



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



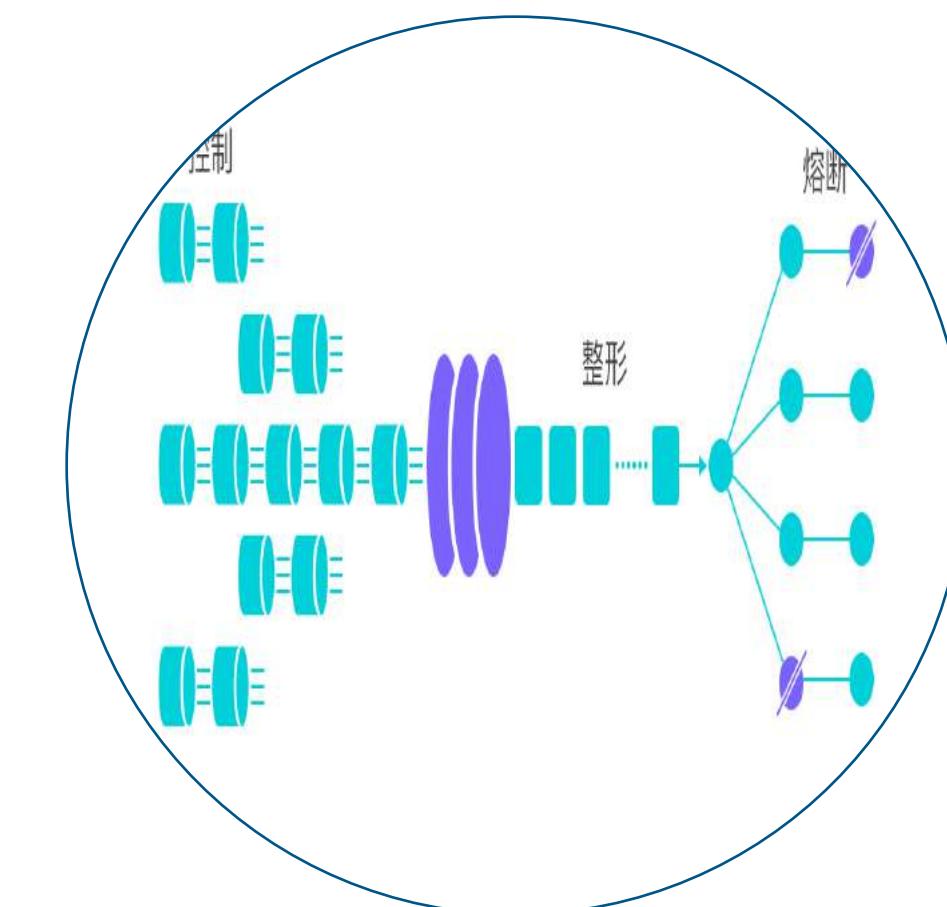


Sentinel

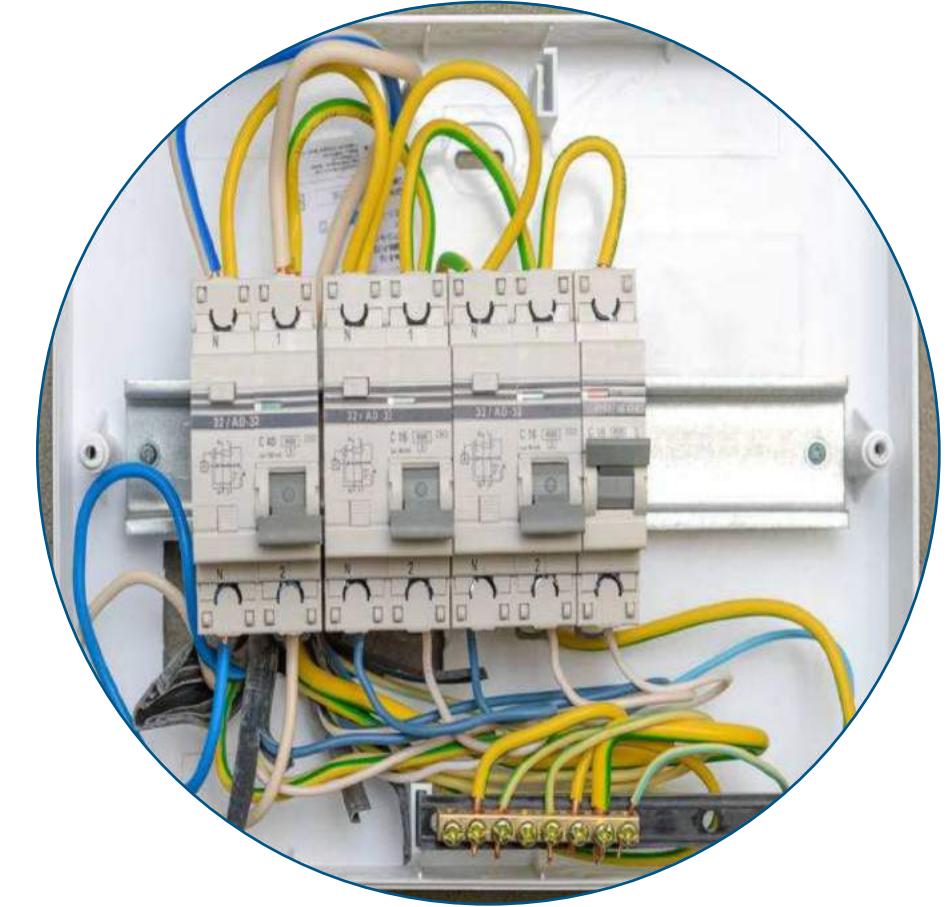
Sentinel 是阿里巴巴开源的，面向分布式服务架构的轻量级流量控制组件



限流



流量整形



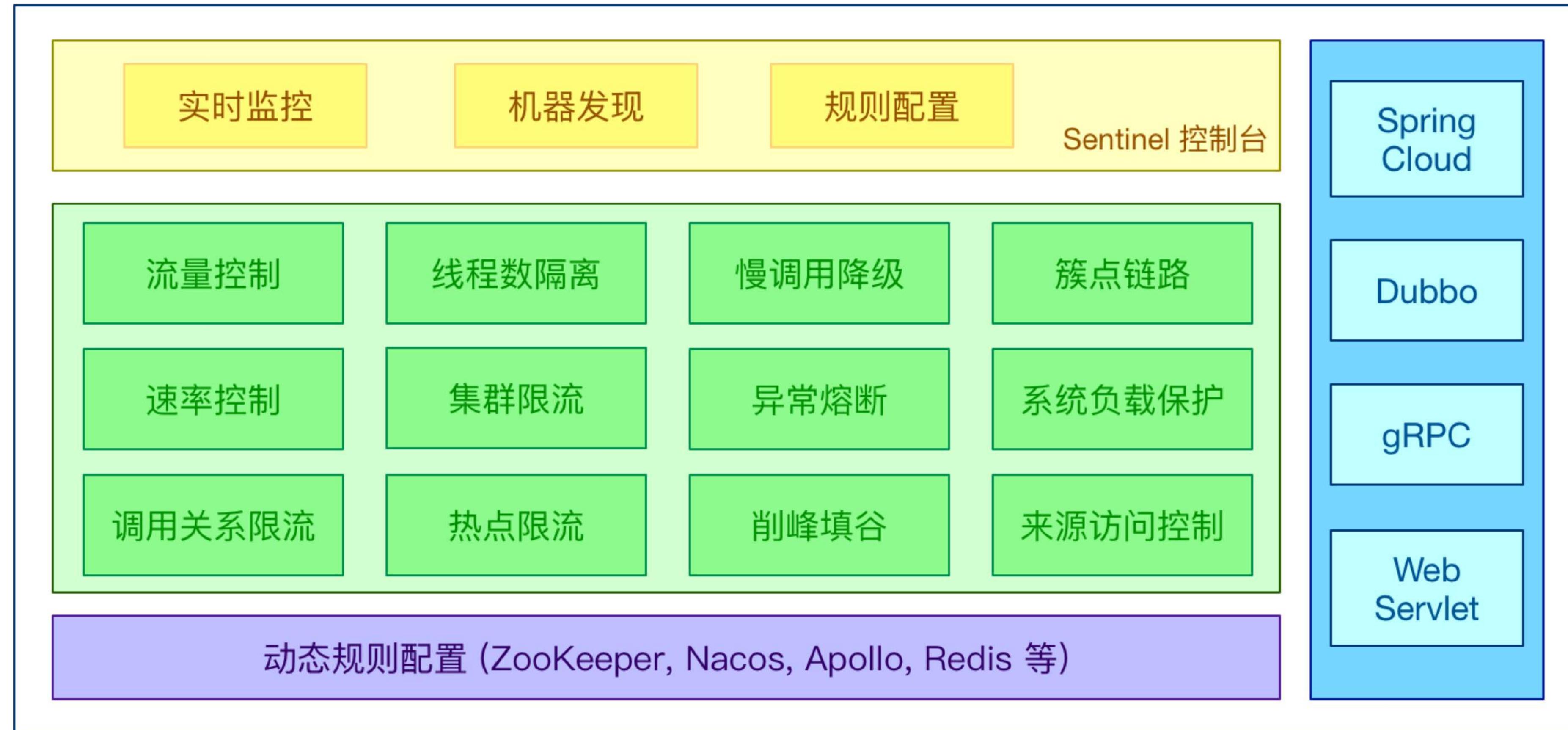
熔断降级



系统自适应保护



Sentinel



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

Sentinel

Hystrix

```
public class FooServiceCommand extends HystrixCommand<String> {  
    protected FooServiceCommand(HystrixCommandGroupKey group) {  
        super(Setter.withGroupKey(HystrixCommandGroupKey.Factory.asKey("OtherGroup"))  
            .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

