Nginx

2018年9月21日 9:49

易学存在的缺点：

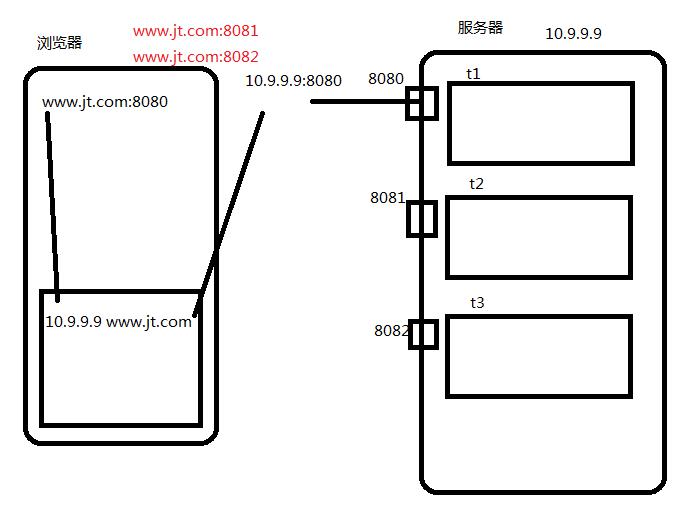
1. 性能不高，搜索使用like，数据全部集中到数据库
2. 并发不高，单节点的tomcat极限2000并发（与服务器性能有关），一般能做到200就不错了。
3. 非高可用。

