ALIBABA CLOUD

阿里云

应用高可用服务

应用防护

文档版本：20220609

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.什么是应用防护 07
2.应用防护规则适用场景 08
3.支持组件列表 12
4.接入应用 14
4.1. 接入应用方式 14
4.2. 接入Java应用 16
4.2.1. 通过Java Agent接入 16
4.2.2. 接入容器服务Kubernetes版应用 18
4.2.3. 接入SAE应用 21
4.3. 接入JAVA SDK应用 22
4.3.1. 接入Spring Boot/Spring Cloud应用 22
4.3.2. 接入Spring应用 24
4.3.3. 接入Dubbo应用 26
4.3.4. 接入HSF应用 28
4.3.5. 接入Web应用 29
4.3.6. 通过自定义埋点接入 31
4.3.7. 接入MyBatis应用 33
4.3.8. 通过开源Sentinel SDK接入 35
4.3.9. 接入Http Client 35
4.4. 接入Go应用 37
4.5. 接入PHP应用 42
5.应用大盘 44
6.配置规则 46
6.1. 配置流控规则 46
6.2. 配置隔离规则 51
6.3. 配置熔断规则 54

6.4. 配置主动降级规则 58
6.5. 自适应流控 60
6.6. 配置热点规则 62
6.6.1. 配置热点规则 62
6.6.2. 查看热点监控详情 65
6.7. 配置自动重试规则 67
7.配置行为 70
7.1. 配置Web行为 70
7.2. 配置RPC行为 72
8.场景防护 77
8.1. Web场景防护 77
9.集群流控 82
9.1. 配置集群流控规则 82
9.2. 查看集群详情 84
10.管理应用 87
10.1. 应用概览 87
10.2. 接口详情 89
10.3. 机器监控 92
10.4. 规则管理 94
10.5. 管理基本信息 95
10.6. 应用基础设置 96
10.7. 应用压测 97
10.7.1. 压测Dubbo服务 97
10.7.2. 压测Spring Cloud服务 102
11.管理告警 107
11.1. 管理告警规则 107
11.2. 查看告警详情 115
12.创建流量大盘 119

13.Mesh防护 120
14.SDK 使用手册 123
14.1. SDK参考概述 123
14.2. 定义资源 123
14.3. 配置触发规则后的逻辑 126
14.4. 常用类及其方法 128
14.5. 扩展接口 132
14.6. Java SDK 参考 134
14.7. 样例工程 134
14.8. 重要日志 135
14.9. Java SDK和Java Agent版本说明 136
15.参考信息 153
15.1. 应用防护原则概述 153
15.2. 查看License 153
15.3. 应用防护方法 154
15.3.1. 削峰填谷 154
15.3.2. 关联限流 155
15.3.3. Warm Up（冷启动） 155
15.3.4. 服务提供方或消费方流控 156
15.3.5. 弱依赖降级 158
15.3.6. 强依赖隔离 158
15.3.7. 系统防护 159
15.4. 性能基准 160
15.5. 客户端启动参数 161
15.6. 开源Sentinel与商业化AHAS流量防护对比 163

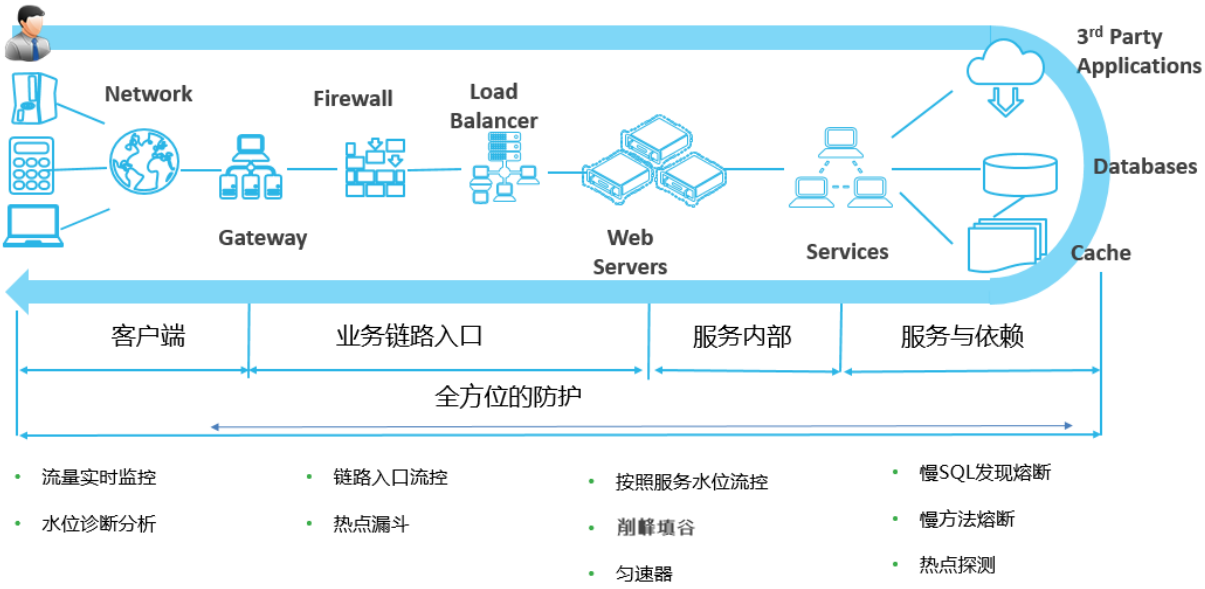
1.什么是应用防护

AHAS应用防护以流量为切入点，从流量控制、熔断降级、系统负载保护等多个维度来保障业务的稳定性，提供更专业稳定的流量防护手段、秒级的流量水位分布分析功能，是阿里巴巴双十一技术体系中的核心组件，同时也是开源框架Sentinel的商业化产品。

使用场景

AHAS应用防护广泛用于秒杀场景、消息削峰填谷、集群流量控制、实时熔断等场景中，从多个维度保障您的业务稳定性。

在一个常见的分布式应用中，如下图所示。一个请求先通过终端到达Gateway，再经过防火墙和网络负载均衡，其中还包括调用下游的其它服务和第三方应用，才能到达前端网络服务。AHAS应用防护在不同的层次以流量为切面提供秒级实时的流量分析（例如在客户端层提供流量实时监控和水位诊断分析功能），帮助运维人员采取针对性的防护措施，全方位地保护应用的稳定性。



功能特性

- 秒级流量分析功能，动态规则实时推送。
- 专业多样化的防护手段：
  - 入口流量控制：按照服务容量进行流量控制，常用于应用入口，例如：Gateway、前端应用、服务提供方等。
  - 热点隔离：将热点和普通流量隔离出来，避免无效热点抢占正常流量的容量。
  - 对依赖方隔离或降级：对应用和应用之间、应用内部采用隔离或降级手段，将不稳定的依赖对应用的影响减至最小，从而保证应用的稳定性。
  - 系统防护：AHAS应用防护可以根据系统的能力（例如Load、CPU使用率等）来动态调节入口的流量，保证系统稳定性。
- 实时的单机监控能力，强大的聚合监控和历史监控查询能力。

参考文档

关于Sentinel的详细介绍，请参见Sentinel介绍。

2.应用防护规则适用场景

微服务的稳定性一直是您非常关注的话题。随着业务从单体架构向分布式架构演进以及部署方式的变化，服务之间的依赖关系变得越来越复杂，业务系统也面临着巨大的高可用挑战。AHAS应用防护就是一款借助流量控制、熔断降级等模块，来提高应用高可用能力的产品。本文介绍各个应用防护规则以及适用的场景。

不稳定场景

在生产环境中您可能遇到过以下不稳定的情况：

- 大促时瞬间洪峰流量使得系统超出最大负载、Load飙高、系统崩溃导致用户无法下单。
- “黑马”热点商品击穿缓存、数据库被打垮、挤占正常流量。
- 调用端被不稳定第三方服务拖垮、线程池被占满、调用堆积，导致整个调用链路卡死。

这些不稳定的场景可能会导致严重后果，但很多时候开发者容易忽视这些与流量、依赖相关的高可用防护。AHAS应用防护功能就可以预防这些不稳定因素带来的影响，针对流量进行高可用的防护，从而保障服务“稳如磐石”。

核心场景

应用防护各规则说明和适用的核心场景如下表所示。

应用防护规则	描述	核心场景	说明文档
流量控制规则	通过AHAS配置QPS模式的流控规则，当每秒的请求量超过设定的阈值时，会自动拒绝多余的请求。	适用于需要限制突发的流量，在尽可能处理请求的同时来保障服务不被击垮的场景。	配置流控规则
集群流控规则	控制某个服务调用整个集群的实时调用量，解决因流量不均导致总体限流效果不佳的问题。	• 单机流量不均导致的单机限流效果不佳 • 集群小流量流控 • 有业务含义的流量控制（分钟小时级）	配置集群流控规则
隔离规则	控制某些调用的并发数（即正在进行的数目），防止过多的慢调用挤占正常的调用。	在调用第三方服务时，防止过多的慢调用挤占正常调用的资源，避免服务不可用。	配置隔离规则

应用防护规则	描述	核心场景	说明文档
熔断规则	对不稳定的弱依赖调用进行自动熔断降级，暂时切断不稳定调用，避免局部不稳定因素导致整体的雪崩。	避免局部不稳定因素（某个慢调用、异常服务）导致整体的雪崩，例如切断某个RT高的第三方服务调用，或针对某个ID的慢SQL访问进行熔断。	配置熔断规则
热点防护规则	自动识别热点参数并控制每个热点值的访问频次或并发量，可以有效地防止过“热”的参数访问挤占正常的调用资源。	适用于针对某些热点数据中访问频次最高的Top数据进行控制的场景，例如针对一段时间内最频繁购买的商品ID进行控制，防止突发热点商品击穿缓存而导致大量请求到数据库的情形。	配置热点规则
系统自适应保护规则	结合系统指标和服务容量，自适应动态调整流量。	用作全局的兜底防护规则。	自适应流控
自动重试规则	在客户端的部分场景下为系统提供自动重试的能力，同时支持自定义重试策略。	适用于服务偶现非致命性异常的场景（如偶现的超时）。	配置自动重试规则

流量控制规则

场景说明

流量是随机、不可预测的，可能就在某一时间会出现流量洪峰，例如双十一零点的场景。然而系统的容量总是有限的，如果突如其来的流量超过了系统的承受能力，就可能会导致请求处理堆积、堆积的请求处理缓慢、CPU/Load飙高，最后导致系统崩溃。因此，您需要针对这种突发的流量来进行限制，在尽可能处理请求的同时来保障服务不被击垮，这就是流量控制。流量控制的场景是非常通用的，适用于脉冲流量类等场景。

流控规则说明

通常在Web入口或服务提供方（Service Provider）的场景下，需要保护服务提供方自身不被流量洪峰打垮。此时通常根据服务提供方的服务能力进行流量控制，或针对特定的服务调用方进行限制。您可以结合前期压测评估核心接口的承受能力，通过AHAS配置QPS模式的流控规则，当每秒的请求量超过设定的阈值时，会自动拒绝多余的请求。关于流控规则的更多信息，请参见配置流控规则。

集群流控规则说明

同时AHAS也提供集群流控（分布式限流）的能力，可以控制某个服务调用整个集群的实时调用量，解决因流量不均导致总体限流效果不佳的问题。关于集群流控的更多信息，请参见配置集群流控规则。

如果您的业务符合以下场景，建议结合集群流控来保障服务稳定性：

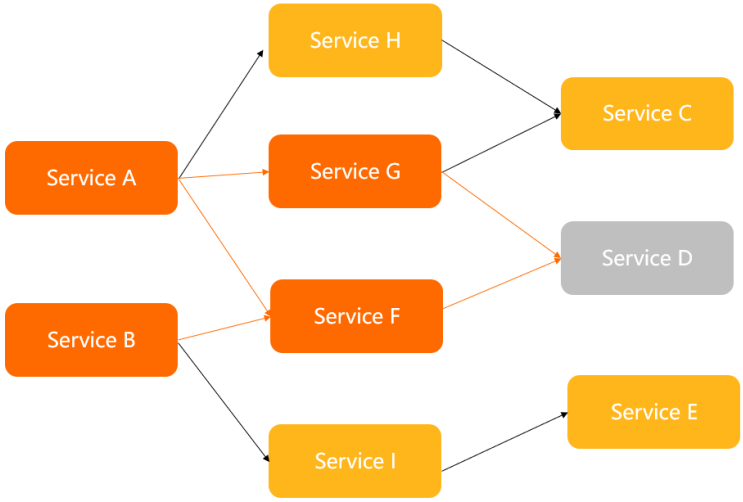
- 单机流量不均：由于负载不均衡等原因导致每台机器的流量不均，此时使用单机流控可能会出现没有达到请求总量，某些机器就开始限流的情况。
- 集群小流量流控：某些高可用防护场景下，需要将服务调用QPS限制到很小的量，此时平均到每台机器的QPS可能小于1，无法通过单机流控进行精确控制。例如限制总QPS为50，但节点数为100个，平均到每个节点QPS为0.5。
- 有业务含义的流量控制：例如限制某个API每个用户每分钟调用不超过10次。

并发控制与熔断规则

场景说明

一个服务常常会调用别的模块，可能是另外一个远程服务、数据库，或者第三方API等。例如，支付的时候，可能需要远程调用银联提供的API；查询某个商品的价格，可能需要进行数据库查询。然而，这个被依赖服务的稳定性是不能保证的。如果依赖的服务出现了不稳定的情况，请求的响应时间变长，则调用服务的方法的响应时间也会变长，线程会产生堆积，最终可能耗尽业务自身的线程池，服务本身也变得不可用。

微服务的架构示例图如下：



现代微服务架构都是分布式的，由许多微服务组成。不同服务之间相互调用，组成复杂的调用链路。以上的问题在链路调用中会产生放大的效果。复杂链路上的某一环不稳定，就可能会层层级联，最终导致整个链路都不可用。

规则说明

AHAS提供以下能力避免慢调用等不稳定因素造成服务不可用：

- 并发控制：作为一种轻量级隔离的手段，控制某些调用的并发数（即正在进行的数目），防止过多的慢调用挤占正常的调用。更多信息，请参见配置隔离规则。
- 熔断：对不稳定的弱依赖调用进行自动熔断降级，暂时切断不稳定调用，避免局部不稳定因素导致整体的雪崩。更多信息，请参见配置熔断规则。
  AHAS熔断规则基于熔断器模式的思想，在服务出现不稳定因素（例如响应时间变长、错误率上升）的时候暂时切断服务的调用，等待一段时间再进行尝试。一方面防止给不稳定服务“雪上加霜”，另一方面保护服务的调用方不被拖垮。AHAS应用防护支持两种熔断策略：基于响应时间（慢调用比例）和基于错误（异常比例），可以有效地针对各种不稳定的场景进行防护。

> **说明**
> - 熔断器模式一般适用于弱依赖调用，即降级后不影响业务主流程，开发者需要设计好降级后的回退逻辑（fallback）和返回值。
> - 即使服务调用方引入了熔断降级机制，开发者还是需要在HTTP或RPC客户端配置请求超时时间，来做一个兜底的防护。

热点防护规则

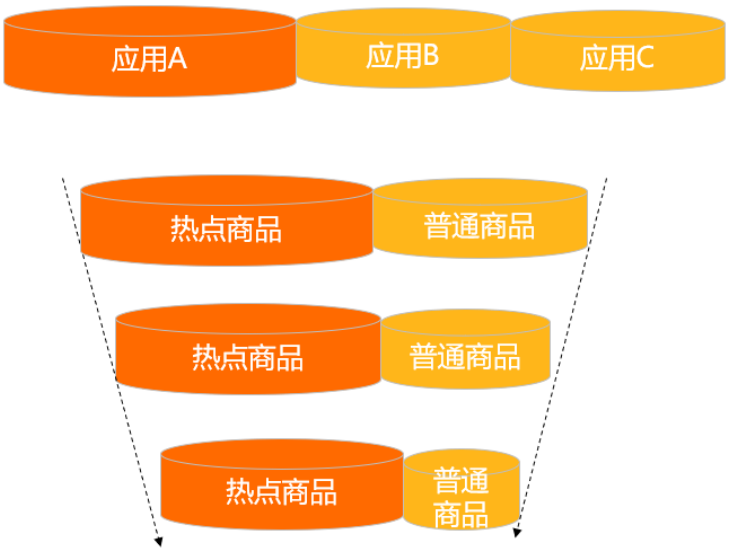
场景说明

为了防止被大流量打垮，您通常会对核心接口配置限流规则，但有的场景下配置普通的流控规则是不够的。

例如大促峰值的时候，会有不少“热点”商品，这些热点商品的瞬时访问量非常高。通常您可以事先预测一波热点商品，并对这些商品信息进行缓存“预热”，以便在出现大量访问时可以快速返回而不会都经过数据库。但每次大促都会涌现出一些“黑马”商品，这些“黑马”商品是无法事先预测的，没有被预热。当这些“黑马”商品访问量激增时，大量的请求会击穿缓存，直接经过数据库，导致数据库访问缓慢，挤占正常商品请求的资源池，最后导致系统崩溃。

规则说明

此时，利用AHAS的热点参数流控能力，自动识别热点参数并控制每个热点值的访问频次或并发量，可以有效地防止过“热”的参数访问挤占正常的调用资源。更多信息，请参见配置热点规则。



系统自适应保护规则

场景说明

当您无法事先准确评估某个接口的容量，甚至无法预知核心接口的流量特征（如是否有脉冲情况）时，靠事先配置的规则可能无法有效地保护当前服务节点。当某些情况下机器的Load和CPU usage等突然开始飙高，但您却无法快速确认原因，也来不及处理异常时，您其实需要做的是快速止损，先通过自动化的兜底防护手段，将濒临崩溃的微服务“拉”回来。

规则说明

针对这些情况，AHAS提供了一种系统自适应保护规则，结合系统指标和服务容量，自适应动态调整流量。AHAS自适应流控结合系统的Load、CPU使用率以及服务的入口QPS、响应时间和并发量等几个维度的监控指标，通过一定的流控策略，让系统的入口流量和系统的负载达到一个平衡，让系统尽可能运行在最大吞吐量，同时保证系统整体的稳定性。更多信息，请参见自适应流控。

系统规则可以作为整个服务的一个兜底防护策略，保障服务运行，对CPU密集型的场景会有较好的效果。

自动重试规则

场景说明

分布式系统中调用关系通常会比较复杂，服务之间的调用可能会偶尔出现异常。这些异常有的是不可恢复的（如业务错误），有的则是可以恢复的（如偶发的超时）。当系统遇到一些非致命性的错误（如偶现的超时等）时，可以通过重试的方式来避免系统最终失败。

规则说明

AHAS的自动重试规则可以在客户端的部分场景下为系统提供自动重试的能力，同时支持自定义重试策略。更多信息，请参见配置自动重试规则。

3.支持组件列表

AHAS为了简化应用的接入流程，对主流框架进行了适配。本文将列出AHAS支持的第三方组件和框架列表。

组件	支持版本	支持该组件的Java Agent版本	支持该组件的SDK版本
Dubbo	Agent：2.7.x，2.6.x SDK：2.7.x，2.6.x	All	All
Web Servlet	SDK：2.x+ Agent：3.0+	All	All
Spring Boot	1.3.x+	All	请参见说明
Spring MVC	4.x+	All	请参见说明
Spring Cloud Gateway	2.x	1.5.0+	请参见说明
Zuul 1.x	1.3.x	1.5.0+	请参见说明
GRPC-Java	1.13+	1.7.0	需另引入sentinel-grpc-adapter依赖
Jetty	8.x+	Servlet 3.0+ 支持：all	通过Servlet支持
Tomcat	7.x+	Servlet 3.0+ 支持：all	通过Servlet支持
WebLogic	10.3	Servlet 3.0+ 支持：all Servlet 2.x支持：1.6.0+	通过Servlet支持
HttpClient 3	3.x+	待支持	待支持
HttpClient 4	4.x+	待支持	支持
JDK HTTP	1.7.x+	待支持	待支持
OKHttp	2.x+	待支持	支持
MyBatis	3.x+	1.8.0+	1.4.1+
MySQL JDBC	5.0.x+	1.6.0+	通过MyBatis支持
Oracle JDBC	12.x	1.6.0+	通过MyBatis支持
PostgreSql JDBC	9.4+	待支持	不支持
SQLServer JDBC	6.4+	待支持	不支持
Redis Client（Jedis）	待支持	1.7.0	待支持
MemCached	2.8+	1.7.0	待支持
MongoDB	3.7+	待支持	待支持

组件	支持版本	支持该组件的Java Agent版本	支持该组件的SDK版本
RocketMQ （callback模式）	4.x	1.7.0	需手动埋点
RabbitMQ	3.7+	1.7.0	需手动埋点
SOFARPC	5.x	待支持	1.5.3+

说明

- 支持的AHAS Agent版本和SDK版本详情请参见版本说明。
- 从SDK/Starter 1.9.1版本开始，AHAS流量防护支持JDK 6～11版本。
- Spring Boot/Spring Cloud Web应用只需要引入 `spring-boot-starter-ahas-sentinel-client` 依赖即可接入。
- Spring Cloud Gateway网关需要引入 `spring-cloud-gateway-starter-ahas-sentinel` 依赖；Zuul 1.x网关需要引入 `spring-cloud-zuul-starter-ahas-sentinel` 依赖，无需引入其它依赖。

4.接入应用

4.1. 接入应用方式

AHAS应用防护可以接入Java、PHP、Go等多种语言的应用，接入后，即可配置流控、降级、隔离等规则来保证系统稳定性。本文提供关于各种接入方式的总览。

AHAS应用防护支持的接入方式

语言	接入方式	说明
Java	Agent接入	若应用使用的第三方组件和框架在支持组件列表中，则无需修改代码，通过Agent接入即可。
	SDK接入	包括以下接入方式： • Spring Boot/Spring Cloud应用 • Spring应用 • Dubbo应用 • Web应用 • 自定义埋点
	K8s接入	AHAS与容器服务Kubernetes版数据互通，因此容器服务Kubernetes版中的应用可以快速接入AHAS应用防护。
	SAE接入	对于在Serverless应用引擎SAE（Serverless App Engine）平台上部署的应用，可以使用AHAS应用防护对其配置流控、隔离等规则来保证系统稳定性。
Go	SDK接入	包括以下接入方式： • Dubbo应用 • Gin Web应用 • gRPC应用 • 自定义埋点
PHP	SDK接入	通过自定义埋点方式接入。

接入Java应用

通过Java Agent接入

通过Agent接入方式无需修改代码。

接入容器服务应用

对于部署在容器服务Kubernetes版中的Java应用，可以使用AHAS应用防护可以对其配置流控、降级和系统规则来保证系统稳定性。

接入SAE应用

对于在Serverless应用引擎SAE平台上部署的应用，可以使用AHAS应用防护对其配置流控、隔离等规则来保证系统稳定性。

接入Java SDK应用

接入Dubbo应用

通过SDK接入的方式，将Dubbo应用接入AHAS控制台，使用应用防护服务。

接入Spring Boot/Spring Cloud应用

通过SDK的方式将Spring Boot/Spring Cloud应用接入AHAS控制台，以使用应用防护服务。

接入Web应用

将Web应用接入AHAS应用防护后，可以对其配置流控、降级和系统规则来保证系统稳定性。

通过自定义埋点接入

通过自定义Java SDK埋点的方式将应用接入AHAS应用防护，可以更加灵活地对任意代码块进行操作。

通过注解接入

通过在业务逻辑上添加注解的方式，将应用接入AHAS控制台，使用应用防护服务。

通过开源Sentinel SDK接入

通过替换安装包中的JAR包即可实现Sentinel和AHAS应用防护的灵活切换。

接入MyBatis应用

接入介绍如何以Agent方式将MyBatis应用接入AHAS应用防护，接入后即可配置流控、降级和系统规则来保证系统稳定性。

接入GO应用

接入GO应用

将Go应用接入AHAS应用防护后，可以对其配置流控、隔离和系统规则来保证系统稳定性。

接入PHP应用

接入PHP应用

将PHP应用接入AHAS应用防护后，可配置流控、降级、隔离和系统规则来保证系统稳定性。

4.2. 接入Java应用

4.2.1. 通过Java Agent接入

若您的应用使用AHAS支持的第三方组件和框架，可使用Agent接入方式，零代码修改接入AHAS应用防护。本文以Java Agent为例介绍Agent接入方法。

前提条件

- 确认应用使用的第三方组件和框架在支持列表中，更多信息，请参见支持组件列表。
- 自1.9.1版本开始，Java Agent兼容JDK 6~11版本。对于通过Spring Cloud Alibaba Sentinel开源方式接入的应用，无需去除对应依赖也可直接通过Java Agent方式接入。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**右上角单击**新应用接入**。
5. 在**JAVA语言**页签，单击**Agent接入**页签。
6. 按需选择以下一种方式安装并启动Java Agent。
   - **方式一：快速接入**
     快速接入方式会识别主机中**所有Java进程**，并将其接入AHAS应用防护。

a. 执行以下命令下载Java Agent。

> **说明** EDAS应用请切换至Admin用户下载安装。

公网下载地址 非公网下载地址

b. 执行以下命令安装Java Agent。

- 非公网

```
sh ./ahas-agent.sh install -a <AppName> -n default&& source ~/.bash_profile
```

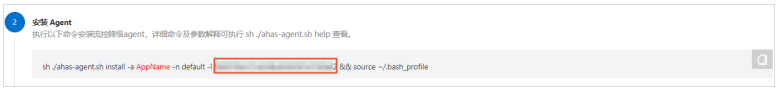
> **说明** 将<AppName>替换为自定义的应用名称。

- 公网

```
sh ./ahas-agent.sh install -a <AppName> -n default -l <license> && source ~/.bash_profile
```

> **说明** 将<AppName>替换为自定义的应用名称，将<license>替换为真实license值。

若在公网地域，需要查看License信息。请在第二步**安装Agent**区域查看（非公网地域不需要），具体操作，请参见查看License。



c. 重启您的应用。

- **方式二：手动接入**
  手动接入方式可以指定**单个应用**接入AHAS应用防护，操作时需要根据Java虚拟机运行的环境来安装Java Agent。

a. 下载ahas-java-agent.jar安装包至任意路径下，例如：*/opt/aliyunahas/agent*。下载安装包请单击ahas-java-agent.jar。

b. 添加JVM启动参数。

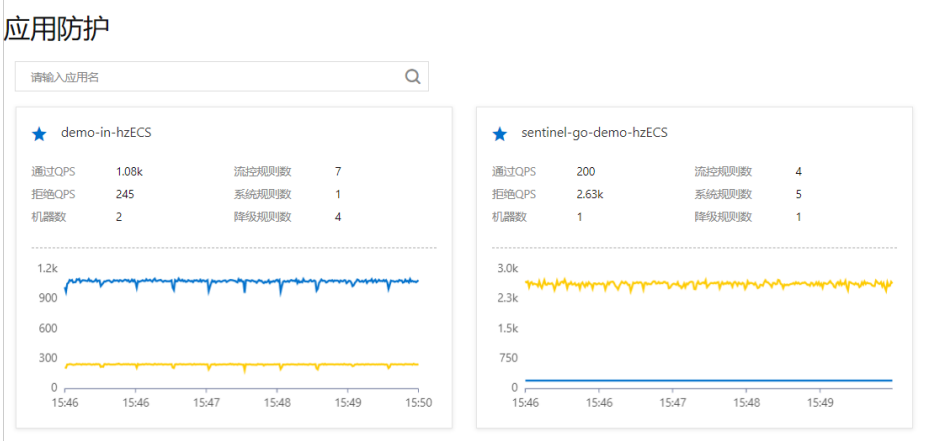
非公网 公网

c. 启动JVM。

> **说明**
>
> - Agent接入配置URL收敛：通过启动参数 `-Dcsp.sentinel.url.clean.config.path` 配置文件路径，默认路径为jar包路径下的*ahas-sentinel-url-clean.properties*文件。支持classpath形式，如 `-Dcsp.sentinel.url.clean.config.path=classpath:ahas-sentinel-url-clean.properties` 。
> - *Properties*文件的格式：匹配前缀等于清理后的资源名称。
> - 目前仅支持前缀匹配模式。如以下示例的配置会针对*/payment/*开头的URL，将它们归一成*/payment/*: `/payment/=/payment/*` 。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



常见问题

如果您已完成应用防护服务接入，应用防护页面仍查看不到您的应用，请参见应用防护常见问题。

4.2.2. 接入容器服务Kubernetes版应用

对于部署在容器服务Kubernetes版中的Java应用，可以使用AHAS应用防护对其配置流控、降级和系统规则来保证系统稳定性。本文将介绍如何将容器服务Kubernetes版中的应用接入AHAS应用防护。

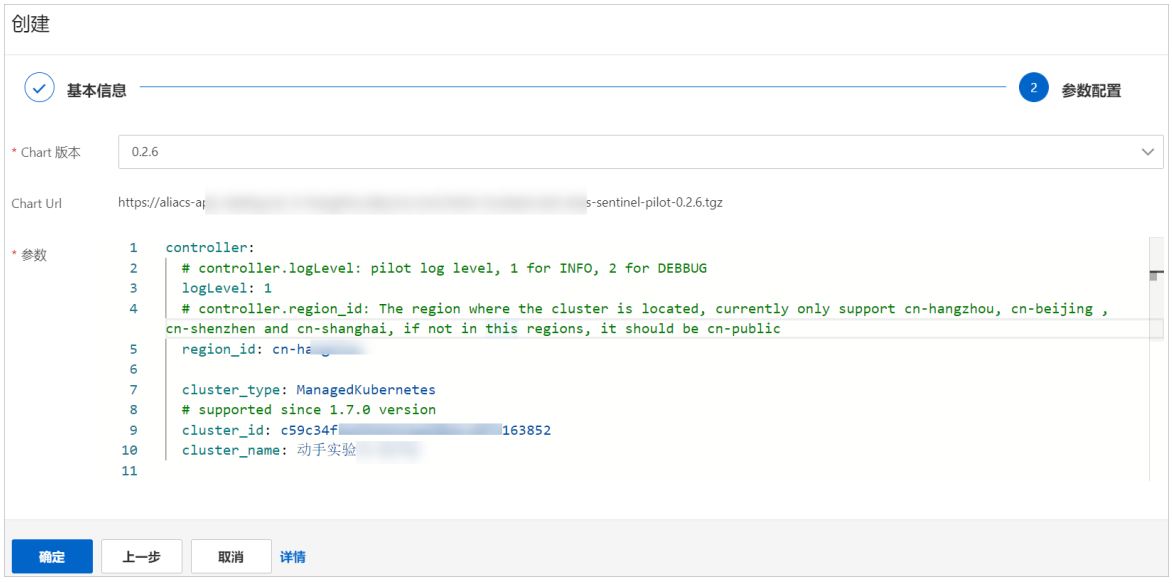
前提条件

快速创建Kubernetes托管版集群

步骤一：安装AHAS组件

在容器服务Kubernetes中安装AHAS组件后才能将Java应用接入AHAS应用防护。

1. 登录容器服务管理控制台。
2. 在控制台左侧导航栏中选择**市场 > 应用市场**。
3. 在**应用目录**页面的搜索框中输入**ack-ahas-sentinel-pilot**应用并单击。
4. 在**ack-ahas-sentinel-pilot**页面右上角单击**一键部署**，在弹出的**创建**面板选择目标集群，单击**下一步**然后单击**确定**。



参数	描述	备注
region_id	○ 如果集群和VPC之间有专线，该参数为专线连接的region。 ○ 如果集群和VPC之间没有专线，该参数填入cn-public。	根据所选集群自动生成。
cluster_id	您的集群ID。	
cluster_name	您的集群名称。	

步骤二：为Java应用开启AHAS应用防护

您可以按需为新建的应用或已有的应用开启AHAS应用防护。

- 如需在创建新应用的同时开启AHAS应用防护，具体步骤如下：
  i. 登录容器服务管理控制台。
  ii. 在控制台左侧导航栏中，单击**集群**。
  iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
  iv. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。
  v. 在**无状态 Deployment**页面右上角单击**使用YAML创建资源**。
  vi. 在**创建**页面上方选择**示例模板**，在**模板**中替换 `image` 为Java应用的image，并在**模板**中将以下 `annotations` 添加到**spec > template > metadata**层级下，然后单击**创建**。
  如需修改YAML文件中其它配置项，配置项说明如下：
  配置项

Parameter	Description	Default
image.imageTag	pilot镜像tag。	0.1.1
image.imagePullPolicy	镜像拉取策略，必须是Always、IfNotPresent、Never三者中的一个。	Always
controller.logLevel	pilot日志级别，1表示INFO，2表示DEBBUG。	1
controller.region_id	目标集群所在的region，如cn-hangzhou、cn-beijing、cn-shenzhen、cn-shanghai。如果是公网，则为cn-public。	cn-hangzhou

  完整YAML示例模板如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-foo
  labels:
    name: agent-foo
spec:
  replicas: 1
  selector:
    matchLabels:
      name: agent-foo
  template:
    metadata:
      labels:
        name: agent-foo
      annotations:
        ahasPilotAutoEnable: "on"
        ahasAppName: "K8sFooTest"
        ahasNamespace: "default"
    spec:
      containers:
      - name: foo
        image: registry.cn-hangzhou.aliyuncs.com/sentinel-docker-repo/foo:0.1.1
        imagePullPolicy: Always
```

公网 非公网

- 如需为现有应用开启AHAS应用防护，操作步骤如下。
  i. 登录容器服务管理控制台。
  ii. 在控制台左侧导航栏中，单击**集群**。
  iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
  iv. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**或**工作负载 > 有状态**。
  v. 在**无状态（Deployment）**或**有状态（StatefulSet）**页面上，单击目标应用右侧**操作**列中选择**更多 > 查看YAML**。
  vi. 在**编辑YAML**对话框中将以下 `annotations` 添加到**spec > template > metadata**层级下，并单击**更新**。

  公网 非公网

结果验证

1. 登录容器服务管理控制台。
2. 在控制台左侧导航栏中，单击**集群**。
3. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
4. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**或**工作负载 > 有状态**。

在目标应用的**操作**列将出现**应用流控**按钮。单击**应用流控**即可跳转至AHAS控制台。

4.2.3. 接入SAE应用

对于在Serverless应用引擎SAE（Serverless App Engine）平台上部署的应用，可以使用AHAS应用防护对其配置流控、隔离等规则来保证系统稳定性。本文介绍如何将SAE平台上的应用接入AHAS应用防护。

前提条件

您已在SAE平台部署应用，具体操作，请参见应用部署。

背景信息

SAE是面向应用的Serverless PaaS平台，能够帮助PaaS层用户免运维IaaS、按需使用、按量计费，从而轻松实现微服务应用上云。相对于其他Serverless产品，SAE抽象了应用的概念，并提供了一整套微服务解决方案，支持Spring Cloud、Dubbo、HSF等主流的微服务开发框架，实现了Serverless架构和微服务架构的完美结合。

操作步骤

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面右上角单击**新应用接入**。然后在**JAVA语言**页签下，单击**SAE接入**。
4. 配置启动参数，添加JVM -D参数。

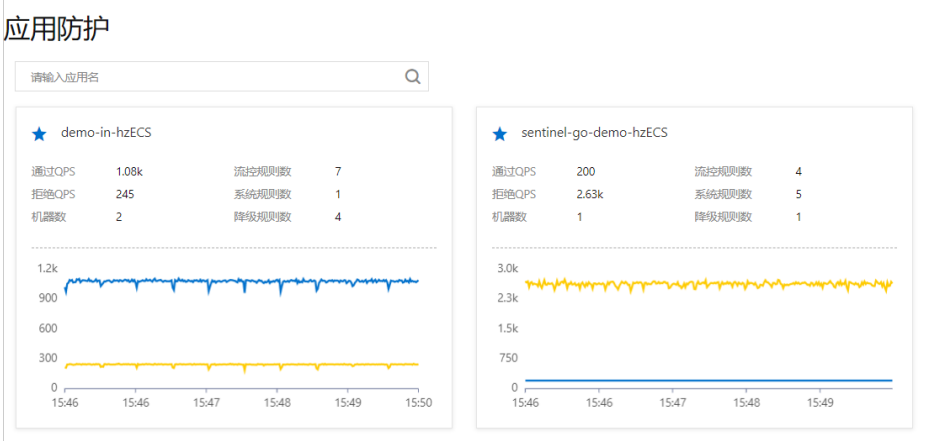
   ```
   -Dahas.namespace=default -Dproject.name=<AppName> -Dahas.license=<your license>
   ```

   注意事项如下：

   - `<AppName>` 只能包含字母、数字、下划线（_）、短划线（-）、英文句号（.）以及英文冒号（:）。
   - 请将 `<AppName>` 参数替换成您的应用名。
   - 请将 `<your license>` 替换成应用防护控制台界面实际显示的license。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



4.3. 接入JAVA SDK应用

4.3.1. 接入Spring Boot/Spring Cloud应用

将Spring Boot/Spring Cloud应用接入AHAS应用防护后，可以对其配置流控、隔离、熔断、系统或热点规则来保证系统稳定性。本文介绍如何使用SDK方式将Spring Boot/Spring Cloud应用接入应用防护。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面右上角单击**新应用接入**。
5. 在**JAVA语言**页签，单击**SDK接入**，然后单击**Spring Boot应用接入**。
6. 在Spring Boot应用的Pom文件中引入依赖。

```
<dependency>
  <groupId>com.alibaba.csp</groupId>
  <artifactId>spring-boot-starter-ahas-sentinel-client</artifactId>
  <!-- 可指定版本号，最新版本见AHAS控制台流量防护新应用接入页。-->
  <version>x.y.z</version>
</dependency>
```

在**Spring Boot应用接入**页面**第一步：添加Pom依赖**中查看Pom依赖最新版本，将 `x.y.z` 替换为新版本的版本号。

7. 在应用工程中添加埋点。
   - 添加HTTP埋点：引入spring-boot-starter-ahas-sentinel-client依赖后，应用会自动添加Web接口埋点。

> **说明**
> - 若您从1.5.1之前的版本升级到1.5.1+版本，或额外引入了Web filter等的bean，需要先将之前注册bean的相关代码去掉，否则可能会导致重复统计。
> - 注解方式blockHandler和fallback函数的方法签名有限制，具体信息，请参见注解方式。
> - 1.8.0及以上版本注解方式埋点支持自动重试规则。

- 添加MyBatis SQL埋点：
  - 若您使用了MyBatis Spring Boot Starter，则引入AHAS依赖后会自动识别DAO埋点（需要spring-boot-starter-ahas-sentinel-client 1.5.1及以上版本）。
  - 若您未使用MyBatis Spring Boot Starter，则需在MyBatis应用的XML配置文件中引入SentinelMyBatisMapperInterceptor拦截器依赖。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <plugins>
      <!-- 引入AHAS Sentinel拦截器。 -->
        <plugin interceptor="com.alibaba.csp.sentinel.adapter.mybatis.SentinelMyBatisMa
pperInterceptor"/>
    </plugins>
</configuration>
```