```
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

```
public class FooServiceCommand extends HystrixCommand<String> {  
  
    protected FooServiceCommand(HystrixCommandGroupKey group) {  
        super(Setter.withGroupKey(HystrixCommandGroupKey.Factory.asKey("OtherGroup"))  
            .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

Sentinel



- 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）

- 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 = "fallbackFor GetUser", commandProperties = {
    @HystrixProperty(name = "circuitBreaker.errorThresholdPercentage", value = "50")
})
User getUserId(String id) {
    throw new RuntimeException("getUserById command failed");
}

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

Sentinel

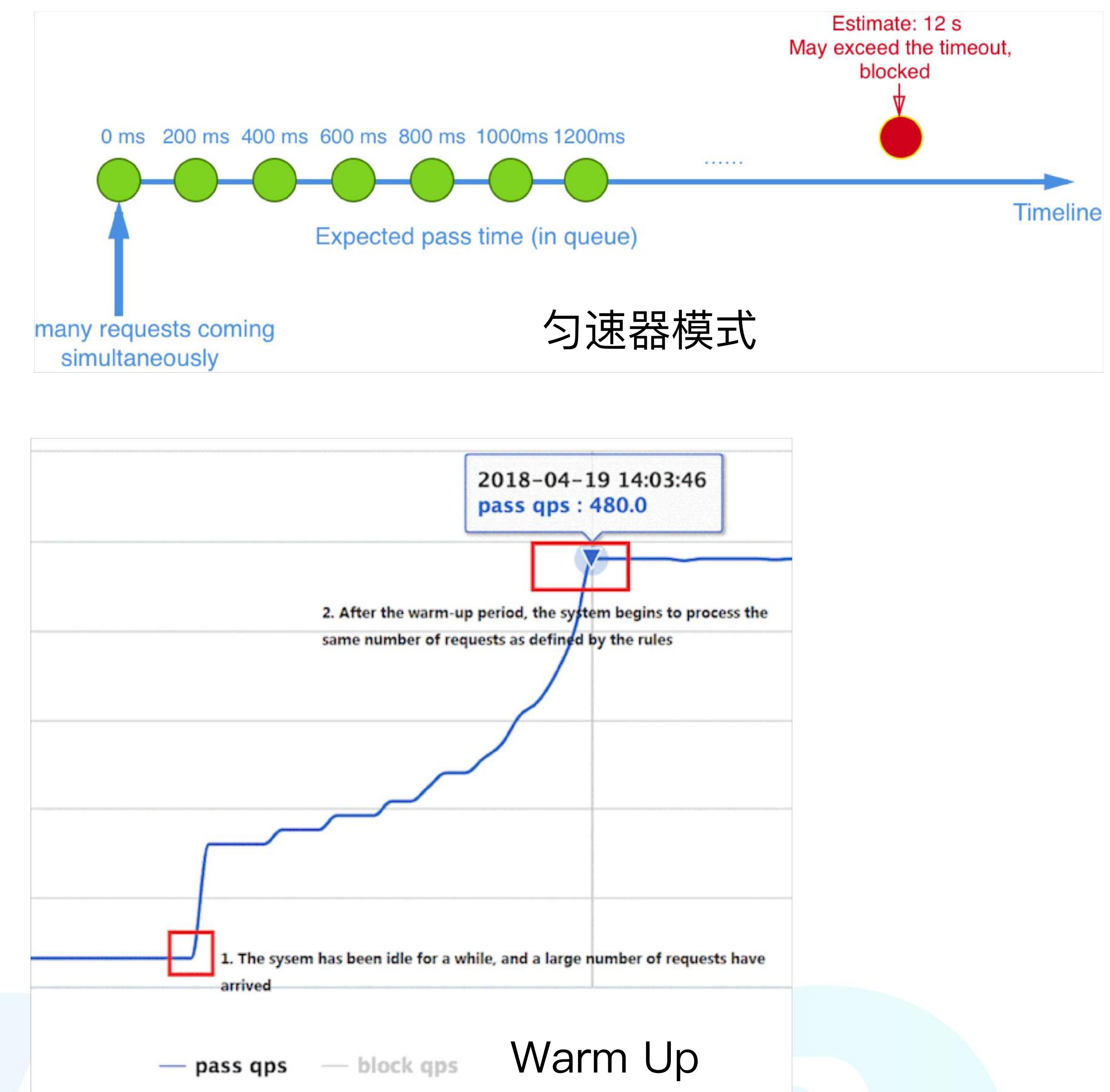
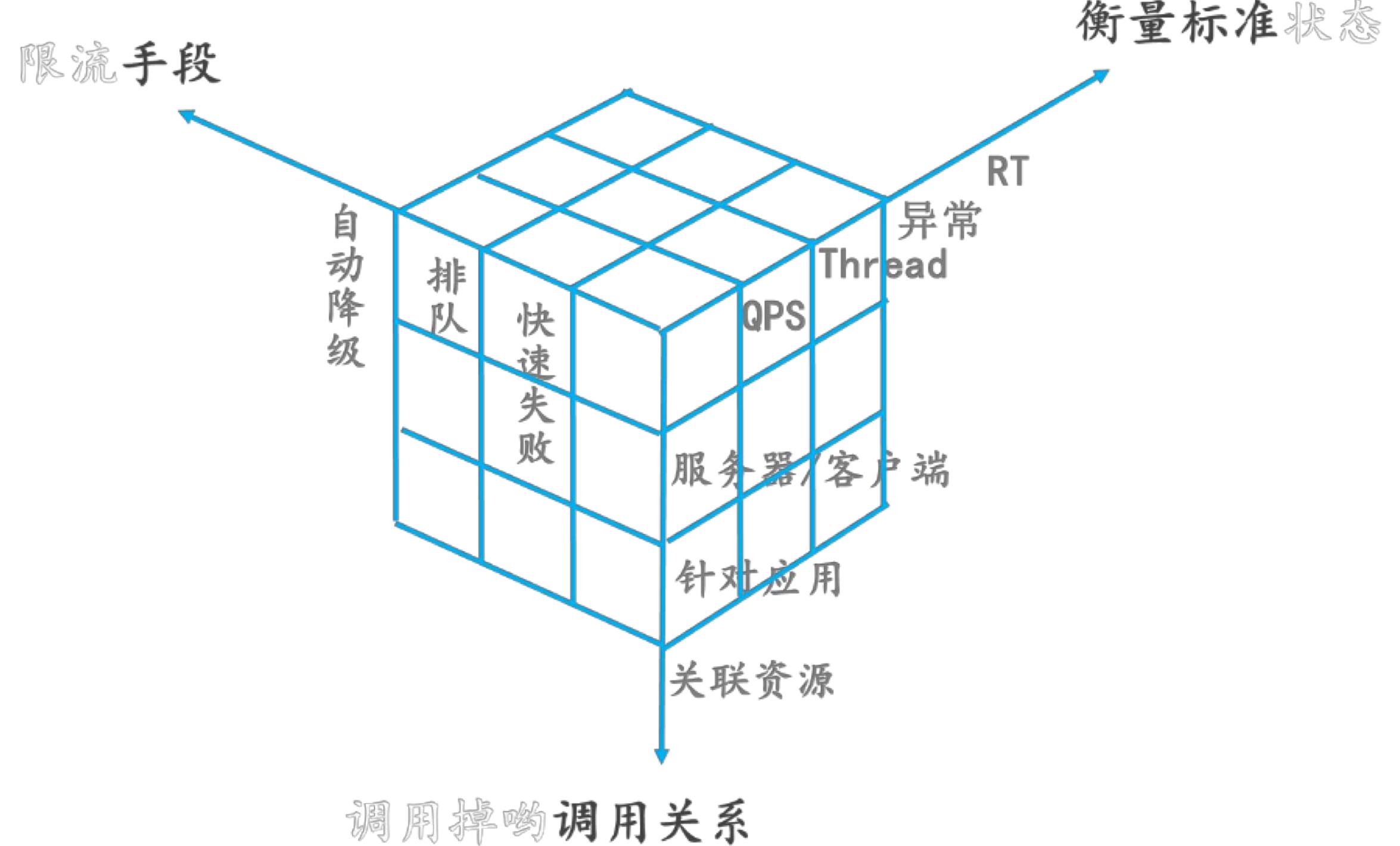
```
// 原本的业务方法.
@SentinelResource(fallback = "fallbackFor GetUser")
User getUserId(String id) {
    throw new RuntimeException("getUserById command failed");
}

