ALIBABA CLOUD

Alibaba Cloud

应用实时监控服务 ARMS

应用监控

文档版本：20220713

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.应用监控概述

ARMS应用监控是一款应用性能管理（Application Performance Management，简称APM）产品。您无需修改代码，只需为应用安装一个探针，ARMS就能够对应用进行全方位监控，帮助您快速定位出错接口和慢接口、重现调用参数、发现系统瓶颈，从而大幅提升线上问题诊断的效率。

自动发现应用拓扑

ARMS应用监控探针能够自动发现应用的上下游依赖关系。具体而言，该探针能够有效捕获、智能计算、自动展示不同应用之间通过RPC框架（例如Dubbo、HTTP、HSF等协议）组成的调用链。您可以通过应用拓扑轻松发现系统中的性能瓶颈和异常调用。

3D拓扑

3D拓扑图能立体展示应用、服务和主机的健康状况，以及应用的上下游依赖关系，帮助您快速定位诱发故障的服务、被故障影响的应用和关联的主机等，全方位地诊断故障根源，从而快速排除故障。

捕获异常事务和慢事务

您可以进一步获取接口的慢SQL、MQ堆积分析报表或者异常分类报表，对错、慢等常见问题进行更细致的分析。

自动发现并监控接口

ARMS应用监控能够自动发现和监控应用代码中常见的Web框架和RPC框架，并自动统计Web接口和RPC接口的调用量、响应时间、错误数等指标。

实时诊断

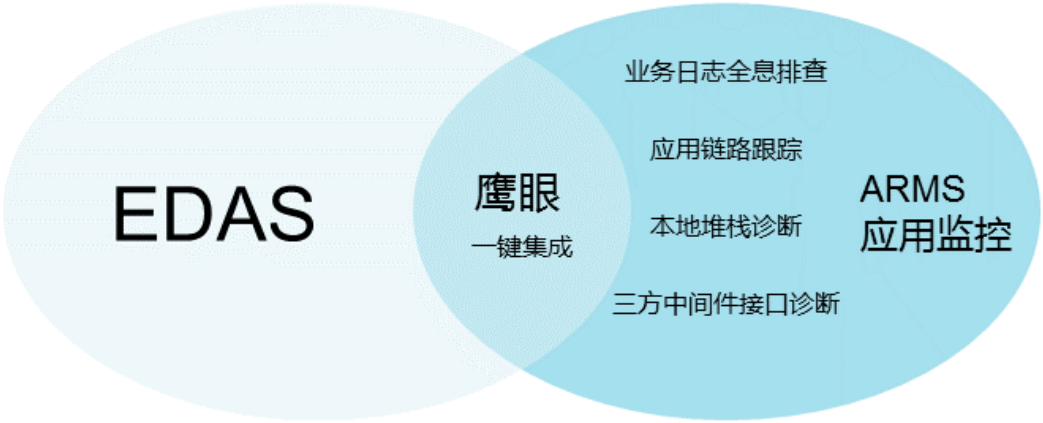
当您需要密切监控一小段时间内的应用性能时，例如发布应用或者对应用进行压测时，可以使用ARMS应用监控的实时诊断功能。开启实时诊断后，ARMS应用监控会持续监控应用5分钟，并在这5分钟内全量上报调用链数据。接下来，您就能以出现性能问题的调用链路为起点，通过方法栈瀑布图和线程剖析等功能定位问题原因。

多维排查

您可以查看分布式及本地方法栈明细，并按应用、IP、耗时等维度进行多维分析。

集成阿里云中间件PaaS平台

ARMS应用监控支持一键集成阿里云PaaS平台EDAS，让运行于阿里中间件分布式架构平台上的应用监控更加有效。



2.接入应用监控

2.1. 应用监控接入概述

应用安装探针的准备工作方便您使用ARMS控制台查看丰富的应用监控指标。本文按应用部署环境和应用语言的维度列出了所有安装探针的文档。

按部署环境开始监控

按应用语言开始监控

> **说明** 上文的星号（*）表示需开通链路追踪Tracing Analysis服务。

2.2. 开始监控Java应用

2.2.1. 为Java应用手动安装Agent

为Java应用安装Agent后，ARMS即可开始监控Java应用，您可以查看应用拓扑、调用链路、异常事务、慢事务和SQL分析等一系列监控数据。您可以选择以手动方式或脚本方式安装Agent，本文介绍如何为Java应用手动安装Agent。

前提条件

- 确保您使用的云服务器ECS实例的安全组已开放8442、8443、8883三个端口的TCP出方向权限。为云服务器ECS开放出方向权限的方法，请参见添加安全组规则。

  > **说明** ARMS不仅可接入阿里云ECS上的应用，还能接入其他能访问公网的服务器上的应用。

- 确保您使用的第三方组件或框架在应用监控兼容性列表范围内，请参见应用监控兼容性列表。
- 检查您的JDK版本。ARMS应用监控支持的JDK版本如下：
  - JDK 1.7.0+
  - JDK 1.8.0_25+

    > **说明**
    > - Kubernetes集群应用部署建议：JDK 1.8.0_191+。
    > - 如果JDK版本为1.8.0_25或者1.8.0_31，可能会出现无法安装探针的情况，请升级至1.8.X最新版本。

  - JDK 11.0.8+

    > **说明** JDK 1.8及以下版本和JDK 11版本对应的探针安装包不同，请根据不同的JDK版本下载对应的探针安装包或调整应用监控组件ack-onepilot的配置。

操作步骤

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域，然后单击**接入应用**。

3. 在**接入应用**面板单击**Java**，然后选择**手动安装**。
4. 下载Agent。
   - 方法一：手动下载。在**接入Java**面板的**STEP1**区域根据JDK版本下载对应的Agent。
   - 方法二：使用**Wget**命令下载。根据您的地域下载对应的Agent安装包。

> **说明** 请使用公网地址，如无法下载则使用VPC地址。

查看各地域对应的Agent（JDK 8及以下版本）安装包下载命令

地域	公网地址	VPC地址
华东1（杭州）	`wget "http://arms-apm-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-hangzhou.oss-cn-hangzhou-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华东2（上海）	`wget "http://arms-apm-cn-shanghai.oss-cn-shanghai.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-shanghai.oss-cn-shanghai-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北1（青岛）	`wget "http://arms-apm-cn-qingdao.oss-cn-qingdao.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-qingdao.oss-cn-qingdao-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北2（北京）	`wget "http://arms-apm-cn-beijing.oss-cn-beijing.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-beijing.oss-cn-beijing-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北3（张家口）	`wget "http://arms-apm-cn-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-zhangjiakou.oss-cn-zhangjiakou-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
华北5（呼和浩特）	`wget "http://arms-apm-cn-huhehaote.oss-cn-huhehaote.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-huhehaote.oss-cn-huhehaote-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北6（乌兰察布）	`wget "http://arms-apm-cn-wulanchabu.oss-cn-wulanchabu.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-wulanchabu.oss-cn-wulanchabu-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华南1（深圳）	`wget "http://arms-apm-cn-shenzhen.oss-cn-shenzhen.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-shenzhen.oss-cn-shenzhen-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华南2（河源）	`wget "http://arms-apm-cn-heyuan.oss-cn-heyuan.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-heyuan.oss-cn-heyuan-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华南3（广州）	`wget "http://arms-apm-cn-guangzhou.oss-cn-guangzhou.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-guangzhou.oss-cn-guangzhou-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
西南1（成都）	`wget "http://arms-apm-cn-chengdu.oss-cn-chengdu.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-chengdu.oss-cn-chengdu-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
中国（香港）	`wget "http://arms-apm-cn-hongkong.oss-cn-hongkong.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-hongkong.oss-cn-hongkong-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
亚太东南1（新加坡）	`wget "http://arms-apm-ap-southeast-1.oss-ap-southeast-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-1.oss-ap-southeast-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东南2（悉尼）	`wget "http://arms-apm-ap-southeast-2.oss-ap-southeast-2.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-2.oss-ap-southeast-2-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东南3（吉隆坡）	`wget "http://arms-apm-ap-southeast-3.oss-ap-southeast-3.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-3.oss-ap-southeast-3-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东南5（雅加达）	`wget "http://arms-apm-ap-southeast-5.oss-ap-southeast-5.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-5.oss-ap-southeast-5-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东北1（东京）	`wget "http://arms-apm-ap-northeast-1.oss-ap-northeast-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-northeast-1.oss-ap-northeast-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
欧洲中部1（法兰克福）	`wget "http://arms-apm-eu-central-1.oss-eu-central-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-eu-central-1.oss-eu-central-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
欧洲西部1（伦敦）	`wget "http://arms-apm-eu-west-1.oss-eu-west-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-eu-west-1.oss-eu-west-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
美国东部1（弗吉尼亚）	`wget "http://arms-apm-us-east-1.oss-us-east-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-us-east-1.oss-us-east-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
美国西部1（硅谷）	`wget "http://arms-apm-us-west-1.oss-us-west-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zipwget "http://arms-apm-us-west-1.oss-us-west-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-us-west-1.oss-us-west-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太南部1（孟买）	`wget "http://arms-apm-ap-south-1.oss-ap-south-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-south-1.oss-ap-south-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华东1金融云	无	`wget "http://arms-apm-cn-hangzhou-finance.oss-cn-hzjbp-b-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华东2金融云	无	`wget "http://arms-apm-cn-shanghai-finance-1.oss-cn-shanghai-finance-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
华南1金融云	无	`wget "http://arms-apm-cn-shenzhen-finance-1.oss-cn-shenzhen-finance-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
政务云	`wget "http://arms-apm-cn-north-2-gov-1.oss-cn-north-2-gov-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-north-2-gov-1.oss-cn-north-2-gov-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

5. 安装Agent。

i. 进入Agent安装包所在目录，并执行以下命令来解压安装包到任意工作目录下。

```
unzip ArmsAgent.zip -d /{user.workspace}/
```

> **说明** *{user.workspace}*是示例目录，请替换为真实的目录。

ii. 添加AppName和LicenseKey参数。

> **说明** 将示例代码中的`{LicenseKey}`替换成STEP2区域获取License Key，`{AppName}`替换为您实际的应用名称（应用名不可包含中文字符），`{user.workspace}`替换成实际Agent安装包的解压目录，将demoApp.jar替换为真实的JAR包地址。

- 方法一：根据您的应用运行环境修改JVM参数。

运行环境	步骤

运行环境	步骤
Tomcat（Linux或macOS操作系统）	在*{TOMCAT_HOME}/bin/setenv.sh*文件中添加以下配置。 `JAVA_OPTS="$JAVA_OPTS -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey={LicenseKey} -Darms.appName={AppName}"` 如果您的Tomcat版本没有*setenv.sh*配置文件，请打开*{TOMCAT_HOME}/bin/catalina.sh*文件，并在JAVA_OPTS后添加上述配置，具体示例，请参见catalina.sh的第256行。

Tomcat（Windows操作系统）	在*{TOMCAT_HOME}/bin/catalina.bat*文件中添加以下配置。
	`set "JAVA_OPTS=%JAVA_OPTS% -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey={LicenseKey} -Darms.appName={AppName}"`
	如果以上配置未生效，可以尝试在*{TOMCAT_HOME}/bin/catalina.bat*文件中添加以下配置。
	`set "CATALINA_OPTS=-javaagent:/{user.workspace}/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey={LicenseKey} -Darms.appName={AppName}"`

Jetty	在*{JETTY_HOME}/start.in*配置文件中添加以下配置。 `--exec -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey={LicenseKey} -Darms.appName={AppName}`
Spring Boot	启动Spring Boot进程时，在启动命令后加上-javaagent参数。 `java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey={LicenseKey} -Darms.appName={AppName} -jar demoApp.jar`

运行环境	步骤

Resin	启动Resin进程时，在*conf/resin.xml*配置文件中添加以下标签。
	`<server-default>`
	`<jvm-arg>-javaagent:{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar</jvm-arg>`
	`<jvm-arg>-Darms.licenseKey={LicenseKey}</jvm-arg>`
	`<jvm-arg>-Darms.appName={AppName}</jvm-arg>`
	`</server-default>`
	在*conf/app-default.xml*文件中添加以下标签。
	`<library-loader path="{user.workspace}/ArmsAgent/plugin"/>`

Windows	使用CMD启动Java进程时，在挂载Agent路径中使用反斜线（\）作为分隔符。 `java -javaagent:\{user.workspace}\ArmsAgent\arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey={LicenseKey} -Darms.appName={AppName} -jar {user.workspace}\demoApp.jar`

如需在一台服务器上部署同一应用的多个实例，可以通过-Darms.agentId参数（逻辑编号）来区分接入的JVM进程，例如：

```
java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey={LicenseKey} -Darms.appName={AppName} -Darms.agentId=001 -jar demoApp.jar
```

- 方法二：在*arms-agent.config*文件中添加以下配置。

```
arms.licenseKey={LicenseKey} arms.appName={AppName}
```

6. 在Java应用的启动脚本中添加以下参数。

```
-javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar
```

7. 重启Java应用。

结果验证

约一分钟后，若Java应用出现**在应用列表**页面中且有数据上报，则说明接入成功。

卸载Agent

当您不需要使用ARMS监控您的Java应用时，请按照以下步骤卸载Agent。

1. 删除步骤7中添加的{AppName}、{LicenseKey}等所有参数。
2. 重启Java应用。

快速更改应用名称

如需更改应用名称，例如忘记将示例应用名称Java-Demo修改为自定义名称，您可以在不重启应用、不重装Agent的情况下更改应用名称，具体操作，请参见以通用方式安装Agent的普通Java应用如何更改应用名称？。

相关文档

- 应用监控常见问题

2.2.2. 使用脚本为Java应用快速安装探针

ARMS提供一键接入方式为Java应用安装探针，安装成功后无需重启应用即可开始监控，适用于新手用户。当应用重启时，探针会自动加载，该Java应用将自动接入ARMS应用监控。

前提条件

- 确保您使用的云服务器ECS实例的安全组已开放8442、8443、8883三个端口的TCP出方向权限。为云服务器ECS开放出方向权限的方法，请参见添加安全组规则。

  > **说明** ARMS不仅可接入阿里云ECS上的应用，还能接入其他能访问公网的服务器上的应用。

- 确保您使用的第三方组件或框架在应用监控兼容性列表范围内，请参见应用监控兼容性列表。
- 检查您的JDK版本。ARMS应用监控支持的JDK版本如下：
  - JDK 1.7.0+
  - JDK 1.8.0_25+

    > **说明**
    > - Kubernetes集群应用部署建议：JDK 1.8.0_191+。
    > - 如果JDK版本为1.8.0_25或者1.8.0_31，可能会出现无法安装探针的情况，请升级至1.8.X最新版本。

  - JDK 11.0.8+

    > **说明** JDK 1.8及以下版本和JDK 11版本对应的探针安装包不同，请根据不同的JDK版本下载对应的探针安装包或调整应用监控组件ack-onepilot的配置。

- 若您的应用已经按照手动接入方式接入ARMS应用监控，则需先卸载探针才能正常使用一键接入方式，请参见卸载探针。

操作步骤

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域，然后单击**接入应用**。
3. 在**接入应用**面板单击**Java**，然后选择**脚本安装**。
4. 执行**STEP1**区域的安装脚本。


```
wget -O- http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/install.sh | sh && ~/.arms/supervisor/cli.sh ****** Java-Demo
```

> **说明**
>
> - 将 `Java-Demo` 替换成您的应用名，应用名暂不支持中文。
> - 执行安装脚本后，该脚本会自动下载最新探针。
> - 若您的服务器只有一个Java进程，安装脚本会默认选择该进程安装探针；若您的服务器有多个Java进程，请根据提示选择一个进程安装探针。

结果验证

约一分钟后，若您的应用出现在应用列表中且有数据上报，则说明接入成功。

卸载探针

1. 当您不需要使用ARMS监控您的Java应用时，执行 `jps -l` 命令查看所有进程，并在执行结果中找到 `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` 对应的进程号。

   在本示例中，`com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` 对应的进程号为：62857。

   ```
   ➜ ~ jps -l
   62800 org.apache.catalina.startup.Bootstrap
   62857 com.alibaba.mw.arms.apm.supervisor.daemon.Daemon
   5411
   62799 org.jetbrains.jps.cmdline.Launcher
   67809 sun.tools.jps.Jps
   ```

2. 执行命令 `kill -9 <进程号>` 。

   例如：`kill -9 62857` 。

3. 执行 `rm -rf /.arms /root/.arms` 。

4. 重启您的应用。

快速更改应用名称

如果因为某些原因希望更改应用名称，例如忘记将示例应用名称Java-Demo修改为自定义名称，您可以在不重启应用、不重装探针的情况下更改应用名称，详情参见以快速方式安装探针的普通Java应用如何更改应用名称。

常见问题

1. 如果在执行一键接入Java应用脚本时出现以下getcwd相关错误该怎么处理？

1. 如果在执行一键接入Java应用脚本时出现以下getcwd相关错误该怎么处理？

```
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory Error occurred during initialization of VM java.lang.Error: Properties init: Could not determine current working directory. at java.lang.System.initProperties(Native Method) at java.lang.System.initializeSystemClass(System.java:1119)
```

   可能原因是执行脚本过程中误删了当前目录。解决办法为：先执行 `cd` ，然后重新运行脚本。

2. 使用一键接入方式安装探针后，在哪里查看日志？
   日志的默认目录为：*/root/.arms/supervisor/logs/arms-supervisor.log*，若此目录下没有日志，请执行命令 `ps -ef |grep arms` 查看日志所在目录。

相关文档

- 应用监控常见问题

2.2.3. 为函数计算中的Java应用安装探针

只需安装ARMS应用监控组件（探针），即可对部署在函数计算中的Java应用进行监控，查看应用拓扑、接口调用、异常事务和慢事务等方面的监控数据。本文介绍如何为函数计算中的Java应用安装探针。

前提条件

使用控制台创建函数

步骤一：获取License Key

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域，然后单击**接入应用**。
3. 在**接入中心**面板单击**Java**，然后在**STEP2**区域获取License Key。

步骤二：配置函数变量

1. 登录函数计算控制台。
2. 在顶部菜单栏，选择地域。
3. 在左侧导航栏中，单击**服务及函数**。
4. 在**服务列表**区域单击目标服务。
5. 在**函数管理**页面找到目标函数，单击**操作**列的**配置**。
6. 在**编辑函数配置**页面的环境变量区域选择使用表单编辑，然后单击**+添加变量**。