- 添加普通接口埋点（注解方式）：
  a. 引入spring-boot-starter-aop。

  > **说明** 若您的工程中已引入此依赖，则跳过此步骤。

  b. 在业务方法使用注解作为埋点。
  若不配置BlockHandler，则被流控降级时方法会直接抛出BlockException，若方法未定义Throws BlockException则会被JVM包装一层UndeclaredThrowableException。BlockHandler和Fallback 函数的方法签名有限制，详情请参见配置触发规则后的逻辑。

```
@SentinelResource(value = "getUserById")
public User getUserById(String id) {
  return new User(id);
}
```

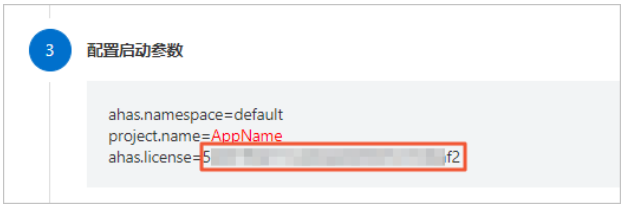
- 添加Feign埋点：引入Starter依赖后，在*application.properties*文件中配置 `feign.sentinel.enabled` 即可。Feign埋点的资源名格式为 `feign:${httpMethod}:${url}` ，例如 `feign:http://localhost:8088/hello` 。

  > **说明** spring-boot-starter-ahas-sentinel-client 1.8.4及以上版本支持。

8. 通过以下任意一种方式，配置应用的启动参数。
   - 添加JVM -D参数。

     非公网 公网

- 在*application.properties*文件中添加以下内容：
  若在公网地域，需要查看License信息。请在**第三步：配置启动参数**区域查看（非公网地域不需要），具体请参见查看License。

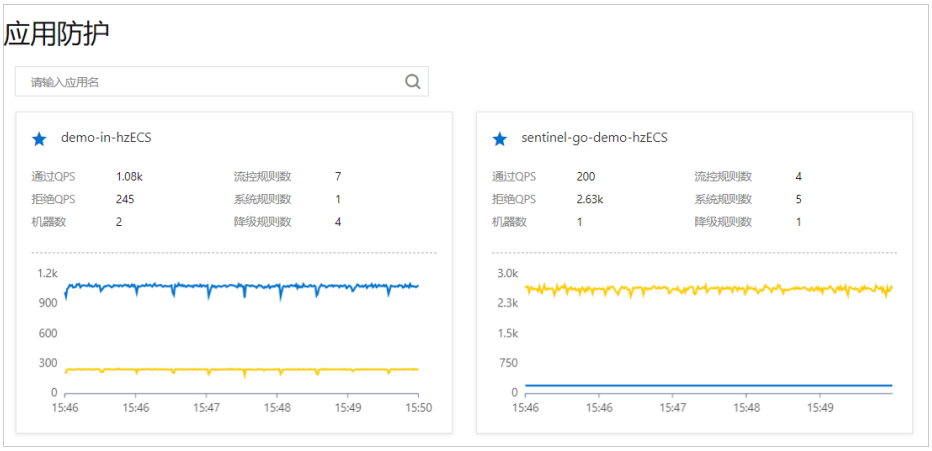


9. （可选）您可以自定义Spring Boot应用触发限流、降级或系统保护规则时的处理逻辑。
   - 若添加HTTP埋点，则使用Web Servlet Filter方式配置处理逻辑。具体操作，请参见Web Servlet Filter。
   - 若添加自定义埋点，则使用注解方式配置处理逻辑。具体操作，请参见注解方式。

> **说明** 若未执行此步骤，当Web接口触发流控降级规则时，返回默认的提示信息（状态码为429）；注解方式接口默认抛出BlockException异常类的子类（触发流控规则，则抛出流控异常FlowException；触发降级规则，则抛出降级异常DegradeException），若方法未定义throws BlockException则会被JVM包装一层UndeclaredThrowableException。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



4.3.2. 接入Spring应用

通过在业务逻辑上添加依赖注解的方式将Spring应用接入应用防护，可以对调用方法进行注解埋点，减小对代码的入侵。本文将介绍如何将Spring应用接入应用防护。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面右上角单击**新应用接入**，然后在**JAVA语言**页签，单击**SDK接入**，然后单击**Spring应用接入**。

5. 通过以下任意一种方式，为应用添加依赖。
   - 方式一：在Pom文件中添加依赖。

     ```
     <dependency>
       <groupId>com.alibaba.csp</groupId>
       <artifactId>ahas-sentinel-client</artifactId>
       <version>x.y.x</version>
     </dependency>
     ```

     在**Spring应用接入**页面**第一步：添加Pom依赖**中查看Pom依赖最新版本，将 `x.y.z` 替换为新版本的版本号。

   - 方式二：添加JAR包依赖。
     在**Spring应用接入**页面单击**请点击此链接下载**下载压缩包，并将压缩包中的所有JAR包解压后放置在*classpath*目录下。
6. 在应用工程中添加埋点。
   - 添加HTTP埋点。

     ```
     @Configuration
     public class SentinelWebConfig implements WebMvcConfigurer {
       @Override
       public void addInterceptors(InterceptorRegistry registry) {
         registry.addInterceptor(new SentinelWebInterceptor());
       }
     }
     ```

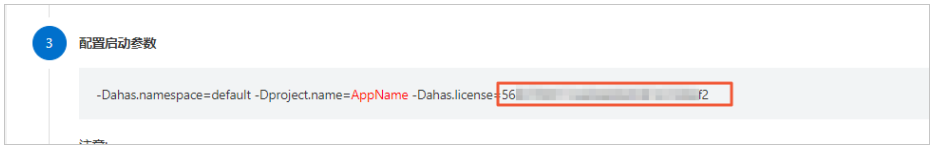
   - 普通接口埋点。
     a. 将SentinelResourceAspect注册为一个Spring Bean。

     ```
     @Configuration
     public class SentinelAspectConfiguration {
       @Bean
       public SentinelResourceAspect sentinelResourceAspect() {
         return new SentinelResourceAspect();
       }
     }
     ```

     b. 在业务方法上添加@SentinelResource注解。

     ```
     // 原本的业务方法
     @SentinelResource(value = "getUserById")
     public User getUserById(String id) {
       return new User(id);
     }
     ```

7. 通过以下任意一种方式，配置应用的启动参数。
   若在公网地域，需要查看License信息。请在**第三步：配置启动参数**区域查看（非公网地域不需要），具体请参见查看License。

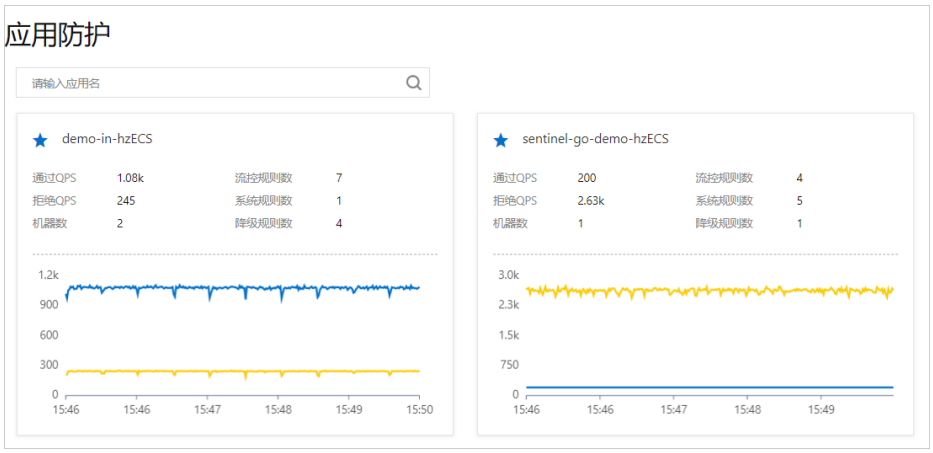


8. 重启应用。
9. （可选）使用注解方式配置应用触发限流、降级或系统保护规则时的处理逻辑。 请参见注解方式。

> **说明** 若未执行此步骤，当应用触发流控降级规则时，默认抛出 `BlockException` 异常类的子类（触发流控规则，则抛出流控异常 `FlowException` ；触发降级规则，则抛出降级异常 `DegradeException` ）。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



4.3.3. 接入Dubbo应用

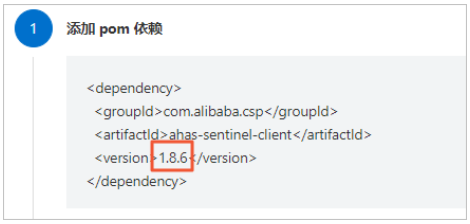
将Dubbo应用接入AHAS应用防护后，可以对其配置流控、降级和系统规则来保证系统稳定性。本文介绍如何使用SDK方式将Dubbo应用接入应用防护。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面右上角单击**新应用接入**。
5. 在**JAVA语言**页签，单击**SDK接入**，然后单击**Dubbo应用接入**。
6. 选择以下任意一种方式，在Dubbo应用中添加应用防护依赖。
   - 在Dubbo应用的pom.xml中添加以下依赖（自SDK/Starter 1.8.6版本起，支持Dubbo 2.6.x/2.7.x主流版本）：

```
<dependency>
  <groupId>com.alibaba.csp</groupId>
  <artifactId>ahas-sentinel-client</artifactId>
  <!-- 可指定版本号，最新版本见AHAS控制台应用防护新应用接入页引导。 -->
  <version>x.y.z</version>
</dependency>
```

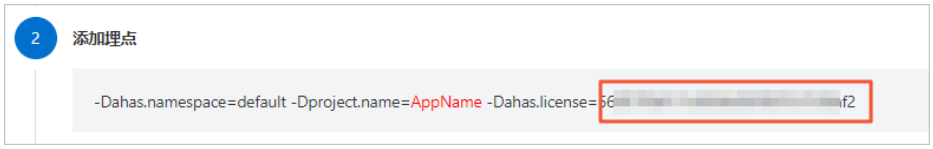
在**Dubbo应用接入**页面查看pom依赖最新版本，将 `x.y.z` 替换为新版本的版本号。



> **说明** 若Dubbo服务是Spring Boot应用，可通过Spring Boot方式快速接入。具体详情，请参见接入Spring Boot/Spring Cloud应用。

- 添加JAR包依赖。
  在**Dubbo应用接入**页面单击**请点此链接下载**下载压缩包，并将压缩包中的所有JAR包放置在*classpath*目录下。

7. 配置应用的启动参数。
   若在公网地域，需要查看License信息。请在**第二步：添加埋点**区域查看（非公网地域不需要），具体请参见查看License。



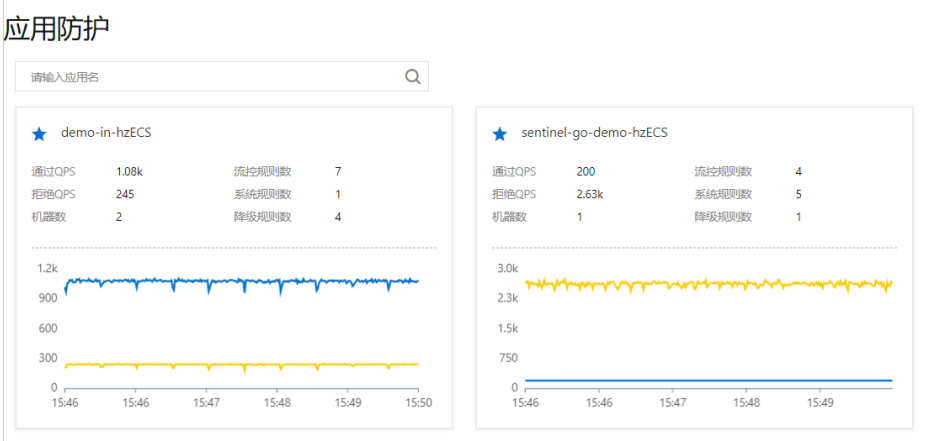
非公网 公网

8. 重启您的应用。
9. （可选）您可以自定义Dubbo应用触发限流、降级或系统保护规则时的fallback处理逻辑，自定义 `DubboFallback` 接口并通过 `DubboFallbackRegistry` 注册即可。配置后，当Dubbo应用触发流控、降级或系统规则时，AHAS会将 `BlockException` 包装后抛出。更多信息，请参见Dubbo Adapter。

> **说明** 若未执行此步骤，当Dubbo应用触发应用防护规则时，默认抛出 `BlockException` 异常类的子类（触发流控规则，则抛出流控异常 `FlowException` ；触发降级规则，则抛出降级异常 `DegradeException` ）。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



4.3.4. 接入HSF应用

将HSF（高速服务框架，High-speed Service Framework）应用接入AHAS应用防护后，可以对其配置流控、降级和系统规则来保证系统稳定性。本文介绍如何使用SDK方式将HSF应用接入应用防护。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面右上角单击**新应用接入**。
5. 在**JAVA语言**页签，单击**SDK接入**，然后单击**Dubbo/HSF应用接入**。
6. 选择以下任意一种方式，在HSF应用中添加应用防护依赖。
   - 若是Spring Boot应用，则可以通过starter方式接入。在pom.xml中引入以下依赖：

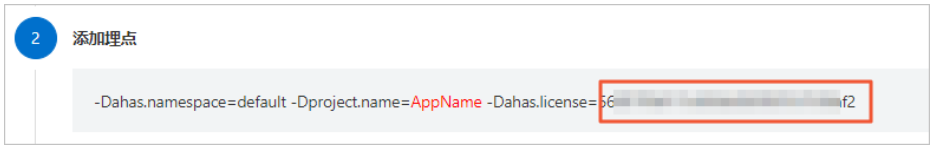
     ```
     <dependency>
       <groupId>com.alibaba.csp</groupId>
       <artifactId>spring-boot-starter-ahas-sentinel-client</artifactId>
       <!-- 需指定版本号，需要≥1.7.2版本。最新版本见AHAS控制台应用防护新应用接入页引导。 -->
       <version>x.y.z</version>
     </dependency>
     ```

   - 若非Spring Boot应用，则在pom.xml中引入以下依赖：

     ```
     <dependency>
       <groupId>com.alibaba.csp</groupId>
       <artifactId>ahas-sentinel-client</artifactId>
       <!-- 需指定版本号，需要≥1.7.2 版本。最新版本见AHAS控制台应用防护新应用接入页引导。 -->
       <version>x.y.z</version>
     </dependency>
     ```

     在**Dubbo/HSF接入**页面**第一步：添加Pom依赖**中查看Pom依赖最新版本，将 `x.y.z` 替换为新版本的版本号。

7. 配置应用的启动参数。
   若在公网地域，需要查看License信息。请在**第二步：添加埋点**区域查看（非公网地域不需要），具体请参见查看License。



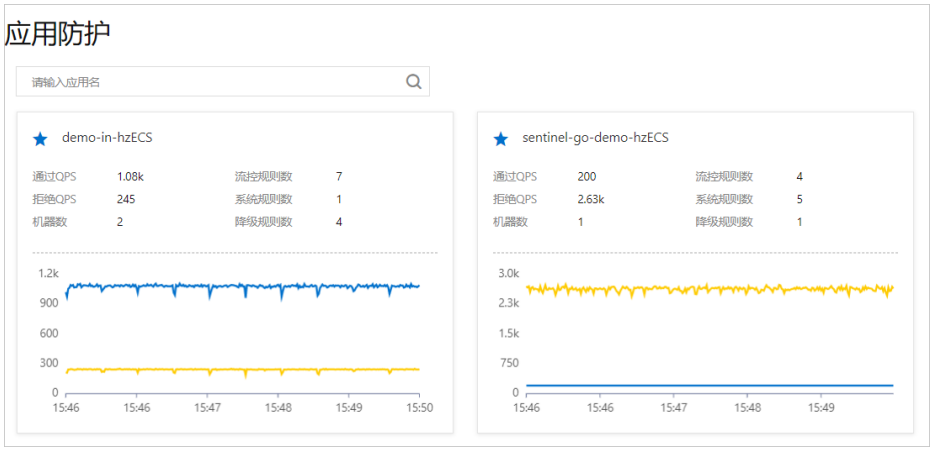
非公网 公网

8. 重启您的应用。
9. （可选）您可以自定义HSF应用触发限流、降级或系统保护规则时的fallback处理逻辑，自定义HsfFallback接口并通过 `HsfFallbackRegistry` 注册即可（ `setProviderFallback` 针对服务提供方， `setConsumerFallback` 针对服务消费方）。完成配置后，当HSF应用触发流控、降级或系统规则时，AHAS会将 `BlockException` 携带到fallback中。

> **说明** 若未执行此步骤，当HSF应用触发应用防护规则时，默认抛出固定的异常 `RuntimeException("SentinelBlockException")` 。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



4.3.5. 接入Web应用

将Web应用接入AHAS应用防护后，可以对其配置流控、降级和系统规则来保证系统稳定性。本文介绍如何使用SDK方式将Web应用接入应用防护。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面右上角单击**新应用接入**，然后在**JAVA语言**页签，单击**SDK接入**，单击**Web应用接入**。
5. 选择以下任意一种方式，在Web应用中添加应用防护依赖。
   - 在Web应用的Pom文件中添加以下依赖。

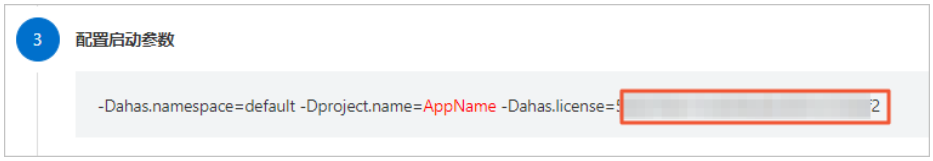
     ```
     <dependency>
       <groupId>com.alibaba.csp</groupId>
       <artifactId>ahas-sentinel-client</artifactId>
       <version>x.y.z</version>
     </dependency>
     ```

     在**Web应用接入**页面**第一步：添加Pom依赖**中查看Pom依赖最新版本，将`x.y.z`替换为新版本的版本号。

   - 添加JAR包依赖。
     在**Web应用接入**页面单击**请点此链接下载**下载压缩包，并将压缩包中的所有JAR包放置在*classpath*目录下。
6. 在web.xml文件中引入拦截器，并将其配置为第一个Filter，示例如下。

   ```
   <filter>
       <filter-name>SentinelCommonFilter</filter-name>
       <filter-class>com.alibaba.csp.sentinel.adapter.servlet.CommonFilter</filter-class>
   </filter>
   <filter-mapping>
       <filter-name>SentinelCommonFilter</filter-name>
   <!-- 配置要拦截的URL模式。 -->
       <url-pattern>/*</url-pattern>
   </filter-mapping>
   ```

7. 配置应用的启动参数。
   若在公网地域，需要查看License信息。请在**第三步：配置启动参数**区域查看（非公网地域不需要），具体请参见查看License。

   非公网 公网
8. （可选）您可以通过以下两种方式来设定自定义的跳转URL，当请求触发流控、降级或系统规则时会自动跳转至设定的URL。

- 通过 `WebServletConfig.setBlockPage(blockPage)` 方法设定自定义的跳转URL。
- 通过UrlBlockHandler接口编写定制化的限流处理逻辑，然后将其注册至WebCallbackManager中来实现，更多信息，请参见Web Servlet Filter扩展接口。

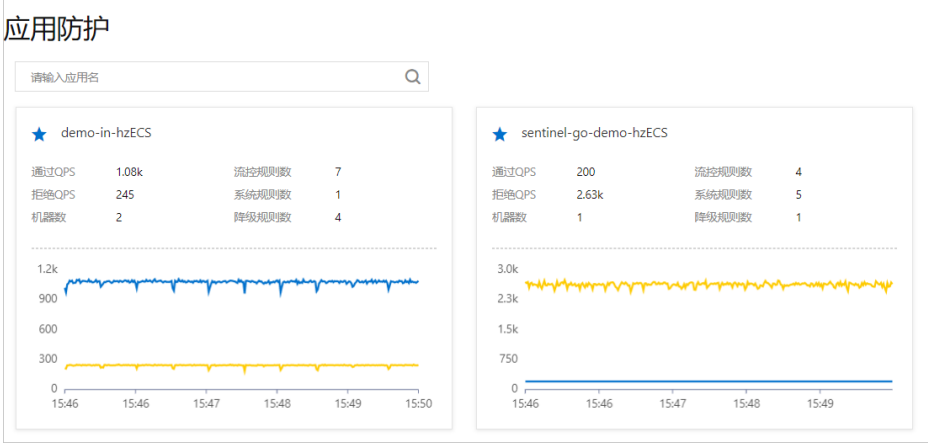
> **说明** 默认情况下，当请求触发流控、降级或系统规则时会返回默认提示页面，提示信息为：`Blocked by Sentinel (flow limiting)` 。

9. （可选）可以自定义URL清理，防止 `/payment/{id}` 这类REST风格的URL堆积过多影响系统性能，配置URL清理的方式有两种：
   - 实现自定义的UrlCleaner接口，自定义URL清理归一处理逻辑，请参见Web Servlet Filter。
   - 通过*properties*文件配置前缀清理。
     将*properties*文件中URL格式更改为：匹配前缀=清理后的资源名称，例如 /payment/=/payment/*。再通过启动参数 `-Dcsp.sentinel.url.clean.config.path` 配置文件路径。

     > **说明** 启动参数中的默认路径为JAR包所在的ahas-sentinel-url-clean.properties文件。启动参数支持classpath形式，例如：`Dcsp.sentinel.url.clean.config.path=classpath:ahas-sentinel-url-clean.properties` 。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



4.3.6. 通过自定义埋点接入

通过自定义Java SDK埋点的方式将应用接入AHAS应用防护，可以更加灵活地对任意代码块进行操作。本文介绍如何通过自定义埋点接入AHAS应用防护。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面右上角单击**新应用接入**，然后在**JAVA语言**页签，单击**SDK接入**，然后单击**自定义埋点**。
5. 选择以下任意一种方式，在应用中添加应用防护依赖。
   - 在应用的Pom文件中添加以下依赖：

```
<dependency>
  <groupId>com.alibaba.csp</groupId>
  <artifactId>ahas-sentinel-client</artifactId>
  <!-- 可指定版本号，最新版本见AHAS控制台应用防护新应用接入页。 -->
  <version>x.y.z</version>
</dependency>
```

在**自定义埋点**页面**第一步：添加Pom依赖**中查看Pom依赖最新版本，将 `x.y.z` 替换为新版本的版本号。

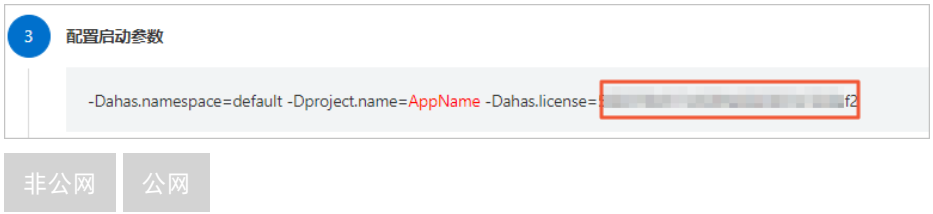
- 添加JAR包依赖。
  在**自定义埋点**页面单击**请点此链接下载**下载压缩包，并将压缩包中的所有JAR包放置在*classpath*目录下。

6. 添加埋点，使用以下代码包住您的业务逻辑。

```
Entry entry = null;
try {
  // HelloWorld为该调用资源名称，控制台上监控、簇点链路会显示该资源。
  // 配置规则的时候也是针对某个资源名称配置。
  // EntryType.IN标识该资源为入口资源，系统规则只针对入口资源生效。
  entry = SphU.entry("HelloWorld", EntryType.IN);
  // 被保护的业务逻辑。
  System.out.println("Hello Sentinel!");
} catch (BlockException e) {
  // 触发流控降级，在此处进行限流处理（如fallback或记录日志）。
} finally {
  // 请确保exit() 逻辑在此处被调用，并且和SphU.entry() 成对出现。
  if (entry != null) {
    entry.exit();
  }
}
```

> **说明**
>
> - `EntryType` 用于区分该资源是入口流量（inbound）还是出口流量（outbound），`EntryType.IN` 代表入口流量，`EntryType.OUT` 代表出口流量。系统规则只对入口资源生效。
> - 控制台展示的监控的应用总流量仅统计 `EntryType.IN` 的流量。
> - SphU相关API文档请参见常用类及其方法。
> - 自定义埋点的资源名称最长不能超过1024个字符。

7. 配置应用的启动参数。
   若在公网地域，需要查看License信息。请在**第三步：配置启动参数**区域查看（非公网地域不需要），具体请参见查看License。

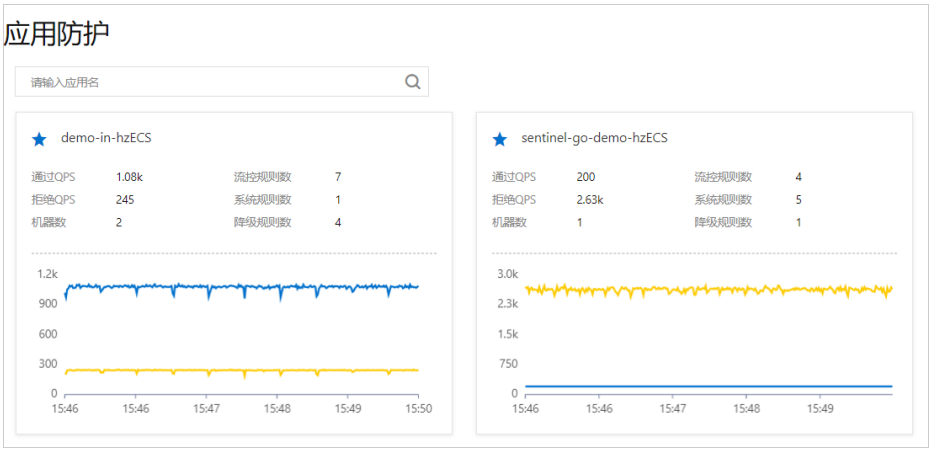


8. 重启应用。
9. （可选）使用注解方式配置应用触发限流、降级或系统保护规则时的处理逻辑。 请参见注解方式。

> **说明** 若未执行此步骤，当应用触发流控降级规则时，默认抛出 `BlockException` 异常类的子类（触发流控规则，则抛出流控异常 `FlowException` ；触发降级规则，则抛出降级异常 `DegradeException` ）。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



4.3.7. 接入MyBatis应用

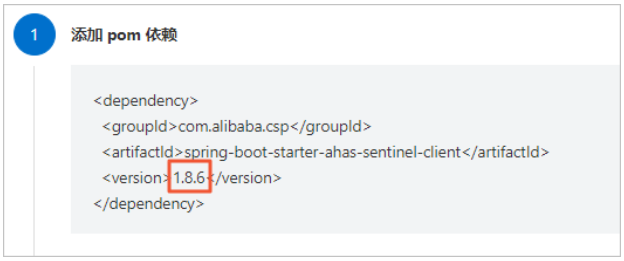
将MyBatis应用接入AHAS应用防护后，可以对其配置流控、降级和系统规则来保证系统稳定性。本文介绍使用SDK方式将MyBatis应用接入应用防护。

操作步骤

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面左上角单击**新应用接入**。
5. 在**JAVA语言**页签，单击**SDK接入**，然后单击**Spring Boot应用接入**。
6. 通过以下任意一种方式，为应用添加依赖。
   - 方式一：在Pom文件中添加依赖：

```
<dependency>
  <groupId>com.alibaba.csp</groupId>
  <artifactId>spring-boot-starter-ahas-sentinel-client</artifactId>
  <!-- 可指定版本号，最新版本见AHAS控制台应用防护应用接入页。 -->
  <version>x.y.z</version>
</dependency>
```

在**Spring Boot应用接入**页面**第一步：添加Pom依赖**中查看Pom依赖最新版本，将 `x.y.z` 替换为新版本的版本号。



- 方式二：添加JAR包依赖。
  a. 下载ahas-sentinel依赖包。
  b. 解压依赖包，并将依赖包中的所有JAR包放置在*classpath*下。

7. 添加拦截器。
   - 若您使用了MyBatis Spring Boot Starter，则引入AHAS依赖后会自动接入（需要spring-boot-starter-ahas-sentinel-client 1.5.1及以上版本）。
   - 若您未使用MyBatis Spring Boot Starter，则需在MyBatis应用的XML配置文件中引入SentinelMyBatisMapperInterceptor拦截器依赖：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <plugins>
      <!-- 引入AHAS Sentinel拦截器 -->
        <plugin interceptor="com.alibaba.csp.sentinel.adapter.mybatis.SentinelMyBatisMapp
erInterceptor"/>
    </plugins>
</configuration>
```

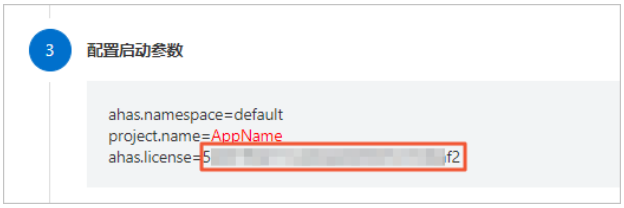
8. 通过以下任意一种方式，配置应用的启动参数。
   - 添加JVM -D参数。

     非公网 公网

   - 在*application.properties*文件中添加以下内容：

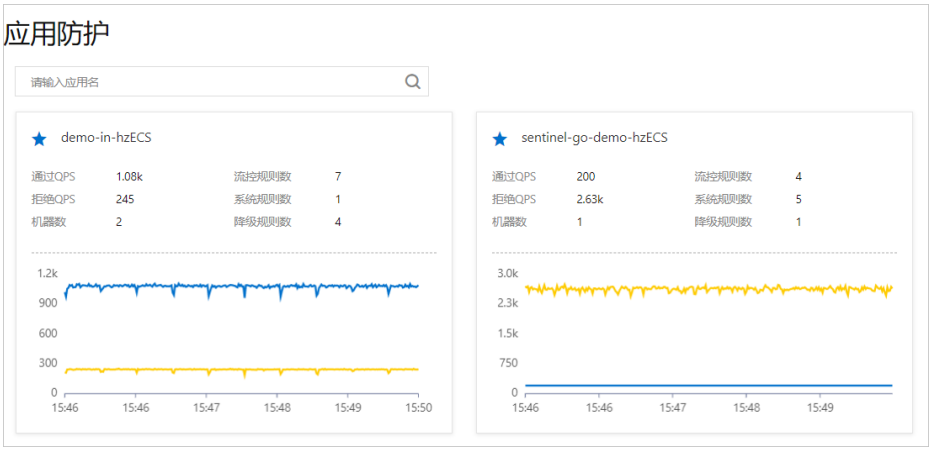
     非公网 公网

若在公网地域，需要查看License信息。请在**Spring Boot应用接入**页面查看（非公网地域不需要），具体请参见查看License。



结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



后续操作

为应用配置流控降级规则请参见以下文档：

- 配置流控规则
- 配置熔断规则
- 自适应流控

当MyBatis应用触发配置的流控、降级或系统保护规则时，会抛出 `MyBatisSentinelBlockException` 类型的异常，您可以自行捕获该异常并进行处理。

4.3.8. 通过开源Sentinel SDK接入

通过开源组件Sentinel，可以便捷地接入AHAS应用防护。

操作步骤

使用开源组件Sentinel将应用接入AHAS应用防护以及连接控制台的具体操作，请参见新手指南。

若您已接入Sentinel控制台，您可以将Pom包中的 `sentinel-transport-simple-http` 模块替换为 `ahas-sentinel-client` 模块，接入AHAS应用防护。

> **注意** 若在本机或非阿里云VPC网络运行，请注意在AHAS控制台左上角选择地域为公网。

4.3.9. 接入Http Client

AHAS的流量防护支持接入Http Client，首先应用通过SDK方式接入，然后将应用中的Http Client调用再接入AHAS控制台，使Http Client可以使用流量防护服务。目前AHAS提供OkHttp及Apache HttpClient（同步模式）框架的适配。本文介绍如何接入Http Client。

前提条件

- 已通过SDK方式在AHAS应用防护中接入应用，具体操作，请参见接入应用方式。
- 如果您要接入的应用是Spring Boot/Spring Cloud，具体操作，请参见接入Spring Boot/Spring Cloud应用。

接入OkHttp

若要将OkHttp接入AHAS流量防护，您只需在创建OkHttpClient的时候通过以下方式注册AHAS Sentinel的Interceptor:

```
OkHttpClient client = new OkHttpClient.Builder()
        .addInterceptor(new SentinelOkHttpInterceptor(new SentinelOkHttpConfig()))
        .build();
```

> **说明** AHAS默认会提取Http Method与URL以 `okhttp:${httpMethod}:${url}` 的模式拼接作为资源名，如 `okhttp:GET:http://localhost:8088/hello` 。

您也可以在创建 `SentinelOkHttpInterceptor` 时在 `SentinelOkHttpConfig` 中自定义解析资源名的逻辑（ `OkHttpResourceExtractor` ），示例如下:

```
OkHttpResourceExtractor extractor = (request, connection) -> {
    String resource = request.url().toString();
    String regex = "/okhttp/back/";
    if (resource.contains(regex)) {
        resource = resource.substring(0, resource.indexOf(regex) + regex.length()) + "{id}";
    }
    return resource;
};
```

默认情况下URL访问被限流后会抛出 `RuntimeException("SentinelBlockException")` 异常。您也可以在创建 `SentinelOkHttpConfig` 时提供自定义的Fallback处理器。Fallback示例如下:

```
public class SomeFallback implements OkHttpFallback {
    @Override
    public Response handle(Request request, Connection connection, BlockException e) {
        //在此处构造限流后的response。
        return new Response(myErrorBuilder);
    }
}
```

接入Apache HttpClient

> **说明** 目前仅支持同步模式的CloseableHttpClient。

若要将Apache HttpClient接入AHAS流量防护，您只需通过以下方式使用 `SentinelApacheHttpClientBuilder` 构造 `CloseableHttpClient` :

```
HttpClientBuilder builder = new SentinelApacheHttpClientBuilder(config); //使用方式同正常的builder，可以传入自定义的配置。
CloseableHttpClient httpClient = builder.build();
```

`SentinelApacheHttpClientBuilder`的使用方式和正常的`HttpClientBuilder`相同，可以传入自定义的Http Client配置。AHAS默认会提取Http Method与URL以`httpclient:${httpMethod}:${url}`的模式拼接作为资源名，如`httpclient:GET:http://localhost:8088/hello`。

您也可以在创建`SentinelApacheHttpClientBuilder`时在`SentinelApacheHttpClientConfig`中自定义解析资源名的逻辑（extractor）。

默认情况下URL访问被限流后会抛出`RuntimeException("SentinelBlockException")`异常。您也可以在创建`SentinelApacheHttpClientConfig`时提供自定义的Fallback处理器。Fallback示例如下：

```
public class MyApacheHttpClientFallback implements ApacheHttpClientFallback {
    @Override
    public CloseableHttpResponse handle(HttpRequestWrapper request, BlockException e) {
        //在此处构造限流后的response。
    }
}
```

相关文档

关于AHAS应用防护接入的相关文档如下：

- 接入Spring Boot/Spring Cloud应用
- 接入Spring应用
- 接入Dubbo应用
- 接入HSF应用
- 接入Web应用
- 通过自定义埋点接入
- 接入MyBatis应用

4.4. 接入Go应用

将Go应用接入AHAS应用防护后，可以对其配置流控、隔离和系统规则来保证系统稳定性。本文介绍如何使用SDK方式将Go应用接入应用防护。

前提条件

确保Go应用的版本为1.13.0或以上，并通过Go Modules管理依赖。

Dubbo、Gin Web、gRPC、Mrcio应用接入步骤

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面右上角单击**新应用接入**，然后在**GO语言**页签下，选择**SDK接入**。
4. 单击**下载AHAS Go SDK**，然后将安装包解压到本地。
5. 打开您的Go工程，并在*go.mod*文件中添加以下依赖。

> **说明**
>
> - */path/to/aliyun-ahas-go-sdk*是示例路径，请替换为Go SDK的解压路径。
> - 请确保您的Go版本≥1.13.0，并开启Go modules支持。

```
require github.com/aliyun/aliyun-ahas-go-sdk v1.0.0
#将后面的路径替换为AHAS Go SDK解压后的本地路径。
replace github.com/aliyun/aliyun-ahas-go-sdk => /path/to/aliyun-ahas-go-sdk
```

6. 添加AHAS Sentinel初始化命令，并在应用启动时生效。

```
import (
    ahas "github.com/aliyun/aliyun-ahas-go-sdk"
)
// 在应用的初始化逻辑中加入以下代码。
// Sentinel core的初始化包含在了这里面。如果之前有调用过Sentinel的初始化函数，需要去掉。
err := ahas.InitAhasDefault()
if err != nil {
    log.Fatalf("Failed to init AHAS: %+v", err)
}
```

7. 使用以下代码包住您的业务逻辑。示例代码如下。

> **说明**
>
> - Dubbo应用是在代码中通过import包的形式引入Dubbo adapter，其中的init()函数会自动注入相关filter。Dubbo-Go版本需要≥1.3.0。Sentinel Dubbo adapter会自动统计所有provider和consumer的调用。
> - Gin Web应用在Gin的初始化代码中引入SentinelMiddleware。Sentinel会对每个API route进行统计，资源名称类似于GET:/foo/:id。默认的限流处理逻辑是返回 `429 (Too Many Requests)` 错误码。
> - gRPC应用在gRPC的初始化代码中引入Sentinel提供的interceptor，Sentinel针对Server和Client都提供了unary和streaming两种interceptor，以上代码以Server端为例。默认的限流处理逻辑是返回Sentinel的BlockError。您也可以在创建interceptor时提供自定义的fallback处理逻辑。
> - 在Go-Micro的初始化代码中引入Sentinel提供的wrapper。Sentinel针对Go-Micro Server和Client都提供了wrapper。以上代码以Server端为例。埋点默认会提取服务method作为资源名，默认的流控处理逻辑是返回Sentinel的BlockError。您也可以在创建wrapper时提供自定义的fallback处理逻辑。

Dubbo应用接入 | Gin Web应用接入 | gRPC应用接入 | Micro应用接入

8. 配置AHAS，在项目目录下新建sentinel.yml文件，配置文件内容如下。

   若在公网地域，需要查看License信息。请在**GO语言**页签查看（非公网地域不需要），具体请参见查看License。

非公网 | 公网

9. 重启您的应用。