京淘框架的目的：分布式高并发高可用的系统架构。

nginx可以解决单个tomcat并发不高的问题；

如果我们需要实现一个高并发的tomcat集群，如何设计？

越多的tomcat节点，整体并发量越高，但是tomcat集群在访问时会有问题？

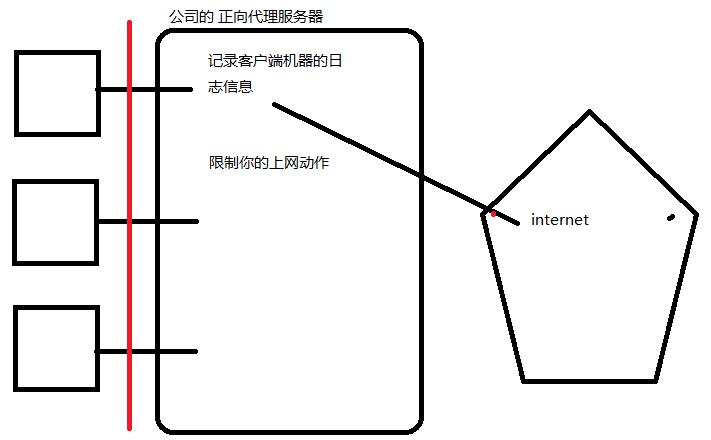


Nginx

是一个高性能的**HTTP服务器**和**反向代理**服务器

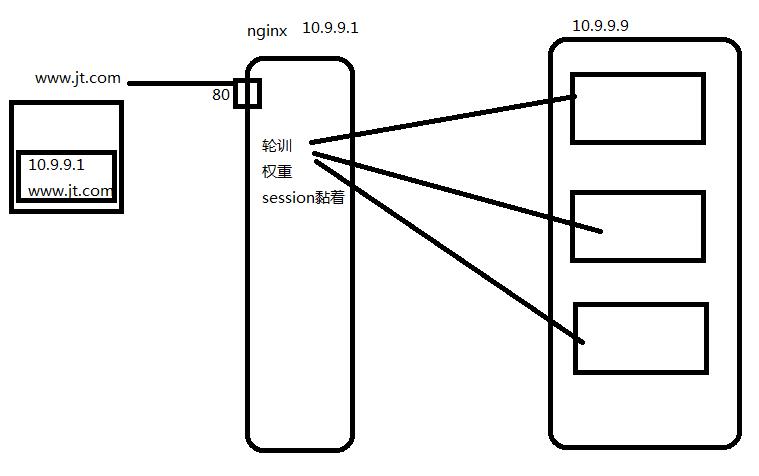
分区 day09all 的第 7 页

代理服务器



收集客户端pc机日志信息，限制pc端的客户端访问某些网站，但是对于可以访问的各种网站不做返回的限制；

反向代理服务器就可以控制访问的服务器资源。



分区 day09all 的第 8 页

如上图，并发压力的入口，集中在nginx的并发访问上nginx作为软件性能超高，并发量在3万-10万；F5是一种硬件的代理服务器，50万。

Nginx安装

windows配置nginx，linux配置到完成学习nginx之后；绿色安装解压文件到没有中文和空格的路径即可；

启动和关闭

windows的nginx默认配置使用多线程，在启东时，可以看到2个进程，一个是master管理守护进程，一个是worker工作进程，并发量配置更高的话需要对worker的数量进行调整；

在nginx的根目录右2个文件

start.bat 启动文件

stop.bat 关闭文件

reload.bat 重启文件（不推荐使用）

重启使用stop，观察任务管理器，确定2个nginx进程消失，在点start，否则会出现4个，6个nginx进程无法工作

配置一个入门的案例

需求：利用nginx访问当前windows的springboot工程

原来访问：localhost:8090,利用nginx在不改变tomcat配置的情况下 使用 [www.springboot.com](http://www.springboot.com)

nginx启动会加载核心配置文件，nginx根目录conf下/nginx.conf

nginx.conf文件的结构

作为http服务器反向代理的能力和功能都是在一个

分区 day09all 的第 9 页

http{}完成配置的

与其相同级别的其他内容都与当前nginx的资源配置有关

目前想通过nginx代理，访问后台的localhost:8090需要配置一个

server的内容

#后台服务器

server { //启动nginx时会根据server的个数加载虚拟机（单

个http服务器）单位

listen 80; 当前的服务器监听的端口

server\_name [www.springboot.com](http://www.springboot.com); //监听访问的host域名，可以通过访问的请求对象中的host头获取域名的监听

[www.springboot.com/index/haha.jsp](http://www.springboot.com/index/haha.jsp)

location / { //当满足监听的条件时，location后跟着的匹配信息，将会对访问的url进行匹配判断,"/"表示任何的字符串都匹配，一旦满足就进入location中的处理逻辑

proxy\_pass <http://127.0.0.1:8090>; //进入location

后，进行转发的替换，把proxy\_pass中的内容替换访问url中的域名和端口

<http://127.0.0.1:8090/index/haha.jsp> proxy\_connect\_timeout 600; proxy\_read\_timeout 600;

}

}

如何让www.springboot.com访问到nginx所在服务器？

在host中配置一个ip地址和域名的映射，ip地址指向nginx所在服务器

127.0.0.1 [www.springboot.com](http://www.springboot.com)

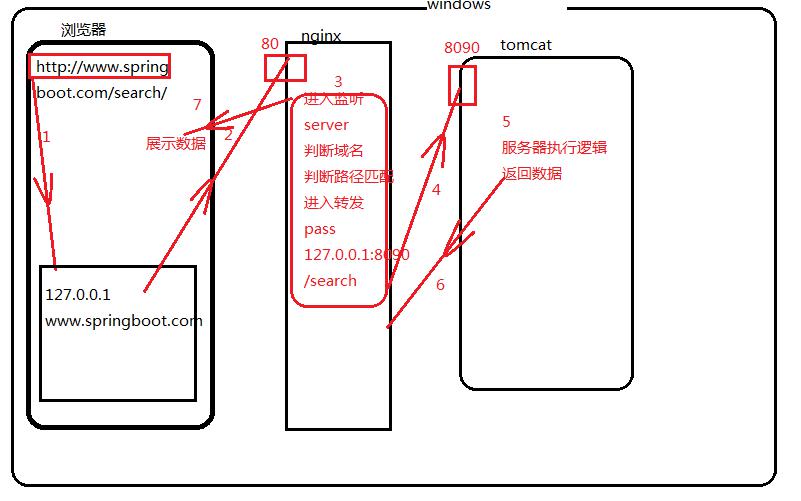
分区 day09all 的第 10 页

启动如果报错，可以观察logs中的error.log文件，内容非常准确

总结配置步骤：

1. 配置nginx的server
2. 配置host
3. 启动后台工程
4. 启动nginx测试访问

具有nginx代理的访问流程



location的匹配原则（优先级）

* 精确匹配=
* 字符串前缀匹配^~
* 按照配置顺序的正则匹配~
* 不带修饰的前缀匹配
* 当前所有匹配完成时，停止匹配，按照当前匹配规则处理请求
* 前缀匹配，有包含关系式，按照最大匹配长度原则优先
* 最低的优先级时通配符 /

