1. 什么是Nginx

一个高性能的web服务器，可以用作反向代理、负载均衡，可以通过模块扩展其功能，比如说通过模块支持动态页面。

1. 为什么要用Nginx(Nginx优点)

* nginx处理静态文件速度快，使用了零拷贝
* 高性能，使用IO 多路复用。
* 高度模块化的设计，编写和使用模块相对简单(ssl模块)
* 轻量级，同样起web 服务，比apache 占用更少的内存及资源
* Nginx 配置简洁
* 反向代理、负载均衡(功能需求)

1. IO 多路复用
2. Nginx缺点

* 动态请求(动态页面?)相对较弱
* Nginx是多进程单线程的，线程异常可能会影响多个连接

1. Nginx的master进程和worker进程
2. Rewrite、proxy\_pass区别

* rewrite

url重写，是在nginx服务内重新匹配新的url，把请求内容带过去。也可以通过返回30x进行重定向。

* proxy\_pass

proxy\_pass是反向代理，把请求内容发送到指定的url中，一般是跨进程的，会有socket网络资源的创建和维护。如果是服务内的url，使用rewiter就行了

1. location表达式优先级

* 精确匹配 location = /abc { }
* 匹配路径的前缀，如果找到停止搜索 location ^~ /abc { }
* 不区分大小写的正则匹配 location ~\* /abc { }

应该是安装配置文件的配置顺序，返回优先匹配到的。

* 正则匹配 location ~ /abc { }

应该是安装配置文件的配置顺序，返回优先匹配到的。

* 普通路径前缀匹配 location /abc { }

应该会匹配最大前缀

(整理一个博客文章)

1. nginx负载均衡策略

* 轮询

此策略适合服务器配置相当，无状态且短平快的服务使用

* Weight

根据权重来指定轮询几率，权重越高分配到需要处理的请求越多。

此策略可以与least\_conn和ip\_hash结合使用。

此策略比较适合服务器的硬件配置差别比较大的情况。

* ip\_hash

指定负载均衡器按照基于客户端IP的分配方式，这个方法确保了相同的客户端的请求一直发送到相同的服务器，以保证session会话

#动态服务器组

upstream ip\_hash {

ip\_hash; #保证每个访客固定访问一个后端服务器

server localhost:8080 weight=2; #tomcat 7.0

server localhost:8081; #tomcat 8.0

server localhost:8082; #tomcat 8.5

server localhost:8083 max\_fails=3 fail\_timeout=20s; #tomcat 9.0

}

在nginx版本1.3.1之前，不能在ip\_hash中使用权重（weight）。

(ip\_hash中使用权重可能是让权重大的hash槽大一点，这样落入的连接多一点)

ip\_hash不能与backup同时使用。

此策略适合有状态服务，比如session。

当有服务器需要剔除，必须手动down掉。

* least\_conn

把请求转发给连接数较少的后端服务器。轮询算法是把请求平均的转发给各个后端，使它们的负载大致相同；但是，有些请求占用的时间很长，会导致其所在的后端负载较高。这种情况下，least\_conn这种方式就可以达到更好的负载均衡效果。

#动态服务器组

upstream dynamic\_zuoyu {

least\_conn; #把请求转发给连接数较少的后端服务器

server localhost:8080 weight=2; #tomcat 7.0

server localhost:8081; #tomcat 8.0

server localhost:8082 backup; #tomcat 8.5

server localhost:8083 max\_fails=3 fail\_timeout=20s; #tomcat 9.0

}

此负载均衡策略适合请求处理时间长短不一造成服务器过载的情况。

* fair（第三方）

按照服务器端的响应时间来分配请求，响应时间短的优先分配。

和less\_conn的区别？

感觉最终效果上看是一致的，处理速度慢的服务器处理的连接都会比较少连接量少的时候，fair算法应该优势更明显一点。

#动态服务器组

upstream dynamic\_zuoyu {

server localhost:8080; #tomcat 7.0

server localhost:8081; #tomcat 8.0

server localhost:8082; #tomcat 8.5

server localhost:8083; #tomcat 9.0

fair; #实现响应时间短的优先分配

}

* url\_hash（第三方）

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，要配合缓存命中来使用。同一个资源多次请求，可能会到达不同的服务器上，导致不必要的多次下载，缓存命中率不高，以及一些资源时间的浪费。而使用url\_hash，可以使得同一个url（也就是同一个资源请求）会到达同一台服务器，一旦缓存住了资源，再此收到请求，就可以从缓存中读取。

Ip\_hash应该也能使用缓存，因为所有相同ip的访问都在同一个服务器上，url\_hash的话，会存储相同ip不同url访问的不同的服务器，不利于有状态的请求。url\_hash和ip\_hash都会存在资源倾斜问题，url\_hash如果有某个几个url经常要用到，且在同一个哈希槽，那么对应的服务器压力就会很大，ip\_has也是，如果很多ip的哈希都在一个哈希槽，那么对应的服务器压力会增大。

#动态服务器组

upstream dynamic\_zuoyu {

hash $request\_uri; #实现每个url定向到同一个后端服务器

server localhost:8080; #tomcat 7.0

server localhost:8081; #tomcat 8.0

server localhost:8082; #tomcat 8.5

server localhost:8083; #tomcat 9.0

}

1. Nginx限流

limit\_req\_zone ：用在http上

limit\_req：用在http、server、location上

1. Nginx 黑名单

Nginx的限流模块，是基于漏桶算法实现的，在高并发的场景下非常实用。

deny 192.168.1.1/24

allow 192.168.1.1/24

deny all

以上是使用子网掩码来表示一个网段，如果不是同一个网段需要写多个allow和deny

默认是都允许访问的，所以当需要配置禁止某个ip时，只需要deny ip就行了。

当配置只允许某个ip访问时，需要先deny all，然后再allow ip

1. Nginx 缓存

proxy\_cache 设置nginx代理端缓存，url进来时先看有没有缓存，有就用缓存，否则就 代理。

1. ngx\_http\_upstream\_module

这个是nginx的upstream模块，可以通过该模块进行tcp代理，比如说ssh连接

(延伸，upstream是作用在第几层？)

1. 解释如何在Nginx服务器上添加模块

编译时使用./configure配置模块，进行编译。