```
version: "v1"
sentinel:
  app:
    # 应用名，会显示在 AHAS 控制台，必须指定。
  # AppName 只能包含字母、数字、以及以下特殊字符 _ - . :
    name: YourAppName
```

自定义埋点接入步骤

通过自定义埋点接入Go语言的步骤如下。

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面右上角单击**新应用接入**，然后在**GO语言**页签下，选择**SDK接入**。
4. 单击**下载AHAS Go SDK**，然后将安装包解压到本地。
5. 打开您的Go工程，并在*go.mod*文件中添加以下依赖。

   > **说明** */path/to/aliyun-ahas-go-sdk*是示例路径，请替换为Go SDK的解压路径。

   ```
   require github.com/aliyun/aliyun-ahas-go-sdk v1.0.0
   replace github.com/aliyun/aliyun-ahas-go-sdk => /path/to/aliyun-ahas-go-sdk
   ```

6. 添加import依赖。

   ```
   import (
     ahas "github.com/aliyun/aliyun-ahas-go-sdk"
     sentinel "github.com/alibaba/sentinel-golang/api"
   )
   ```

7. 添加AHAS Sentinel初始化命令，并在应用启动时生效。

   ```
   // 在应用的初始化逻辑中加入以下代码
   // Sentinel core的初始化包含在了这里面。如果之前有调用过Sentinel的初始化函数，需要去掉。
   err:= ahas.InitAhasDefault()
   if err!= nil{
     log.Fatalf("Failed to init AHAS:%+v", err)
   }
   ```

8. 使用以下代码包住您的业务逻辑，埋点方法及示例请参见Sentinel Go API。

   ```
   import (
       sentinel "github.com/alibaba/sentinel-golang/api"
   )
   // Entry方法用于埋点。
   e, b := sentinel.Entry("your-resource-name", sentinel.WithTrafficType(base.Inbound))
   if b != nil {
       // 请求被流控，可以从BlockError中获取限流详情。
   } else {
       // 请求可以通过，在此处编写您的业务逻辑。
       // 务必保证业务逻辑结束后Exit。
       e.Exit()
   }
   ```

9. 使用以下任意一种方式配置AHAS Sentinel。

- YAML文件配置。
  AHAS支持的YAML配置项如下：

> 说明
> - 若通过 `InitAhasFromFile(path)` 初始化AHAS，则会从path指定的YAML文件中读取配置项。
> - 若通过 `InitAhasDefault()` 初始化AHAS，则会尝试从SENTINEL_CONFIG_FILE_PATH环境变量读取path并读取对应的文件。若未指定则默认从项目目录下的sentinel.yml文件读取配置。
> - 若均不存在，AHAS则会通过环境变量读取基础的配置（如项目名称），其他配置项采用默认值。注意环境变量配置会覆盖配置文件的配置。

Key	对应的YAML配置项	含义	备注
`SENTINEL_APP_NAME`	sentinel.app.name	项目名称	必需的配置项，若不配置则标为 `unknown_go_service` 。
`SENTINEL_CONFIG_FILE_PATH`	无	YAML配置文件路径	若未指定则默认从项目目录下的 `sentinel.yml` 文件读取配置。
`AHAS_LICENSE`	ahas.license	AHAS License	公网环境必需，Region环境不需要。
`AHAS_NAMESPACE`	ahas.namespace	AHAS命名空间	默认为Default。需在控制台预先添加后配置，不可随便指定。
`SENTINEL_LOG_DIR`	无	日志路径	默认路径为 `~/logs/csp` 。
`SENTINEL_LOG_USE_PID`	无	日志文件是否带Pid	默认为False。

配置文件示例：

非公网 公网

```
sentinel:
  app:
    # 应用名，会显示在AHAS控制台，必须指定。
name: sentinel-go-service
```

配置环境变量。

非公网 公网

```
SENTINEL_APP_NAME=应用名
```

Sentinel Go API

使用Sentinel的Entry API将业务逻辑封装起来，这一步称为**埋点**。每个埋点都有一个资源名称（resource），代表触发了这个资源的调用或访问。

埋点API位于API包中，一般为 `Entry(resource string, opts ...Option) (*base.SentinelEntry, *base.BlockError)` ，其中resource代表埋点资源名，opts代表埋点配置。目前支持以下埋点配置：

- WithTrafficType(entryType base.TrafficType)：标记该埋点资源的流量类型，其中Inbound代表入口流量，Outbound代表出口流量。若不指定，默认为Outbound。
- WithResourceType(resourceType base.ResourceType)：标记该埋点资源的分类。
- WithAcquireCount(acquireCount uint32)：标记每次触发该埋点计为几次调用（可以理解为batch count）。若不指定，默认为1。
- WithArgs(args ...interface{})：埋点携带的参数列表，为热点参数统计预留。

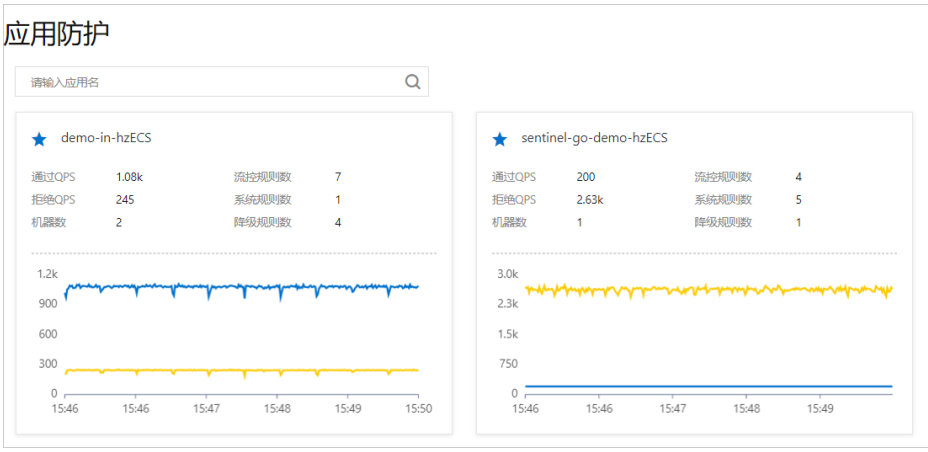
埋点API示例。

```
import (
    sentinel "github.com/alibaba/sentinel-golang/api"
)
// Entry方法用于埋点。
e, b := sentinel.Entry("your-resource-name", sentinel.WithTrafficType(base.Inbound))
if b != nil {
    // 请求被流控，可以从BlockError中获取限流详情。
} else {
    // 请求可以通过，在此处编写您的业务逻辑。
    // 务必保证业务逻辑结束后Exit。
e.Exit()
}
```

说明 若该次调用被拒绝，则Entry API会返回BlockError代表被Sentinel限流。BlockError提供了限流原因以及触发的规则等信息，可以方便开发者获取相关信息进行记录和处理。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。



相关操作

给Go SDK应用配置规则，目前支持流控规则（暂不支持集群流控）、隔离规则、降级规则、系统规则和热点规则。

- 配置流控规则

- 配置隔离规则
- 配置熔断规则
- 自适应流控
- 配置热点规则

4.5. 接入PHP应用

将PHP应用接入AHAS应用防护后，可以对其配置流控、降级、隔离和系统规则来保证系统稳定性。本文介绍如何使用SDK方式将PHP应用接入应用防护。

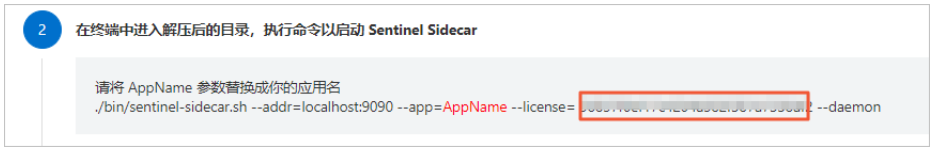
前提条件

目前仅支持Linux 64位环境，请确保您的PHP版本≥7.0或者≥5.5并通过composer管理依赖。

PHP应用接入步骤

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面右上角单击**新应用接入**，然后单击**PHP语言**，选择**SDK接入**。
4. 在**自定义埋点**单击**下载AHAS Sentinel Sidecar**，然后将压缩包解压到本地。
5. 进入压缩包解压后的路径，然后执行以下命令启动Sentinel Sidecar。

   若在公网地域，需要查看License信息。请在**PHP语言**页签查看（非公网地域不需要），具体请参见查看License。

   停止应用后，可执行以下命令退出Sentinel Sidecar:

   ```
   ./bin/sentinel-sidecar.sh --quit
   ```

   非公网 公网

   ```
   //将AppName替换为自定义的应用名称。
   ./bin/sentinel-sidecar.sh --addr=localhost:9090 --app=AppName --daemon
   ```

6. 在您的PHP项目中添加AHAS Sentinel依赖。

   ```
   composer require ahas/sentinel "^0"
   ```

   > **说明** 通常需要在PHP源码文件头部引入composer autoload文件，以支持自动加载PHP类。

7. 在代码中创建SentinelClient实例。

   ```
   // 创建AHAS Sentinel客户端，本地sidecar默认监听9090端口。
   sentinel = new \Sentinel\SentinelClient("localhost", 9090);
   ```

8. 使用以下代码包住您的业务逻辑。

```
try {
  // 获取受保护的资源访问入口，参数为埋点资源名称。
  // 注意：必须定义一个变量保存入口对象，否则入口对象将被自动销毁并释放资源。
  sentinelEntry = $sentinel->entry("your-resource-name");
  // 在此处编写受保护的业务逻辑。
} catch (\Sentinel\BlockException $e) {
  // 被Sentinel流控，在此处编写流控处理逻辑，如提示用户请求过多。
} finally {
    // 将保存入口对象的变量置为null，使不再有变量引用入口对象，以销毁入口对象并释放资源。
    sentinelEntry = null;
}
```

9. 重启您的PHP应用。

结果验证

登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**，若在**应用防护**页面出现该应用的资源卡片且有数据上报，则说明接入成功。

相关操作

PHP SDK目前支持流控规则、降级规则、隔离规则和系统规则：

- 配置流控规则
- 配置熔断规则
- 配置隔离规则
- 自适应流控

5.应用大盘

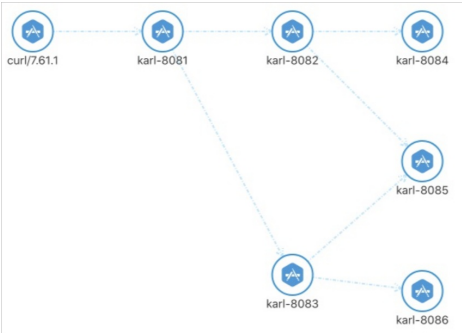
应用大盘用于展示具有调用关系的应用，在调用拓扑结构上透出应用概览信息，如节点资源，应用防护信息，流量防护事件等，方便您及时感知系统情况、调用关系、应用内部问题或者告警等，帮助您采取必要的问题解决措施。本文以应用通过Agent方式接入AHAS介绍如何展示应用大盘。

前提条件

- 开通PTS，更多信息，请参见什么是性能测试PTS。
- 应用需要以Agent或K8s方式接入AHAS。具体操作，请参见通过Java Agent接入。

应用大盘展示

以应用通过Agent方式接入AHAS为例，为您介绍如何查看应用大盘。当前接入的应用间实际调用拓扑关系如下：



1. 执行以下代码，设置启动命令。

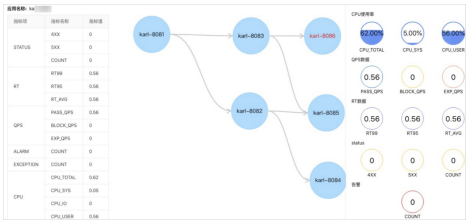
```
java -Dproject.name=karl-8084 -Dserver.port=8084 -Drequest.port=none -Drequest.interface=he
llo -Dahas.license=6523c21873cc41d5bexxxxxxxxxxde6 -Dahas.regionId=cn-shenzhen -javaagent:a
has-java-agent.jar -jar karl-demo.jar &
java -Dproject.name=karl-8085 -Dserver.port=8085 -Drequest.port=none -Drequest.interface=he
llo -Dahas.license=6523c21873cc41d5bexxxxxxxxxxde6 -Dahas.regionId=cn-shenzhen -javaagent:a
has-java-agent.jar -jar karl-demo.jar &
java -Dproject.name=karl-8086 -Dserver.port=8086 -Drequest.port=none -Drequest.interface=he
llo -Dahas.license=6523c21873cc41d5bexxxxxxxxxxde6 -Dahas.regionId=cn-shenzhen -javaagent:a
has-java-agent.jar -jar karl-demo.jar &
java -Dproject.name=karl-8082 -Dserver.port=8082 -Drequest.port=8084,8085 -Drequest.interfa
ce=test -Dahas.license=6523c21873cc41d5bexxxxxxxxxxde6  -Dahas.regionId=cn-shenzhen -javaag
ent:ahas-java-agent.jar -jar karl-demo.jar &
java -Dproject.name=karl-8083 -Dserver.port=8083 -Drequest.port=8086,8085 -Drequest.interfa
ce=test -Dahas.license=6523c21873cc41d5bexxxxxxxxxxde6  -Dahas.regionId=cn-shenzhen -javaag
ent:ahas-java-agent.jar -jar karl-demo.jar &
java -Dproject.name=karl-8081 -Dserver.port=8081 -Drequest.port=8082,8083 -Drequest.interfa
ce=test -Dahas.license=6523c21873cc41d5bexxxxxxxxxxde6 -Dahas.regionId=cn-shenzhen -javaage
nt:ahas-java-agent.jar -jar karl-demo.jar &
```

> **说明**
>
> - 其中 `Dahas.regionId` 和 `Dahas.license` 参数需要您根据实际情况自定义，且 `Dahas.license` 参数需要设置正确，不然会无法采集应用间的调用关系，从而无法显示调用链。
> - 您可以单击下载demo地址。

2. 测试脚本。

```
#!/bin/bash
for ((;;))
    do
        curl localhost:8081/test &
        sleep 2
    done
```

3. 查看应用大盘。
   i. 登录AHAS控制台。
   ii. 在AHAS控制台左上角，选择应用接入的地域。
   iii. 在左侧导航栏中选择**流量防护 > 应用防护**。
   iv. 在**应用防护**页面单击**应用大盘**页签，查看应用大盘，当前应用大盘的展示示例如下：



> **说明** 调用链展示与实际调用存在10s左右延时，若应用下线，调用关系仅会保留两天。为了保障调用关系的实时性，您在查看应用大盘时需要每10s刷新一次**应用大盘**页面。

6.配置规则

6.1. 配置流控规则

配置流控规则的原理是监控应用或服务流量的QPS指标，当指标达到设定的阈值时立即拦截流量，避免应用被瞬时的流量高峰冲垮，从而保障应用高可用性。本文介绍如何配置管理流控规则，以及三种常用场景的流控配置规则。

前提条件

- 开通企业版。相关内容，请参见：微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心
  - ECS微服务应用接入MSE治理中心

背景信息

流量控制在网络传输中是一个常用的概念，常用于调整网络包的发送数据。系统需处理的请求是随机不可控的，而系统的处理能力是有限的，因此就需要根据系统的处理能力对流量进行控制。

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 选择以下任意一种方法进入**新增规则**页面：
   - 在左侧导航栏单击**应用概览**，然后单击页面下方目标接口**操作**列中的**流控**。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 + 或 ⚙ 的图标，然后在**管理规则**对话框中单击**新增规则**。
   - 在左侧导航栏单击**规则设置**，在**流控规则**页签的左上角单击**新增流控防护规则**。
6. 在**新增规则**对话框中配置规则信息，参数说明请参见更多信息。
7. 单击**新建**。

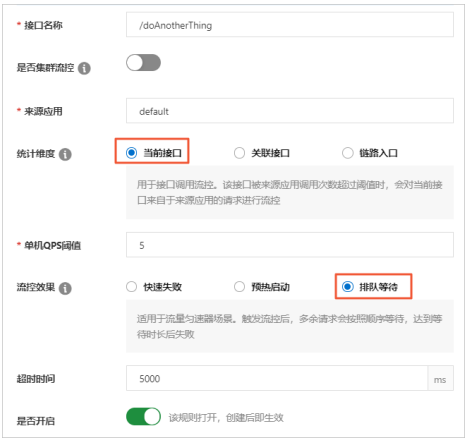
常用场景1：削峰填谷，使流量匀速通过

请求流量具有波峰波谷的特点，流控的原理是将前面的峰值流量延迟（排队时长）到后面再处理，既能最大化满足所有请求，又能保证用户体验。详情请参见削峰填谷。

在**新建流控规则**对话框中配置以下规则信息：

- **统计维度**选择**当前接口**。
- **流控效果**选择**排队等待**。
- 配置匀速模式下请求**单机QPS阈值**为5。
- **等待时长**为5s。

系统则每200 ms处理一条请求，多余的处理任务将排队；同时设置了等待时长为5s，则预计排队时长超过5s的处理任务将快速失败，直接返回默认流控信息，如文本、静态页面等。



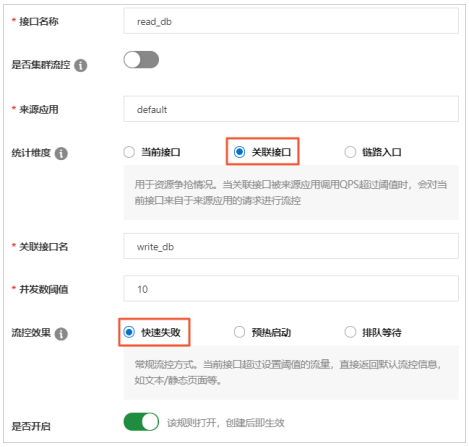
常用场景2：当资源争抢时，需留足资源给优先级高的接口

`read_db` 和 `write_db` 这两个资源分别代表数据库读写。为保证提交的数据不丢失，`write_db` 接口优先级更高。当写库操作过于频繁时，读数据的请求会被限流。详情请参见关联限流。

在**新增规则**对话框中配置以下规则信息：

- **统计维度**选择**关联接口**。
- **流控效果**选择**快速失败**。
- **并发数阈值**为10。

当 `write_db` 资源的QPS超过10之后，`read_db` 会被限流以保证留足资源给 `write_db`，避免 `write_db` 数据丢失。



常用场景3：预热启动避免大流量冲击

流控的原理是在流量入口处控制流量，让通过的流量缓慢增加，在一定时间内逐渐增加到阈值上限，以便系统可以预热。最适合突发流量的场景。详情请参见Warm Up（冷启动）。

在**新增规则**对话框中配置以下规则信息：

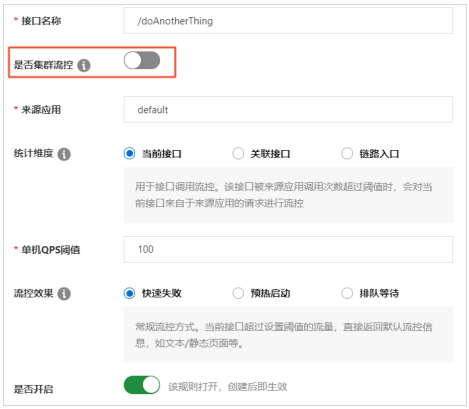
- **统计维度**选择**链路入口**。
- **流控效果**选择**预热启动**。
- **单机QPS阈值**为60。
- **预热时间**为2s。

预热流控方式下，默认会从设置的QPS阈值的1/3开始慢慢往上增加至QPS设置值。本示例中，当入口的QPS超过20（即60÷3）时，会在预热的2s内缓慢增长至60。

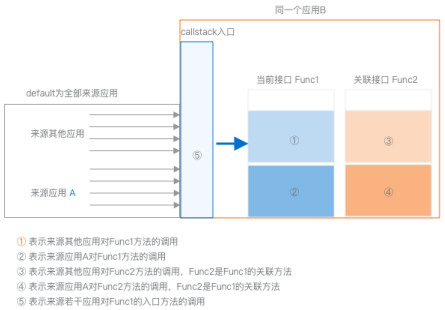
更多信息

新增规则页面参数说明如下：

- 单机模式：关闭集群流控的状态。



参数	描述
接口名称	待流控的资源名称。

参数	描述
来源应用	该规则针对的来源应用，默认来源应用设为 `default` ，代表不区分来源应用。 ○ 若是Dubbo服务，请填写对应调用方的Dubbo applicationName，注意调用方也需要接入Sentinel。 ○ 若是Web服务，请参见扩展接口。 ○ 若是通过注解、自定义埋点，请参见常用类及其方法，ContextUtil只在调用链入口生效，即首个埋点生效。 ○ 当前账号下已经接入了流量防护的其他应用，填写 `default` 即表示不特殊区分。 **注意** 来源数目不宜过多，200个以内内存使用量可控，过多则会影响性能。 
统计维度	选择资源调用关系进行流控。 ○ **当前接口**：直接控制来自**来源应用**中调用来源的访问流量，如果**来源应用**为 `default` 则不区分调用来源。通常应用于流量匀速通过的场景，详情请参见常用场景1。 ○ **关联接口**：控制当前资源的关联资源的流量。通常应用于资源争抢时，留足资源给优先级高接口的场景，详情请参见常用场景2。 ○ **链路入口**：控制该资源所在的调用链路的入口流量。选择**链路入口**后需要继续配置**入口资源**，即该调用链路入口的上下文名称。通常应用于预热启动避免大流量冲击的场景，详情请参见常用场景3。
单机QPS 阈值	触发对流控接口的统计维度对象的QPS阈值。

参数	描述
流控效果	选择流控方式来处理被拦截的流量。 ○ **快速失败**：达到阈值时，立即拦截请求。按照应用系统设置中的适配模块配置信息，进行内容返回。 ○ **预热启动**：需设置具体的预热时间。详情请参见Warm Up（冷启动）。 如果系统在此之前长期处于空闲的状态，当流量突然增大的时候，该方式会让处理请求的速率缓慢增加，经过设置的预热时间以后，到达系统处理请求速率的设定值。默认会从设置的QPS阈值的1/3开始慢慢往上增加至设置的QPS值，多余请求会按照快速失败处理。 ○ **排队等待**：请求匀速通过，允许排队等待，通常用于请求调用削峰填谷等场景。需设置具体的超时时间，达到超时时间后请求会快速失败。 详情请参见削峰填谷。
是否开启	打开开关表示启用该规则，关闭开关表示禁用该规则。开关修改之后会立即生效。

- 集群流控模式：开启集群流控的状态。

参数	描述
是否集群流控	开启集群流控，对集群内此资源的调用总量进行限制。
接口集群总QPS	该接口预估的集群最大QPS，表示最大流量，用于为Token Server自动分配提供参考，当流量超出该值的请求会退化到单机模式。
阈值模式	可选择设置**集群阈值**或**单机QPS阈值**。 ○ **集群阈值**：设置的阈值等同于整个集群的总阈值。 ○ **单机QPS阈值**：设置的阈值等同于单机能够承受的限额，Token Server会根据连接数来计算总的阈值。

参数	描述
失败退化策略	当出现连接失败、通信失败或Token Server不可用等情况时，流控规则是退化到单机限流的模式或是直接通过忽略失败情况。 ○ **退化到单机限流**：当出现失败的情况时，退化到根据的单机阈值来进行流控。需要设置**退化单机阈值**，代表单机的兜底阈值。 ○ **直接通过**：当出现失败的情况时，忽略失败情形，直接通过。

6.2. 配置隔离规则

隔离规则通过控制接口或依赖的并发线程数，来保证系统的稳定性。通常适用于应用内部或下游依赖出现不稳定的场景，例如慢SQL、下游应用响应时间变长等。本文介绍如何配置和管理隔离规则。

前提条件

将应用接入AHAS应用防护，具体操作，请参见接入应用方式。

背景信息

当强依赖的方法或接口出现不稳定的时候，可以通过配置并发线程数来限制不稳定的强依赖并发数，起到隔离异常的效果。若运行该请求的响应时间变长，会导致线程的并发数变大。当并发数超过阈值以后，AHAS将拒绝多余的请求，直到堆积的任务完成，并发线程数变少。达到将异常隔离，减小不稳定性的效果。

如何设定并发线程数阈值，可参见以下内容：

- 并发线程数 = 期望QPS*响应时间+冗余量。
- 例如预期的SQL执行时间为20毫秒，预期该请求每秒有20个，并发最大时候是6个，建议并发线程数按照以下逻辑设置：Max（20/1000*20，6）= 6 ，再加上冗余量2 ，则建议并发数阈值设置为8。
- 设置好后，当这个SQL发生死锁或者有性能问题，SQL运行特别慢成为慢SQL时，即使请求不断的进来，也仅仅会占用8个线程，不会因为持续进来的请求（请求也无法在短时间内退出），从而耗光进程的活跃线程。
- 当这个SQL恢复正常后，并发数会迅速减少。当并发数减少至低于预设的阈值时，系统就不会拒绝请求，应用的处理能力也快速的恢复。通过这样的方式，起到了根据响应时间自动调节的效果，隔离了不稳定的应用。

隔离规则配置通常用于强依赖隔离场景，详情请参见强依赖隔离。

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 选择以下任意一种方法新建隔离规则：
   - 在左侧导航栏单击**应用概览**，然后单击页面下方目标接口**操作**列中的**隔离**。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 ＋ 图标，在**新建规则**对话框中，选择**新增隔离规则**页签。

> **说明** 如果接口中已存在规则，单击 图标，在**隔离规则**页签中创建隔离规则。

- 在左侧导航栏单击**规则管理**，单击**隔离规则**页签，在**隔离规则**页签，单击**新增隔离规则**。

6. 在**新建隔离规则**对话框中配置规则信息：
   i. 在**选择防护场景**页面，修改接口名称，然后单击**下一步**。
   ii. 在**配置防护规则**页面，配置防护规则，然后单击**下一步**。

   > **说明** 若需对隔离防护规则进行编辑，则直接进入配置防护规则。

   iii. 在**配置限流行为**页面，新增和选择关联规则，然后单击**下一步 > 新增**。
   iv. 在**管理规则**页面，状态栏下单击**开启**。
   v. 在**温馨提示**页面，单击**确定**，开启已配置的防护规则。

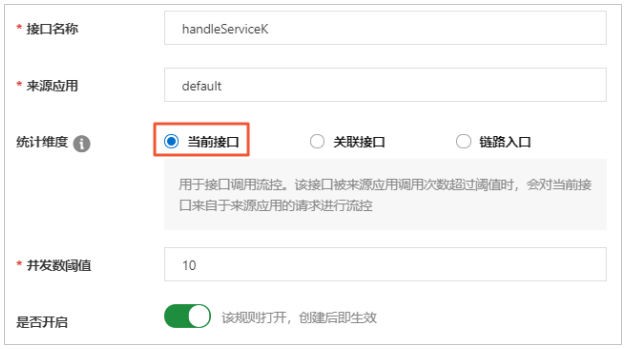
> **说明** 参数说明具体详见更多信息。

常用场景1 保障自身资源充足

当运行该请求的响应时间变长，会导致线程的并发数变大。当并发数超过阈值以后，AHAS将拒绝多余的请求，直到堆积的任务完成，并发线程数变少。达到将异常隔离，减小不稳定性的效果。例如某个SQL执行时间为20毫秒，预期该请求每秒有20个。

在**新建隔离规则**对话框中配置以下规则信息：

- 填写**接口名称**和**来源应用**。
- **统计维度**选择**当前接口**。
- **并发数阈值**为10。



设置完成后，当这个SQL发生死锁或者存在性能问题时，该SQL运行变慢，成为慢SQL，此时即使请求不断进来，也仅仅会占用10个线程，不会因为持续进来的请求（请求也无法在短时间内退出），从而耗光进程的活跃线程。当这个SQL恢复正常后，并发数会迅速减少。当并发数减少至低于预设的阈值时，系统就不会拒绝请求，应用的处理能力也快速的恢复。通过这样的方式，起到了根据响应时间自动调节的效果，隔离了不稳定的应用。

常用场景2 有一定相关联性的接口

当关联接口被来源应用调用QPS超过阈值时，会对当前接口来源应用的请求进行限流，有一定的相关性的方法来配置规则。例如 `read_db` 和 `write_db` 这两个资源分别代表数据库读写，`write_db` 接口优先级更高。

为保证读写资源争抢时，`write_db` 的接口可以留足资源，可在**新建隔离规则**对话框中配置以下规则信息：

- **接口名称**为 `write_db` 。
- **统计维度**选择**关联接口**。
- **关联接口名**为 `read_db` 。
- **并发数阈值**为10。

这样在 `read_db` 接口被调用QPS超过10次后，会对 `write_db` 接口来自于来源应用的请求进行隔离限流，保证 `write_db` 的足够资源。

常用场景3 针对入口链路来配置隔离规则

从入口处将资源进行分别隔离，以保障更高优先级入口。当callstack入口被来源应用调用QPS数超过阈值时，会对当前接口来自于来源应用的请求进行隔离流控。

在**新建隔离规则**对话框中配置以下规则信息：

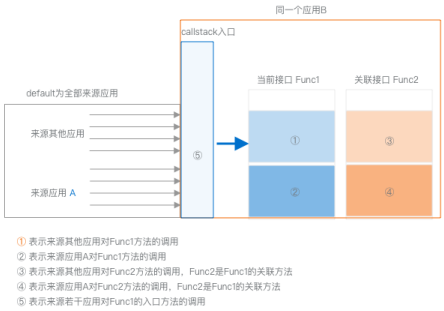
- 填写**接口名称**和**来源应用**。
- **统计维度**选择**链路入口**。
- **并发数阈值**设置为10。

当**callstack入口**的接口被调用超过10次，当前接口 `user_test` 会对来自于来源应用的请求进行隔离流控。

更多信息

新建隔离规则页面参数解释如下：

参数	描述
接口名称	待隔离的资源名称。

参数	描述
来源应用	该规则针对的来源应用，默认来源应用设为 `default`，表示不区分来源应用。  • 若是Dubbo服务，请填写对应调用方的Dubbo applicationName，注意调用方也需要接入Sentinel。 • 若是Web服务，请参见扩展接口。 • 若是通过注解、自定义埋点，请参见常用类及其方法，ContextUtil只在调用链入口生效，即首个埋点生效。 • 当前账号下已经接入了流量防护的其他应用，填写 `default` 即表示不特殊区分。 **注意** 来源数目不宜过多，200个以内内存使用量可控，过多则会影响性能。
统计维度	选择资源调用关系进行隔离流控。 • **当前接口**：直接控制来自**来源应用**中调用来源的访问流量，如果**来源应用**为 `default` 则不区分调用来源，通常应用于保障自身资源充足的场景，请参见常用场景1 保障自身资源充足。 • **关联接口**：控制当前资源的关联资源的流量。通常应用于资源争抢时，留足资源给优先级高接口的场景，请参见常用场景2 有一定相关联性的接口。 • **链路入口**：控制该资源所在的调用链路的入口流量。选择**链路入口**后需要继续配置**callstack入口**，即该调用链路入口的上下文名称。通常应用于接口有多入口资源的场景，请参见常用场景3 针对入口链路来配置隔离规则。
并发数阈值	资源的并发线程数（即该资源正在执行的线程数）阈值。

6.3. 配置熔断规则

AHAS的熔断规则可以监控应用内部或者下游依赖的响应时间或异常比例，当达到指定的阈值时立即降低下游依赖的优先级。在指定的时间内，系统不会调用该不稳定的资源，避免应用受到影响，从而保障应用高可用性。当指定时间过后，再重新恢复对该资源的调用。

前提条件

已在AHAS应用防护中接入应用，具体操作请参见接入应用方式。

背景信息

除了流量控制以外，对调用链路中不稳定的方法或者下游依赖进行熔断也是重要措施之一。由于调用关系的复杂性，如果调用链路中的某一环节出现了错误，会导致这个请求失败，甚至会放大不稳定性，导致整个链路无法正常服务。熔断功能会在调用链路中某个方法出现不稳定时（例如某方法出现Timeout或异常比例升高），对这个方法的调用进行限制，让请求快速失败，避免此错误影响整个链路。关于熔断的相关信息，请参见熔断结构图。

熔断规则配置通常用于弱依赖降级场景，更多信息，请参见弱依赖降级。

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 选择以下任意一种方法进入熔断规则的设置页面：
   - 在左侧导航栏单击**应用概览**，然后单击页面下方目标接口**操作**列中的**熔断**。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 ＋ 或 ⚙ 的图标，在**管理规则**对话框中单击目标方案的页签，然后单击**熔断规则**页签，单击**新增熔断规则**。
   - 在左侧导航栏单击**规则设置**，单击目标方案的页签，然后单击**熔断规则**页签，在页面右上角单击**新增熔断规则**。
6. 在**新增熔断规则**或**新增规则**的对话框中配置规则信息。

> **说明** 慢调用比例、统计窗口时长、最小请求数目等配置项需要Java SDK版本≥1.6.0或Java Agent版本≥1.7.5。

常用场景1：慢调用熔断示例

例如调用第三方服务，但响应时间太慢，会影响当前接口，所以对其进行熔断操作。

在**新增熔断规则**或**新增规则**对话框中配置以下示例规则信息。

参数	示例值	描述
接口名称	test	接口名称。
统计窗口时长	1	统计时长为1秒。
阈值类型	慢调用比例	选择以慢调用比例作为阈值。
慢调用RT	1000	超过1000 ms则判定为慢请求。
降级阈值	80%	触发熔断的慢调用比例阈值为80%。
熔断时长	10	熔断时长有10秒。

参数	示例值	描述
最小请求数目	10	触发熔断的最小请求数目为10。
熔断恢复策略	单次探测恢复	经过熔断时长后，熔断器会对接下来的一个请求进行探测，若该请求符合预期（不为慢调用或没有异常），则结束熔断；否则重新回到熔断阶段。

规则开启后，在统计时长1秒内，当请求数目大于10，并且慢调用的比例大于80%的时候，则在接下来10秒的熔断时长内，请求都会快速失败。经过10秒后熔断器会进入探测恢复状态，若接下来的一个请求响应时间小于设置的1000 ms则结束熔断，若大于1000 ms则会再次被熔断。

常用场景2：异常熔断示例

例如第三方内容展示时，系统会出现异常，当异常比例较高时，可以对其进行熔断操作，以保证更好的用户体验。

在**新增熔断规则**或**新增规则**对话框中配置以下示例规则信息。

参数	示例值	描述
接口名称	test	接口名称。
统计窗口时长	1	统计时长为1秒。
阈值类型	异常比例	选择以异常比例作为阈值。
降级阈值	80%	触发熔断的异常比例阈值为80%。
熔断时长	10	熔断时长有10秒。
最小请求数目	10	触发熔断的最小请求数目为10。

参数	示例值	描述
熔断恢复策略	单次探测恢复	经过熔断时长后，熔断器会对接下来的一个请求进行探测，若该请求符合预期（不为慢调用或没有异常），则结束熔断；否则重新回到熔断阶段。

规则开启后，在统计时长1秒内，当请求数目大于10，并且异常的比例大于80%的时候，则在接下来10秒的熔断时长内，请求都会快速失败。经过10秒后熔断器会进入探测恢复状态，若接下来的一个请求没有异常则结束熔断，否则会再次熔断。

更多信息

新增熔断规则或**新增规则**对话框的参数说明如下。

参数	描述
接口名称	适用该规则的应用资源。
统计窗口时长	统计的时间窗口长度，取值范围为1秒~120分钟。
最小请求数目	触发熔断的最小请求数目，若当前统计窗口内的请求数小于此值，即使达到熔断条件规则也不会触发。

参数	描述
阈值类型	选择以**慢调用比例**或**异常比例**作为阈值。 • 选择以**慢调用比例**作为阈值，需要设置允许的**慢调用RT**（即最大的响应时间），请求的响应时间大于该值则统计为慢调用。 在**降级阈值**中设置触发熔断的慢调用比例。规则开启后，在单位统计时长内请求数目大于设置的**最小请求数目**，并且慢调用的比例大于阈值，则接下来的**熔断时长**内请求会自动被熔断。经过熔断时长后熔断器会进入探测恢复状态，若接下来的一个请求响应时间小于设置的**慢调用RT**则结束熔断，若大于设置的**慢调用RT**则会再次被熔断。 • 选择以**异常比例**作为阈值，需要在**降级阈值**中设置触发熔断的异常比例。 规则开启后，在单位统计时长内业务异常数目大于设置的**最小请求数目**，并且异常的比例大于阈值，则接下来的**熔断时长**内请求会自动被熔断。
熔断时长	即熔断触发后持续的时间。资源进入熔断状态后，在配置的熔断时长内，请求都会快速失败。
熔断恢复策略	熔断器进入恢复阶段（半开启状态）的恢复策略。 • **单次探测恢复**：经过熔断时长后，熔断器会对接下来的一个请求进行探测，若该请求符合预期（不为慢调用或没有异常），则结束熔断；否则重新回到熔断阶段。 • **渐进式恢复**：需要设置**恢复阶段数**和**每步最小通过数目**。 ◦ 经过熔断时长后，熔断器按照设定的恢复阶段数进行渐进式恢复，若该阶段内请求达到一定量即每步最小通过数目，则触发检查。检查的请求若都未超过阈值，则逐步提高允许通过的请求比例，直到请求完全恢复；若某一步的指标超出阈值，则重新回到熔断阶段。 ◦ 请求比例T=100/恢复阶段数N，则第一阶段请求比例为T，第二阶段为2T……直到100%。 ◦ 例如恢复阶段数为3，每步最小通过数目为5，则三个阶段分别按照33%、67%和100%的比例放入请求，当每阶段的请求数目大于等于5时进行检查，若请求的指标未超阈值则进入下一恢复阶段，直至完全恢复。 **说明** 渐进式恢复策略功能需要Java SDK版本≥1.6.2。

6.4. 配置主动降级规则

AHAS主动降级规则可以指定对某些接口进行降级，被降级的接口会触发自定义的降级行为（如返回指定内容）而不会执行原有的逻辑。本文介绍如何新增主动降级规则。

前提条件

将应用接入AHAS应用防护，具体操作，请参见接入应用方式。

注意事项

主动降级规则支持的场景说明如下：

- 主动降级规则仅支持AHAS Sentinel Java SDK或Agent 1.8.4及以上版本。
- 降级规则中的行为配置目前仅对AHAS自带的Web埋点生效（Servlet、Spring Web、Spring Cloud Gateway适配），其它埋点类型仍会按照原有的Fallback逻辑进行处理。更多信息，请参见配置触发规则后的逻辑。
- 若有在代码中注册Block Handler的方式自定义Fallback逻辑，则控制台配置的主动降级规则行为配置不生效。
- 主动降级规则暂不支持其他多语言SDK方式接入。

新增主动降级规则

1. 登录AHAS控制台，然后在页面左上角选择**地域**。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 选择以下任意一种方法进入设置主动降级规则的对话框。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 ＋ 或 ⚙ 的图标，在**管理规则**对话框中单击目标方案的页签，然后单击**降级规则**页签，在页面右上角单击**新增主动降级规则**。
   - 在左侧导航栏单击**规则设置**，单击目标方案的页签，然后单击**降级规则**页签，在页面右上角单击**新增主动降级规则**。
5. 在设置主动降级规则的对话框，完成以下配置，然后单击**新建**。

参数	描述	示例值
接口名称	适用该规则的资源名称，需要与监控页面上的资源名（埋点传入的资源名）保持一致。	/hello
降级行为选择	表示开启该降级规则后，该接口调用的行为。 **默认行为**对应**应用设置**页面**基础设置**中的**模块适配设置**Web的配置。具体操作，请参见设置适配模块。 **说明** 目前降级行为只对AHAS自带的Web埋点生效。	默认行为

若需要新增行为，单击**新增行为**，完成以下配置，然后单击**新增**。更多信息，请参见配置Web行为。

参数	描述	示例值
行为名称	该行为的名称。长度不超过128个字符，同个应用内名称不能重复。	测试行为
针对的资源类型	目前仅支持Web类型。	Web

参数	描述	示例值
Web限流处理策略	定义Web接口访问触发某种规则后的行为表现。目前支持以下两种策略： ○ 自定义返回：需设置HTTP返回状态码、返回内容的格式和返回的内容。表示Web接口访问触发规则后返回自定义的内容。 ○ 跳转到指定页面：需设置指定跳转的URL。表示Web接口访问触发规则后系统会跳转指定的页面URL。	自定义返回
HTTP返回状态码	默认429。当Web限流处理策略为自定义返回时，需要填写。	429
返回content-type	设置返回内容的格式为普通文本（TEXT）或JSON。当Web限流处理策略为自定义返回时，需要填写。	JSON字符串
HTTP返回文本	输入当Web接口访问触发规则后返回的内容。当Web限流处理策略为自定义返回时，需要填写。	{"message": "blocked oops"}
跳转地址	输入当Web接口访问触发规则后系统会跳转的页面URL。当Web限流处理策略为跳转到指定页面时，需要填写。 **说明** 跳转的本质是返回302状态码。对于后端服务直接渲染返回的页面，跳转是有效的；对于前端通过AJAX请求到后端服务后，再解析后端返回到前端展示的页面，跳转无效。	http://ahas.console.aliyun.com

设置完成的主动降级规则会展示在**规则设置**页面主动降级规则列表中。

6.5. 自适应流控

系统支持自适应流控或手动设置系统规则，自适应流控是根据系统的CPU使用率自动动态地调整应用程序的入口流量；系统规则是从整体维度手动设置规则，对应用入口流量进行控制。目的都是为了让系统的入口流量和系统的负载达到一个平衡，保证系统在最大吞吐量状态下稳定运行。

前提条件

将应用接入AHAS应用防护，具体操作请参见接入应用方式。

背景信息

系统支持开启自适应流控或手动设置系统规则：

- 自适应流控：当开启自适应流控，系统会根据CPU使用率动态调节应用的入口流量，在尽可能保证吞吐量的同

时保证高负载下系统稳定。

- 系统规则：当关闭自适应流控，需要您手动设置系统规则。系统规则从整体维度对应用入口流量进行控制，结合应用的负载、CPU使用率、总体平均RT、入口QPS和并发线程数等几个维度的监控指标，让系统的入口流量和系统的负载达到一个平衡，保证系统在最大吞吐量状态下稳定运行。

系统保护规则是应用整体维度的，而不是资源维度的，并且仅对入口流量生效。入口流量指的是进入应用的流量，例如Web服务或Dubbo服务端接收的请求。通过自定义埋点接入和通过注解接入的应用，入口流量为 `EntryType.IN` 的逻辑被调用时产生的流量。配置系统规则的对象必须是入口流量，且默认接口为Dubbo或HSF才生效，如默认埋点的Dubbo、Servelet等，或者通过修改 `EntryType.IN` 来让系统规则对该方法生效， `EntryType.IN` 的详情请参见通过自定义埋点接入和接入Spring应用。

系统规则支持以下的模式：

- Load（仅对Linux、Unix-like机器生效）：当系统Load 1超过阈值且系统当前的并发线程数超过系统容量时才会触发系统保护。
- CPU使用率：当系统CPU使用率超过阈值（0.0~1.0）即触发系统保护。
- RT：当单台机器上所有入口流量的平均RT达到阈值即触发系统保护。
- 线程数：当单台机器上所有入口流量的并发线程数达到阈值即触发系统保护。
- 入口QPS：当单台机器上所有入口流量的QPS达到阈值即触发系统保护。

对于一个应用来说，每种相同的系统保护规则最多只能存在一条，即一个应用最多配置五条系统保护规则。配置系统规则的原理请参见系统防护。

新建系统规则

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 在左侧导航栏单击**规则设置**，然后单击**自适应流控**页签。
6. 在页面左上角，关闭**自适应流控**，在对话框中单击**确定关闭**。
7. 在**自适应流控**页签右上角单击**新建系统保护规则**。
8. 在**新建系统保护规则**对话框中，配置规则信息。

参数	描述	使用场景说明
CPU使用率	当系统CPU使用率超过阈值即触发系统保护，阈值设置范围为0.0~1.0（代表0%~100%）。	适用于设置基础资源水位的场景，比如需要保证一定的冗余水位。但系统水位不宜过高，需要留部分水位。
Load	当系统的Load1超过阈值，且系统当前的并发线程数超过系统容量时才会触发系统保护。系统容量由系统的maxQps * minRt计算得出。	适用于设置基础资源水位的场景，比如需要保证一定的冗余水位。但系统水位不宜过高，需要留部分水位。
线程数	当单台机器上所有入口流量的并发线程数达到阈值即触发系统保护。	适用于设置基础资源水位的场景，比如需要保证一定的冗余水位。但系统水位不宜过高，需要留部分水位。
入口平均RT	当单台机器上所有入口流量的平均RT达到阈值即触发系统保护，单位是毫秒。	适用于衡量入口请求的场景。

参数	描述	使用场景说明
入口总QPS	当单台机器上所有入口流量的QPS达到阈值即触发系统保护。	适用于衡量入口请求的场景。

9. 单击**新增**。

6.6. 配置热点规则

6.6.1. 配置热点规则

为应用配置热点规则后，AHAS将分析统计参数，即资源调用过程中的调用次数较高的参数，并根据配置的热点规则对包含热点参数的资源调用进行限流，保护系统稳定性。本文介绍如何为应用配置热点规则。

背景信息

热点即经常被访问的数据。例如在以下场景中需要统计某个热点数据中访问频次最高的Top数据，并对其访问进行限制。

- 针对一段时间内最频繁购买的商品ID进行限制，防止击穿缓存而导致大量请求到数据库的情形。
- 针对一段时间内频繁访问的用户ID进行限制，防止恶意刷单。

AHAS利用LRU（Least Recently Used）策略统计最近最常访问的热点参数，结合令牌桶算法来进行参数级别的流控。

使用说明

将应用接入AHAS应用防护，具体操作请参见接入应用方式。

- 若您通过Dubbo应用方式接入，AHAS应用防护将在方法级别的埋点即方法签名的接口名，自动按次序传入调用的参数，因此可以直接对调用参数配置热点规则。详情请参见接入Dubbo应用。
- 若您通过注解方式接入，Sentinel会将@SentinelResource注解方式定义的资源作为参数传入*SphU.entry(res, args)*。详情请参见接入Spring应用。
  例如以下示例中，*uid*和*type*会分别作为第一个和第二个参数传入Sentinel API，从而可以用于热点规则判断：

