

分组限流终版方案

需求背景

概要设计

UI

规则

详细设计

规则构造

匹配规则

需求背景

存在下面这样的场景：期望将多个服务下的多个方法绑定成一个组进行限流，在方法级别的配置支持非、与 等表达式。

用一个具体的例子说明，考虑存在以下三个服务：

▼ ServiceA

Java

📄 复制代码

```
1 public interface ServiceA {  
2     void methodA1();  
3  
4     void methodA2();  
5  
6     void methodA3();  
7 }
```

▼ ServiceB

Java

📄 复制代码

```
1 public interface ServiceB {  
2     void methodB1();  
3  
4     void methodB2();  
5  
6     void methodB3();  
7 }
```

ServiceC Java | 复制代码

```
1 public interface ServiceC {  
2     void methodC1();  
3  
4     void methodC2();  
5  
6     void methodC3();  
7 }
```

需要支持以下的 **分组规则**：

- **规则1** 将 ServiceA、ServiceB、ServiceC 三个服务下的所有方法捆绑成一个组，这个组共享一个限流规则
- **规则2** 除了对 ServiceA 服务下的 methodA1 方法不限流外，仅对 ServiceB 服务下的 methodB1 限流，ServiceA、ServiceB、ServiceC 三个服务共享一个限流规则进行限流
- **规则3** 对 ServiceA 服务下的 methodA1 和 methodA2 方法不限流，对 ServiceB 服务下的 methodB1 和 methodB2 方法限流，对 ServiceC 服务下的所有方法限流，且共享一个限流规则

概要设计

UI

UI设计如下：

添加限流规则

* 规则名称

请输入规则名称

* 应用

请输入或选择应用名称

限流方向

服务端限流

客户端限流

☒ 分组限流

QPS ×

+

* 限流类型 ⓘ
QPS

* 运行模式
拦截模式
满足条件的请求会被限流

限流配置

* 限流算法 ⓘ
令牌桶算法

* 令牌桶系数
1

* 单位时间 ms
1000

* 限流阈值
100

流量精确匹配 ⓘ
与 +

分组规则 ⓘ

或

A 包含 a1 x a2 x

B 包含 b1 x b2 x

+

一个服务至多一条分组规则。

一条规则包含 服务名 、 操作符 、 方法集合 。操作符有： 包含 、 排除 、 包含所有 、 排除所有 四种，当选择 包含所有 和 排除所有 操作符时， 方法集合 输入框隐藏。

一条限流规则里，一个服务的所有 分组规则 互为 或 的逻辑关系。一条限流规则里， 分组规则 与其他流量精确匹配规则是 与 的逻辑关系。

规则

坚持复用规则下发模型的原则，对个别字段作出调整。

上面规则配置，数据面收到的规则如下：

```
1 {
2     // ..... 省略其它字段
3     "limitRules":
4     [
5         {
6             "resource": "#group#.#group#:o:r",
7             "limitRuleItems":
8             [
9                 {
10                    // ..... 省略其它字段
11                    "trafficConditions":
12                    [
13                        {
14                            "type": "group",
15                            "field": "A",
16                            "operation": "INCLUDE",
17                            "value":
18                            [
19                                "a1",
20                                "a2",
21                            ]
22                        },
23                        {
24                            "type": "group",
25                            "field": "B",
26                            "operation": "EXCLUDE",
27                            "value":
28                            [
29                                "b1",
30                                "b2",
31                            ]
32                        }
33                    ]
34                }
35            ]
36        }
37    ]
38 }
```

resource 格式为： `${service}.${method}:${classification}:${flowtype}`，如上所示，分组限流规则的 resource 表示为 `#group#.#group#:o:r` 或 `#group#.#group#:i:r`。

还是复用 `limitRules.limitRuleItems.trafficConditions` 属性，以 `type:group` 表示分组规则，`field` 表示服务名，`operation` 增加 `INCLUDE`、`EXCLUDE`、`INCLUDE_ALL` 和 `EXCLUDE_ALL`，分别表示 包含、排除、包含所有、排除所有，`value`是方法的集合。

详细设计

分组规则在内存中的数据结构采用《分组限流》第一版中的设计，即：

Go | 复制代码

```
1 // < service , < method , limited > >
2 ▾ UnionParts map[string]map[string]bool
```

如上所示，用一个嵌套的两层map表示，外层map的key表示服务，内层map的key表示方法，内层map的value的值是一个bool值，true表示命中，false表示未命中。

用 `other_services` 表示未明确在控制台上配置的服务，`other_methods` 表示未明确在控制台上配置的方法。

为方便后续讨论，假设存在 A、B、C、D 四个服务，每个服务有多个方法，每个服务的方法用服务小写字母加数字表示，如 `a1,a2,a3...` 表示 A 服务的方法，以此类推。

规则构造

对于特定服务的特定方法的 `limited` 取值逻辑依赖页面上的 `选择逻辑` 字段：包含 取true，排除 取false，对于 `other_methods` 则“取反”（除 包含所有 和 排除所有 外）；包含所有 构造 `other_methods = true`，排除所有 构造 `other_methods = false`。

举例说明：

控制台规则配置如下：

A 排除 `a1` `a2`

B 包含 `b1` `b2`

C 包含所有

对应内存数据模型为：

< A : < (a1:false),(a2:false),(other_methods:true) > >

< B : < (b1:true),(b2:true),(other_methods:false) > >

< C : < (other_methods:true) > >

< other_services : < (other_methods:false) > >

匹配规则

一个请求，解析出服务名 service 以及对应方法 method，用 service 和 method 去上述规则中检索 `limited` 值，如果内存中表示分组规则的 map 中未存在 service 或 method，则检索默认的 other_services 或 other_methods，检索到的 `limited` 值 true 表示限流，false 表示不限流。