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

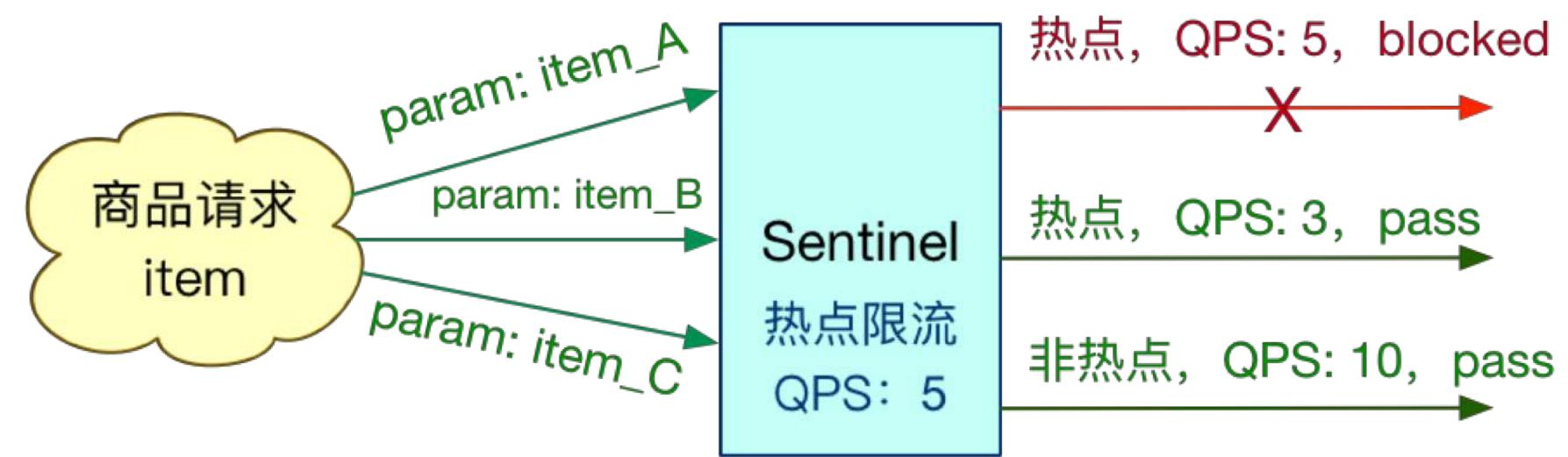
- 注解定义资源并配置规则，支持配置 fallback

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

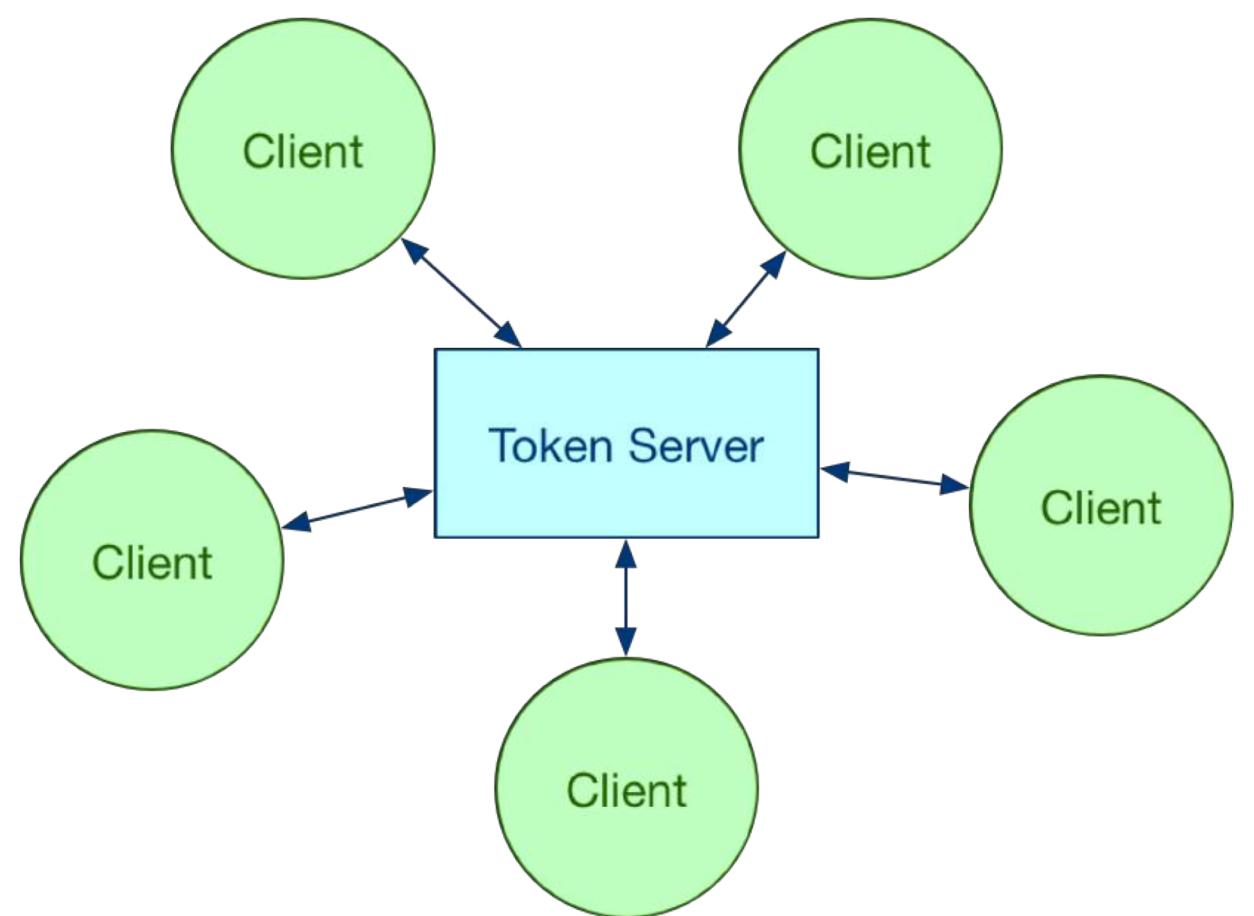
专业、多样化的流量控制



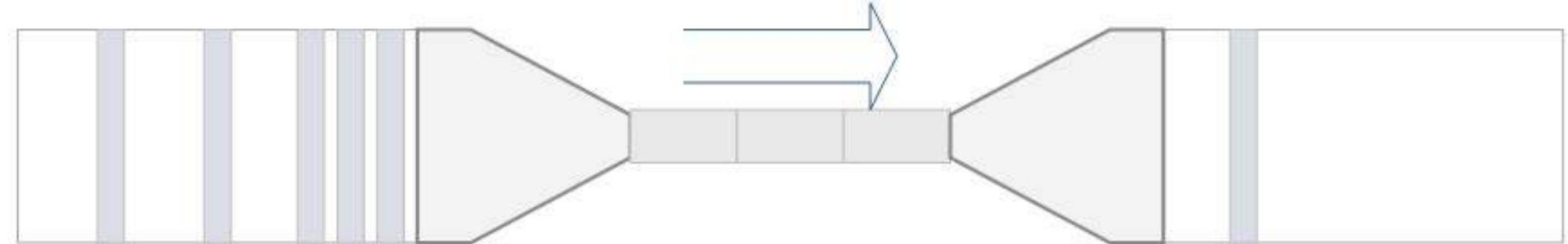
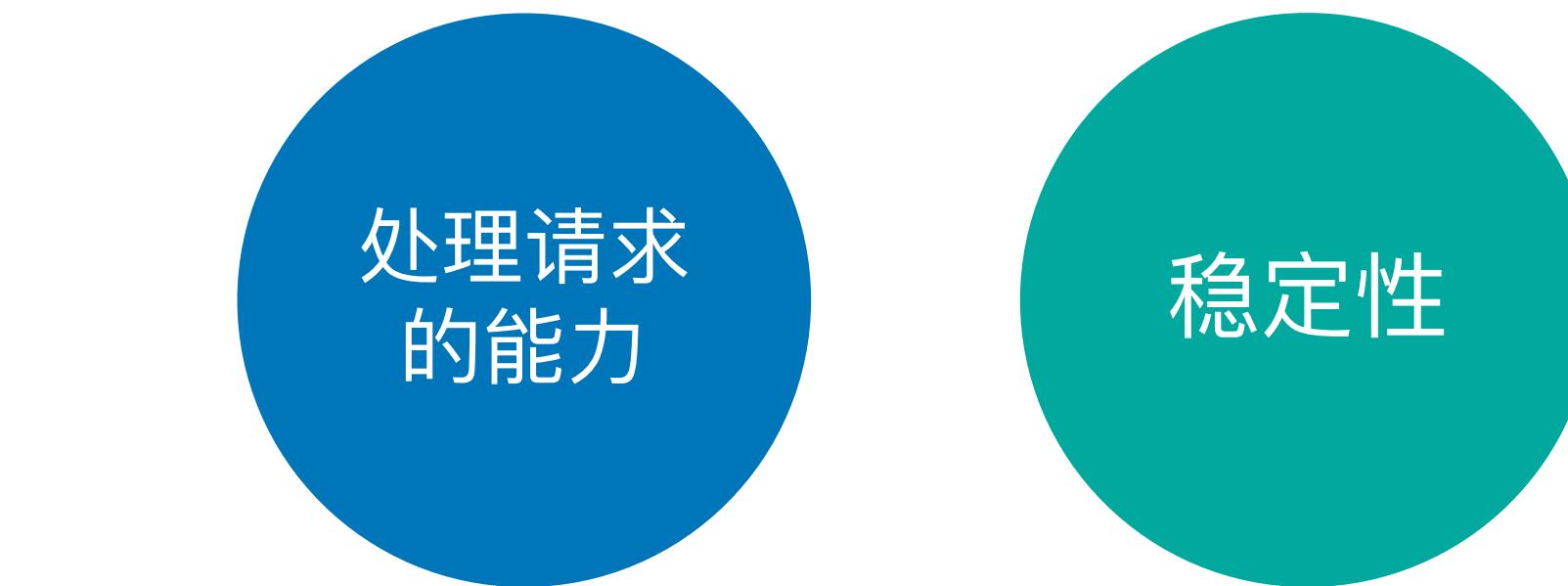
专业、多样化的流量控制