```
@SentinelResource(value = "myMethod", blockHandler = "myBlockHandlerFunc")
public Result doSomething(String uid, int type) {
  // uid参数索引为0，type参数索引为1。
  // some logic here...
}
```

- 对于其它方式接入的应用，您需要自定义埋点传入需要进行热点限流的参数，如IP、User ID等。请参见资源定义类SphU / SphO。

```
// name：资源名称，用于标识该调用。
// EntryType用于区分该资源是入口流量（inbound）还是出口流量（outbound），EntryType.IN代表入口流量，EntryType.OUT代表出口流量。
// count表示每次调用计数为多少，通常传1。
// args为传入的参数，用于热点参数限流。
public static Entry entry(String name, EntryType entryType, int count, Object... args) throws BlockException
```

*args*即待传入的参数，若有多个待传入参数，则按照次序依次传入。例如需要传入两个参数paramA和paramB，传入方式如下：

```
// paramA对应的参数索引为0，paramB对应的参数索引为1。
// 若需要配置例外项或者使用集群维度流控，则传入的参数只支持基本类型。
SphU.entry(resourceName, EntryType.IN, 1, paramA, paramB);
```

若使用SphU.entry()传入了热点参数，那么使用exit的时候也要带上对应的参数（exit(count, args)），否则并发线程数等指标会有统计错误。正确使用如下：

```
Entry entry = null;
try {
    entry = SphU.entry(resourceName, EntryType.IN, 1, paramA, paramB);
    // 用户的业务逻辑。
} catch (BlockException ex) {
    // 触发限流降级，在此处进行处理。
} finally {
    if (entry != null) {
        // 务必保证entry成功后最终会exit，并且传入了对应的参数。
        entry.exit(1, paramA, paramB);
    }
}
```

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 在左侧导航栏单击**规则设置**，然后单击**热点规则**页签。
6. 在**热点规则**页签右上角单击**新增热点限流规则**。
7. 在**新增热点限流规则**对话框中，配置规则信息。
8. 单击**新增**。
9. （可选）（可选）若需对某些特殊的热点参数值单独配置阈值，则在规则列表中单击目标规则**操作**列的**添加例外项**，然后在**热点规则例外项**对话框中单击**添加**，并填写单独配置的参数信息，然后单击**确定**。

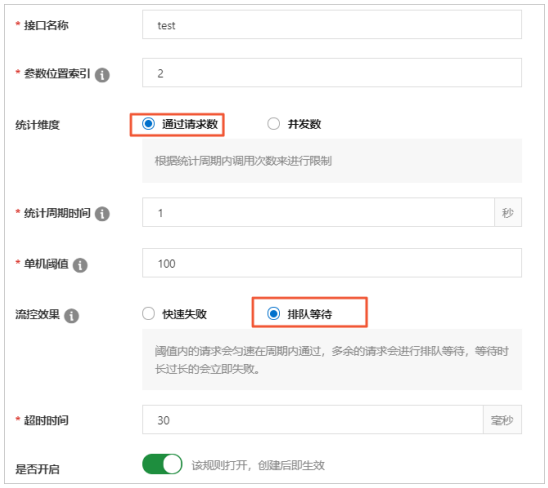
> **说明** 热点规则例外项仅支持基本类型和字符串类型。由于一条规则只对应一个参数索引位置，因此例外项类型需要保持统一。例如在资源R的参数索引0处配置一条热点规则，阈值为5QPS。若在这条规则新增两条例外项：参数Sentinel阈值为100，参数AHAS阈值为200。则实际生效的时候，其它热点参数会限制每秒钟访问不超过5次，而Sentinel和AHAS这两个参数值作为例外项，会按照对应的例外项阈值生效。

常用场景1 秒杀场景

秒杀等抢购商品的时候，由于流量较大会导致系统响应不及时，甚至系统崩溃。为保证系统的稳定性，可配置热点规则，超过一定量的阈值后，系统会让购买热点商品的流量排队等待。

例如购买同一商品，1s内调用超过100次请求后，则其余请求进行等待。在**新建热点规则**对话框中配置以下规则信息。

- 填写**接口名称**。
- **统计维度**选择**通过请求数**。
- **统计周期时间**设置为1s，**单机阈值**设置为100。
- **流控效果**选择**排队等待**。
- **超时时间**设置为30 ms。



1s内调用此接口超过100次，多余的请求要进行排队等待，等待时长超过30 ms的请求就会立即失败。

常用场景2 调用请求频繁，占用较多系统资源

例如秒杀的时候还需要修改下单地址，当调用修改请求较多的时候，会占用了写数据库较多资源，则可以对其进行热点快速失败的处理，稍后再修改。在**新建热点规则**对话框中配置以下规则信息。

- 填写**接口名称**。
- **统计维度**选择**并发数**。
- **统计周期时间**设置为1s，**单机阈值**设置为100。
- **流控效果**选择**快速失败**。

表示1s内只能最多处理100条修改请求，其余超出的请求都会快速失败。

更多信息

新建热点规则页面参数说明如下：

参数	描述
接口名称	适用该规则的资源名称，与埋点传入的资源名保持一致。
参数位置索引	埋点传入参数的索引位置。对应 `SphU.entry(xxx,args)` 中的参数索引位置。例如 `SphU.entry(resourceName,EntryType.IN,1,paramA,paramB)` 埋点中，`paramA` 的参数索引是0，`paramB` 的参数索引是1。

参数	描述
统计维度	可选择通过请求数或并发线程数。 • **通过请求数**：限制一段时间内的调用次数。 • **并发数**：限制该资源调用的并发数。
统计周期时间	统计窗口时间长度（单位为秒）。例如统计窗口时长为10s，QPS阈值为5代表限制10s内每个热点参数访问不超过5次。
单机阈值	是作用于每个热点参数的阈值。
流控效果	当**统计维度**为**通过请求数**时，可以选择流控效果来处理被拦截的流量。 • **快速失败**：达到阈值时，立即拦截请求。该模式下可以额外设置一个**缓冲请求数**，即针对突发请求额外允许的请求数目。 • **排队等待**：请求匀速通过，允许排队等待，通常用于消息队列削峰填谷等场景。需设置具体的超时时间，排队时会计算预计的排队时长，若超过最大超时时间则请求会直接被拒绝。例如，**单机阈值**配置为5，则代表请求每200 ms才能通过一个，多出的请求将排队等待通过。**超时时间**配置1000 ms，则当前排队请求超过5个（>1000 ms）时，新到来的请求将会直接被拒绝。

6.6.2. 查看热点监控详情

热点指的是频繁被访问的数据。通过热点监控统计某个热点数据中访问频次最高的数据，并对这些Top数据的访问进行相关限制。本文介绍配置热点规则后，如何查看热点监控详情。

前提条件

配置热点规则。

操作步骤

在热点规则配置完成后，即可查看热点监控详情。

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏单击**热点详情**。
   在左侧**热点规则**页签中可查看最近10秒被触发的热点规则信息。

5. 单击被触发的热点规则，在左侧的**热点概览**页签可查看该热点的概览信息。

   热点概览字段通过词云图的方式实时展示，热点字段大小根据热点被阻塞的流量设置，被阻塞流量越多，热点字段越大。热点概览生命周期与热点规则相同。

6. 单击**热点详情**页签，可查看该热点的详细信息。

   热点详情中将会保留最近5分钟的热点字段信息。

7. 单击**热点概览**或者**热点详情**页签中的任一热点字段，在**节点详情**页签中可查看该热点分节点限流的情况。

   如下图所示，单击**热点概览**页签中的热点字段**foo9263**，即可查看该几点分节点的限流信息。

6.7. 配置自动重试规则

当您的系统遇到一些非致命性的错误（如偶现的超时等）时，可以通过重试的方式来避免系统最终失败。AHAS的自动重试规则可以在客户端的部分场景下为系统提供自动重试的能力，同时支持自定义重试策略。本文介绍如何配置自动重试规则。

前提条件

将应用接入AHAS应用防护，具体操作，请参见接入应用方式。

背景信息

自动重试规则仅支持AHAS Sentinel Java SDK 1.8.0及以上版本，且仅支持通过注解方式埋点（@SentinelResource）或SentinelWrapper自定义埋点，以及Apache HttpClient/OkHttp适配接入的应用。通过Java Agent以及其他多语言SDK方式接入的应用暂不支持。关于接入Http Client的操作步骤，请参见接入Http Client。

场景说明

以下场景建议您不要设置自动重试规则：

- 建议您不要对致命错误（Fatal Error）重试，重试条件策略一般采取白名单机制（如只对超时异常、限流异常重试）。典型的报错像Error、ClassNotFoundException、IllegalArgumentException这些都建议您不要进行重试，业务错误建议您不要进行重试。
- 重试需要注意幂等性，非幂等的操作在部分场景下重试（如超时，请求已到达对端）可能会不符合预期。
- 前端应用，超时时间长且慢的接口，建议您不要进行重试。
- 对于NIO、Netty、Reactive场景而言，建议您不要在EventLoop线程中重试，避免等待时阻塞事件传递，降低系统吞吐量。

新建自动重试规则

1. 登录AHAS控制台，然后在页面左上角选择**地域**。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**规则设置**，然后选择方案页签，单击**自动重试规则**页签。
5. 单击**规则设置**页面的**新增自动重试规则**。
6. 在**新增自动重试规则**对话框，完成以下配置，然后单击**新建**。

参数	描述
资源名称	适用该规则的资源名称，需要与监控页面上的资源名（埋点传入的资源名）保持一致。
重试策略	重试时间间隔策略，目前支持**固定间隔重试**和**指数间隔重试**两种策略： ○ **固定间隔重试**：每次重试之间的时间间隔固定（时间间隔按照下方设置的**基准重试间隔**）。 ○ **指数间隔重试**：以设置的**基准重试间隔**为基准，每次以一个固定乘积因子指数增长（base *（factor^n）），默认乘积因子为1.5（如基准1000 ms，则每次重试间隔按1500、2250、3375这样的趋势递增）。
基准重试间隔	重试的基准重试时长，每次重试等待的时间会基于基准时长和重试策略来计算，取值范围1 ms~10000 ms，建议不超过5000 ms。
最大重试次数	最大允许的重试次数，建议在3次以内，取值范围1~10。
重试条件策略	设置符合重试的异常条件。 ○ 异常白名单模式：只有在白名单列表内的异常触发时才会重试。 ○ 异常黑名单模式：在黑名单列表里的异常触发时不会进行重试，其余异常都会进行重试。
白名单异常	异常条件集合，每个异常需要填写异常类的全限定名（如java.lang.IllegalArgumentException），目前匹配条件为精确匹配。
黑名单异常	

新增成功的自动重试规则将会出现在**自动重试规则**页签中。

7.配置行为

背景信息

1.

7.1. 配置Web行为

Web行为可以在Web类型埋点资源触发了某种规则后，返回对应的自定义处理行为，例如，某个Web接口触发流控规则后返回Blocked by Sentinel的提示文本。本文介绍如何新增、修改、删除和关联Web行为。

前提条件

将应用接入AHAS应用防护。具体操作，请参见接入应用方式。

背景信息

配置行为主要是配置Fallback行为。Fallback行为定义某个埋点资源触发了某种规则（如流控、熔断、降级）后的处理行为。目前Fallback行为仅支持Web和RPC两种资源类型。本文介绍Fallback行为中Web行为的配置方法。

新增行为

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
5. 单击**新增行为**，在**新增行为**对话框中完成以下配置，然后单击**新建**。

参数	描述	示例值
行为名称	该行为的名称。长度不超过128个字符，同个应用内名称不能重复。	测试行为
针对的资源类型	包括Web和RPC两种类型，此处选择Web类型。	Web
Web限流处理策略	定义Web接口访问触发某种规则后的行为表现。目前支持以下两种策略： ○ 自定义返回：需设置HTTP返回状态码、返回内容的格式和返回的内容。表示Web接口访问触发规则后返回自定义的内容。 ○ 跳转到指定页面：需设置指定跳转的URL。表示Web接口访问触发规则后系统会跳转指定的页面URL。	自定义返回
HTTP返回状态码	默认为429。当Web限流处理策略为自定义返回时，需要填写。	429
返回content-type	设置返回内容的格式为普通文本（TEXT）或JSON。当Web限流处理策略为自定义返回时，需要填写。	JSON字符串

参数	描述	示例值
HTTP返回文本	输入当Web接口访问触发规则后返回的内容。当Web限流处理策略为自定义返回时，需要填写。	{"message": "blocked oops"}
跳转地址	输入当Web接口访问触发规则后系统会跳转的页面URL。当Web限流处理策略为跳转到指定页面时，需要填写。 **说明** 跳转的本质是返回302状态码。对于后端服务直接渲染返回的页面，跳转是有效的；对于前端通过AJAX请求到后端服务后，再解析后端返回到前端展示的页面，跳转无效。	http://ahas.console.aliyun.com

新增的行为会显示在**应用管理**页面的**行为管理**页签中。

修改或删除行为

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
   在行为列表页，您可以查看各个行为的具体描述，修改或删除行为。

关联行为

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 选择以下任意一种方式进入**新增流控防护规则**页面：
   - 在左侧导航栏单击**接口详情**后，在**WEB服务**页签下单击目标接口卡片右上角的图标进入**管理规则**页面，然后单击**新增流控规则**。
   - 在左侧导航栏单击**规则管理**，然后在**流控规则**页签下单击**新增流控规则**。
5. 完成选择防护场景和配置防护规则后，在**配置限流行为**区域，完成下列设置。
   i. 选择**接口类型**为**Web**。
   ii. 在**关联行为**的下拉列表中选择目标行为进行关联，或单击**新增行为**来创建新的行为进行关联。

说明
- 如果您不需要自定义限流后的Fallback行为，则选择默认行为即可，默认接口类型为空。
- 新增规则时，若当前接口已有绑定行为，后续若绑定新的行为则会覆盖接口下已有的行为。
- 选择Fallback类型时，若不选择接口类型，则绑定默认行为；选择接口类型并绑定相应类型行为后，不可修改。

6. 单击**下一步**后，单击**新增**。

7.2. 配置RPC行为

RPC行为可以在RPC类型埋点资源触发了某种规则后，返回相应的自定义的处理行为，例如某个RPC接口触发流控规则后返回自定义的接口返回值。本文介绍如何新增、修改、删除和关联RPC行为。

前提条件

- 将应用接入AHAS应用防护。具体操作，请参见接入应用方式。
- RPC Fallback行为仅支持AHAS Sentinel Java SDK 1.9.5及以上版本以及Java Agent 1.10.1及以上版本。
- RPC行为在AHAS SDK 1.9.5~1.10.3版本仅支持在Dubbo类型接口下生效，在AHAS SDK 1.10.4及以上版本支持Dubbo、HSF、SOFARPC类型接口。
- RPC行为暂不支持其他多语言SDK方式接入。

背景信息

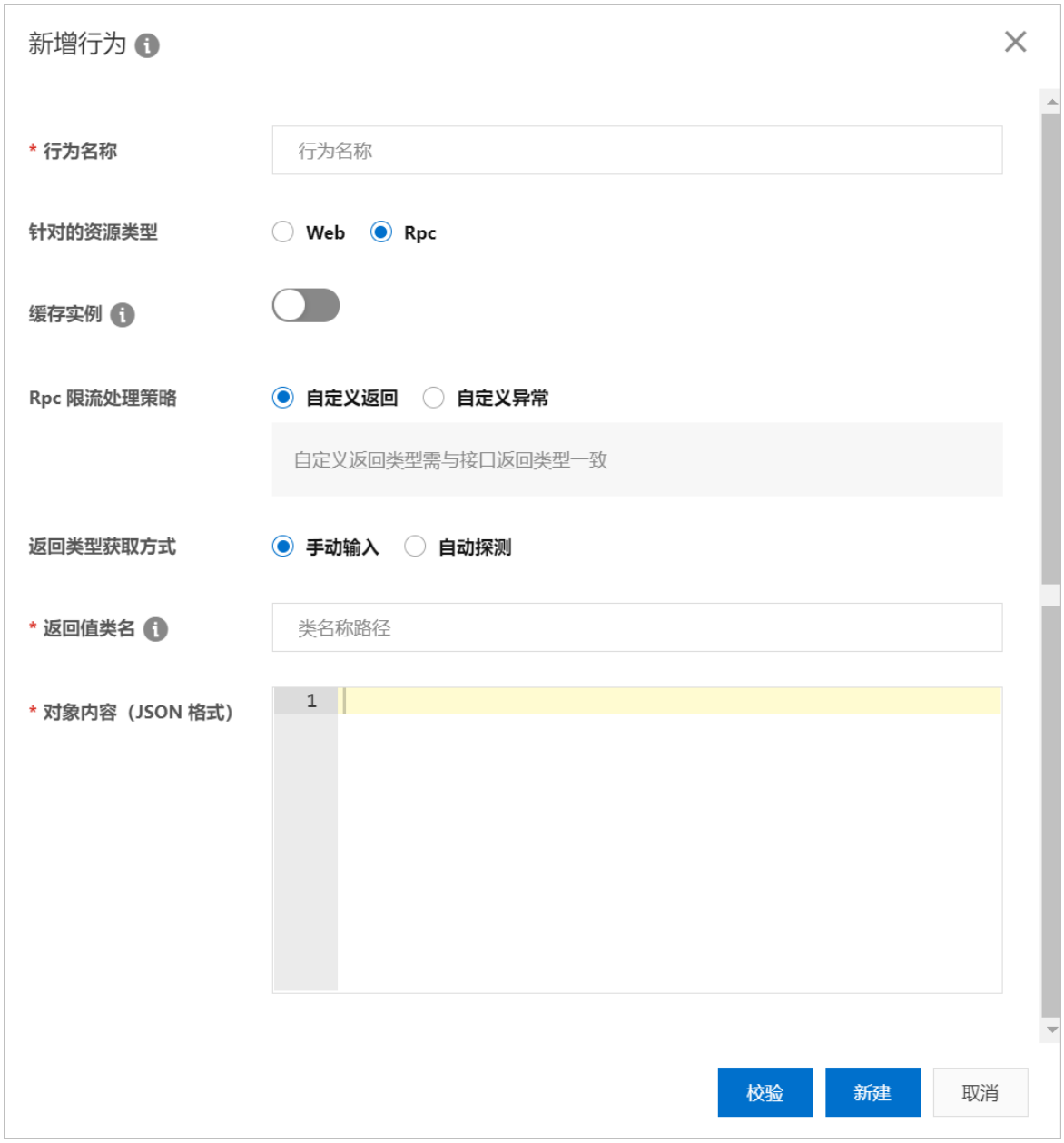
配置行为主要是配置Fallback行为。Fallback行为定义某个埋点资源触发了某种规则（如流控、熔断、降级）后的处理行为。目前Fallback行为仅支持Web和RPC两种资源类型。本文介绍Fallback行为中RPC行为的配置方法。

在AHAS SDK 1.10.4及以上版本，RPC行为支持返回值包含未确定类型的泛型、自动探测接口方法返回类型。

> 注意
>
> - 返回值类名中不允许存在接口、抽象类，请使用相应具体继承类型。
> - 返回值若包含未确定类型的泛型，需要在类名中确定其相应的具化类型。
> - 自动探测模式仅支持Dubbo、HSF类型接口。

新增行为

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
5. 单击**新增行为**，在**新增行为**对话框中完成以下配置。



参数	描述	示例值
行为名称	该行为的名称。长度不超过128个字符，同个应用内名称不能重复。	测试行为
针对的资源类型	包括Web和RPC两种类型，此处选择RPC类型。	RPC
缓存实例	是否缓存返回值。若开启，则会缓存生成的Fallback对象，在该行为触发时，均复用同一对象。	开启

参数	描述	示例值
RPC限流处理策略	定义RPC接口访问触发某种规则后的行为表现。目前支持以下两种策略： ○ **自定义返回**：自定义返回结果。需设置返回类型和返回的内容，表示RPC接口访问触发规则后返回自定义的实体类。 ○ **自定义异常**：抛出自定义异常。需设置异常的类名和异常文本，表示RPC接口访问触发规则后系统会返回指定的异常信息。	自定义返回
返回类型获取方式	填充返回值类名的方式，分别为： ○ **手动输入**：选择手动输入模式，需要填写自定义返回值的全限定类名（返回值类名），然后单击**校验**进行有效性校验，校验通过后可以创建行为。 ○ **自动探测**：选择自动探测模式，需要选择创建RPC行为的接口方法（行为关联方法），控制台会根据选择的行为关联方法，自动填充方法返回值类名，然后填写完成对象内容（JSON格式）后可以创建行为。 AHAS SDK 1.10.4及以上版本支持使用自动探测功能。 **注意** 如果当前接口方法返回值包含未具化的参数类型，需要手动填写泛型的具体类型，通过校验后即可以在控制台创建行为。	手动输入
行为关联方法	当返回类型获取方式为**自动探测**时，需要选择当前RPC行为关联的接口方法，方法返回值类型即为当前自定义返回值的类型。	com.alibaba.demo.RpcResult
返回值类名	选择RPC限流处理策略为**自定义返回**时，需填写的类名称路径。 **说明** **自定义返回**目前不支持对象类型中包含未确定类型的泛型，如Map<K, V>、List<T>等。	com.alibaba.demo.OrderService:getOrder(long)
对象内容（JSON格式）	选择RPC限流处理策略为**自定义返回**时，填写当RPC接口访问触发规则时返回结果的对象内容（JSON格式）。	{"id": "123", "name": "test"}

参数	描述	示例值
异常类名	选择RPC限流处理策略为**自定义异常**时，需填写的异常类名称路径。	java.lang.RuntimeException
异常信息文本	选择RPC限流处理策略为**自定义异常**时，填写RPC接口访问触发规则后抛出自定义异常的文本信息。 **说明** 当前异常仅支持包含String类型构建函数的异常类。	"Operation failed"

6. 单击**校验**。

   说明 RPC行为的数据结构需要与客户端相应接口的数据结构保持一致。因此，新增RPC行为时，需完成数据类型的校验才能完成创建。校验失败时可以根据相应异常信息提示进行修改。

7. 单击**新建**。
   新增的行为会显示在**应用管理**页面的**行为管理**页签中。

修改或删除行为

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
   在行为列表页，您可以查看各个行为的具体描述，修改或删除行为。

关联行为

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 选择以下任意一种方式进入**新增流控防护规则**页面：
   - 在左侧导航栏单击**接口详情**后，在**RPC服务**页签下单击目标接口卡片右上角的图标进入**管理规则**页面，然后单击**新增流控规则**。
   - 在左侧导航栏单击**规则管理**，然后在**流控规则**页签下单击**新增流控规则**。
5. 完成选择防护场景和配置防护规则后，在**配置限流行为**区域，完成下列设置。
   i. 选择**接口类型**为**Rpc**。
   ii. 在**关联行为**的下拉列表中选择目标行为进行关联，或单击**新增行为**来创建新的行为进行关联。

> **说明**
>
> - 如果您不需要自定义限流后的Fallback行为，则选择默认行为即可，默认接口类型为空。
> - 新增规则时，若当前接口已有绑定行为，后续若绑定新的行为则会覆盖接口下已有的行为。
> - 选择Fallback类型时，若不选择接口类型，则绑定默认行为；选择接口类型并绑定相应类型行为后，不可修改。

6. 单击**下一步**后，单击**新增**。

8.场景防护

8.1. Web场景防护

AHAS的Web场景防护功能面向提供Web服务的应用，针对访问请求中的一些参数项进行精细化的流量控制。对于使用了主流Web框架（Servlet容器、Spring Web、Spring Boot）的应用，AHAS实现了API粒度的请求参数解析，通过配置Web流控规则，可以对请求中IP、Host、Header、URL Param等参数维度的资源调用进行流量控制，保护业务与系统的稳定性。本文介绍如何为应用配置Web场景防护。

前提条件

- 将应用接入AHAS应用防护。具体操作，请参见接入应用方式。
- Web场景防护要求AHAS Sentinel Java SDK版本≥1.10.1或Java Agent版本≥1.10.4。

> **说明** Web场景防护暂不支持其他多语言SDK方式接入。

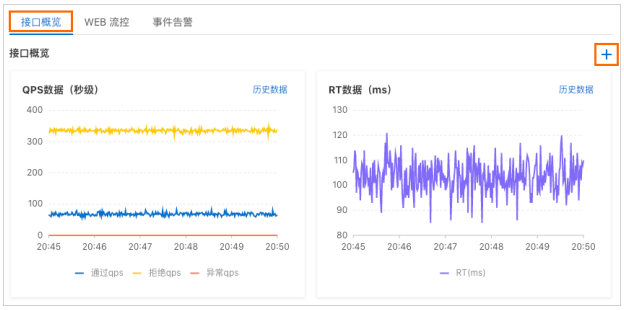
背景信息

在提供Web服务的场景下，除了API维度的限流降级防护，针对访问请求来源IP、访问请求Param参数等资源调用的限流防护是各种业务场景下能更好保证业务应用正常运行的手段。在一些大流量的Web业务场景下，可能不单是对当前接口进行限制，而需要针对当前访问频次最高的来源IP或访问频次最高的商品ID，有针对性地对其访问进行限制，比如：

- 对一段时间内最频繁购买的商品ID进行限制，以防击穿缓存而导致大量请求到数据库的情况。
- 对一段时间内频繁大量访问的来源IP进行限制，防止利用虚假信息恶意刷单。

操作步骤

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**场景防护 > WEB场景**。
5. 选择以下任意一种方式进入Web防护的设置页面：
   - 在**接口概览**页签右侧单击+图标。



   - 单击**WEB流控**页签，然后单击**新增Web防护规则**。
6. 在**新增Web防护规则**对话框中完成规则配置。

参数	说明
选择防护场景	

参数	说明
API	选择需配置规则的接口API。
防护类型	默认为**WEB防护**。
配置防护规则	
参数属性	针对所选API的参数属性进行流量控制： ○ **Client IP**：请求端的IP地址。 **说明** 若请求经过代理，会优先尝试从X-Forwarded-For请求头中获取IP信息，如果其IP信息存在，将会作为实际请求端IP地址。 ○ **Remote Host**：请求端的Host Header。 ○ **Header**：根据指定的HTTP Header进行解析，要填写某个具体的Header Key，则该规则针对这个Header Key下面的热点值分别进行限制。选择**Header**后，可选择配置请求属性值的匹配策略，只有匹配该模式的请求属性值会纳入统计和流控。 ○ **URL参数**：根据指定的HTTP请求参数进行解析，需要填写对应的参数名称。选择**URL参数**后，可选择配置请求属性值的匹配策略，只有匹配该模式的请求属性值会纳入统计和流控。
（可选）匹配模式和匹配串	若选择参数属性为**Header**或**URL参数**，可打开**属性值匹配**开关，并设置**匹配模式**和**匹配串**。 **匹配模式**： ○ **精确**：严格按照给定的匹配串来匹配值。 ○ **子串**：若请求属性值包含该子串则匹配成功，比如若子串设置匹配ab，则aba和cabc都可以匹配，而cba则不能匹配。 ○ **正则**：按给定的正则表达式匹配串进行匹配。
阈值类型	默认为**请求数**。
阈值	触发对流控接口的统计维度对象的QPS阈值。设置时，需选择统计时间间隔，支持秒、分钟、小时和天4种维度。 例如，若阈值填写为*10*，统计间隔选择**分**，则表示每分钟对应的请求数目不超过10个。

参数		说明
高级选项	流控方式	○ **快速失败**：当阈值类型为QPS时，被拦截的流量将快速失败，即达到阈值时，立即拦截请求。 **说明** 被拦截拒绝掉的请求，将返回行为管理中配置的自定义信息，若未配置会返回默认行为，即429错误码加上默认文本信息。 ○ **匀速排队**：当阈值类型为QPS时，被拦截的请求将匀速通过，允许排队等待。需设置具体的超时时间，预计达到超时时间的请求会立即失败，而不会排队。例如，QPS配置为10，则代表请求每100 ms才能通过一个，多出的请求将排队等待通过。超时时间代表最大排队时间，超出最大排队时间的请求将会直接被拒绝。 **说明** 匀速排队时，QPS不要超过1000（请求间隔1 ms）。
	Burst size	当**流控方式**选择为**快速失败**时，可以额外设置一个Burst Size，即针对突发请求额外允许的请求数目。
	超时时间	当**流控方式**选择为**匀速排队**时，需设置具体的超时时间，单位为ms。例如，QPS配置为5，则代表请求每200 ms才能通过一个，多出的请求将排队等待通过。超时时间代表最大排队时间，超出最大排队时间的请求将会直接被拒绝。
是否开启		○ 开启：Web防护规则创建后即生效。 ○ 关闭：Web防护规则创建后不生效。
配置限流行为		
接口类型		默认为**Web**。
关联行为		○ **默认行为**：默认为此选项。 **说明** ■ 如您无需自定义限流后处理行为，选择默认行为即可。 ■ **默认行为**对应**应用基础设置 > Web fallback 行为**里面配置的全局行为。更多信息，请参见应用基础设置.。 ■ 默认行为为返回429错误码加上默认的文本信息。 ○ **新增行为**：新增自定义限流后处理行为，创建完成后您可在**关联行为**选择框中下拉选择。更多详细信息，请参见配置Web行为。 **新增Web行为参数说明 ❯** ○ **取消行为**：取消接口与关联行为的关联。

7. 操作完成后，单击**新增**。
   您可在**WEB流控**页签查看该条防护规则。

8. 在**WEB流控**页签，选择目标Web防护规则，单击**批量开启**。
   规则开启后，当前资源会按照配置的Web流控规则进行流量控制。

Web防护规则常用场景

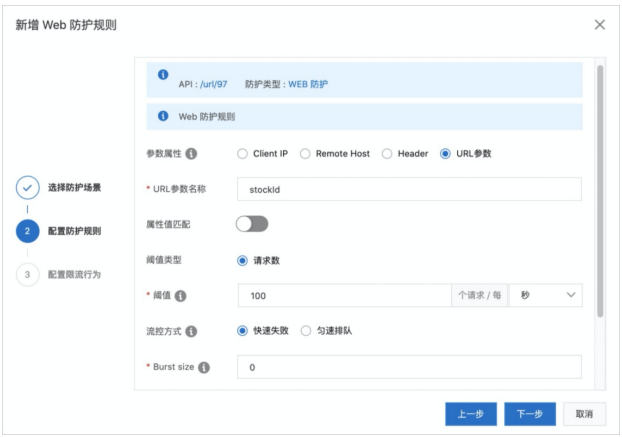
在秒杀等抢购商品的场景下，由于流量较大，可能会导致系统响应不及时甚至崩溃。为保证系统稳定，可配置热点规则，超过一定阈值后，系统会让购买热点商品的流量排队等待。

例如购买同一商品，希望1秒内同个商品超过100次请求后，对多余的请求进行拒绝，可在**新建Web防护规则**对话框中配置以下规则信息，表示1秒内，针对热点商品ID进行下单的请求，每个单独商品ID每秒最多只能允许100个请求，其余超出的当前商品下单请求都会被拒绝，返回自定义的信息。

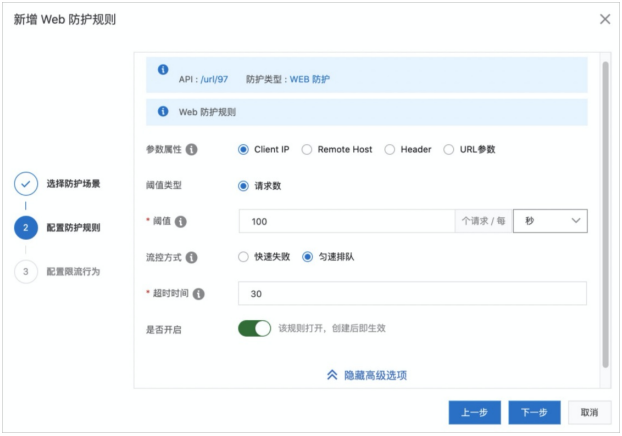
- **参数属性**输入*URL参数*。

  > **说明** 在参数属性中，选择当前热点商品ID所对应的参数字段，例如，假设URL参数中存在一个stockId字段对应请求的商品ID，那么参数属性可以选择URL参数，并在参数名称中填写属性在请求中对应的字段名称。

- **URL参数名称**输入stockId。
- **阈值**输入*100*个请求/每**秒**。
- **流控方式**选择**快速失败**。



例如在促销活动中，某些恶意刷单请求较多的时候，会占用较多商品库存或服务器资源。这种情况下可以针对其IP来源进行排队等待的处理，使访问请求匀速通过，防止过量的请求对服务稳定性产生影响。在**新建Web防护规则**对话框中配置以下规则信息，表示每个不同来源IP调用此接口的请求会以每10 ms一个的速度匀速通过（1s/100=10 ms），后续多余的请求要进行排队等待。排队中的请求如果等待时长超过30 ms就会立即失败。



参数	说明
参数属性	选择**Client IP**。
阈值类型	默认选择*请求数*。
阈值	输入*100*个请求/每**秒**。
流控方式	选择**匀速排队**。
超时时间	输入*30*。
是否开启	选择*开启*。

场景二：防止恶意刷单

9.集群流控

9.1. 配置集群流控规则

集群流控可以控制某个服务调用整个集群的实时调用量，可以解决因流量不均匀导致总体限流效果不佳的问题。集群流控可以精确地控制整个集群的调用总量，结合单机限流兜底，更好地发挥流量防护的效果。本文主要介绍设置集群流控的操作步骤。

商业化说明

集群流控功能已于2021.11.26日暂停商业化收费，目前处于免费公测期。

集群流控试用档位的Token Server可供您测试使用，且不会产生任何费用。试用档位单个应用QPS阈值之和不超过2000，接口总流量不超过3000。

> **说明** 试用档位仅供测试效果使用，不保证稳定性，请勿在生产环境使用。

前提条件

- 已开通AHAS专业版，若没有开通，请进入开通页面。
- SDK版本≥1.6.0，Agent版本≥1.7.5。

背景信息

集群流控通常适用于以下场景：

- 单机流量不均：由于负载不均衡等原因导致每台机器的流量不均，这时使用单机流控可能会出现没有达到请求总量，某些机器就开始限流的情况。
- 集群小流量流控：某些高可用防护场景下需要将服务调用QPS限制到很小的量，此时平均到每台机器的QPS可能小于1，无法通过单机流控进行精确控制。例如希望限制总QPS为50，但节点数有100个。
- 有业务含义的流量控制：例如限制某个API每个用户每分钟调用不超过10次。

> **说明** Token Client与Server的通信会带来一定的网络开销，响应时间可能会上升2 ms~5 ms左右。

步骤一：选择档位创建集群

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在应用防护管理页左侧导航栏单击**集群流控**。
5. 在**集群流控资源配置**区域内，选择**集群类型**为**生产**，然后滑动指针选择集群流控的总配置量级，单击**创建**，然后在对话框中单击**确认**。

   总配置量级即最大QPS，表示需要流控的接口所能承载的预估的最大QPS，代表可能到来的最大流量。

   > **说明** 实际流量（无论是否被流控）超出配置的最大QPS后，流控策略会退化到单机模式。为保证流控效果，阈值之和上限为配置最大QPS的95%，例如最大QPS选择100000，则所有规则阈值之和最大值为95000。

   选定总配置量级档位并创建集群后，系统会自动为该应用分配集群的Token Server。
6. （可选）单击**Token Client设置**区域**操作**列的**编辑**，设置**Token请求超时时间**，然后单击**确定**。

   在某些场景下，集群流控Client与Token Server之间的网络通信时延较高，需要调整超时时间。

> **说明**
> - AHAS Sentinel Client 1.6.0及以上版本支持设置Token Client。
> - Token请求超时时间单位为ms，取值范围为(0,10000]，一般不建议超过20 ms。公网环境网络延时较高，建议设置超时时长约为50 ms，但不建议超过80 ms。

步骤二：设置集群流控规则

1. 在应用防护管理页左侧导航栏单击**规则管理**，选择目标方案页签，单击**流控规则**页签，然后单击**新增流控规则**。
2. 在**新增流控规则**对话框，开启**是否集群流控**，并设置相关参数。

参数	描述	示例
接口名称	设置接口名称。	function_9
是否集群流控	开启此开关，即对集群内此资源的调用总量进行限制。	开启
是否开启	开启此开关，规则即生效；关闭此开关，规则不生效。	开启
集群阈值	表示该接口的限流阈值。	100
统计窗口时长	集群流控统计的时间窗口长度，取值范围为1秒~24小时。	1秒
失败退化策略	当出现连接失败、通信失败或Token Server不可用等情况时，流控规则是退化到单机限流的模式或是直接通过忽略失败情况： ○ 退化到单机限流：当出现通信失败的情况时，退化到设置的单机阈值来进行流控。需要在规则中配置单机退化阈值，代表单机的兜底阈值。 ○ 直接通过：当出现通信失败的情况时，请求直接通过。	退化到单机限流
退化阈值自动调整	开启后会自动调整退化阈值，默认关闭。 **说明** 此功能需要SDK版本≥1.8.6支持。	关闭
退化单机阈值	代表单机的兜底阈值，当失败退化策略选择退化到单机限流时，需要设置此选项。	10

参数	描述	示例
自动调整增量值	当开启退化阈值自动调整时，需要设置自动调整的增量。这是在根据接口阈值与应用机器数量计算出的单机均摊流量基础上，用来提供保护退化阈值的一个增量。即单机均摊流量加上增量值为实际生效退化阈值。	2

3. 单击**新建**，完成规则创建。
   创建规则完成后，可以在**规则设置**页面查看到创建的集群流控规则，**阈值模式**为**集群总体**。

9.2. 查看集群详情

完成集群流控配置后，您可以在**集群详情**页面查看集群整体以及集群下各接口的数据详情，包括集群限流和分节点限流统计详情、限流比、接口流量环比、Token Client请求耗时和Token Client响应类型。

前提条件

1. 将应用接入AHAS应用防护。具体操作，请参见接入应用方式。
2. 完成集群流控规则配置。具体操作，请参见配置集群流控规则。

功能入口

1. 登录AHAS控制台，在控制台左上角选择应用接入的地域。
2. 在左侧导航栏，选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏，选择**集群流控 > 集群详情**。

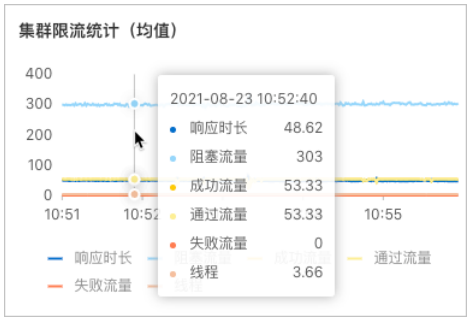
查看数据详情

目标集群详情页面显示了集群的具体情况。您可以在**集群名称**区域单击**全部接口**或目标接口名称，选择查看集群整体详情或各接口详情。

> **说明** 所有图表仅展示最近5分钟的数据。所展示数据每10秒钟刷新一次。

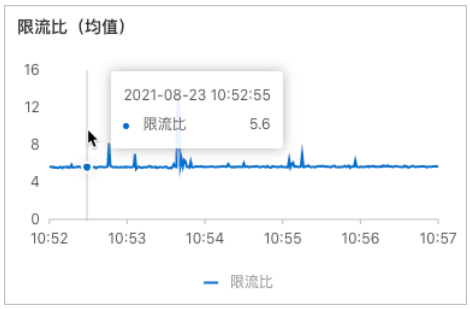
该图表展示统计时间段内集群限流的总体情况，其指标包括响应时长、阻塞流量、成功流量、通过流量、失败流量和线程。

您可以将鼠标悬浮在图上，查看指定时刻下指标的具体数据。您还可以单击面板上的指标名称（例如响应时长），打开或关闭该指标在图表中的可见性。



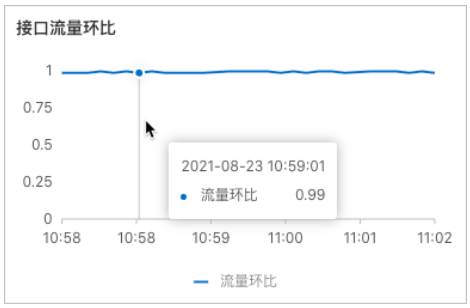
该图表展示被限流流量与通过流量的比值。若比值较大，说明过多流量被阻塞。此时，您可以根据实际需求调整限流规则。

您可以将鼠标悬浮在图上，查看指定时刻下指标的具体数据。

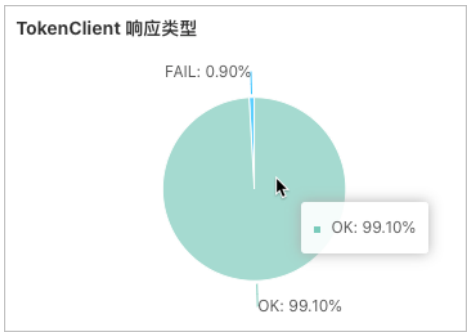


该图表通过计算当前10秒与前10秒的外部流量的比值，展示外部流量随时间变化的情况。

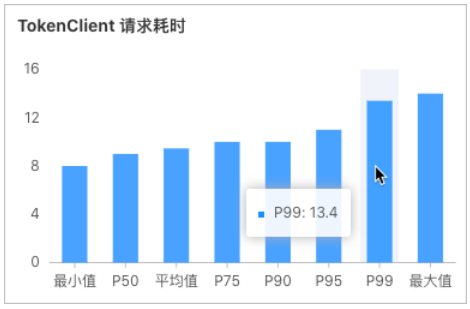
您可以将鼠标悬浮在图上，查看指定时刻下指标的具体数据。



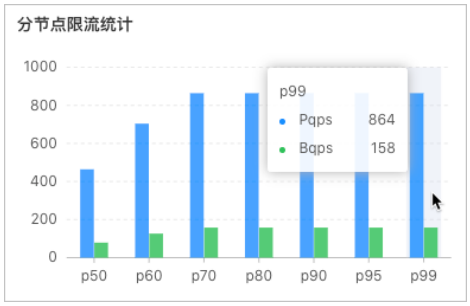
该图表展示Token Client响应成功或失败的比例。



该图表展示Token Client请求的实际耗时情况。您可以参考P99或最大值显示的实际值来调整集群配置中所设置的Token请求超时时间。



该图表展示流量在不同节点的分布情况，以检查负载均衡是否符合预期。PQPS为通过流量，BQPS为阻塞流量。



> **说明** 您可以滑动**选择查询时刻**区域的指针，查看**Token Client响应类型**、**Token Client请求耗时**和**分节点限流统计**图表在指定时刻下的具体数据。

分节点限流统计

Token Client响应类型

Token Client请求耗时

10.管理应用

10.1. 应用概览

将应用接入AHAS应用防护后，AHAS将监控各应用、接口、机器的实时数据，从而评估系统的整体表现，并为流控降级规则提供重要依据。本文介绍应用概览页的主要功能。

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。

功能介绍

应用概览页面会动态刷新，展示应用的限流指标详情，以TOP形式罗列请求、流控、响应时间、异常事件等信息。

- **限流指标**：统计了应用QPS、RT、CPU等数据。

> **说明** **应用概览**中涉及到的QPS、响应时间均为应用入口接口的统计，不包括应用内部方法调用的统计。

  - **QPS数据**：展示了近5分钟通过请求数、流控请求数和异常请求数的时序图。
    - 单击**历史数据**，可以查看QPS更详细的历史数据，最多可以查看7天内的历史数据。
    - 单击图例，可以隐藏或展示该指标的时序图。
  - **防护事件**：展示了该应用在流量防护期间触发的防护事件。
    - 单击**详情**，查看该防护事件的详情。
    - 单击**防护事件查询**，可查看更多的防护事件。
  - **RT**：响应时间，单位为ms。展示了近5分钟响应时间的时序图。单击**历史数据**，可以查看RT更详细的历史数据，最多可以查看7天内的历史数据。
  - **CPU**：用户CPU使用率。展示了近5分钟CPU使用率。单击**历史数据**，可以查看CPU更详细的历史数据，最多可以查看7天内的历史数据。
  - **状态码统计**：展示了近5分钟不同时间HTTP各个状态码的数量。
    - 单击**历史数据**，可以自定义查看任意时间段的状态数据和对比数据，最多查看7天历史数据。
    - 单击图例，可以隐藏或展示该指标的时序图。

- **TOP列表**：包括通过QPS、流控QPS、平均RT的TOP接口列表，以及CPU的TOP节点列表。

  - TOP接口列表会动态刷新，按**通过QPS**由大到小排列。
  - 单击接口名称或该区域右上角的**查看全部**，进入**接口详情**页面，查看所有接口的QPS、CPU、Load等详细信息。
  - 单击**CPU TOP**列表的节点名，进入**机器监控**页面，查看所有节点的应用指标。
  - 单击目标接口**操作**列的**流控**、**隔离**或**降级**，可为该资源配置相应规则。具体操作，请参见配置流控规则、配置熔断规则和配置隔离规则。

- **异常数据**：统计了该应用近5分钟的异常情况，为您排查系统问题提供有效信息。

> **说明** 该区域默认展示最近一次统计情况。

  - 单击**异常统计数据**柱形图中的柱形，则在右侧**异常TOP**列表中，展示该异常的具体接口、类型、次数。
  - 单击**异常统计数据**右上角的**历史数据**，可以查看任意时间段发生的异常次数，最多可以查看7天内的历史数据。
  - 单击**异常TOP**列表**操作**列的**查看**，进入到**接口详情**页面，查看该接口的异常详情。
  - 您在**异常统计数据**区域查看到异常数据后，可在应用本地日志中查看异常记录。

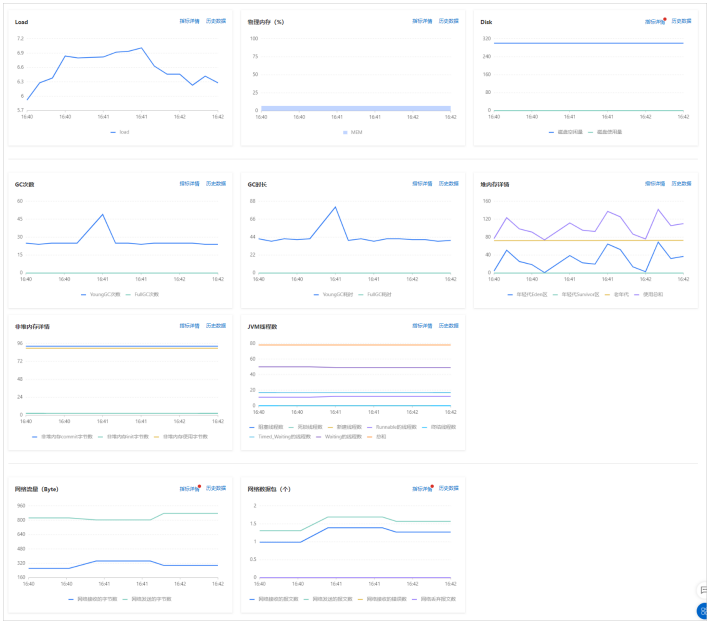
> **说明** 控制台查看以秒为单位的异常统计情况，异常详细内容您可以在本地异常记录文件中查看。

    - 异常记录的文件地址：与*metrics.log*同个目录。
    - 异常记录的文件名称：
      - *{appName}-ahas_exceptions.log.{日期，精确到天}*
      - *{appName}-ahas_exceptions.log.{日期，精确到天}.idx*
    - 异常记录格式：`{时间戳}|{时间，精确到毫秒}|{异常接口}|{异常详情}`

    样例如下所示：

