Nginx概要

解决的问题：

1. 什么是高并发和负载均衡？
2. 如何解决高并发和负载均衡
3. 什么是高可用？
4. 如何实现高可用
5. 负载均衡的session一致性问题

# 第一章：nginx使用场景，编译安装nginx，加入到启动服务

## 1.总结nginx使用场景和功能

场景：

1. 高并发
2. 优化并发，动静分离
3. 负载均衡, (负载均衡，英文名称为Load Balance，其意思就是分摊到多个操作单元上进行执行，例如Web[服务器](http://baike.baidu.com/view/899.htm)、[FTP服务器](http://baike.baidu.com/view/106754.htm)、[企业](http://baike.baidu.com/view/38340.htm)关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。)
4. 高可用场景
5. 上行下行数据量不均衡的问题（lvs）

Nginx的两个功能：

1. 作为前端服务器拥有响应静态页面功能
2. 作为集群构建者拥有反向代理功能

## 2.编译安装nginx，加入到启动服务

1. ./configure --prefix=/usr/tengine

2. make && make install

1. 将nginx里面**内容拷贝**（注意这里不要上传文件，会有编码问题）到/etc/init.d、目录下
2. 修改软件安装路径如图：



1. 修改nginx文件的执行权限

chmod +x nginx

1. 添加该文件到系统服务中去

chkconfig --add nginx

查看是否添加成功

chkconfig --list nginx

启动，停止，重新装载

service nginx start|stop|reload

htttpd

# 第二章:nginx的配置,虚拟主机配置规则等

## 1.Nginx配置文件及各个配置项含义

Nginx.conf配置文件参数说明

#定义Nginx运行的用户和用户组

user www www;

#nginx进程数，建议设置为等于CPU总核心数。

worker\_processes 8;

#全局错误日志定义类型，[ debug | info | notice | warn | error | crit ]

error\_log /var/log/nginx/error.log info;

#进程文件

pid /var/run/nginx.pid;

#一个nginx进程打开的最多文件描述符数目，理论值应该是最多打开文件数（系统的值ulimit -n）与nginx进程数相除，但是nginx分配请求并不均匀，所以建议与ulimit -n的值保持一致。

worker\_rlimit\_nofile 65535;

#工作模式与连接数上限

events

{

# 参考事件模型，use [ kqueue | rtsig | epoll | /dev/poll | select | poll ]; epoll模型是Linux 2.6以上版本内核中的高性能网络I/O模型，如果跑在FreeBSD上面，就用kqueue模型。

use epoll;

#单个进程最大连接数（最大连接数=连接数\*进程数）

worker\_connections 65535;

}

#设定http服务器

http

{

include mime.types; #文件扩展名与文件类型映射表

default\_type application/octet-stream; #默认文件类型

#charset utf-8; #默认编码

server\_names\_hash\_bucket\_size 128; #服务器名字的hash表大小

client\_header\_buffer\_size 32k; #上传文件大小限制

large\_client\_header\_buffers 4 64k; #设定请求缓

client\_max\_body\_size 8m; #设定请求缓

sendfile on; #开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，对于普通应用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理速度，降低系统的负载。注意：如果图片显示不正常把这个改 成off。

autoindex on; #开启目录列表访问，合适下载服务器，默认关闭。

tcp\_nopush on; #防止网络阻塞

tcp\_nodelay on; #防止网络阻塞

keepalive\_timeout 120; #长连接超时时间，单位是秒

#FastCGI相关参数是为了改善网站的性能：减少资源占用，提高访问速度。下面参数看字面意思都能理解。

fastcgi\_connect\_timeout 300;

fastcgi\_send\_timeout 300;

fastcgi\_read\_timeout 300;

fastcgi\_buffer\_size 64k;

fastcgi\_buffers 4 64k;

fastcgi\_busy\_buffers\_size 128k;

fastcgi\_temp\_file\_write\_size 128k;