7. 设置变量为*FC_EXTENSIONS_ARMS_LICENSE_KEY*，值为步骤一：获取License Key中获取的License Key，然后单击**保存**。

结果验证

在函数计算控制台多次调用该函数后，若Java应用出现在ARMS控制台的**应用列表**页面中且有数据上报，则说明接入成功。

相关文档

- 应用监控常见问题

2.2.4. 为容器服务Kubernetes版Java应用安装探针

您只需安装ARMS应用监控组件（探针）ack-onepilot，即可监控部署在容器服务Kubernetes版中的Java应用，并查看应用拓扑、接口调用、异常事务和慢事务等相关监控数据。本文介绍如何为容器服务Kubernetes版Java应用安装探针。

前提条件

- 创建Kubernetes集群。您可按需选择创建Kubernetes专有版集群、创建Kubernetes托管版集群或创建Serverless Kubernetes集群。
- 创建命名空间，具体操作，请参见管理命名空间。本文示例中的命名空间名称为*arms-demo*。
- 检查您的JDK版本。ARMS应用监控支持的JDK版本如下：
  - JDK 1.7.0+
  - JDK 1.8.0_25+

    > **说明**
    > - Kubernetes集群应用部署建议：JDK 1.8.0_191+。
    > - 如果JDK版本为1.8.0_25或者1.8.0_31，可能会出现无法安装探针的情况，请升级至1.8.X最新版本。

  - JDK 11.0.8+

    > **说明** JDK 1.8及以下版本和JDK 11版本对应的探针安装包不同，请根据不同的JDK版本下载对应的探针安装包或调整应用监控组件ack-onepilot的配置。

步骤一：安装ARMS应用监控组件

1. 登录容器服务管理控制台。
2. 在左侧导航栏选择**市场 > 应用市场**，在右侧**应用目录**页签通过关键字搜索**ack-onepilot**，然后单击目标卡片。
3. 在**ack-onepilot**页面上，单击右上角的**一键部署**。
4. 在**创建**面板中选择安装组件的集群、命名空间，并输入组件发布名称，然后单击**下一步**。
5. 单击**确定**。

步骤二：授予ARMS资源的访问权限

- 如果需监控ASK（Serverless Kubernetes）或对接了ECI的集群应用，请在云资源访问授权页面完成授权，然后重启ack-onepilot组件下的所有Pod。
- 如果需监控ACK集群应用，但ACK集群中不存在ARMS Addon Token，请执行以下操作手动为集群授予

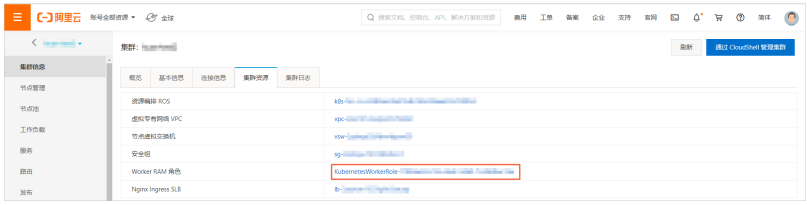
ARMS资源的访问权限。

> **说明**
> 查看是否存在ARMS Addon Token的操作，请参见查看集群是否存在ARMS Addon Token。
> 集群存在ARMS Addon Token时，ARMS会进行免密授权。Kubernetes托管版集群默认存在ARMS Addon Token，但对于部分早期创建的Kubernetes托管版集群，可能会存在没有ARMS Addon Token的情况，因此，对于Kubernetes托管版集群，建议首先检查ARMS Addon Token是否存在。若不存在，需进行手动授权。

i. 登录容器服务管理控制台。

ii. 在左侧导航栏选择**集群**，在**集群列表**页面，单击目标集群名称或其右侧**操作**列的**详情**。

iii. 在目标集群的**集群信息**页面上单击**集群资源**页签，然后单击**Worker RAM角色**右侧的链接。

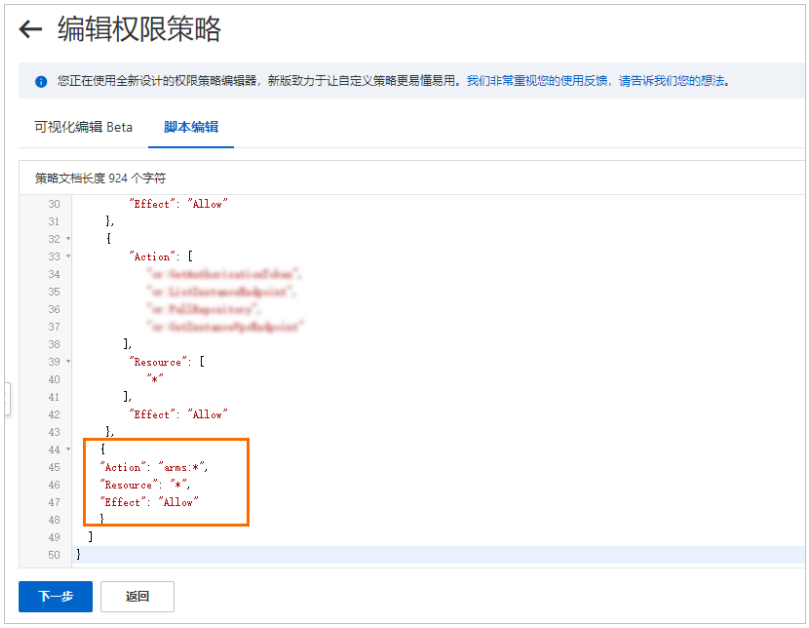


iv. 在访问控制RAM控制台的RAM角色管理页面上，单击**权限管理**页签上的权限策略名称。

v. 在**策略内容**页签上单击**修改策略内容**。

vi. 在**脚本编辑**页签中添加以下内容，然后单击**下一步**。

```
{
    "Action": "arms:*",
    "Resource": "*",
    "Effect": "Allow"
}
```



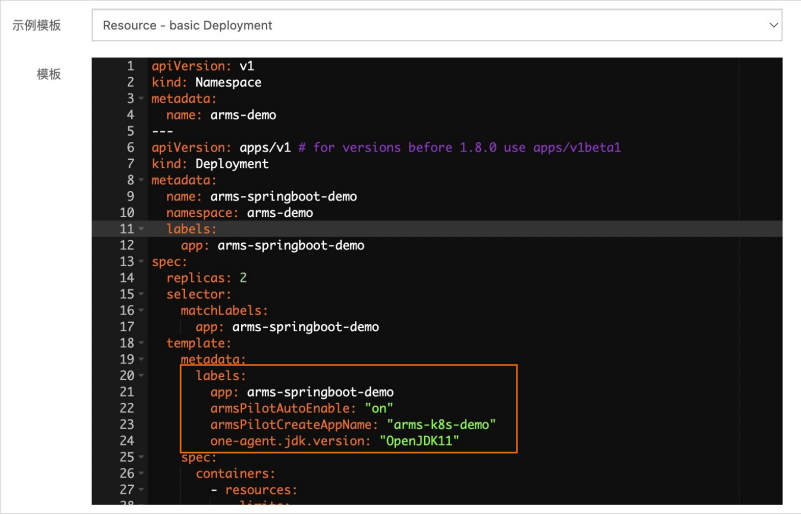
vii. 确认策略内容无误后，单击**确定**。

步骤三：为Java应用开启ARMS应用监控

如需在创建新应用的同时开启ARMS应用监控，请完成以下操作。

1. 在容器服务Kubernetes版控制台左侧导航栏单击**集群**，在**集群列表**页面上的目标集群右侧**操作**列单击**应用管理**。
2. 在**无状态**页面右上角单击**使用YAML创建资源**。
3. 选择**示例模板**，并在**模板**（YAML格式）中将以下 `labels` 添加到*spec.template.metadata*层级下。

```
labels:
  armsPilotAutoEnable: "on"
  armsPilotCreateAppName: "<your-deployment-name>"     #请将<your-deployment-name>替换为您的应用名称。
  one-agent.jdk.version: "OpenJDK11"     #如果应用的JDK版本是JDK 11，则需要配置此参数。
```



创建一个无状态（Deployment）应用并开启ARMS应用监控的完整YAML示例模板如下：

执行结果

在**无状态**页面上，目标应用的**操作**列将出现**ARMS控制台**按钮。

卸载探针

1. 登录容器服务管理控制台。
2. 在左侧导航栏单击**集群**，在**集群列表**页面上的目标集群右侧**操作**列单击**应用管理**。
3. 在左侧导航栏选择**应用 > Helm**。
4. 在**Helm**页面，ARMS Agent对应的发布名称**ack-onepilot**右侧**操作**列，单击**删除**。
5. 在**删除应用**对话框单击**确定**。
6. 重启您的业务Pod。

常见问题

如何更改应用名称

如需更改应用名称，您需要修改Deployment内的armsPilotCreateAppName参数。具体操作，请参见部署在容器服务K8s集群中的Java应用如何更改应用名称。

为什么Agent安装失败？

可能原因：目标集群中不存在ARMS Addon Token。

解决方案：查看集群是否存在ARMS Addon Token并为集群授予ARMS资源的访问权限。具体操作，请参见容器服务K8s集群中的应用安装Agent失败怎么处理？。

如何跨区域上报数据？

在容器集群arms-pilot Namespace下的Deployment中添加ARMS_REPORT_REGION环境变量，值为ARMS所支持的RegionId（例如cn-hangzhou、cn-beijing），然后重启现有应用。更多信息，请参见ACK集群如何跨区域上报数据？。

相关文档

- 创建Kubernetes专有版集群
- 管理命名空间
- 应用监控常见问题

2.2.5. 为开源Kubernetes环境中的应用安装探针

借助ARMS应用监控，您可以对开源Kubernetes环境的应用进行应用拓扑、接口调用、异常事务和慢事务监控、SQL分析等监控。本文将帮助您将开源Kubernetes环境中的应用接入ARMS应用监控。

前提条件

- 确保您的Kubernetes api-server组件接口版本在1.10及以上。
- 确保您的集群连通公网。
- 检查您的JDK版本。ARMS应用监控支持的JDK版本如下：
  - JDK 1.7.0+
  - JDK 1.8.0_25+

    > **说明**
    > - Kubernetes集群应用部署建议：JDK 1.8.0_191+。
    > - 如果JDK版本为1.8.0_25或者1.8.0_31，可能会出现无法安装探针的情况，请升级至1.8.X最新版本。

  - JDK 11.0.8+

    > **说明** JDK 1.8及以下版本和JDK 11版本对应的探针安装包不同，请根据不同的JDK版本下载对应的探针安装包或调整应用监控组件ack-onepilot的配置。

方式一：通过ACK注册集群接入ARMS

1. 将开源Kubernetes接入ACK One。具体操作，请参见创建注册集群并接入本地数据中心集群。
2. 为注册集群安装ack-arms-pilot组件。具体操作，请参见将应用实时监控服务ARMS接入注册集群。

方式二：将开源Kubernetes直接接入ARMS

步骤1：安装Helm3

步骤2：安装探针

ARMS应用监控目前仅支持无状态（Deployment）和有状态（StatefulSet）两种类型的应用接入，两种类型的应用接入方法相同。此处以将开源Kubernetes环境中的无状态（Deployment）类型的应用接入ARMS应用监控为例。

1. 执行以下 `wget` 命令下载ack-onepilot安装包。

```
wget 'https://arms-apm-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/ack-onepilot/ack-onepilot-2.0.6.tgz'
```

2. 执行以下命令解压ack-onepilot安装包。

```
tar xvf ack-onepilot-2.0.6.tgz
```

3. 编辑安装包下的*values.yaml*文件，根据实际情况修改以下参数，然后保存。

```
image: registry-vpc.__ACK_REGION_ID__.aliyuncs.com/ack-onepilot/ack-onepilot
cluster_id: __ACK_CLUSTER_ID__
accessKey: __ACCESSKEY__
accessKeySecret: __ACCESSKEY_SECRET__
uid: "__ACK_UID__"
region_id: __ACK_REGION_ID__
```

   - `image`：ack-onepilot镜像地址。

     说明 以上示例为VPC环境镜像地址，公网环境镜像获取地址如下：

     ```
     registry.__ACK_REGION_ID__.aliyuncs.com/ack-onepilot/ack-onepilot
     ```

   - `__ACK_REGION_ID__`：阿里云地域ID，应用监控支持的地域，请参见应用监控目前支持的地域。
   - `__ACK_CLUSTER_ID__`：自定义Kubernetes集群ID，集群的唯一标识。建议格式为 `<uid>-<clusterid>`。
   - `__ACCESSKEY__` 和 `__ACCESSKEY_SECRET__`：阿里云账号的AccessKey ID和AccessKey Secret。获取方法，请参见获取AccessKey。
   - `__ACK_UID__`：阿里云账号ID。将鼠标悬浮于阿里云控制台右上角的头像上可以获取。

4. 执行以下命令安装ack-onepilot。

```
helm3 upgrade --install ack-onepilot ack-onepilot --namespace ack-onepilot --create-namespace
```

步骤3：获取License Key

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域，然后单击**接入应用**。
3. 在**接入中心**面板单击**Java**，然后在**STEP2**区域获取License Key。

步骤4：修改应用的YAML文件

1. 执行以下命令查看目标无状态（Deployment）应用的YAML文件。

   ```
   kubectl get deployment {deployment名称} -o yaml
   ```

   > **说明** 若您不清楚 `{deployment名称}` ，请先执行以下命令查看所有无状态（Deployment）应用，在执行结果中找到目标无状态（Deployment）应用，再查看目标无状态（Deployment）应用的YAML文件。
   >
   > ```
   > kubectl get deployments --all-namespace
   > ```

2. 启动编辑目标无状态（Deployment）应用的YAML文件。

   ```
   kubectl edit deployment {Deployment名称} -o yaml
   ```

3. 在YAML文件中的*spec.template.metadata*层级下添加以下内容。

   ```
   ARMSApmAppName: xxx
   ARMSApmLicenseKey: xxx
   one-agent.jdk.version: "OpenJDK11"    //如果应用的JDK版本是JDK11，则需要配置此参数。
   ```

   > **注意**
   >
   > - 请将 `xxx` 分别替换成您的License Key和应用名称，应用名暂不支持中文。
   > - 将License Key中的符号 `@` 替换为符号 `_` 。

   如果您需要在开源K8s环境中创建一个新的无状态（Deployment）应用并接入ARMS，则应用的完整YAML文件如下：

4. 保存配置后，应用将自动重启，以上配置生效。
   2~5分钟后，若您的应用出现在ARMS控制台的**应用监控 > 应用列表**页面中且有数据上报，则说明接入成功。

卸载探针

当您不需要再监控开源Kubernetes环境中的Java应用时，可以卸载ack-onepilot。

1. 执行以下命令卸载ack-onepilot。

```
helm3 uninstall --namespace ack-onepilot ack-onepilot
```

2. 重启您的应用Pod。

相关文档

- 应用监控常见问题

2.2.6. 为Docker中的Java应用安装Agent

为Docker中的Java应用安装ARMS Agent后，ARMS即可开始监控Java应用，且ARMS将自动适配该应用运行的环境，不需要针对Tomcat、Jetty或Spring Boot等应用单独配置运行环境。本文介绍如何为Docker中的Java应用安装Agent。

前提条件

- 您已在Docker中部署Java应用。
- 检查您的JDK版本。ARMS应用监控支持的JDK版本如下：
  - JDK 1.7.0+
  - JDK 1.8.0_25+

    > **说明**
    > - Kubernetes集群应用部署建议：JDK 1.8.0_191+。
    > - 如果JDK版本为1.8.0_25或者1.8.0_31，可能会出现无法安装探针的情况，请升级至1.8.X最新版本。

  - JDK 11.0.8+

    > **说明** JDK 1.8及以下版本和JDK 11版本对应的探针安装包不同，请根据不同的JDK版本下载对应的探针安装包或调整应用监控组件ack-onepilot的配置。

背景信息

对于Java应用镜像*{original-docker-image:tag}*，可以通过编辑*Dockerfile*文件来集成已有镜像，然后构建和启动新的镜像，即可将Java应用接入ARMS应用监控。

步骤一：获取License Key

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域，然后单击**接入应用**。
3. 在**接入中心**面板单击**Java**，然后在**STEP2**区域获取License Key。

步骤二：集成已有镜像

参考以下*Dockerfile*示例修改您的*Dockerfile*文件。

```
###################################
##                               ###
##       ARMS APM DEMO Docker    ###
##           For Java            ###
##       withAgent   V0.1        ###
##                               ###
###################################
# 将{original-docker-image:tag}替换为您的镜像地址。
FROM {original-docker-image:tag}
WORKDIR /root/
# 根据所在地域替换Agent的下载地址。
RUN wget "http://arms-apm-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/ArmsAgent.zip" -O ArmsAg
ent.zip
RUN unzip ArmsAgent.zip -d /root/
# 按照步骤一的说明获取License Key。
# {AppName}为自定义ARMS监控应用名称，不可包含中文字符。
# 若所有镜像都接入同一个应用监控任务，配置此处的arms_licenseKey和arms_appName即可。
ENV arms_licenseKey={LicenseKey}
ENV arms_appName={AppName}
ENV JAVA_TOOL_OPTIONS ${JAVA_TOOL_OPTIONS} '-javaagent:/root/ArmsAgent/arms-bootstrap-1.7.0
-SNAPSHOT.jar -Darms.licenseKey='${arms_licenseKey}' -Darms.appName='${arms_appName}
### for check the args
RUN env | grep JAVA_TOOL_OPTIONS
### 可在下方添加自定义Dockerfile逻辑。
### ......
```

请按照以下说明替换上述配置文件中的示例值。

- 将`{original-docker-image:tag}`替换为您的镜像地址。若您没有自定义镜像，可使用系统镜像。
- 根据所在地域替换Agent的下载地址。