分区 day09all 的第 11 页

监听80和localhost

location= /images{

return 205;}

location^~/images{

return 204}

location^~/images/test.png{

return 203}

location~\*.png${

return 202}

location~\*.(png|jpg|gif)${

return 201}

location /{

return 200}

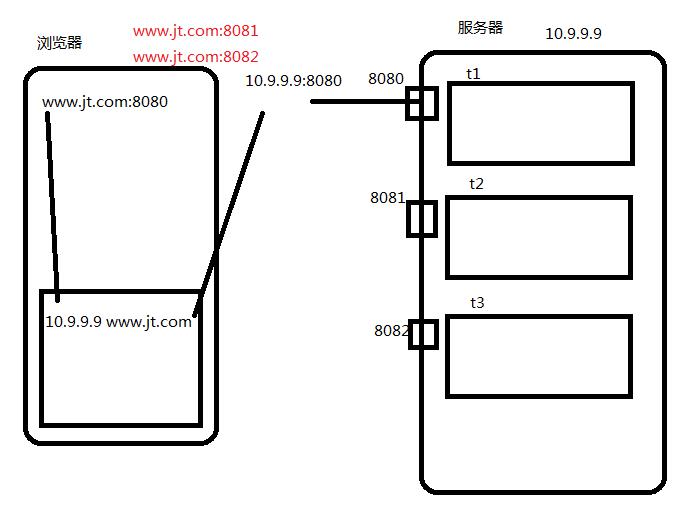
<http://localhost/>返回200 <http://localhost/images>205 <http://localhost/images/id>204 <http://locahost/images/test>204 <http://localhost/images/test.png>203 <http://localhost/haha.png>202

分区 day09all 的第 12 页

tomcat多实例部署

2018年9月21日 11:16

tomcat的多实例部署结构：一台服务器有多个tomcat实例



1. 安装3个tomcat

分别解压到不同的文件名称中

需要将apache的tomcat的tar包解压到一个文件目录,拷贝成3份

/home/software/tomcat/t1，/home/software/tomcat/t2，/home/software/tomcat/t3，

linux的tomcat启动，需要进入bin目录

执行命令

#startup.sh

查看日志，需要调用命令

分区 day09all 的第 13 页

#tail -f t1/logs/catalina.out

查看java的进程ps -ef|grep java

还可以使用jps查看java进程

1. 准备访问的默认网页

t1,t2,t3编写各自的首页，区分访问；

将每个tomcat的webapp中的内容删除，保留一个ROOT下的

index.jsp，编写各自的内容

hello I am from t1,t2,t3

1. 修改各自的端口

server.xml当中修改启动相关的端口

8080（69行）,8005(22行),8009（116左右）

t1:8091,8015,8019

t2:8092,8025,8029

t3:8093.8035,8039

4 各自启动tomcat

分区 day09all 的第 14 页

Nginx负载均衡

2018年9月21日 11:41

负载均衡：

代理可以根据配置，根据机器的性能，分配不同比例的访问压力；

Nginx三种负载均衡的配置方式

1. 轮训（物理均衡，平均分配）

测试访问后台的3个tomcat，分别循环返回t1,t2,t3

在nginx的配置文件中添加负载均衡的配置关键字

upstream jt1805 { //upstream 开启了负载均衡可能使用到的访问的服务器连接库，jt1805自定义的名称，相当于域名，虚拟域名，**利用nginx访问tomcat9时，虚拟域名不能有特殊符号\_,**

server 106.75.120.140:8091 ;

server 106.75.120.140:8092 ;

server 106.75.120.140:8093 ;

