

1. Sentinel持久化模式
1.1 原始模式
1.2 拉模式
拉模式改造
1.3 推模式
1.3.1 基于Nacos配置中心控制台实现推送

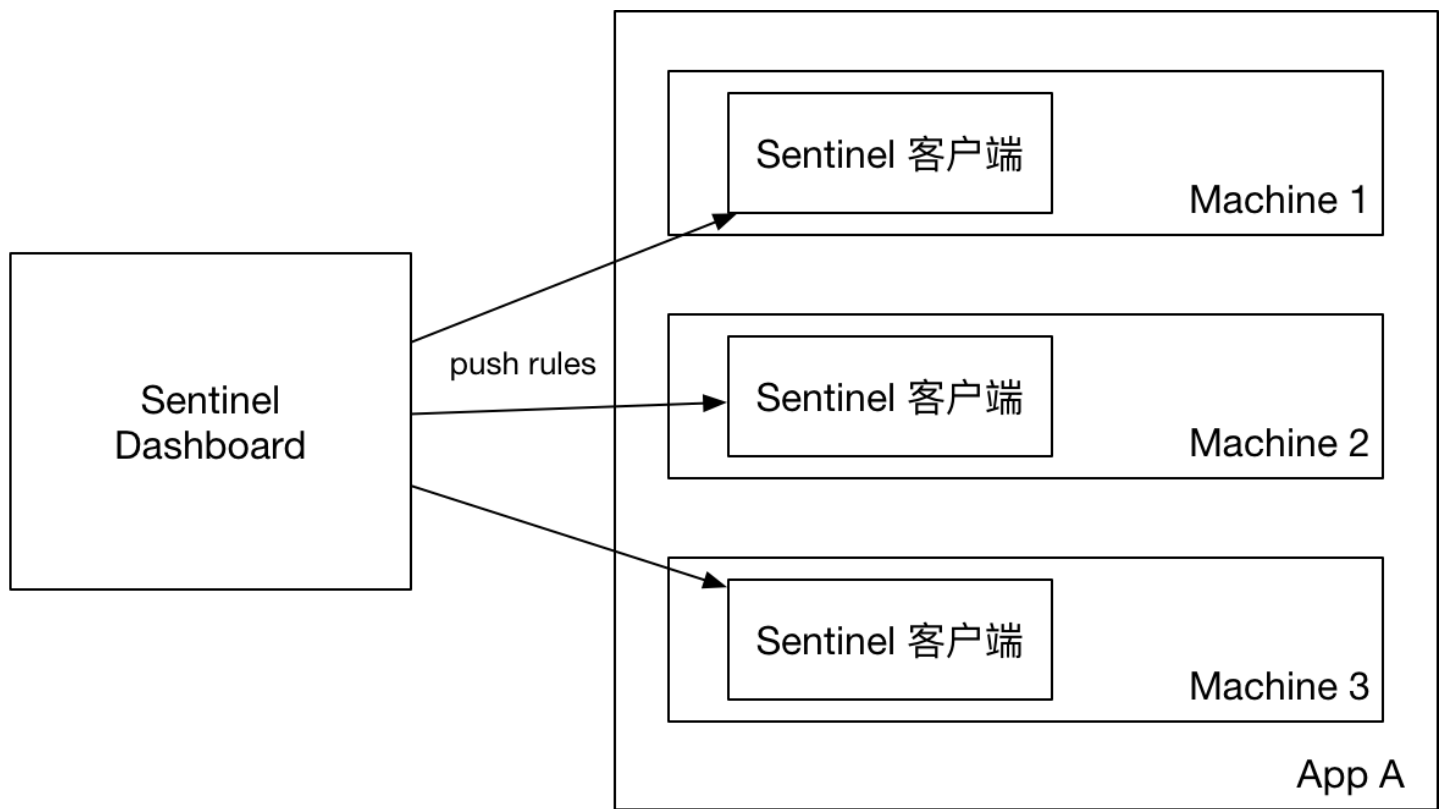
1. Sentinel持久化模式

Sentinel规则的推送有下面三种模式:

推送模式	说明	优点	缺点
原始模式	API 将规则推送至客户端并直接更新到内存中，扩展写数据源 (WritableDataSource)	简单，无任何依赖	不持久化，重启丢失
Pull 模式	扩展写数据源 (WritableDataSource)，客户端主动向某个规则管理中心定期轮询拉取规则，这个规则中心可以是 RDBMS、文件 等	简单，无任何依赖；规则持久化	不过时
Push 模式	扩展读数据源 (ReadableDataSource)，规则中心统一推送，客户端通过注册监听器的方式时刻监听变化，比如使用 Nacos、Zookeeper 等配置中心。这种方式有更好的实时性和一致性保证。生产环境下一一般采用 push 模式的数据源。	规则持久化；一致性；快速	引入依赖

1.1 原始模式

如果不做任何修改，Dashboard 的推送规则方式是通过 API 将规则推送至客户端并直接更新到内存中：



这种做法的好处是简单，无依赖；坏处是应用重启规则就会消失，仅用于简单测试，不能用于生产环境。

1.2 拉模式

pull 模式的数据源（如本地文件、RDBMS 等）一般是可写入的。使用时需要在客户端注册数据源：将对应的读数据源注册至对应的 RuleManager，将写数据源注册至 transport 的 `WritableDataSourceRegistry` 中。

1.3 推模式

生产环境下一般更常用的是 push 模式的数据源。对于 push 模式的数据源,如远程配置中心（ZooKeeper, Nacos, Apollo等等），推送的操作不应由 Sentinel 客户端进行，而应该经控制台统一进行管理，直接进行推送，数据源仅负责获取配置中心推送的配置并更新到本地。因此推送规则正确做法应该是 **配置中心控制台/Sentinel 控制台 → 配置中心 → Sentinel 数据源 → Sentinel**，而不是经 Sentinel 数据源推送至配置中心。这样的流程就非常清晰了：

1.3.1 基于Nacos配置中心控制台实现推送

官方demo: sentinel-demo-nacos-datasource

引入依赖

```
1 <dependency>
2     <groupId>com.alibaba.csp</groupId>
3     <artifactId>sentinel-datasource-nacos</artifactId>
4 </dependency>
```

nacos配置中心中配置流控规则

```
1  [  
2    {  
3      "resource": "TestResource",  
4      "controlBehavior": 0,  
5      "count": 10.0,  
6      "grade": 1,  
7      "limitApp": "default",  
8      "strategy": 0  
9    }  
10 ]
```

* **Data ID:**

* **Group:**

[更多高级选项](#)

描述:

Beta发布: ☐ 默认不要勾选。

配置格式: ☒ TEXT ☐ JSON ☐ XML ☐ YAML ☐ HTML ☐ Properties

配置内容 ? :

```
1  [  
2    {  
3      "resource": "TestResource",  
4      "controlBehavior": 0,  
5      "count": 15.0,  
6      "grade": 1,  
7      "limitApp": "default",  
8      "strategy": 0  
9    }  
10 ]
```

2) yml中配置

```
1  spring:  
2    application:
```

```
3     name: mall-user-sentinel-demo
4 cloud:
5     nacos:
6         discovery:
7             server-addr: 127.0.0.1:8848
8
9     sentinel:
10        transport:
11            # 添加sentinel的控制台地址
12            dashboard: 127.0.0.1:8080
13            # 指定应用与Sentinel控制台交互的端口，应用本地会起一个该端口占用的HttpServer
14            port: 8719
15        datasource:
16            ds1:
17                nacos:
18                    server-addr: 127.0.0.1:8848
19                    dataId: ${spring.application.name}
20                    groupId: DEFAULT_GROUP
21                    data-type: json
22                    rule-type: flow
```

3) nacos配置中心中添加

```
1  [
2      {
3          "resource": "userinfo",
4          "limitApp": "default",
5          "grade": 1,
6          "count": 1,
7          "strategy": 0,
8          "controlBehavior": 0,
9          "clusterMode": false
10     }
11 ]
```

