注:

Os: Linux 3.10.0-327.el7.x86\_64 x86\_64

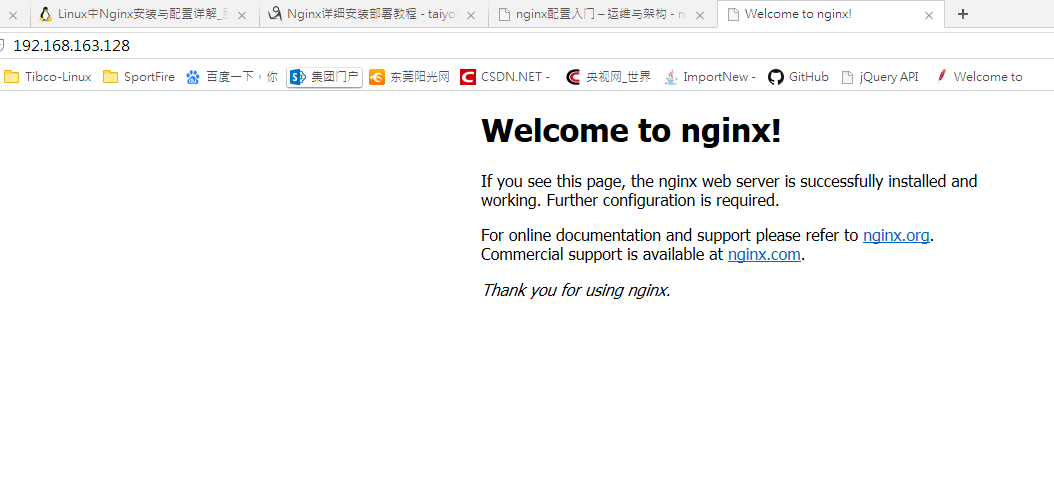
Nignx: nginx-1.12.2

Pcre:8.40

# Nginx安装

1. 运行指令yum install -y gcc gcc-c++

GCC是C语言的编译工具.由于Nginx HTTP模块可以使用C++来进行编写,所以可以安装G++编译器

1. 先官网下载对应的Nginx解压包. tar -xzvf pcre-8.40.tar.gz解压对应的文件
2. 下载<http://www.pcre.org/>. tar -xzvf pcre-8.40.tar.gz.因为在Nginx的配置文件中使用到了则表达式.所以需要安装pcre来解析配置文件.(也可以使用yum install -y pcre pcre-devel指令来安装pcre)
3. 运行指令yum install -y zlib-devel安装zlib库,由于Nginx会对HTTP包的内容做gzip格式的压缩.如果我们在Nginx.config里面配置了Gzip on,并指定对于某些类型的HTTP响应使用gzip来进行压缩减少网络传输量.那么必须在编译时吧zlib编译进Nginx
4. 分别进入已经解压的nginx和pcre的文件夹.运行指令./configure && make && make install进行编译安装,注意如果需要集成其他模块就必须先进行安装对应的模块功能再进行编译.
5. 确认防火墙是否开放访问.可以直接使用systemctl stop firewalld.service关闭防火墙.
6. 运行whereis nginx,发现安装路径为/usr/local/nginx.运行指令/usr/local/nginx/sbin/nginx运行Nginx
7. Nginx是高度自由化的Web服务器,它的功能由许多模块来支持,而这些模块可根据我们的需求来定制.如(zlib,OpenSSL)
8. 访问默认的index,成功安装则显示如下页面.

# Nginx配置

# Nginx指令集

指令规则nginx [-?hvVtq] [-s signal] [-c filename] [-p prefix] [-g directives]

./sbin/nginx 启动Nginx

./sbin/nginx -s reload|reopen|stop|quit #重新加载配置|重启|停止|退出 nginx

./sbin/nginx –h 打开帮助

./sbin/nginx -v : 显示版本信息并退出

./sbin/nginx -q : 在检测配置文件期间屏蔽非错误信息

-p prefix : 设置前缀路径（默认是：/usr/local/Cellar/nginx/1.2.6/）

-c filename : 自定配置文件（默认是：/usr/local/etc/nginx/nginx.conf）

-g directives : 设置配置文件外的全局指令

# 平滑升级Nginx

当Nginx服务升级到新的版本的时候,必须要将旧的二进制文件Nginx替换掉,

通常这种情况下需要重新启动Nginx.但是Nginx支持版本的平滑升级.

