Sentinel

Sentinel: 分布式服务的流量防卫兵

赵奕豪（宿何）@sczyh30

01 Sentinel 简介

02 From Hystrix To Sentinel

03 集群流控

04 展望未来

Sentinel

秒杀

API 调用限制

消息队列削峰填谷

商品查询

热点商品限流

服务冷启动

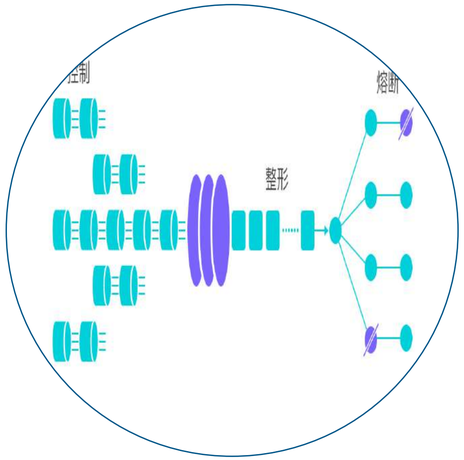
处理订单

双十一零点持续洪峰

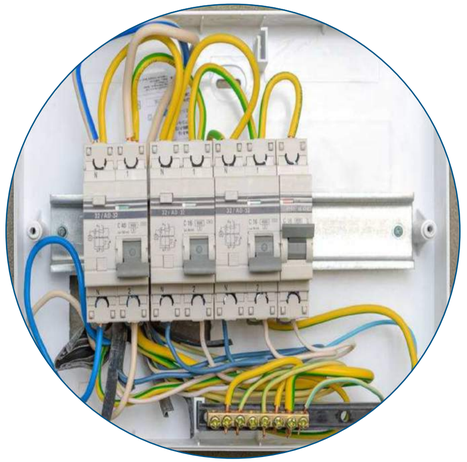
Sentinel

Sentinel 是阿里巴巴开源的，面向分布式服务架构的轻量级流量控制组件

限流



流量整形



熔断降级

系统自适应保护

Sentinel


实时监控
机器发现
规则配置
Sentinel 控制台
流量控制
线程数隔离
慢调用降级
簇点链路
速率控制
集群限流
异常熔断
系统负载保护
调用关系限流
热点限流
削峰填谷
来源访问控制
动态规则配置 (ZooKeeper, Nacos, Apollo, Redis 等)
Spring Cloud
Dubbo
gRPC
Web Servlet

01 Sentinel 简介

02 From Hystrix To Sentinel

03 集群流控

04 展望未来

Hystrix?

	Sentinel	Hystrix	resilience4j
隔离策略	信号量隔离（并发线程数限流）	线程池隔离/信号量隔离	信号量隔离
熔断降级策略	基于响应时间、异常比率、异常数	基于异常比率	基于异常比率、响应时间
实时统计实现	滑动窗口（LeapArray）	滑动窗口（基于 RxJava）	Ring Bit Buffer
动态规则配置	支持多种数据源	支持多种数据源	有限支持
扩展性	多个扩展点	插件的形式	接口的形式
基于注解的支持	支持	支持	支持
限流	基于 QPS，支持基于调用关系的限流	有限的支持	Rate Limiter
流量整形	支持预热模式、匀速器模式、预热排队模式	不支持	简单的 Rate Limiter 模式
系统自适应保护	支持	不支持	不支持
控制台	提供开箱即用的控制台，可配置规则、查看秒级监控、机器发现等	简单的监控查看	不提供控制台，可对接其它监控系统

资源模型/执行模型

Hystrix

```
public class FooServiceCommand extends HystrixCommand<String> {

    protected FooServiceCommand(HystrixCommandGroupKey group) {
        super(Setter.withGroupKey(HystrixCommandGroupKey.Factory.asKey("OtherGroup"))
            // command key
            .andCommandKey(HystrixCommandKey.Factory.asKey("fooService"))
            .andCommandPropertiesDefaults(HystrixCommandProperties.Setter()
                .withExecutionTimeoutInMilliseconds(500)
                .withCircuitBreakerRequestVolumeThreshold(5)
                .withCircuitBreakerErrorThresholdPercentage(50)
                .withCircuitBreakerSleepWindowInMilliseconds(10000)
            ));
    }

    @Override
    protected String run() throws Exception {
        return "some_result";
    }
}
```

资源定义

规则配置

被保护的逻辑

Sentinel

```
Entry entry = null;
try {
    entry = SphU.entry("resourceName");
    // 真正的业务逻辑.
    return doSomeThing();
} catch (BlockException ex) {
    // 处理限流降级异常
} finally {
    if (entry != null) {
        entry.exit();
    }
}
```

资源定义

被保护的逻辑

资源与规则分离

- RuleManager API 硬编码配置规则
- 动态规则源配置规则
- 通过控制台配置规则

隔离/熔断降级

Hystrix

```java
public class FooServiceCommand extends HystrixCommand<String> {

    protected FooServiceCommand(HystrixCommandGroupKey group) {
        super(Setter.withGroupKey(HystrixCommandGroupKey.Factory.asKey("OtherGroup"))
            // command key
            .andCommandKey(HystrixCommandKey.Factory.asKey("fooService"))
            .andCommandPropertiesDefaults(HystrixCommandProperties.Setter()
                .withExecutionTimeoutInMilliseconds(500)
                .withCircuitBreakerRequestVolumeThreshold(5)
                .withCircuitBreakerErrorThresholdPercentage(50)
                .withCircuitBreakerSleepWindowInMilliseconds(10000)
            ));
    }

    @Override
    protected String run() throws Exception {
        return "some_result";
    }
}
```

资源定义

隔离策略/
熔断规则配置

被保护的逻辑

- 支持 线程池隔离 和 信号量隔离 两种策略，主推线程池隔离
- 支持异常比率熔断（熔断器模式）

Sentinel

- 并发线程数限流（信号量隔离）
- 基于异常比率与平均响应时间（RT）的熔断降级

注解支持

Hystrix

```
// 原本的业务方法
@HystrixCommand(fallbackMethod = "fallbackForGetUser", commandProperties = {
    @HystrixProperty(name = "circuitBreaker.errorThresholdPercentage", value = "50")
})
User getUserById(String id) {
    throw new RuntimeException("getUserById command failed");
}

// fallback 方法，原方法被降级的时候调用
User fallbackForGetUser(String id) {
    return new User("admin");
}
```

