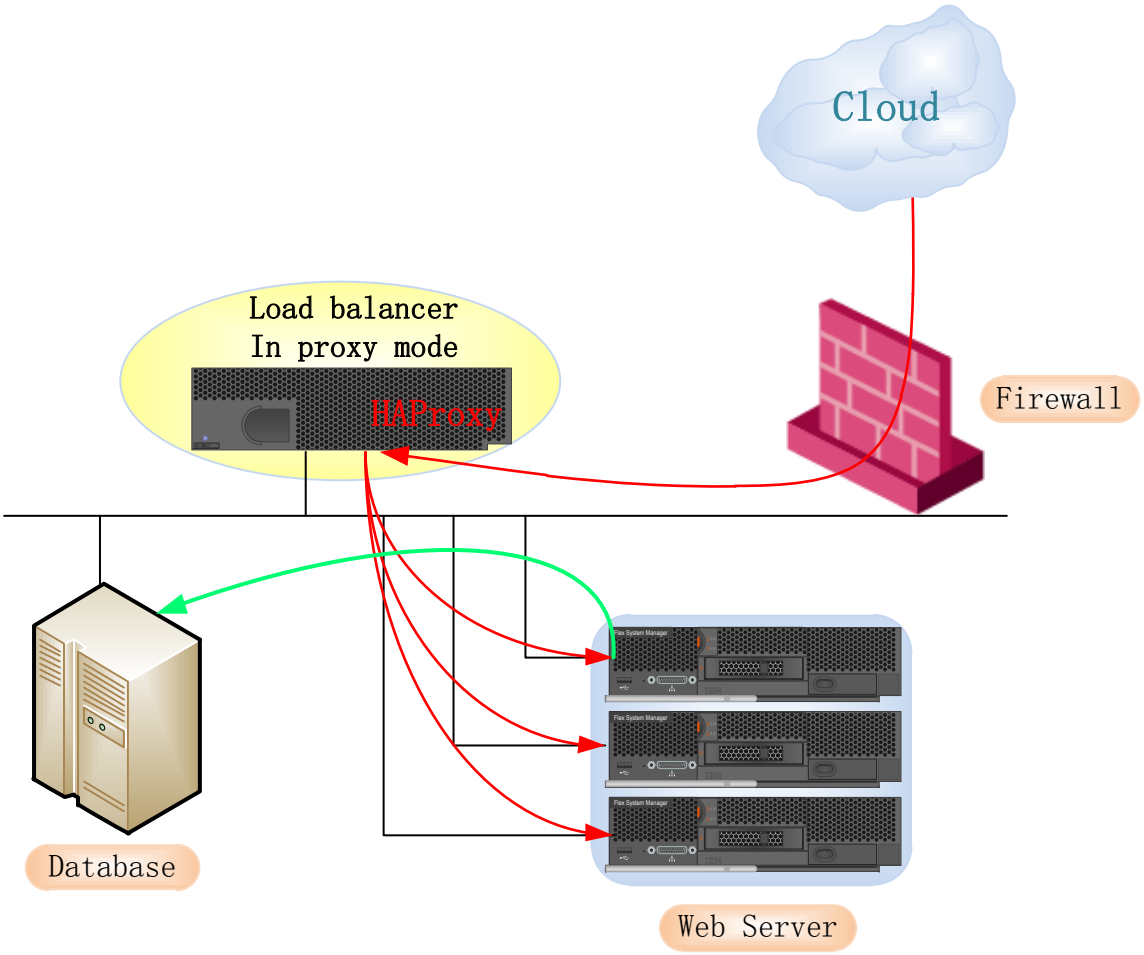
负载均衡代理—HAProxy

一、HAProxy 简述

在众多的负载均衡解决方案中，有基于硬件的负载均衡设备，例如 F5，Big-Ip 等硬件，也有基于软件的负载均衡产品，如 LVS、Nginx 以及本文的 HAProxy 等。在软件的负载均衡产品中，又分为两种实现方式，分别是基于操作系统的实现和基于第三方应用的实现。LVS 就是基于 Linux 操作系统的一种软负载均衡实现方式，而 HAProxy 就是基于第三方应用的软负载均衡实现。HAProxy 是基于 TCP(四层)和 HTTP（七层）应用的负载均衡软件。