```
1619749703000|2021-04-30 10:28:23.062|/exception1|com.alibaba.csp.sentinel.demo.exception.TestException: test exception_1 15
1619749703000|2021-04-30 10:28:23.065|/exception|com.alibaba.csp.sentinel.demo.exception.AnotherException: test another exception 71
1619749705000|2021-04-30 10:28:25.065|/exception1|com.alibaba.csp.sentinel.demo.exception.TestException: test exception_1 38
```

- **系统资源指标**：展示了系统的Load、物理内存、Disk等指标的时序图。



  - 单击各指标时序图中的**指标详情**，查看各指标的节点分位图、TOP列表等详细信息。
  - 单击各指标时序图中的**历史数据**，可以查看各个指标更详细的历史数据，最多可以查看7天内的历史数据。
  - 鼠标悬浮在图中某一点，可查看该时刻该指标的具体数据。

10.2. 接口详情

在接口详情页面，主要展示该应用所有接口的通过QPS、限流QPS、异常QPS指标、RT、并发数据等，还可以在此页面为接口管理流控规则。本文介绍接口详情页的主要功能。

功能入口

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**接口详情**。

功能介绍

接口详情页面展示了该应用的所有接口的详细信息，包括统计的QPS、RT、并发等数据。

您还可以在此页面进行以下操作：

- （图标①）在页面右上角选择**展示模式**，默认**详情展示**。
  - **详情展示**：以时序图和时序列表的形式展现接口的通过QPS、限流QPS、RT等信息。
  - **统计展示**：以列表的形式展现某一天接口的指标占比、通过总请求数、拒绝总请求数等信息。
- （图标②）在页面右上角可以选择**回放时间**，查看接口的历史数据。

  > **说明** 最多保留7天的历史数据。
- （图标③）在顶部菜单栏中单击各类型的页签可以过滤对应类型的接口，包括**WEB**、**RPC**等。
- （图标④）在接口列表区域，单击接口名称，可以具体查看该接口QPS数据时序图、RT数据时序图、并发数据时序图以及防护事件等，以及该接口在不同节点上的流量情况。
- （图标⑤）在时序图区域，可以选择要展示或隐藏的指标，还可以选择接口指标的展现形式。
  - **节点对比**：各接口以卡片的形式展现各接口的数据。
  - **集群统计**：以QPS、RT、并发各数据的统计维度展现接口的数据。

  单击右上角的图标，可以导出接口详情的数据，包括某一时间段某些接口秒级的QPS数据、RT数据等。一次最多可导出5个接口的数据，也可以选择导出PDF格式或者CSV格式。

  > **说明** 模式的切换仅在全部接口场景下支持。
- （图标⑥）在时序图区域，还可以对各接口设置流控规则等操作。
  - 单击图标，可以将该接口添加至流量大盘，便于在流量大盘中观测系统整体流量，具体操作，请参见创建流量大盘。
  - 单击或图标，进入**管理规则**或**新增规则**对话框，可以新增或删除流控、隔离和降级规则，也可以编辑已有的规则或开启关闭规则。具体操作，请参见配置流控规则、配置隔离规则和配置熔断规则。
  - 单击图标，可以查看该接口指标的历史数据。

    > **说明** 最多保留7天的历史数据。
  - 单击图标，可以导出该接口详情的数据，包括某一时间段秒级的QPS数据、RT数据等，也可以选择导出PDF格式或者CSV格式。

- 单击接口名称后，会在右侧**接口概览**页展示该接口对应的各数据时序图。
  - 单击**节点详情**页签，筛选查看不同接口的数据。
  - 单击**状态统计**页签，展示该接口的HTTP状态码、错误数状态码时序图等。

    > **说明** 错误码默认为非2xx和3xx的状态码。

  - 单击**异常统计**页签，查看该应用近5分钟的异常情况。
    - 单击**异常统计数据**柱形图中的柱形，则在右侧**异常TOP**列表中，展示该异常的具体接口、类型、次数。
    - 单击**异常统计数据**右上角的**历史数据**，可以查看任意时间段发生的异常次数，最多可以查看7天内的历史数据。
    - 您在**异常统计数据**区域查看到异常数据后，可在应用本地日志中查看异常记录。

      > **说明** 控制台查看以秒为单位的异常统计情况，异常详细内容您可以在本地异常记录文件中查看。

      - 异常记录的文件地址：与*metrics.log*同个目录。
      - 异常记录的文件名称：
        - *{appName}-ahas_exceptions.log.{日期，精确到天}*
        - *{appName}-ahas_exceptions.log.{日期，精确到天}.idx*
      - 异常记录格式：`{时间戳}|{时间，精确到毫秒}|{异常接口}|{异常详情}`

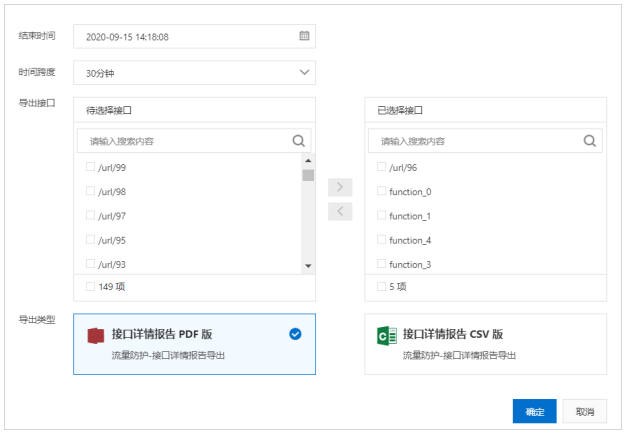
      样例如下所示：

```
1619749703000|2021-04-30 10:28:23.062|/exception1|com.alibaba.csp.sentinel.demo.exception.TestException: test exception_1 15
1619749703000|2021-04-30 10:28:23.065|/exception|com.alibaba.csp.sentinel.demo.exception.AnotherException: test another exception 71
1619749705000|2021-04-30 10:28:25.065|/exception1|com.alibaba.csp.sentinel.demo.exception.TestException: test exception_1 38
```

导出接口数据

AHAS可以导出各个接口详情的数据，包括某一时间段接口秒级的QPS数据、RT数据等，便于您统计分析数据。

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**接口详情**。
5. 您可以通过以下任意一种方式导出数据。
   - 在**接口详情**页时序图区域的右上角单击图标，可以导出多个接口的数据。
   - 在**接口列表**区域，单击接口名称，在时序图区域的右上角单击图标，可以导出该接口的数据。
6. 在**导出接口详情数据**对话框中，选择时间、导出接口以及导出类型，单击**确定**。



> **说明** 目前支持一次性最多导出5个接口的数据。

10.3. 机器监控

在机器监控页面，主要展示了所有节点的通过QPS、限流QPS、异常QPS、RT、并发等指标，还可以在此页面为接口管理流控规则。一个节点对应一个JVM进程，当多个JVM接入单机后，即展示为多个节点。本文介绍机器监控页的主要功能。

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 在左侧导航栏中选择**机器监控**。

功能介绍

机器监控页面展示了应用的所有节点详细信息以及这些节点的QPS、CPU、LOAD时序图。

您可以在此页面进行以下操作：

- （图标①）在页面右上角选择**回放时间**，查看该时间前5分钟内的历史数据。

> **说明** 最多保留7天的历史数据。

- （图标②）在**节点名称**区域，罗列了全部节点和对应的通过QPS、限流QPS、异常QPS、RT等信息。单击节点名称可以查看对应的各数据时序图。
- （图标③）在时序图区域，可以进行以下操作：
  - 在时序图区域，可以选择要展示或隐藏的指标，还可以选择接口指标的展现形式。
    - **节点对比**：各接口以卡片的形式展现各接口的数据。
    - **集群统计**：以QPS、RT、并发各数据的统计维度展现接口的数据。

    > **说明** 模式的切换仅在全部接口场景下支持。

  - 在**节点概览**页签，单击**QPS**、**CPU**、**LOAD**等页签，可以分别查看全部节点相关指标的时序图，还可以选择要展示或隐藏的指标。
  - 在**JVM监控**页签，单击**GC次数**、**GC耗时**等页签，可以查看全部节点JVM的数据。
  - 单击 图标，可以查看该接口指标的历史数据。

    > **说明** 最多保留7天的历史数据。

- 单击节点名称后，会在右侧**节点概览**页展示该节点对应的各数据时序图。
  - 单击**JVM监控**页签，查看该接口的GC次数、GC耗时等时序图。
  - 单击**接口详情**页签，筛选查看不同接口的数据。
  - 单击**callstack信息**页签，查看所有接口的信息，并可以设置该接口的限流规则、查看历史数据。
    - **平铺展示**：不区分调用链路关系，平铺展示接口的运行情况。
    - **树状展示**：根据接口的调用链路关系，展示树状结构。
    - 单击目标接口**操作**列中的**流控**、**隔离**或**降级**，可以快速管理限流规则。更多信息，请参见配置流控规则、配置隔离规则和配置熔断规则。
    - 单击目标接口**操作**列中的**更多 > 查看监控**，可查看该接口指标的历史数据和数据对比情况。
  - 单击**异常统计**页签，查看该应用近5分钟的异常情况。
    - 单击**异常统计数据**柱形图中的柱形，则在右侧**异常TOP**列表中，展示该异常的具体接口、类型、次数。
    - 单击**异常统计数据**右上角的**历史数据**，可以查看任意时间段发生的异常次数，最多可以查看7天内的历史数据。
    - 您在**异常统计数据**区域查看到异常数据后，可在应用本地日志中查看异常记录。

      > **说明** 控制台查看以秒为单位的异常统计情况，异常详细内容您可以在本地异常记录文件中查看。

      - 异常记录的文件地址：与*metrics.log*同个目录。
      - 异常记录的文件名称：
        - *{appName}-ahas_exceptions.log.{日期，精确到天}*
        - *{appName}-ahas_exceptions.log.{日期，精确到天}.idx*
      - 异常记录格式：`{时间戳}|{时间，精确到毫秒}|{异常接口}|{异常详情}`

      样例如下所示：