- 注解定义资源并配置规则，支持配置 fallback

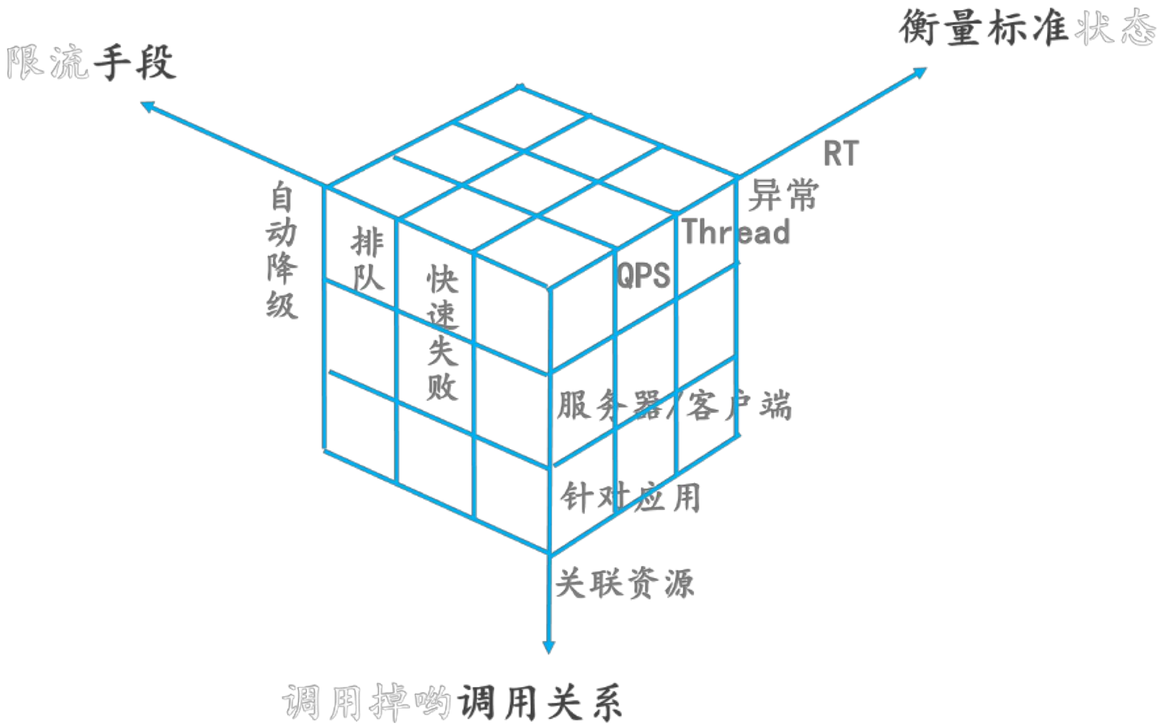
Sentinel

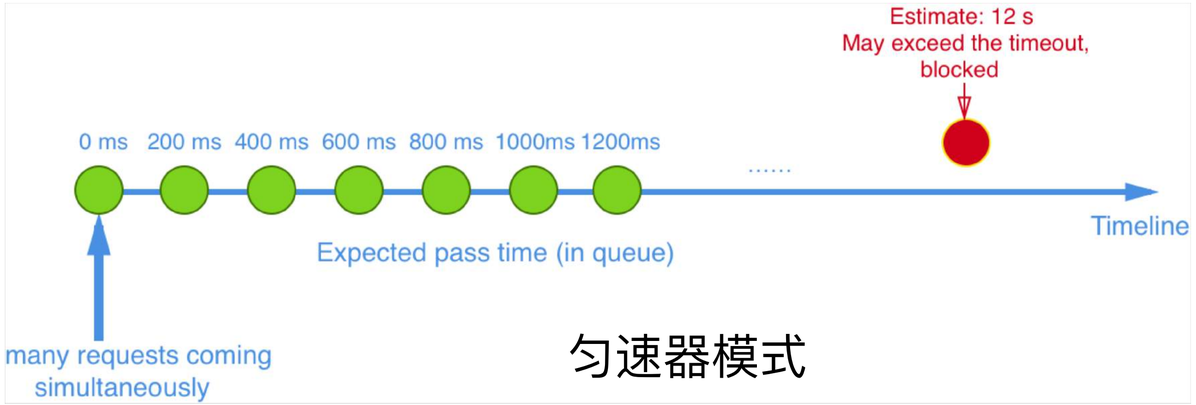
```
// 原本的业务方法.
@SentinelResource(fallback = "fallbackForGetUser")
User getUserById(String id) {
    throw new RuntimeException("getUserById command failed");
}

// fallback 方法，原方法被降级的时候调用
User fallbackForGetUser(String id) {
    return new User("admin");
}
```

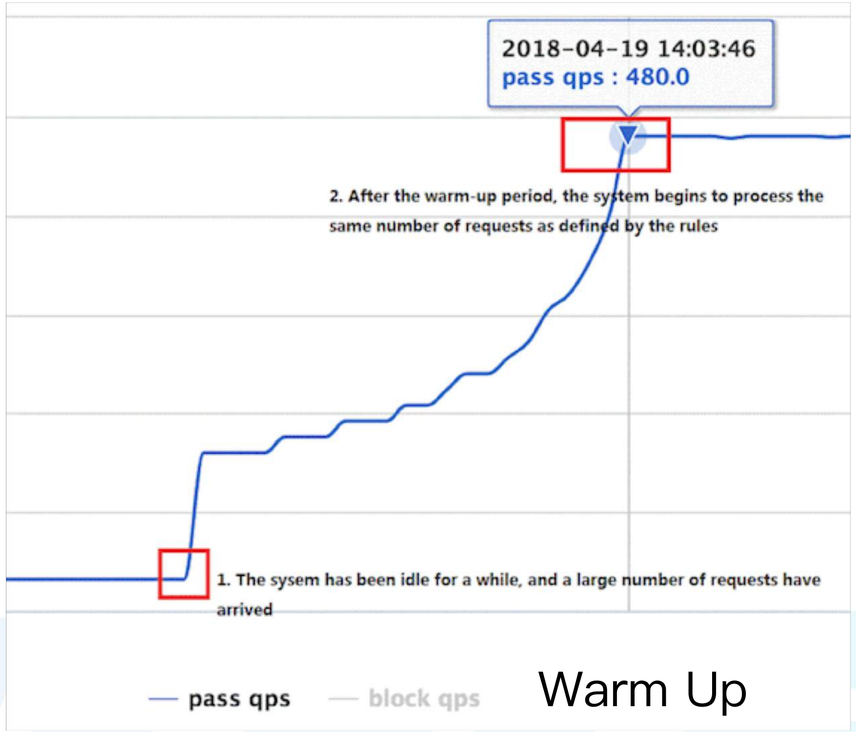
- 支持配置 blockHandler 与 fallback 函数
- 资源与规则分离，规则单独配置

