

Java学习中.....

博客园

首页

联系

管理

随笔-140 文章-3 评论-1816

Nginx详解-服务器集群

Nginx是什么

代理服务器：一般是指局域网内部的机器通过代理服务器发送请求到互联网上的服务器,代理服务器一般作用在客户端。应用比如：GoAgent，FQ神器。



一个完整的代理请求过程为：客户端首先与代理服务器创建连接，接着根据代理服务器所使用的代理协议，请求对目标服务器创建连接、或者获得目标服务器的指定资源。Web代理（proxy）服务器是网络的中间实体。代理位于Web客户端和Web服务器之间，扮演“中间人”的角色。HTTP的代理服务器即是Web服务器又是Web客户端。

代理服务器是介于客户端和Web服务器之间的另一台服务器，有了它之后，浏览器不是直接到Web服务器去取回网页而是向代理服务器发出请求，信号会先送到代理服务器，由代理服务器来取回浏览器所需要的信息并传送给你的浏览器。

正向代理：是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。

反向代理服务器：在服务器端接受客户端的请求，然后把请求分发给具体的服务器进行处理，然后再将服务器的响应结果反馈给客户端。Nginx就是其中的一种反向代理服务器软件。

Nginx：Nginx ("engine x")，Nginx ("engine x") 是俄罗斯人Igor Sysoev(塞索耶夫)编写的一款高性能的 HTTP 和反向代理服务器。也是一个IMAP/POP3/SMTP代理服务器；也就是说，Nginx本身就可以托管网站，进行HTTP服务处理，也可以作为反向代理服务器使用。

说明：客户端必须设置正向代理服务器，当然前提是要知道正向代理服务器的IP地址，还有代理程序的端口。

反向代理正好与正向代理相反，对于客户端而言代理服务器就像是原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理的命名空间(name-space)中的内容发送普通请求，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回给客户端。



用户A始终认为它访问的是原始服务器B而不是代理服务器Z，但实际上反向代理服务答，从原始资源服务器B中取得用户A的需求资源，然后发送给用户A。由于防火墙的作用，只允许代理

35

推荐

0

反对

[关注](#) | [顶部](#) | [评论](#)

服务器Z访问原始资源服务器B。尽管在这个虚拟的环境下，防火墙和反向代理的共同作用保护了原始资源服务器B，但用户A并不知情。

Nginx的应用现状

Nginx 已经在俄罗斯最大的门户网站—— Rambler Media (www.rambler.ru) 上运行了3年时间，同时俄罗斯超过20%的虚拟主机平台采用Nginx作为反向代理服务器。在国内，已经有 淘宝、新浪博客、新浪播客、网易新闻、六间房、56.com、Discuz!、水木社区、豆瓣、YUPOO、海内、迅雷在线 等多家网站使用 Nginx 作为Web服务器或反向代理服务器。

Nginx的特点

- 跨平台：Nginx 可以在大多数 Unix like OS编译运行，而且也有Windows的移植版本。
- 配置异常简单：非常容易上手。配置风格跟程序开发一样，神一般的配置
- 非阻塞、高并发连接：数据复制时，磁盘I/O的第一阶段是非阻塞的。官方测试能够支撑5万并发连接，在实际生产环境中跑到2~3万并发连接数.(这得益于Nginx使用了最新的epoll模型)
- 事件驱动：通信机制采用epoll模型，支持更大的并发连接。

Nginx的事件处理机制

对于一个基本的web服务器来说，事件通常有三种类型，网络事件、信号、定时器。

首先看一个请求的基本过程：建立连接---接收数据---发送数据 。

再次看系统底层的操作：上述过程（建立连接---接收数据---发送数据）在系统底层就是读写事件。

1) 如果采用阻塞调用的方式，当读写事件没有准备好时，必然不能够进行读写事件，那么久只好等待，等事件准备好了，才能进行读写事件。那么请求就会被耽搁。阻塞调用会进入内核等待，cpu就会让出去给别人用了，对单线程的worker来说，显然不合适，当网络事件越多时，大家都在等待呢，cpu空闲下来没人用，cpu利用率自然上不去，更别谈高并发了。

