TCP
sorry_server 127.0.0.1 80
real_server 172.16.0.11 80 {
        weight 1
        HTTP_GET {
        url {
                path /
                status_code 200
        }
        connect_timeout 2
        nb_get_retry 3
        delay_before_retry 3
        }
}
```

双主模式的lvs集群

```
real_server 172.16.0.12 80 {
        weight 1
        HTTP_GET {
                url {
                        path /
                        status_code 200
                }
                connect_timeout 2
                nb_get_retry 3
                delay_before_retry 3
                }
        }
}
```

keepalived调用脚本进行资源监控

- keepalived调用外部的辅助脚本进行资源监控，并根据监控的结果状态能实现优先动态调整
- vrrp_script:自定义资源监控脚本，vrrp实例根据脚本返回值，公共定义，可被多个实例调用，定义在vrrp实例之外
- track_script:调用vrrp_script定义的脚本去监控资源，定义在实例之内，调用事先定义的vrrp_script
- 分两步：(1) 先定义一个脚本；(2) 调用此脚本

```
vrrp_script <SCRIPT_NAME> {
    script ""
    interval INT
    weight -INT
}
track_script {
    SCRIPT_NAME_1
    SCRIPT_NAME_2
}
```

示例：高可用nginx服务

```
! Configuration File for keepalived
global_defs {
        notification_email {
                root@localhost
        }
        notification_email_from keepalived@localhost
        smtp_server 127.0.0.1
        smtp_connect_timeout 30
        router_id node1
        vrrp_mcast_group4 224.0.100.100
}
```

示例：高可用nginx服务

```
vrrp_script chk_down {
        script "[[ -f /etc/keepalived/down ]] && exit 1 || exit 0"
        interval 1
        weight -20
}
vrrp_script chk_nginx {
        script "killall -0 nginx && exit 0 || exit 1"
        interval 1
        weight -20
        fall 2     #2次检测失败为失败
        rise 1     #1次检测成功为成功
}
```

示例：高可用nginx服务

```
vrrp_instance VI_1 {
        state MASTER
        interface eth0
        virtual_router_id 14
        priority 100
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 571f97b2
        }
        virtual_ipaddress {
                172.18.0.93/16 dev eth0
        }
        track_script {
                chk_down
                chk_nginx
        }
        notify_master "/etc/keepalived/notify.sh master"
        notify_backup "/etc/keepalived/notify.sh backup"
        notify_fault "/etc/keepalived/notify.sh fault"
}
```

同步组

- LVS NAT模型VIP和DIP需要同步，需要同步组
- vrrp_sync_group VG_1 {

```
        group {
                VI_1   # name of vrrp_instance (below)
                VI_2   # One for each moveable IP.
        }
}
vrrp_instance VI_1 {
        eth0
        vip
 }
vrrp_instance VI_2 {
        eth1
        dip
 }
```

作业

- keepalived 单实例，高可用性IPVS集群
  IPVS集群提供php，如phpwind
- keepalived双主nginx
- vrrp_script 高可用性nginx

关于马哥教育

- ◆博客：http://mageedu.blog.51cto.com
- ◆主页：http://www.magedu.com
- ◆QQ：1661815153, 113228115
- ◆QQ群：203585050, 279599283

祝大家学业有成

谢　谢

咨询热线 400-080-6560