LaveCoral

nginx配置负载均衡

nginx负载均衡

> 负载均衡建立在现有网络结构之上，提供了一种廉价有效透明的方法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。

随着网站的发展，服务器压力越来越大，我们可能首先会将数据库，静态文件分离出去。但是随着发展，单独业务API的请求的压力也会变得很大，这时候我们可能需要做负载均衡将一台服务器面临的压力分散到多台服务器上。

nginx 不仅可以作为强大的web服务器，也可以作为反向代理服务器，而且nginx还可以按照调度规则实现动静分离，还可以对后端的服务器做负载均衡。

nginx负载均衡配置

nginx的负载均衡主要是对proxy_pass和upstream的配置。

我们首先需要建立一个Spring Boot的项目对外提供服务，来模拟我们实际的服务，还可以配置其它可以提供网络请求处理的框架来提供服务，这里具体由什么来提供服务和nginx配置并不相关。

Spring Boot 项目建立的过程就不提了，这里只贴一下主要的代码：

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
```

Spring Boot(2)
nginx(1)
Spring Cloud(1)

随笔分类
Spring Cloud(1)

随笔档案
2019年3月(3)
2019年2月(1)

阅读排行榜
1. nginx配置负载均衡(41259)
2. 在Spring Boot项目中使用Redis集群(782)
3. 在Spring Boot项目中使用Redis集群(续)(119)
4. Spring Cloud - 前言(118)

推荐排行榜
1. nginx配置负载均衡(2)

```
@RestController
@SpringBootApplication
public class NginxtestApplication {

    @Value("${server.port:}")
    private String port;

    public static void main(String[] args) {
        SpringApplication.run(NginxtestApplication.class, args);
    }

    @GetMapping("")
    public String hello() {
        System.out.println("call me " + port);
        return "i am " + port;
    }

}
```

将项目打包后，我们执行下面的命令

```
java -jar test.jar --server.port=8001
java -jar test.jar --server.port=8002
```

接下来我们打开nginx的配置文件

```
http {
    upstream upstream_name{
        server 192.168.0.28:8001;
        server 192.168.0.28:8002;
    }

    server {
        listen       8080;
        server_name  localhost;
```

```
        location / {
            proxy_pass http://upstream_name;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }
}
```

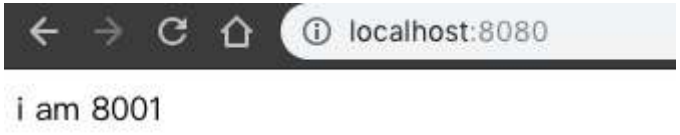
我这里没有把默认的一些配置贴出来。

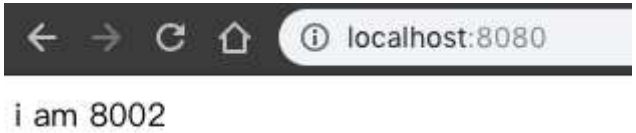
首先，在http下添加 `upstream upstream_name {}` 来配置要映射的服务器。

其中的 `upstream_name` 大家可以指定为服务的域名或者项目的代号。

server下的location 我们将 `/` 下的全部请求转发到 `http://upstream_name` ，也就是我们上面配置的服务器列表中的某一台服务器上。具体是哪台服务器，nginx会根据配置的调度算法来确认。

我们在浏览器中打开 `localhost:8080` 。多刷新几次就可以看到页面上的内容发生了变化。





nginx负载均衡策略

nginx的负载均衡策略有4种：

轮询(默认)

最基本的配置方法，它是upstream的默认策略，每个请求会按时间顺序逐一分配到不同的后端服务器。

参数有：

参数	描述
fail_timeout	与max_fails结合使用
max_fails	设置在fail_timeout参数设置的时间内最大失败次数，如果在这个时间内，所有针对该服务器的请求都失败了，那么认为该服务器会被认为是停机了
fail_time	服务器会被认为停机的时间长度,默认为10s。
backup	标记该服务器为备用服务器。当主服务器停止时，请求会被发送到它这里。
down	标记服务器永久停机了。

注意：

- 在轮询中，如果服务器down掉了，会自动剔除该服务器。
- 缺省配置就是轮询策略。
- 此策略适合服务器配置相当，无状态且短平快的服务使用。

权重

在轮询策略的基础上制定沦陷的几率。例如

```
upstream foo {
    server localhost:8001 weight=2;
    server localhost:8002;
    server localhost:8003 backup;
    server localhost:8004 max_fails=3 fail_timeout=20s;
}
```

这里例子中，weight参数用于制定轮询的几率，weight默认值为1；weight的数值和被访问的几率成正比。

注意：

- 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

负载均衡器按照客户端IP地址的分配方式，可以确保相同客户端的请求一直发送到相同的服务器。这样每个访客都固定访问一个后端服务器。

```
upstream foo {
    ip_hash;
    server localhost:8001 weight=2;
    server localhost:8002;
    server localhost:8003;
    server localhost:8004 max_fails=3 fail_timeout=20s;
}
```

注意：

- 在nginx版本1.3.1之前，不能在ip_hash中使用权重（weight）。
- 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 最小连接

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

```
upstream foo {
    least_conn;
    server localhost:8001 weight=2;
    server localhost:8002;
    server localhost:8003 backup;
```

```
    server localhost:8004 max_fails=3 fail_timeout=20s;
}
```

注意：

- 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。

除了上面这些调度策略之后，还有一些第三方的调度策略可以集成到nginx中。

在实际运用中，需要根据不同的场景选择不同的策略，大多是多种策略结合使用以达到实际需求的性能。

标签: nginx

LaveCoral
关注 - 3
粉丝 - 1
+加关注

2 0

« 上一篇： Spring Cloud - 前言
» 下一篇： 在Spring Boot项目中使用Redis集群

posted @ 2019-03-05 16:00 LaveCoral 阅读(41260) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页