1）通知正在运行的旧版本Nginx准备升级。通过向master进程发送USR2信号可达到目

的。例如：

kill -s SIGUSR2 <nginx master pid>

这时，运行中的Nginx会将pid文件重命名，如将usrlocal/nginx/logs/nginx.pid重命名

为usrlocal/nginx/logs/nginx.pid.oldbin，这样新的Nginx才有可能启动成功。

2）启动新版本的Nginx，可以使用以上介绍过的任意一种启动方法。这时通过ps命令可

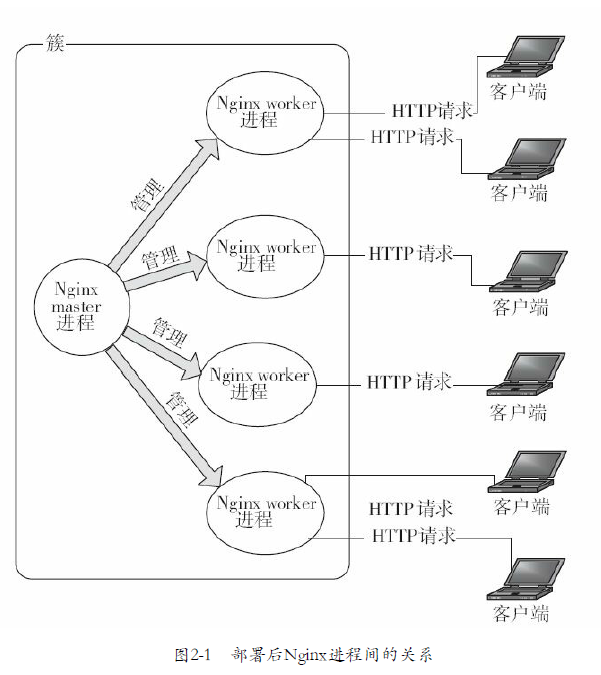
以发现新旧版本的Nginx在同时运行。

3）通过kill命令向旧版本的master进程发送SIGQUIT信号，以“优雅”的方式关闭旧版本的

Nginx。随后将只有新版本的Nginx服务运行，此时平滑升级完毕。

# 性能配置项

## 1.Worker进程数和Worker内核绑定



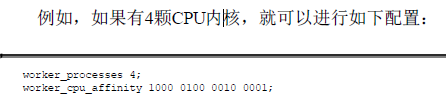
Nginx的设计模式:Master-Worker模式是一种将串行任务并行化的方案，被分解的子任务在系统中可以被并行处理，同时，如果有需要，Master进程不需要等待所有子任务都完成计算，就可以根据已有的部分结果集计算最终结果集。

在正式提供服务的产品环境下，部署Nginx时都是使用一个master进程来管理多个worker进程，一般情况下，worker进程的数量与服务器上的CPU核心数相等。每一个worker进程都是繁忙的，它们在真正地提供互联网服务，master进程则很“清闲”，只负责监控管理worker进程。worker进程之间通过共享内存、原子操作等一些进程间通信机制来实现负载均衡等功能

由于设计模式上的原因,Worker进程的数量设置会直接影响Nginx的工作性能,在默认的情况下worker\_processes 1;每个Worker进程都是单线程的进程,他们会调用各个模块以实现多样的功能,当大量的阻塞性调用出现时,过少的Worker数会导致服务整体性能下降,但若Worker数多于CPU的内核数时,会增大进程间的切换带来的消耗(Linux抢占式内核).也由于Linux的抢占式内核,会出现多个Worker进程抢占同一个CPU的情况,在这种情况下会出现同步问题.如果这个时候指定每个Worker都独享一个CPU内核没这样在内核的调度策略上实现了完成的并发,也减低了不同进程之间切换带来的消耗.

通过上述描述,正确的设置Worker数量和绑定Worker进程到指定的CPU内核可以大大的增强Nginx的性能.

推荐下Worker的数量应该给CPU的内核数保持一致达到最优性能.



## 2.SSL硬件加速

如果服务器上有SSL硬件加速设备,那么可以进行通过语法ssl\_engine device配置,加快SSL协议的处理速度.

## 3.系统调用Gettimeofday的执行频率

## 4.Nginx Worker进程优先级

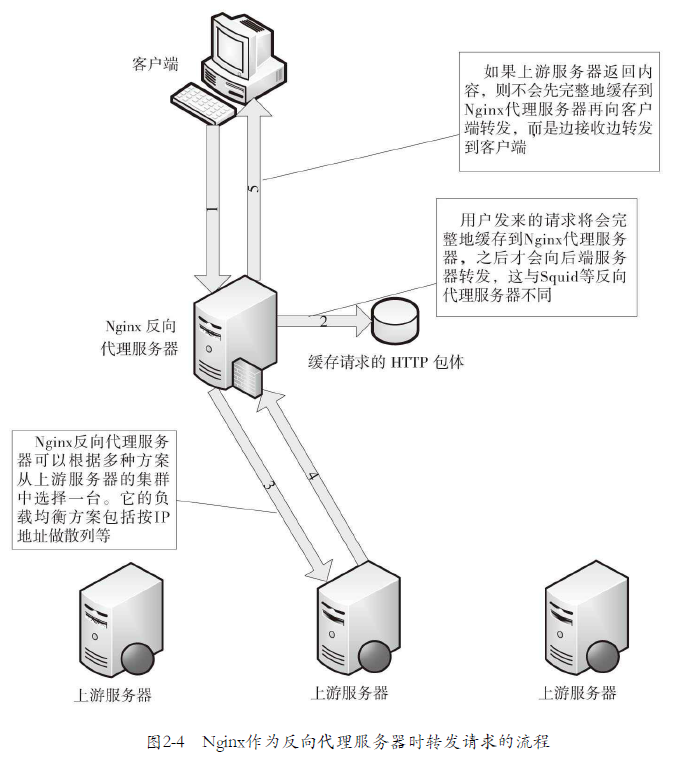
在Linux或者其他类Unix系统中,许多进程都处于可执行状态时,按照所有进程的优先级来决定本次内核选择哪一个进程执行.进程所分配的CPU时间片的大小也与进程优先级相关,优先级越高,进程分配到的时间片也就越大.

语法： worker\_priority nice;

默认： worker\_priority 0;

# 反向代理

## 1.代理流程



## 2.优缺点

Nginx的这种工作方式有什么优缺点呢？很明显，缺点是延长了一个请求的处理时间，并增加了用于缓存请求内容的内存和磁盘空间。而优点则是降低了上游服务器的负载，尽量把压力放在Nginx服务器上。

## 3.负载均衡

作为代理服务器,一般需要想上游服务器的集群转发请求.这里的负载均衡是指一种策略,尽量把请求平均地分布到每一台上有服务器上.

1.upstream定义一个上有服务器的集群,方便方向代理中的proxy\_pass使用.例如:

upstream backend{

server backend1.example.com;

server backend2.example.com;

server backend3.example.com;

}

server{

location /

}