> **说明** 请使用公网地址，如无法下载则使用VPC地址。

查看各地域对应的Agent（JDK 8及以下版本）安装包下载命令

地域	公网地址	VPC地址
华东1（杭州）	`wget "http://arms-apm-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-hangzhou.oss-cn-hangzhou-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华东2（上海）	`wget "http://arms-apm-cn-shanghai.oss-cn-shanghai.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-shanghai.oss-cn-shanghai-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北1（青岛）	`wget "http://arms-apm-cn-qingdao.oss-cn-qingdao.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-qingdao.oss-cn-qingdao-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北2（北京）	`wget "http://arms-apm-cn-beijing.oss-cn-beijing.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-beijing.oss-cn-beijing-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北3（张家口）	`wget "http://arms-apm-cn-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-zhangjiakou.oss-cn-zhangjiakou-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北5（呼和浩特）	`wget "http://arms-apm-cn-huhehaote.oss-cn-huhehaote.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-huhehaote.oss-cn-huhehaote-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华北6（乌兰察布）	`wget "http://arms-apm-cn-wulanchabu.oss-cn-wulanchabu.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-wulanchabu.oss-cn-wulanchabu-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
华南1（深圳）	`wget "http://arms-apm-cn-shenzhen.oss-cn-shenzhen.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-shenzhen.oss-cn-shenzhen-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华南2（河源）	`wget "http://arms-apm-cn-heyuan.oss-cn-heyuan.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-heyuan.oss-cn-heyuan-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华南3（广州）	`wget "http://arms-apm-cn-guangzhou.oss-cn-guangzhou.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-guangzhou.oss-cn-guangzhou-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
西南1（成都）	`wget "http://arms-apm-cn-chengdu.oss-cn-chengdu.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-chengdu.oss-cn-chengdu-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
中国（香港）	`wget "http://arms-apm-cn-hongkong.oss-cn-hongkong.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-hongkong.oss-cn-hongkong-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东南1（新加坡）	`wget "http://arms-apm-ap-southeast-1.oss-ap-southeast-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-1.oss-ap-southeast-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
亚太东南2（悉尼）	`wget "http://arms-apm-ap-southeast-2.oss-ap-southeast-2.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-2.oss-ap-southeast-2-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东南3（吉隆坡）	`wget "http://arms-apm-ap-southeast-3.oss-ap-southeast-3.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-3.oss-ap-southeast-3-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东南5（雅加达）	`wget "http://arms-apm-ap-southeast-5.oss-ap-southeast-5.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-southeast-5.oss-ap-southeast-5-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太东北1（东京）	`wget "http://arms-apm-ap-northeast-1.oss-ap-northeast-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-northeast-1.oss-ap-northeast-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
欧洲中部1（法兰克福）	`wget "http://arms-apm-eu-central-1.oss-eu-central-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-eu-central-1.oss-eu-central-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
欧洲西部1（伦敦）	`wget "http://arms-apm-eu-west-1.oss-eu-west-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-eu-west-1.oss-eu-west-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
美国东部1（弗吉尼亚）	`wget "http://arms-apm-us-east-1.oss-us-east-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-us-east-1.oss-us-east-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
美国西部1（硅谷）	`wget "http://arms-apm-us-west-1.oss-us-west-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zipwget "http://arms-apm-us-west-1.oss-us-west-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-us-west-1.oss-us-west-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
亚太南部1（孟买）	`wget "http://arms-apm-ap-south-1.oss-ap-south-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-ap-south-1.oss-ap-south-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华东1金融云	无	`wget "http://arms-apm-cn-hangzhou-finance.oss-cn-hzjbp-b-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华东2金融云	无	`wget "http://arms-apm-cn-shanghai-finance-1.oss-cn-shanghai-finance-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`
华南1金融云	无	`wget "http://arms-apm-cn-shenzhen-finance-1.oss-cn-shenzhen-finance-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

地域	公网地址	VPC地址
政务云	`wget "http://arms-apm-cn-north-2-gov-1.oss-cn-north-2-gov-1.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`	`wget "http://arms-apm-cn-north-2-gov-1.oss-cn-north-2-gov-1-internal.aliyuncs.com/ArmsAgent.zip" -O ArmsAgent.zip`

- 将 `{LicenseKey}` 替换成您的License Key。将 `{AppName}` 替换成您的应用名称（应用名不可包含中文字符）。

步骤三：构建并启动新镜像

1. 运行 `docker build` 命令来构建镜像。

```
docker build -t registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-springboot-demo:v0.1 -f /{workspace}/Dockerfile /{workspace}/
```

> **说明** 将 `registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-springboot-demo:v0.1` 替换为真实镜像名称。

2. 运行 `docker run` 命令来启动镜像。

```
docker run -d -p 8081:8080 registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-springboot-demo:v0.1
```

> **说明** 将 `registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-springboot-demo:v0.1` 替换为真实镜像名称。

结果验证

约一分钟后，若您的应用名称出现在**应用列表**页面中且有数据上报，则说明接入成功。

卸载Agent

当您不需要再监控Docker集群中的Java应用时，请按照以下步骤卸载Agent。

1. 删除按照步骤二：集成已有镜像的说明添加的*Dockerfile*内容。
2. 运行 `docker build` 命令来构建镜像。
3. 运行 `docker run` 命令来启动镜像。

相关文档

- 应用监控常见问题

2.2.7. 通过OpenTelemetry接入ARMS

本文介绍如何将OpenTelemetry Trace数据接入ARMS并使用。

接入OpenTelemetry Trace数据

ARMS支持多种方式接入OpenTelemetry Trace数据，您可以将OpenTelemetry Trace数据直接上报至ARMS，或通过OpenTelemetry Collector转发。

- 通过ARMS Java Agent接入OpenTelemetry Trace数据
  Java应用推荐安装ARMS Java Agent。ARMS Java Agent内置了大量通用组件的链路埋点，能够自动上报OpenTelemetry格式的Trace数据，开箱即用，无需额外配置。具体操作，请参见监控Java应用。
- 结合ARMS Java Agent与OpenTelemetry Java SDK上报Trace数据
  v2.7.1.3及以上版本的ARMS Java Agent支持OpenTelemetry Java SDK扩展。您在使用ARMS Java Agent自动获取通用组件Trace数据的同时，还可以通过OpenTelemetry SDK扩展自定义的方法埋点。具体操作，请参见通过OpenTelemetry Java SDK进行手工埋点。
- 通过OpenTelemetry直接上报Trace数据
  您也可以使用OpenTelemetry SDK进行应用埋点，并通过Jaeger Exporter直接上报Trace数据。具体操作，请参见通过OpenTelemetry上报Java应用数据。

通过ARMS for OpenTelemetry Collector转发Trace数据

在容器服务ACK环境下，您可以一键安装ARMS for OpenTelemetry Collector，通过它进行Trace数据的转发。ARMS for OpenTelemetry Collector实现了链路无损统计（本地预聚合，统计结果不受采样率影响），动态配置调参，状态管理以及开箱即用的Trace Dashboard on Grafana，同时更加易用、稳定、可靠。

ARMS for OpenTelemetry Collector的接入流程如下：

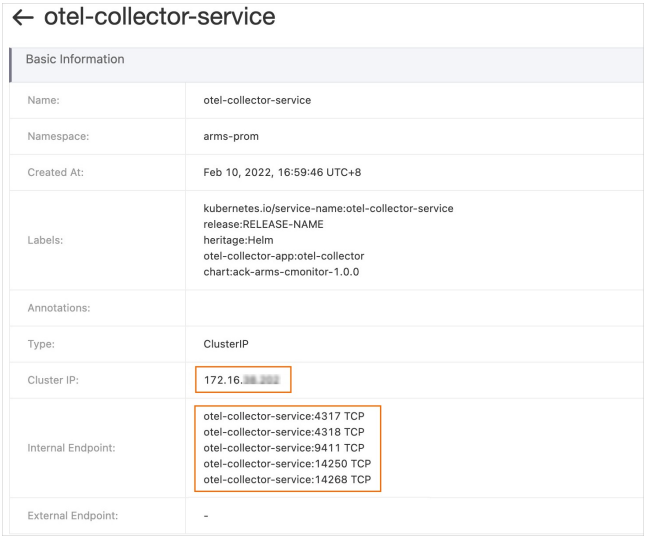
1. 通过ACK控制台应用目录安装ARMS for OpenTelemetry Collector。
   i. 登录容器服务管理控制台。
   ii. 在左侧导航栏选择**市场 > 应用市场**。
   iii. 在**应用市场**页面通过关键字搜索ack-arms-cmonitor组件，然后单击**ack-arms-cmonitor**。
   iv. 在**ack-arms-cmonitor**页面单击右上角的**一键部署**。
   v. 在**创建**面板中，选择目标集群，然后单击**下一步**。

      > **说明** 命名空间默认为arms-prom。

   vi. 单击**确定**。
   vii. 在左侧导航栏单击**集群**，然后单击刚刚安装了ack-arms-cmonitor组件的集群名称。
   viii. 在左侧导航栏选择**工作负载 > 守护进程集**，在页面顶部选择命名空间为arms-prom。

ix. 单击otel-collector-service。

查看otel-collector-service（Service）是否正常运行，如下图所示对外暴露了多种Receivers端口接收OpenTelemetry数据，则表示安装成功。



2. 修改应用SDK中的Exporter Endpoint地址为 `otel-collector-service:Port` 。

```go
// Initializes an OTLP exporter, and configures the corresponding trace and
// metric providers.
func initProvider() func() {
    ctx := context.Background()

    otelAgentAddr := "otel-collector-service:4317"

    metricClient := otlpmetricgrpc.NewClient(
        otlpmetricgrpc.WithInsecure(),
        otlpmetricgrpc.WithEndpoint(otelAgentAddr))
    metricExp, err := otlpmetric.New(ctx, metricClient)
    handleErr(err, "Failed to create the collector metric exporter")

    pusher := controller.New(
        processor.NewFactory(
            simple.NewWithExactDistribution(),
            metricExp,
        ),
        controller.WithExporter(metricExp),
        controller.WithCollectPeriod(2*time.Second),
    )
    global.SetMeterProvider(pusher)

    err = pusher.Start(ctx)
    handleErr(err, "Failed to start metric pusher")

    traceClient := otlptracegrpc.NewClient(
        otlptracegrpc.WithInsecure(),
        otlptracegrpc.WithEndpoint(otelAgentAddr),
        otlptracegrpc.WithDialOption(grpc.WithBlock()))
    traceExp, err := otlptrace.New(ctx, traceClient)
    handleErr(err, "Failed to create the collector trace exporter")
```

通过开源OpenTelemetry Collector转发Trace数据

使用开源的OpenTelemetry Collector转发Trace数据至ARMS，只需要修改Exporter中的接入点（Endpoint）和鉴权信息（Token）。

```
exporters:
  otlp:
    endpoint: <endpoint>:8090
    tls:
      insecure: true
    headers:
      Authentication: <token>
```

> 说明
>
> - 将 `<endpoint>` 替换为您上报区域对应的Endpoint，例如：`http://tracing-analysis-dc-bj.aliyuncs.com:8090` 。
> - 将 `<token>` 替换为您控制台上获取的Token，例如：`b590lhguqs@3a7*********9b_b590lhguqs@53d*****8301` 。

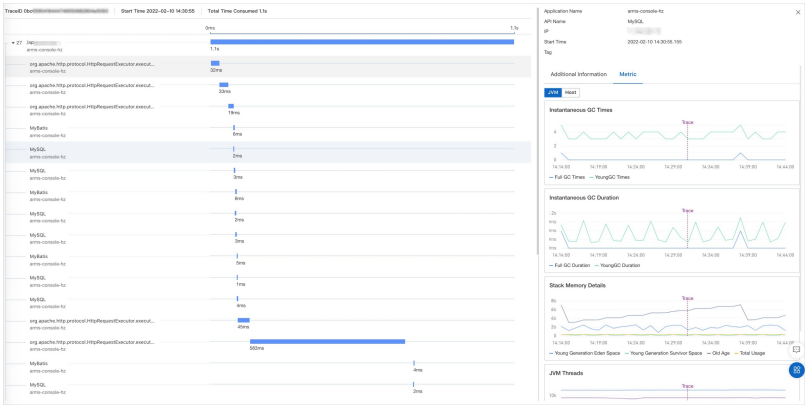
直接上报

通过OpenTelemetry Collector转发

OpenTelemetry Trace使用指南

为了更好的发挥OpenTelemetry Trace数据价值，ARMS提供了链路详情、预聚合大盘、Trace Explorer后聚合分析、调用链路关联业务日志等多种诊断能力。

- 链路详情
  在链路详情面板左侧可以查看链路的接口调用次序与耗时，面板右侧展示了详细的附加信息和关联指标，例如数据库SQL，JVM和Host监控指标等。

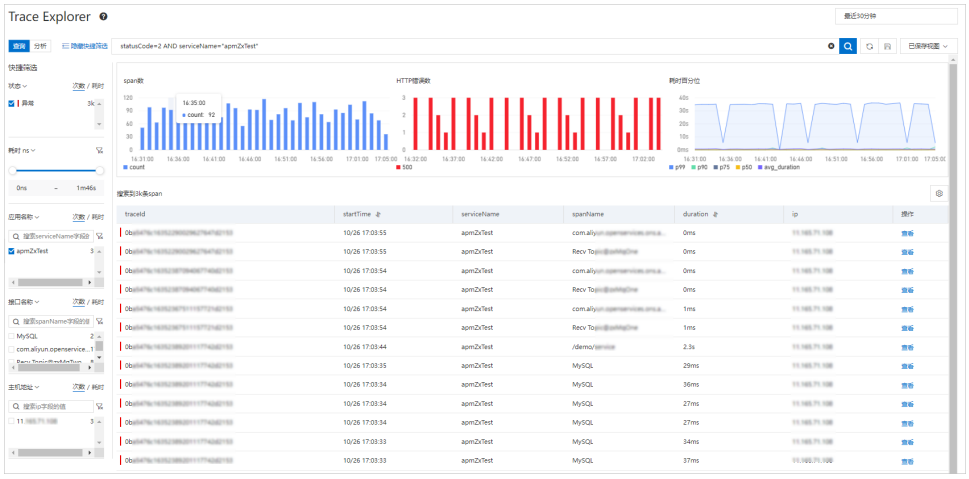


- 预聚合大盘
  ARMS基于OpenTelemetry Trace数据提供了多种预聚合指标大盘，包括应用总览，接口调用，数据库调用等。更多信息，请参见查看应用性能关键指标和拓扑图。

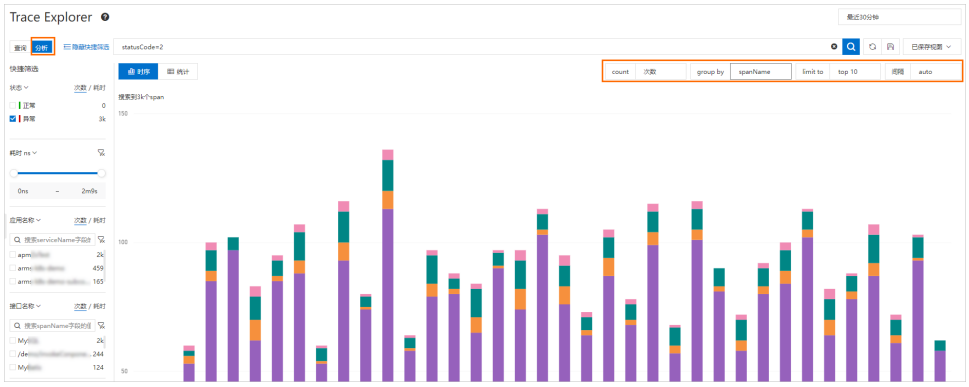
- Trace Explorer后聚合分析

针对OpenTelemetry Trace数据，ARMS提供了灵活的多维筛选与后聚合分析能力，例如查询特定应用的异常链路。还可以根据IP、接口等维度对Trace数据进行聚合。更多信息，请参见Trace Explorer。

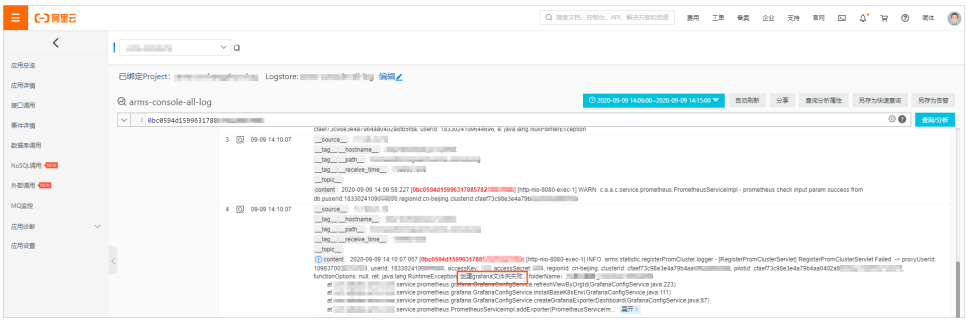
链路查询



链路分析



- 调用链路关联业务日志
  ARMS支持将OpenTelemetry Trace与业务日志相关联，从应用接口角度排查业务异常问题。更多信息，请参见通过调用链路和日志分析定位业务异常问题。



2.3. 开始监控除Java之外的应用

对于Java之外的应用，例如C++、Go、Node.js、.NET等，您可以使用链路追踪Tracing Analysis来监控。链路追踪具备分布式调用链追踪和汇总、应用性能实时汇总和分布式拓扑动态发现等能力，帮助您全方位监控应用。

背景信息

链路追踪可以帮助开发者快速分析和诊断分布式应用架构下的性能瓶颈，提高微服务时代下的开发诊断效率。链路追踪具备以下能力：

- 分布式调用链追踪和汇总：追踪分布式架构中的所有微服务用户请求，并将它们汇总成分布式调用链。
- 应用性能实时汇总：通过追踪整个应用程序的用户请求，来实时汇总组成应用程序的单个服务和资源。
- 分布式拓扑动态发现：针对您的分布式微服务应用和相关PaaS产品，链路追踪均可收集到分布式调用信息。
- 开源社区兼容：基于OpenTracing标准，兼容Jaeger、Zipkin等开源产品。
- 下游场景对接：收集的链路可直接用于日志分析，且可对接到MaxCompute等下游分析平台。