```
1619749703000|2021-04-30 10:28:23.062|/exception1|com.alibaba.csp.sentinel.demo.except
ion.TestException: test exception_1 15
1619749703000|2021-04-30 10:28:23.065|/exception|com.alibaba.csp.sentinel.demo.excepti
on.AnotherException: test another exception 71
1619749705000|2021-04-30 10:28:25.065|/exception1|com.alibaba.csp.sentinel.demo.except
ion.TestException: test exception_1 38
```

10.4. 规则管理

在规则管理页面，主要展示了流控、隔离、降级等规则下包含的接口信息，还可以通过此页面管理各个接口的限流规则。本文介绍规则管理页面的主要功能。

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 在左侧导航栏中选择**规则管理**。

功能介绍

在规则管理页面以不同类型的规则为维度，展现了各个规则下包含的接口信息，包括接口的来源、阈值模式、流控效果等，还可以在此页面管理接口的流控、隔离、降级等规则。

您可以在此页面进行以下操作：

- 单击**新建方案**，可以新增不同的规则的方案组合。
- 单击各个页签，查看各个规则下包含的接口名称、来源应用、规则状态等信息。
- 在各规则页面，单击**新增XX规则**，可以快速创建流控、隔离等规则。
  具体操作，请参见以下文档：
  - 配置流控规则
  - 配置隔离规则
  - 配置熔断规则
  - 配置主动降级规则
  - 配置热点规则
  - 自适应流控
  - 配置自动重试规则
- 单击目标接口**操作**列的**编辑**或**删除**，可以快速管理限流规则。
- 选择目标接口**操作**列的**更多 > 操作日志**，查看该接口相关的操作日志。
- 单击目标接口**操作**列的**更多 > 历史事件**，可以查看该接口各指标的历史数据。

> **说明** 最多保留7天的历史数据。

10.5. 管理基本信息

在应用管理页面可以查看所有资源相关的接入节点、操作日志、事件中心的详情，本文介绍如何管理接入节点、事件和操作日志。

管理接入节点

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**应用管理**，然后单击**接入节点**页签。

   在节点列表，展示了各个节点的名称、IP地址、健康状态等信息。

管理事件中心

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**应用管理**，然后单击**事件中心**页签。

   在事件中心列表，罗列了相关的防护事件，包括事件的级别、类型、起始时间等信息，其中级别包含了WARNING和ERROR。
5. 单击**操作**列的**查看详情**，查看该接口的历史监控数据。

管理操作日志

1. 登录AHAS控制台，然后在页面左上角选择地域。

2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏选择**应用管理**，然后单击**操作日志**页签。

   在操作日志列表，罗列了相关的操作日志，包括具体的操作内容、类别、操作人ID等信息。
5. 在左上角的搜索框里输入资源名，搜索目标资源的相关操作日志。

10.6. 应用基础设置

在应用管理页面的基础设置中，您可以切换防护模式来提升系统防护能力，配置Web应用触发流控等规则后的处理逻辑，以及设置簇点数目限制、来源数目限制等信息。本文介绍如何进行应用的基础设置。

前提条件

若需使用模块适配设置和通用设置，则需满足以下条件：

- 若使用SDK方式接入，则确保ahas-sentinel-client为1.5.0或以上版本。
- 若使用Agent方式接入，则确保Agent为1.7.1或以上版本。

功能入口

1. 登录AHAS控制台。
2. 在AHAS控制台左上角，选择应用接入的地域。
3. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
4. 在**应用防护**页面单击目标应用卡片。
5. 在左侧导航栏单击**应用管理**，然后单击**基础设置**页签。

设置适配模块

通过设置适配模块功能，可以动态配置Web类型应用触发流控等规则后的处理逻辑。具体操作步骤如下：

1. 在**模块适配设置**区域单击目标应用**操作**列的**修改**。
2. 在**模块适配设置修改**对话框中配置参数。

参数	描述	示例值
Web fallback行为	定义Web接口访问触发某种规则后的行为表现。目前支持以下两种策略： ○ 返回指定内容：需设置HTTP状态码、返回内容的格式和返回的内容。表示Web接口访问触发规则后返回自定义的内容。 ○ 跳转到指定页面：需设置指定跳转的URL。表示Web接口访问触发规则后系统会跳转指定的页面URL。	返回指定内容
HTTP状态码	默认429。当Web限流处理策略为自定义返回时，需要填写。	429
返回content-type	设置返回内容的格式为普通文本（TEXT）或JSON。当Web限流处理策略为自定义返回时，需要填写。	JSON字符串

参数	描述	示例值
HTTP返回文本	输入当Web接口访问触发规则后返回的内容。当Web限流处理策略为自定义返回时，需要填写。	{"message": "blocked oops"}
跳转URL	输入当Web接口访问触发规则后系统会跳转的页面URL。当Web限流处理策略为跳转到指定页面时，需要填写。 **说明** 跳转的本质是返回302状态码。对于后端服务直接渲染返回的页面，跳转是有效的；对于前端通过AJAX请求到后端服务后，再解析后端返回到前端展示的页面，跳转无效。	http://ahas.console.aliyun.com

3. 单击**确定**。

说明

- 适配模块配置的配置项会覆盖JVM参数传入的相关配置项。
- 适配模块配置仅针对默认的Web流控处理逻辑生效。若您注册了自定义的UrlBlockHandler，则适配模块配置无效。

设置通用配置

通过通用设置功能可修改应用的簇点数目限制、来源数目限制、入口数目限制和最大统计RT等配置。具体步骤如下：

1. 在**通用设置**区域**操作**列单击**修改**。
2. 在对话框中按需配置**簇点数目限制**、**来源数目限制**、**入口数目限制**和**最大统计RT**。

配置项	说明
簇点数目限制	限制埋点资源数，默认6000。建议设置限制不超过6000，当实际资源数超过6000时，会导致占用内存较大。
来源数目限制	限制每个资源下来源节点数，用于按调用来源限流。
入口数目限制	限制入口context数，用于链路限流。
最大统计RT	限制统计的最大RT（ms），默认为4900 ms。当统计数超出设置上限时，则按照设置上限记录统计数。

3. 单击**确定**。

10.7. 应用压测

10.7.1. 压测Dubbo服务

应用压测功能通过对系统的压测，得到一系列的性能指标，从而指导您配置最优的防护规则，实现业务的高可用。本文介绍压测Dubbo服务的操作步骤。

前提条件

应用已接入AHAS应用防护，具体操作，请参见接入应用方式。

背景信息

应用压测需要满足以下条件才有功能入口：

- 应用SDK版本是1.8.5及以上。
- 应用防护为高级防护模式。
- 非公网环境。

步骤一：创建压测场景

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在应用详情页的左侧导航栏选择**接口详情**，然后单击**RPC服务**页签。
5. 在**接口详情**页面，单击图标。
6. 在**压测配置**面板中设置**场景配置**和**压力配置**相关参数。

   场景配置页签相关参数说明如下。

参数	描述	示例
场景名称	压测场景名称。	test-dubbo
服务地址	包括IP和端口。您可以根据应用实际端口进行配置，多个服务地址以英文逗号（,）分隔。	○ 192.168.xx.xx:20880 ○ 192.168.xx.xx,192.168.xx.xx:20880
应用	输入需要压测的应用。	Demo-service
组别（可选）	Dubbo微服务组别group。	local
版本（可选）	Dubbo微服务版本version。	1.0
服务	输入应用的服务。	com.alibaba.csp.demo.OrderService
方法	输入服务的方法。关于Dubbo服务的方法参数类型及配置方式，请参见Dubbo参考示例。	String sayHello(String name);
方法参数类型	需要压测服务的函数包含的参数类型。填写的内容是一个字符串类型的JSON数组，数组的每一位代表对应位置的参数类型。除了Java基本类型，其余类型需要填写完整的类路径。关于Dubbo服务的方法参数类型及配置方式，请参见Dubbo参考示例。	["java.lang.String"]

参数	描述	示例
请求参数	设置请求参数。关于Dubbo服务的方法参数类型及配置方式，请参见Dubbo参考示例。	["hello, dubbo"]
超时时间（毫秒）	设置超时时间，单位：毫秒。	1000
打印日志	开启可自动打印日志信息，但会影响到服务压测性能，建议正常压测时关闭。	关闭

压力配置页签相关参数说明如下。

参数	描述
压测模式	TPS吞吐量模式（Transaction Per Second），指系统每秒处理的事务数量。
流量模型	流量模型包括固定压力、阶梯压力和脉冲压力。 ○ **固定压力**：以配置的固定并发值TPS进行施压。公测期间最大TPS为1000。 ○ **阶梯压力**：设置最大值、最小值、递增时长等信息。从最小值开始按照阶梯逐步递增，达到最大值后按照最大值持续施压。 ○ **脉冲压力**：设置峰值、谷值以及持续时间等信息，施压流量以峰值、峰谷的锯齿波的形式进行施压。
TPS	设置TPS，上限为1000。
压测时长（分钟）	指压测总时长，公测期间压测时长3~10分钟。

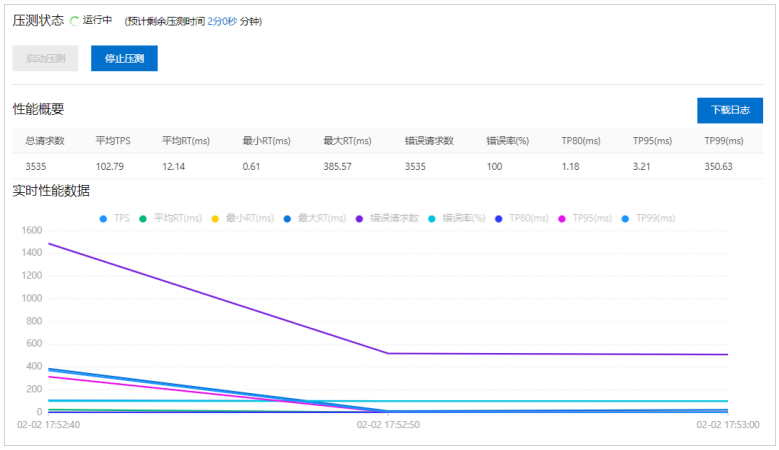
步骤二：执行压测任务

压测场景创建成功后，可以执行压测任务。

1. 在**压测配置**面板，单击**保存配置并启动压测**。

   > **说明** 压测环境准备需要等待约1分钟。

2. 在**性能概要**和**实时性能数据**区域，可以实时查看压测的性能指标。



步骤三：查看压测报告

压测完成后，可以查看压测报告。

1. 在应用详情页的左侧导航栏选择**应用压测**，单击目标场景**操作**列的**详情**。
2. 在**性能数据**面板，查看压测报告。

> **说明** **实时性能数据**统计的是每10秒的所有施压机数据，具体根据压测总时间长度会有所变化。单击图上方的图例，可以显示或隐藏某些数据曲线。

参数	说明
总请求数	整个压测过程中，共发起的请求个数。
平均TPS	压测周期内，所有施压机发出的平均TPS值，TPS=调用总次数/总运行时间。
平均RT（ms）	所有压力机发出平均响应时间。
最小RT（ms）	所有压力机中最小的一次响应时间。
最大RT（ms）	所有压力机中最大的一次响应时间。
错误请求数	所有压力机中错误请求数之和。
错误率	所有压力机中的平均错误率。

参数	说明
TP80（ms）	所有压力机中80分位（P80）的平均值。
TP95（ms）	所有压力机中95分位（P95）的平均值。
TP99（ms）	所有压力机中99分位（P99）的平均值。

3. 在**性能数据**面板，单击**下载日志**，可获取压测过程中的日志。

步骤四：设置防护规则

您可以根据压测报告的结果，在AHAS的应用防护中设置防护规则。在应用详情页的左侧导航栏，选择**规则管理**。具体操作，请参见配置流控规则。

Dubbo参考示例

方法	参数类型填写方式	参数填写方式
String sayHello(String name);	["java.lang.String"]	["hello, dubbo"]
String helloBean(HelloBean helloBean);	["com.alibaba.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloBean(HelloBean helloBean1, HelloBean helloBean2);	["com.alibaba.dubbo.api.DemoService","com.alibaba.pts.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap);	["java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap1, Map helloMap2);	["java.util.Map", "java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]

方法	参数类型填写方式	参数填写方式
String helloList(List helloList);	["java.util.List"]	[[1]]
String helloList(List helloList1, List helloList2);	["java.util.List","java.util.List"]	[[1],[1,2]]
String helloString(String helloString);	["java.lang.String"]	[[1],[1,2],[1,3]]
String helloString(String helloString1, String helloString2);	["java.lang.String","java.lang.String"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt);	["int"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt1, int helloInt2);	["int","int"]	["1","2"]
String helloBoolean(boolean helloBoolean);	["boolean"]	["true"]
String helloBoolean(boolean helloBoolean1, boolean helloBoolean2);	["boolean","boolean"]	["true","false"]
String helloVoid();	[]	[]

10.7.2. 压测Spring Cloud服务

应用压测功能通过对系统的压测，得到一系列的性能指标，从而指导您配置最优的防护规则，实现业务的高可用。本文介绍压测Spring Cloud服务的操作步骤。

前提条件

应用已接入AHAS应用防护，具体操作，请参见接入应用方式。

背景信息

应用压测需要满足以下条件才有功能入口：

- 应用SDK版本是1.8.5及以上。
- 应用防护为高级防护模式。
- 非公网环境。

步骤一：创建压测场景

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在应用详情页的左侧导航栏选择**接口详情**，然后单击**WEB服务**页签。
5. 在**接口详情**页面，单击图标。
6. 在**压测配置**面板中设置**场景配置**和**压力配置**相关参数。

   场景配置页签相关参数说明如下。

参数	描述	示例
场景名称	压测场景名称。	test-springcloud
服务地址	包括IP和端口。您可以根据应用实际端口进行配置，多个服务地址以英文逗号（,）分隔。	○ 192.168.xx.xx:8080 ○ 192.168.xx.xx,192.168.xx.xx:8081
URI路径	请求路径URL。	/demo
请求方式	选择请求方式，包括GET、POST、PUT、DELETE。 **说明** GET和DELETE只支持修改URL的Path路径。POST和PUT支持ContentType及参数编写格式。	POST
请求头	设置请求头。关于Spring Cloud微服务支持的ContentType类型和参数编写格式，请参见Spring Cloud参考示例。	ContentType：application/json
请求参数	设置请求头参数信息。关于Spring Cloud微服务支持的ContentType类型和参数编写格式，请参见Spring Cloud参考示例。 **说明** 请求方式为POST和PUT时，需填写请求参数。	{"name": "cart", "age": 20}
超时时间（毫秒）	设置超时时间，单位：毫秒。	1000
打印日志	开启可自动打印日志信息，但会影响到服务压测性能，建议正常压测时关闭。	关闭

压力配置页签相关参数说明如下。

参数	描述
压测模式	TPS吞吐量模式（Transaction Per Second），指系统每秒处理的事务数量。

参数	描述
流量模型	流量模型包括固定压力、阶梯压力和脉冲压力。 ○ **固定压力**：以配置的固定并发值TPS进行施压。公测期间最大TPS为1000。 ○ **阶梯压力**：设置最大值、最小值、递增时长等信息。从最小值开始按照阶梯逐步递增，达到最大值后按照最大值持续施压。 ○ **脉冲压力**：设置峰值、谷值以及持续时间等信息，施压流量以峰值、峰谷的锯齿波的形式进行施压。
TPS	设置TPS，上限为1000。
压测时长（分钟）	指压测总时长，公测期间压测时长3~10分钟。

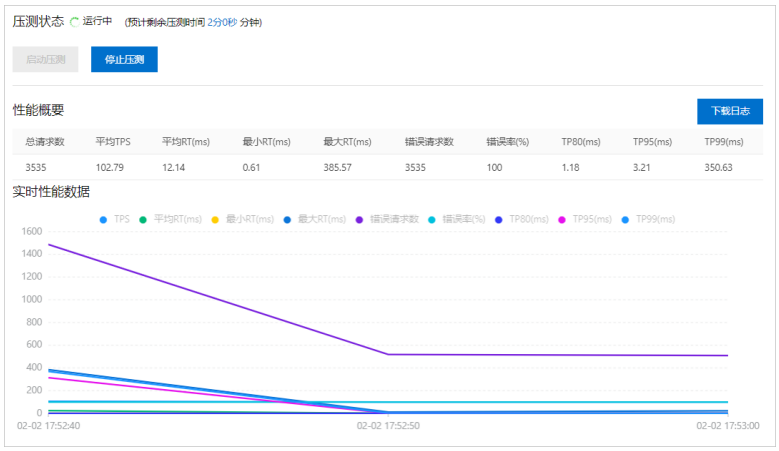
步骤二：执行压测任务

压测场景创建成功后，可以执行压测任务。

1. 在**压测配置**面板，单击**保存配置并启动压测**。

> **说明** 压测环境准备需要等待约1分钟。

2. 在**性能概要**和**实时性能数据**区域，可以实时查看压测的性能指标。



步骤三：查看压测报告

压测完成后，可以查看压测报告。

1. 在应用详情页的左侧导航栏选择**应用压测**，单击目标场景**操作**列的**详情**。
2. 在**性能数据**面板，查看压测报告。

说明 **实时性能数据**统计的是每10秒的所有施压机数据，具体根据压测总时间长度会有所变化。单击图上方的图例，可以显示或隐藏某些数据曲线。

参数	说明
总请求数	整个压测过程中，共发起的请求个数。
平均TPS	压测周期内，所有施压机发出的平均TPS值，TPS=调用总次数/总运行时间。
平均RT（ms）	所有压力机发出平均响应时间。
最小RT（ms）	所有压力机中最小的一次响应时间。
最大RT（ms）	所有压力机中最大的一次响应时间。
错误请求数	所有压力机中错误请求数之和。
错误率	所有压力机中的平均错误率。
TP80（ms）	所有压力机中80分位（P80）的平均值。
TP95（ms）	所有压力机中95分位（P95）的平均值。
TP99（ms）	所有压力机中99分位（P99）的平均值。

3. 在**性能数据**面板，单击**下载日志**，可获取压测过程中的日志。

步骤四：设置防护规则

您可以根据压测报告的结果，在AHAS的应用防护中设置防护规则。在应用详情页的左侧导航栏，选择**规则管理**。具体操作，请参见配置流控规则。

Spring Cloud参考示例

ContentType	参数编写格式
application/x-www-form-urlencoded	[{"name": "cart"},{"age": 20}]

ContentType	参数编写格式
application/json（默认）	{"name": "cart", "age": 20}

11.管理告警

11.1. 管理告警规则

将应用接入AHAS应用防护后，您可以使用告警管理功能为目标应用配置相应的告警规则。当监控的指标超过设定的阈值时，系统会自动发送告警，以便您及时处理。本文介绍如何为接入的应用创建告警规则并对告警规则进行管理。

前提条件

已在AHAS控制台添加钉钉群作为联系人接收告警信息。具体详情，请参见配置告警联系人。

功能入口

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。
4. 在左侧导航栏中选择**告警管理 > 告警规则**。

创建告警规则

您目前可以使用AHAS告警管理功能为指定的系统指标和状态码创建告警规则。

1. 在**告警规则**页面的**系统指标**页签下，单击**新增告警规则**。
2. 在**新增告警规则**对话框中，完成以下参数的配置，然后单击**新建**。

参数	说明
告警名称	自定义的告警名称。
告警级别	分为通知、告警和严重三类。您可以根据实际情况选择相应的告警级别。
告警条件	分为所有和任意两类： ○ 所有：当设置的触发条件全部满足时，系统会发送告警。 ○ 任意：当设置的触发条件满足一条或以上时，系统会发送告警。
是否开启	○ 打开**是否开启**开关：在告警规则创建完成时，规则立即生效。 ○ 关闭**是否开启**开关：在告警规则创建完成时，规则不生效。您可以后续编辑，手动打开开关，使规则生效。

参数	说明
触发条件	单击**+添加触发条件**，在弹出的**新增告警触发条件**对话框中，配置以下参数，然后单击**添加**，为告警规则创建相应的触发条件： ○ 资源类型：需进行告警的系统指标。具体包含的所有系统指标及其含义，请参见告警指标说明。 ○ 检测周期：对设置的触发条件进行检测的周期。单位为分钟，取值范围为1 min～60 min。假设检测周期设置为统计粒度5分钟，则意味系统会每5分钟进行一次检查，判断是否满足当前设置的告警触发条件。 ○ 生效关系：所选资源类型和设定的告警阈值之间的关系。 ○ 告警阈值：单位为%，取值范围为[0,100)。 ○ 等待周期：首次告警后再次收到告警通知的时间间隔，取值范围为[0,100)。每个等待周期时长与检测周期时长相同。假设检测周期设置为统计粒度5分钟，且等待周期设置为首次告警等待2个周期，那么您在收到首次告警通知后，系统会等待10分钟，若10分钟后仍满足告警触发条件，则您会再次接收到告警通知。
告警周期	若满足告警规则的触发条件，且已经经过了告警等待周期之后，系统再次发送告警通知的时间间隔。 假设您的某一告警规则设置如下： ○ 检测周期：统计粒度5分钟 ○ 等待周期：首次告警等待1个周期 ○ 告警周期：每5分钟告警一次 那么当您第一次接收到告警之后，系统会先等待1个周期（也就是5分钟），如果此时问题仍然存在，您会收到第二次告警。如果问题依然持续，那么之后系统就会按照告警周期设置的每5分钟告警一次来发送告警通知。

1. 在**告警规则**页面下，单击**状态码**页签，然后单击**新增告警规则**。
2. 在**新增告警规则**对话框中，完成以下参数的配置，然后单击**新建**。

参数	说明
告警名称	自定义的告警名称。
告警级别	分为通知、告警和严重三类。您可以根据实际情况选择相应的告警级别。
告警条件	分为所有和任意两类： ○ 所有：当设置的触发条件全部满足时，系统会发送告警。 ○ 任意：当设置的触发条件满足一条或以上时，系统会发送告警。
是否开启	○ 打开**是否开启**开关：在告警规则创建完成时，规则立即生效。 ○ 关闭**是否开启**开关：在告警规则创建完成时，规则不生效。您可以后续编辑，手动打开开关，使规则生效。

参数	说明
触发条件	单击**+添加触发条件**，在弹出的**新增告警触发条件**对话框中，配置以下参数，然后单击**添加**，为告警规则创建相应的触发条件： ○ 资源类型：需进行告警的状态码指标。具体包含的所有状态码指标及其含义，请参见告警指标说明。 ○ 检测周期：对设置的触发条件进行检测的周期。单位为分钟，取值范围为1 min～60 min。假设检测周期设置为统计粒度5分钟，则意味系统会每5分钟进行一次检查，判断是否满足当前设置的告警触发条件。 ○ 生效关系：所选资源类型和设定的告警阈值之间的关系。 ○ 告警阈值：单位为%，取值范围为[0,100)。 ○ 等待周期：首次告警后再次收到告警通知的时间间隔，取值范围为[0,100)。每个等待周期时长与检测周期时长相同。假设检测周期设置为统计粒度5分钟，且等待周期设置为首次告警等待2个周期，那么您在收到首次告警通知后，系统会等待10分钟，若10分钟后仍满足告警触发条件，则您会再次接收到告警通知。
告警周期	若满足告警规则的触发条件，且已经经过了告警等待周期之后，系统再次发送告警通知的时间间隔。 假设您的某一告警规则设置如下： ○ 检测周期：统计粒度5分钟 ○ 等待周期：首次告警等待1个周期 ○ 告警周期：每5分钟告警一次 那么当您第一次接收到告警之后，系统会先等待1个周期（也就是5分钟），如果此时问题仍然存在，您会收到第二次告警。如果问题依然持续，那么之后系统就会按照告警周期设置的每5分钟告警一次来发送告警通知。

说明 目前告警事件和告警触发时间存在一分钟左右的误差，但不会影响数据采集。

1. 在**告警规则**页面的**流量**页签下，单击**新增告警规则**。
2. 在**新增告警规则**对话框中，完成以下参数的配置，然后单击**新建**。

参数	说明
告警名称	自定义的告警名称。
接口名称	需输入真实接口名称。
告警级别	分为通知、告警和严重三类。您可以根据实际情况选择相应的告警级别。
告警条件	分为所有和任意两类： ○ 所有：当设置的触发条件全部满足时，系统会发送告警。 ○ 任意：当设置的触发条件满足一条或以上时，系统会发送告警。
是否开启	○ 打开**是否开启**开关：在告警规则创建完成时，规则立即生效。 ○ 关闭**是否开启**开关：在告警规则创建完成时，规则不生效。您可以后续编辑，手动打开开关，使规则生效。

参数	说明
触发条件	单击**+添加触发条件**，在弹出的**新增告警触发条件**对话框中，配置以下参数，然后单击**添加**，为告警规则创建相应的触发条件： ○ 资源类型：需进行告警的系统指标。具体包含的所有系统指标及其含义，请参见告警指标说明。 ○ 检测周期：对设置的触发条件进行检测的周期。单位为分钟，取值范围为1 min～60 min。假设检测周期设置为统计粒度5分钟，则意味系统会每5分钟进行一次检查，判断是否满足当前设置的告警触发条件。 ○ 生效关系：所选资源类型和设定的告警阈值之间的关系。 ○ 告警阈值：单位为%，取值范围为[0,100)。 ○ 等待周期：首次告警后再次收到告警通知的时间间隔，取值范围为[0,100)。每个等待周期时长与检测周期时长相同。假设检测周期设置为统计粒度5分钟，且等待周期设置为首次告警等待2个周期，那么您在收到首次告警通知后，系统会等待10分钟，若10分钟后仍满足告警触发条件，则您会再次接收到告警通知。
告警周期	若满足告警规则的触发条件，且已经经过了告警等待周期之后，系统再次发送告警通知的时间间隔。 假设您的某一告警规则设置如下： ○ 检测周期：统计粒度5分钟 ○ 等待周期：首次告警等待1个周期 ○ 告警周期：每5分钟告警一次 那么当您第一次接收到告警之后，系统会先等待1个周期（也就是5分钟），如果此时问题仍然存在，您会收到第二次告警。如果问题依然持续，那么之后系统就会按照告警周期设置的每5分钟告警一次来发送告警通知。

1. 在**告警规则**页面的**时延**页签下，单击**新增告警规则**。
2. 在**新增告警规则**对话框中，完成以下参数的配置，然后单击**新建**。

参数	说明
告警名称	自定义的告警名称。
接口名称	需输入真实接口名称。
告警级别	分为通知、告警和严重三类。您可以根据实际情况选择相应的告警级别。
告警条件	分为所有和任意两类： ○ 所有：当设置的触发条件全部满足时，系统会发送告警。 ○ 任意：当设置的触发条件满足一条或以上时，系统会发送告警。
是否开启	○ 打开**是否开启**开关：在告警规则创建完成时，规则立即生效。 ○ 关闭**是否开启**开关：在告警规则创建完成时，规则不生效。您可以后续编辑，手动打开开关，使规则生效。

参数	说明
触发条件	单击**+添加触发条件**，在弹出的**新增告警触发条件**对话框中，配置以下参数，然后单击**添加**，为告警规则创建相应的触发条件： ○ 资源类型：需进行告警的系统指标。具体包含的所有系统指标及其含义，请参见告警指标说明。 ○ 检测周期：对设置的触发条件进行检测的周期。单位为分钟，取值范围为1 min～60 min。假设检测周期设置为统计粒度5分钟，则意味系统会每5分钟进行一次检查，判断是否满足当前设置的告警触发条件。 ○ 生效关系：所选资源类型和设定的告警阈值之间的关系。 ○ 告警阈值：单位为%，取值范围为[0,100)。 ○ 等待周期：首次告警后再次收到告警通知的时间间隔，取值范围为[0,100)。每个等待周期时长与检测周期时长相同。假设检测周期设置为统计粒度5分钟，且等待周期设置为首次告警等待2个周期，那么您在收到首次告警通知后，系统会等待10分钟，若10分钟后仍满足告警触发条件，则您会再次接收到告警通知。
告警周期	若满足告警规则的触发条件，且已经经过了告警等待周期之后，系统再次发送告警通知的时间间隔。 假设您的某一告警规则设置如下： ○ 检测周期：统计粒度5分钟 ○ 等待周期：首次告警等待1个周期 ○ 告警周期：每5分钟告警一次 那么当您第一次接收到告警之后，系统会先等待1个周期（也就是5分钟），如果此时问题仍然存在，您会收到第二次告警。如果问题依然持续，那么之后系统就会按照告警周期设置的每5分钟告警一次来发送告警通知。

为系统指标创建告警规则

为状态码创建告警规则

为流量创建告警规则

为时延创建告警规则

管理已创建的告警规则

您可以在**告警规则**页面查看您所创建的告警规则，并对其进行管理，包括查看详细信息、更改生效状态、编辑和删除等。

具体操作	说明
搜索目标告警规则	在**新增告警规则**右侧的搜索框中输入告警规则名称即可进行搜索。支持使用关键词进行模糊搜索。

具体操作	说明
查看告警规则详情	单击目标告警规则一行左侧的+图标即可展开更多详情。
更改告警规则生效状态	打开或关闭**状态**列的开关即可更改告警规则生效状态。
编辑告警规则	单击**操作**列的**编辑**按钮即可在弹出的**编辑告警规则**的对话框中进行编辑。 **注意** 只有在告警规则处在未生效状态时，您才可以对其进行编辑。若想对处在生效状态的告警规则进行编辑，您需要先关闭其状态开关。
删除告警规则	单击**操作**列的**删除**按钮即可删除告警规则。 **注意** 只有在告警规则处在未生效状态时，您才可以进行删除操作。若想删除处在生效状态的告警规则，您需要先关闭其状态开关。

接收告警通知

告警规则创建成功后，若触发告警，系统会自动将告警通知推送到您指定的钉钉群中。

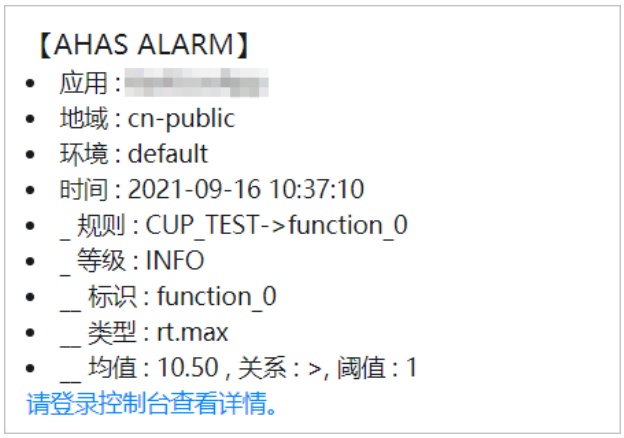
钉钉群内显示的告警通知示例如下：

系统指标告警通知

状态码告警通知

流量告警通知

时延告警通知



告警指标说明

目前AHAS告警管理功能中可用于创建告警规则的系统指标和状态码如下：

- 系统指标：

类型	说明
system.cpu.total	CPU资源平均使用率
system.mem.used	内存资源平均使用率
system.disk.partition.used	磁盘资源平均使用率
system.load.1min	系统1min平均负载
jvm.gc.old.count	老年代GC平均次数
jvm.gc.old.time	老年代GC平均时长
jvm.gc.young.count	年轻代GC平均次数
jvm.gc.young.time	年轻代GC平均时长

> **说明** CPU资源平均使用率、内存资源平均使用率和磁盘资源平均使用率的计算方式如下：
> - CPU资源平均使用率=等待IO完成的CPU使用率+系统CPU使用率+用户CPU使用率
> - 内存资源平均使用率=已使用内存量/总量
> - 磁盘资源平均使用率=已使用磁盘量/总量

- 状态码：

类型	说明
status.200	状态为 200的接口平均请求数
status.2xx	状态为 2xx的接口平均请求数
status.total	接口请求数总量平均
status.3xx	状态为3xx的接口平均请求数
status.4xx	状态为4xx的接口平均请求数
status.5xx	状态为5xx的接口平均请求数

> **说明** 接口平均请求数为该接口在各节点请求数总和的平均值。

- 流量：

类型	说明
qps.exceptionMax	异常流量最大值
qps.exceptionMin	异常流量最小值
qps.exceptionAvg	异常流量平均值

类型	说明
qps.exceptionStd	异常流量方差
qps.exceptionP99	异常流量P99分位值
qps.exceptionP95	异常流量P95分位值
qps.exceptionP75	异常流量P75分位值
qps.blockedMax	阻塞流量最大值
qps.blockedMin	阻塞流量最小值
qps.blockedAvg	阻塞流量平均值
qps.blockedStd	阻塞流量方差
qps.blockedP99	阻塞流量P99分位值
qps.blockedP95	阻塞流量P95分位值
qps.blockedP75	阻塞流量P75分位值

- 时延：

类型	说明
rt.max	接口耗时最大值
rt.min	接口耗时最小值
rt.avg	接口耗时平均值
rt.std	接口耗时方差
rt.p99	接口耗时99分位值
rt.p95	接口耗时95分位值
rt.p75	接口耗时75分位值

11.2. 查看告警详情

您可以在AHAS控制台的告警详情页面查看所有历史告警信息，以及具体告警规则的操作日志。

前提条件

已完成告警规则创建并使其处于生效状态。具体详情，请参见管理告警规则。

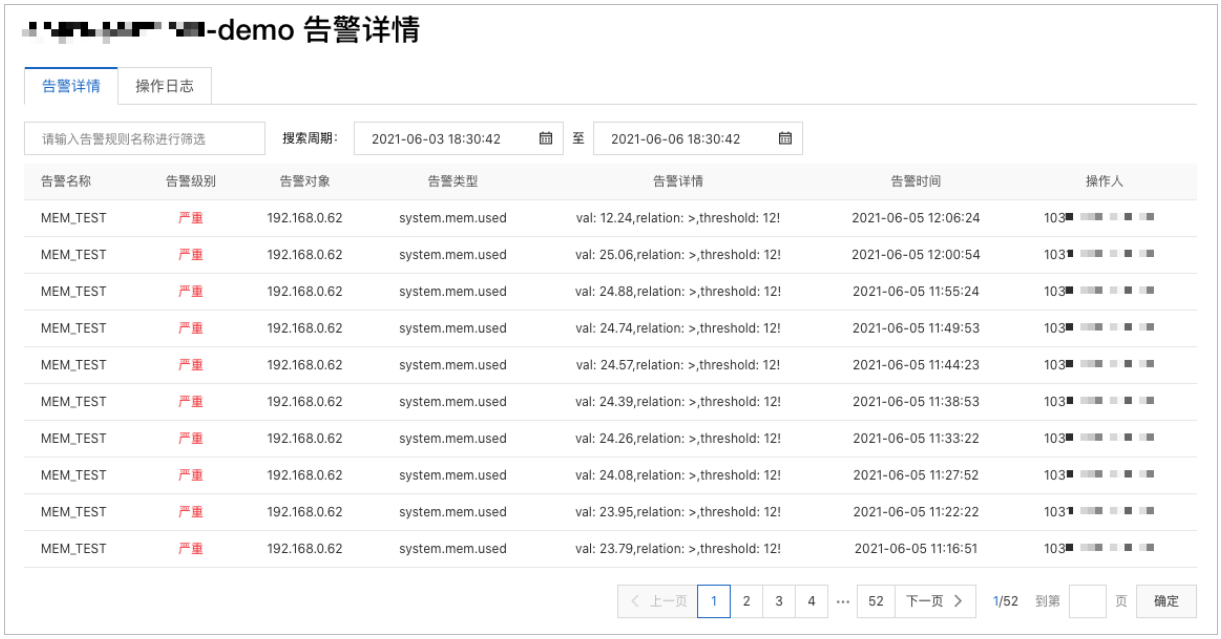
功能入口

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在控制台左侧导航栏中选择**流量防护 > 应用防护**。
3. 在**应用防护**页面单击目标应用卡片。

4. 在左侧导航栏中选择**告警管理 > 告警详情**。

查看历史告警信息

在**告警详情**页签下，您可以查看所有已触发的告警的详细信息，包括告警级别、告警对象、告警类型和告警详情等。您还可以通过搜索具体的告警规则名称或指定具体的时间范围来快速查找到符合要求的告警信息。



参数	说明
告警名称	所创建的告警规则的名称。
告警级别	所创建的告警规则的级别，分为通知、告警和严重三类。
告警对象	根据告警规则的告警类别而定。若告警类型为系统指标，则告警对象为节点；若告警类型为状态码，则告警对象为接口。

告警类型	目前所支持的告警类型如下：
	• 系统指标
	  ◦ system.cpu.total：CPU资源平均使用率
	  ◦ system.mem.used：内存资源平均使用率
	  ◦ system.disk.partition.used：磁盘资源平均使用率
	  ◦ system.load.1min：系统1 min平均负载
	  ◦ jvm.gc.old.count：老年代GC平均次数
	  ◦ jvm.gc.old.time：老年代GC平均时长
	  ◦ jvm.gc.young.count：年轻代GC平均次数
	  ◦ jvm.gc.young.time：年轻代GC平均时长
	• 状态码
	  ◦ status.3xx：状态为3xx的接口平均请求数
	  ◦ status.4xx：状态为4xx的接口平均请求数
	  ◦ status.5xx：状态为5xx的接口平均请求数

参数	说明
告警详情	• val：触发告警时的数值。 • relation：告警规则中设置的生效关系。 • threshold：告警规则中设置的告警阈值。
告警时间	此条告警被触发的时间。 **说明** 目前告警事件和告警触发时间存在一分钟左右的误差，但不会影响数据采集。
操作人	创建此条告警的账号UID。

以下面这条告警历史为例：

告警名称	告警级别	告警对象	告警类型	告警详情	告警时间	操作人
MEM_TEST	严重	192.168.0.62	system.mem.used	val: 12.24,relation: >,threshold: 12!	2021-06-05 12:06:24	103

此条告警历史所对应的告警规则监控的资源类型为内存资源平均使用率，且触发告警的具体节点为192.168.0.62。此节点在某一时间点的内存资源平均使用率达到12.24%，超过了设定的告警阈值12%，因此产生告警。

说明 若告警规则被删除，该告警规则在被删除前所产生的告警详情仍会作为历史记录被保留在**告警详情**页签中，不会被一并删除。

查看操作日志

单击**操作日志**页签即可查看告警规则相关的操作历史记录。您还可以通过指定时间范围来快速查找符合要求的操作历史记录。

参数	说明
告警名称	所创建的告警规则的名称。
告警级别	所创建的告警规则的级别，分为通知、告警和严重三类。
操作类型	针对该告警规则进行的操作，包括新增、删除、开启规则、关闭规则和规则变更。
操作时间	执行该操作的时间。
操作人	执行该操作的账号UID。

12.创建流量大盘

若您需要关注多个系统的整体流量情况，可以通过创建流量大盘功能来实现。流量大盘可展示不同应用、不同接口的通过QPS、拒绝QPS、异常QPS、RT和并发数等信息，给您带来更好的监控体验。

前提条件

接入AHAS应用防护或网关防护。

- 接入应用防护，请参见接入应用方式。
- 接入网关防护，请参见什么是网关防护。

背景信息

流量大盘可以提高运维效率，例如在常见电商场景中，购物网站后端需要多个系统支撑，多个系统之间调用密切。若需查看下单操作的表现情况，可以将下单操作相关接口添加至大盘中，即可方便地查看各接口流量信息。运维人员无需关注各个系统，提高运维效率。

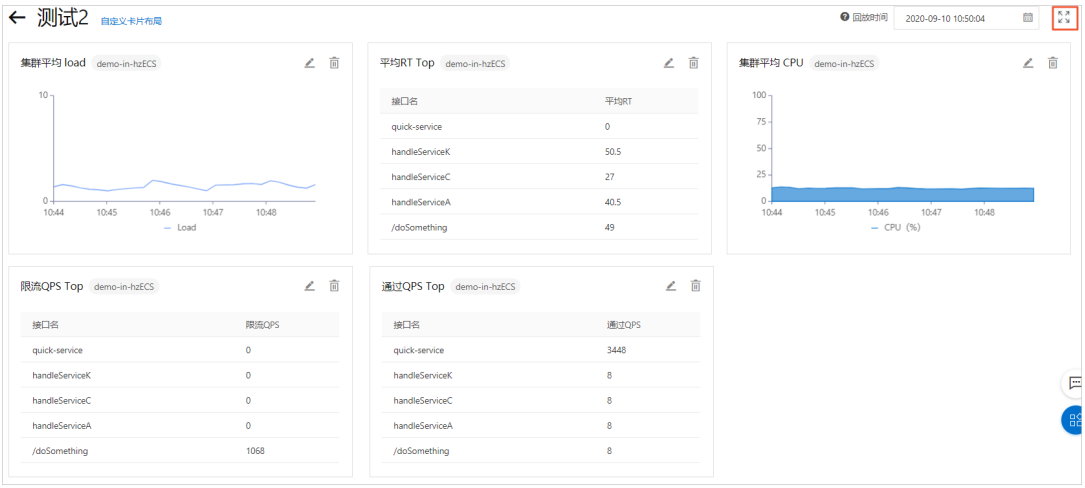
接入应用防护和接入网关防护的应用均可添加至流量大盘。

操作步骤

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在左侧导航栏选择**流量防护 > 流量大盘**。
3. 在**流量大盘**页面单击**创建大盘**。
4. 在**创建大盘**对话框中完成以下操作：
   i. 输入**大盘名**。
   ii. 选择配置流量大盘的应用。
   展示的应用包括接入应用防护和网关防护的应用。
   iii. 在双选框左侧面板中选择需查看的接口，然后单击>。
   所选择的接口将出现在双选框右侧面板中。
   iv. 单击**确认**。

执行结果

- 操作结束后，添加的大盘将显示在**流量大盘**页面。
- 单击目标流量大盘右上角的编辑图标，可对大盘进行编辑操作。
- 单击目标流量大盘，即可展示该大盘所添加接口的通过QPS、拒绝QPS、异常QPS、RT和并发数等信息。
- 在页面右上角单击展开图标，即可在全屏模式查看流量大盘。



13.Mesh防护

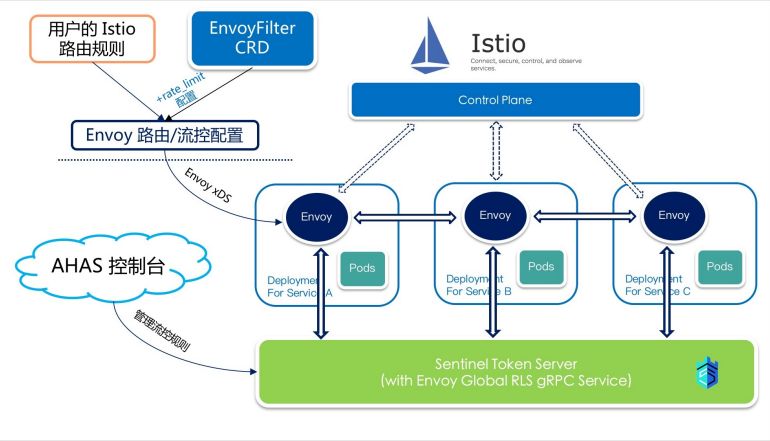
AHAS为网格化应用提供控制全局流量的能力，例如Istio、Envoy服务网格。本文介绍如何在AHAS中接入Istio服务网格（包括阿里云服务网格ASM）、Envoy集群并为其配置流控规则。

前提条件

已开通AHAS专业版，若没有开通，请进入开通页面。

背景信息

阿里云服务网格ASM（Alibaba Cloud Service Mesh）提供了一个全托管式的服务网格平台，兼容于社区Istio开源服务网格，用于简化服务的治理。更多信息，请参见什么是服务网格ASM？。AHAS Mesh防护基于Envoy Global Rate Limiting Service（RLS）接口实现，防护原理如下图所示。



> **说明** Service Mesh与AHAS Token Server的通信会带来一定的网络开销，响应时间可能会略有上升（同个Region一般会在2 ms~5 ms左右）。

操作步骤

1. 登录AHAS控制台，在左侧导航栏选择**流量防护 > Mesh防护**。
2. 单击页面右上角的**申请Mesh流控集群**。
3. 在**防护详情**页面，填写**集群名称**等信息。

参数	描述
集群名称	填写集群的名称。集群名称应包含1~63个字符。
Envoy流控domain	流控服务的域，需要与步骤5中要配置的EnvoyFilter里面的domain相对应。
集群预计QPS	该集群内需要流控的接口预估的最大QPS，代表可能到来的最大流量，而非限流阈值，用于为Token Server自动分配提供参考。

4. 单击**下一步**，集群申请成功。
5. 单击**Mesh流控接入**，可选择接入Istio或Envoy。
   - **接入Istio**
     AHAS为Istio服务网格提供集群流控的能力。要为Istio集群接入AHAS Mesh流控，需要为Istio创建对应的EnvoyFilter CRD配置。

a. 单击**Istio接入**页签。

b. 复制代码，将其粘贴至新建的YAML文件中。

c. 在Istio集群中，通过执行 `kubectl apply` 命令使其生效，完成接入。

> **说明** 目前Istio流控基于Envoy global rate limiting实现，支持根据来源集群、目标集群、Header值等请求属性进行流控，详情请参见Envoy相关文档。

- **接入Envoy**

  AHAS为Envoy HTTP流量提供集群流控的能力。Envoy接入需要按下面步骤编辑Envoy配置文件。

  a. 单击**Envoy接入**页签。

  b. 复制页面中第一步的示例代码，为需要进行流控的路由（route）所在的VirtualHost配置流控项（rate_limits），流控项由一系列的流控参数rate_limit_action构成。

  > **说明**
  > - action里面的generic_key需要配置成AHAS集群流控的license，否则流控将不会生效。该项将不会作为流控的判断属性。当前集群流控的license会在页面的代码中显示。
  > - Envoy目前支持根据来源集群、目标集群、通用标识（generic_key）、Header值等请求属性进行流控，详情请参考Envoy相关文档。

  c. 复制页面中第二步的示例代码，在Clusters中添加AHAS Mesh流控Cluster，配置对应的RLS token server IP和端口。

  d. 复制页面中第三步的示例代码，在http_filters中添加envoy.rate_limit配置，指定上一步配置的AHAS Mesh流控Cluster。

  e. 配置文件修改完成后，重启Envoy或通过xDS动态reload配置文件，完成接入。

6. 在**集群规则配置**区域，单击**添加**，进入**规则配置**对话框，为目标集群添加规则。

参数	描述
规则名称	用于标识该规则，不超过64个字符，规则名称不能重复。
阈值	集群阈值，满足流控条件的请求量到达该阈值后会被拒绝。
统计窗口时长	集群流控统计的时间窗口长度，取值范围为1秒~24小时。
匹配规则	流控针对的请求属性的匹配规则，如针对不同来源的请求分别限制不同的规则。目前支持以下匹配规则（与Envoy descriptor保持一致）： ○ 来源集群：source_cluster。 ○ 目标集群：destination_cluster。 ○ 通用标识：generic_key。 ○ 自定义：自定义Key，名称不可重复。

7. 单击**确定**，完成规则创建。

   执行结果：

触发Mesh中的服务调用，观察服务的返回值来判断流控是否生效。返回 `429 Too Many Requests` 状态码即表示流控成功。

14.SDK 使用手册

14.1. SDK参考概述

通过SDK接入方式将应用接入AHAS应用防护后，如果默认的配置无法满足您的业务需求，可以使用AHAS应用防护的SDK可以配置对代码块的流控、配置应用被流控降级后的行为以及配置扩展接口等。

配置对代码块的流控

AHAS是围绕着资源来工作的。可通过定义资源来实现对代码块的流控。定义资源后，在AHAS控制台配置相应的规则即可生效。可通过以下几种方式来定义资源：

- 注解方式定义资源
- 抛出异常的方式定义资源
- 返回布尔值方式定义资源
- 异步调用支持
- 主流框架的默认适配

定义资源详情请参见定义资源。

配置触发规则后的逻辑

当应用触发流控、降级或系统规则时，默认抛出 `BlockException` 异常类的子类（触发流控规则，则抛出流控异常 `FlowException` ；触发降级规则，则抛出降级异常 `DegradeException` ）。

若默认配置不能满足您的需求，可通过以下几种方式配置应用触发流控降级规则后的逻辑。

- 注解方式：适用于使用自定义埋点的Spring Boot应用接入、自定义埋点接入和注解接入等方式。
- Web Servlet Filter：适用于使用HTTP埋点的Spring Boot应用接入和Web应用接入等方式。
- Dubbo Adapter：适用于Dubbo应用接入方式。

配置触发规则后的逻辑详情请参见配置触发规则后的逻辑。

常用类、方法及扩展接口

AHAS应用防护提供了常用类及其扩展方法和扩展接口，详情请参见常用类及其方法和扩展接口。

更多信息

- AHAS提供了样例工程来帮助您体验应用防护功能，详情请参见样例工程。
- 可以通过查看日志，来快速查看单机运行情况，排查问题。日志详情请参见重要日志。

14.2. 定义资源

若您需要对特定的方法或者代码块进行流控，可以使用定义资源来实现。AHAS 提供了 5 种定义资源的方法，定义资源后，在 AHAS 控制台为应用配置相应规则即可生效。

背景信息

AHAS 是围绕着资源来工作的。编码的时，只需关注如何定义资源，即哪些方法或代码块可能需要保护，而无需关注这个资源要如何保护。可通过定义资源来实现对代码块的流控，定义资源方式如下：

- 注解方式定义资源
- 抛出异常的方式定义资源
- 返回布尔值方式定义资源
- 异步调用支持
- 主流框架的默认适配

方式一：注解方式定义资源

通过 `@SentinelResource` 注解定义资源并配置 `blockHandler` 和 `fallback` 函数来进行限流之后的处理。

示例：

```
// 原本的业务方法.
@SentinelResource(blockHandler = "blockHandlerForGetUser")
public User getUserById(String id) {
    throw new RuntimeException("getUserById command failed");
}
// blockHandler 函数，原方法调用被限流/降级/系统保护的时候调用
public User blockHandlerForGetUser(String id, BlockException ex) {
    return new User("admin");
}
```

> **说明** `blockHandler` 函数会在方法触发限流、降级或系统保护规则的时候调用，而 `fallback` 函数仅会在原方法被降级时作为 fallback 方法，其它时候不会被调用。

更多信息请参见 Sentinel 注解支持文档。

方式二：抛出异常的方式定义资源

使用抛出异常的方式定义资源后，当资源发生了限流之后会抛出 `BlockException` 。您可以按需捕捉异常，并进行限流之后的逻辑处理。示例代码如下：

```
Entry entry = null;
// 务必保证finally会被执行
try {
  // 资源名可使用任意有业务语义的字符串
  entry = SphU.entry("自定义资源名");
  // 被保护的业务逻辑
  // do something...
} catch (BlockException ex) {
  // 资源访问阻止，被限流或被降级
  // 进行相应的处理操作
} finally {
  if (entry != null) {
    entry.exit();
  }
}
```

> **注意** `SphU.entry(xxx)` 需要与 `entry.exit()` 方法需匹配调用，否则会导致调用链记录异常，抛出 `ErrorEntryFreeException` 异常

方式三：返回布尔值方式定义资源

使用返回布尔值方式定义资源方式后，当资源发生了限流之后会返回 `false` 。可以根据返回值，进行限流之后的逻辑处理。示例代码如下：

```
// 资源名可使用任意有业务语义的字符串
  if (SphO.entry("自定义资源名")) {
    // 务必保证finally会被执行
    try {
      /**
      * 被保护的业务逻辑
      */
    } finally {
      SphO.exit();
    }
  } else {
    // 资源访问阻止，被限流或被降级
    // 进行相应的处理操作
  }
```

方式四：异步调用支持

AHAS 支持异步调用链路的统计。在异步调用中，需要通过 `SphU.asyncEntry(xxx)` 方法定义资源，并通常需要在异步的回调函数中调用 `exit` 方法。示例如下：

```
try {
    AsyncEntry entry = SphU.asyncEntry(resourceName);
    // 异步调用
    doAsync(userId, result -> {
        try {
            // 在此处处理异步调用的结果
        } finally {
            // 在回调结束后 exit
            entry.exit();
        }
    });
} catch (BlockException ex) {
    // Request blocked
    // Handle the exception (e.g. retry or fallback)
}
```

`SphU.asyncEntry(xxx)` 不会影响当前调用线程的 Context，因此以下两个 entry 在调用链上是平级关系（处于同一层），而不是嵌套关系：

```
// 调用链类似于：
// -parent
// ---asyncResource
// ---syncResource
asyncEntry = SphU.asyncEntry(asyncResource);
entry = SphU.entry(normalResource);
```

若在异步回调中需要嵌套其它的资源调用（无论是 `entry` 还是 `asyncEntry`），只需要借助 Sentinel 提供的上下文切换功能，在对应的地方通过 `ContextUtil.runOnContext(context, f)` 进行 Context 变换，将对应资源调用处的 Context 切换为生成的异步 Context，即可维持正确的调用链路关系。示例如下：

```
public void handleResult(String result) {
    Entry entry = null;
    try {
        entry = SphU.entry("handleResultForAsync");
        // Handle your result here
    } catch (BlockException ex) {
        // Blocked for the result handler
    } finally {
        if (entry != null) {
            entry.exit();
        }
    }
}
public void someAsync() {
    try {
        AsyncEntry entry = SphU.asyncEntry(resourceName);
        // Asynchronous invocation
        doAsync(userId, result -> {
            // 在异步回调中进行上下文变换，通过 AsyncEntry 的 getAsyncContext 方法获取异步 Context
            ContextUtil.runOnContext(entry.getAsyncContext(), () -> {
                try {
                    // 此处嵌套正常的资源调用
                    handleResult(result);
                } finally {
                    entry.exit();
                }
            });
        });
    } catch (BlockException ex) {
        // Request blocked
        // Handle the exception (e.g. retry or fallback)
    }
}
```

此时的调用链如下：