专业、多样化的流量控制



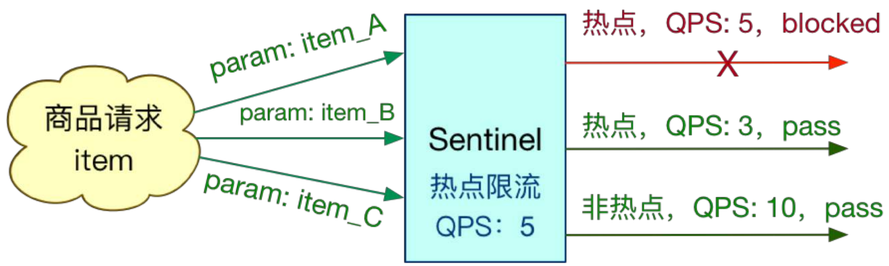


匀速器模式

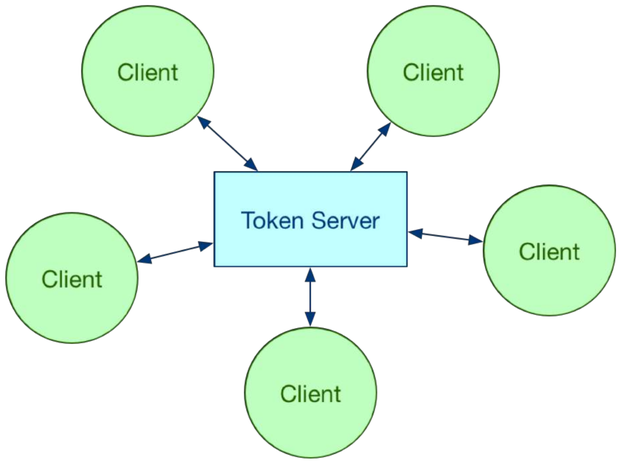


Warm Up

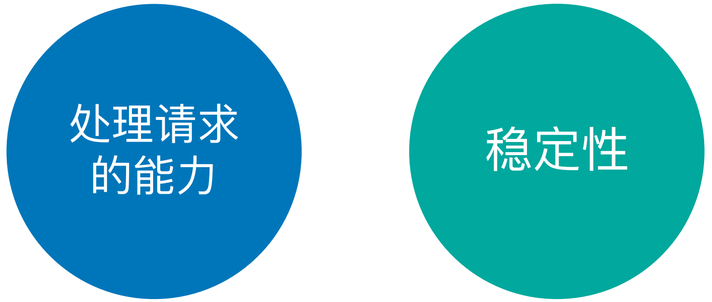
专业、多样化的流量控制

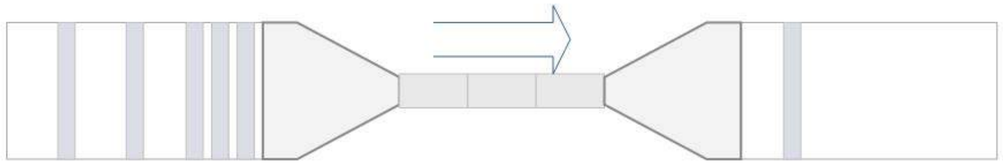


热点限流



集群限流

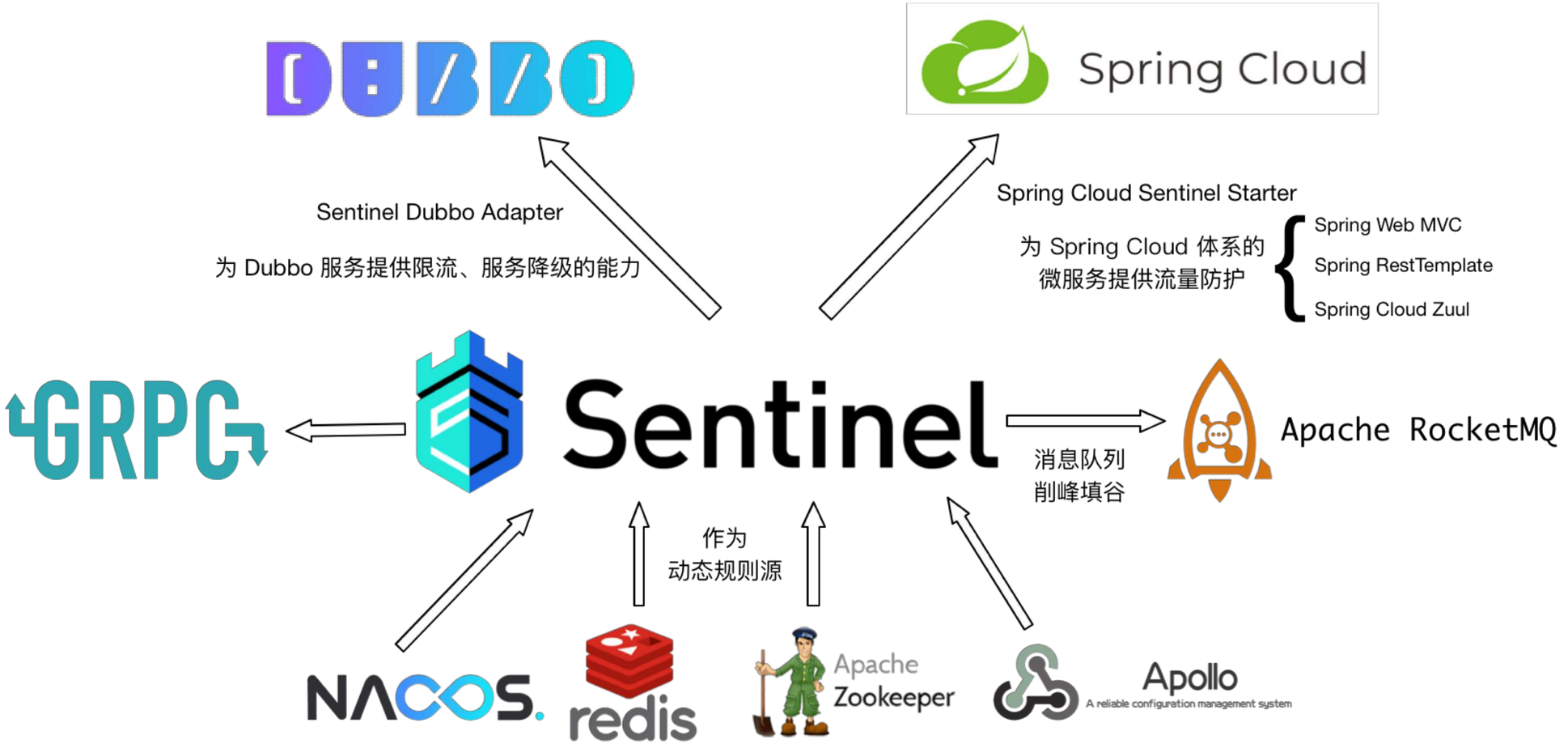




min RT * max QPS = estimated capacity
(based on TCP BBR)

系统自适应保护

开源生态


Sentinel Dubbo Adapter
为 Dubbo 服务提供限流、服务降级的能力
Spring Cloud
Spring Cloud Sentinel Starter
为 Spring Cloud 体系的
微服务提供流量防护
Spring Web MVC
Spring RestTemplate
Spring Cloud Zuul
GRPC
Sentinel
消息队列
削峰填谷
Apache RocketMQ
作为
动态规则源
NACOS.
redis
Apache
Zookeeper
Apollo
A reliable configuration management system