HAProxy 实现了一种事件驱动、单一进程模型，此模型支持非常大的并发连接数。多进程或多线程模型受内存限制 、系统调度器限制以及无处不在的锁限制，很少能处理数千并发连接。而事件驱动模型因为有更好的资源和时间管理用户端(User-Space) 实现所有这些任务，所以没有这些问题。此模型的弊端是，在多核系统上，这些程序通常扩展性较差。这就是为什么他们必须进行优化以 使每个 CPU 时间片(Cycle)做更多的工作。

HAProxy 现多于线上的 Mysql 集群环境和 web 请求的读写分离，常用于作为 MySQL（读）负载均衡。

1、HAProxy 优点：

(1)可靠性与稳定性都非常出色，可与硬件级设备媲美。

(2)支持多种负载均衡调度算法，并且也支持 session 保持，客户端侧的长连接。

(3)理论上 1G 可用 RAM 空间可维持 4W-5W 个并发连接，性能强大可见一斑。

(4)拥有功能强大的后端服务器的状态监控 web 页面，可以实时了解设备的运行状态。

(5)HAProxy 还拥有功能强大的 ACL(Access Control List 访问控制列表)支持。

2、HAProxy 的负载均衡算法：

(1)动态调度算法，支持动态调整权重，可以在运行中调整而不用重启服务；

a、rr(轮询，round robin)在服务器间公平调度，每个后端服务器最多可承载 4128 个并发链接；

b、leastconn：将新的连接转发至后端连接数较少的服务器上；

c、source：将连接请求的原地址进行 hash 计算，并有后端服务的权重总数相处转发至匹配的服务器，选择 consistent(一致性 hash)方法，则为动态调度，若是 Map-hash(取模法)，则为静态调度算法。

(2)静态调度算法，支持权重，不支持动态调整，调整后需重启服务；

a、static-rr，表示根据权重将请求转发至后端主机

b、uri：对 uri 的左半部分或对整个 URI 进行 hash 计算，并有服务器的总权重相除转发至匹配后的某个服务器，特别适用于代理缓存服务应用场景（注：红色的即为 URI 的左半部分，蓝色的即为 URI 的右半部分）

URL syntax:

<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>

⑥url_param：通过<argument>为 URI 指定的参数在每个 HTTP　GET 请求中将会被检索，若找到指定的参数并且其通过"="被赋予某一个值，那么此值将被执行 hash 运算并被服务器的总权重相除转发至后端某个匹配的服务器。

<scheme>://<user>:<password>@<host>:<port>/<path>;**<params>**?<query>#<frag>

http://www.mywebsite.com/sj/test;id=**8079**?name=sviergn&x=true&stuff

⑦hdr(name)：根据报文中指定的 header(如 Useragent，reference，hostname)进行调度

⑧rdp-cookie(name)，表示根据 cookie(name)来锁定并哈希每一次 TCP 请求。

二、HAProxy 配置文件简述(参考官网：http://cbonte.github.io/haproxy-dconv/)

HAProxy 配置文件中分五大部分：

(1)**global**：全局配置参数，进程级的，用来控制 Haproxy 启动前的一些进程及系统设置

(2)**defaults**：配置一些默认的参数，可以被 frontend，backend，listen 段继承使用，配置默认

配置参数可由下一个“defaults”所重新设定。

(3)**frontend**：定义一系列面向客户端监听的套接字(IP:PORT)，这些套接字可接受客户端请求并与之建立连接，以及匹配接收客户所请求的域名，uri 等，并针对不同的匹配，做不同的请求处理。