2) 既然没有准备好阻塞调用不行，那么采用非阻塞方式。非阻塞就是，事件，马上返回EAGAIN，告诉你，事件还没准备好呢，你慌什么，过会再来吧。好吧，你过一会，再来检查一下事件，直到事件准备好了为止，在这期间，你就可以先去做其它事情，然后再来看看事件好了没。虽然不阻塞了，但你得不时地过来检查一下事件的状态，你可以做更多的事情了，但带来的开销也是不小的

小结：非阻塞通过不断检查事件的状态来判断是否进行读写操作，这样带来的开销很大。

3) 因此才有了异步非阻塞的事件处理机制。具体到系统调用就是像select/poll/epoll/kqueue这样的系统调用。他们提供了一种机制，让你可以同时监控多个事件，调用他们是阻塞的，但可以设置超时时间，在超时时间之内，如果有事件准备好了，就返回。这种机制解决了我们上面两个问题。

以epoll为例：当事件没有准备好时，就放入epoll(队列)里面。如果有事件准备好了，那么就去处理；如果事件返回的是EAGAIN，那么继续将其放入epoll里面。从而，只要有事件准备好了，我们就去处理她，只有当所有时间都没有准备好时，才在epoll里面等着。这样，我们就可以并发处理大量的并发了，当然，这里的并发请求，是指未处理完的请求，线程只有一个，所以同时能处理的请求当然只有一个了，只是在请求间进行不断地切换而已，切换也是因为异步事件未准备好，而主动让出的。这里的切换是没有任何代价，你可以理解为循

环处理多个准备好的事件，事实上就是 这样的。

4) 与多线程的比较:

与多线程相比，这种事件处理方式是很优势的，不需要创建线程，每个请求占用的内存也很少，没有上下文切换，事件处理非常的轻量级。并发数再多也不会导致无谓的资源浪费（上下文切换）。

小结：通过异步非阻塞的事件处理机制，Nginx实现由进程循环处理多个准备好的事件，从而实现高并发和轻量级。

master/worker结构：一个master进程，生成一个或多个worker进程

内存消耗小：处理大并发的请求内存消耗非常小。在3万并发连接下，开启的10个Nginx 进程才消耗150M内存（15M*10=150M）
成本低廉：Nginx为开源软件，可以免费使用。而购买F5 BIG-IP、NetScaler等硬件负载均衡交换机则需要十多万至几十万人民币