热点限流



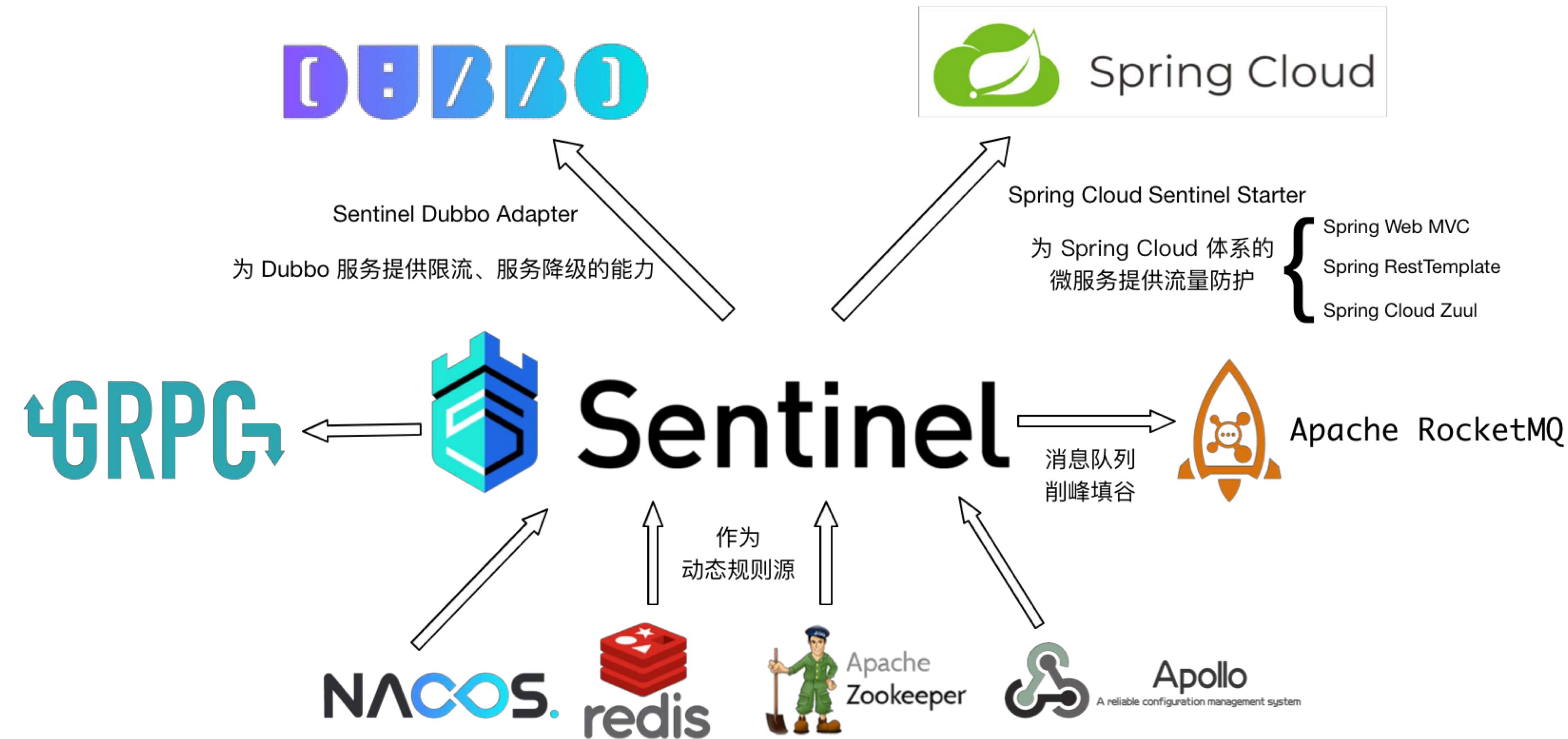
集群限流



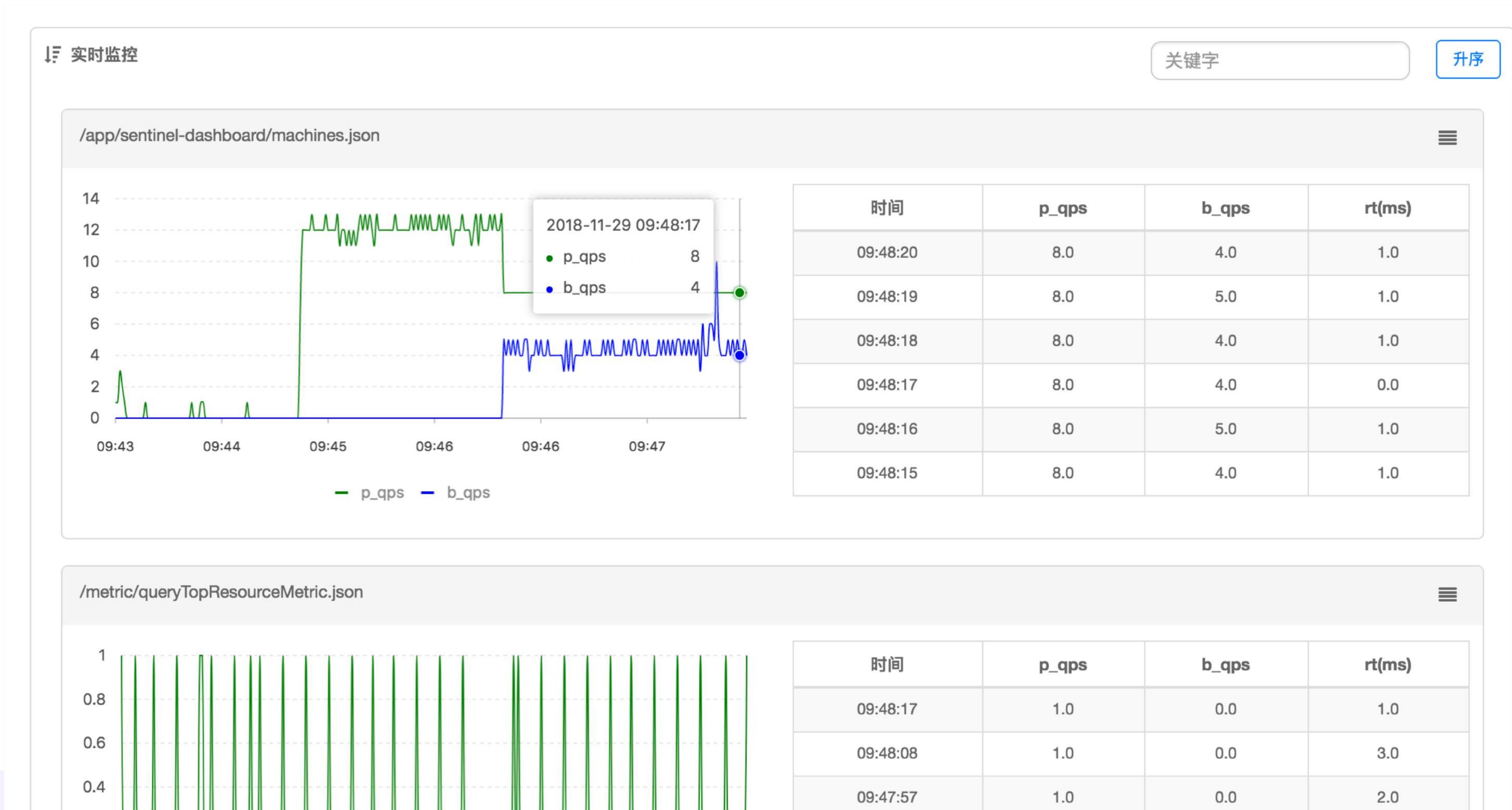
$\min RT * \max QPS = \text{estimated capacity}$
(based on TCP BBR)

系统自适应保护

开源生态



Sentinel 控制台 – 实时监控



Sentinel 控制台 — 请求链路

Sentinel 控制台

应用名

⌚ 首页

sentinel-async-demo

- 📊 实时监控
- 簇点链路**
- ▼ 流控规则
- ⚡ 降级规则
- 🔥 热点规则
- 🔒 系统规则
- ☑ 授权规则
- 机器列表

sentinel-dashboard

sentinel-async-demo

簇点链路 关键字

资源名	通过QPS	拒绝QPS	线程数	平均RT	分钟通过	分钟拒绝	操作
async-context	0	0	0	0	1	0	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>
test-top	0	0	0	0	1	0	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>
test-sync	0	0	0	0	1	0	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>
test-async	0	0	0	0	1	0	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>
test-sync-in-async	0	0	0	0	1	0	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>
test-another-async	0	0	0	0	5	2	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>
test-another-sync-in-async	0	0	0	0	5	0	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>
sentinel_default_context	0	0	0	0	0	0	<input type="button" value="+ 流控"/> <input type="button" value="+ 降级"/> <input type="button" value="+ 热点"/> <input type="button" value="+ 授权"/>

共 8 条记录, 每页 16 条记录

Sentinel 控制台 – 规则配置

Sentinel 控制台

应用名 搜索

首页

appA (3/3)
实时监控
簇点链路
流控规则
降级规则
热点规则
系统规则
授权规则
集群流控
机器列表

sentinel-dashboard (1/1)