```
-parent
---asyncInvocation
-----handleResultForAsync
```

普通资源与异步资源之间嵌套示例请参见 AsyncEntryDemo.java。

方式五：主流框架的默认适配

为了减少开发的复杂程度，AHAS 对主流框架进行了适配，详情请参见支持组件列表。使用时只需要引入对应的依赖，它们框架的方法和服务都会自动被定义为资源，无需修改现有代码。

14.3. 配置触发规则后的逻辑

若默认配置不能满足您的需求时，您可以自定义应用触发流控、降级或系统规则后的逻辑。本文将介绍适用于SDK应用的逻辑配置方法。

背景信息

当应用触发流控、降级或系统规则时，默认抛出 `BlockException` 异常类的子类（触发流控规则，则抛出流控异常 `FlowException` ；触发降级规则，则抛出降级异常 `DegradeException` ）。

默认配置中，可以通过 `BlockException.isBlockException(Throwable t);` 方法判断是否为流控降级异常。

若默认配置不能满足您的需求，可通过以下几种方式配置应用触发流控、降级或系统规则后的逻辑：

- 注解方式：适用于使用自定义埋点的Spring Boot应用接入、自定义埋点接入和注解接入等方式。
- Web Servlet Filter：适用于使用HTTP埋点的Spring Boot应用接入和Web应用接入等方式。
- Dubbo Adapter：适用于Dubbo应用接入方式。

注解方式

若通过注解方式定义的资源，需要在 `@SentinelResource` 注解上给方法配置 `fallback` 函数以及 `blockHandler` 函数，来进行逻辑处理。

示例：

```
public class TestService {
    // 对应的 `handleException` 函数需要位于 `ExceptionUtil` 类中，并且必须为static函数。
    @SentinelResource(value = "test", blockHandler = "handleException", blockHandlerClass = {Ex
ceptionUtil.class})
    public void test() {
        System.out.println("Test");
    }
    // 原函数。
    @SentinelResource(value = "hello", blockHandler = "exceptionHandler", fallback = "helloFall
back")
    public String hello(long s) {
        return String.format("Hello at %d", s);
    }
    // Fallback函数，函数签名与原函数一致或加一个Throwable类型的参数。
    public String helloFallback(long s) {
        return String.format("Halooooo %d", s);
    }
    // Block异常处理函数，参数最后多一个BlockException，其余与原函数一致。
    public String exceptionHandler(long s, BlockException ex) {
        // Do some log here.
        ex.printStackTrace();
        return "Oops, error occurred at " + s;
    }
}
```

相关配置：

- `value` ：资源名称（必需项）。
- `entryType` ：entry类型（可选项），默认为 `EntryType.OUT` 。
- `blockHandler` / `blockHandlerClass` ： `blockHandler` 对应处理 `BlockException` 的函数名称，为可选项。blockHandler函数访问范围需要是 `public` ，返回类型需要与原方法相匹配，参数类型需要和原方法相匹配并且最后加一个额外的类型为 `BlockException` 的参数。blockHandler函数默认需要和原方法在同一个类中。若希望使用其它类的函数，则可以指定 `blockHandlerClass` 为对应的类的 `Class` 对象，对应的函数必需为static函数，否则无法解析。
- `fallback` ：fallback函数名称（可选项）。用于在抛出异常的时候提供fallback处理逻辑。fallback函数可以针对除了 `exceptionsToIgnore` 里排除掉的异常类型之外的异常进行处理。fallback函数签名和位置要求：

- 返回值类型必须与原函数返回值类型一致。
- 方法参数列表需要和原函数一致，或者可以额外多一个 `Throwable` 类型的参数用于接收对应的异常。
- fallback函数默认需要和原方法在同一个类中。若需使用其它类的函数，则可以指定 `fallbackClass` 为对应的类的 `Class` 对象，注意对应的函数必需为static函数，否则无法解析。

- `defaultFallback` （since 1.6.0）：默认的fallback函数名称（可选项），通常用于通用的fallback逻辑。默认fallback函数可以针对所有类型的异常（除了 `exceptionsToIgnore` 里面排除掉的异常类型）进行处理。若同时配置了fallback和defaultFallback，则只有fallback会生效。defaultFallback函数签名要求：
  - 返回值类型必须与原函数返回值类型一致。
  - 方法参数列表需要为空，或者可以额外多一个 `Throwable` 类型的参数用于接收对应的异常。
  - defaultFallback函数默认需要和原方法在同一个类中。若希望使用其他类的函数，则可以指定 `fallbackClass` 为对应的类的 `Class` 对象，注意对应的函数必需为static函数，否则无法解析。

> **说明** 若blockHandler和fallback都进行了配置，则被限流降级而抛出 `BlockException` 时只会进入 `blockHandler` 处理逻辑。若未配置 `blockHandler` 、 `fallback` 和 `defaultFallback` ，则被限流降级时会将 `BlockException` 直接抛出。

Web Servlet Filter

默认情况下，当请求被限流时会返回默认的提示页面，提示信息为： `Blocked by Sentinel (flow limiting)` 。

您可以通过以下两种方式来设定自定义的跳转URL，当请求触发限流、降级或系统规则时会自动跳转至设定好的URL。

- 通过 `WebServletConfig.setBlockPage(blockPage)` 方法设定自定义的跳转URL。
- 通过UrlBlockHandler接口编写定制化的限流处理逻辑，然后将其注册至WebCallbackManager中来实现，详情可参见Web Servlet Filter扩展接口。

Dubbo Adapter

Sentinel Dubbo Adapter支持配置全局的fallback函数，可以在Dubbo服务触发限流、降级或系统规则时进行相应的fallback处理。可通过以下两种方式来实现：

- 自定义DubboFallback接口，然后通过 `DubboFallbackRegistry` 注册。默认情况会直接将 `BlockException` 包装后抛出。
- 配合Dubbo的fallback机制来实现。

14.4. 常用类及其方法

本文为您介绍流控降级常用类和方法。

流控降级异常类BlockException

在Sentinel中所有流控降级相关的异常都是异常类 `BlockException` 的子类：

- 流控异常： `FlowException`
- 降级异常： `DegradeException`
- 系统保护异常： `SystemException`
- 热点参数限流异常： `ParamFlowException`

您可以通过以下方法判断是否为流控降级异常：

```
BlockException.isBlockException(Throwable t);
```

资源定义类SphU / SphO

`SphU` 和 `SphO` 是两个常用的用于资源定义的工具类。其中 `SphU` 以try-catch的形式定义资源，而 `SphO` 以if-else的形式定义资源。

`SphU` 类包含以下几组静态方法：

传入资源名定义资源：

- `public static Entry entry(String name) throws BlockException`
- `public static Entry entry(String name, int batchCount) throws BlockException`
- `public static Entry entry(String name, EntryType type) throws BlockException`
- `public static Entry entry(String name, EntryType type, int batchCount) throws BlockException`
- `public static Entry entry(String name, EntryType type, int batchCount, Object... args) throws BlockException`

其中资源名为传入的 `name` 。

传入Method对象定义方法资源：

- `public static Entry entry(Method method) throws BlockException`
- `public static Entry entry(Method method, int batchCount) throws BlockException`
- `public static Entry entry(Method method, EntryType type) throws BlockException`
- `public static Entry entry(Method method, EntryType type, int count) throws BlockException`
- `public static Entry entry(Method method, EntryType type, int count, Object... args) throws BlockException`

其中资源名将从传入的 `Method` 对象解析，格式为 `类名:方法签名` ，如 `com.alibaba.csp.sentinel.demo.DemoService:foo(java.lang.String)` 。

异步资源定义：

- `public static AsyncEntry asyncEntry(String name) throws BlockException`
- `public static AsyncEntry asyncEntry(String name, EntryType type) throws BlockException`
- `public static AsyncEntry asyncEntry(String name, EntryType type, int batchCount, Object... args) throws BlockException`

其中的参数解释如下。

参数名	类型	解释	默认值
entryType	`EntryType`	资源调用的流量类型，是入口流量（ `IN` ）、出口流量（ `OUT` ）、内部调用（ `INTERNAL` ）。注意系统自适应保护只对 `IN` 类型生效。	`EntryType.OUT`
resourceType	`Int`	资源调用分类，如Web/RPC/DB_SQL。	`COMMON` (0)
batchCount	`Int`	本次资源调用请求的Token数目（即算作几次调用）。	1

参数名	类型	解释	默认值
args	`Object[]`	传入的参数，用于热点参数限流。	无

返回值类型：

- 普通的资源定义返回 `Entry` 对象，代表本次资源的调用。
- 异步资源定义返回 `AsyncEntry` 对象，代表本次异步资源的调用。

更多使用请参见定义资源。

托管资源定义类SentinelWrapper

> **说明** SentinelWrapper在Java SDK 1.8.0及以上版本引入。

`SentinelWrapper` 用于定义托管执行的资源埋点。`SentinelWrapper` 与 `SphU/SphO` 的不同在于 `SentinelWrapper` 需要您提供要执行的函数，并托管执行（与@SentinelResource注解方式类似），可以支持自动重试、超时熔断等机制。`SentinelWrapper` 的主要函数有两个：

- `execute`：在出现异常（包括被限流）时会直接抛出异常。
- `executeWithFallback`：可以接受一个 `fallback` 函数来处理异常，返回正常的结果。

托管资源定义类SentinelWrapper如下：

- `public static <R> R execute(Callable<R> func, String resource, EntryType trafficType, int resourceType) throws Exception`
- `public static <R> R execute(Callable<R> func, String resource, EntryType trafficType, int resourceType, Object[] args) throws Exception`
- `public static <R> R executeWithFallback(Callable<R> func, CheckedFunction<Throwable, R> fallbackFunction, String resource, EntryType trafficType) throws Exception`
- `public static <R> R executeWithFallback(Callable<R> func, CheckedFunction<Throwable, R> fallbackFunction, String resource, EntryType trafficType, int resourceType) throws Exception`
- `public static <R> R executeWithFallback(Callable<R> func, CheckedFunction<Throwable, R> fallbackFunction, String resource, EntryType trafficType, int resourceType, Object[] args) throws Exception`

参数名	类型	解释	默认值
func	`Callable<R>`	用户要执行的函数，结果会体现在返回值上。	无（必传）
fallbackFunction	`CheckedFunction<Throwable, R>`	fallback函数，当出现异常时通过这个函数生成fallback结果。	无
entryType	`EntryType`	资源调用的流量类型，是入口流量（`IN`）、出口流量（`OUT`）、内部调用（`INTERNAL`）。注意系统自适应保护只对 `IN` 类型生效。	`EntryType.OUT`

参数名	类型	解释	默认值
resourceType	`Int`	资源调用分类，如Web/RPC/DB_SQL。	`COMMON` (0)
args	`Object[]`	传入的参数，用于热点参数限流。	无

Entry

`Entry` 对象代表某一次资源调用，通过 `SphU` 或 `SphO` 定义资源后会返回此对象。主要方法：

- `public void exit() throws ErrorEntryFreeException` ：表示资源调用结束，需要与 `entry` 方法成对出现。

异常描述：

- `ErrorEntryFreeException` ：当前资源调用exit与entry不匹配会抛出此异常。资源的entry与exit必须成对出现。

业务异常记录类Tracer

业务异常记录类Tracer用于记录业务异常。相关方法：

- `public static void trace(Throwable e)` ：记录业务异常（非 `BlockException` 异常）。
- `public static void trace(Throwable e, int count)` ：记录业务异常，异常数目为传入的 `count` 。

如果用户通过 `SphU` 或 `SphO` 手动定义资源，则Sentinel不能感知上层业务的异常，需要手动调用 `Tracer.trace(ex)` 来记录业务异常，否则对应的异常不会统计到Sentinel异常计数中。

注解方式定义资源支持自动统计业务异常，无需手动调用 `Tracer.trace(ex)` 来记录业务异常。Web Servlet适配、Dubbo适配也会自动统计业务异常，无需手动统计。

上下文工具类ContextUtil

相关方法：

标识进入调用链入口（上下文）：

以下静态方法用于标识调用链路入口，用于区分不同的调用链路：

- `public static Context enter(String contextName)`
- `public static Context enter(String contextName, String origin)`

其中 `contextName` 代表调用链路入口名称（上下文名称）， `origin` 代表调用来源名称。默认调用来源为空。返回值类型为 `Context` ，即生成的调用链路上下文对象。

> **说明** `ContextUtil.enter(xxx)` 方法仅在调用链路入口处生效，即仅在当前线程的初次调用生效，后面再调用不会覆盖当前线程的调用链路，直到exit。 `Context` 存于ThreadLocal中，因此切换线程时可能会丢掉，如果需要跨线程使用可以结合 `runOnContext` 方法使用。

流控规则中若选择“流控方式”为“链路”方式，则入口资源名即为上面的 `contextName` 。

退出调用链（清空上下文）：

- `public static void exit()` ：该方法用于退出调用链，清理当前线程的上下文。

获取当前线程的调用链上下文：

- `public static Context getContext()` ：获取当前线程的调用链路上下文对象。

在某个调用链上下文中执行代码：

- `public static void runOnContext(Context context, Runnable f)`：常用于异步调用链路中context的变换。

14.5. 扩展接口

本文为您介绍Web Servlet Filter、Dubbo Adapter等扩展接口。

Web Servlet Filter

- **自定义限流页面/处理逻辑**
  默认情况下，当请求被限流时会返回默认的提示页面。可通过三种方式设置自定义的跳转URL：
  - 方式一：`WebServletConfig.setBlockPage(blockPage)` 方法
    示例：

    ```
    //设置全局生效，被流控的所有页面都会跳转到这里。
    WebServletConfig.setBlockPage("https://www.example.test/");
    ```

  - 方式二：JVM `-Dcsp.sentinel.web.servlet.block.page=xxx`
    示例：

    ```
    //设置全局生效，被流控的所有页面都会跳转到这里。
    -Dcsp.sentinel.web.servlet.block.page=https://www.fallback.page.com/
    ```

- 方式三：更灵活的方式是定义 `UrlBlockHandler` 接口限流处理逻辑，并将其注册至 `WebCallbackManager`。
  自定义处理逻辑的示例：

```
// 全局设置一次即可，比如在某个全局的init()方法里加入。
WebCallbackManager.setUrlBlockHandler(new UrlBlockHandler() {
    @Override
    public void blocked(HttpServletRequest request, HttpServletResponse response, BlockException ex)
        throws IOException {
        // request里包含了此次请求所有的信息，可以从其中解析出URL、请求参数等。
        logger.info("blocked: " + request.getPathInfo());
        // response表示响应对象，直接向其中写fallback结果即可。
        response.sendRedirect("https://www.fallback.page.com/"); // 将请求重定向到fallback地址
    }
});
```

  返回Status 500的示例：

```
// 全局设置一次即可，比如在某个全局的init()方法里加入。
WebCallbackManager.setUrlBlockHandler(new UrlBlockHandler() {
    @Override
    public void blocked(HttpServletRequest request, HttpServletResponse response, BlockException ex)
        throws IOException {
        // request里包含了此次请求所有的信息，可以从其中解析出URL、请求参数等。
        logger.info("blocked: " + request.getPathInfo());
        // response表示响应对象，直接向其中写fallback结果即可。
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        response.getWriter().println("flow control");
    }
});
```

- **URL 资源清洗**
  Sentinel Web Servlet Filter会将每个到来的不同的URL都作为不同的资源处理，因此对于REST风格的API，需要自行实现 `UrlCleaner` 接口清洗资源，例如，将满足 `/foo/:id` 的URL都归到 `/foo/*` 资源下，然后将其注册至 `WebCallbackManager` 中。否则会导致资源数量过多，超出资源数量阈值（6000）时多出的资源的规则将**不会生效**。
  示例如下：

```
@PostConstruct
public void init() {
     // 全局注册一次即可
    WebCallbackManager.setUrlCleaner(new UrlCleaner() {
        @Override
        public String clean(String originUrl) {
             // 对originUrl进行变换，得到归一化后的URL
            if (originUrl == null || originUrl.isEmpty()) {
                return originUrl;
            }
             // 比如将满足/foo/:id的URL都归到/foo/*
            if (originUrl.startsWith("/foo/")) {
                return "/foo/*";
            }
            return originUrl;
        }
    });
}
```

- **解析请求来源**
  若希望对HTTP请求按照来源限流，则可以自己实现 `RequestOriginParser` 接口从HTTP请求中解析origin并注册至 `WebCallbackManager` 中，示例如下：

```
WebCallbackManager.setRequestOriginParser(new RequestOriginParser() {
    @Override
    public String parseOrigin(HttpServletRequest request) {
        return request.getRemoteAddr();
    }
});
```

Dubbo Adapter

Sentinel Dubbo Adapter支持配置全局的Fallback函数，可以在Dubbo服务被限流、降级或负载保护的时候进行相应的Fallback处理。用户只需要实现自定义的 `DubboFallback` 接口，并通过 `DubboFallbackRegistry` 注册即可。默认情况会直接将 `BlockException` 包装后抛出。

其它扩展接口

Sentinel提供多样化的SPI接口用于提供扩展的能力。用户可以在用同一个 `sentinel-core` 的基础上自行扩展接口实现，从而可以方便地根据业务需求给Sentinel添加自定义的逻辑。目前Sentinel提供如下的扩展点：

- 初始化过程扩展：提供 `InitFunc` SPI接口，可以添加自定义的一些初始化逻辑，如动态规则源注册等。
- Slot Chain扩展：用于给Sentinel功能链添加自定义的功能并自由编排。
- 指标统计扩展（StatisticSlot Callback）：用于扩展StatisticSlot指标统计相关的逻辑。
- Transport扩展：提供 `CommandHandler` 、 `CommandCenter` 等接口，用于对心跳发送、监控API Server进行扩展。

14.6. Java SDK 参考

14.7. 样例工程

通过试用Sentinel样例工程，您能够更快捷地了解AHAS应用防护功能。

- sentinel-demo-basic：Sentinel的基本使用示例，包括流控、降级、系统保护和异步调用资源定义等demo。
- sentinel-demo-annotation-spring-aop：Sentinel使用注解的示例。

- sentinel-demo-dubbo：在Dubbo应用中接入Sentinel的示例。
- sentinel-demo-rocketmq：在RocketMQ中接入Sentinel的示例。

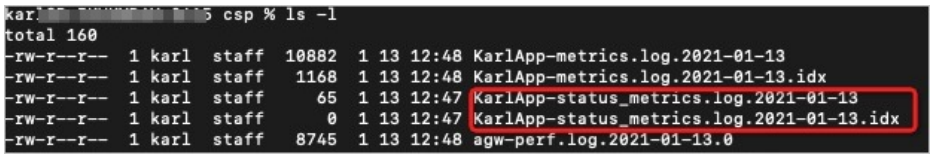
您也可以查看应用防护快速入门或更多的Sentinel样例工程集锦，熟悉AHAS应用防护（Sentinel）服务。

14.8. 重要日志

您可以通过查看日志快速查看单机运行情况，从而排查问题。本文列出了AHAS提供的重要日志，适用于所有接入AHAS应用防护的应用。

文件	路径	说明
秒级监控日志	`${user_home}/logs/csp/${app_name}-${pid}-metrics.log`	资源都会产生秒级日志，可以用来查看资源的运行情况。
*status.log日志	`/** * 写入格式： * String resource; "xx" + "_status" * long passQps; statistics[1] ---- 2xx * long blockQps; statistics[3] ---- 4xx * long successQps; statistics[0] ---- 200 * long exceptionQps; statistics[4] ---- 5xx * long rt; statistics[2] ---- 3xx * 清理工作由MetricAggregatorTask执行 */`	每分钟刷新一次，记录REST请求状态。
拦截详情日志	`{user_home}/logs/csp/sentinel-block.log`	规则生效之后，请求被拦截，就会产生对应的日志。
业务日志	`{user_home}/logs/csp/sentinel-record.log.${当天的日志}`	记录了规则的推送、接收、处理；资源调用情况，排查问题的时候会非常有帮助。
上报日志	`{user_home}/logs/csp/command-center.log.${当天的日志}`	应用和Dashboard发生通讯的记录，用于排查和Dashboard的通信问题。

您在使用SDK时会生成*status.log文件，如下图所示。



秒级监控日志

所有的资源都会产生秒级日志，它在 `${user_home}/logs/csp/${app_name}-${pid}-metrics.log` 里。格式如下：

```
1532415661000|2018-07-24 15:01:01|sayHello(java.lang.String)|12|3|4|2|295|10
```

1. `1532415661000` ：时间戳。
2. `2018-07-24 15:01:01` ：格式化之后的时间戳。
3. `sayHello(java.lang.String)` ：资源名。

4. `12`：表示到来的数量，即此刻通过Sentinel规则check的数量（passed QPS）。
5. `3`：实际该资源被拦截的数量（blocked QPS）。
6. `4`：每秒结束的资源个数（完成调用），包括正常结束和异常结束的情况（exit QPS）。
7. `2`：异常的数量。
8. `295`：资源的平均响应时间（RT）。
9. `10`：并发数。

拦截详情日志

无论限流，降级还是系统保护，它们的秒级拦截详情日志都记录在`{user_home}/logs/csp/sentinel-block.log`文件里，格式如下。

