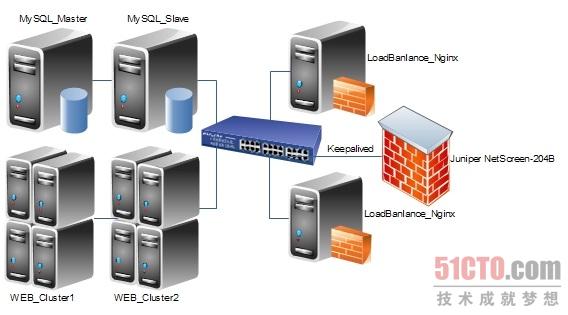
[Nginx+Keepalived说明及环境说明](http://network.51cto.com/art/201007/209823.htm)

此系统架构仅映射内网VIP的80及443端口于外网的Juniper防火墙下，其它端口均关闭，内网所有机器均关闭iptables及ipfw防火墙；外网DNS指向即通过Juniper映射出来的外网地址，而此映射的地址对映的其实是内网VIP地址。这里说下端口的问题，有的朋友可能会很疑惑，这样映射端口行不？通过项目实践得知，这样完全是可行的，php-cgi需要的9000端口及MySQL的3306端口均可走内网，完全不影响业务系统的运行。

另外，我维护的电子商务网站并发大约在1000左右，此时,Nginx+Apache集群运行得非常稳定，尤其是apache，并没有想象中那般弱；其实，在内存足够(>=8G)的情况，测试时不连数据库的话，单台apache+php5能顶得住6000并发，而且相当稳定。在网站升级架构方面，我不赞成全面淘汰生级，锦上添花式的升级会更好。

[](http://images.51cto.com/files/uploadimg/20100707/145407797.jpg)

**第一部分：Nginx+Keepalived的说明及环境说明**

喜欢看我博客或文章的朋友都知道，我一直主力推崇Nginx+Keepalived作web的负载均衡高可用架构，并积极将其用于项目方案中；Nginx负载均衡作服务器遇到的故障一般有①服务器网线松动等网络故障；②服务器硬件故障从而crash；③nginx服务死掉；遇到前二者情况，keeaplived是能起到HA的作用的；然而遇到③种情况就没有办法了，但可以通过shell监控解决这问题，从而实现真正意义上的负载均衡高可用。此篇的最新更新时间为2010年6月25号，下面将其安装步骤详细说明下：

**环境：**

* 1. centos5.3(64位)、nginx-0.7.51、keepalived-1.1.15
  2. 主nginx负载均衡器：192.168.0.154
  3. 辅nginx负载均衡器：192.168.9.155
  4. vip：192.168.0.188

[装Nginx负载均衡器及相关配置脚本](http://network.51cto.com/art/201007/209823_1.htm)

先安装Nginx负载均衡器，nginx负载的配置就用一般的模板来配置了

* 1. #添加运行nginx的用户和组www
  2. groupadd www
  3. useradd -g www www
  4. wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.8.tar.gz
  5. tar zxvf pcre-7.8.tar.gz
  6. cd pcre-7.8/
  7. ./configure
  8. make && make install
  9. wget http://sysoev.ru/nginx/nginx-0.7.51.tar.gz
  10. tar zxvf nginx-0.7.51.tar.gz
  11. cd nginx-0.7.51/
  12. ./configure --user=www --group=www --prefix=/usr/local/webserver/nginx --with-http\_stub\_status\_module --with-http\_ssl\_module
  13. make && make install

配置nginx负载均衡器的配置文件vim /usr/local/nginx/conf/nginx.conf,此篇文章仅仅只是我的某项目的配置文档，纯80转发；如果对nginx配置有https要求的可参考张宴的相关文章。