appA

+ 新增流控规则

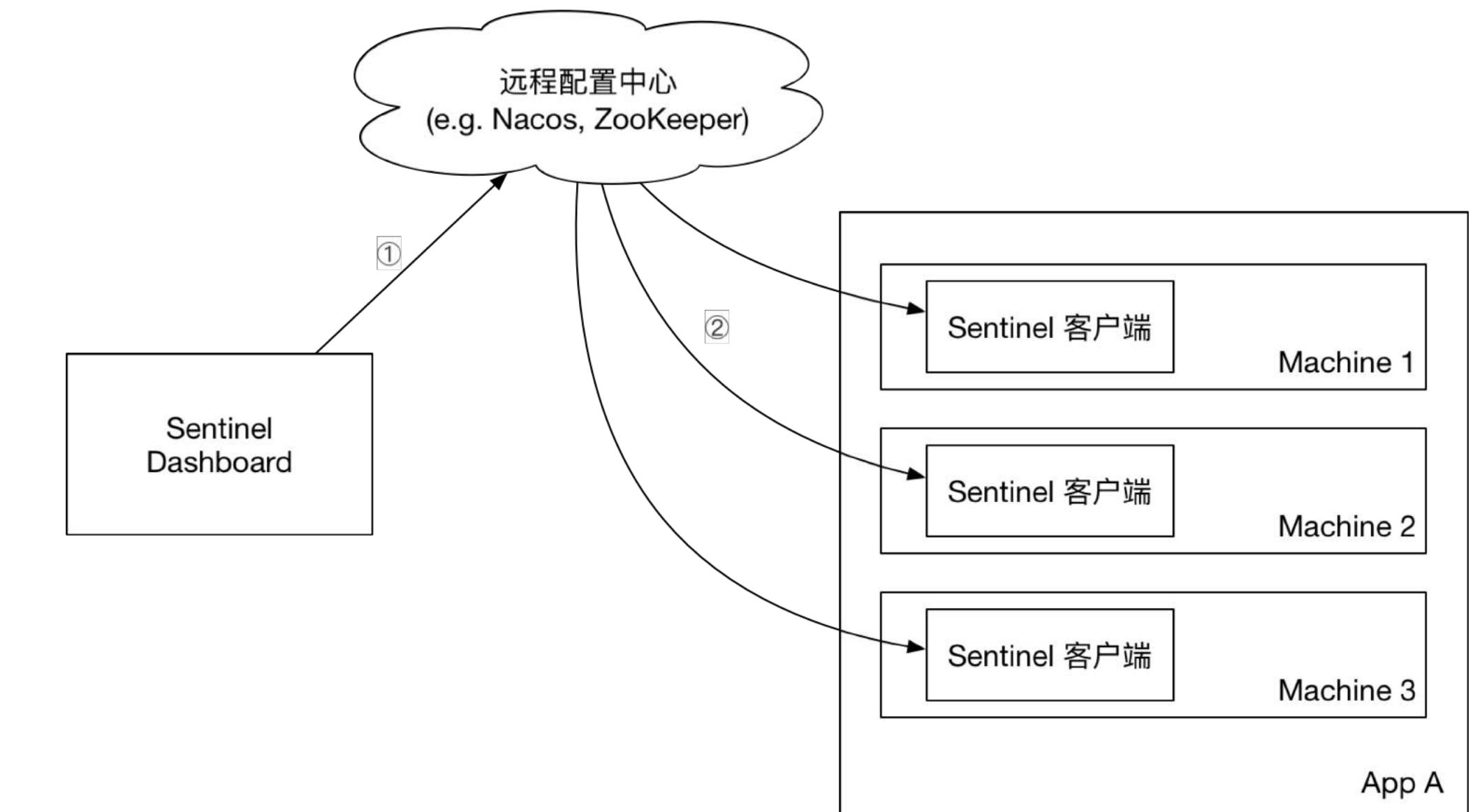
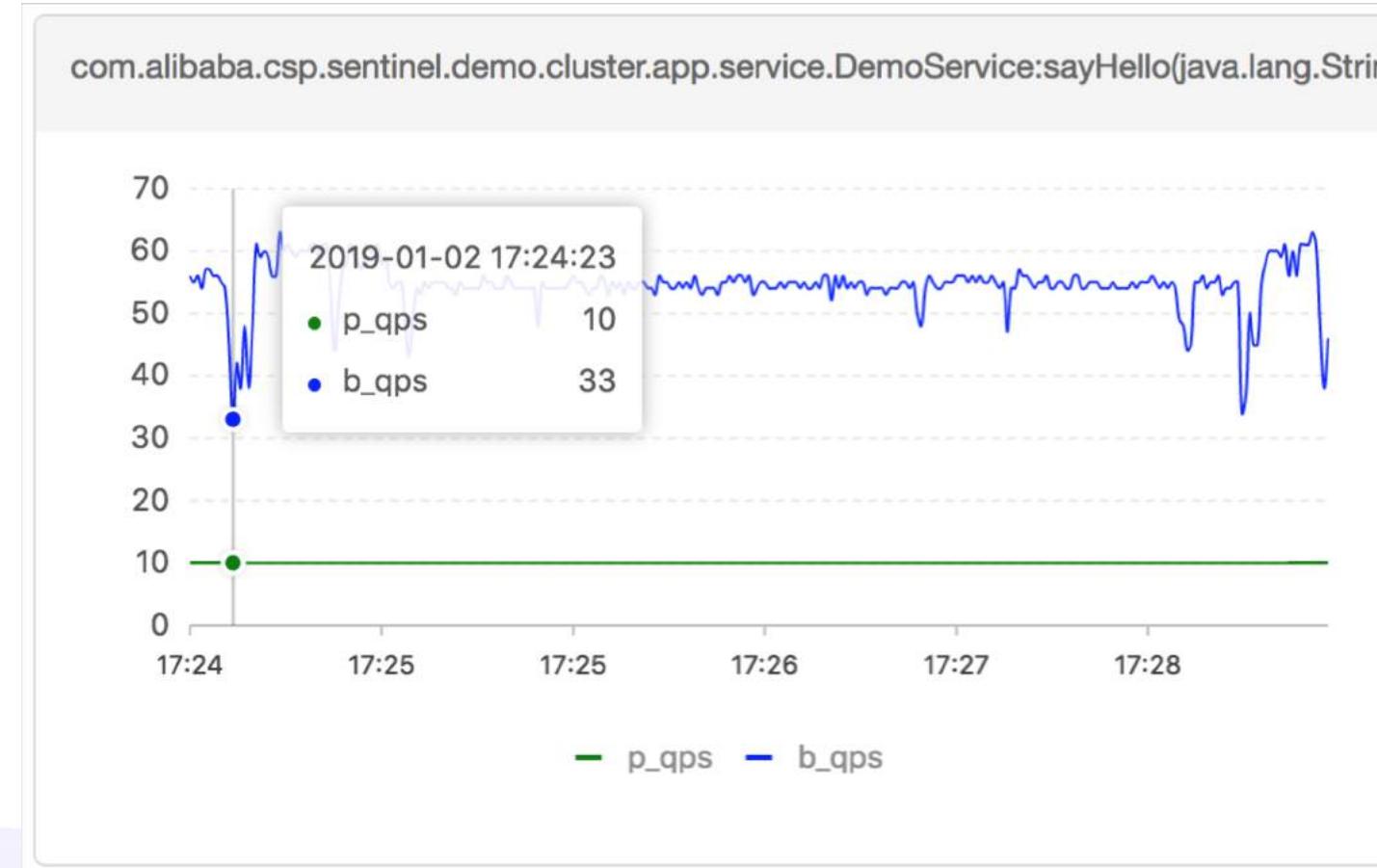
流控规则

资源名	来源应用	流控模式	阈值类型	阈值	阈值模式	流控效果	操作
com.alibaba.csp.sentinel.demo.cluster.app.service.DemoService:sayHello(java.lang.String)	default	直接	QPS	15	集群总体	快速失败	<button>编辑</button> <button>删除</button>
demo.cluster.app.service.AnotherDemoService:foo(java.lang.String)	default	直接	QPS	100	单机	Warm Up	<button>编辑</button> <button>删除</button>
cluster-demo-entry	default	直接	QPS	5	集群均摊	快速失败	<button>编辑</button> <button>删除</button>
db:read_my_db	default	关联	QPS	50	单机	排队等待	<button>编辑</button> <button>删除</button>