(4)**backend**：后端服务器组的定义，以及对后端服务器的一些权重、队列、连接数等选项的设置，可将其理解为 Nginx 中的 upstream 块

(5)**listen**：通过关联前端(fronted)和后端(backend)定义了一个完整的代理。

1、global 全局配置段选项介绍

```
#---------------------------进程管理及安全相关的参数---------------------------#
#deamon                         #haproxy 以守护进程的方式运行与后台，也可在命令行中加参数"-db"将其禁用
#chroot <dir>                   #指定 haproxy 运行的路径，要确保指定的目录为空目录且任何用户均不能有写权限
#group <group name>             #指定启动 haproxy 进程的用户组名
#user <user name>               #指定启动 haproxy 进程的用户名
#log <adress><facility> [max level [min level]]      #定义全局的 syslog 服务器，最多可定义两个。
#log-send-hostname [<string>]      #在 syslog 首部添加主机名称，可以为"string"指定的名称，也可以使用缺省使用当前主机名称
#stats socket <path>                #定义统计信息保存位置。
#pidfile          #定义 pid 文件的路径
#gid <number>     #以指定的 GID 运行 haproxy，建议使用 haproxy 专用的 GID(188)
#uid <number>     #以指定的 UID 运行 haproxy，建议使用 haproxy 专用的 UID(188)
#node             #定义当前节点的名称，用于 HA 场景中多 haproxy 进程共享一个 IP 地址时
#description      #当前实例的描述信息;
#---------------------------------性能调整相关的参数---------------------------------#
#nbproc <number>            #指定启动的 haproxy 进程的个数，只能用于守护进程模式的 haproxy；默认只启动一个进程，在单进程仅能打开少数文件描述符的场景中建议使用多进程模式
#maxconn <number>           #设定每个 haproxy 进程所能接受的最大并发连接数，等同于命令行选项"-n"
```

2、default 配置段选项介绍

此部分设置的参数值，默认会自动引用到下面的 frontend、backend、listen 部分中，因此，某些参数属于公用的配置，只需要在 defaults 部分添加一次即可。而如果 frontend、backend、listen 部分也配置了与 defaults 部分一样的参数，defaults 部分参数对应的值自动被覆盖

```
#---------------------------日志管理和进程相关的参数---------------------------#
```

```
#mode <http|tcp|health>                #设置 haproxy 的运行模式(有三种)，tcp 模式下客户端和服务器之间建立全双工的连接，不会对七层报文做任何检测，默认为 tcp 模式，常用与 SSL、SSH、SMTP 等；http 模式下客户端请求转发至后端服务器之前会深度分析，所有与 RFC 不兼容的请求会被拒绝；health 不常用。
#log global                            #继承全局段的日志配置信息
#option httplog                        #打开记录 http 请求的日志功能
#option dontlog                        #产生空连接将不会记录到日志内
#option http-server-close              #打开 http 协议中服务器端关闭功能，使得支持长连接，使得会话可以被重用，使得每一个日志记录都会被记录
#option forwardfor except 127.0.0.0/8      #如果后端服务器的应用程序想记录客户端的真实 IP 地址，haproxy 会把客户端的 IP 信息发送给后端服务器，在 HTTP 请求中添加”X-Forwarded-For”字段,但当是 haproxy 自身的健康检测机制去访问后端服务器时是不应该把这样的访问日志记录到日志中的,所以用来排除 127.0.0.0,即 haproxy 身。
#option redispatch                     #当与后端服务器的会话失败(服务器故障或其他原因)时，把会话重新分发到其他健康的服务器上。当原来故障的服务器恢复时，会话又被定向到已恢复的后端服务器上
#retries 3                             #向后端服务器尝试连接的最大次数，超过此值就认为后端服务器不可用。
#option abortonclose                   #当 haproxy 负载很高时，自动结束掉当前队列处理比较久的链接。

#-------------------------------C/S 超时相关的参数---------------------------------#

#timout http-request 10s   #客户端发送 http 请求的超时时间。
#timeout queue 1m          #当后端服务器在高负载响应 haproxy 时,会把 haproxy 发送来的请求放进一个队列中,该选项定义放入这个队列的超时时间。
#timeout connect 5s        #haproxy 与后端服务器连接超时时间，如果在同一个局域网可设置较小的时间。
#timeout client 1m         #定义客户端与 haproxy 连接后数据传输完毕不再有数据传输,即非活动连接超时时间。
#timeout server 1m         #定义 haproxy 与上游服务器非活动连接的超时时间。
#timeout http-keep-alive 10s       #设置新的 http 请求连接建立的最大超时时间，时间较短时可节约资源。
#timeout check 10s         #健康检测的时间的最大超时时间。
#timeout connect 5000      #设置成功连接到一台服务器的最长等待时间，默认单位是毫秒
#timeout client 3000       #设置连接客户端发送数据时的成功连接最长等待时间，默认单位是毫秒
#timeout server 3000       #设置服务器端回应客户度数据发送的最长等待时间，默认单位是毫秒
#maxconn 3000              #指定此服务器接受的最大并发连接数;如果发往此服务器的连接数目高于此处指定的值，其将被放置于请求队列，以等待其它连接被释放;

#-------------------------------设置负载均衡算法---------------------------------#

#balance <roundrobin|static-rr|source|leastconn|uri|uri_param|hdr|rdp-cookie(name)>
#在八中负载均衡算法中选一个，算法的具体含义见上文。
```