#gzip模块设置

gzip on; #开启gzip压缩输出

gzip\_min\_length 1k; #最小压缩文件大小

gzip\_buffers 4 16k; #压缩缓冲区

gzip\_http\_version 1.0; #压缩版本（默认1.1，前端如果是squid2.5请使用1.0）

gzip\_comp\_level 2; #压缩等级

gzip\_types text/plain application/x-javascript text/css application/xml;

#压缩类型，默认就已经包含text/html，所以下面就不用再写了，写上去也不会有问题，但是会有一个warn。

gzip\_vary on;

#limit\_zone crawler $binary\_remote\_addr 10m; #开启限制IP连接数的时候需要使用

upstream blog.ha97.com {

#upstream的负载均衡，weight是权重，可以根据机器配置定义权重。weigth参数表示权值，权值越高被分配到的几率越大。

server 192.168.80.121:80 weight=3;

server 192.168.80.122:80 weight=2;

server 192.168.80.123:80 weight=3;

}

#虚拟主机的配置

server

{

#监听端口

listen 80;

#域名可以有多个，用空格隔开

server\_name [www.ha97.com](http://www.ha97.com/) ha97.com;

index index.html index.htm index.php;

root /data/www/ha97;

location ~ .\*.(php|php5)?$

{

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

#图片缓存时间设置

location ~ .\*.(gif|jpg|jpeg|png|bmp|swf)$

{

[exp](http://www.lengmo.net/)ires 10d;

}

#JS和CSS缓存时间设置

location ~ .\*.(js|css)?$

{

expires 1h;

}

#日志格式设定

log\_format access ‘$remote\_addr – $remote\_user [$time\_local] “$request” ‘

‘$status $body\_bytes\_sent “$http\_referer” ‘

‘”$http\_user\_agent” $http\_x\_forwarded\_for’;

#定义本虚拟主机的访问日志

access\_log /var/log/nginx/ha97access.log access;

#对 “/” 启用反向代理

location / {

proxy\_pass http://127.0.0.1:88;

proxy\_redirect off;

proxy\_set\_header X-Real-IP $remote\_addr;

#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

#以下是一些反向代理的配置，可选。

proxy\_set\_header Host $host;

client\_max\_body\_size 10m; #允许客户端请求的最大单文件字节数

client\_body\_buffer\_size 128k; #缓冲区代理缓冲用户端请求的最大字节数，

proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)

proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时)

proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)

proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小

proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的设置

proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）

proxy\_temp\_file\_write\_size 64k;

#设定缓存文件夹大小，大于这个值，将从upstream服务器传

}

#设定查看Nginx状态的地址

location /NginxStatus {

stub\_status on;

access\_log on;

auth\_basic “NginxStatus”;

auth\_basic\_user\_file conf/htpasswd;

#htpasswd文件的内容可以用apache提供的htpasswd[工具](http://www.lengmo.net/)来产生。

}

#本地动静分离反向代理配置

#所有jsp的页面均交由tomcat或resin处理

location ~ .(jsp|jspx|do)?$ {

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass http://127.0.0.1:8080;

}

#所有静态文件由nginx直接读取不经过tomcat或resin

location ~ .\*.(htm|html|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma)$

{ expires 15d; }

location ~ .\*.(js|css)?$

{ expires 1h; }

}

}

## 2.演示nginx作为前端服务器虚拟主机功能

演示1：通过不同servername的server模拟不同项目在同一服务器

演示2：同一server不同location匹配到不同路径不同页面

## 3.location配置规则

**location 优先级官方文档**

1. Directives with the = prefix that match the query exactly. If found, searching stops.
2. All remaining directives with conventional strings, longest match first. If this match used the ^~ prefix, searching stops.
3. Regular expressions, in order of definition in the configuration file.
4. If #3 yielded a match, that result is used. Else the match from #2 is used.
5. =前缀的指令严格匹配这个查询。如果找到，停止搜索。
6. 所有剩下的常规字符串，最长的匹配。如果这个匹配使用^〜前缀，搜索停止。
7. 正则表达式，在配置文件中定义的顺序。
8. 如果第3条规则产生匹配的话，结果被使用。否则，如同从第2条规则被使用。