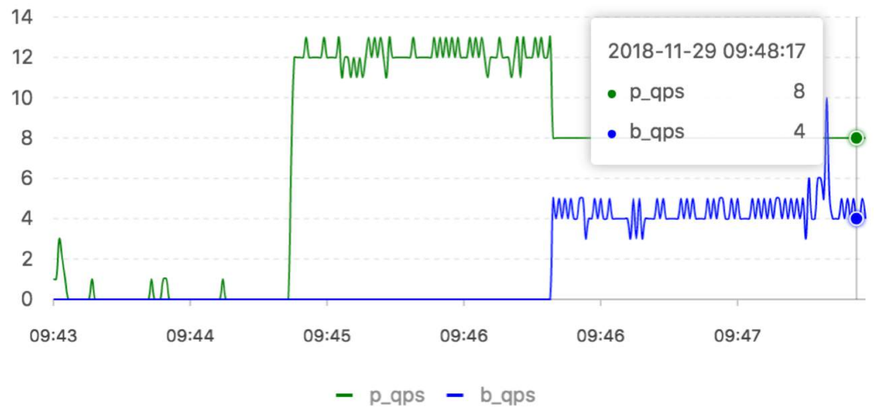
Sentinel 控制台 – 实时监控

实时监控

关键字

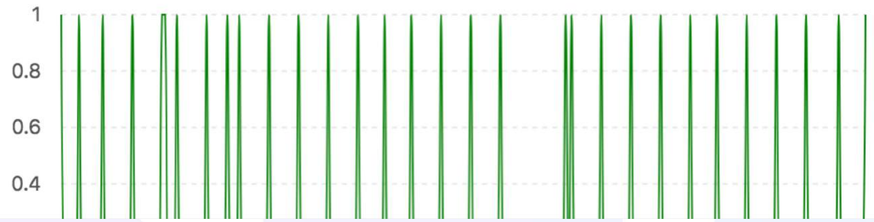
升序

/app/sentinel-dashboard/machines.json



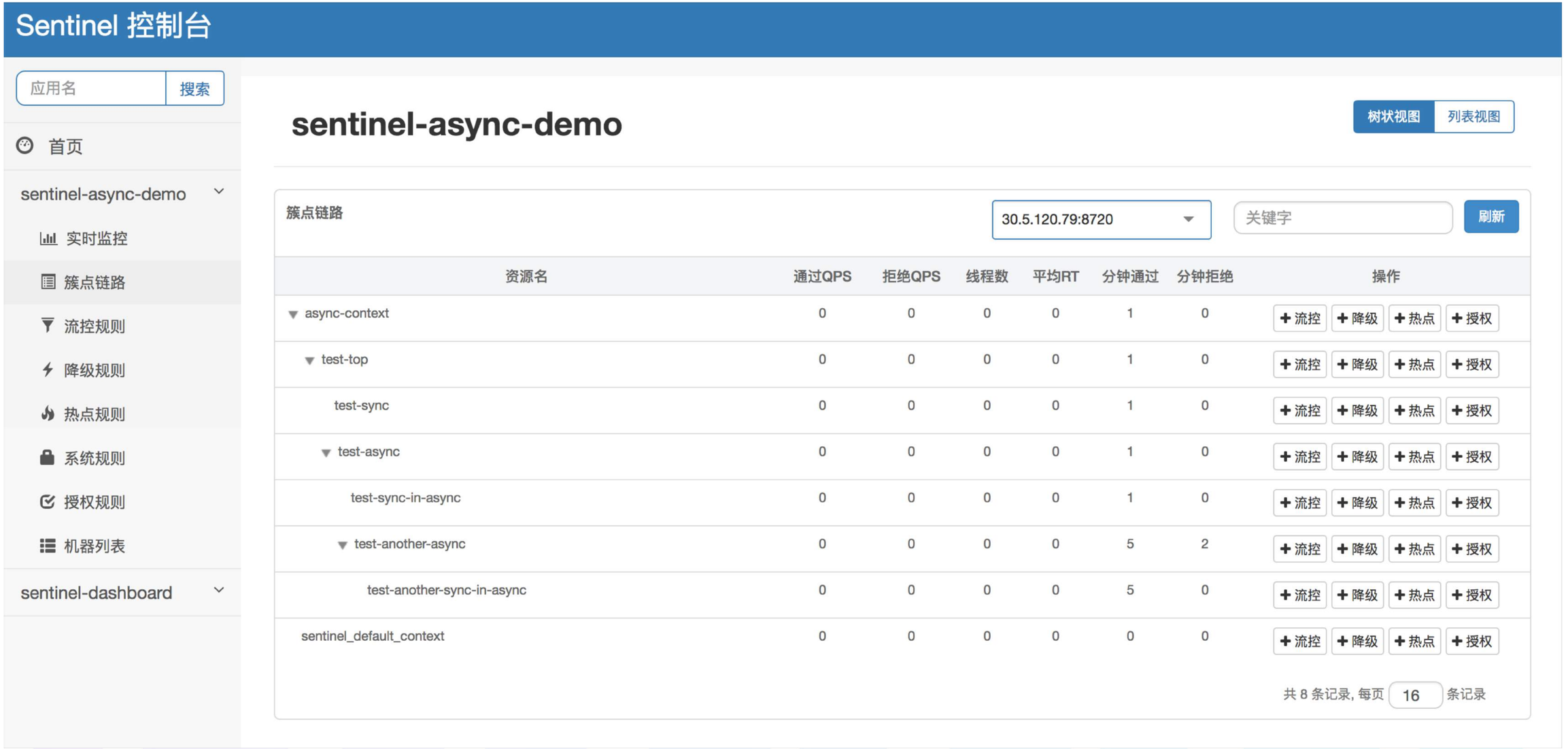
时间	p_qps	b_qps	rt(ms)
09:48:20	8.0	4.0	1.0
09:48:19	8.0	5.0	1.0
09:48:18	8.0	4.0	1.0
09:48:17	8.0	4.0	0.0
09:48:16	8.0	5.0	1.0
09:48:15	8.0	4.0	1.0

/metric/queryTopResourceMetric.json

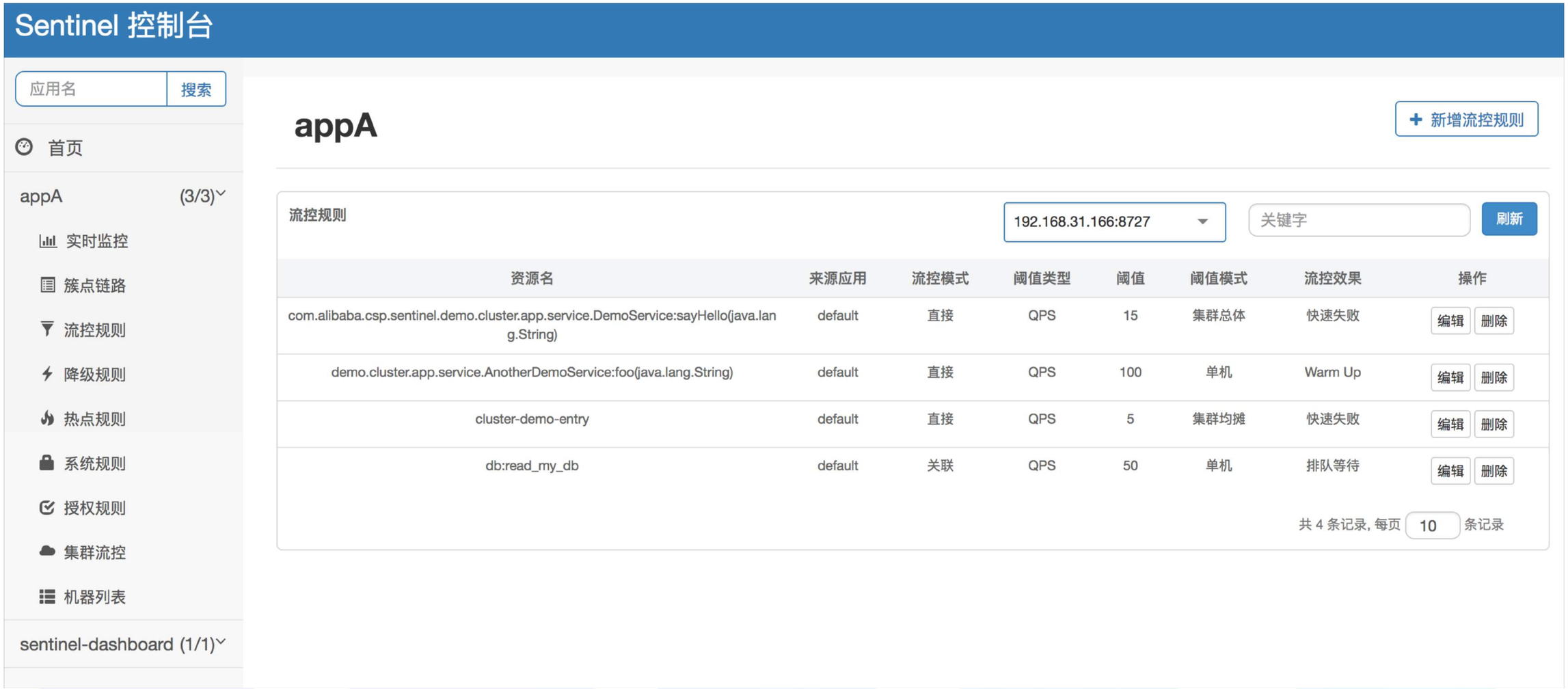


时间	p_qps	b_qps	rt(ms)
09:48:17	1.0	0.0	1.0
09:48:08	1.0	0.0	3.0
09:47:57	1.0	0.0	2.0