内置的健康检查功能：如果 Nginx Proxy 后端的某台 Web 服务器宕机了，不会影响前端访问。

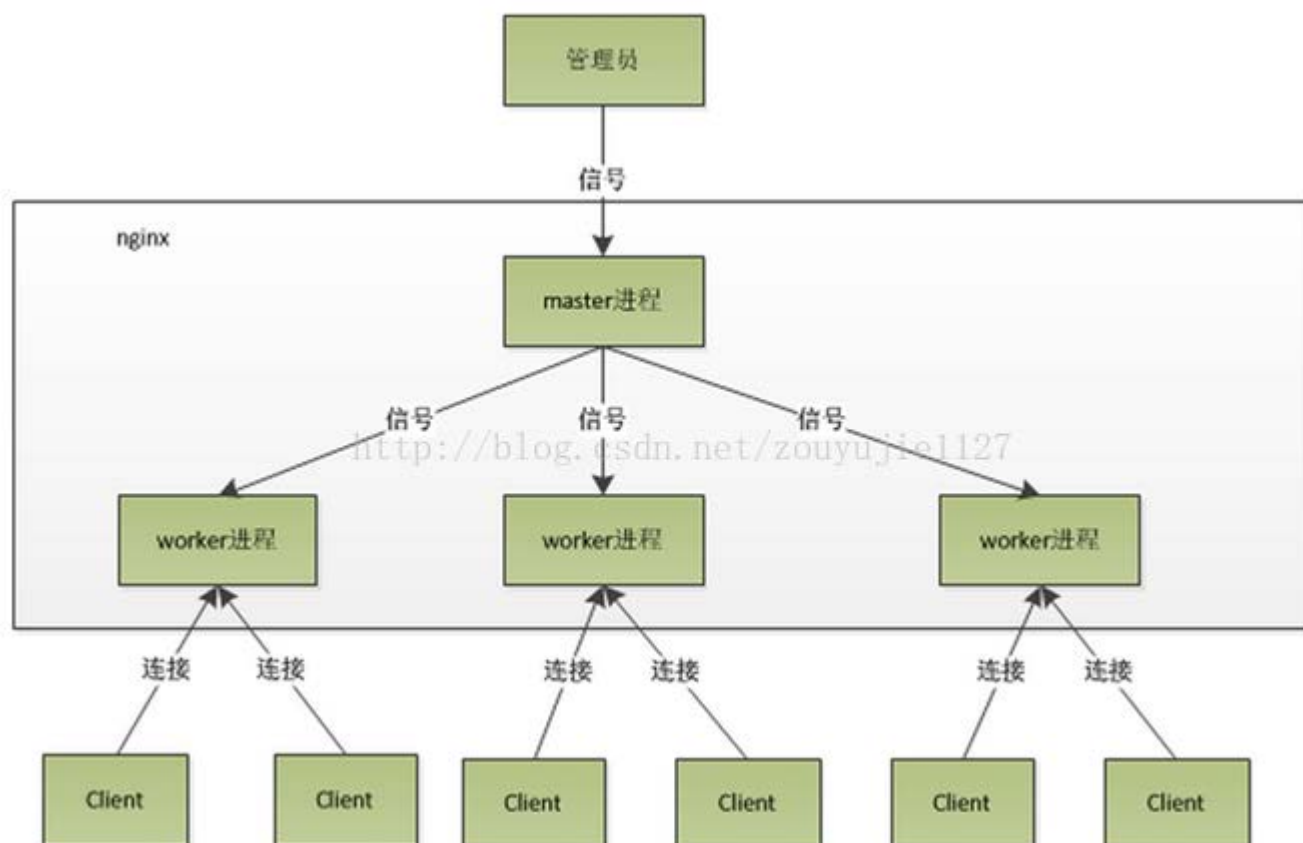
节省带宽：支持 GZIP 压缩，可以添加浏览器本地缓存的 Header 头。

稳定性高：用于反向代理，宕机的概率微乎其微

Nginx的不为人知的特点

- 1、nginx代理和后端web服务器间无需长连接；
- 2、接收用户请求是异步的，即先将用户请求全部接收下来，再一次性发送后后端web服务器，极大的减轻后端web服务器的压力
- 3、发送响应报文时，是边接收来自后端web服务器的数据，边发送给客户端的
- 4、网络依赖型低。NGINX对网络的依赖程度非常低，理论上讲，只要能够ping通就可以实施负载均衡，而且可以有效区分内网和外网流量
- 5、支持服务器检测。NGINX能够根据应用服务器处理页面返回的状态码、超时信息等检测服务器是否出现故障，并及时返回错误的请求重新提交到其它节点上

Nginx的内部(进程)模型



nginx是以多进程的方式来工作的，当然nginx也是支持多线程的方式的,只是我们主流的方式还是多进程的方式，也是nginx的默认方式。nginx采用多进程的方式有诸多好处。

(1) nginx在启动后，会有一个master进程和多个worker进程。master进程主要用来管理worker进程，包含：接收来自外界的信号，向各worker进程发送信号，监控 worker进程的运行状态,当worker进程退出后(异常情况下)，会自动重新启动新的worker进程。而基本的网络事件，则是放在worker进程中来处理了。多个worker进程之间是对等的，他们同等竞争来自客户端的请求，各进程互相之间是独立的。一个请求，只可能在一个worker进程中处理，一个worker进程，不可能处理其它进程的请求。worker进程的个数是可以设置的，一般我们会设置与机器cpu核数一致，这里面的原因与nginx的进程模型以及事件处理模型是分不开的。

(2) Master接收到信号以后怎样进行处理（./nginx -s reload）?首先master进程在接到信号后，会先重新加载配置文件，然后再启动新的进程，并向所有老的进程发送信号，告诉他们可以光荣退休了。新的进程在启动后，就开始接收新的请求，而老的进程在收到来自master的信号后，就不再接收新的请求，并且在当前进程中的所有未处理完的请求处理完成后，再退出。