}

需要在location中指定proxy\_pass，不能指向一个固定的后台服务器；

server {

listen 80;

server\_name [www.springboot.com](http://www.springboot.com); location / {

分区 day09all 的第 15 页

#proxy\_pass <http://127.0.0.1:8090>; proxy\_pass <http://jt1805>; proxy\_connect\_timeout 600; proxy\_read\_timeout 600;}}

测试

[www.springboot.com](http://www.springboot.com) 访问默认首页，看返回结果

可以完成tomcat集群访问的高可用；

1. 权重

轮训的基础上，给对应服务器访问地址配置关键字 weight = 权重值down；不在访问当前服务器

利用代码，计算访问A,B,C,D服务器的访问比

例，4:3:2:1？

利用hash取余

“访问客户端的ip地址+随机数”对10取余

0,1,2,3，交给A

4,5,6，交给B

7,8，交给C

9，交给D

在upstream中的服务器库里，配置权重值

server 106.75.120.140:8091 weight=5 ; server 106.75.120.140:8092 weight=1; server 106.75.120.140:8093 down;

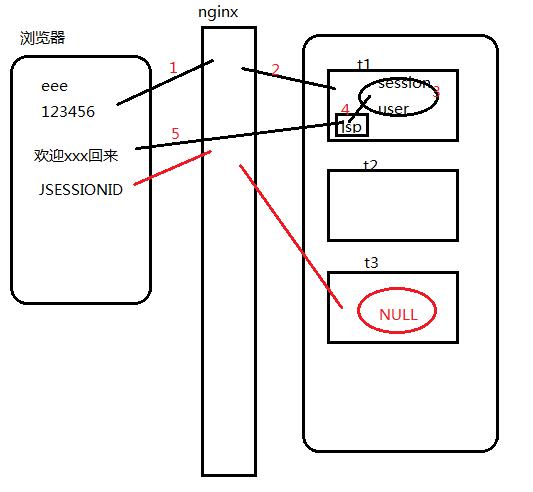
分区 day09all 的第 16 页

1. session黏着(不使用) session共享问题；

**单个工程**的逻辑中，经常使用会话技术的session将数据从web容器中返回页面使用；

比如：欢迎XXXX回来；

登录--校验成功存入session--响应页面--页面jsp获取session--取得user信息--展示页面欢迎XXXX回来



单个web容器的访问。不存在session共享，一旦搭建tomcat集群，每次访问的都不是同一个tomcat，会导致不同tomcat间的session无法相互共享数据的问题，如果以登录逻辑为例，会导致用户的每次访问都会要求重新登录；

这种问题就是session共享的问题；

nginx的解决办法就是

分区 day09all 的第 17 页

ip\_hash的黏着；

upstream jt1805{

ip\_hash

server 106.75.120.140:8091;

server 106.75.120.140:8092;

server 106.75.120.140:8093;}

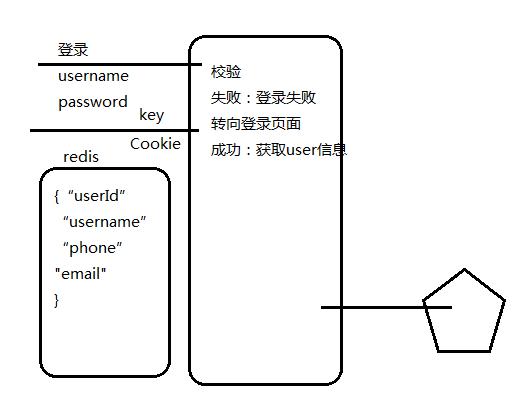
这种黏着逻辑的算法非常简单：

利用客户端的ip地址进行hash取余计算，轮训的情况对服务器的数量取余；每次访问时，计算的取余结果都是一样的，造成当前nginx只会把同一个ip客户端转发到同一个tomcat，不会出现session共享的问题了；

目前阶段解决session共享问题：

某种技术配合redis

京淘：SSO单点登录+redis



springSession+redis

分区 day09all 的第 18 页

你该怎么用session存储数据，就怎么用session；

需要在使用的session对象上，把这个对象放到redis中；

第 29 页