监控多语言应用

更多信息

- 查看应用列表
- 查看应用性能关键指标和拓扑图
- 查看应用详情
- 查看接口调用情况
- 查询调用链
- 调用链分析
- 管理应用和标签

3.控制台功能

3.1. 应用监控3D拓扑图

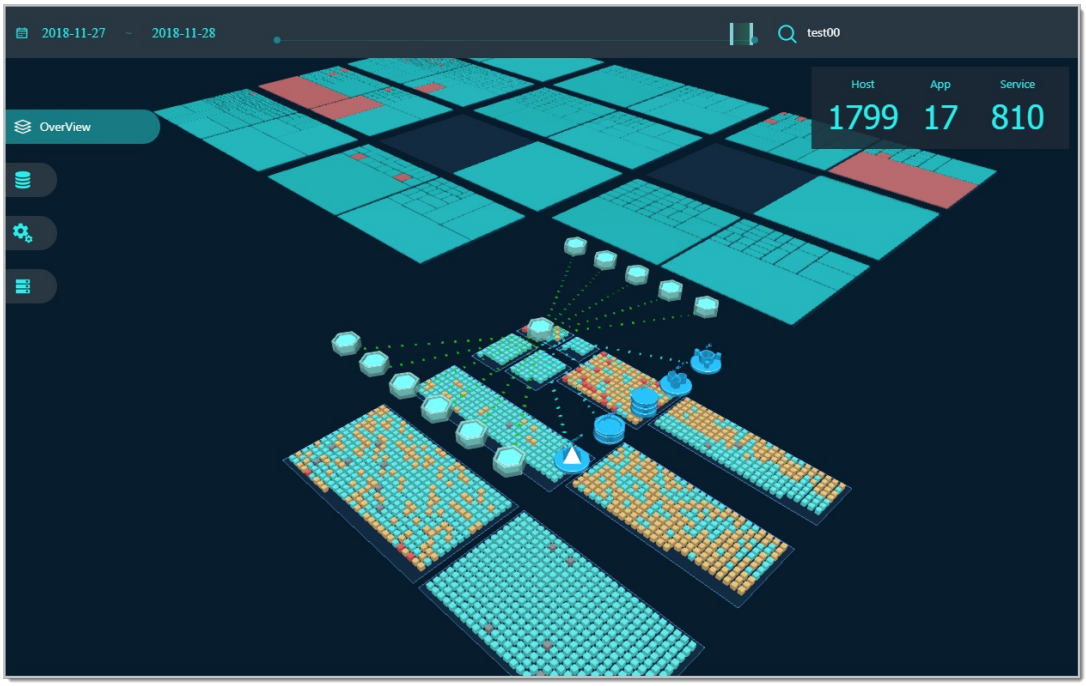
ARMS应用监控3D拓扑图能立体展示应用、服务和主机的健康状况，以及应用的上下游依赖关系。借助3D拓扑图，您可以快速定位诱发故障的服务、被故障影响的应用和关联的主机等，全方位地诊断故障根源，从而快速排除故障。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用**操作**列的**3D拓扑**。

总览层（Overview）

在默认展示的Overview页面上，您可以看到服务层、应用层和主机层的全部内容。页面右上角显示的是主机、应用和服务的数量。



在总览层，您可以执行以下操作：

- 在页面左上角单击时间范围区域，然后在弹出的时间范围选择器内选择精确的起止时间。
- 在页面顶部的时间轴上，随意拖动时间滑块来改变当前视图对应的时间范围。
- 在页面右上角的搜索框内，输入关键字并按回车进行搜索。
- 用鼠标拖放，以任意角度查看三层数据。
- 单击视图中的任意对象，在右侧面板中查看该对象的相关指标。

服务层（Service）

服务层展示应用所依赖的服务信息。

每个应用下的服务对应一个板块。调用次数越多，所占面积越大。服务的不同状态以不同颜色表示。

- ：服务调用正常
- ：服务出错率较高
- ：服务无返回数据

> **说明** 服务的响应时长阈值是可以配置的。在左侧导航栏中单击服务层（Service）右侧的三角形图标，即可展开耗时阈值设置框。在该设置框中拖动滑块即可设置阈值。

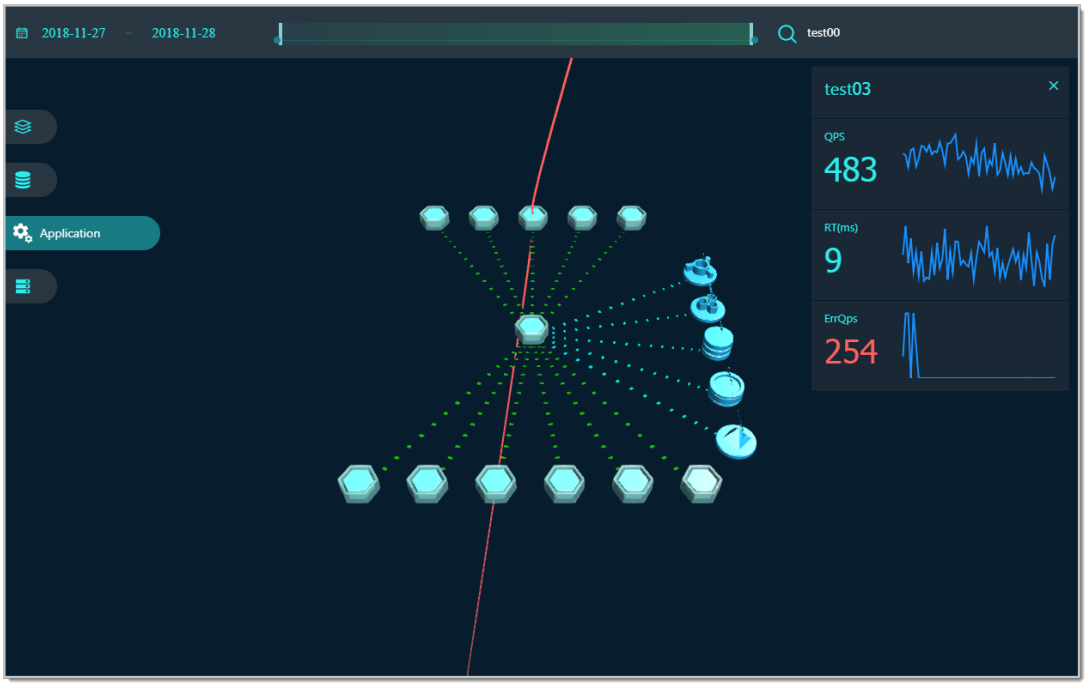
单击一个服务后，右侧面板将展示该服务的以下信息：

- 服务名称
- QPS：Query Per Second（每秒查询数）
- RT(ms)：Response Time（响应时间，单位为毫秒）
- Error：Error QPS（每秒错误查询数）

> **说明** 在QPS、RT(ms)和Error区域框中，左侧的数值是所选时间范围内的平均值，右侧是所选时间范围内的曲线图。

应用层（Application）

应用层展示应用及其上下游依赖关系，包括依赖的中间件在内。通过连接线的流向，您可以看到调用方向。



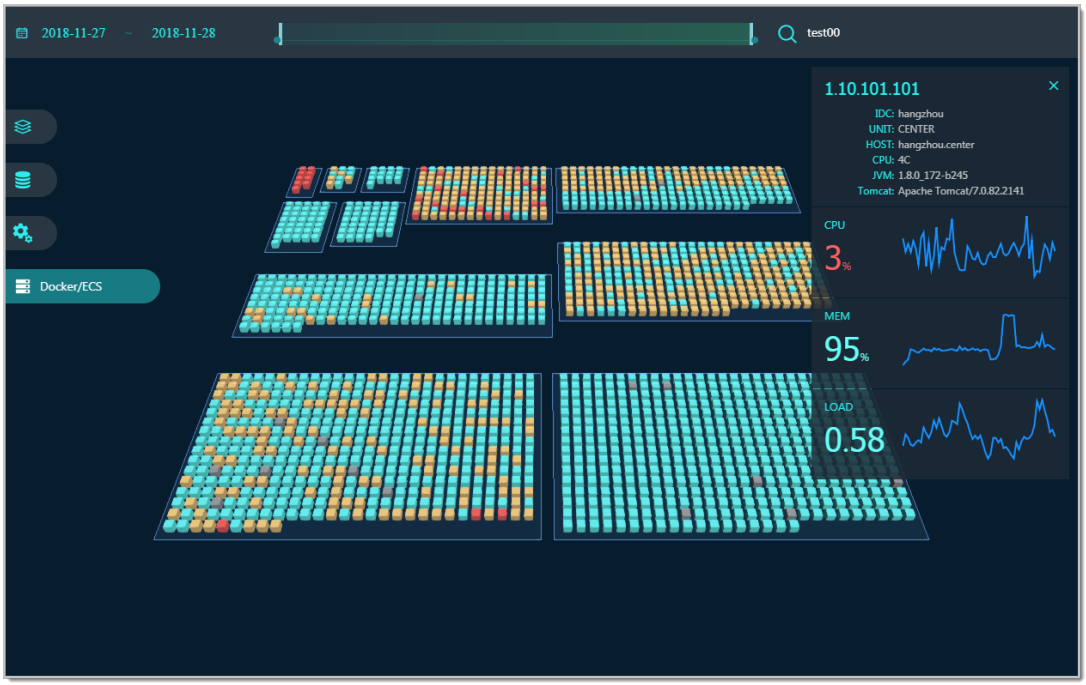
单击一个应用后，右侧面板将展示该应用的以下信息：

- 应用名称
- QPS：Query Per Second（每秒查询数）
- RT(ms)：Response Time（响应时间，单位为毫秒）
- Error：Error QPS（每秒错误查询数）

> **说明** 在QPS、RT(ms)和Error区域框中，左侧的数值是所选时间范围内的平均值，右侧是所选时间范围内的曲线图。

主机层（Docker/ECS）

主机层展示应用的主机详情。



每个方块代表一个主机。所有主机按应用分区。主机的不同状态以不同颜色表示。

- ：正常
- ：缓慢
- ：警告
- ：异常
- ：离线

单击一个主机后，右侧面板将展示该主机的以下信息：

- 主机IP地址及基础信息：
  - 响应时间
  - 请求数
  - 错误数
- CPU：CPU使用率
- MEM：内存使用率
- DISK：磁盘使用率
- GC TIME：GC耗时
- GC COUNT：GC次数

> **说明** 在CPU、MEM和DISK区域框中，左侧的数值是所选时间范围内的平均值，右侧是所选时间范围内的曲线图。

3.2. 调用链路查询

在**调用链路查询**页面，您可以通过TraceId精确查询调用链路详细情况，或结合多种条件筛选查询调用链路。也可以对多条调用链进行聚合分析。

查询调用链路

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 调用链路查询**，并在顶部菜单栏选择目标地域。
3. 在**调用链路查询**页面选择**参数类型**，在**参数值**中填入自定义标签，单击**添加到查询条件**。
   可以使用TraceId参数进行精确查询。
   参数类型说明

参数类型	描述
TraceId	输入TraceId。
接口名称	应用调用的接口名称，不支持模糊搜索。
客户端应用名	客户端的应用名称。
服务端应用名	服务端的应用名称。
耗时大于	调用的耗时大于指定毫秒数。
调用类型	选择调用类型。
是否异常调用	所有包含异常调用的链路。
仅含线程剖析快照	所有包含线程剖析快照的链路。
客户端IP	调用发起应用IP。
服务端IP	请求被调用的应用IP。
业务主键	搜索业务事件所使用的字段。
响应码	输入响应码。

4. 单击需要查看的TraceID名称，进入**调用链路**页面。

调用链路页面字段说明如下：

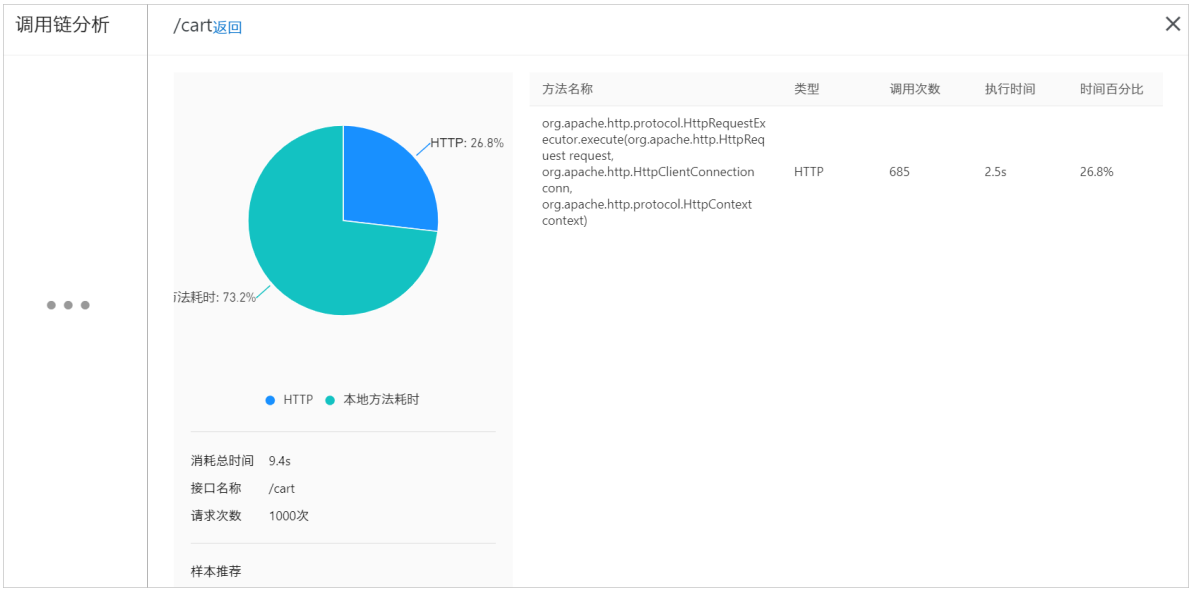
- 应用名称：所属应用名称。
- 日志产生时间：日志产生的时间。
- 状态： 红色表示该服务调用的本地调用链路中存在异常，绿色表示正常。
- IP地址：该应用的IP地址。
- 调用类型：该次调用的调用类型，与即席查询的调用类型选项对应。
- 服务名称：该次调用的服务接口名称。
- 时间轴：各服务间调用链路的耗时，以及相对于整条调用链路的耗时分布。

分析调用链路

在**调用链路查询**页面选中所有需要分析的调用链路，单击**分析选中调用链路**。

在**调用链分析**面板可以查看所有选中调用链路的Span名称、应用名、调用类型、请求数/请求比例、异常数/异常比例、平均自身耗时和平均耗时。

- 将鼠标放在Span名称上，可以查看包含该Span的TraceID。
- 单击应用名可以进入该应用的**应用总览**页面。
- 单击目标Span**操作**列的**统计分析**，可以查看该Span的详细信息，包括各接口的调用类型占比、消耗总时间、接口名称、请求次数、样本推荐和所有调用方法的名称、类型、调用次数、执行时间和时间百分比。



相关操作

在**调用链路**页面，单击**指标监控**列的折线图，可以查看不同时间段的请求数、响应时间和错误数。

在**调用链路**页面，单击**方法栈**列的图标，进入**方法栈**对话框。

方法栈对话框字段说明如下：

- 调用方法：本地方法栈调用方法，展开后显示的是该方法的下一层调用。
- 行号：本地方法的代码所在行数。
- 扩展信息：
  - 参数：调用的输入参数等
  - SQL：数据库调用的SQL语句等
  - 异常：抛错的信息等
- 时间轴：本地调用链路每次方法调用的时间分布。

3.3. Trace Explorer

ARMS应用监控的链路分析Trace Explorer功能是基于已存储的全量链路明细数据，自由组合筛选条件与聚合维度进行实时分析，可以满足不同场景的自定义诊断需求。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > Trace Explorer**，然后在顶部菜单栏选择目标地域。

   首次进入**Trace Explorer**页面需要授权ARMS操作SLS权限，以便数据投递到您名下的SLS进行二次开发，但不会产生额外费用。
   授权完成后会自动进行资源准备，预计等待2分钟即可完成。

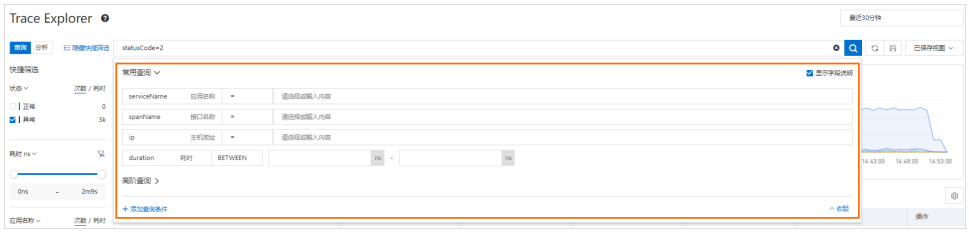
   > **说明**
   > - 开通Trace Explorer功能不会产生额外费用。
   > - 如果开通失败，请单击重试或联系技术支持。

3. 在**Trace Explorer**页面右上角的时间选择框设置需要查询的时间段。
4. 筛选链路。
   - 在左侧**快捷筛选**区域，通过状态、耗时、应用名称、接口名称和主机地址维度快速筛选链路。