(3) worker进程又是如何处理请求的呢？我们前面有提到，worker进程之间是平等的，每个进程，处理请求的机会也是一样的。当我们提供80端口的http服务时，一个连接请求过来，每个进程都有可能处理这个连接，怎么做到的呢？首先，每个worker进程都是从master进程fork过来，在master进程里面，先建立好需要listen的socket之后，然后再fork出多个worker进程，这样每个worker进程都可以去accept这个socket(当然不是同一个socket，只是每个进程的socket会监控在同一个ip地址与端口，这个在网络协议里面是允许的)。一般来说，当一个连接进来后，所有在accept在这个socket上面的进程，都会收到通知，而只有一个进程可以accept这个连接，其它的则accept失败，这是所谓的惊群现象。当然，nginx也不会视而不见，所以nginx提供了一个accept_mutex这个东西，从名字上，我们可以看这是一个加在accept上的一把共享锁。有了这把锁之后，同一时刻，就只会会有一个进程在accept连接，这样就不会有惊群问题了。accept_mutex是一个可控选项，我们可以显示地关掉，默认是打开的。当一个worker进程在accept这个连接之后，就开始读取请求，解析请求，处理请

求，产生数据后，再返回给客户端，最后才断开连接，这样一个完整的请求就是这样的了。我们可以看到，一个请求，完全由worker进程来处理，而且只在一个worker进程中处理。

(4):, nginx采用这种进程模型有什么好处呢？采用独立的进程，可以让互相之间不会互相影响，一个进程退出后，其它进程还在工作，服务不会中断，master进程则很快重新启动新的worker进程。当然，worker进程的异常退出，肯定是程序有bug了，异常退出，会导致当前worker上的所有请求失败，不过不会影响到所有请求，所以降低了风险。当然，好处还有很多，大家可以慢慢体会。

(5).有人可能要问了，nginx采用多worker的方式来处理请求，每个worker里面只有一个主线程，那能够处理的并发数很有限啊，多少个worker就能处理多少个并发，何来高并发呢？非也，这就是nginx的高明之处，nginx采用了异步非阻塞的方式来处理请求，也就是说，nginx是可以同时处理成千上万个请求的。对于IIS服务器每个请求会独占一个工作线程，当并发数上到几千时，就同时有几千的线程在处理请求了。这对操作系统来说，是个不小的挑战，线程带来的内存占用非常大，线程的上下文切换带来的cpu开销很大，自然性能就上不去了，而这些开销完全是没有意义的。我们之前说过，推荐设置worker的个数为cpu的核数，在这里就很容易理解了，更多的worker数，只会导致进程来竞争cpu资源了，从而带来不必要的上下文切换。而且，nginx为了更好的利用多核特性，提供了cpu亲和性的绑定选项，我们可以将某一个进程绑定在某一个核上，这样就不会因为进程的切换带来cache的失效