* 1. user www www;
  2. worker\_processes 8;
  4. pid /usr/local/nginx/logs/nginx.pid;
  5. worker\_rlimit\_nofile 65535;
  7. events
  8. {
  9. use epoll;
  10. worker\_connections 65535;
  11. }
  12. http{
  13. include       mime.types;
  14. default\_type application/octet-stream;
  15. server\_names\_hash\_bucket\_size 128;
  16. client\_header\_buffer\_size 32k;
  17. large\_client\_header\_buffers 4 32k;
  18. client\_max\_body\_size 8m;
  19. sendfile on;
  20. tcp\_nopush     on;
  21. keepalive\_timeout 60;
  22. tcp\_nodelay on;
  23. fastcgi\_connect\_timeout 300;
  24. fastcgi\_send\_timeout 300;
  25. fastcgi\_read\_timeout 300;
  26. fastcgi\_buffer\_size 64k;
  27. fastcgi\_buffers 4 64k;
  28. fastcgi\_busy\_buffers\_size 128k;
  29. fastcgi\_temp\_file\_write\_size 128k;
  30. gzip on;
  31. gzip\_min\_length 1k;
  32. gzip\_buffers     4 16k;
  33. gzip\_http\_version 1.0;
  34. gzip\_comp\_level 2;
  35. gzip\_types       text/plain application/x-javascript text/css application/xml;
  36. gzip\_vary on;
  38. upstream backend
  39. {
  40. server 192.168.1.102:80;
  41. server 192.168.1.103:80;
  42. server 192.168.1.105:80;
  43. }
  44. server {
  45. listen 80;
  46. server\_name www.yuhongchun027.com;
  47. location / {
  48. root /var/www ;
  49. index index.jsp index.htm index.html;
  50. proxy\_redirect off;
  51. proxy\_set\_header Host $host;
  52. proxy\_set\_header X-Real-IP $remote\_addr;
  53. proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;
  54. proxy\_pass http://backend;
  55. }
  57. location /nginx {
  58. access\_log on;
  59. auth\_basic "NginxStatus";
  60. auth\_basic\_user\_file /usr/local/nginx/htpasswd;
  61. }
  63. log\_format access '$remote\_addr - $remote\_user [$time\_local] "$request" '
  64. '$status $body\_bytes\_sent "$http\_referer" '
  65. '"$http\_user\_agent" $http\_x\_forwarded\_for';
  66. access\_log /var/log/access.log access;
  68. }
  69. }

**小节：**

第一部分和第二部分讲的是如何通过安装Nginx来达到负载均衡后端web集群的过程，Nginx能实现自动切换后端有故障的web服务器；但Nginx负载均衡器出了问题怎么办呢，它们之间是如何实现无故障转移的呢？

[Keepalived分别作web及Nginx的HA](http://network.51cto.com/art/201007/209823_2.htm)

安装keepalived，并将其做成服务模式，方便以后调试。

* 1. wget http://www.keepalived.org/software/keepalived-1.1.15.tar.gz
  2. #tar zxvf keepalived-1.1.15.tar.gz
  3. #cd keepalived-1.1.15
  4. #./configure --prefix=/usr/local/keepalived
  5. #make
  6. #make install
  7. #cp /usr/local/keepalived/sbin/keepalived /usr/sbin/
  8. #cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/
  9. #cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/
  10. #mkdir /etc/keepalived
  11. #cd /etc/keepalived/
  13. vim keepalived.conf
  14. ! Configuration File for keepalived
  15. global\_defs {
  16. notification\_email {
  17. yuhongchun027@163.com
  18. }
  19. notification\_email\_from keepalived@chtopnet.com
  20. smtp\_server 127.0.0.1
  21. smtp\_connect\_timeout 30
  22. router\_id LVS\_DEVEL
  23. }
  24. vrrp\_instance VI\_1 {
  25. state MASTER
  26. interface eth0
  27. virtual\_router\_id 51
  28. mcast\_src\_ip 192.168.0.154    **<**==主nginx的IP地址
  29. priority 100
  30. advert\_int 1
  31. authentication {
  32. auth\_type PASS
  33. auth\_pass chtopnet
  34. }
  35. virtual\_ipaddress {
  36. 192.168.0.188                      **<**==vip地址
  37. }
  38. }
  39. #service keepalived start

我们来看一下日志：

* 1. [root@ltos ~]# tail /var/log/messages
  2. Oct 6 03:25:03 ltos avahi-daemon[2306]: Registering new address record for 192.168.0.188 on eth0.
  3. Oct 6 03:25:03 ltos avahi-daemon[2306]: Registering new address record for 192.168.0.154 on eth0.
  4. Oct 6 03:25:03 ltos avahi-daemon[2306]: Registering HINFO record with values 'I686'/'LINUX'.
  5. Oct 6 03:25:23 ltos avahi-daemon[2306]: Withdrawing address record for fe80::20c:29ff:feb9:eeab on eth0.
  6. Oct 6 03:25:23 ltos avahi-daemon[2306]: Withdrawing address record for 192.168.0.154 on eth0.
  7. Oct 6 03:25:23 ltos avahi-daemon[2306]: Host name conflict, retrying with **<ltos-31>**