共 4 条记录, 每页 10 条记录

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

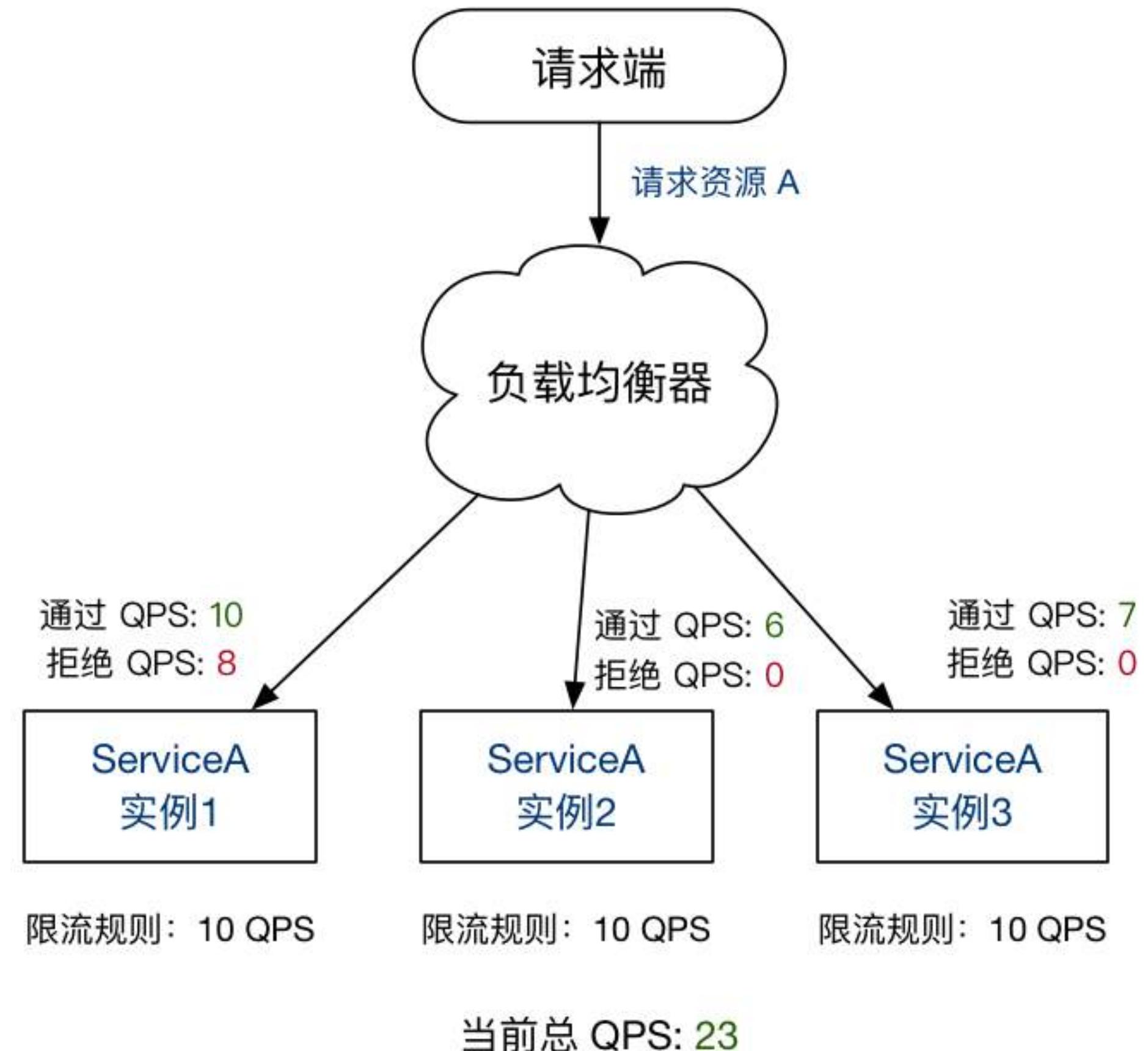
集群流控

04

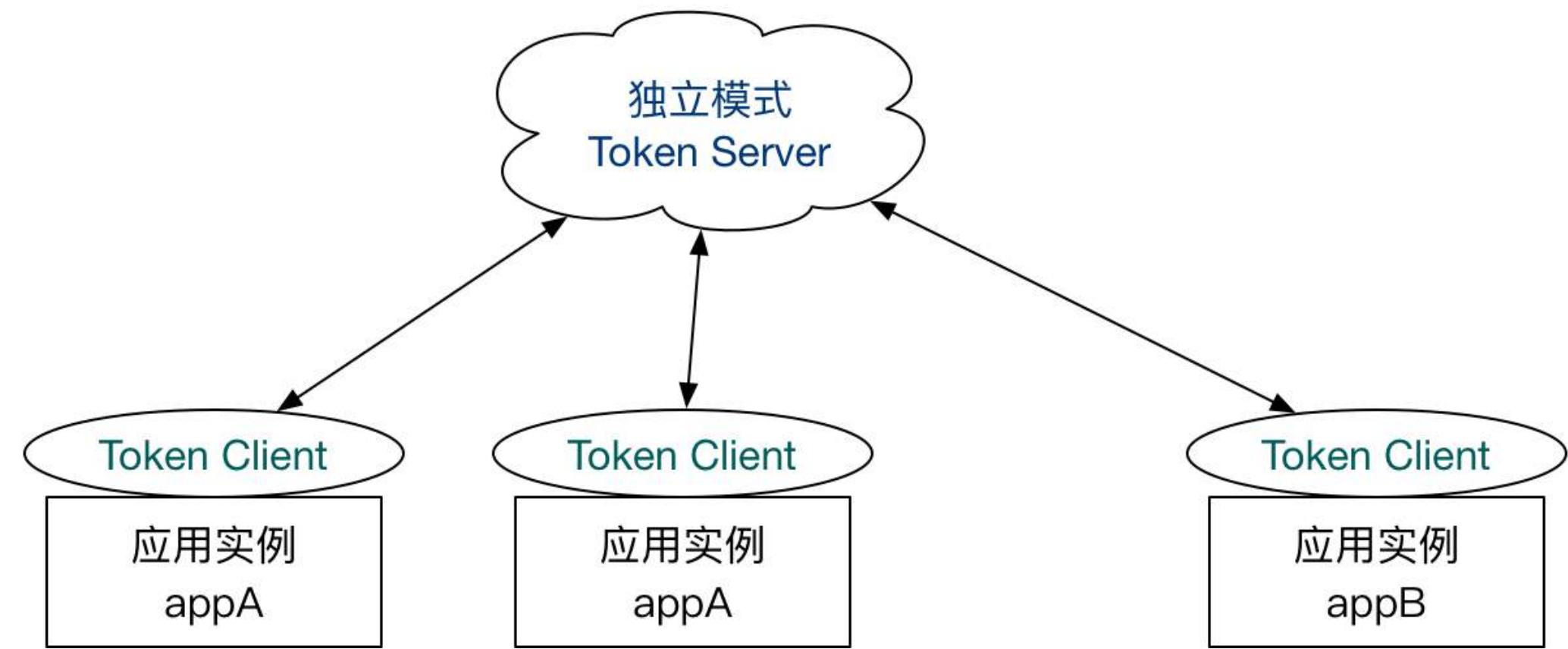
展望未来

为什么要使用集群限流？

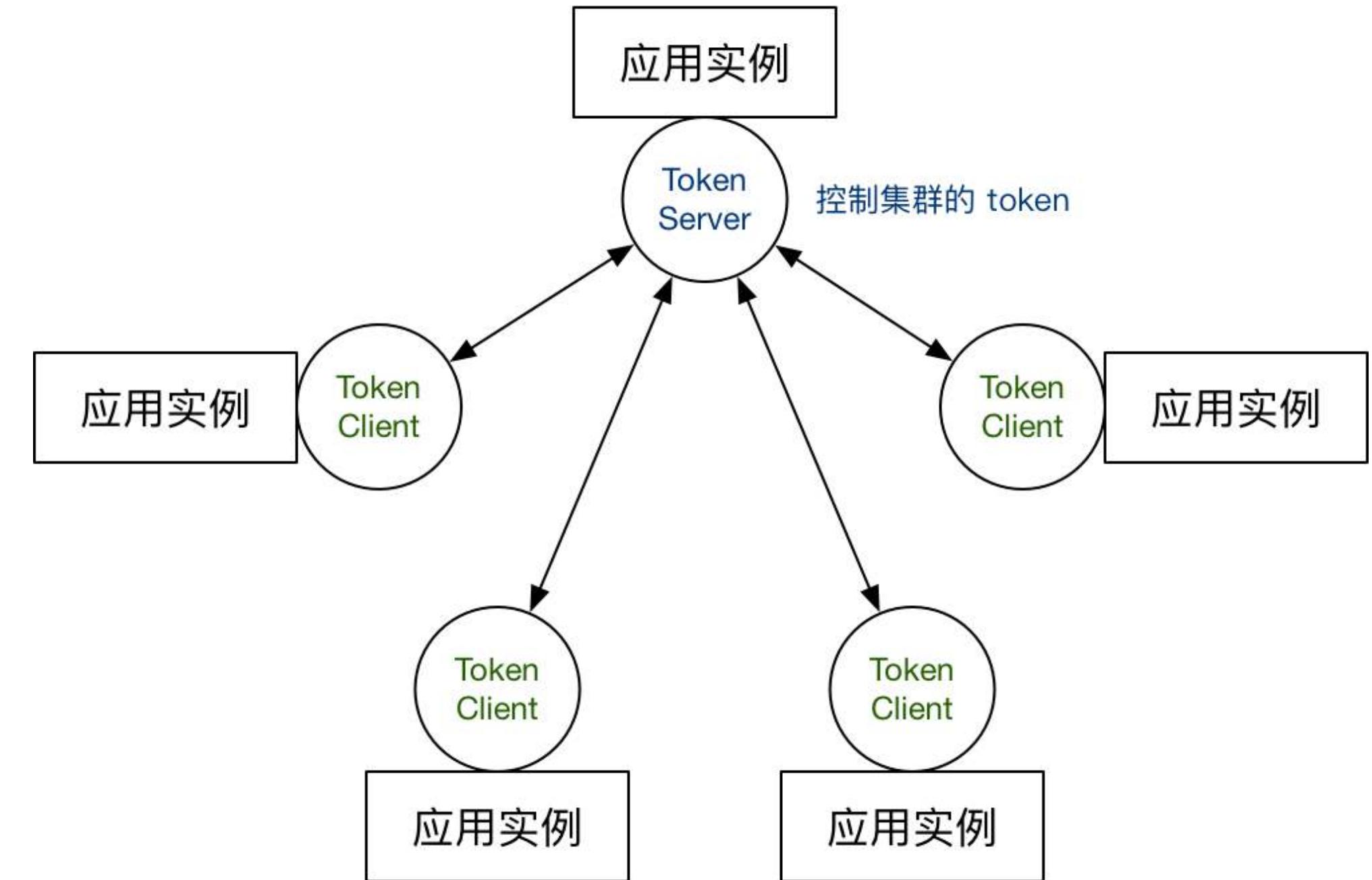
- 假设有 100 台机器，现在需要限制整个集群调用某个服务的总 QPS 不超过 50
- 在 API Gateway 处统计某个 API 的总访问量，并对某个 API 或服务的总 QPS 进行限制
- Service Mesh 中对服务间的调用进行全局流控
- 集群内对热点商品的总访问频次进行限制



部署方式

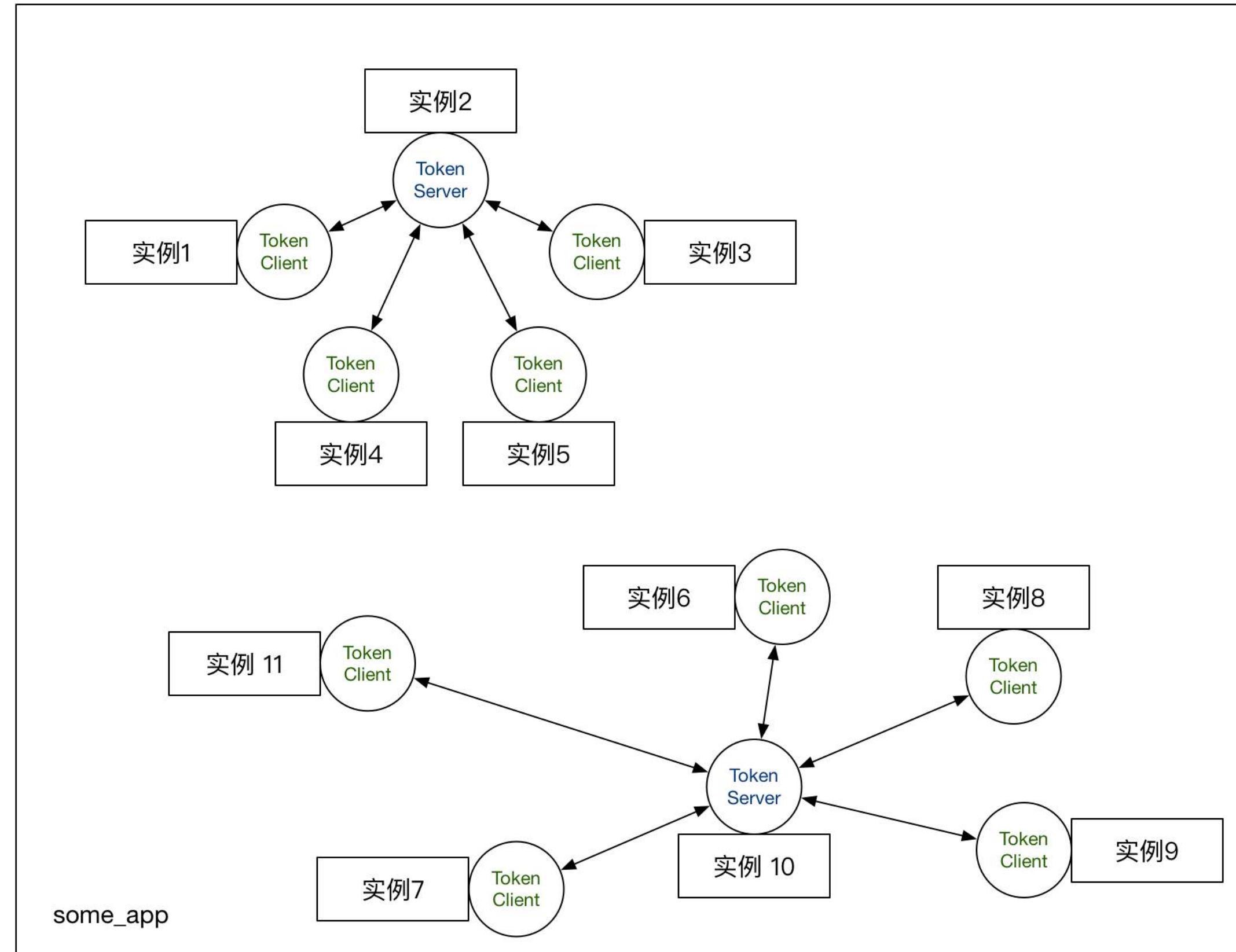


独立模式 (Alone)