Nginx是如何处理一个请求

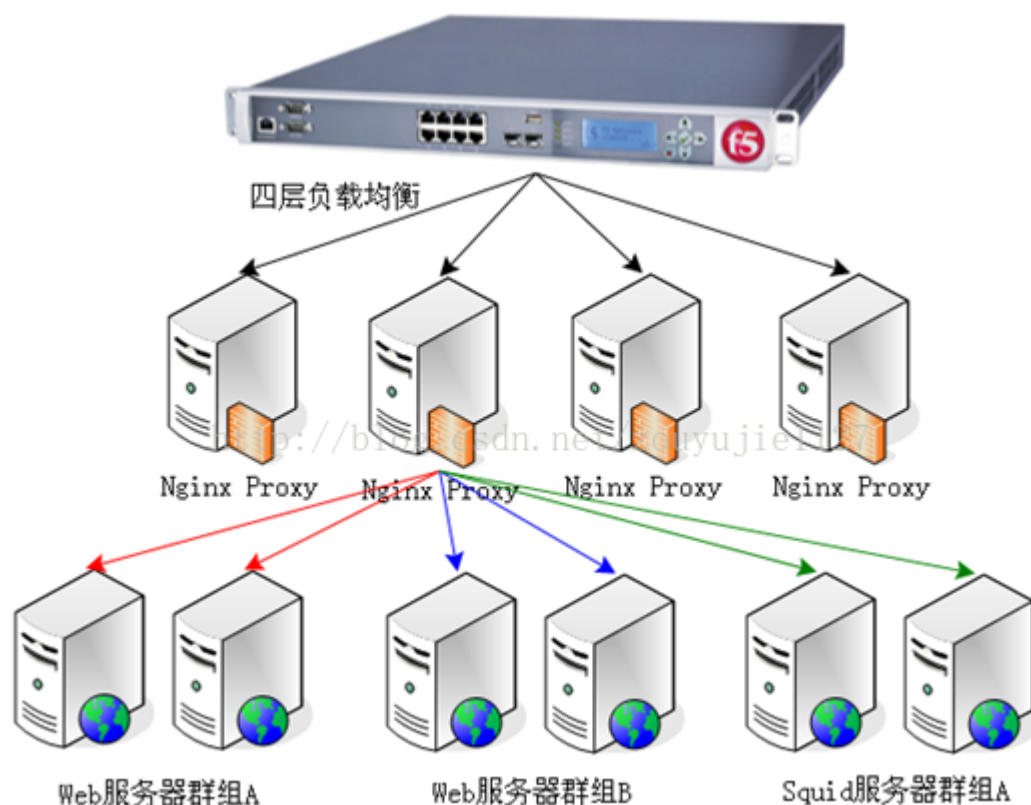
首先，nginx在启动时，会解析配置文件，得到需要监听的端口与ip地址，然后在nginx的master进程里面，先初始化好这个监控的socket(创建socket，设置addrreuse等选项，绑定到指定的ip地址端口，再listen)，然后再fork(一个现有进程可以调用fork函数创建一个新进程。由fork创建的新进程被称为子进程)出多个子进程出来，然后子进程会竞争accept新的连接。此时，客户端就可以向nginx发起连接了。当客户端与nginx进行三次握手，与nginx建立好一个连接后，此时，某一个子进程会accept成功，得到这个建立好的连接的socket，然后创建nginx对连接的封装，即ngx_connection_t结构体。接着，设置读写事件处理函数并添加读写事件来与客户端进行数据的交换。最后，nginx或客户端来主动关掉连接，到此，一个连接就寿终正寝了。

当然，nginx也是可以作为客户端来请求其它server的数据的（如upstream模块），此时，与其它server创建的连接，也封装在ngx_connection_t中。作为客户端，nginx先获取一个ngx_connection_t结构体，然后创建socket，并设置socket的属性（比如非阻塞）。然后再通过添加读写事件，调用connect/read/write来调用连接，最后关掉连接，并释放ngx_connection_t。

说明：nginx在实现时，是通过一个连接池来管理的，每个worker进程都有一个独立的连接池，连接池的大小是worker_connections。这里的连接池里面保存的其实不是真实的连接，它只是一个worker_connections大小的一个ngx_connection_t结构的数组。并且，nginx会通过一个链表free_connections来保存所有的空闲ngx_connection_t，每次获取一个连接时，就从空闲连接链表中获取一个，用完后，再放回空闲连接链表里面。

在这里，很多人会误解worker_connections这个参数的意思，认为这个值就是nginx所能建立连接的最大值。其实不然，这个值是表示每个worker进程所能建立连接的最大值，所以，一个nginx能建立的最大连接数，应该是worker_connections * worker_processes。当然，这里说的是最大连接数，对于HTTP请求本地资源来说，能够支持的最大并发数量是worker_connections * worker_processes，而如果是HTTP作为反向代理来说，最大并发数量应该是worker_connections * worker_processes/2。因为作为反向代理服务器，每个并发会建立与客户端的连接和与后端服务的连接，会占用两个连接。

Nginx典型的应用场景



负载均衡技术在现有网络结构之上提供了一种廉价、有效、透明的方法，来扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。它有两方面的含义：首先，大量的并发访问或数据流量分担到多台节点设备上分别处理，减少用户等待响应的时间；其次，单个重负载的运算分担到多台节点设备上做并行处理，每个节点设备处理结束后，将结果汇总，返回给用户，系统处理能力得到大幅度提高。