Nginx location的匹配规则：

location 的执行逻辑跟 location 的编辑顺序无关。

矫正：这句话不全对，“普通 location ”的匹配规则是“最大前缀”，因此“普通 location ”的确与 location 编辑顺序无关；但是“正则 location ”的匹配规则是“顺序匹配，且只要匹配到第一个就停止后面的匹配”；“普通location ”与“正则 location ”之间的匹配顺序是？先匹配普通 location ，再“考虑”匹配正则 location 。注意这里的“考虑”是“可能”的意思，也就是说匹配完“普通 location ”后，有的时候需要继续匹配“正则 location ”，有的时候则不需要继续匹配“正则 location ”。两种情况下，不需要继续匹配正则 location ：（ 1 ）当普通 location 前面指定了“ ^~ ”，特别告诉 Nginx 本条普通 location 一旦匹配上，则不需要继续正则匹配；（ 2 ）当普通location 恰好严格匹配上，不是最大前缀匹配，则不再继续匹配正则。

## 4.location的一般应用

提下今天的作业，配置网站静态化，为什么要静态化？搜索优化需要静态化

动静分离的好处？缓存的利用

动静分离：

      location ^~ /(images|scripts|styles|upload)/  {

            root    /www/abc.com/www/htdocs;

            expires 30d;

       }

           location ~\*\.(gif|jpg|jpeg|png|css|ico|html)$ {

            root    /www/abc.com/www/htdocs;

            expires 30d;

       }

# 第三章:nginx访问禁止允许,用户认证,反向代理理论

## 1.nginx访问禁止允许规则

按照顺序依次检测，直到匹配到第一条规则

样例：

location / {

deny 192.168.1.1;

allow 192.168.1.0/24;

allow 10.1.1.0/16;

allow 2001:0db8::/32;

deny all;

}

测试屏蔽外面的主机显示，注意这里需要用vmnet8的ip

禁止使用IP访问网站

server {

    listen       80;

    server\_name  "";

    return       444;

}

nginx特有的，非http标准的返回码444，它可以用来关闭连接

## 2.nginx用户认证

需要安装httpd

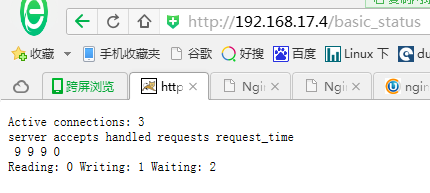
使用命令htpasswd才创建文件和密码

## 3.nginx访问状态监控

location /basic\_status {

stub\_status on;

}



## 4.代理和反向代理

什么是代理？

代理是为网络用户代理了来访问网络的，比如Google agent代理翻墙

什么是反向代理？

以[代理服务器](http://baike.baidu.com/view/751.htm)来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器

# 第四章: 反向代理和tengine的状态监控

## 1.反向代理

首先在两台服务器分别配置并启动tomcat，修改tomcat首页为tomcat1和tomcat2

配置反向代理和upstream

upstream **backend** {

server backend1.example.com weight=5;

server backend2.example.com:8080;

server unix:/tmp/backend3;

server backup1.example.com:8080 backup;

server backup2.example.com:8080 backup;

}

server {

location / {

proxy\_pass http://**backend**;

}

}

从上面可以看出定义一组服务器。 这些服务器可以监听不同的端口。 而且，监听在TCP和UNIX域套接字的服务器可以混用

访问分别显示tomcat1和tomcat2

## 2.tengine的状态监控

Tengine的状态监控有两种



这里演示一个健康检查模块功能

配置一个status的location

location /status {

check\_status;

}

在upstream配置如下

check interval=3000 rise=2 fall=5 timeout=1000 type=http;

check\_http\_send "HEAD / HTTP/1.0\r\n\r\n";

check\_http\_expect\_alive http\_2xx http\_3xx;

这里可以故意出个问题，修改tomcat里面的index.jsp然后head里面有个引用css文件，会导致遵循轮询的机制出问题，可以通过nginx的acess的日志查看