```
2014-06-20 16:35:10|1|sayHello(java.lang.String,long),FlowException,default,origin|61,0
2014-06-20 16:35:11|1|sayHello(java.lang.String,long),FlowException,default,origin|1,0
```

1. `2014-06-20 16:35:10`：时间戳。
2. `1`：序号。
3. `sayHello(java.lang.String,long)`：资源描述符。
4. `XXXException`：表示被限制的种类。`FlowException`表示被限流，`DegradeException`表示被降级，`SystemException`表示被系统保护。
5. `default`规则上配置的限制应用。
6. `origin`：实际被限制的来源应用，可能为空字符串。
7. `61,0`：61代表这一秒内限流降级发生的次数，0无含义（可忽略）。

业务日志

业务日志在`{user_home}/logs/csp/sentinel-record.log.xxx`中，包含规则的推送、接收、处理等记录。

上报日志

每一次和Dashboard的通信，都会记录在`{user_home}/logs/csp/command-center.log.${date}`日志中。可以用来排查Dashboard能否成功与机器连通等问题。

14.9. Java SDK和Java Agent版本说明

本文记录了Java SDK和Java Agent的版本发布说明。

版本说明

本文包含以下Java SDK和Java Agent版本说明：

- Java SDK
  - AHAS-Sentinel-Client版本说明
  - Spring-Boot-Starter-AHAS-Sentinel-Client版本说明
  - Spring-Cloud-Gateway-Starter-AHAS-Sentinel版本说明
  - Spring-Cloud-Zuul-Starter-AHAS-Sentinel版本说明
- Java Agent

Java SDK

AHAS-Sentinel-Client版本说明

版本号	发布时间	版本说明
1.10.8	2022年05月12日	• 修复集群流控通信监控在请求失败下可能存在内存泄漏的问题。 • 完善初始化时对ECS Metadata请求异常情况下的处理。 • 支持通过启动参数配置指定网卡上报IP。
1.10.7	2022年03月23日	支持法兰克福地域（Region）。
1.10.6	2022年01月14日	• 支持新加坡地域（Region）。 • 修复Kubernetes Request/Limit都未配置时CPU使用率监控异常的问题。
1.10.5	2021年12月24日	• 新增Spring WebFlux及gRPC适配。 • 完善agw.log日志滚动策略。
1.10.4	2021年12月10日	• RPC Fallback支持HSF/SOFARPC。 • RPC Fallback自定义类型支持泛型类型。
1.10.2	2021年12月03日	新增热点参数Block监控。
1.10.1	2021年11月02日	• 新增Web场景化规则支持。 • 完善对非Docker容器运行时的判断逻辑，修复部分K8s环境下上报机器不正确的问题。
1.9.7	2021年09月22日	• 单机流控规则支持任意统计时长。 • 优化部分小流量场景下CPU的占用。
1.9.6	2021年08月13日	• 新增政务云环境支持。 • 新增默认熔断规则支持。
1.9.5	2021年08月06日	• Dubbo适配支持在控制台动态自定义RPC Fallback行为，支持自定义返回值和异常两种方式。 • 支持集群防护通信监控。

版本号	发布时间	版本说明
1.9.4	2021年06月28日	• 优化集群流控批量请求模式的准确性。 • 修复部分容器内核版本采集不到User/Sys CPU usage监控的问题。 • Apache HttpClient适配默认资源名提取时不包含URL params/fragment，防止资源名数目过多。
1.9.3	2021年06月11日	• 修复集群流控批量请求模式在部分情况下不准确的问题，完善在不同流量分布情况下的限流准确性。 • 修复依赖导致部分版本阿里云SDK无法正常工作的问题。
1.9.2	2021年05月28日	• OkHttp适配默认资源名提取时不包含URL params，防止资源名数目过多。 • 提升初始化速度。
1.9.1	2021年05月14日	• 修复Spring Web适配模块在异步Servlet场景下可能出现的统计问题。 • 修复容器内CPU usage监控展示问题。 • 完善对JDK 1.6的支持。
1.9.0	2021年05月01日	• SDK支持JDK 6-11。 • 新增异常信息统计支持。
1.8.10	2021年04月24日	控制台支持展示具体的触发规则。
1.8.9	2021年04月09日	新增上海金融云环境支持。
1.8.8	2021年04月02日	• 新增自适应流控采样日志。 • 完善HSF adapter判断本机调用及解析来源的逻辑。
1.8.7	2021年03月12日	修复Web Fallback自定义返回文本包含中文时会出现乱码的问题。
1.8.6	2021年02月05日	• 集群流控支持自动调整单机退化阈值。 • 完善对Apache Dubbo 2.7.1~2.7.3版本的支持。
1.8.5	2021年02月01日	支持VPC环境内的MST压测。

版本号	发布时间	版本说明
1.8.4	2021年01月28日	• 新增主动降级规则，针对Servlet、Spring Web、Spring Cloud Gateway的主动降级规则支持配置自定义降级行为。 • 完善MyBatis DAO埋点支持的场景。
1.8.3	2021年01月13日	• 修复HTTP状态码统计模块线程池满时抛出异常的问题。 **说明** 建议1.8.x旧版本至少升级至此版本。 • 支持通过启动参数指定全局滑动窗口配置。
1.8.2	2021年01月09日	• 完善日志管理、清除逻辑。 • 完善对Dubbo 2.7.x版本的支持（无需额外引入依赖）。
1.8.1	2020年12月23日	• 修复HTTP、状态码统计模块可能导致CPU被占满的问题。 • 支持通过sentinel.properties文件配置AHAS License、NameSpace。
1.8.0	2020年12月16日	• 新增自动重试规则，新增SentinelWrapper托管调用埋点方式。目前SentinelWrapper埋点、注解埋点及HTTP Client适配支持自动重试规则。 • 新增HTTP接口状态码统计，支持Spring Web、Spring Cloud Gateway、Zuul 1.x。 • 新增指标统计，操作系统指标新增内存、网络、磁盘等指标种类；JVM指标新增GC、JVM内存、线程数等指标。
1.7.5	2020年12月07日	支持与AHAS Sentinel Java Agent同时使用，SDK在探测到Agent相关类存在时不会触发初始化，Web、MyBatis等相关适配模块不会重复生效。
1.7.3	2020年11月03日	修复熔断降级模块在同资源配置多个规则时可能出现无法从half-open状态中恢复的问题。

版本号	发布时间	版本说明
1.7.2	2020年10月20日	• 新增支持HSF适配模块，原生支持接入EDAS HSF服务。 • 控制台监控支持区分各个埋点的流量类型（入口流量/出口流量/内部调用）。
1.7.1	2020年09月25日	• 改进CPU使用率自适应流控策略，提高容器环境下获取CPU使用率的准确性。 • 新增一键自适应流控开关。
1.7.0	2020年09月03日	AHAS SDK公有云和专有云版本统一。
1.6.6	2020年08月28日	修复Spring Web适配在内部forward请求的场景下可能会导致统计错误的问题。
1.6.5	2020年08月07日	• 修复部分阈值配置下匀速排队模式间隔时间控制不准确的问题。 • Web适配模块默认流控逻辑支持自定义CORS配置。
1.6.4	2020年07月17日	• 集群流控支持单机大流量场景（批量请求）。 • 修复熔断降级规则阈值比例为100%时不生效的问题。
1.6.3	2020年06月12日	• 注解方式支持void返回的Fallback/defaultFallback，支持配置全局统一的Fallback。 • 优化集群流控。
1.6.2	2020年05月28日	熔断降级新增渐进式恢复策略。
1.6.1	2020年04月29日	支持VPC环境通过License绑定用户。
1.6.0	2020年04月24日	• 改进熔断降级特性：引入半开启探测模式，支持任意统计时长，RT模式改进为慢调用比例模式。 • 优化集群流控通信与异常处理。 • 支持张家口Region。 • 优化日志依赖。
1.5.5	2020年04月14日	修复在不支持的Region即使配了License也不会走公网的问题。
1.5.4	2020年03月31日	• 支持集群流控特性。 • 支持SAE方式接入。

版本号	发布时间	版本说明
1.5.3	2020年03月9日	• 添加SOFARPC埋点支持。 • 埋点资源名增加长度限制（最长1024字符）。 • MyBatis adapter新增DAO mapper方法埋点支持。 • 日志依赖优化。
1.5.2	2020年02月19日	• 去除RT统计限制。 • 去除多余日志依赖。
1.5.1	2019年12月27日	• 修复控制台应用名和实际监听规则对应的appName不同步的问题。 • MyBatis适配改进SQL含换行符情况下的处理，避免影响监控日志格式。
1.5.0	2019年11月27日	• 同步基线至Sentinel开源1.7.0版本。 • 增加通用配置功能，支持动态配置。 • 增加热点规则支持。 • 优化内存占用情况。
1.4.5	2019年10月23日	• 适配低版本的MyBatis，修复不兼容低版本MyBatis的问题。 • 支持通过参数的方式自定义Web默认限流页面的HTTP状态码。
1.4.3	无	• 系统规则maxCpuUsage模式策略调整，不再进行BBR判断。 • 修复依赖冲突问题。
1.4.2	无	修复PropertyUrlCleaner自动注册判断逻辑的问题。
1.4.1	无	• 优化系统指标信息采集方式，拉取监控的时候系统指标信息通过Dubbo Metrics采集。 • 引入MyBatis adapter支持。 **说明** 此版本开始不支持和开源依赖混用。

版本号	发布时间	版本说明
1.4.0	无	• 同步基线至Sentinel开源1.6.3版本。 • 支持文件配置URL埋点清理。
1.3.6	无	同步基线至Sentinel开源1.6.2版本。
1.3.5	无	• 同步基线至Sentinel开源1.6.1版本。 • 监控指标支持CPU使用率和load。

Spring-Boot-Starter-AHAS-Sentinel-Client版本说明

版本	发布时间	版本说明
1.10.11	2022年05月12日	• 特性对应AHAS-Sentinel-Client 1.10.8。 • Feign适配模块新增支持Spring Cloud 2020.x及以上版本。
1.10.10	2022年03月31日	• 特性对应AHAS-Sentinel-Client 1.10.7。 • 支持德国法兰克福地域（Region）。
1.10.6	2022年01月14日	• 特性对应AHAS-Sentinel-Client 1.10.6。 • 支持新加坡地域（Region）。
1.10.5	2021年12月24日	• 特性对应AHAS-Sentinel-Client 1.10.5。 • 新增Spring WebFlux框架支持。 • 支持Dubbo/HSF RPC Fallback自动提取返回类型。
1.10.2	2021年12月03日	特性对应AHAS-Sentinel-Client 1.10.2。
1.10.1	2021年11月02日	特性对应AHAS-Sentinel-Client 1.10.1。
1.9.7	2021年09月22日	特性对应AHAS-Sentinel-Client 1.9.7。
1.9.6	2021年08月13日	特性对应AHAS-Sentinel-Client 1.9.6。
1.9.5	2021年08月06日	特性对应AHAS-Sentinel-Client 1.9.5。

版本	发布时间	版本说明
1.9.4	2021年06月29日	特性对应AHAS-Sentinel-Client 1.9.4。
1.9.2	2021年05月28日	• 特性对应AHAS-Sentinel-Client 1.9.2。 • 完善对Spring Cloud E release train的支持（Feign组件）。 • 缓解首次调用耗时长的问题。
1.9.1	2021年05月14日	• 特性对应AHAS-Sentinel-Client 1.9.1。 • 完善对Spring Boot 1.x版本的支持。
1.9.0	2021年05月01日	特性对应AHAS-Sentinel-Client 1.9.0。
1.8.10	2021年04月24日	特性对应AHAS-Sentinel-Client 1.8.10。
1.8.9	2021年04月09日	特性对应AHAS-Sentinel-Client 1.8.9。
1.8.8	2021年04月02日	新增Spring RestTemplate适配。
1.8.7	2021年03月12日	• 完善Spring Cloud动态配置场景下解析AppName的逻辑。 • 特性对应AHAS-Sentinel-Client 1.8.7。
1.8.6	2021年02月05日	特性对应ahas-sentinel-client 1.8.6。
1.8.5	2021年02月01日	特性对应AHAS-Sentinel-Client 1.8.5。
1.8.4	2021年01月28日	添加Feign埋点支持（支持Spring Cloud F/G/H三大版本体系）。
1.8.3	2021年01月13日	特性对应AHAS-Sentinel-Client 1.8.3。
1.8.2	2021年01月09日	完善starter在application.properties动态配置场景下解析、更新配置的逻辑。
1.8.1	2020年12月23日	完善starter配置解析逻辑以解决部分场景下启动顺序导致配置读不到的问题。
1.8.0	2020年12月16日	特性对应AHAS-Sentinel-Client 1.8.0。

版本	发布时间	版本说明
1.7.5	2020年12月08日	特性对应AHAS-Sentinel-Client 1.7.5。
1.7.3	2020年11月03日	特性对应AHAS-Sentinel-Client 1.7.3。
1.7.2	2020年10月20日	特性对应AHAS-Sentinel-Client 1.7.2。
1.7.1	2020年09月28日	特性对应AHAS-Sentinel-Client 1.7.1。
1.6.6	2020年08月28日	特性对应AHAS-Sentinel-Client 1.6.6。
1.6.5	2020年08月07日	特性对应AHAS-Sentinel-Client 1.6.5。
1.6.4	2020年07月17日	特性对应AHAS-Sentinel-Client 1.6.4。
1.6.3	2020年06月12日	特性对应AHAS-Sentinel-Client 1.6.3。
1.6.2	2020年05月28日	特性对应AHAS-Sentinel-Client 1.6.2。
1.6.1	2020年04月29日	特性对应AHAS-Sentinel-Client 1.6.1。
1.6.0	2020年04月24日	特性对应AHAS-Sentinel-Client 1.6.0。
1.5.5	2020年04月14日	特性对应AHAS-Sentinel-Client 1.5.5。
1.5.4	2020年03月31日	特性对应AHAS-Sentinel-Client 1.5.4。
1.5.3	2020年03月09日	• 特性对应AHAS-Sentinel-Client 1.5.3。 • Spring Web Interceptor埋点资源名支持添加HTTP Mmethod前缀。 • MyBatis默认埋点改为DAO Mapper埋点，可通过开关控制。
1.5.2	2020年02月19日	特性对应AHAS-Sentinel-Client 1.5.2。

版本	发布时间	版本说明
1.5.1	2019年12月27日	• 特性对应AHAS-Sentinel-Client 1.5.1。 • 该版本开始引入后会自动注册Spring Web interceptor（仅针对Web Servlet应用）、注解aspect、MyBatis interceptor（仅针对用了MyBatis的应用）。 **说明** 用户从旧版本升级至1.5.1+ 版本时，需要去除之前自行注册的CommonFilter、SpringWebInterceptor等bean。
1.5.0	2019年11月27日	特性对应AHAS-Sentinel-Client 1.5.0。
1.3.6	无	特性对应AHAS-Sentinel-Client 1.4.2。 **说明** 此版本开始不支持和开源依赖混用。
1.3.5	无	• 特性对应AHAS-Sentinel-Client 1.4.0。 • 增加Spring Web Interceptor支持(REST API pattern自动提取）。

Spring-Cloud-Gateway-Starter-AHAS-Sentinel版本说明

版本号	版本说明
1.5.10	特性对应Spring-boot-starter-ahas-sentinel-client 1.10.11版本。
1.5.9	特性对应Spring-boot-starter-ahas-sentinel-client 1.10.10版本。
1.5.4	特性对应spring-boot-starter-ahas-sentinel-client 1.10.5版本。
1.5.2	特性对应spring-boot-starter-ahas-sentinel-client 1.10.2版本。
1.5.1	特性对应spring-boot-starter-ahas-sentinel-Client 1.10.1版本。
1.4.6	特性对应spring-boot-starter-ahas-sentinel-client 1.9.7版本。

版本号	版本说明
1.4.5	特性对应spring-boot-starter-ahas-sentinel-client 1.9.6版本。
1.4.4	特性对应spring-boot-starter-ahas-sentinel-client 1.9.5版本。
1.4.3	特性对应spring-boot-starter-ahas-sentinel-client 1.9.4版本。
1.4.2	特性对应spring-boot-starter-ahas-sentinel-client 1.9.2版本。
1.4.1	特性对应spring-boot-starter-ahas-sentinel-client 1.9.1版本。
1.4.0	特性对应spring-boot-starter-ahas-sentinel-client 1.9.0版本。
1.3.6	• 完善Spring Cloud动态配置场景下解析AppName的逻辑。 • 特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.7版本。
1.3.5	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.6版本。
1.3.4	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.5版本。
1.3.3	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.4版本，主动降级规则支持在控制台配置自定义Fallback行为。
1.3.2	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.3版本。
1.3.1	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.2版本。
1.3.0	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.1版本。
1.2.4	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.0版本。
1.2.3	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.7.5版本。
1.2.2	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.7.3版本。
1.2.1	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.7.1版本。

版本号	版本说明
1.1.7	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.6.3版本。
1.1.6	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.6.1版本。
1.1.5	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.6.0版本。
1.1.4	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.3版本。
1.1.3	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.2版本。
1.1.2	修复某些特殊情况下getServletPath为空导致API匹配不成功的问题。
1.1.1	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.1版本。
1.1.0	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.0版本。 **说明** 此版本开始不支持和开源依赖混用。

Spring-Cloud-Zuul-Starter-AHAS-Sentinel版本说明

版本号	版本说明
1.5.10	特性对应Spring-boot-starter-ahas-sentinel-client 1.10.11版本。
1.5.9	特性对应Spring-boot-starter-ahas-sentinel-client 1.10.10版本。
1.5.4	特性对应spring-boot-starter-ahas-sentinel-client 1.10.5版本
1.5.2	特性对应spring-boot-starter-ahas-sentinel-client 1.10.2版本
1.5.1	特性对应spring-boot-starter-ahas-sentinel-client 1.10.1版本。
1.4.6	• 特性对应spring-boot-starter-ahas-sentinel-client 1.9.7版本。 • 支持控制台配置防护规则触发后的自定义返回行为。
1.4.5	特性对应spring-boot-starter-ahas-sentinel-client 1.9.6版本。

版本号	版本说明
1.4.4	特性对应spring-boot-starter-ahas-sentinel-client 1.9.5版本。
1.4.3	特性对应spring-boot-starter-ahas-sentinel-client 1.9.4版本。
1.4.2	特性对应spring-boot-starter-ahas-sentinel-client 1.9.2版本。
1.4.1	特性对应spring-boot-starter-ahas-sentinel-client 1.9.1版本，完善对Spring Boot 1.x版本的支持。
1.4.0	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.9.0版本。
1.3.6	• 完善Spring Cloud动态配置场景下解析AppName的逻辑。 • 特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.7版本。
1.3.5	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.6版本。
1.3.4	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.5版本。
1.3.3	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.4版本。
1.3.2	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.3版本。
1.3.1	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.2版本。
1.3.0	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.1版本。
1.2.4	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.8.0版本。
1.2.3	支持与AHAS Sentinel Java Agent同时使用，SDK在探测到Agent相关类存在时不会触发初始化，网关适配模块不会重复生效。
1.2.2	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.7.3版本。
1.2.1	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.7.1版本。
1.1.7	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.6.3版本。

版本号	版本说明
1.1.6	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.6.1版本。
1.1.5	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.6.0版本。
1.1.4	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.3版本。
1.1.3	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.2版本。
1.1.2	修复某些特殊情况下getServletPath为空导致API匹配不成功的问题。
1.1.1	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.1版本。
1.1.0	特性对应Spring-Boot-Starter-AHAS-Sentinel-Client 1.5.0版本。 **说明** 此版本开始不支持和开源依赖混用。

Java Agent

版本号	版本说明
1.11.2	• 启动速度及运行时性能损耗优化。 • 修复部分Spring适配模块的兼容性问题。
1.11.1	新增功能开关模块，支持Switch动态日志功能，开箱即用。
1.11.0	• 增加Feign、Spring WebFlux组件支持适配。 • 网关防护支持通过控制台自定义Fallback返回。
1.10.8/1.10.9	完善流量防护模块中agw.log等日志的滚动策略。
1.10.6/1.10.7	• 支持热点规则Block参数监控。 • RPC Fallback自定义返回对象支持泛型类型。
1.10.5	• 完善部分适配模块对JDK 9+的支持，启动时无需指定add-opens参数。 • 完善探针在部分动态语言场景下的内存占用。

版本号	版本说明
1.10.4	• 新增Web场景化防护规则特性，支持Spring Web应用与传统Servlet容器。 • 完善对非Docker容器运行时的判断逻辑，修复部分K8s环境下上报机器不正确的问题。 • 修复API网关接入后规则不生效的问题。
1.10.1	大幅度优化启动速度。
1.10.0	• 从该版本开始，AHAS/PTS Java探针融合，同时具备流量防护和流量隔离能力。 • 新增Spring Data JPA框架支持，自动识别DAO层的埋点。 • Dubbo RPC支持控制台动态自定义Fallback逻辑（自定义返回值/自定义异常）。
1.9.1	• 在兼容性方面，完善对JDK 6~11的支持；支持与开源Sentinel/Spring Cloud Alibaba共用（Spring Boot应用）。 • 修复集群流控批量请求模式在部分情况下不准确的问题，完善在不同流量分布情况下的限流准确性。 • 修复部分容器内核版本采集不到User/Sys CPU usage监控的问题。
1.9.0	• 支持触发规则统计、异常统计。 • 修复Spring Web适配模块在异Servlet场景下可能出现的统计问题。
1.8.6	• Dubbo插件支持Dubbo 2.7.6+版本。 • 自适应流控新增采样日志。 • 支持上海金融云环境。 **说明** 建议1.8.x版本Java Agent至少升级至该版本。
1.8.5	• 新增主动降级规则支持。 • 支持控制台配置Web Fallback行为。
1.8.4	支持MST压测功能。
1.8.3	• 新增Dubbo 2.7.x adapter支持。 • 支持通过启动参数指定全局滑动窗口配置。 • 修复HTTP状态码统计模块线程池的问题。 • 完善对非HotSpot JVM的支持。

版本号	版本说明
1.8.2	修复HTTP状态码统计模块可能导致CPU被占满的问题。
1.8.1	• 新增HTTP接口状态码统计，支持Spring MVC/Spring Boot/Spring Cloud/Spring Cloud Gateway/Zuul 1.x。 • 新增指标统计，操作系统指标新增内存、网络、磁盘等指标种类；JVM指标新增GC、JVM内存、线程数等指标。
1.8.0	• 新增Spring Web插件支持（REST API会自动提取pattern），原有的Servlet插件默认不开启。 • 新增MyBatis插件支持，原有的JDBC插件默认不开启。 • 同步SDK 1.7.x最新相关特性（自适应流控）。
1.7.10	• 修复熔断降级模块在同资源配置多个规则时可能出现无法从half-open状态中恢复的问题。 • 修复fastjson版本导致JDK 1.7下Agent无法正常工作的问题。
1.7.9	• 集群流控支持单机大流量场景（批量请求）。 • 修复熔断降级规则阈值比例为100%时不生效的问题。
1.7.8	• 修复Spring Cloud Gateway、Zuul等网关插件不生效的问题。 • 应用appType支持通过环境变量配置。
1.7.7	修复JDK 11下agent初始化失败的问题。
1.7.6	熔断降级支持渐进式恢复策略。
1.7.5	• 熔断降级特性改进：引入半开启探测模式，支持任意统计时长，RT模式改进为慢调用比例模式。 • 集群流控通信与异常处理优化。 • 支持张家口Region。
1.7.4	• 支持集群流控特性。 • 支持SAE方式接入。
1.7.3	支持MySQL CJ Driver的SQL埋点。
1.7.2	• 支持启动参数和环境变量控制所有插件模块的开启和关闭。 • 改进SQL含换行符情况下的处理，添加SQL埋点长度限制。 • 去除RT统计限制。 • 修复部分非daemon线程导致main函数执行结束后无法退出的问题。

版本号	版本说明
1.7.1	• 同步基线至开源1.7.0版本，优化内存占用。 • 增加通用配置功能，支持簇点数目限制、context数目限制、来源数目限制、最大统计RT等基础配置项，支持动态配置。 • 增加热点规则支持。 • 默认关闭MQ相关插件以提升启动速度。
1.7.0	新增Memcached (spymemcached)、gRPC、Jedis、RocketMQ（callback模式）、RabbitMQ插件支持。 **说明** MQ插件可能会导致应用启动速度变慢。
1.6.0	新增MySQL（5.x driver）或Oracle SQL埋点支持。

15.参考信息

15.1. 应用防护原则概述

您在使用该功能前，请先了解以下原则。

AHAS应用防护原则

- AHAS是围绕着资源来工作的。
- 编码时，只需要关心如何定义资源，即哪些方法、代码块需要保护，而不需要关注如何保护这个资源。
- 通过添加规则来保护资源，规则添加即时生效。

规则配置原则

- 按照应用处理能力进行流控：
  - 按服务提供方流控原则，详情请参见服务提供方或消费方流控。
  - 削峰填谷原则，详情请参见削峰填谷。
  - 冷启动原则，详情请参见Warm Up（冷启动）。
  - 联动控制原则，详情请参见关联限流。
- 强依赖隔离原则，详情请参见强依赖隔离。
- 弱依赖降级原则，详情请参见弱依赖降级。
- 系统保护原则，详情请参见系统防护。

15.2. 查看License

License作为客户端的启动参数，用于关联客户端跟阿里云账户的关系。本文介绍如何查看License。

背景信息

若用户的机器是阿里云在VPC网络环境内，且部署在杭州、上海、北京、深圳、张家口这些地域内，则无需配置License参数，默认会关联到机器所属用户下。其他情况下，用户则需要配置License参数关联到AHAS公网地域，才能使用AHAS。

查看License

1. 登录AHAS控制台，在左侧导航栏选择**流量防护 > 应用防护**。
2. 在**应用防护**页面右上角单击**查看License**。

注意事项

- 2020年05月20日之前开通AHAS的用户，不同地域License可能不同步，若需要接入到AHAS公网，请先在AHAS控制台页面的左上角选择公网，再查看License。

- License属于用户敏感信息，请勿泄露给他人使用。

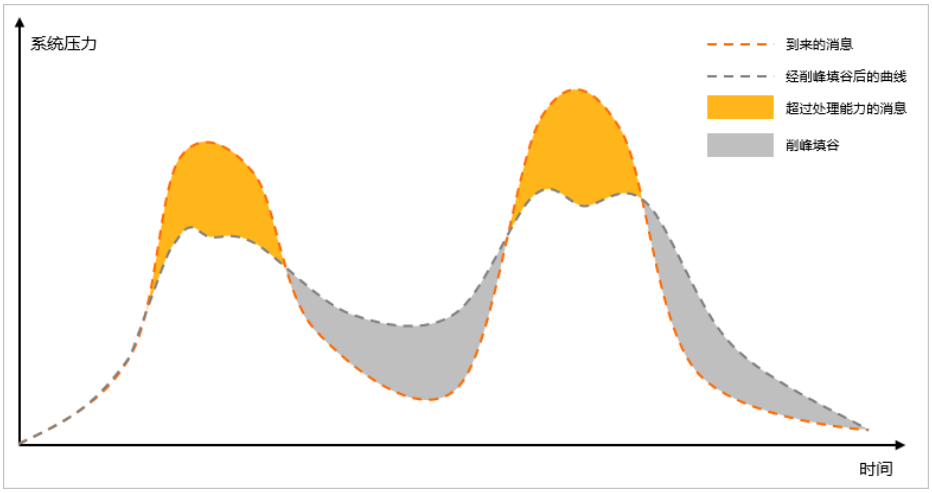
15.3. 应用防护方法

15.3.1. 削峰填谷

当消费端请求骤增时，可以为其配置排队等待的流控规则，以稳定的速度逐步处理这些请求，起到削峰填谷的效果，从而避免流量骤增造成系统负载过高。

背景信息

在实际应用中，收到的请求是没有规律的。例如：某应用的处理请求的能力是每秒10个。在某一秒，突然到来了30个请求，而接下来两秒，都没有请求到达。在这种情况下，如果直接拒绝20个请求，应用在接下来的两秒就会空闲。所以，需要把骤增的请求平均到一段时间内，让系统负载保持在请求处理水位之内，同时尽可能地处理更多请求。



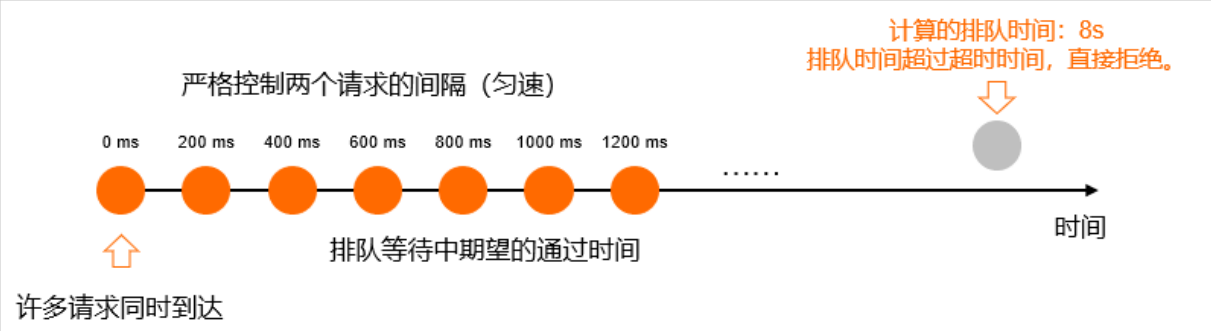
上图中，黄色的部分代表超出消息处理能力的部分。把黄色部分的消息平均到之后的空闲时间去处理，这样既可以保证系统负载处在一个稳定的水位，又可以尽可能地处理更多消息。通过配置流控规则，可以达到消息匀速处理的效果。

功能原理

AHAS流控降级的排队等待功能，可以把骤增的大量请求匀速分配，以固定的间隔时间让请求通过，起到“削峰填谷”的效果，从而避免流量骤增造成系统负载过高的情况。堆积的请求将会被排队处理，当请求的预计排队时间超过最大超时时长时，AHAS将拒绝这部分超时的请求。

例如：配置匀速模式下请求QPS为5，则每200 ms处理一条请求，多余的处理任务将排队；同时设置了超时时间为5s，则预计排队时长超过5s的处理任务将被直接拒绝。具体操作步骤，请参见新建流控规则。

示意图如下：



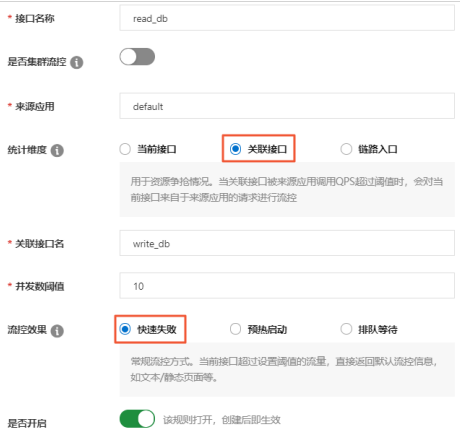
15.3.2. 关联限流

使用关联限流策略，可以避免具有关联关系的资源之间过度的争抢，造成的资源不可用问题。
当两个资源之间具有资源争抢或者依赖关系的时候，这两个资源便具有了关联。例如对数据库同一个字段的读操作和写操作存在争抢，读的速度过高会影响写得速度，写的速度过高会影响读的速度。如果放任读写操作争抢资源，则争抢本身带来的开销会降低整体的吞吐量。可使用关联限流来避免具有关联关系的资源之间过度的争抢。

示例

`read_db` 和 `write_db` 这两个资源分别代表数据库读写。给 `read_db` 设置以下规则来达到写优先的目的。具体操作步骤请参见配置流控规则常用场景。

当写库操作过于频繁时，读数据的请求会被限流。 `read_db` 会在 `write_db` 资源的QPS超过10之后，调用被拒绝。



15.3.3. Warm Up（冷启动）

对于长期处于低水位状态的系统，可以使用 Warm Up（冷启动）功能来避免流量骤增导致水位瞬间升高系统不可用的情况。

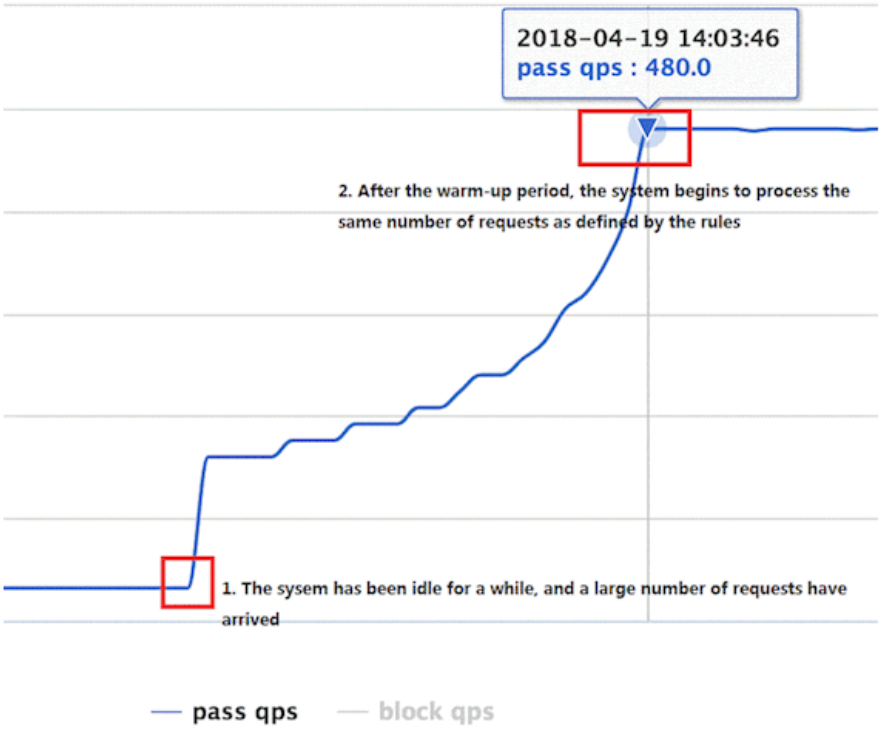
功能原理

Warm Up，即冷启动 / 预热的方式。当系统长期处于低水位的情况下，若流量突然增加，可能会把系统水位瞬间拉高把系统压垮。通过配置冷启动规则，可以让通过的流量缓慢增加，在一定时间内逐渐增加到阈值上限，给冷系统一个预热的时间，避免冷系统被压垮。

冷启动，参考了 Guava 的算法，通过随时调整斜率，把流量在指定的时间之类缓慢调整到特定的阈值。

示例

若对系统设置**流控模式**为直接，**流控方式**为 Warm Up，**预热时间**为 200 的流控规则，具体操作步骤，参见新建流控规则。设置规则后，可以看到流量的增长趋势如下图所示：



15.3.4. 服务提供方或消费方流控

AHAS 应用防护可以根据服务提供方的能力和服务消费方的分配能力进行流量控制。其中服务提供方（Service Provider）是指对外提供请求的服务或应用；服务消费方（Service Consumer）是指调用该服务的下游应用。

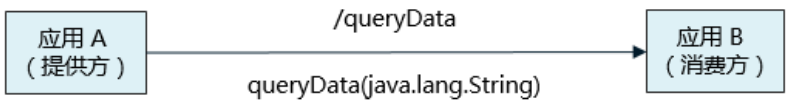
根据服务提供方限流

为了保护服务提供方不被激增的流量拖垮影响稳定性，您可以为其配置 QPS 模式的流控规则。当每秒的请求量超过设定的阈值时，AHAS 将拒绝多余的请求。提供方限流可以分为服务接口限流和服务方法限流。

- 服务接口限流：适用于整个服务接口的 QPS 不超过一定数值的情况。例如：为对应服务接口资源配置 QPS 阈值。
- 服务方法限流：适用于服务的某个方法的 QPS 不超过一定数值的情况。例如：为对应服务方法资源配置 QPS 阈值。

示例：

若应用 A 为服务提供方，其 Web 接口 `/queryData` 对应的方法是 `queryData(java.lang.String)`。假设应用 A 最多每秒只能承受 10 次调用。若调用次数超过，则应用 A 宕机。



针对此类场景，可以对应用 A 按服务方法粒度设置流控规则，限制 `queryData(java.lang.String)` 方法每秒最多只能被调用 10 次，具体操作步骤，参见新建流控规则。若超过阈值，消费方将会收到一个 `BlockException` 异常，并且快速返回。

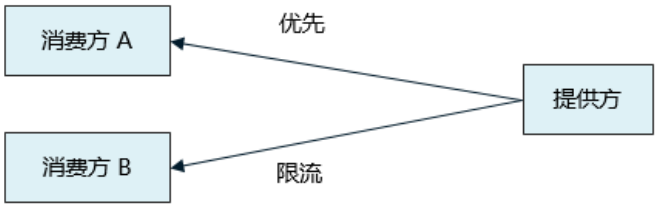
根据服务消费方限流

根据消费方限流是指根据调用方的需求来分配服务提供方的处理能力。

> **说明** 若限流规则未设置调用方（default），则该限流规则对所有调用方生效。若限流规则设置了调用方，则限流规则仅对指定调用方生效。

示例：

有两个服务消费者 A 和 B 都向服务提供方发起调用请求，我们希望优先服务消费方 A，而只对来自消费方 B 的请求进行限流。通过对消费方 B 设置限流规则（limitApp）来实现这个目的。



对于默认框架例如 Dubbo，AHAS 会自动解析 Dubbo 消费方的 `application name` 作为调用方名称（origin），在进行资源保护的时候都会带上调用方名称。而对于非默认框架，只需要在 Sentinel 原有的代码中加入 `ContextUtil.enter(resourceName, origin)` 和 `ContextUtil.exit()` 即可。示例代码如下：

```
Entry entry = null;
//这里代表消费者 A 的调用
ContextUtil.enter(queryData, "A");
try {
  // 资源名可使用任意有业务语义的字符串
  entry = SphU.entry("queryData");
  // 被保护的业务逻辑
  // do something...
} catch (BlockException e1) {
  // 资源访问阻止，被限流或被降级
  // 进行相应的处理操作
} finally {
  if (entry != null) {
    entry.exit();
  }
}
//调用结束
ContextUtil.exit();
```

> **说明** `ContextUtil.enter(xxx)` 方法仅在初次调用链路入口时才生效，可参考相关 API 文档。

更多设置

限流规则中的 `limitApp`（针对来源）支持以下三种选项，分别对应不同的场景：

1. `default`：适用于不区分消费者的场景。来自任何调用者的请求都将进行限流统计。如果这个资源名的调用总和超过了这条规则定义的阈值，则触发限流。
2. `{some_origin_name}`：适用于对特定的消费者限流的场景。只有来自这个调用者的请求才会进行流量控制。
   例如，资源 A 配置了一条针对消费者 caller1 的规则，那么当且仅当来自消费者 caller1 对资源 A 的请求才会触发流量控制。

3. `other`：表示针对除 `{some_origin_name}` 以外的其余调用方的流量进行流量控制。这个场景适用于资源对大部分消费者都有一个通用的阈值，对特定消费者有不一样的阈值的场景。
   例如，资源 A 可以对大部分消费者可以每秒提供 10 个请求，但是对于消费者 caller1 是个例外，对 caller1，每秒可以提供 200 个请求。需配置两条规则，说明如下：
   - 为消费者 caller1 配置一条 `limitApp` 为 caller1 的限流规则，这条规则每秒的最大请求量设置为 200。
   - 同时，配置一条 `limitApp` 为 other 的规则，这条规则每秒的最大请求量设置为 10，那么任意来自非 caller1 对该 resource 的调用，都不能超过 10。

> **说明** 同一个资源名可以配置多条规则，规则的生效顺序为：{some_origin_name} > other > default。

15.3.5. 弱依赖降级

当弱依赖的第三方应用出错不会影响整体流程，则称之为弱依赖。对于弱依赖不稳定时，需要配置降级规则来保护系统稳定性。

背景信息

在实际业务中，应用通常会调用依赖方（远程服务、数据库、第三方 API 等）来完成服务。例如，支付的时候需要远程调用银联提供的 API。然而依赖方的稳定性是不能保证的。若依赖方出现不稳定的情况，则请求和调用依赖方的方法的响应时间变长，线程产生堆积，最终可能耗尽自身的线程数，导致应用本身不可用。

在复杂链路中，若某一环不稳定，就可能会层层渲染，最终导致整个链路都不可用。

针对以上情况，可以使用 AHAS 应用防护功能对依赖方配置降级规则来保证系统稳定性。

功能原理

若应用依赖于多个下游服务（弱依赖），当下游服务调用过慢，则会严重影响当前服务的调用。为调用端配置基于平均响应时间或错误率的降级规则后，当调用链路中某个下游服务调用的平均响应时间或错误率超过阈值，AHAS 就会对此调用进行降级操作，拒绝多余的调用，保护应用不被调用端短板影响。

配置降级规则具体操作步骤，请参见配置熔断规则。

同时可以配合 Fallback 功能使用，在被降级的时候提供相应的处理逻辑。

15.3.6. 强依赖隔离

若依赖的第三方应用或组件，或者应用自身的内部方法出错会影响而整体流程，则称之为强依赖。对于强依赖，需要配置隔离原则来保护系统稳定性。

功能原理

当强依赖出现不稳定的时候，可以通过配置并发线程数隔离原则来限制不稳定的强依赖并发数，隔离强依赖。配置并发线程数隔离原则后，无需再进行线程池隔离，AHAS会控制资源的线程数。当请求数超过阈值时，AHAS将拒绝多余的请求，直到堆积的线程处理完成，以此来达到信号量隔离的效果。

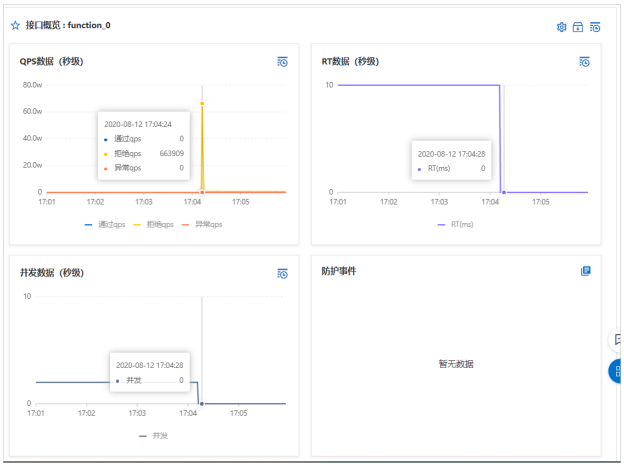
线程数目超出时，设置能够有效地防止自己被慢调用所影响。快速失败

示例

若有应用A的一个接口 `function_0` 不稳定，出现慢SQL。

针对此类场景，可以为调用设置隔离规则（将阈值类型选择为线程数，并将阈值设置为0），具体操作步骤请参见配置隔离规则。

配置规则后，`function_0` 方法被限流，各SQL调用情况如下图所示。



15.3.7. 系统防护

系统防护即从整体维度对应用入口流量进行控制，结合应用的 Load、总体平均 RT、入口 QPS 和线程数等几个维度的监控指标，让系统的入口流量和系统的负载达到一个平衡，让系统尽可能跑在最大吞吐量的同时保证系统整体的稳定性。

背景信息

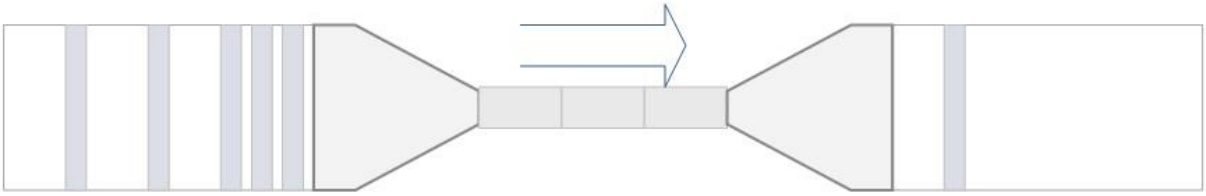
长期以来，系统自适应保护的思路是根据硬指标即系统的负载来做系统过载保护。即当系统负载高于某个阈值，就禁止或者减少流量的进入；若负载恢复，则恢复流量的进入。这样会造成两个不可避免的问题：

- 若根据负载的情况来调节流量的通过率，则会产生延迟。若当前通过率的调整会导致负载增大，那么至少要过 1 秒之后才能被观测到；同理，若当前通过率调整会使负载降低，也需要 1 秒之后才能继续调整。这种方法会浪费系统的处理能力。导致我们看到的负载曲线产生锯齿。
- 通过率恢复慢。在下游应用不可靠，应用响应时间很长，从而导致负载很高的场景中，若下游应用恢复时，应用响应时间也会随之减短，此时通过率理应会大幅度增大。但由于此时负载仍然很高，所以通过率的恢复慢。

为解决上述问题，AHAS 应用流控降级在系统自适应保护的做法是：用每分钟的负载作为启动控制流量，使用请求的响应时间以及当前系统正在处理的请求速率来决定通过的流量。旨在在系统不被拖垮的情况下，提高系统的吞吐率。

功能原理

我们把系统处理请求的过程想象为一个水管，到来的请求是往这个水管灌水，当系统处理顺畅的时候，请求不需要排队，直接从水管中穿过，这个请求的RT是最短的；反之，当请求堆积的时候，那么处理请求的时间则会变为：排队时间 + 最短处理时间。



若用 T 来表示水管内部的水量，用 RT 来表示请求的处理时间，用 P 来表示进来的请求数，那么一个请求从进入水管道到从水管出来，这个水管会存在 `P * RT` 个请求。即当 `T ≈ QPS * Avg(RT)` 的时候，可以认为系统的处理能力和允许进入的请求个数达到了平衡，系统的负载不会继续增加。当入口的流量是水管出来的流量的最大的值的时候，水管的处理能力达到最大利用。

系统规则

系统保护规则是应用整体维度的，而不是资源维度的，并且仅对入口流量生效。入口流量指的是进入应用的流量（EntryType.IN），例如 Web 服务或 Dubbo 服务端接收的请求，都属于入口流量。系统规则支持四种阈值类型：

- Load（仅对 Linux/Unix-like 机器生效）：当系统 load1 超过阈值且系统当前的并发线程数超过系统容量时才会触发系统保护。
- RT：当单台机器上所有入口流量的平均 RT 达到阈值即触发系统保护。
- 线程数：当单台机器上所有入口流量的并发线程数达到阈值即触发系统保护。
- 入口 QPS：当单台机器上所有入口流量的 QPS 达到阈值即触发系统保护。

为应用配置系统规则，具体操作步骤请参见新建系统规则。

15.4. 性能基准

本文列出了 AHAS 应用流控降级在特定 CPU、OS、Java 版本的测试环境下的基准表现。

测试环境

基准的测试环境：

- CPU：Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz （32 cores）
- OS：Linux 2.6.32-220.23.2.ali927.el5.x86_64
- Java 版本：

```
java version "1.8.0_77"
Java(TM) SE Runtime Environment (build 1.8.0_77-b03)
Java HotSpot(TM) 64-Bit Server VM (build 25.77-b03, mixed mode)
```

吞吐量对比

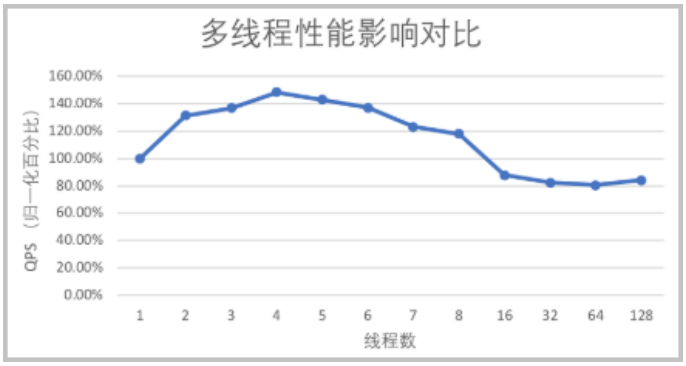
所有吞吐量测试都基于 JMH 编写。

- 单线程吞吐量：
  JVM 参数：`-Xmn256m -Xmx1024m -XX:+UseConcMarkSweepGC`
  通过执行 CPU 密集型操作（如小数组排序）模拟不同 QPS 下的情况，来测试单线程模式下接入 AHAS 应用流控降级与不接入 AHAS 应用流控降级的对比。测试逻辑见 SentinelEntryBenchmark。相关结果如下：

数组长度	Baseline（QPS）	With Sentinel（QPS）	性能损耗
length=25	604589.916	401687.765	33.56%
length=50	221307.617	192407.832	13.06%
length=100	97673.228	91535.146	6.28%
length=200	43742.960	42711.129	2.36%
length=500	15332.924	15171.024	1.06%
length=1000	7012.848	6984.036	0.41%

  - 当单机 QPS 为 20 万以上时，AHAS 应用流控降级带来的性能损耗比较大。这种情况业务场景（如缓存读取操作）本身的耗时非常小，而 AHAS 应用流控降级的统计、检查操作会消耗一定的时间。
  - 单机 QPS 在 5 万以下时，AHAS 应用流控降级性带来的能损耗比较小，适用于大多数场景。

- 多线程吞吐量影响：
  在相同逻辑（对 length=25 的数组进行 shuffle 并排序）的情况下，测试不同线程数下接入 AHAS 应用流控降级的性能表现：



内存占用情况

测试场景：6000 个资源循环执行（即单机的极端场景，目前最多支持 6000 个 Entry）。

- 单线程循环执行：内存占用约 185 MB。
- 8 线程循环执行：内存占用约 1 GB（若系统持续高并发持续很，将导致底层的 LongAdder 内存占用很高）。

15.5. 客户端启动参数

将应用接入AHAS应用防护时，您可以在启动命令中修改客户端启动参数来控制客户端行为。本文分别介绍了AHAS支持的客户端启动参数、SDK启动参数和Java Agent启动参数。

客户端启动参数

名称	含义	类型	默认值	备注
-Dproject.name	指定应用的名称，会显示在AHAS应用防护的应用列表中。	String	Main函数所在的类全名	非必需，建议配置。
-Dcsp.sentinel.app.type	指定应用的类型。	Int	0（APP_TYPE_COMMON）	非必需，1.3.0+版本支持，一般不需要手动配置。
-Dcsp.sentinel.metric.file.single.size	单个监控日志文件的大小。	Long	52428800（50MB）	非必需
-Dcsp.sentinel.metric.file.total.count	监控日志文件的总数上限。	Int	6	非必需
-Dcsp.sentinel.statistic.max.rt	最大的有效响应时长（ms），超出此值则按照此值记录。	Int	4900	非必需
-Dcsp.sentinel.log.dir	Sentinel日志文件目录。	String	${user.home}/logs/csp/	非必需
-Dcsp.sentinel.log.use.pid	日志文件名中是否加入进程号，用于单机部署多个应用的情况。	Boolean	false	非必需

名称	含义	类型	默认值	备注
-Dcsp.sentinel.log.output.type	Record日志输出的类型，`file`代表输出至文件，`console`代表输出至终端。	String	file	非必需，1.3.6+版本支持。
-Dcsp.sentinel.heartbeat.interval.ms	心跳包发送周期，单位ms。	Long	5s	非必需，若不进行配置，则会从相应的`HeartbeatSender`中提取默认值。
-Dcsp.sentinel.web.servlet.block.page	限流页。	String	null	非必需
-Dcsp.sentinel.url.clean.config.path	URL收敛规则配置文件路径。	String	空	非必需
-Dcsp.sentinel.url.suffix.exclude.pattern	收敛的URL后缀。	String	png,gif,mjs,css,htm,html,jpg,jpeg,map,ico,ttf,woff	非必需

说明 列表中的版本信息即ahas-sentinel-client的版本。

SDK启动参数

AHAS支持的Spring Boot starter配置项如下表。

说明 以下配置项需配置到application.properties配置文件中。

配置项	描述	默认值
`spring.cloud.ahas.sentinel.aspectj.enabled`	自动开启注解支持	true
`spring.cloud.ahas.sentinel.mybatis.enabled`	自动开启MyBatis埋点支持	true
`spring.cloud.ahas.sentinel.web.enabled`	自动开启Spring Web埋点支持	true
`spring.cloud.ahas.sentinel.web.http-method-prefix`	Web埋点带上HTTP method前缀（since 1.5.3）	false

Java Agent启动参数

传入参数方式：-D参数。

参数	控制的插件	默认状态
ahas.sentinel.agent.plugin.web.spring.enabled	Spring Web插件	开启
ahas.sentinel.agent.plugin.mybatis.enabled	MyBatis插件	开启
ahas.sentinel.agent.plugin.dubbo.enabled	Dubbo插件	开启
ahas.sentinel.agent.plugin.grpc.enabled	gRPC插件	开启
ahas.sentinel.agent.plugin.gateway.enabled	API gateway插件	开启
ahas.sentinel.agent.plugin.web.servlet.enabled	Web Servlet插件	关闭
ahas.sentinel.agent.plugin.db.enabled	所有JDBC插件	关闭
ahas.sentinel.agent.plugin.cache.enabled	所有cache插件	关闭
ahas.sentinel.agent.plugin.mq.enabled	所有MQ client插件	关闭

示例：

```
-Dahas.sentinel.agent.plugin.web.servlet.enabled=false
```

15.6. 开源Sentinel与商业化AHAS流量防护对比

Sentinel是面向分布式服务架构的轻量级流量控制产品，主要以流量为切入点，从流量控制、熔断降级、系统负载保护等多个维度来保护服务的稳定性。AHAS流量防护是开源框架Sentinel的商业化产品，是阿里巴巴双十一技术体系中的核心组件。它在Sentinel的基础上，支持更多的业务功能。本文对比AHAS流量防护与开源Sentinel，并介绍AHAS流量防护的优势。

应用侧

核心能力

功能描述	AHAS流量防护	开源Sentinel
基础的流控降级、系统保护功能	支持	支持
慢SQL识别与熔断	支持	不支持
异常自动重试	支持	不支持

功能描述	AHAS流量防护	开源Sentinel
弱依赖接口主动降级	支持	不支持
全自动托管、高可用的集群流控服务	支持	不支持
Nginx及Ingress网关流量控制	支持	不支持
全自动托管的Service Mesh集群防护	支持	不支持
Web、RPC流控处理逻辑一键配置	支持	仅支持硬编码
缓存防护（Redis、Memcached）	支持	不支持
专业的自动化熔断（渐进式熔断恢复、默认熔断规则）	支持	不支持
K8s HPA自动弹性	支持	不支持
PHP或者C++应用高可用防护	支持	不支持

快速接入

功能描述	AHAS流量防护	开源Sentinel
Java Agent方式无侵入快速接入	支持近20种主流框架	不支持
K8s Java应用快速接入	支持近20种主流框架	不支持
PHP、Go应用快速接入	支持	不支持
Nginx、Ingress、Envoy网关快速接入	支持	不支持

运维侧

功能描述	AHAS流量防护	开源Sentinel
控制台实时秒级监控	支持接口和应用维度监控，支持按调用类型分类，支持QPS、RT、CPU usage、Load等指标。	有限的支持（内存存储5分钟）
Top N接口统计	支持	不支持
业务异常监控、详情查询	支持	不支持
历史监控查询	支持	不支持
机器水位热力图	支持	不支持
业务场景流量大盘	支持	不支持
规则实时推送	秒级生效，实时可靠。	HTTP方式推送，不可靠。
规则管理	多重持久化，高可用。	内存存储，不可靠。

功能描述	AHAS流量防护	开源Sentinel
针对流控、异常、系统指标飙高等场景的自动化告警	支持	不支持
主子账号权限控制	支持	不支持

AHAS流量防护优势

应用侧

除了支持开源Sentinel的功能，AHAS流量防护还支持以下功能：

- 异常自动重试，避免业务抖动。
- 弱依赖接口主动降级。
- 慢SQL识别与熔断场景（MyBatis，Spring Data JPA，MySQL JDBC，Oracle JDBC）
- 缓存防护场景（Jedis，Memcached）。
- 专业的自动化熔断能力：针对通用场景的渐进式熔断恢复策略、默认熔断规则。
- 全自动托管、高可用的集群流控服务，支持分钟小时级别流控、超大规模流量集群流控。
- 托管的Service Mesh集群流控服务，支持Envoy、Istio集群快速接入。
- PHP应用高可用防护支持。
- K8s容器HPA弹性：支持根据应用实时QPS、RT进行自动扩缩容。
- PHP和C++应用高可用防护支持。

AHAS流量防护还提供多种快速接入的方式：

- Java Agent方式和K8s Java应用零侵入快速接入，支持近20种主流框架和API Gateway。
- K8s Java应用零侵入快速接入，支持近20种主流框架和API Gateway。
- Nginx、Ingress、Envoy网关快速接入。
- PHP、Go、C++应用快速接入。

运维侧

AHAS流量防护拥有企业级的控制台，包括以下功能：

- 可靠的实时监控和历史秒级监控数据查询，包含接口维度的QPS、响应时间、并发量、异常及系统指标监控、JVM监控等指标，支持按照调用类型分层展示，支持同比和环比展示。
- Top K接口监控统计，快速了解系统的慢调用和大流量接口。
- 热力图概览，快速定位不稳定的机器。
- 业务场景流量大盘，便于关注多个系统的整体流量情况。
- 多种规则管理和实时推送，秒级生效，实时可靠。
- 告警中心（触发流控、异常、慢调用、CPU使用率高等事件），支持通过钉钉机器人等形式自动触发报警。
- Web、RPC流控处理逻辑支持在控制台一键配置，支持自定义返回值、返回异常、跳转等行为。
- 细粒度的权限控制支持。

相关文档

- Sentinel介绍