Nginx的应用

- 1、到官网下载Windows版本，下载地址：<http://nginx.org/en/download.html>
- 2、解压到磁盘任一目录
- 3、修改配置文件：具体参考备注。
- 4、启动服务：直接运行nginx.exe，缺点控制台窗口关闭，服务关闭。守护进程的方式启动：start nginx.exe
- 5、停止服务：nginx -s stop

重新加载配置：nginx -s reload

Nginx常见配置说明

worker_processes 8;

#nginx进程数，建议设置为等于CPU总核心数

worker_connections 65535;

#单个进程最大连接数（最大连接数=连接数*进程数）

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

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

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

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。weight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 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;

}
```

模块参数



```
#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 ar/loginx/error.log info;

#进程文件

pid ar/runinx.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textml，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。weight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 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)$
{
    expires 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 ar/loginx/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 confpasswd;
    #htpasswd文件的内容可以用apache提供的htpasswd工具来产生。
}

#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; }
}
}
```



更详细的模块参数请参考：<http://wiki.nginx.org/Main>

案例

Nginx+IIS服务器搭建服务器集群



配置静态资源



```
location ~ \.(jpg|png|jpeg|bmp|gif|swf|css)$
{
    expires 30d;
    root /nginx-1.4.7;#root:
    break;
}
```



扩展

负载均衡策略：<http://baidutech.blog.51cto.com/4114344/1033718/>



博客地址：<http://www.cnblogs.com/jiekzou/>

本文以学习、研究和分享为主，欢迎转载，但必须在文章页面明显位置给出原文连接。

博客版权：

如果文中有不妥或者错误的地方还望高手的你指出，以免误人子弟。如果觉得本文对你有所帮助不如【推荐】一下！如果你有更好的建议，不如留言一起讨论，共同进步！

再次感谢您耐心的读完本篇文章。

打开支付宝首页搜索“515227989”，领红包就是对我写作最大的支持！

其它：

.net-QQ群4：612347965 java-QQ群：805741535

我的拙作[《ASP.NET MVC企业级实战》](#)已经出版，希望大家多多支持！

分类: [Redis](#)

好文要顶

关注我

收藏该文



邹琼俊

关注 - 31

粉丝 - 1805

+加关注

« 上一篇：[6、ASP.NET MVC入门到精通——ASP.Net的两种开发方式](#)

» 下一篇：[ASP.NET Redis 开发](#)

posted @ 2015-05-07 23:10 [邹琼俊](#) 阅读(17997) 评论(11) 编辑 收藏

评论列表

#1楼 2015-05-08 09:22 [李维](#)

非常不错，学习了！

支持(0) 反对(0)

#2楼 2015-05-08 09:35 [wudhu](#)

写得很容易让人理解，谢谢分享。