# 第五章：抛出session一致性问题，给出memcached缓存方案，安装memcached

## 1.抛出Session一致性问题

## 2.安装memcached缓存数据库

将.repo文件拷贝到/etc/yum.repo.d目录下yum makecache构建yum库

yum –y install memcached

通过telnet测试memcached数据库

## 3.tomcat配置名称和session一致性配置

将jar包拷贝到tomcat的lib下

在tomcat的server.xml中配置jvmRoute

<Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat2">

在context.xml的Context配置Manager

<Manager className="de.javakaffee.web.msm.MemcachedBackupSessionManager"

memcachedNodes="n1:192.168.17.4:11211"

sticky="false"

lockingMode="auto"

sessionBackupAsync="false"

requestUriIgnorePattern=".\*\.(ico|png|gif|jpg|css|js)$"

sessionBackupTimeout="1000" transcoderFactoryClass="de.javakaffee.web.msm.serializer.kryo.KryoTranscoderFactory"

/>

## 4.index.jsp配置

<%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>

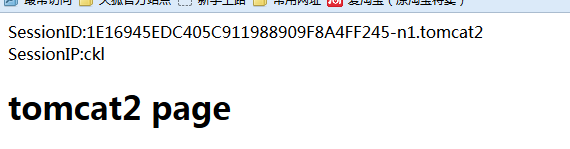
SessionID:<%=session.getId()%>

<br/>

SessionIP:<%=request.getServerName()%>

<br/>

<h1>tomcat2 page</h1>



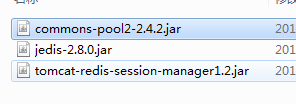
## 5.利用redis实现session共享

**安装redis缓存数据库，修改配置文件vi /etc/redis.conf**

**将bind的127.0.0.1修改为本机地址，否则只能本机访问了**



**导入redislib中三个jar包到tomcat中，**



**在tomcat添加配置**

**<Valve className="com.orangefunction.tomcat.redissessions.RedisSessionHandlerValve" />**

**<Manager className="com.orangefunction.tomcat.redissessions.RedisSessionManager"**

**host="192.168.17.4"**

**port="6379"**

**database="0"**

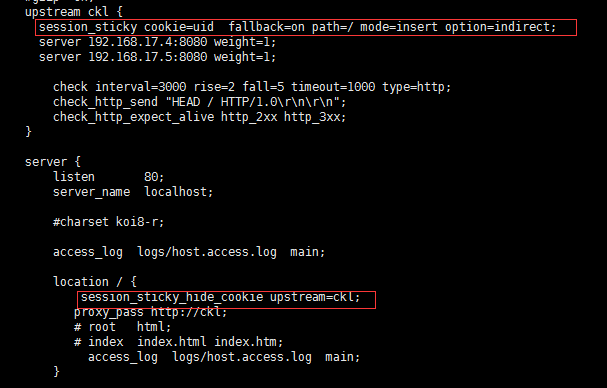
**maxInactiveInterval="60" />**

**和memcached类似**

# 第六章：利用memcached配置session一致性，session一致性的另外一种方案tengine的会话保持功能

## 1.tengine的session一致性会话保持

如图配置，相当于粘滞模式



# 第七章：nginx+keepalived高可用

## 1.基本服务器宕机的主从切换配置

两台nginx服务器，分别安装keepalived，配置**/etc/keepalived/keepalived.conf**

**主从配置：**

主nginx

修改主nginx下/etc/keepalived/keepalived.conf文件

! Configuration File for keepalived

#全局配置

global\_defs {

notification\_email { #指定keepalived在发生切换时需要发送email到的对象，一行一个

XXX@XXX.com

}

notification\_email\_from XXX@XXX.com #指定发件人

#smtp\_server XXX.smtp.com #指定smtp服务器地址

#smtp\_connect\_timeout 30 #指定smtp连接超时时间

router\_id LVS\_DEVEL #运行keepalived机器的一个标识

}

vrrp\_instance VI\_1 {

state MASTER #标示状态为MASTER 备份机为BACKUP

interface eth0 #设置实例绑定的网卡

virtual\_router\_id 51 #同一实例下virtual\_router\_id必须相同

priority 100 #MASTER权重要高于BACKUP 比如BACKUP为99

advert\_int 1 #MASTER与BACKUP负载均衡器之间同步检查的时间间隔，单位是秒

authentication { #设置认证

auth\_type PASS #主从服务器验证方式

auth\_pass 8888

}

virtual\_ipaddress { #设置vip

192.168.17.10 #可以多个虚拟IP，换行即可