* Data ID: mall-user-sentinel-demo

* Group: DEFAULT_GROUP

[更多高级选项](#)

描述:

Beta发布: ☐ 默认不要勾选。

配置格式: ☐ TEXT ☒ JSON ☐ XML ☐ YAML ☐ HTML ☐ Properties

配置内容 :

```
1  [  
2    {  
3      "resource": "userinfo",  
4      "limitApp": "default",  
5      "grade": 1,  
6      "count": 1,  
7      "strategy": 0,  
8      "controlBehavior": 0,  
9      "clusterMode": false  
10   }  
11 ]
```

引入依赖

```
1  <!--sentinel持久化 采用 Nacos 作为规则配置数据源-->  
2  <dependency>  
3    <groupId>com.alibaba.csp</groupId>  
4    <artifactId>sentinel-datasource-nacos</artifactId>  
5  </dependency>
```

增加yml配置

```
1  server:  
2    port: 8806  
3  
4  spring:  
5    application:  
6      name: mall-user-sentinel-rule-push-demo #微服务名称  
7  
8    #配置nacos注册中心地址  
9    cloud:  
10     nacos:
```

```
11     discovery:
12         server-addr: 127.0.0.1:8848
13
14     sentinel:
15         transport:
16             # 添加sentinel的控制台地址
17             dashboard: 127.0.0.1:8080
18             # 指定应用与Sentinel控制台交互的端口，应用本地会起一个该端口占用的HttpServer
19             #port: 8719
20         datasource:
21             # ds1: #名称自定义，唯一
22             #     nacos:
23             #         server-addr: 127.0.0.1:8848
24             #         dataId: ${spring.application.name}
25             #         groupId: DEFAULT_GROUP
26             #         data-type: json
27             #         rule-type: flow
28         flow-rules:
29             nacos:
30                 server-addr: 127.0.0.1:8848
31                 dataId: ${spring.application.name}-flow-rules
32                 groupId: SENTINEL_GROUP # 注意groupId对应Sentinel Dashboard中的定义
33                 data-type: json
34                 rule-type: flow
35         degrade-rules:
36             nacos:
37                 server-addr: 127.0.0.1:8848
38                 dataId: ${spring.application.name}-degrade-rules
39                 groupId: SENTINEL_GROUP
40                 data-type: json
41                 rule-type: degrade
42         param-flow-rules:
43             nacos:
44                 server-addr: 127.0.0.1:8848
45                 dataId: ${spring.application.name}-param-flow-rules
46                 groupId: SENTINEL_GROUP
47                 data-type: json
48                 rule-type: param-flow
49         authority-rules:
50             nacos:
51                 server-addr: 127.0.0.1:8848
```

```
52     dataId: ${spring.application.name}-authority-rules
53     groupId: SENTINEL_GROUP
54     data-type: json
55     rule-type: authority
56 system-rules:
57     nacos:
58         server-addr: 127.0.0.1:8848
59         dataId: ${spring.application.name}-system-rules
60         groupId: SENTINEL_GROUP
61         data-type: json
62         rule-type: system
```

以流控规则测试，当在sentinel dashboard配置了流控规则，会在nacos配置中心生成对应的配置。

* Data ID:

mall-user-sentinel-rule-push-demo-flow-rules

* Group:

SENTINEL_GROUP

[更多高级选项](#)

描述:

Beta发布:

☐ 默认不要勾选。

配置格式:

☒ TEXT ☐ JSON ☐ XML ☐ YAML ☐ HTML ☐ Properties

配置内容 ?

```
1 all-user-sentinel-rule-push-demo", "clusterConfig": {"fallbackToLocalWhenFail": true, "sampleCount": 10,
: 0, "thresholdType": 0, "windowIntervalMs": 1000}, "clusterMode": false, "controlBehavior": 0, "count": 2.0,
": 1615958553275, "gmtModified": 1615958961828, "grade": 1, "id": 1, "ip": "192.168.3.1", "limitApp": "default"
2, "resource": "/user/findOrderByUserId/{id}", "strategy": 0}]
```