筛选条件将会显示在页面右侧顶部文本框内。

- 单击右侧顶部文本框，在下拉弹窗中修改筛选条件或设置其他维度的筛选条件。

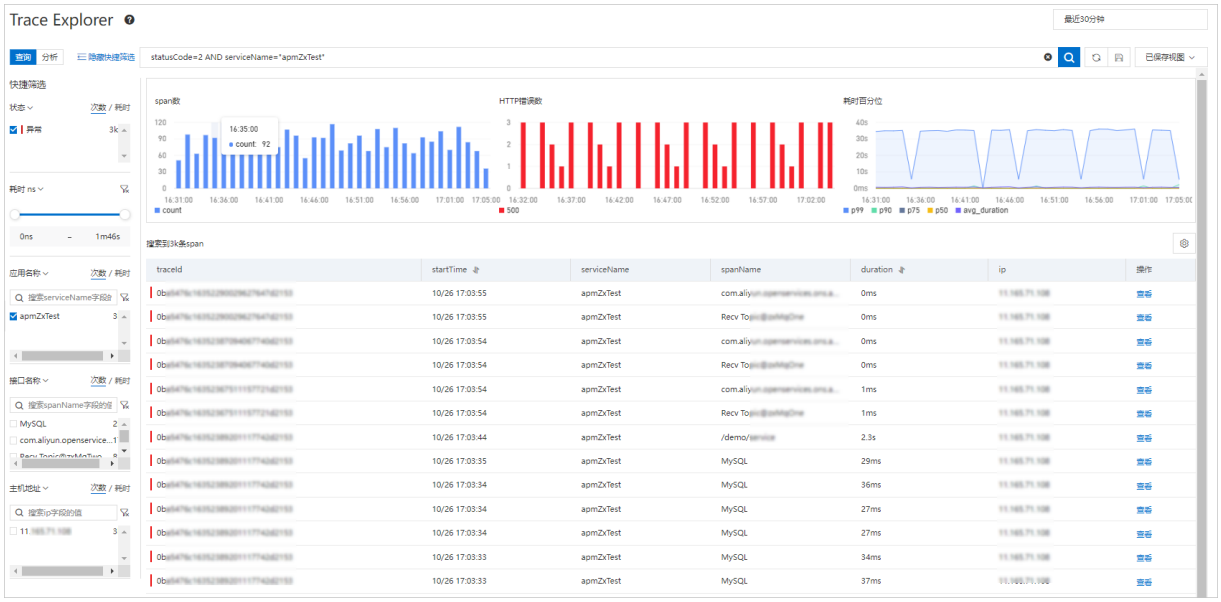


> **说明**
>
> - 单击文本框右侧的 图标可以保存当前筛选条件。
> - 单击文本框右侧**已保存视图**可以查看已保存的筛选条件，单击目标的筛选条件可以快速查看到对应筛选条件下的链路信息。

- 在右侧顶部文本框直接输入查询条件。查询语法说明，请参见Trace Explorer查询用法说明。

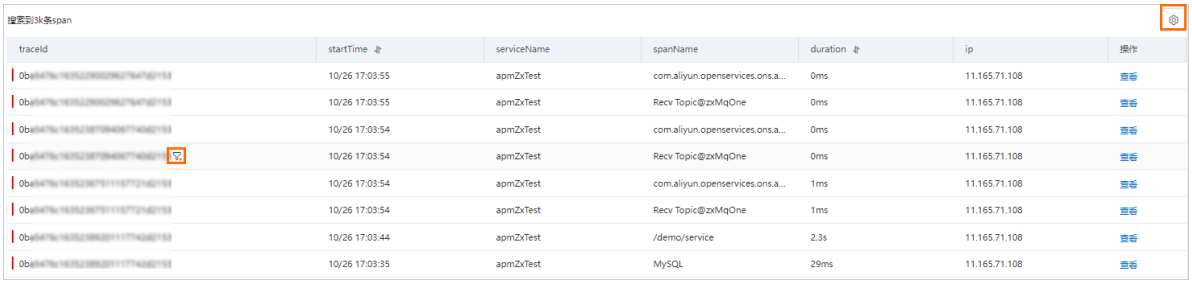
查询链路

筛选设置完成后，**Trace Explorer**页面将会显示筛选过滤后的链路查询信息，包括Span数和HTTP错误数的柱状图，耗时百分位的时序曲线，以及Span列表。



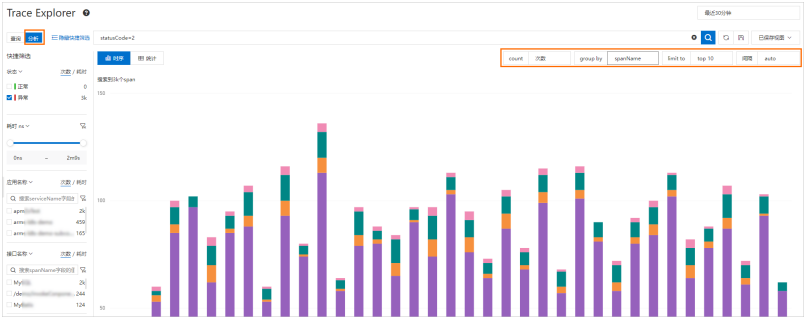
在Span列表区域，您可以执行以下操作：

- 单击目标Span右侧**操作**列的**查看**，可以查看完整的链路信息。更多信息，请参见链路详情。
- 单击右上角的 图标可以增加或隐藏列表显示的参数。
- 将鼠标悬浮于Span参数上，单击右侧的图标，可以将当前参数值添加为筛选条件。



分析链路

筛选设置完成后，在**Trace Explorer**页面左上角单击**分析**，在右侧数据展示区域右上角，设置对Span的分析条件。



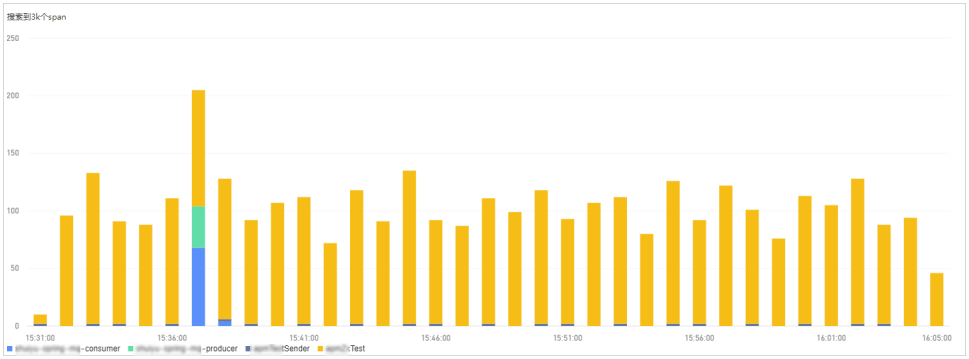
- count：显示Span被调用次数。
- group by：按指定分组显示Span。
- limit to：设置显示分析数据的Top 10、Top 5、Bottom 10或Bottom 5。

> **说明** group by设置为**no group**时不支持设置limit to。

- 间隔：按指定时间间隔分析Span。

设置完成后，可以查看筛选过滤后的Span被调用次数的时序图、具体统计数据、全链路聚合和全链路拓扑。

时序图



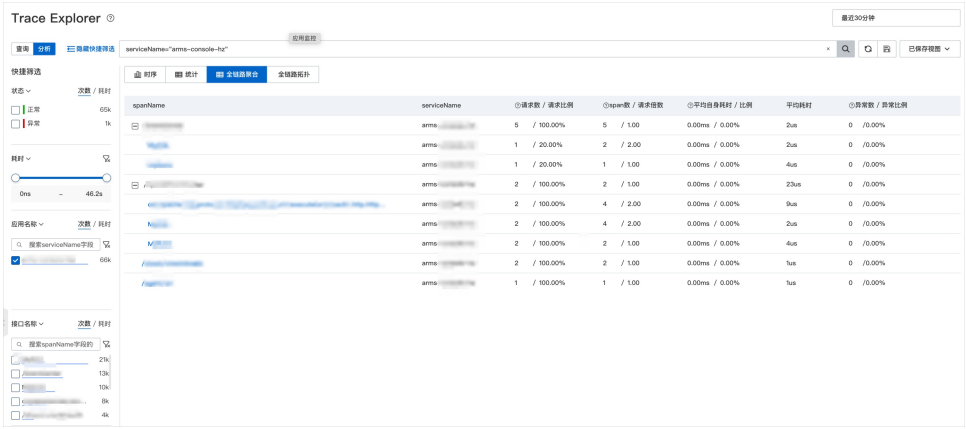
> **说明**
>
> - 将鼠标悬浮于时序图上，可以查看对应分组下Span被调用的次数。
> - 单击时序图底部的分组指标，可以隐藏或显示对应指标数据。

统计数据

全链路聚合

Trace Explorer对查询到的Span可以按照各个维度进行分析，但这些分析是针对单个Span，并未在链路级别深度分析。而全链路聚合功能支持通过指定条件查询分布式调用链路的TraceId（最多5000个），然后基于这些TraceId查询对应的Span，并聚合这些Span得出最终结果，整个过程保证聚合的链路完整性。

> **说明** 由于全链路聚合是按照查询条件后聚合计算相应数据的，当您选择的条件比较多时，查询计算存在一定延迟，请耐心等待。



参数	说明
spanName	Span名称。
serviceName	Span对应的应用名。
请求数/请求比例	请求比例表示调用当前Span节点的请求比例数。 例如10%表示10%的请求会调用当前Span。 计算公式：请求比例=当前Span的请求数/总请求数*100%
span数/请求倍数	请求倍数表示平均每个请求调用当前Span的次数。 例如1.5表示每个请求会调用当前Span 1.5次 。 计算公式：请求倍数=Span数/Span的请求数
平均自身耗时/比例	平均自身耗时表示不包括子Span的耗时。 例如，对于Span A和其子Span B， 其中A耗时为10 ms， B耗时为8 ms，那A的自身耗时为2 ms。 计算公式：自身耗时=Span耗时-所有子Span耗时总和 > **注意** 如果是异步调用，自身耗时即Span耗时，无需减去子Span耗时。
平均耗时	该Span的平均耗时。

参数	说明
异常数/异常比例	异常比例表示出现异常的请求比例。 例如3%表示有3%的请求出现异常。 计算公式：异常比例=异常请求数/总请求数 **注意** 异常请求数不等于异常数（Span调用异常的次数），当请求倍数大于1时，一个异常请求可能对应多个异常数。

示例：如下表所示，Span A调用Span B和Span C，各参数含义如下。

spanName		serviceName	请求数/请求比例	span数/请求倍数	平均自身耗时/比例	平均耗时	异常数/异常比例
A	-	demo	10/100.00%	10/1.00	5.00ms/25.00%	20ms	2/20.00%
-	B	demo	4/40.00%	8/2.00	16.00ms/100.00%	16ms	2/50.00%
-	C	demo	1/10.00%	1/1.00	4.00ms/100.00%	4ms	1/100.00%

对于入口Span，A的请求数/请求比例表示A的请求总数为10次，比例为100%。B的请求数/请求比例为4/40.00%，表示只有4次请求调用了B，同理只有1次请求调用了C，对应的请求比例分别为40%和10%。其余的请求可能因逻辑判断或者异常而未调用B和C。这里反映了请求的分布比例。

A的span数/请求倍数为10/1.00，表示每次请求只调用了一次A，但是对于B而言，4次请求有8个Span，每次请求调用了2次B。这里反应了一次请求中Span的分布比例。

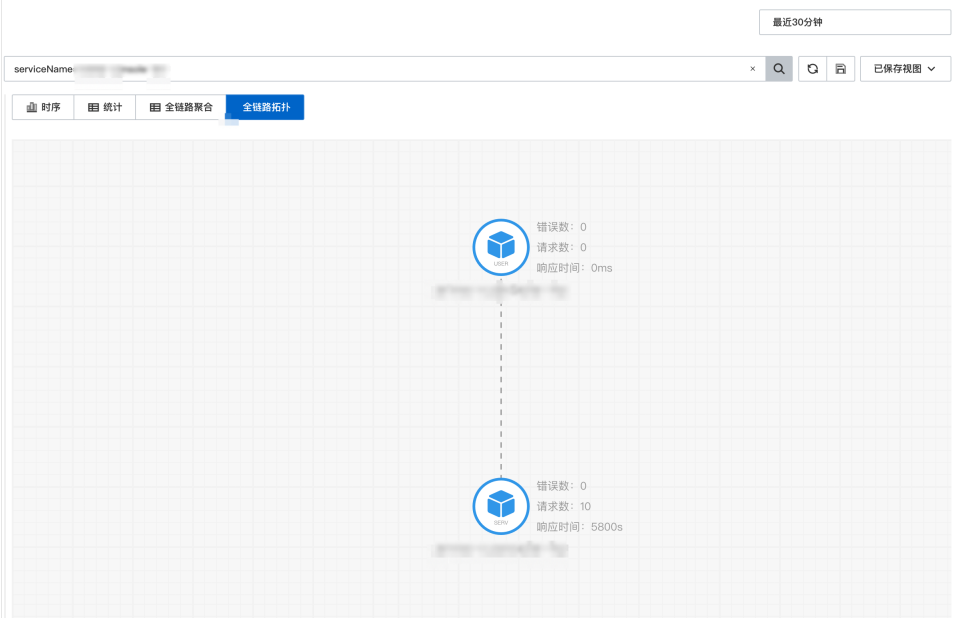
A的平均自身耗时/比例为5.00ms/25.00%，表示A除了B和C之外的平均耗时为5ms，只占整体平均耗时的25%。而子Span B和C因为没有子Span，所以自身耗时即整体耗时。这里反应了耗时的分布比例。

A的异常数/异常比例为2/20.00%，表示A发生了2次异常，占整体请求的20%。B的异常数/异常比例为2/50.00%，因为每次请求调用了2次B，总的请求数是4，异常比例是50%，那么2次请求发生了异常。所以B的分布可能是：一共有4次请求，其中有2次请求调用的4个Span B都是正常的，剩下2次请求中，首次Span B的调用都发生异常，然后重新调用成功。

说明 如果需要查看具体的调用链详情，可以将鼠标悬浮于蓝色的Span名称上，在悬浮框中可以看到推荐的调用链ID，单击TraceId即可查看。

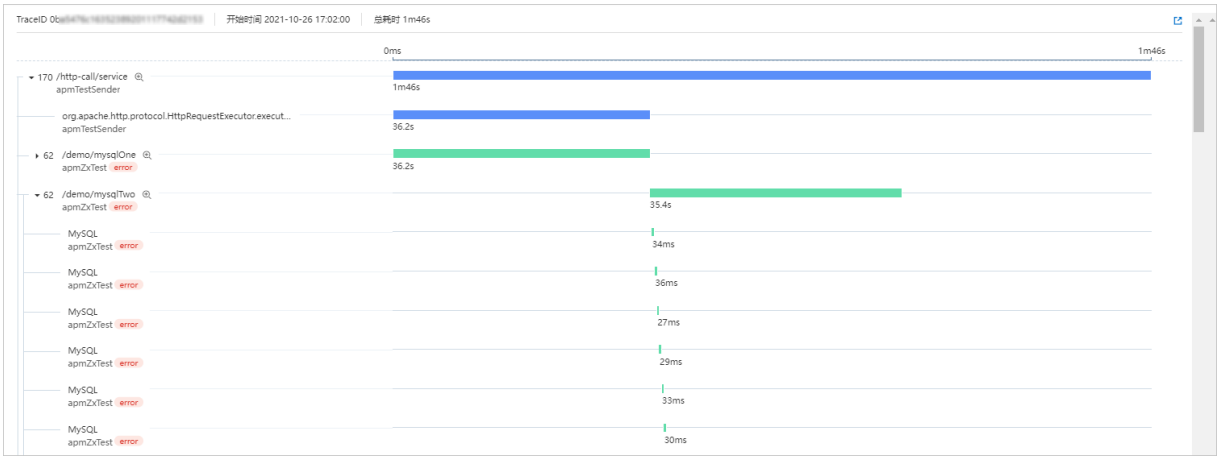
全链路拓扑

全链路拓扑页签显示调用链聚合后的应用间拓扑。如下图所示，表示两个应用间存在调用关系，同时展示相应的请求数、错误数、响应时间等数据。



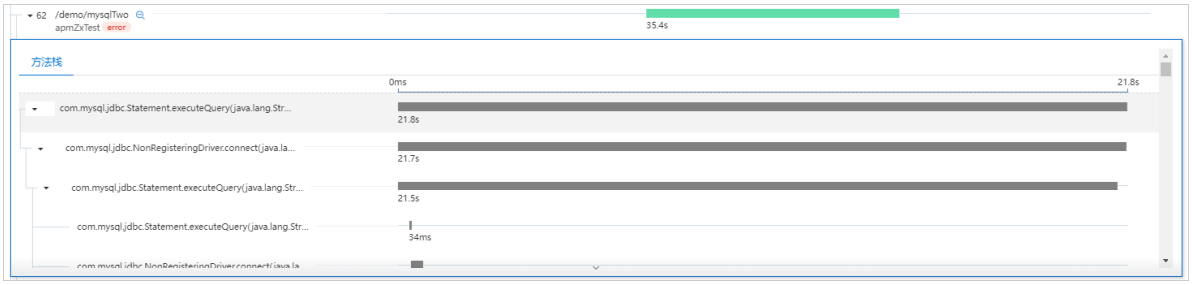
链路详情

在链路详情面板可以查看链路下的所有接口、链路开始时间、总耗时、接口是否异常、以及每个接口的调用耗时。



在链路详情面板，您可以执行以下操作：

- 将鼠标悬浮于接口名称上，可以查看接口对应的应用名、接口名、IP和开始时间。
- 单击目标接口右侧的图标，可以查看目标接口下的方法栈信息。

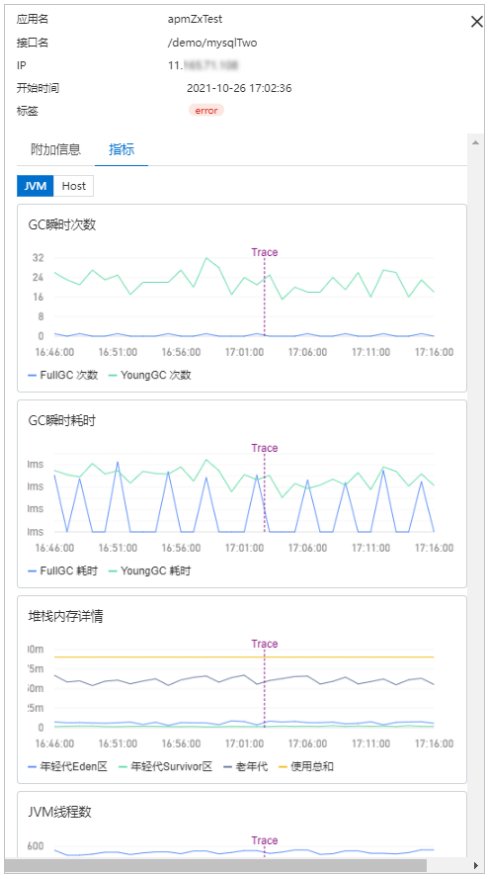


- 单击接口名称，可以在面板右侧查看对应接口的附加信息和指标详情。附加信息常见参数请参见Trace Explorer参数说明。JVM和Host详情请参见JVM监控和主机监控。