}

}

备nginx

修改备nginx下/etc/keepalived/keepalived.conf文件

**配置备nginx时需要注意：需要修改state为BACKUP , priority比MASTER低，virtual\_router\_id和master的值一致**

! Configuration File for keepalived

#全局配置

global\_defs {

notification\_email { #指定keepalived在发生切换时需要发送email到的对象，一行一个

XXX@XXX.com

}

notification\_email\_from XXX@XXX.com #指定发件人

#smtp\_server XXX.smtp.com #指定smtp服务器地址

#smtp\_connect\_timeout 30 #指定smtp连接超时时间

router\_id LVS\_DEVEL #运行keepalived机器的一个标识

}

vrrp\_instance VI\_1 {

state BACKUP #标示状态为MASTER 备份机为BACKUP

interface eth0 #设置实例绑定的网卡

virtual\_router\_id 51 #同一实例下virtual\_router\_id必须相同

priority 99 #MASTER权重要高于BACKUP 比如BACKUP为99

advert\_int 1 #MASTER与BACKUP负载均衡器之间同步检查的时间间隔，单位是秒

authentication { #设置认证

auth\_type PASS #主从服务器验证方式

auth\_pass 8888

}

virtual\_ipaddress { #设置vip

192.168.17.10 #可以多个虚拟IP，换行即可

}

}

此时如果主机宕机则备机会顶替主机获得虚拟ip

## 2.基于nginx进程死掉的主从切换配置

keepalived是通过检测keepalived进程是否存在判断服务器是否宕机，如果keepalived进程在但是nginx进程不在了那么keepalived是不会做主备切换，所以我们需要写个脚本来监控nginx进程是否存在，如果nginx不存在就将keepalived进程杀掉。

在主nginx上需要编写nginx进程检测脚本（check\_nginx.sh），判断nginx进程是否存在，如果nginx不存在就将keepalived进程杀掉，check\_nginx.sh内容如下：

#!/bin/bash

# 如果进程中没有nginx则将keepalived进程kill掉

A=`ps -C nginx --no-header |wc -l` ## 查看是否有 nginx进程 把值赋给变量A

if [ $A -eq 0 ];then ## 如果没有进程值得为 零

service keepalived stop ## 则结束 keepalived 进程

fi

将check\_nginx.sh拷贝至/etc/keepalived下，

注意修改/etc/keepalived/check\_nginx.sh的可执行权限

修改主nginx的keepalived.conf，添加脚本定义检测：

注意下边红色标识地方：

#全局配置

global\_defs {

notification\_email { #指定keepalived在发生切换时需要发送email到的对象，一行一个

XXX@XXX.com

}

notification\_email\_from miaoruntu@itcast.cn #指定发件人

#smtp\_server XXX.smtp.com #指定smtp服务器地址

#smtp\_connect\_timeout 30 #指定smtp连接超时时间

router\_id LVS\_DEVEL #运行keepalived机器的一个标识

}

vrrp\_script check\_nginx {

script "/etc/keepalived/check\_nginx.sh" ##监控脚本

interval 2 ##时间间隔，2秒

weight 2 ##权重

}

vrrp\_instance VI\_1 {

state MASTER #标示状态为MASTER 备份机为BACKUP

interface eth0 #设置实例绑定的网卡

virtual\_router\_id 51 #同一实例下virtual\_router\_id必须相同

priority 100 #MASTER权重要高于BACKUP 比如BACKUP为99

advert\_int 1 #MASTER与BACKUP负载均衡器之间同步检查的时间间隔，单位是秒

authentication { #设置认证

auth\_type PASS #主从服务器验证方式

auth\_pass 8888

}

track\_script {

check\_nginx #监控脚本

}

virtual\_ipaddress { #设置vip

192.168.101.100 #可以多个虚拟IP，换行即可

}

}

修改后重启keepalived

# 第八章：nginx内置缓存

## 1.添加安装清除缓存模块

使用参数重新配置

./configure --prefix=/app/nginx -user=nobody -group=nobody --with-http\_stub\_status\_module \