很显然vrrp已经启动，我们还可以通过命令来检查

* 1. [root@ltos html]# ip addr
  2. 1: lo: **<LOOPBACK**,UP,LOWER\_UP**>** mtu 16436 qdisc noqueue
  3. link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  4. inet 127.0.0.1/8 scope host lo
  5. inet6 ::1/128 scope host
  6. valid\_lft forever preferred\_lft forever
  7. 2: eth0: **<BROADCAST**,MULTICAST,UP,LOWER\_UP**>** mtu 1500 qdisc pfifo\_fast qlen 1000
  8. link/ether 00:0c:29:ba:9b:e7 brd ff:ff:ff:ff:ff:ff
  9. inet 192.168.0.154/24 brd 192.168.0.255 scope global eth0
  10. inet 192.168.0.188/32 scope global eth0
  11. inet6 fe80::20c:29ff:feba:9be7/64 scope link
  12. valid\_lft forever preferred\_lft forever
  13. 3: sit0: **<NOARP>** mtu 1480 qdisc noop
  14. link/sit 0.0.0.0 brd 0.0.0.0

说明vip已经启动，这样主服务器就配置好了，辅机的配置大致一样，除了配置文件有少部分的变化，下面贴出辅机的配置文件：

* 1. ! Configuration File for keepalived
  2. global\_defs {
  3. notification\_email {
  4. yuhongchun027@163.com
  5. }
  6. notification\_email\_from keepalived@chtopnet.com
  7. smtp\_server 127.0.0.1
  8. smtp\_connect\_timeout 30
  9. router\_id LVS\_DEVEL
  10. }
  11. vrrp\_instance VI\_1 {
  12. state BACKUP
  13. interface eth0
  14. virtual\_router\_id 51
  15. mcast\_src\_ip 192.168.0.155              **<**==辅nginx的IP的地址
  16. priority 100
  17. advert\_int 1
  18. authentication {
  19. auth\_type PASS
  20. auth\_pass chtopnet
  21. }
  22. virtual\_ipaddress {
  23. 192.168.0.188
  24. }
  25. }

[实现真正意义上的负载均衡高可用](http://network.51cto.com/art/201007/209823_3.htm)

针对Nginx+Keepalived，编写nginx监控脚本nginx\_pid.sh，此脚本思路其实也很简单，即放置在后台一直监控nginx进程；如进程消失，尝试重启nginx，如是失败则立即停掉本机的keepalived服务，让另一台负载均衡器接手，此脚本直接从生产环境下载：

* 1. vim /root/nginx\_pid.sh
  2. #!/bin/bash
  3. while  :
  4. do
  5. nginxpid=`ps -C nginx --no-header | wc -l`
  6. if [ $nginxpid -eq 0 ];then
  7. /usr/local/nginx/sbin/nginx
  8. sleep 5
  9. if [ $nginxpid -eq 0 ];then
  10. /etc/init.d/keepalived stop
  11. fi
  12. fi
  13. sleep 5
  14. done

然后置于后台运行 sh /root/nginx\_pid.sh &，这种写法是错误的，这样你用root用户logout后，此进程会消失；正确写法为nohup/bin/bash /root/nginx\_pid.sh &,附带下注释:如果你正在运行一个进程，而且你觉得在退出帐户时该进程还不会结束，那么可以使用nohup命令。该命令可以在你退出root帐户之后继续运行相应的进程。nohup就是不挂起的意思( no hang up)，哈哈，差点老马失蹄了。

**后记：**

我的线上环境网络非常复杂，这也是LVS+Keepalived失败的原因。目前此套架构在1000并发的电子商务网站非常稳定，带来的直接影响就是nginx\_backup一直处于闲置状态。相对于张宴的双机轮询而言，我感觉他的可能更加完美，因为目前我的Nginx仅仅只做了负载均衡器，如果以后有机会我会尝试做负载均衡器/反向代理加速。