

## 系统保护

- 限流，限制系统访问流量。(部分用户不可用)

## 如何实现限流

- 限制总的并发数，线程池、连接池
- 显示瞬时并发数，
- 限制时间窗口内的平均速率 Guava RateLimiter

## 常见的限流算法

访问的限制规则 (tps=1000)

限制的访问资源

[统计指标数据]

- 计数器(Semaphore )  
发短信验证码 (60s 只允许发一次)
- 滑动窗口(Hystrix)
- 漏桶
- 令牌桶

## 分布式限流

Redis - Lua实现令牌桶 ( Redisson)

Zookeeper

## Sentinel

Sentinel的核心概念

1. 资源 (针对谁去限流) , 服务、接口、方法、链路。
2. 规则 (限流的判断规则) , 总的并发数量? 并发线程数量?

流量控制之后的处理方法:

- 抛异常
- 排队
- 降级..

目前最新: 1.8.3

Spring Cloud Alibaba 2.2.6.RELEASE , sentinel版本 ->1.8.1

## Sentinel控制台安装

1. 下载sentinel-dashboard-1.8.0.jar
2. 在cmd中运行下面的命令

```
java -Dserver.port=8080 -Dcsp.sentinel.dashboard.server=localhost:8080 -  
Dproject.name=sentinel-dashboard -jar sentinel-dashboard-1.8.1.jar
```

- `-Dserver.port=8080` 用于指定 Sentinel 控制台端口为 8080
- `csp.sentinel.dashboard.server`，控制台的地址，指定控制台后客户端会自动向该地址发送心跳包
- `project.name`，指定应用名称

更多参数，详见[Sentinel启动参数配置表](#)

3. 访问 <http://localhost:8080>，输入 sentinel/sentinel 帐号密码即可访问。

## Sentinel 集成到Spring Cloud

Sentinel集成到Spring Cloud中，其实本质上是一样的，都是基于Sentinel-Core来完成流量控制等功能。

不管是集成到Spring Cloud Netflix、还是Dubbo、亦或者像Spring Cloud Gateway、gRPC等框架中，其原理都是一样。

就像前面我们分析的Spring Cloud Netflix集成Nacos实现服务注册的道理类似，一定都是围绕Spring生态下的扩展来实现集成，并提供友好的方式来给到开发者使用。

注意，Spring Boot版本为2.3.13.RELEASE，spring Cloud Alibaba版本为：2.2.6.RELEASE，Spring Cloud版本为：Hoxton.SR12。

## 具体实现代码

下面，我们再演示一下和Spring Cloud的集成。

修改的项目是在上一次内容中的 `gpmall-pc-alibaba`，直接修改 `gpmall-alibaba-portal` 即可。

1. 添加jar包依赖

```
<dependency>  
    <groupId>com.alibaba.cloud</groupId>  
    <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>  
</dependency>
```

2. TestService

```
@Service
public class TestService {

    @SentinelResource(value = "doTest",blockHandler = "handerException")
    public String doTest(){
        return "Hello,"+new Date();
    }

    public String handerException(BlockException e){
        return "被限流了";
    }
}
```

### 3. SentinelController

```
@RestController
public class SentinelController {
    @Autowired
    TestService testService;

    @GetMapping("/hi")
    public String doTest(){
        return testService.doTest();
    }
}
```

### 4. FlowRuleInitFunc

```
public class FlowRuleInitFunc implements InitFunc {
    @Override
    public void init() throws Exception {
        List<FlowRule> rules=new ArrayList<>();
        FlowRule rule=new FlowRule();
        rule.setResource("doTest");
        rule.setGrade(RuleConstant.FLOW_GRADE_QPS);
        rule.setCount(2);
        rules.add(rule);
        FlowRuleManager.loadRules(rules);
    }
}
```

在META-INF/services/com.alibaba.csp.sentinel.init.InitFunc文件中，添加自定义扩展点的全路径

按照上述步骤配置，即可完成Spring Cloud的整合。

细心的小伙伴可以发现，集成Spring Cloud的实现其实和Spring Boot差不多，除了Jar包依赖有差异。

但是实际上，我们限流的目标可能是OpenFeign/或者Dubbo服务，那么这个时候怎么实现呢？

在Sentinel官方文档有详细说明，文档地址如下：

主流框架适配文档：<https://github.com/alibaba/Sentinel/wiki/%E4%B8%BB%E6%B5%81%E6%A1%86%E6%9E%B6%E7%9A%84%E9%80%82%E9%85%8D#feign>