3、frontend 配置段选项介绍

frontend 根据任意 HTTP 请求头内容做 ACL 规则匹配，然后把请求定向到相关的 backend，在很大程度上简化了 haproxy 配置文件的负载性。

(1)ACL 概念

Access Control List(访问控制列表)，ACL 使用包过滤技术，在路由器上读取第三层及第四层包头中的信息如源地址、目的地址、源端口、目的端口等，根据预先定义好的规则对包进行过滤，从而达到访问控制的目的。根据协议类型、IP 地址、端口等特征自定义防火墙规则。

(2)ACL 的功能以及执行过程：

功能：ACL 的主要功能就是一方面保护资源节点，阻止非法用户对资源节点的访问，另一方面限制特定的用户节点所能具备的访问权限。

执行过程：一个端口执行哪条 ACL，需要按照列表中的条件语句执行顺序来判断。如果一个数据包的报头跟表中某个条件判断语句相匹配，那么后面的语句就将被忽略，不再进行检查。数据包只有在跟第一个判断条件不匹配时，它才被交给 ACL 中的下一个条件判断语句进行比较。如果匹配（假设为允许发送），则不管是第一条还是最后一条语句，数据都会立即发送到目的接口。如果所有的 ACL 判断语句都检测完毕，仍没有匹配的语句出口，则该数据包将视为被拒绝而被丢弃。这里要注意，ACL 不能对本路由器产生的数据包进行控制。

```
#----------------------------ACL 定义和常用的 ACL-----------------------------#
## ACL 的定义：Usage：acl <aclname> <criterion> [flags] [operator] <value> ...
## <criterion>：测试标准，即对什么信息发起测试；
## [flags]：目前 haproxy 的 acl 支持的标志位有 3 个：
##                -i：不区分<value>中模式字符的大小写；
##                -f：从指定的文件中加载模式；
##                --：标志符的强制结束标记，在模式中的字符串像标记符时使用；
#####HAProxy 常用的 ACL 测试标准：
acl being_scanned be_sess_rate gt 50        #测试指定后端上会话创建速率大于 50 的标准，为了避免攻击
acl too_fast fe_sess_rate ge 50             #指定的 frontend 上的会话创建速率是否满足指定的条件；常用于为 frontend 指定一个合理的会话创建速率上限以防止服务被滥用。
acl url_static       path_beg       -i /static /images /javascript /stylesheets
##  path_beg <string>  用于测试请求的 URL 是否以指定的模式开头。如上用于测试 URL 是否以/static、/images、/javascript 或/stylesheets 头。
acl url_static       path_end       -i .jpg .gif .png .css .js
##  path_end <string>  用于测试请求的 URL 是否以指定的模式结尾。如上用户测试 URL 是否以 jpg、gif、png、css 或 js 结尾
```

更多 ACL 案例参考：http://zhang789.blog.51cto.com/11045979/1873435

```
#-----------------------------监听端口相关的参数------------------------------#
fronted  main *.:<port>             #定义前端以及前端监听的端口
##定义错页面重定向，Usage：errorfile <code> <url>
```

```
#errorfile 404 /demo.php            #当用户访问 404 页面时，将连接重定向到该路径下
#errorfile 503 /503sorry.http       #当用户访问 503 页面时，将连接重定向到该路径下
#log 127.0.0.1 local3               #定义日志级别，可继承 global 设置
#bind :80,:443                      #绑定端口  只能定义在 frontend  listen
#--------------------------ACL 定义匹配转发相关的参数--------------------------#
#acl index path -i /index.html
#acl url_static path_beg -i /static /images /javascript      #地址的开始部分
#acl url_static path_end -i .jpg .gif .png .css .js          #地址的结尾
#use_backend <backend-name1> if url_static #如果条件符合上面的定义，则转发到后端主机
#use_backend <backend-name2> unless url_static      #如果条件不符合上面的定义,则转发到后端其他主机
#default_backend <backend-name3>            #默认转发的后端
#--------------------------------定义 ACL 转发举例--------------------------------#
#用法一、允许 10.0.0.0/24 的用户访问，其他用户将禁止
acl  goodguys  src  10.0.0.0/24
tcp-request content  accept  if  goodguys
tcp-request  content  reject
tcp-request content accept [{if | unless} <condition>]
Accept a connection if/unless a content inspection condition is matched
#用法二、将源 IP 为 172.16.254.52 的用户禁止、将 403 的错误重定向到其他服务器；
acl  badguy  src  172.16.254.52
block if badguy
errorloc  403  http://www.afwing.com/
#用法三、当用户访问 172.16.1.100 时，重定向到 http://www.afwing.com
acl  dstipaddr  hdr(Host) 172.16.1.100
redirect  location  http://www.afwing.com if  dstipaddr
#用法四、读写分离：
acl  read method GET
acl  read method HEAD
acl write method PUT
acl write method POST
use_backend imgservers if read
use_backend uploadservers if write
#用法五、限制某个 IP 访问的路径（文件）
acl  badguy  src  172.16.254.52
acl  denyfile  path  /index.html
http-request deny if denyfile  badguy
#用法六、动静分离
```

```
acl url_static       path_beg       -i /static /images /javascript /stylesheets
acl url_static       path_end       -i .jpg .gif .png .css .js
#或者
acl url_static       path_end       -i .jpg$ .gif$ .png$ .css$ .js$
```

