# 1、缓存

# <http://www.cnblogs.com/caoguo/p/5012447.html> 不错的教程

利用请求的局部性原理，将请求过的内容在本地建立一个副本，下次访问时不再连接到后端服务器，直接响应本地内容

Nginx服务器启动后，会对本地磁盘上的缓存文件进行扫描，在内存中建立缓存索引，并有专门的进程对缓存文件进行过期判断、更新等进行管理

对于缓存，我们大概会有以下问题：  
  
（1）缓存文件放在哪儿？  
（2）缓存的空间大小是否可以限定?  
（3）如何指定哪些请求被缓存？  
（4）缓存的有效期是多久？  
（5）对于某些请求，是否可以不走缓存？  
  
解决这些问题后，nginx的缓存也就基本配置完成了，下面看详细配置过程

## 1、首要问题，缓存配置路径问题

### 解释：要使用缓存，首先要使用 proxy\_cache\_path 这个指令（必须放在 http 上下文的顶层位置），然后在目标上下文中使用 proxy\_cache 指令，

#### 1、有两个必填参数，

#### 第一个参数为 缓存目录，

#### 第二个参数keys\_zone指定缓存名称和占用内存空间的大小（注：示例中的10m是对内存中缓存内容元数据信息大小的限制，如果想限制缓存总量大小，需要用 max\_size 参数）

在nginx.conf里的http块加入以下代码

#proxy\_temp\_path和proxy\_cache\_path指定的路径必须在同一分区

proxy\_temp\_path   /tmp/proxy\_temp\_dir;

#设置名称为nginx\_cache，

levels=1:2 表示 目录的级别

keys\_zone=nginx\_cache:500m内存缓存空间大小为500MB，不要设置太大

inactive=1d 缓存的时间为1天，1天以后自动清除

max\_size=30g;硬盘缓存空间大小为30GB。

proxy\_cache\_path  /tmp/proxy\_cache\_dir  levels=1:2   keys\_zone=nginx\_cache:500m inactive=1d max\_size=30g;

|  |
| --- |
| #允许客户端请求的最大单文件字节数  client\_max\_body\_size 300m;  #缓冲区代理缓冲用户端请求的最大字节数，可以理解为先保存到本地再传给用户  client\_body\_buffer\_size 600;  #跟后端服务器连接的超时时间—发起握手等候相应超时时间(代理连接超时)  proxy\_connect\_timeout 600;  #连接成功后，等待后端服务器响应时间，其实已经进入后端排队之中等候处理(代理发送超时)  proxy\_send\_timeout 600;  #后端服务器回传时间，就是在规定时间内容，后端服务器必须传完所有数据(代理接收超时)  proxy\_read\_timeout 600; #  #设置代理服务器（nginx）保存用户头信息的缓冲区大小，这个缓存区间会保存用户的头信息用来提供给Nginx  #进行规则处理，一般只要保存下 头信息即可  proxy\_buffer\_size 16k;  ##proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置，同上，告诉Nginx保存单个用的几个Buffer及最大用多大空间  proxy\_buffers 4 32k;  #高负荷下缓冲大小（proxy\_buffers\*2） 如果系统很忙的时候可以申请更大的proxy\_buffers 官方推荐\*2  proxy\_busy\_buffers\_size 64;  #设定缓存文件夹大小，大于这个值，将从upstream服务器传  proxy\_temp\_file\_write\_size 64k; |

|  |
| --- |
| #使用web缓存区cache\_one   proxy\_cache cache\_one;   #设置主机头和客户端真实地址，以便服务器获取客户端真实IP  #后端的Web服务器可以通过X-Forwarded-For获取用户真实IP  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  #记录远程的ip  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; |