接口附加信息

接口附加信息

接口指标详情



二次开发

链路数据保存在您的SLS中，Project名称为proj-xtrace-<encode>-<region-id>，logstore名称为logstore-tracing。其中<region-id>是Trace Explorer对应的地域，例如cn-hangzhou。数据格式的含义，请参见Trace Explorer参数说明。您可以基于已存储的全量链路明细数据进行二次开发，自由组合筛选条件与聚合维度进行实时分析，可以满足不同场景的自定义诊断需求。更多使用方法，请参见通过TraceExplorer实时分析链路数据。

3.4. 应用总览

应用总览页面显示应用的关键指标、上下游依赖组件以及3D拓扑图，帮助您掌握应用的总体健康情况。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**中选择您想查看的应用，进入**应用总览**页面。
   您可以在**应用总览**页面顶部选择**概览分析**、**拓扑图**和**3D拓扑**页签查看相应信息。

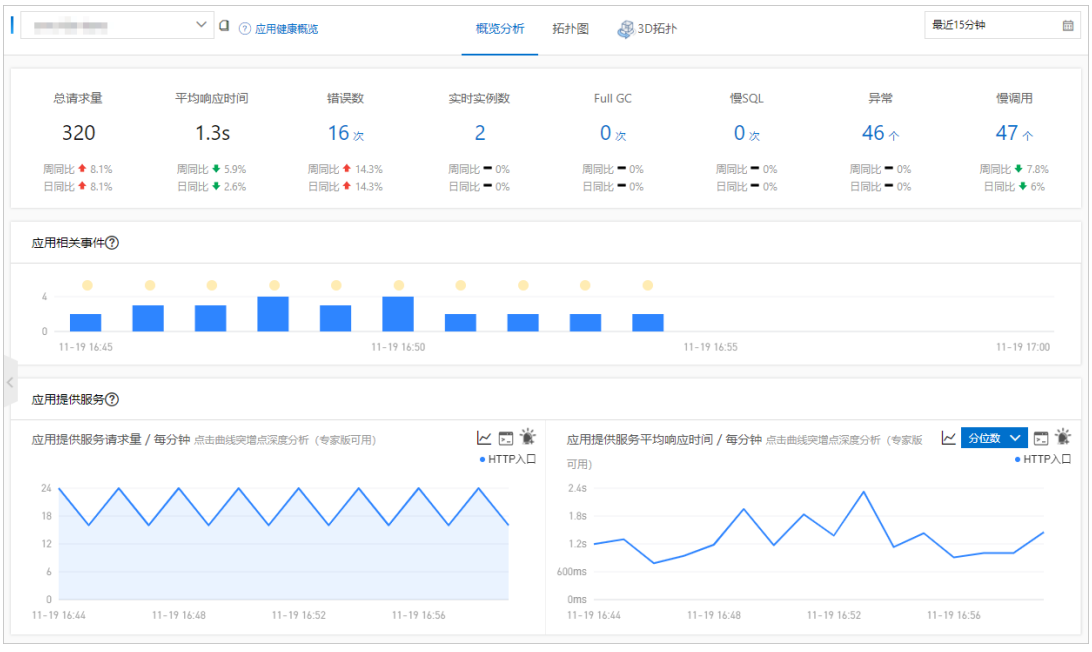
概览分析

概览分析页签上展示以下关键指标：

- 选定时间内的总请求量、平均响应时间、错误数、实例数、Insights、Full GC次数、慢SQL次数、异常次数和慢调用次数，以及这些指标和上周、上一天的同比升降幅度。
- 应用相关事件：应用相关的事件，比如0-1报警，应用监控报警，K8s集群事件等。
- 应用提供服务：应用提供服务的请求量和平均响应时间的时序曲线。

> **说明** 如需查看平均响应时间各分位数曲线，请先在**自定义配置**页面的**高级设置**区域开启**分位数统计**开关。具体操作，请参见自定义配置。

- 应用依赖服务：应用依赖服务的请求量、平均响应和应用实例数的时序曲线，以及HTTP-状态码统计。
- 系统信息：CPU、MEM和负载的时序曲线。
- 慢调用：慢调用的时序曲线和调用详情。
- 统计分析：接口慢调用分析和异常类型分析。

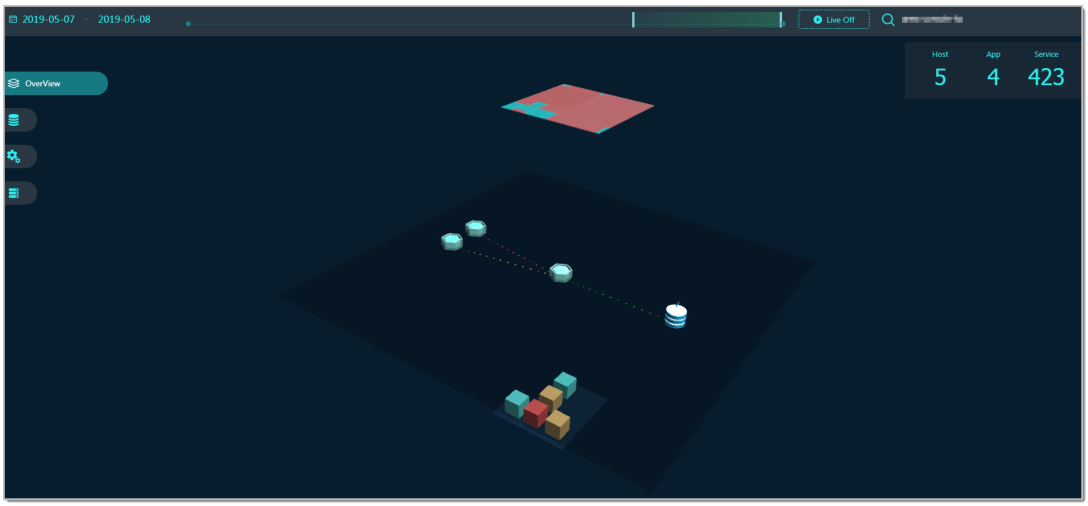


应用拓扑

在**拓扑图**页签上，您可以通过拓扑图更加直观地看到应用的上下游组件以及与它们的调用关系，从而更快速地找出应用的瓶颈。

3D拓扑

在**3D拓扑**页签上，立体地展示了应用、服务和主机的健康状况，以及应用的上下游依赖关系。借助3D拓扑图，您可以快速定位诱发故障的服务、被故障影响的应用和关联的主机等，全方位地诊断故障根源，从而快速排除故障。3D拓扑详细介绍，请参见应用监控3D拓扑图。



3.5. 应用详情

3.5.1. 概览

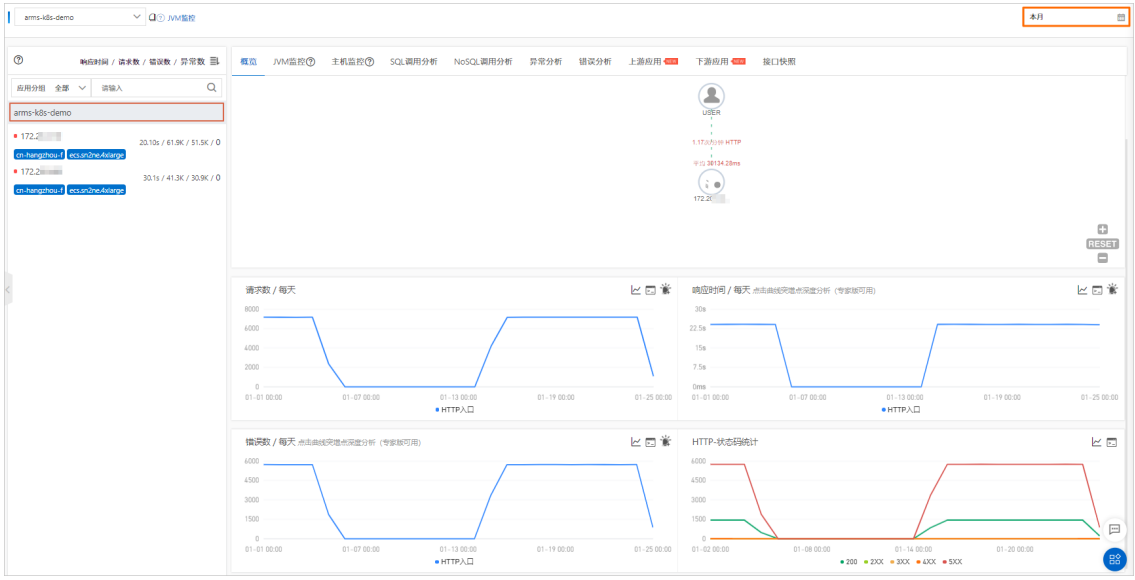
本文说明如何查看应用概览，从而了解应用拓扑、请求数、响应时间、错误数、HTTP状态码等信息。

前提条件

接入应用监控

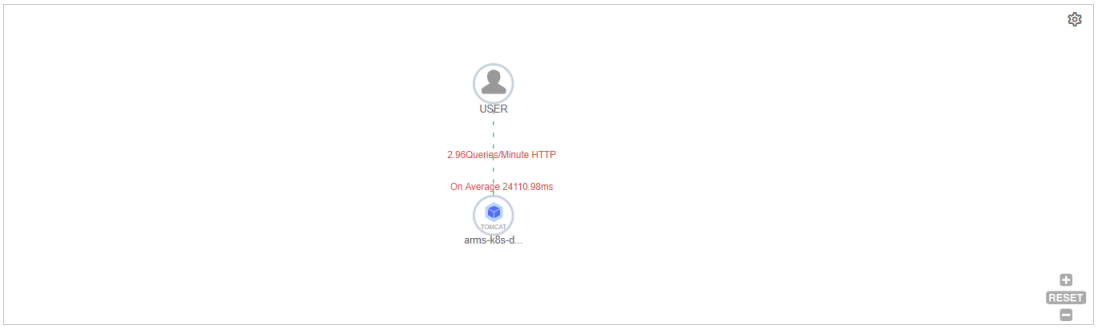
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择应用实例，设置时间段，单击**概览**页签。



应用拓扑

应用拓扑区域显示该应用在指定时间段的内部服务的调用关系拓扑图。



1. （可选）在应用拓扑区域，您可以执行以下操作：
   - 单击图标，设置应用拓扑图的显示设置。

     > **说明** 设置会被浏览器存储，下次依然生效。

   - 单击加号图标或向上滑动鼠标滚轮，放大应用拓扑图。
   - 单击减号图标或向下滑动鼠标滚轮，缩小应用拓扑图。
   - 单击RESET图标，将应用拓扑图恢复至默认大小。

请求数

请求数区域显示该应用在指定时间段的请求数时序曲线。

1. （可选）在**请求数**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。
   - 单击图标，为该指标创建报警。具体操作，请参见创建报警。

响应时间

响应时间区域显示该应用在指定时间段的响应时间时序曲线。

1. （可选）在**响应时间**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击曲线突增点，进行深度分析。

     > **说明** 仅专家版支持该功能。

   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。
   - 单击图标，为该指标创建报警。具体操作，请参见创建报警。

错误数

错误数区域显示该应用在指定时间段的错误数时序曲线。

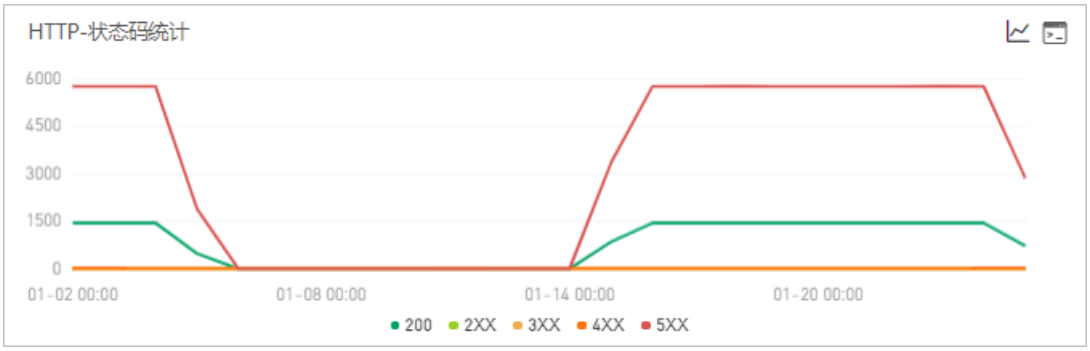
1. （可选）在**错误数**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击曲线突增点，进行深度分析。

     > **说明** 仅专家版支持该功能。

   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。
   - 单击图标，为该指标创建报警。具体操作，请参见创建报警。

HTTP状态码

HTTP状态码区域显示该应用在指定时间段的HTTP状态码时序曲线。



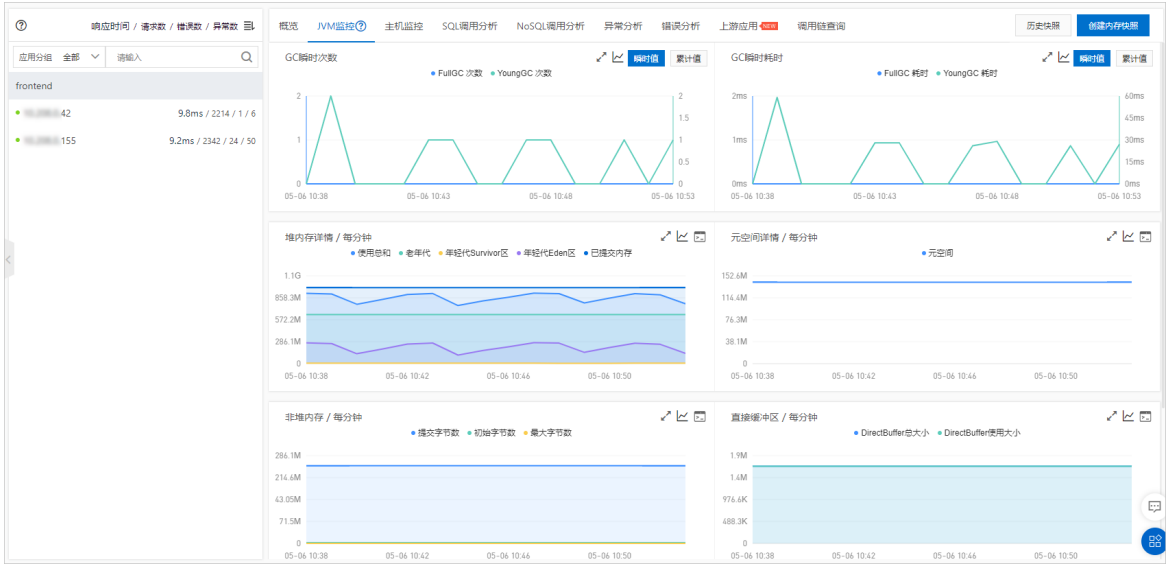
1. （可选）在**HTTP-状态码统计**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图例，隐藏或显示数据。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。

3.5.2. JVM监控

JVM监控功能用于监控重要的JVM指标，包括堆内存指标、非堆内存指标、直接缓冲区指标、内存映射缓冲区指标、GC（Garbage Collection）累计详情和JVM线程数等。本文介绍JVM监控功能和查看JVM监控指标的操作步骤。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用管理 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**中单击您想查看的应用。
4. 在左侧导航栏单击**应用详情**。
5. 在**应用详情**页面选择您想查看的实例，并在页面右侧单击**JVM监控**页签。



查看JVM监控指标

JVM监控页签内展示了GC瞬时次数、GC瞬时耗时、堆内存详情、元空间详情、非堆内存、直接缓冲区和JVM线程数的时序曲线。

- 单击**GC瞬时次数/每分钟**和**GC瞬时耗时/每分钟**面板右上角的**瞬时值**和**累计值**按钮，可以切换查看GC瞬时次数和GC瞬时耗时的时序曲线。
- 单击各监控面板上的指标名称（例如FullGC次数），可打开或关闭该指标在图表中的可见性。

> **说明** 每个图表必须至少有一个指标设为可见，这意味着当图表中只有一个指标时，您无法关闭该指标的可见性。

- 单击**堆内存详情/每分钟**、**元空间详情/每分钟**、**非堆内存/每分钟**、**直接缓冲区/每分钟**和**JVM线程数/每分钟**的右上角的查看API按钮，可看该监控指标的API详情。

功能介绍

JVM监控功能可监控以下指标：

- GC（垃圾收集）瞬时和累计详情
  - FullGC次数
  - YoungGC次数
  - FullGC耗时

- YoungGC耗时

- 堆内存详情
  - 堆内存总和
  - 堆内存老年代字节数
  - 堆内存年轻代Survivor区字节数
  - 堆内存年轻代Eden区字节数
  - 已提交内存字节数

- 元空间
  元空间字节数
- 非堆内存
  - 非堆内存提交字节数
  - 非堆内存初始字节数
  - 非堆内存最大字节数

- 直接缓冲区
  - DirectBuffer总大小（字节）
  - DirectBuffer使用大小（字节）

- JVM线程数
  - 线程总数量
  - 死锁线程数量
  - 新建线程数量
  - 阻塞线程数量
  - 可运行线程数量
  - 终结线程数量
  - 限时等待线程数量
  - 等待中线程数量