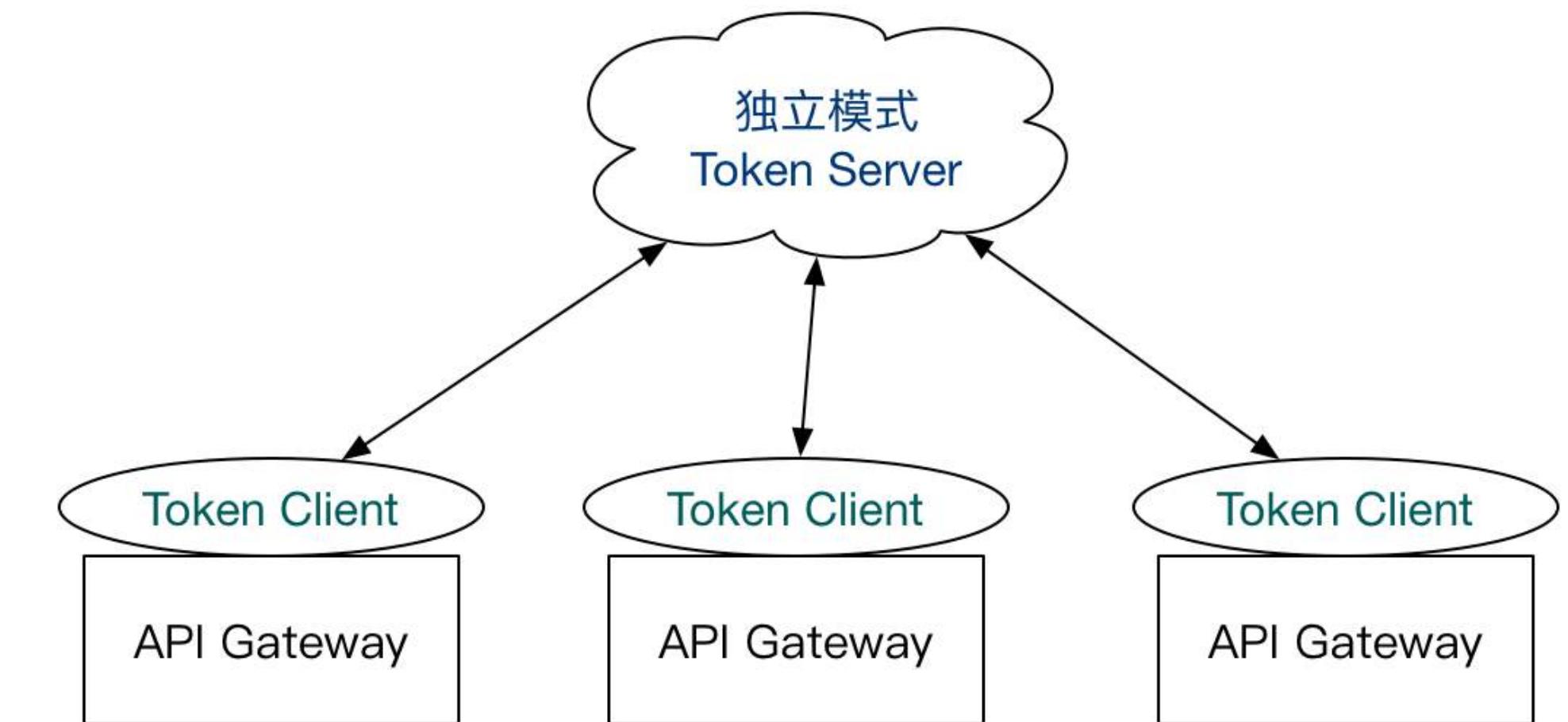


嵌入模式 (Embedded)

Token Server / Client 分配

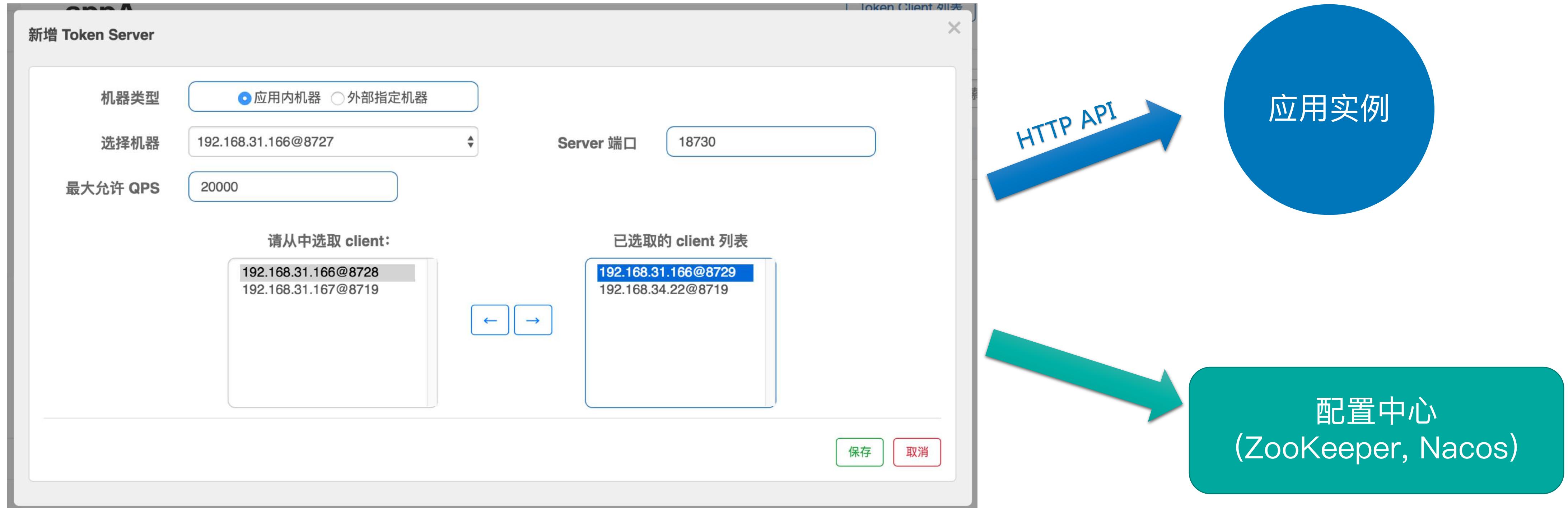


多 token server
单机均摊阈值模式



单台 token server
总体阈值模式、单机均摊阈值模式

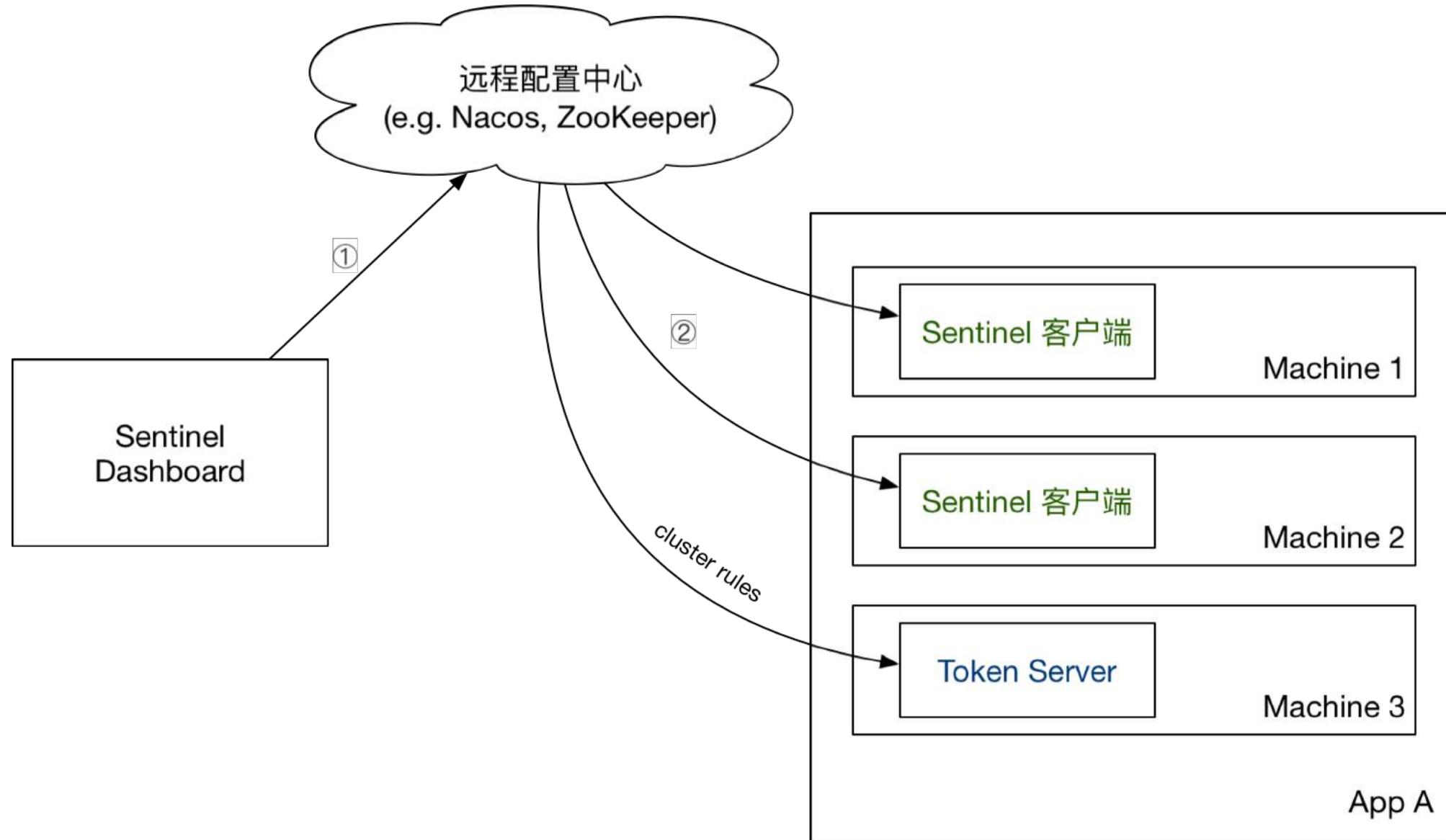
Token Server / Client 分配



```
[{"ip": "112.12.88.68", "machineld": "112.12.88.68@8728", "port": 11111,  
{"clientSet": ["112.12.88.66@8729", "112.12.88.67@8727"]}]
```