--with-http\_ssl\_module --with-http\_realip\_module \

--add-module=../nginx\_upstream\_hash-0.3.1/ \

--add-module=/opt/sxt/ngx\_cache\_purge-2.3

前面怎么安装的这里参数全部相同，路径什么的也需要相同

Make

不执行make install

然后替换nginx二进制文件:

cp /opt/sxt/soft/tengine/sbin/nginx /app/nginx/sbin/nginx.bak

cp ./objs/nginx /opt/sxt/soft/tengine/sbin/

## 2.配置缓存

注意缓存的基本上都是静态的东西，动态的插了java代码之类的是缓存不了的

在http的范围里面需要配置的就两个东西：

proxy\_temp\_path和proxy\_cache\_path以下是配置意义

当然levels和keys\_zone这种东西也是需要配置的

#proxy\_temp\_path代理临时目录

proxy\_temp\_path /var/temp/nginx/proxy;

#proxy\_cache\_path代理缓存目录，和proxy\_temp\_path必须在同一个分区

#/var/temp/nginx/proxy\_cache\_dir缓存目录

#levels指定该缓存空间有两层hash目录，第一层目录名是1个字母或数字长度，第二层目录名为2个字母或数字长度

#keys\_zone=cache\_one:50m缓存区名称为cache\_one，在内存中的空间是50M，inactive=1d表示1天清空一次缓存 ，max\_size指定磁盘空间大小为500M

proxy\_cache\_path /var/temp/nginx/proxy\_cache\_dir levels=1:2 keys\_zone=cache\_one:50m inactive=1d max\_size=500m;

然后是配置缓存的东西和缓存的条件

这里最重要的是proxy\_cache cache\_one;（注：这里的proxy\_cache一定是前面的keys\_zone）和proxy\_cache\_key $host$uri$is\_args$args，以及后面缓存没有的时候的proxy\_pass

location ~ /item(/.\*){

#指定缓存区域名称

proxy\_cache cache\_one;

#以域名、URI、参数组合成Web缓存的Key值，Nginx根据Key值哈希

proxy\_cache\_key $host$uri$is\_args$args;

#请求头中添加请求的主机名

proxy\_set\_header Host $host;

#请求头中添加真实的客户端ip，通过X-Forwarded-For可获取

proxy\_set\_header X-Forwarded-For $remote\_addr;

#代理访问后端tomcat

#请求item从缓存中找，如果缓存中没有则向tomcat请求

#$1表示取出正则表达式(/.\*)所匹配的内容，使用$1的效果例如请求http://ccc.test.com/item/100010.html则请求tomcat服务器 http://ip:port/100010.html

#如果不使用$1则会将/item/...加在tomcat服务地址之后访问，即http://ip:port/item/100010.html

proxy\_pass http://tomcat\_server\_pool$1;

}

然后就是在匹配之前配置一个purge，最重要的匹配的上和proxy\_cache\_purge cache\_one $host$1$is\_args$args

location ~ /purge(/.\*)

{

#安全设置，指定请求客户端的IP或IP段才可以清除URL缓存，这里为了方便测试设置为all

#allow 127.0.0.1;

allow all;

#指定清空缓存的区域名称cache\_one(要和上边proxy\_cache\_path缓存配置中指定的缓存区域名称一致)

#指定缓存的key规则$host$1$is\_args$args，要和下边设置缓存的key一致$host$uri$is\_args$args

#注意$host$1$is\_args$args中的$1表示当前请求的uri，$host$1$is\_args$args=$host$uri$is\_args$args

proxy\_cache\_purge cache\_one $host$1$is\_args$args;

}

## 3.测试

可以配置缓存后看看缓存文件，也可以通过查看相应的tomcat没有请求到达，注意这里是tomcat没有请求来，但是nginx还是有请求的