3.5.3. 池化监控

您可以通过池化监控功能监控具体应用所使用的线程池的各项指标，包括核心线程数量、当前线程数量、最大线程数量、活动线程数量、提交任务数量和任务队列容量。

前提条件

接入应用监控

> **注意** 仅专家版支持池化监控功能。

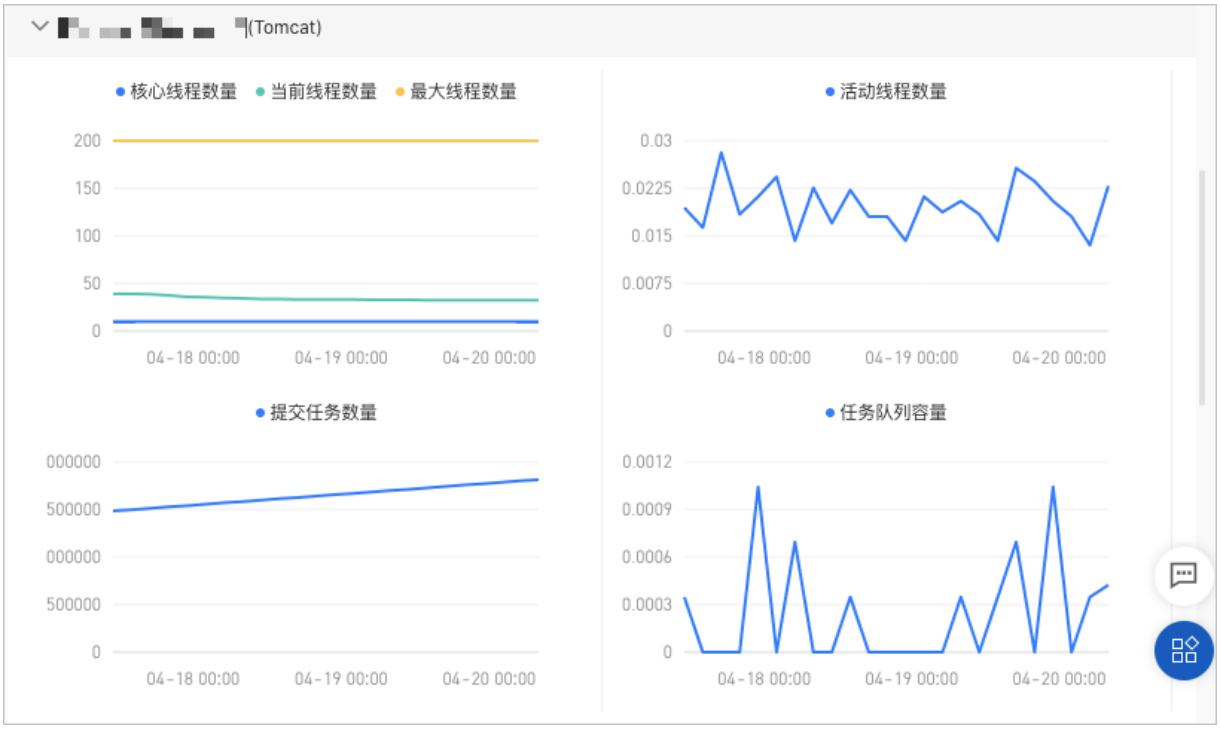
开启池化监控

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在顶部菜单栏，选择地域，然后单击目标应用名称。
3. 在左侧导航栏单击**应用设置**。
4. 在**自定义配置**页签下的**高级设置**区域内，打开**线程池监控**的开关。

> **注意** 池化监控支持Tomcat/Dubbo/HSF/Druid/HikariCp/SchedulerX等框架的线程池指标监控，您需要将Agent升级至最新版本。详细信息，请参见升级探针。应用重启后，池化监控功能生效。

查看池化监控指标

开启池化监控功能后，您可以在**应用详情 > 池化监控**下查看各项池化监控指标。**池化监控**页签包含以下指标的时序曲线：核心线程数量、当前线程数量、最大线程数量、活动线程数量、提交任务数量和任务队列容量。



监控指标	说明
核心线程数量	线程池内核心线程数量。
当前线程数量	线程池内当前存在的线程数量。
最大线程数量	线程池可容纳的最大线程数量。
活动线程数量	当前正在处理任务的线程数量。
提交任务数量	已提交到线程池内的任务数量。
任务队列容量	线程池的阻塞队列大小。

> **说明** 当线程池内当前线程数少于核心线程数时，即使存在空闲线程，在提交新任务时也会创建新的线程执行该任务。当线程池内线程数超过核心线程数量，但未超过最大线程数量，且任务队列容量已满时，提交的新任务会通过创建新的线程来执行。线程池内线程数不会超过最大线程数量。

在**池化监控**页签下，您可以执行以下操作：

- 将光标移到统计图上，查看统计情况。
- 单击各监控面板上的指标名称（例如核心线程数量），打开或关闭该指标在图表中的可见性。

> **说明** 每个图表必须至少有一个指标设为可见，因此当图表中只有一个指标时，您无法关闭该指标的可见性。

3.5.4. 主机监控

主机监控功能用于监控CPU、内存、Disk（磁盘）、Load（负载）、网络流量和网络数据包的各项指标。本文介绍主机监控功能和查看主机监控指标的操作步骤。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面选择您想查看的应用。
4. 在左侧导航栏单击**应用详情**。
5. 在**应用详情**页面选择您想查看的节点，并在页面右侧单击**主机监控**页签。

查看主机监控指标

主机监控页签展示了CPU、内存、Disk（磁盘）、Load（负载）、网络流量和网络数据包的时序曲线。

- 单击各监控面板上的指标名称（例如系统CPU使用率），可打开或关闭该指标在图表中的可见性。

> **说明** 每个图表必须至少有一个指标设为可见，这意味着当图表中只有一个指标时，您无法关闭该指标的可见性。

- 单击右上角的折线图按钮，可选择区间查看和对比查看。
- 单击右上角的查看API按钮，可查看该监控指标的API详情。
- 单击监控面板右上角的和图标，可以查看已有报警的报警点和创建新的报警。创建报警的方法，请参见创建报警。

功能介绍

主机监控功能可监控以下指标：

- CPU
  - CPU使用率总和
  - 系统CPU使用率
  - 用户CPU使用率
  - 等待IO完成的CPU使用率
- 物理内存
  - 系统总内存
  - 系统空闲内存
  - 系统已使用内存
  - 系统PageCache中的内存
  - 系统BufferCache中的内存
- Disk（磁盘）
  - 系统磁盘总字节数
  - 系统磁盘空闲字节数
  - 系统磁盘使用字节数
- Load（负载）
  系统负载数
- 网络流量
  - 网络接收的字节数
  - 网络发送的字节数
- 网络数据包
  - 每分钟网络接收的报文数
  - 每分钟网络发送的报文数
  - 每分钟网络接收的错误数
  - 每分钟网络丢弃的报文数

3.5.5. Pod监控

本文说明如何查看Pod监控，从而了解应用的Pod情况，包括CPU、物理内存、网络流量、网络数据包等信息。

前提条件

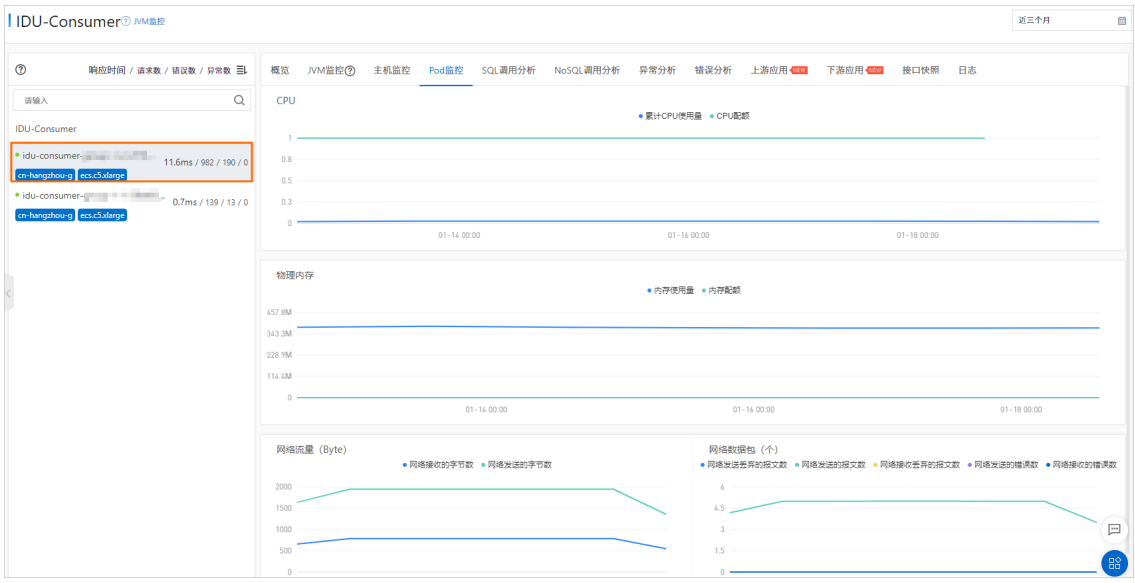
接入应用监控

> **说明** 仅部署在Pod中的应用支持查看Pod监控。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。

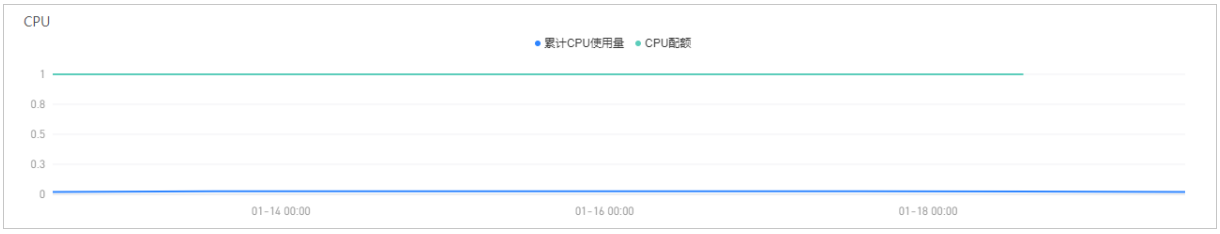
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择Pod实例，设置时间段，单击**Pod监控**页签。



CPU

CPU区域显示该应用在指定时间段的Pod的CPU情况，包括以下指标：

- 累计CPU使用量
- CPU配额

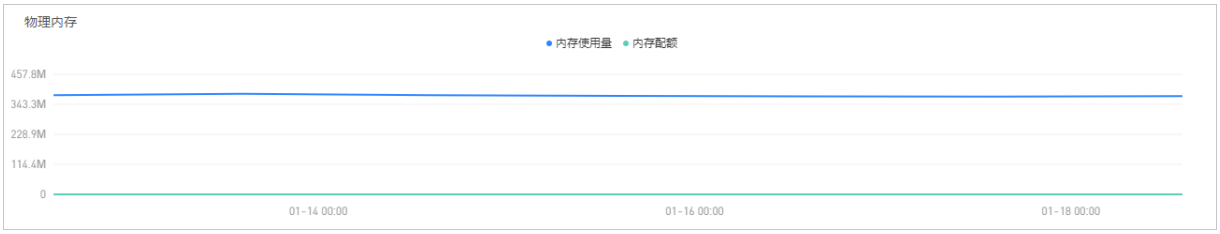


1. （可选）在**CPU**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图例，隐藏或显示数据。

物理内存

物理内存区域显示该应用在指定时间段的Pod的物理内存情况，包括以下指标：

- 内存使用量
- 内存配额

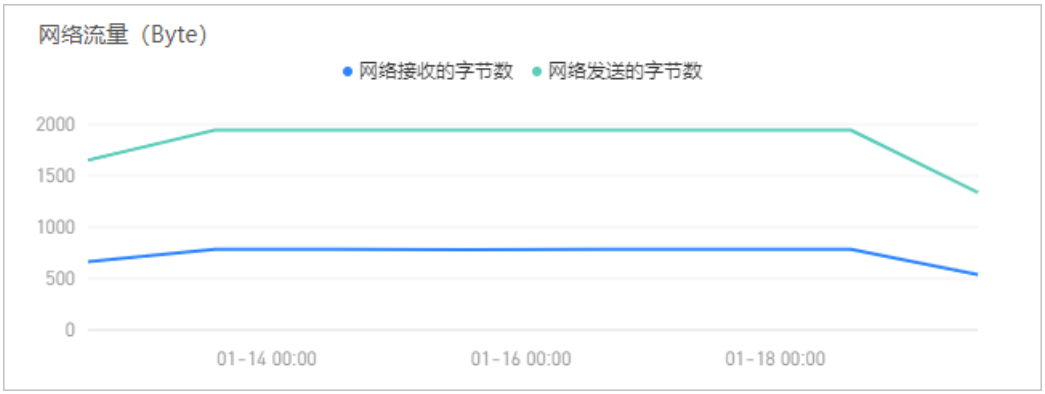


1. （可选）在**物理内存**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图例，隐藏或显示数据。

网络流量

网络流量区域显示该应用在指定时间段的Pod的网络流量情况，包括以下指标：

- 网络接收的字节数
- 网络发送的字节数



1. （可选）在**网络流量**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图例，隐藏或显示数据。

网络数据包

网络数据包区域显示该应用在指定时间段的Pod的网络数据包情况，包括以下指标：

- 网络发送丢弃的报文数
- 网络发送的报文数
- 网络接收丢弃的报文数
- 网络发送的错误数
- 网络接收的错误数

1. （可选）在**网络数据包**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。

- 使用光标选中一段时间，查看指定时间段的统计情况。
- 单击图例，隐藏或显示数据。

3.5.6. SQL调用分析

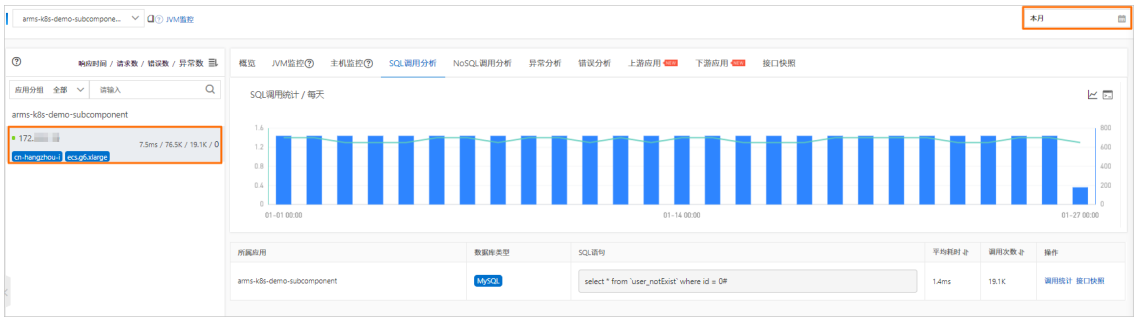
本文说明如何查看SQL调用分析，从而了解应用的SQL调用情况。

前提条件

接入应用监控

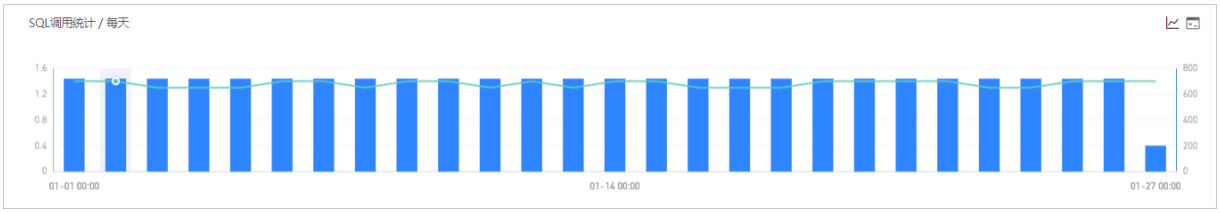
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择应用实例，设置时间段，单击**SQL调用分析**页签。



SQL调用统计

SQL调用统计区域显示该应用在指定时间段的SQL调用时序曲线。



1. （可选）在**SQL调用统计**页签下，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。

SQL语句列表

SQL语句列表显示该应用在指定时间段的所有SQL语句的列表。

1. （可选）在SQL语句列表，您可以执行以下操作：
   - 在SQL语句的**操作**列，单击**调用统计**，查看该SQL语句的SQL调用时序曲线。
   - 在SQL语句的**操作**列，单击**接口快照**，查看该SQL语句调用的接口的快照。
     接口快照说明，请参见接口快照。

3.5.7. NoSQL调用分析

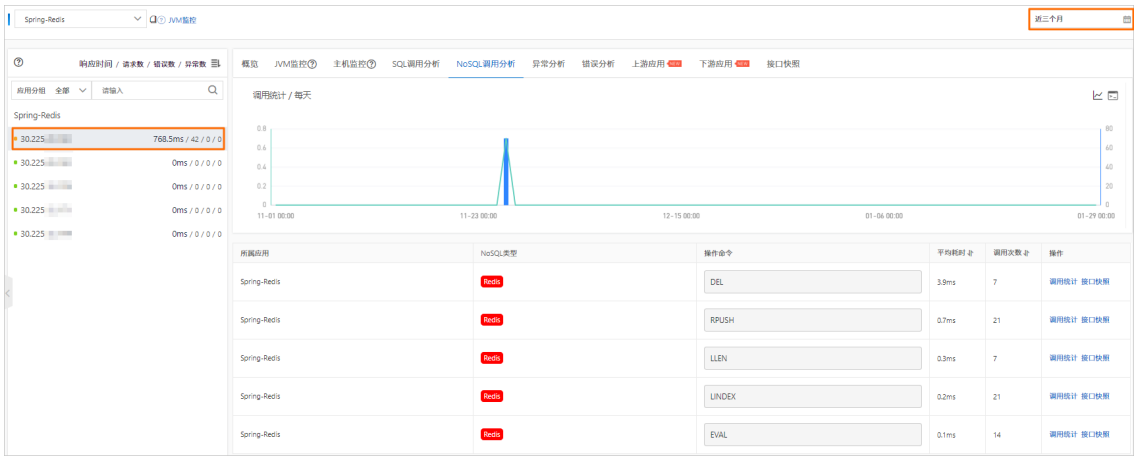
本文说明如何查看NoSQL调用分析，从而了解应用的NoSQL调用情况。

前提条件

接入应用监控

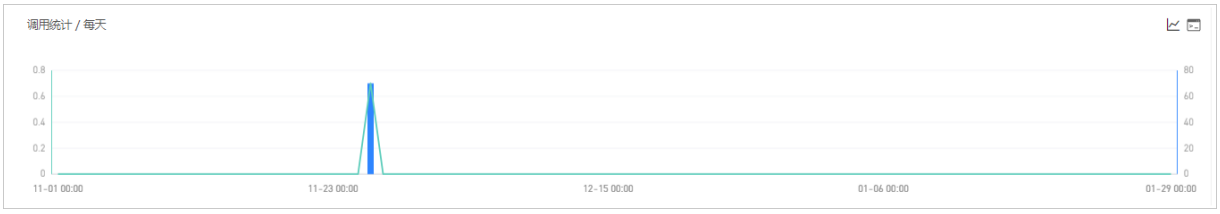
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择应用实例，设置时间段，单击**NoSQL调用分析**页签。