两种推送方式

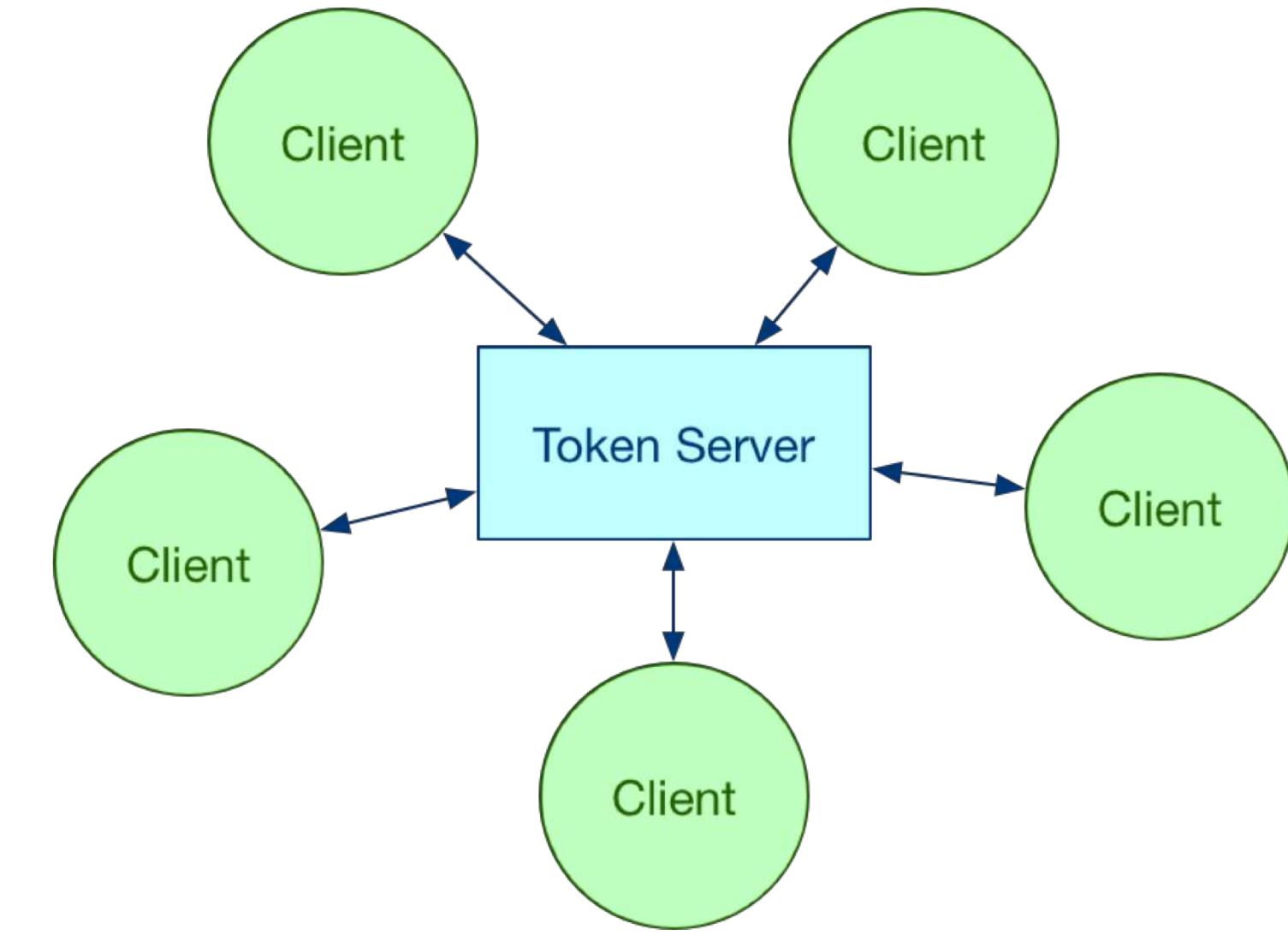
集群规则配置



- 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

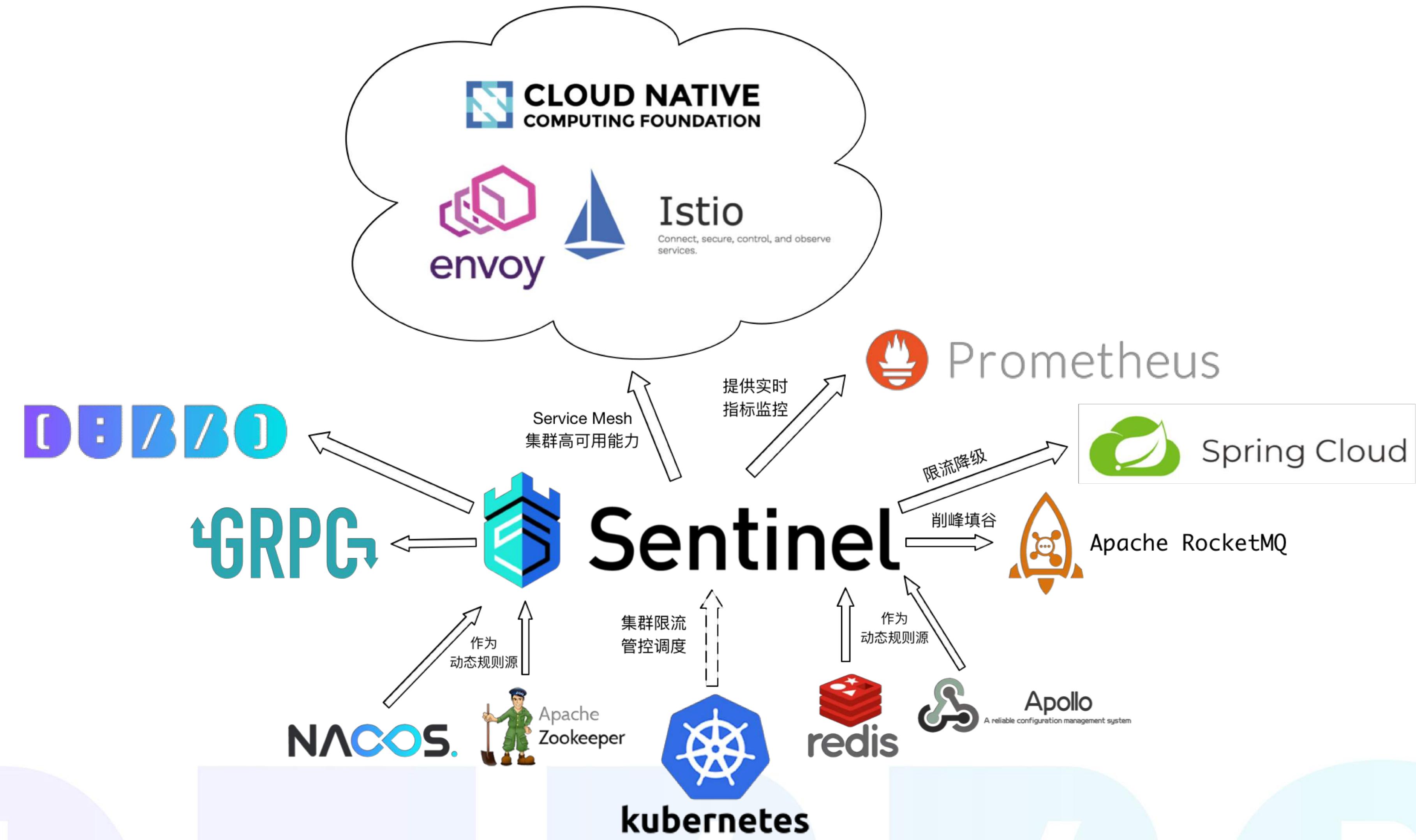
展望未来

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

The screenshot shows the GitHub repository interface for the 'sentinel' project. The top navigation bar includes links for Code, Issues (78), Pull requests (11), Projects (0), Wiki, Insights, and Settings. A prominent red box highlights the 'Issues' tab. A modal window titled 'Label issues and pull requests for new contributors' provides guidance on identifying issues for beginners, specifically those labeled 'help wanted' or 'good first issue'. A red arrow points to the 'good first issue' button. Below the modal, the 'Pinned issues' section displays two pinned issues: '#18 Wanted: Who is using Sentinel' and '#391 How to contribute | 如何参与开源 贡献'. A red arrow labeled '贡献指南' points to the second pinned issue. The main issue list is filtered by 'is:issue is:open' (indicated by a red box). A red arrow labeled 'search issue here' points to the search bar. Other filtering options include 'Labels' and 'Milestones'. A green 'New issue' button is located on the right. The issue list itself contains several items, such as '#415 sentinel 客户端的向控制台提交的 和端口 能否支持 spring boot 的配置文件, 而不是使用 JVM参数 kind/question', '#414 How to do persistence for parameter flow rules and cluster config kind/question', '#413 Add id field or extended identifiers in flow rules kind/enhancement', and '#408 sentinel-dashboard 的监控数据持久化扩展方式希望可以更优雅一些 kind/dashboard'.

Contact Us

Sentinel开源讨论群

1276人



钉钉群号 : 21977771



在钉钉上扫一扫加我

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

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



Thank you !