4、backend 配置段选项介绍

用来定义后端服务集群的配置，真实服务器，一个 backend 对应一个或者多个实体服务器，frontend 调用的后端服务器。

```
#--------------------------------设置后端相关的参数---------------------------------#
#backend <group-name>      #定义一个名为<group-name>的后端部分，frontend 定义的请求会转发至此
#mode http                 # 设置为 http 模式
#balance source   #设置 haproxy 的调度算法为源地址 hash
#cookie SERVERID  #允许向 cookie 插入 SERVERID，每台服务器的 SERVERID 可在下面使用 cookie 关键字定义
#-----------------定义后端服务器和健康监测相关的参数-------------------------#
option httpchk GET /test/index.php          #通过 GET /test/index.php 来判断后端服务器的健康情况
server <server-name-1> IP:PORT cookie 1 check inter 2000 rise 3 fall 3 weight 1
server <server-name-3> IP:PORT cookie 3 check inter 1500 rise 3 fall 3 backup
#server 关键字来设置后端服务器；为后端服务器所设置的名称< server-name>，名称会出现在日志或警报中；
#支持端口映射[IP:PORT]；
#指定该服务器的 SERVERID 为 1[cookie 1]，第一次为此值挑选的后端服务器将在后续的请求中一直被选中，其目的在于实现持久连接的功能。
#接受健康监测[check]；
#监测的间隔时长，单位毫秒[inter 2000]；
#监测正常多少次后被认为后端服务器是可用的[rise 3]；
#监测失败多少次后被认为后端服务器是不可用的[fall 3]；
#分发的权重[weight 2]，最大值为 256，设置为 0 表示不参与负载均衡，默认为 1
#备份用的后端服务器，当正常的服务器全部都宕机后，才会启用备份服务器[backup]
```

5、listen 配置段选项介绍

用于状态页面监控，以及后端 server 检查：

```
########统计页面配置########
listen stats
    bind *:8009                    #设置 Frontend 和 Backend 的组合体，监控组的名称，按需要自定义名称
    stats enable                   #启用 web 界面检测 backend
    mode http                      #http 的 7 层模式
    option httplog                 #采用 http 日志格式
```

```
    log 127.0.0.1 local0 err        #错误日志记录
    maxconn 10                      #默认的最大连接数
    stats refresh 30s               #统计页面自动刷新时间
    stats uri /stats                #统计页面 url 地址
    stats realm XingCloud\ Haproxy #统计页面密码框上提示文本输入账号密码
    stats auth admin:admin          #设置监控页面的用户和密码:admin,可以设置多个用户名
    stats hide-version              #隐藏统计页面上 HAProxy 的版本信息
    stats admin if TRUE             #认证通过则允许管理
    acl allow src 192.168.1.0/24   #定义允许访问段
```