## Dubbo集成Sentinel实现限流

下面，我们来演示一下Dubbo基于Sentinel实现流量控制。

Sentinel 提供 Dubbo 的相关适配 `Sentinel Dubbo Adapter`，主要包括针对 Service Provider 和 Service Consumer 实现的 Filter。相关模块：

- `sentinel-apache-dubbo-adapter`（兼容 Apache Dubbo 2.7.x 及以上版本，自 Sentinel 1.5.1 开始支持）
- `sentinel-dubbo-adapter`（兼容 Dubbo 2.6.x 版本）

对于 Apache Dubbo 2.7.x 及以上版本，使用时需引入以下模块（以 Maven 为例）：

```
<dependency>
    <groupId>com.alibaba.csp</groupId>
    <artifactId>sentinel-apache-dubbo-adapter</artifactId>
    <version>x.y.z</version>
</dependency>
```

对于 Dubbo 2.6.x 及以下版本，使用时需引入以下模块（以 Maven 为例）：

```
<dependency>
    <groupId>com.alibaba.csp</groupId>
    <artifactId>sentinel-dubbo-adapter</artifactId>
    <version>x.y.z</version>
</dependency>
```

引入此依赖后，Dubbo 的服务接口和方法（包括调用端和服务端）就会成为 Sentinel 中的资源，在配置了规则后就可以自动享受到 Sentinel 的防护能力。

## Dubbo集成Sentinel实现限流实战

下面演示的案例，是在gpmall-alibaba-pc的基础上做的修改，主要影响的模块是 user-service

### 1. 添加jar包依赖

```
<dependency>
    <groupId>com.alibaba.csp</groupId>
    <artifactId>sentinel-apache-dubbo-adapter</artifactId>
    <version>1.8.1</version>
</dependency>
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>
    <version>1.8.1</version>
</dependency>
```

引入上述依赖后，Dubbo 的服务接口和方法（包括调用端和服务端）就会成为 Sentinel 中的资源，在配置了规则后就可以自动享受到 Sentinel 的防护能力。

### 2. 设置资源访问规则:FlowRuleInitFunc

```
public class FlowRuleInitFunc implements InitFunc {

    @Override
    public void init() throws Exception {
        List<FlowRule> rules=new ArrayList<>();
        FlowRule rule=new FlowRule();
        rule.setResource("com.gupaoedu.gpmall.user.IHelloService");
        rule.setGrade(RuleConstant.FLOW_GRADE_QPS);
        rule.setCount(2);
        rules.add(rule);
        FlowRuleManager.loadRules(rules);
    }
}
```

- 在META-INF/services/com.alibaba.csp.sentinel.init.InitFunc文件中，添加自定义扩展点的全路径
- 修改application.properties文件，增加监控上报

```
spring.cloud.sentinel.transport.dashboard=localhost:9999
```

Dubbo 的服务接口和方法（包括调用端和服务端）就会成为 Sentinel 中的资源，在配置了规则后就可以自动享受到 Sentinel 的防护能力。

限流粒度可以是服务接口和服务方法两种粒度：

- 服务接口：resourceName 为 **接口全限定名**，如 `com.alibaba.csp.sentinel.demo.dubbo.FooService`
- 服务方法：resourceName 为 **接口全限定名:方法签名**，如 `com.alibaba.csp.sentinel.demo.dubbo.FooService:sayHello(java.lang.String)`

## Dubbo限流异常处理

Dubbo目标服务被限流后，Dubbo会把异常信息抛到调用段，也就是在 `gpmall-alibaba-portal` 模块中看到如下异常。

```
java.lang.RuntimeException: SentinelBlockException: FlowException
    at com.alibaba.csp.sentinel.slots.block.BlockException.block(BlockException:0) ~
[sentinel-core-1.8.1.jar:1.8.1]
```

服务调用端可以捕获该异常，提供限流后的处理方法！