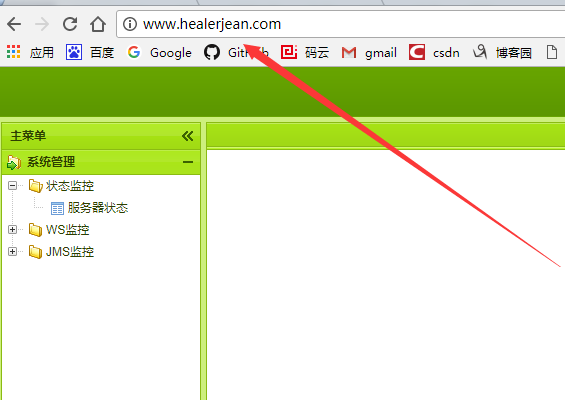
# 1、反向代理

## 1、前言，这里我准备了两个服务器，（加载的是两套代码），分别是刘传宪10.3.181.60 和我的10.3.181.48

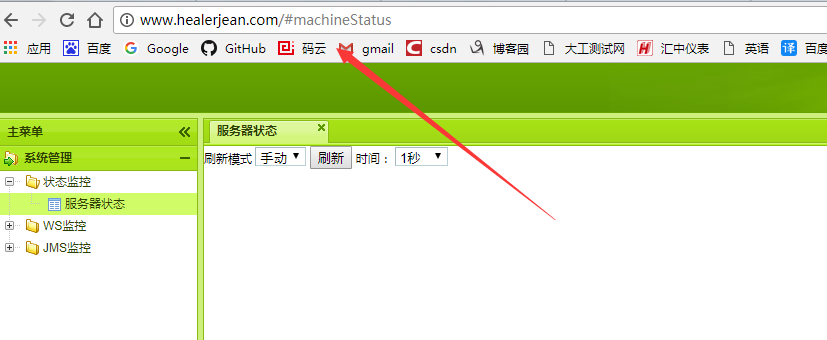
## 2、代理服务器到60服务器，代理路径直接 包含端口， 即可实现浏览器访问只有域名没有端口号

|  |
| --- |
| location / {  #设置主机头和客户端真实地址，以便服务器获取客户端真实IP  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  #禁用缓存  proxy\_buffering off;  #设置反向代理的地址  proxy\_pass http://10.3.181.60:8080/dsp\_oa/;  } |

## 3、浏览器直接访问



## 4、访问路径也可以实现域名的访问



# 2、负载均衡

Nginx负载均衡一些基础知识：

**nginx 的 upstream目前支持 4 种方式的分配**

1)、轮询（默认）

　　每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。

2)、weight

　　指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。

2)、ip\_hash

　　每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。

3)、fair（第三方）

　　按后端服务器的响应时间来分配请求，响应时间短的优先分配。

4)、url\_hash（第三方）

## 1、编写 upstream ，名字backend 任意，但是下面的代理路径中要用到，下面的的ip为提供服务的ip的地址，也就是后端的多个服务器。这个代码写到的http中，再server外面放置

|  |
| --- |
| upstream backend {  #ip\_hash;  server 192.168.1.251;  server 192.168.1.252;  server 192.168.1.247;  } |

1. #tomcat的三个服务
2. upstream mysite {
3. server localhost:18080 weight=5;
4. server localhost:28080 weight=5;
5. server localhost:38080 weight=5;
6. }

## 2、添加跳代理路径

|  |
| --- |
| location / {  #设置主机头和客户端真实地址，以便服务器获取客户端真实IP  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  #禁用缓存  proxy\_buffering off;  #设置反向代理的地址  #proxy\_pass http://10.3.181.60:8080/dsp\_oa/;  proxy\_pass http://backend;  } |

## 3、upstream详解

## 1、轮询(weight=1)

### 解释：默认选项，当weight不指定时，各服务器weight相同，  每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。

upstream bakend {   
server 192.168.1.10;   
server 192.168.1.11;   
}

## 2、weight

### 解释：指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。  如果后端服务器down掉，能自动剔除。  比如以下配置，则1.11服务器的访问量为1.10服务器的两倍。

upstream bakend {   
server 192.168.1.10 weight=1;   
server 192.168.1.11 weight=2;   
}

## 3、ip\_hash

### 解释：每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session不能跨服务器的问题。  如果后端服务器down掉，要手工down掉。（www.jbxue.com 整理）

upstream resinserver{   
ip\_hash;   
server 192.168.1.10:8080;   
server 192.168.1.11:8080;   
}

## 4、fair（第三方插件）

解释：按后端服务器的响应时间来分配请求，响应时间短的优先分配。

upstream resinserver{   
server 192.168.1.10:8080;   
server 192.168.1.11:8080;   
fair;   
}

## 5、url\_hash（第三方插件）

### 解释：按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，后端服务器为缓存服务器时比较有效。  在upstream中加入hash语句，hash\_method是使用的hash算法。

upstream resinserver{   
server 192.168.1.10:8080;   
server 192.168.1.11:8080;   
hash $request\_uri;   
hash\_method crc32;   
}