以上，配置文件可参考：http://blief.blog.51cto.com/6170059/1750952

http://marvin89.blog.51cto.com/9163436/1786140

http://zhang789.blog.51cto.com/11045979/1873435

http://jackyan.blog.51cto.com/2589874/1303760

三、编译安装 HAProxy 以及简单的代理实践

1、下载安装 HAProxy

在官网下载相应版本的软件：http://www.haproxy.org/download/

```
[root@vs1 ~]#wget http://www.haproxy.org/download/1.5/src/haproxy-1.5.15.tar.gz
##创建 haproxy 用户和用户组
[root@vs1 ~]#group -r haproxy
[root@vs1 ~]#useradd -g haproxy -M -s /sbin/nologin haproxy
##编译安装 haproxy
[root@vs1 ~]#tar zxvf haproxy-1.5.15.tar.gz
##TARGET=linux26 根据"uname -r"查看内核版本
[root@vs1 ~]#make TARGET=linux26 ARCH=x86_64 PREFIX=/usr/local/haproxy
[root@vs1 ~]#make install PREFIX=/usr/local/haproxy
##创建 haproxy 的主配置文件——haproxy.cfg
[root@vs1 ~]#mkdir /etc/haproxy
[root@vs1 ~]#cat /etc/haproxy/haproxy.cfg
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    stats socket /var/lib/haproxy/stats
```

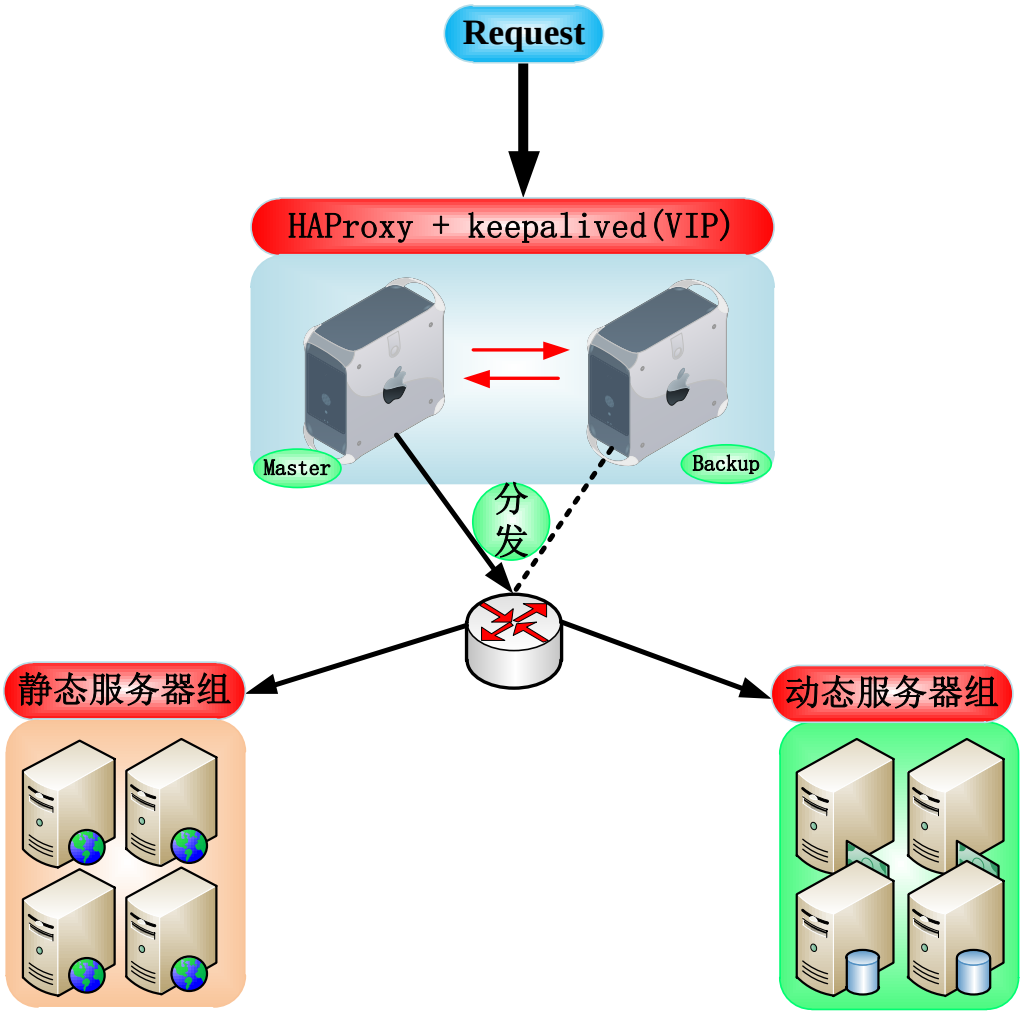
```
defaults
    mode                    http
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
frontend  main *:5000
    acl url_static       path_beg       -i /static /images /javascript /stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js
    use_backend static          if url_static
    default_backend             app
backend static
    server      static 127.0.0.1:4331 check
##为了启动的方便也可编写 startup 文件，/etc/rd.d/init.d/haproxy 具体见附录
参考链接：http://blief.blog.51cto.com/6170059/1750573
```