Sentinel 控制台 — 请求链路

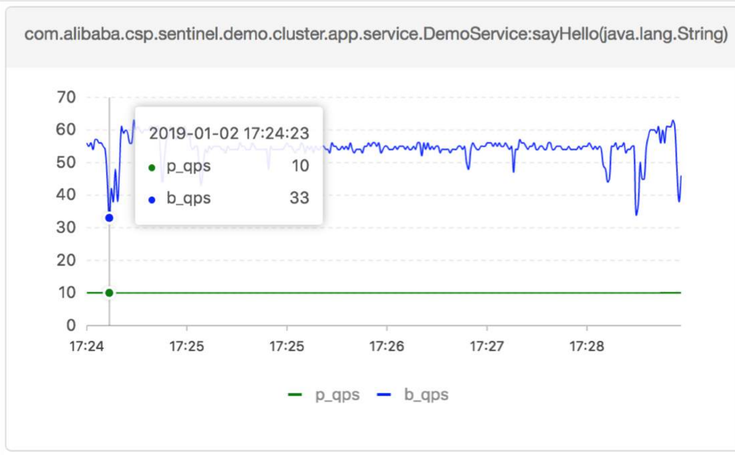


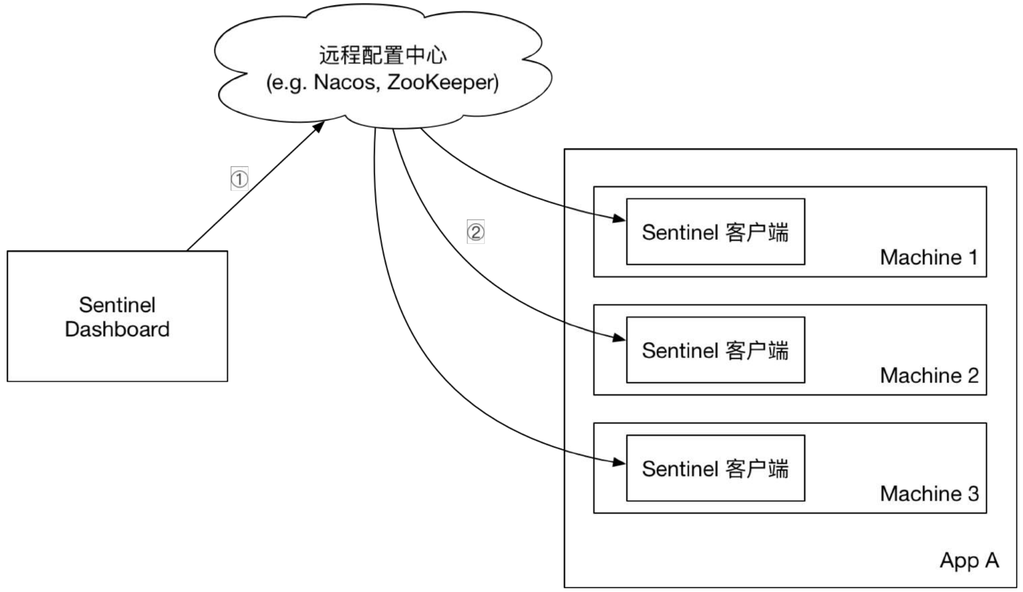
Sentinel 控制台 — 规则配置



在生产环境使用 Sentinel

- 专业的流控降级功能
- 所见即所得的监控，支持可靠、快速的实时监控和历史监控数据查询
- 规则动态管理及推送
- 鉴权，区分用户角色来进行操作





https://github.com/alibaba/Sentinel/wiki/在生产环境中使用-Sentinel

01 Sentinel 简介

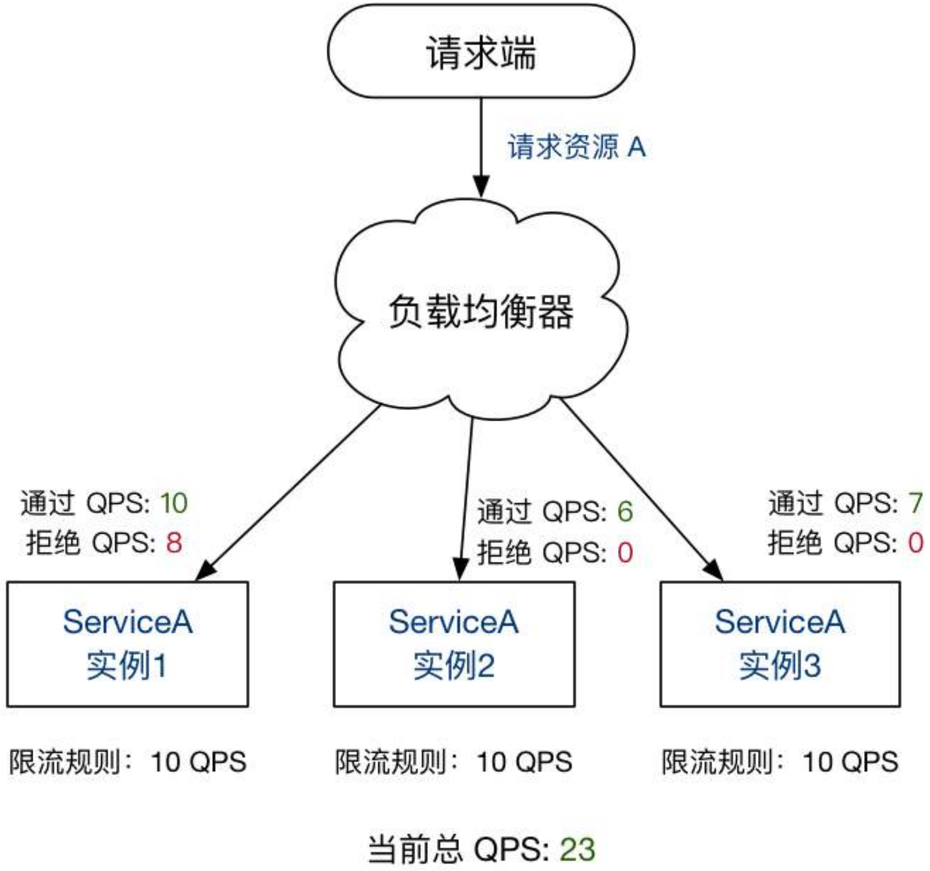
02 From Hystrix To Sentinel

03 集群流控