支持(0) 反对(0)

#3楼 2015-05-08 11:32 [seabluescn](#)

用这个方式的集群，可以实现session共享吗？如果一台应用服务器宕机，是否可以无缝切换到其他应用服务器？

支持(0) 反对(0)

#4楼[楼主] 2015-05-08 11:36 [邹琼俊](#)

@ seabluescn


当然可以，具体实现请参考：<http://blog.csdn.net/xluren/article/details/16951247>

支持(0) 反对(0)

#5楼 2015-05-08 16:13 datoutie 

想请教下 我只用过windows版的 用来负载均衡
默认worker_connections是1024 写成 65535的话
在高并发的时候会出现
2015/01/26 17:40:26 [alert] 34464#60068: 1024 worker_connections are not enough
2014/12/09 12:21:44 [error] 89352#4408: *3107977 maximum number of descriptors supported by select() is 1024 while waiting for request, client: 222.67.196.116, server: 0.0.0.0:9200
您知道是什么原因吗

支持(0) 反对(0)

#6楼[楼主 

@ datoutie
你修改worker_connections值时，是不能超过worker_rlimit_nofile的这个值的，看下你worker_rlimit_nofile的值是多少，nginx的最大连接数=worker_connections* worker_processes
你修改两台nginx.conf配置文件中的worker_connections = 小于等于worker_rlimit_nofile的值（原来为1024），并重启nginx

支持(0) 反对(0)

#7楼 2015-05-08 18:04 gisTao 

嗯不错，楼主码字辛苦

支持(0) 反对(0)

#8楼 2015-05-14 10:40 datoutie 

@ 邹琼俊
谢谢 记得以前有写过还是报错就改回1024了 下次并发的时候测试下

这两个我运行不了会报错 是我版本问题吗 现在用的是1.6.2
use epoll;
#limit_zone crawler \$binary_remote_addr 10m; #开启限制IP连接数的时候需要使用

支持(0) 反对(0)

#9楼 2015-07-19 21:23 CanntBelieve 


赞赞赞

支持(0) 反对(0)

#10楼 2015-12-15 14:35 \ X線長 

感觉在传智播客的公开课上听过！

支持(1) 反对(0)

#11楼 2016-06-20 01:05 monkey's 

写得很容易让人理解，谢谢分享。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

最新IT新闻:

· [拼多多创始人黄铮个人身价有望达99亿美元 超越刘强东](#)

- [谷歌的云平台Google Cloud全局负载均衡服务发生中断](#)
- [SUSE开发者提议在GCC编译器中用Python替代AWK](#)
- [谷歌语音助理服务新功能：为用户整合日常有用信息](#)
- [他们干地球最危险的工作：几乎全是男人 日薪近1万](#)
- » [更多新闻...](#)

最新知识库文章:

- [危害程序员职业生涯的三大观念](#)
- [断点单步跟踪是一种低效的调试方法](#)
- [测试 | 让每一粒尘埃有的放矢](#)
- [从Excel到微服务](#)
- [如何提升你的能力？给年轻程序员的几条建议](#)
- » [更多知识库文章...](#)

作者简介



喜欢双截棍、书法、历史、古典文学...

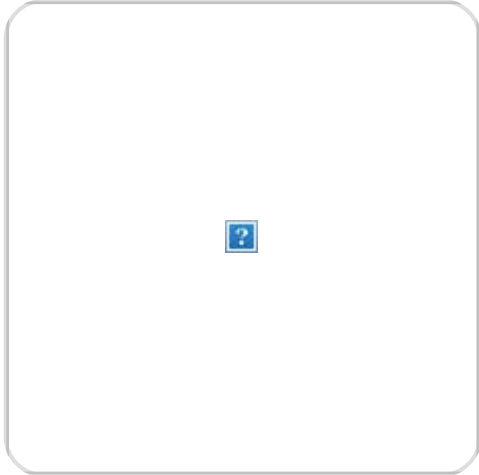
.net-QQ群4：[612347965](#)

java-QQ群：[805741535](#)

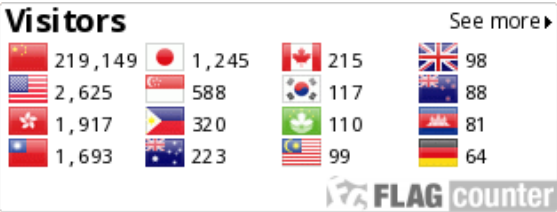
座右铭：天分是持续不断的忍耐

毕业于：湖南第一师范(2008)

我的著作



我的课程





[.net项目驱动学习](#)

难度：中级

类型：技术教程



[.Net高手速成秘籍](#)

难度：中级

类型：技术教程



[.NET开发技巧之工具篇](#)

难度：初级

类型：技巧分享



[赢在.NET面试求职](#)

难度：初级

类型：技巧分享

昵称：[邹琼俊](#)

园龄：[16年1个月](#)

荣誉：[.Net三好码农](#)

粉丝：[101805](#)

关注：[31](#)

[+加关注](#)

随笔分类(149)