以上是的编译安装 haproxy，haproxy 集成在 CentOS6.4+，故也可采用 yum 安装的方式：

```
[root@vs1 ~]#yum install haproxy -y
```

2、高可用 HAProxy 代理实现 web 的动静分离

根据前文的配置文件可简单实现 web 请求的动静分离，但是 haproxy 容易成为单点故障，因此容易联想到 keepalived 实现代理的高可用。如下即为 haproxy 高可用架构图：



(1)HAProxy+keepalived 的配置

```
##此前已经详细编译配置过 keepalived，故在此使用 yum 安装 keepalived，需在两台代理上均安装：
[root@ha1 ~]#yum install haproxy keepalived -y

#-------------------------------keepalive 的配置脚本--------------------------------#
[root@ha1 ~]#cat /etc/keepalived/keepalived.conf
  ! Configuration File for keepalived
global_defs {
       notification_email {
              root@localhost
       }
       notification_email_from root@localhost
       smtp_server 127.0.0.1
       smtp_connect_timeout 30
       router_id LVS_DEVEL
       vrrp_mcast_group4 224.0.100.18
```

```
}
vrrp_script chk_haproxy {
        script "killall -0 haproxy"
        interval 1
        fall 2
        rise 1
        weight 30
}
vrrp_instance VI_1 {
        state MASTER
        interface eth1
        virtual_router_id 51
        priority 100
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 1111
        }
        track_script {
                chk_haproxy
        }
        virtual_ipaddress {
                192.168.88.111 dev eth1 label eth1:0
        }
        notify_master  "/etc/keepalived/notify.sh master"
        notify_backup  "/etc/keepalived/notify.sh backup"
        notify_fault  "/etc/keepalived/notify.sh fault"
}
#----------------------------------状态转化检测脚本----------------------------------#
[root@ha1 ~]#cat /etc/keepalived/notify.sh
#!/bin/bash
#
vip=192.168.88.111
PID=$$
FILE="/var/log/proxy.log"
TIME=`date +'%F-%H:%M:%S'`
notify(){
        echo "$TIME:`hostname`|$PID Runing the script <notify.sh>" >> $FILE
        echo "$TIME:`hostname`|$PID vrrp status changed,`hostname` became $1" >> $FILE
```

```
}
case $1 in
master)
        notify master
        echo "$TIME:`hostname`|$PID IP:$vip come bake in `hostname`" >> $FILE ;;
backup)
        notify backup
        echo "$TIME:`hostname`|$PID IP:$vip go away `hostname`" >> $FILE ;;
fault)
        notify fault
        echo "$TIME:`hostname`|$PID IP:$vip go away `hostname`" >> $FILE ;;
*)
        echo "Usage:$(basename $0) {master|backup|fault}"
        exit 1;;
esac
```

#----------haproxy 配置脚本----------#

```
[root@ha1 ~]#cat /etc/haproxy/haproxy.conf
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    stats socket /var/lib/haproxy/stats
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
```

```
    timeout client         1m
    timeout server         1m
    timeout http-keep-alive 10s
    timeout check          10s
    maxconn               3000
frontend  main *:80
    acl url_static      path_beg      -i /static /images /javascript /stylesheets
    acl url_static      path_end      -i .jpg .gif .png .css .js .html
    use_backend   static        if url_static                          ##实现动静分离
    default_backend dynamic                                            ##实现动静分离
backend static
    server  web1 192.168.88.77:8000 check
backend dynamic
    server  web2 192.168.88.66:8000 check
    server  web3 192.168.88.111:8000backup
listen stats
    bind *:8090
    mode http
    option httplog
    maxconn 10
    stats refresh 30s
    stats uri /stats
    stats realm XingCloud\ Haproxy
    stats auth admin:admin
    stats auth mc:mc
    stats admin if TRUE
##开启记录 haproxy 的日志：
[root@ha1 ~]#echo "local2.*       /var/log/haproxy.log" >> /etc/rsyslog.conf
[root@ha1 ~]#service rsyslog restart
```