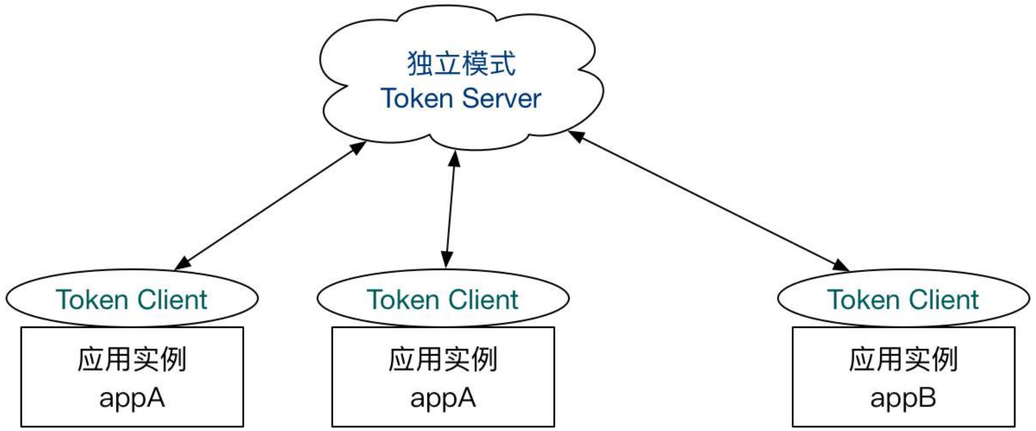
04 展望未来

为什么要使用集群限流？

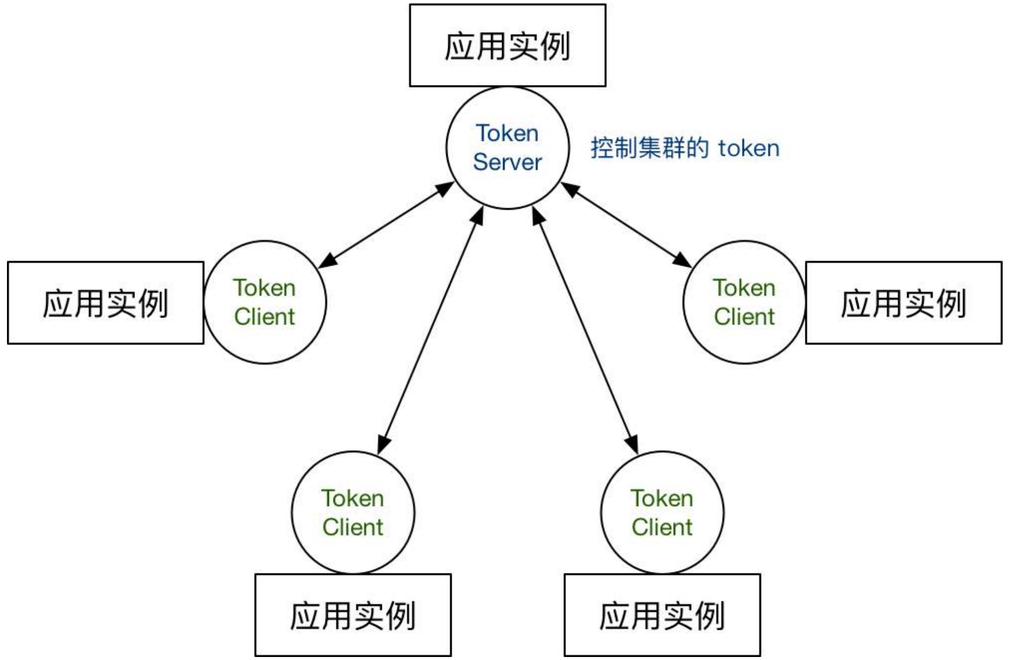
- 假设有 100 台机器，现在需要限制整个集群调用某个服务的总 QPS 不超过 50
- 在 API Gateway 处统计某个 API 的总访问量，并对某个 API 或服务的总 QPS 进行限制
- Service Mesh 中对服务间的调用进行全局流控
- 集群内对热点商品的总访问频次进行限制



部署方式

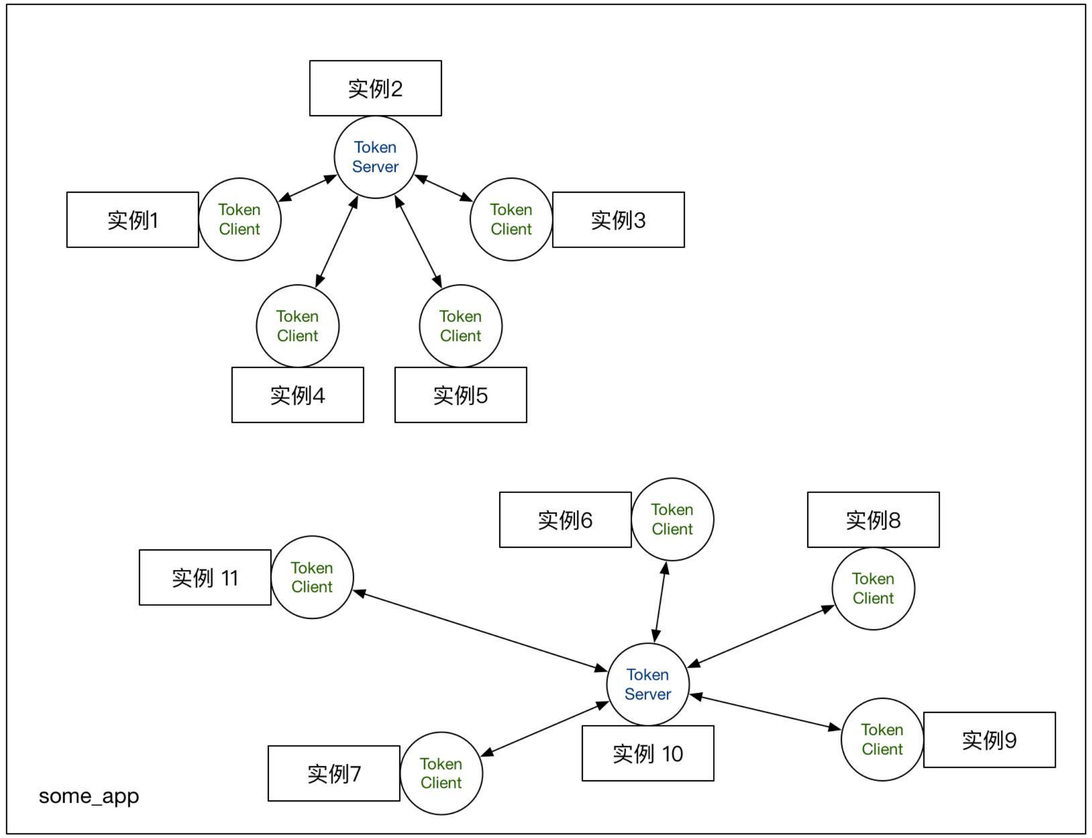


独立模式 (Alone)

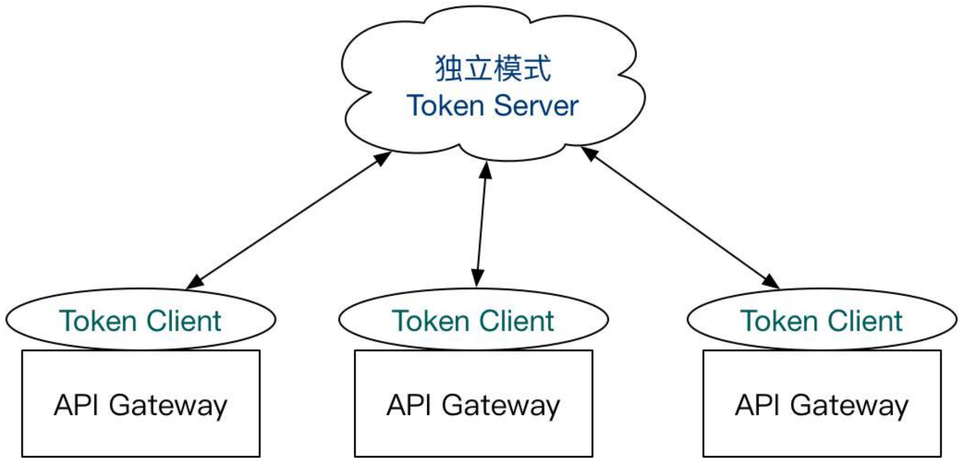


嵌入模式 (Embedded)