## 设备的状态有:

1、down 表示单前的server暂时不参与负载   
2、weight 权重,默认为1。 weight越大，负载的权重就越大。   
3、max\_fails 允许请求失败的次数默认为1。当超过最大次数时，返回proxy\_next\_upstream 模块定义的错误   
4、fail\_timeout max\_fails次失败后，暂停的时间。   
5backup 备用服务器, 其它所有的非backup机器down或者忙的时候，请求backup机器。所以这台机器压力会最轻。-



# 3、backup备份服务器 不停往更新，（redis保存session）

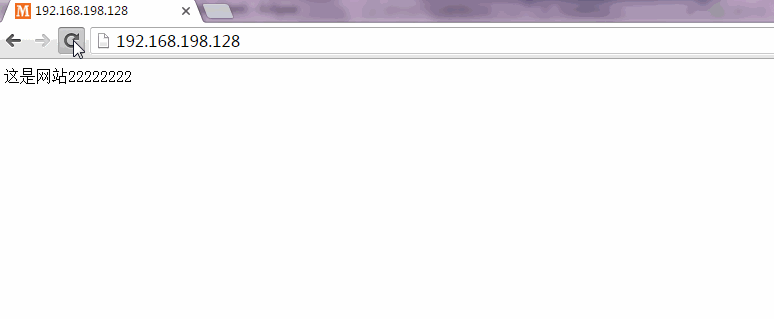
具体配置文字版：

* upstream backend {
* server 192.168.198.128:8080 weight=1;
* server 192.168.198.128:8090 weight=4;
* server 192.168.198.128:8091 backup;
* }
* server {
* listen 80;
* server\_name localhost;
* location / {
* root html;
* index index.html index.htm;
* proxy\_pass http:*//backend;*
* }

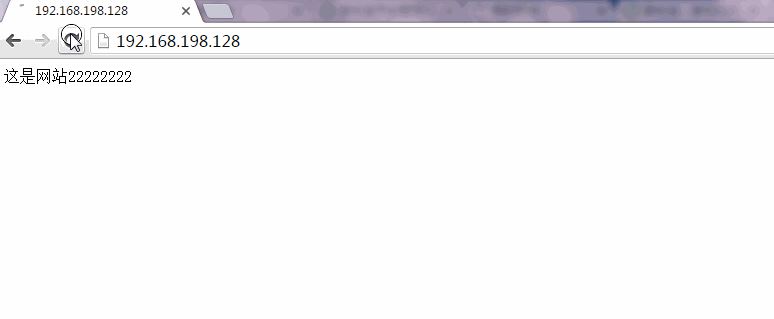
### 我这里先做一个必要的小说明：

* 192.168.198.128:8080 服务器下面会称为 服务器1 ，每五次请求会到这里一次;
* 192.168.198.128:8090 服务器下面会称为 服务器2 ，每五次请求会到这里四次;
* 192.168.198.128:8091 服务器下面会称为 服务器3 ，即备份服务器。

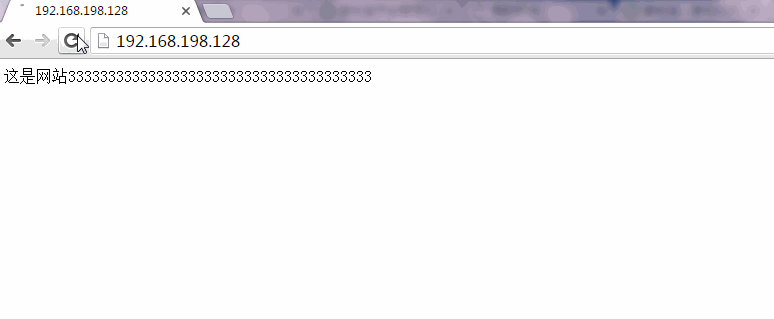
### 1、这个时候服务器1、服务器2和服务器3都正常启动，我刷新页面，会按照权重规则请求服务器1和服务器2，并不会请求服务器3（备份服务器）。



### 2、接下来我把服务器1停掉，服务器2和服务器3都正常启动，我刷新页面，这个时候所有请求都是服务器2，也并不会请求服务器3（备份服务器）。



### 3、接下来我把服务器1、服务器2都停掉，服务器3正常启动，我刷新页面，所有请求都转发到服务器3（备份服务器），看图片：



### 4、我再次依次启动服务器1和服务器2，所有请求都又按照权重规则分配了。**在你所有正常服务器都挂掉时，系统依然高可用，这就是备份服务器的用处**！