[ASP.NET\(14\)](#)

[ASP.NET MVC\(46\)](#)

[C#\(7\)](#)

[HTML5\(7\)](#)

[IOC\(1\)](#)

[Java\(11\)](#)

[MongoDB\(5\)](#)

[MUI\(5\)](#)

[MySql\(5\)](#)

[Oracle\(1\)](#)

[ORM\(7\)](#)

[Redis\(4\)](#)

[SqlServer\(4\)](#)

[WCF\(3\)](#)

[WebAPI\(2\)](#)

[WF\(3\)](#)

[Winform\(3\)](#)

[开发工具\(12\)](#)

[前端\(2\)](#)

软件设计(1)
系统架构(1)
杂谈(5)

随笔档案(140)

- 2018年7月 (3)
- 2018年6月 (10)
- 2018年5月 (2)
- 2018年4月 (2)
- 2018年2月 (1)
- 2018年1月 (2)
- 2017年12月 (2)
- 2017年11月 (1)
- 2017年9月 (1)
- 2017年8月 (2)
- 2017年7月 (1)
- 2017年6月 (6)
- 2017年5月 (1)
- 2017年4月 (1)
- 2017年3月 (2)
- 2017年2月 (4)
- 2017年1月 (4)
- 2016年12月 (9)
- 2016年11月 (1)
- 2016年10月 (1)
- 2016年7月 (3)
- 2016年6月 (2)
- 2016年5月 (1)
- 2016年4月 (6)
- 2016年3月 (6)
- 2016年1月 (2)
- 2015年12月 (8)
- 2015年11月 (3)
- 2015年10月 (3)
- 2015年9月 (7)
- 2015年8月 (4)
- 2015年7月 (4)
- 2015年6月 (4)
- 2015年5月 (11)
- 2015年4月 (12)
- 2015年3月 (8)

积分与排名

积分 - 325303

排名 -17

最新评论

1. Re:MUI开发大全

@心存善念当然可以啊...

--邹琼俊

2. Re:ASP.NET MVC4入门到精通系列目录汇总

@心存善念18k~25k...

--邹琼俊

3. Re:MUI开发大全

之前下载了安装，然后就没然后了，想请假下 MUI做txt小说阅读器可行吗？

--心存善念

4. Re:ASP.NET MVC4入门到精通系列目录汇总

大佬，按你的标准的 高级.NET工程师 在深圳能有多少薪资

--心存善念

5. Re:ASP.NET Redis 开发

感谢楼主分享，赞一个。

--howdy

阅读排行榜

1. ASP.NET MVC搭建项目后台UI框架—1、后台主框架(26255)

2. C#批量插入数据到SqlServer中的四种方式(26223)

3. ASP.NET MVC企业级实战目录(22600)

4. 2、ASP.NET MVC入门到精通——Entity Framework入门(21728)

5. ASP.NET MVC4入门到精通系列目录汇总(20591)

6. Nginx详解-服务器集群(17994)

7. ASP.NET MVC导出excel（数据量大，非常耗时的，异步导出）(16306)

8. 22、ASP.NET MVC入门到精通——搭建项目框架(14709)

9. 网站服务架构(12959)

10. ASP.NET Redis 开发(12833)

评论排行榜

1. ASP.NET MVC企业级实战目录(180)

2. 博客搬家——从CSDN到博客园(112)

3. ASP.NET MVC4入门到精通系列目录汇总(92)

4. 从.Net到Java学习第一篇——开篇(69)

5. 学习.NET是因为热爱 or 兴趣 or 挣钱？(68)

6. ASP.NET MVC搭建项目后台UI框架—5、Demo演示Controller和View的交互(46)

7. 腾讯.NET&PHP面试题(41)

8. C#批量插入数据到SqlServer中的四种方式(39)

9. 网站服务架构(37)

10. ASP.NET MVC搭建项目后台UI框架—1、后台主框架(33)

推荐排行榜

- 1. ASP.NET MVC企业级实战目录(190)
- 2. ASP.NET MVC4入门到精通系列目录汇总(95)
- 3. 网站服务架构(92)
- 4. 博客搬家——从CSDN到博客园(79)
- 5. C#批量插入数据到SqlServer中的四种方式(67)
- 6. 腾讯.NET&PHP面试题(57)
- 7. ASP.NET MVC搭建项目后台UI框架—1、后台主框架(51)
- 8. ASP.NET Redis 开发(51)
- 9. 2、ASP.NET MVC入门到精通——Entity Framework入门(50)
- 10. 学习.NET是因为热爱 or 兴趣 or 挣钱 ? (49)