Token Server / Client 分配

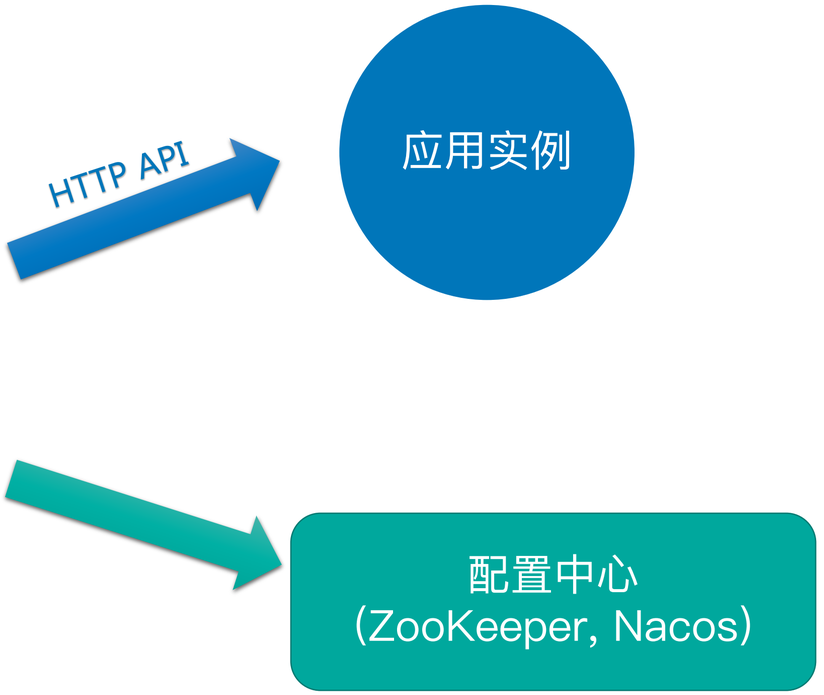


多 token server
单机均摊阈值模式



单台 token server
总体阈值模式 、单机均摊阈值模式

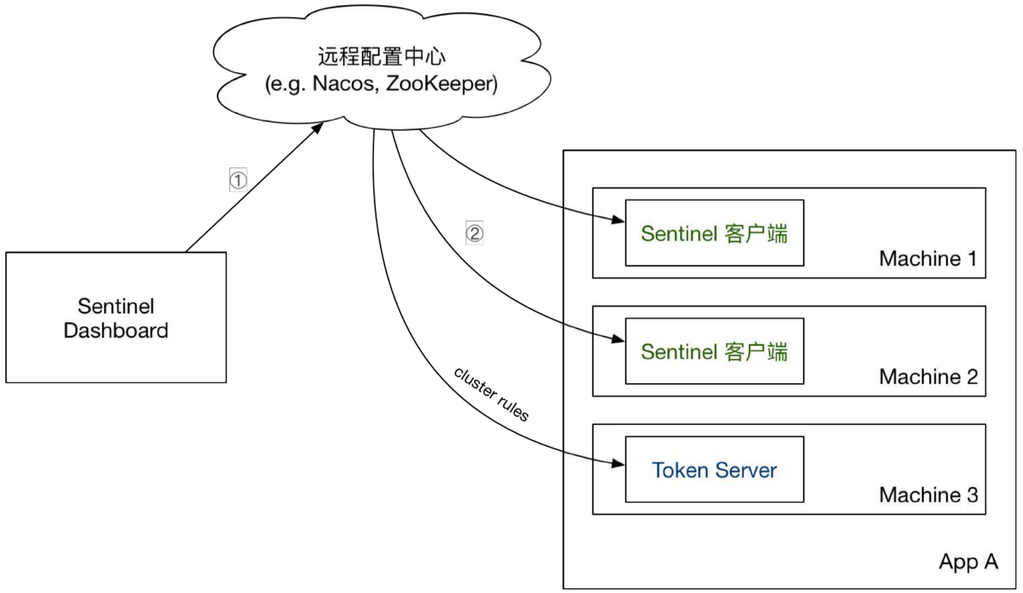
Token Server / Client 分配



[{“ip”:“112.12.88.68”,“machineId”:“112.12.88.68@8728”,“port”:11111,
{“clientSet”:[“112.12.88.66@8729”,“112.12.88.67@8727”]}]

两种推送方式

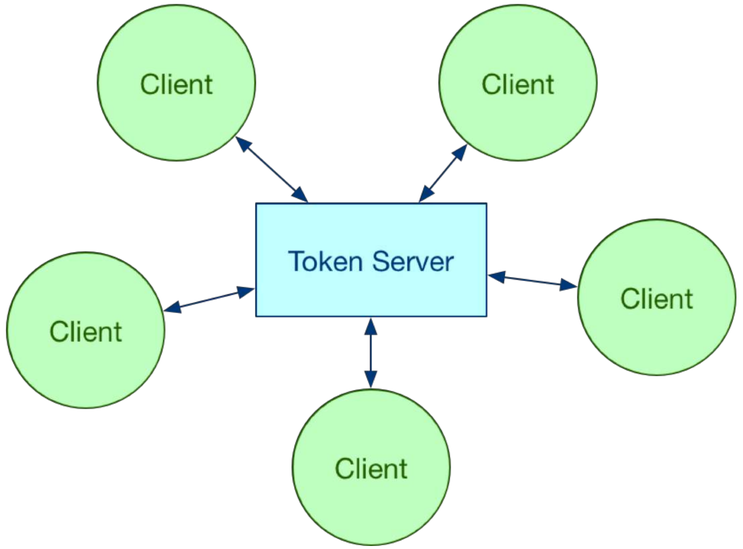
集群规则配置



- Client 规则源：FlowRuleManager
- Token Server 规则源：ClusterFlowRuleManager

集群限流管控

还需要注意哪些问题？



- Token Server 自动管理（自动分配/选举 Token Server）
- Token Server 高可用，在某个 server 不可用时自动 failover 到其它机器
- 嵌入模式下 Token Server 的稳定性

