Alibaba Sentinel 预研说明文档

2021/5/21

业务场景

- 场景一：脉冲数据 背压

服务器宕机、路灯数据上报等场景下，高频数据会堆积，从而出现数据脉冲现象，这种场景下，期望通过限流模块逐一输出堆积的数据。

- 场景二：高频数据采集

对于高频数据，同场景一处理，大于漏桶容器的数据自动丢弃。

Sentinel 预研目标

预研目标：确认 Sentinel 的**限流设置**，重点是`maxQueueingTimeMs`属性的作用。

Sentinel 核心概念

QPS

QPS (Query Per Second, 每秒查询率)，是一台服务器每秒能够响应的查询次数，是对一个特定的查询服务器在规定时间内所处理流量多少的衡量标准。

Resource

Sentinel 通过资源来保护具体的业务代码或其他后方服务。

Sentinel 把复杂的逻辑给屏蔽掉了，用户只需要为受保护的代码或服务定义一个资源，然后定义规则就可以了，剩下的交给 Sentinel 处理。

Entry

Entry 表示一次资源操作，每次执行 SphU.entry() 或 SphO.entry() 都会返回一个Entry。

Sentinel Dashboard (控制台)

概述

预研案例主要分为以下几个步骤：

> 1. 定义需要限流资源 (对需要保护的资源设置限流规则)
> 2. 定义限流规则
> 3. 检验限流规则是否生效

启动控制台

```
java -jar sentinel-dashboard-1.8.1.jar
```

控制台应用

pom.xml

```
<dependencies>
 <dependency>
     <groupId>com.alibaba.csp</groupId>
     <artifactId>sentinel-transport-simple-http</artifactId>
     <version>1.8.1</version>
 </dependency>
 <dependency>
     <groupId>com.alibaba.csp</groupId>
     <artifactId>sentinel-annotation-aspectj</artifactId>
     <version>1.8.1</version>
 </dependency>
 <dependency>
     <groupId>com.alibaba.csp</groupId>
     <artifactId>sentinel-parameter-flow-control</artifactId>
     <version>1.8.1</version>
     <scope>test</scope>
 </dependency>
</dependencies>
```

控制台服务器配置

```
csp.sentinel.dashboard.server=localhost:8080
```

Hello.java

针对资源`Hello_Sentinel`，通过`initFlowRules`设置限流规则，其中参数的含义如下：

> - resource：设置需要保护的资源，这个资源的名称必须和`SphU.entry()`中使用的名称一致
> - grade：限流阈值类型
>   - `1`：QPS 模式
>   - `0`：并发线程数模式
> - count：限流阈值

```
import java.util.ArrayList;
import java.util.List;

import com.alibaba.csp.sentinel.Entry;
import com.alibaba.csp.sentinel.SphU;
import com.alibaba.csp.sentinel.slots.block.BlockException;
import com.alibaba.csp.sentinel.slots.block.RuleConstant;
import com.alibaba.csp.sentinel.slots.block.flow.FlowRule;
import com.alibaba.csp.sentinel.slots.block.flow.FlowRuleManager;

public class Hello {
    public static void main(String[] args) {
        // 配置规则.
        initFlowRules();

        while (true) {
            doSomething();
        }
    }

    private static void initFlowRules() {
        List<FlowRule> rules = new ArrayList<>();
        FlowRule rule = new FlowRule();
        rule.setResource("Hello_Sentinel");
        rule.setGrade(RuleConstant.FLOW_GRADE_QPS);
        // Set limit QPS to 20. (每秒最多允许通过20个请求)
        rule.setCount(20);
        rules.add(rule);
        FlowRuleManager.loadRules(rules);
    }

    private static void doSomething() {
        // 1.5.0 版本开始可以直接利用 try-with-resources 特性
        try (Entry entry = SphU.entry("Hello_Sentinel")) {
            // 被保护的逻辑 (业务逻辑处理)
            System.out.println("hello world" + System.currentTimeMillis());
        } catch (BlockException ex) {
            // 处理被流控的逻辑 (业务逻辑处理)
            System.out.println("blocked!");
        }
    }
```

```
}
```

Sentinel 滑动时间窗口

- window Size：1s
- window Slide：1s / QPS * 1000ms

Sentinel 相关属性

limitApp (调用方限流)

根据请求来源进行流量控制，选项：

> 1. `default`：表示不区分调用者
>
>   也就是任何访问调用者的请求都会进行限流统计。
> 2. `<some_origin_name>`：设置特定的调用者
>
>   只用来自这个的调用者请求才会进行流量统计和控制。
> 3. `other`：表示针对除 `<some_origin_name>` 之外的其他调用者进行流量控制。

同一个资源可以配置多条规则，生效顺序为：`<some_origin_name>` -> `other` -> `default`

controlBehavior (QPS 流量控制行为)

当 QPS 超过阈值时，触发流量控制行为。

这种行为是通过`controlBehavior`来设置的，包含：

> 1. 直接拒绝 (RuleConstant.CONTROL_BEHAVIOR_DEFAULT)，默认
>
>   请求流量超过阈值时，直接抛出异常。
> 2. Warm Up (RuleConstant.CONTROL_BEHAVIOR_WARM_UP)，冷启动
>
>   处理场景：当流量突然增加的时候，我们希望处理逐步递增。
> 3. 匀速排队 (RuleConstant.CONTROL_BEHAVIOR_RATE_LIMITER)
>
>   这种方式会严格控制请求通过的时间间隔，也就是让请求匀速通过。这种可以处理间隔性突发流量。
> 4. 冷启动 + 匀速排队 (RuleConstant.CONTROL_BEHAVIOR_WARM_UP_RATE_LIMITER)

count

限流阈值，定义了每秒最多接收的请求个数 (set limit QPS to)。

maxQueueingTimeMs 及其他

参数	说明
controlBehavior	0：直接拒绝 (默认) 2：匀速通过
maxQueueingTimeMs	当 controlBehavior=2 时，排队等待时间
burstCount	应对突发流量额外允许的数量

预研结论

流控规则 (rule) 设置：

> - rule.setCount()： 设置 QPS 阈值，比如，设置为 `20` 。
> - rule.setMaxQueueingTimeMs()： 排队等待时间设置范围大于 `窗口步长`，比如，50ms (`1 / 20 * 1000`) 以上。