01 Sentinel 简介

02 From Hystrix To Sentinel

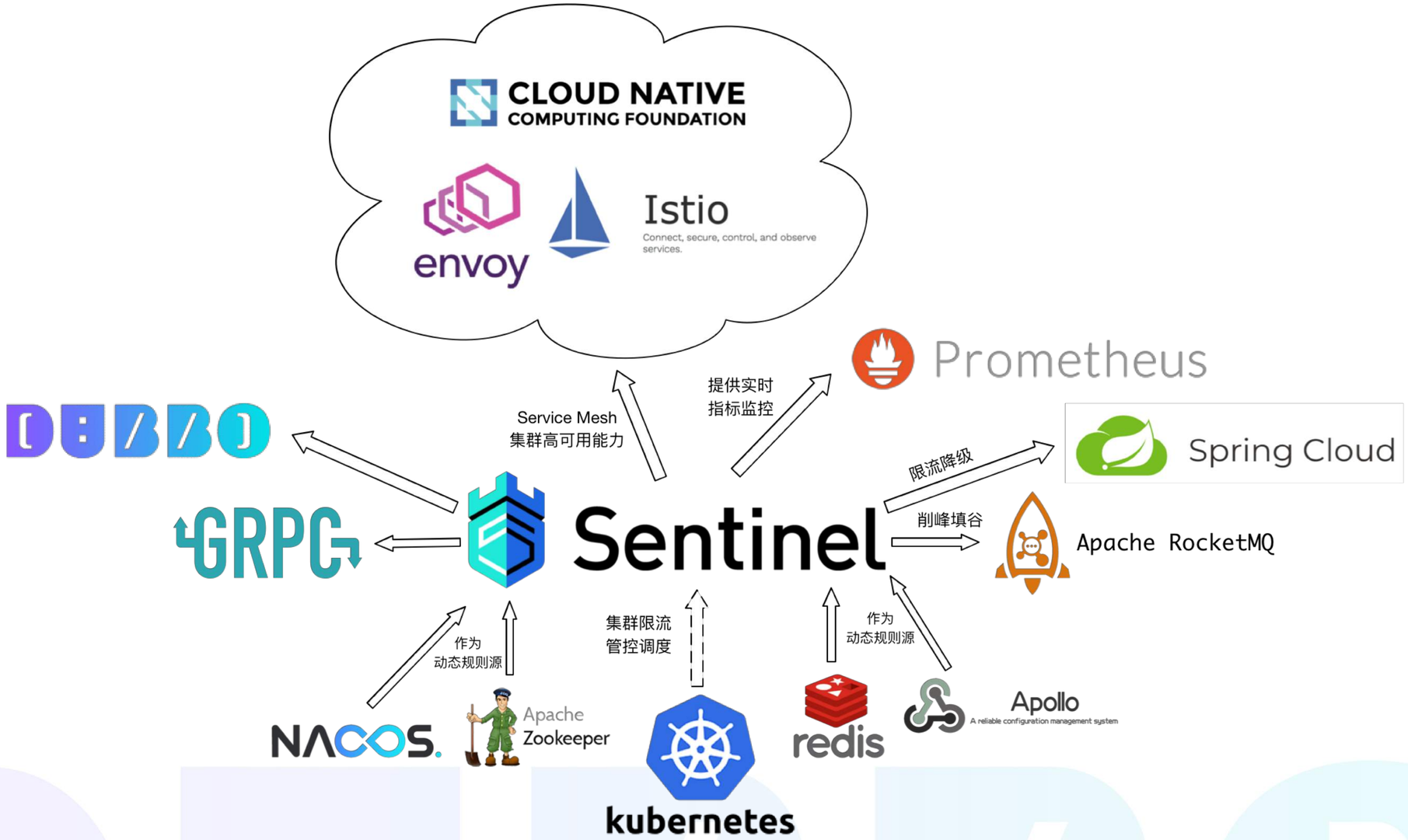
03 集群流控

04 展望未来

展望未来

- Reactive 支持 (RxJava, Reactor)
- 集群限流多语言支持、Service Mesh 支持
- 对接更多的监控系统
- 智能化（自适应流控）

未来的 Landscape



From Zero to Contributor

文档改进
与修正

分享实践经
验与文章

Report bug
Bug fix

官网维护

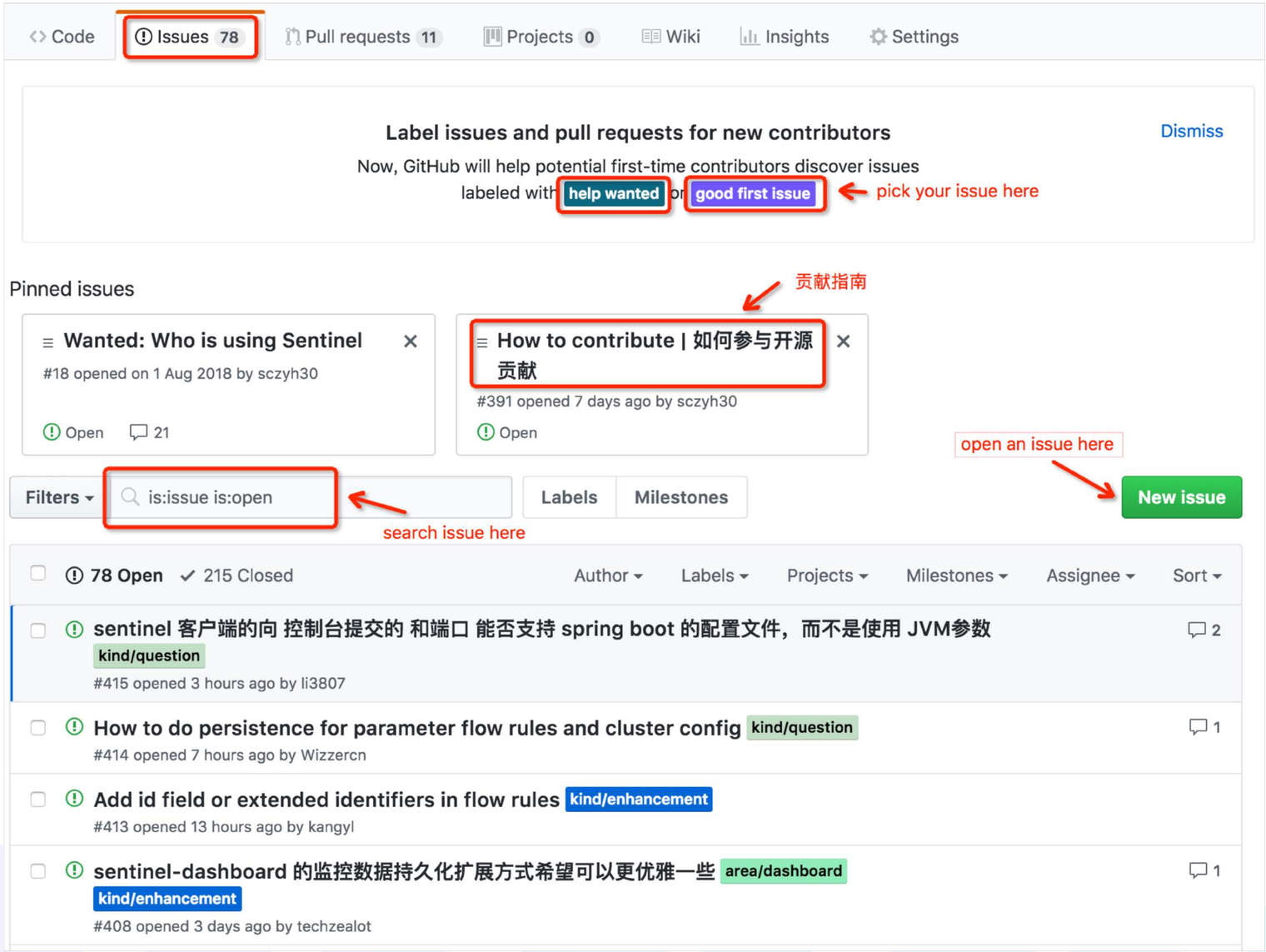
Sentinel
控制台改进

开源框架
的整合

测试用例
补充

Feature
讨论/贡献

From Zero to Contributor



Contact Us

GitHub: https://github.com/alibaba/Sentinel

企业用户欢迎留言： https://github.com/alibaba/Sentinel/issues/18

Thank you !