NoSQL调用统计

NoSQL调用统计区域显示该应用在指定时间段的NoSQL调用时序曲线。



1. （可选）在**NoSQL调用统计**页签下，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。

- 单击图标，查看该指标的API详情。

操作命令列表

操作命令列表显示该应用在指定时间段的所有NoSQL调用的操作命令。

所属应用	NoSQL类型	操作命令	平均耗时	调用次数	操作
Spring-Redis	Redis	DEL	3.9ms	7	调用统计 接口快照
Spring-Redis	Redis	RPUSH	0.7ms	21	调用统计 接口快照
Spring-Redis	Redis	LLEN	0.3ms	7	调用统计 接口快照
Spring-Redis	Redis	LINDEX	0.2ms	21	调用统计 接口快照
Spring-Redis	Redis	EVAL	0.1ms	14	调用统计 接口快照

1. （可选）在操作命令列表，您可以执行以下操作：
   - 在操作命令右侧**操作**列，单击**调用统计**，查看该操作命令的NoSQL调用统计。
   - 在操作命令右侧**操作**列，单击**接口快照**，查看该操作命令的接口的快照。
     接口快照说明，请参见接口快照。

3.5.8. 异常分析

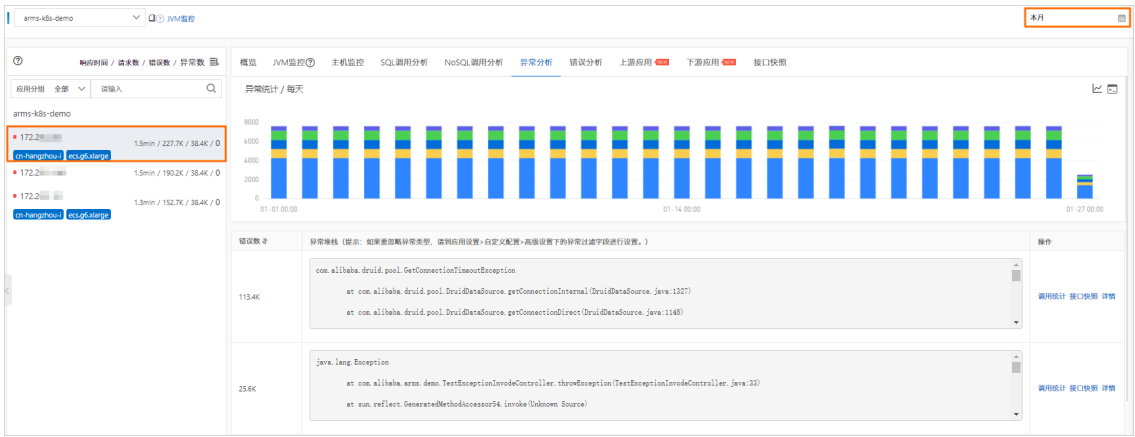
本文说明如何查看异常分析，从而了解应用的异常情况。

前提条件

接入应用监控

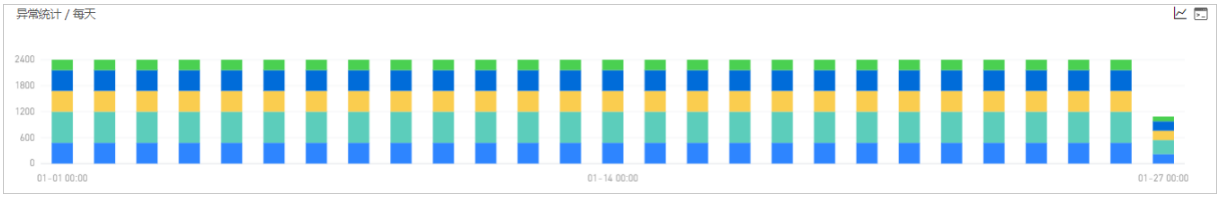
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择应用实例，设置时间段，单击**异常分析**页签。



异常统计

异常统计区域显示该应用在指定时间段的异常的堆积柱状图和异常列表。



1. （可选）在**异常统计**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。

异常列表

异常列表显示该应用在指定时间段的所有异常。

1. （可选）在异常列表，您可以执行以下操作：

> **说明** 如果要过滤异常，在**应用设置 > 自定义设置 > 高级义设置**的**异常过滤**文本框设置。

   - 在异常的**操作**列，单击**调用统计**查看该异常的堆积柱状图。
   - 在异常的**操作**列，单击**接口快照**查看该异常的接口的快照。
     接口快照说明，请参见接口快照。
   - 在异常的**操作**列，单击**详情**查看该异常的详细信息。

3.5.9. 错误分析

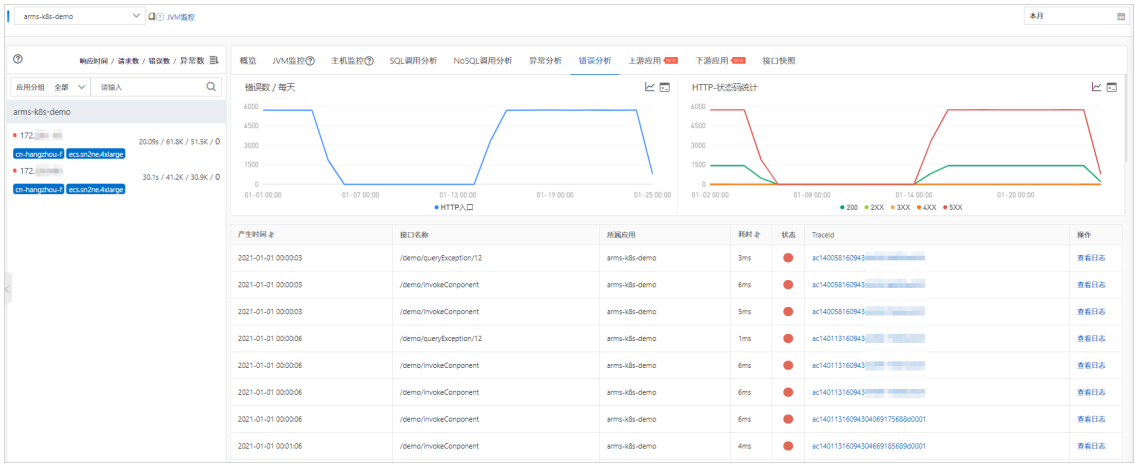
本文说明如何查看错误分析，从而了解应用的错误情况。

前提条件

接入应用监控

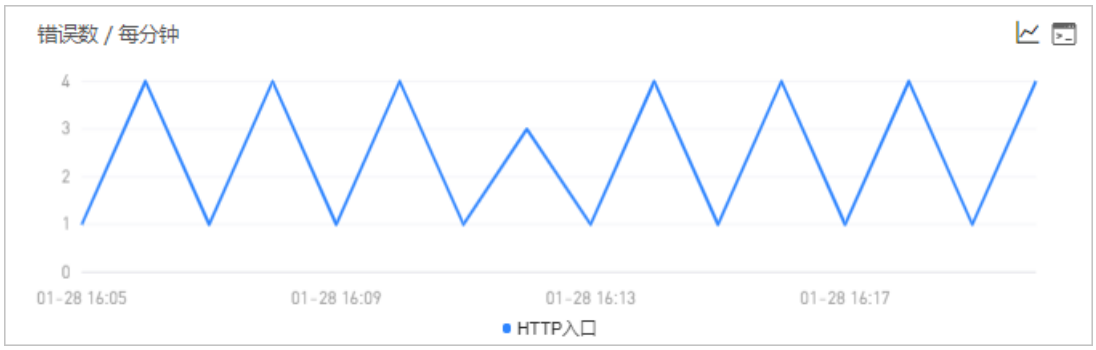
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择应用实例，设置时间段，单击**错误分析**页签。



错误数

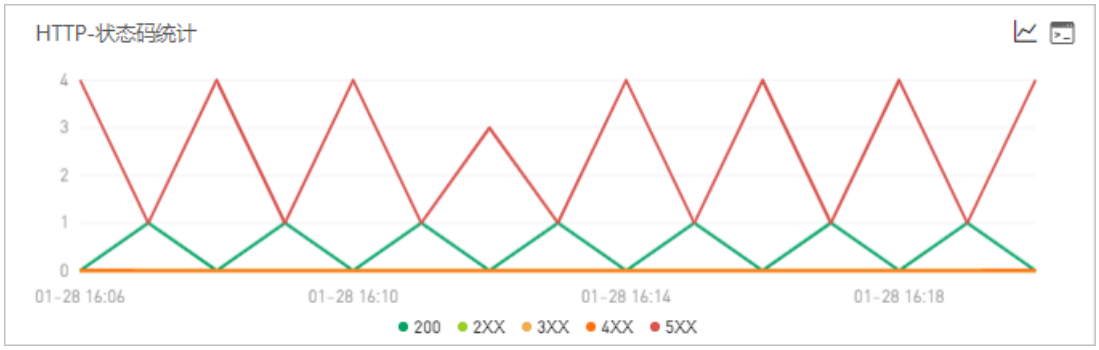
错误数区域显示该应用在指定时间段的错误时序曲线。



1. （可选）在**错误数**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。

HTTP状态码

HTTP-状态码统计区域显示该应用在指定时间段的HTTP状态码时序曲线。



1. （可选）在**HTTP-状态码统计**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。

- 单击图例，隐藏或显示数据。
- 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
- 单击图标，查看该指标的API详情。

错误列表

错误列表显示该应用在指定时间段的所有错误。

产生时间	接口名称	所属应用	耗时	状态	TraceId	操作
2021-01-28 16:06:06	/demo/queryException/12	arms-k8s-demo	1ms		ac1401131611821	查看日志
2021-01-28 16:06:06	/demo/invokeConponent	arms-k8s-demo	4ms		ac1401131611821	查看日志
2021-01-28 16:06:06	/demo/invokeConponent	arms-k8s-demo	4ms		ac1401131611821	查看日志
2021-01-28 16:06:06	/demo/invokeConponent	arms-k8s-demo	4ms		ac1401131611821	查看日志
2021-01-28 16:07:06	/demo/invokeConponent	arms-k8s-demo	5ms		ac1401131611821	查看日志
2021-01-28 16:07:35	/demo/queryNotExistDB/11	arms-k8s-demo	2min		ac1401131611821	查看日志
2021-01-28 16:08:06	/demo/queryException/12	arms-k8s-demo	1ms		ac1401131611821	查看日志
2021-01-28 16:08:06	/demo/invokeConponent	arms-k8s-demo	4ms		ac1401131611821	查看日志
2021-01-28 16:08:06	/demo/invokeConponent	arms-k8s-demo	4ms		ac1401131611821	查看日志

1. （可选）在错误列表，您可以执行以下操作：
   - 在错误右侧的**TraceId**列，单击TraceId名称查看该错误的调用链。
   - 在错误右侧的**操作**列，单击**查看日志**查看该错误的日志。

3.5.10. 上游应用

对于某个应用而言，上游应用是指向该应用发送数据的应用。本文说明如何查看上游应用情况，包括响应时间、请求数、错误数等信息。

前提条件

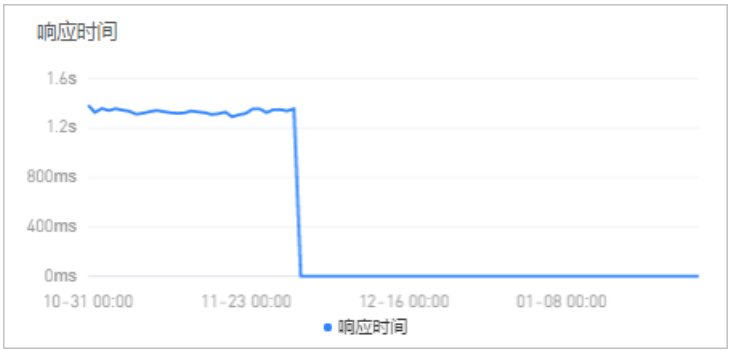
接入应用监控

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择应用实例，设置时间段，单击**上游应用**页签。

响应时间

响应时间区域显示该应用的上游应用在指定时间段的响应时间时序曲线。



1. 在**响应时间**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。

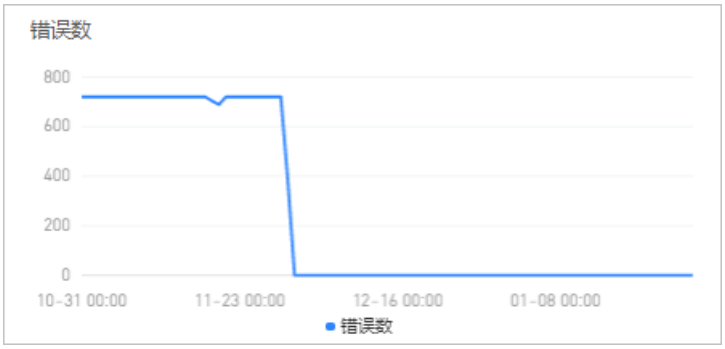
请求数

请求数区域显示该应用的上游应用在指定时间段的请求数时序曲线。

1. 在**请求数**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。

错误数

错误数区域显示该应用的上游应用在指定时间段的错误数时序曲线。



1. 在**错误数**区域，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。

3.5.11. 调用链查看

本文说明如何查看调用链，从而了解应用的所有接口的被调用情况，包括产生时间、耗时、状态等信息。

前提条件

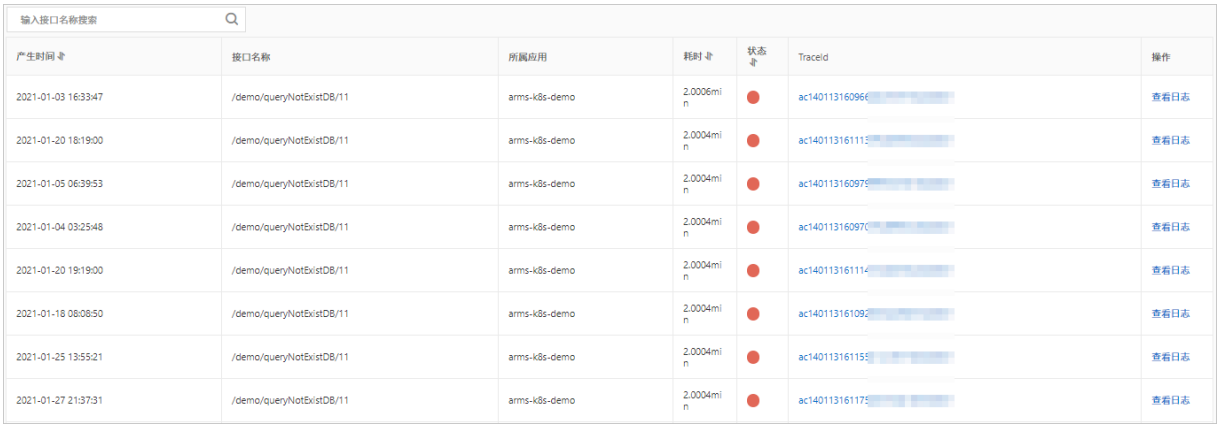
接入应用监控

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择应用实例，设置时间段，单击**调用链查询**页签。

接口快照

接口快照页签下显示该应用在指定时间段内被调用的所有接口的列表。



产生时间	接口名称	所属应用	耗时	状态	TraceId	操作
2021-01-03 16:33:47	/demo/queryNotExistDB/11	arms-k8s-demo	2.0006min		ac140113160966	查看日志
2021-01-20 18:19:00	/demo/queryNotExistDB/11	arms-k8s-demo	2.0004min		ac140113161113	查看日志
2021-01-05 06:39:53	/demo/queryNotExistDB/11	arms-k8s-demo	2.0004min		ac140113160975	查看日志
2021-01-04 03:25:48	/demo/queryNotExistDB/11	arms-k8s-demo	2.0004min		ac140113160970	查看日志
2021-01-20 19:19:00	/demo/queryNotExistDB/11	arms-k8s-demo	2.0004min		ac140113161114	查看日志
2021-01-18 08:08:50	/demo/queryNotExistDB/11	arms-k8s-demo	2.0004min		ac140113161092	查看日志
2021-01-25 13:55:21	/demo/queryNotExistDB/11	arms-k8s-demo	2.0004min		ac140113161155	查看日志
2021-01-27 21:37:31	/demo/queryNotExistDB/11	arms-k8s-demo	2.0004min		ac140113161175	查看日志

1. （可选）在**接口快照**页签下，您可以执行以下操作：
   - 在搜索框，输入接口名称，然后单击图标，查看指定接口的快照。
   - 在接口的**TraceId**列，单击TraceId名称，查看该接口的调用链路。
   - 在接口的**操作**列，单击**查看日志**，查看该接口的日志。

3.5.12. 日志

本文说明如何查看应用的Pod日志，从而了解Pod情况。

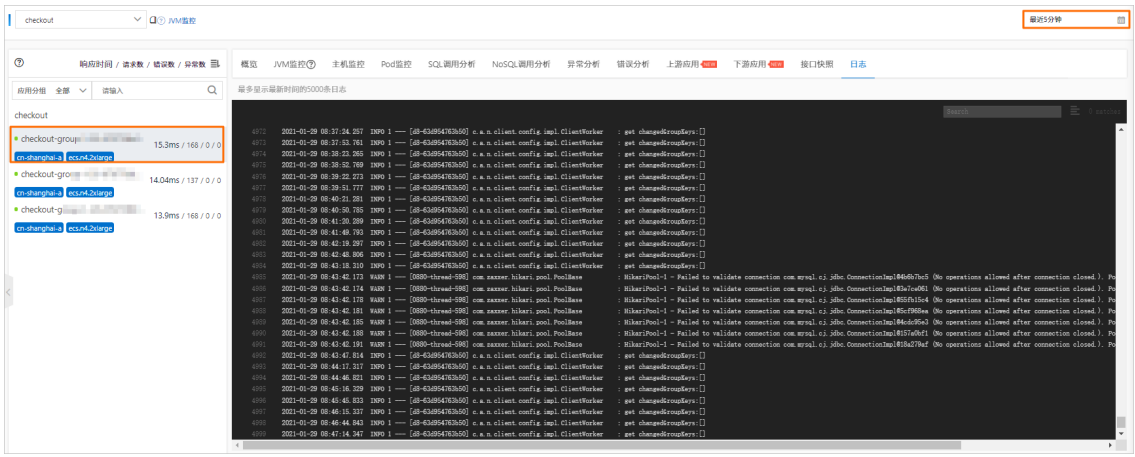
前提条件

接入应用监控

> **说明** 仅部署在Pod中的应用支持查看Pod日志。

功能入口

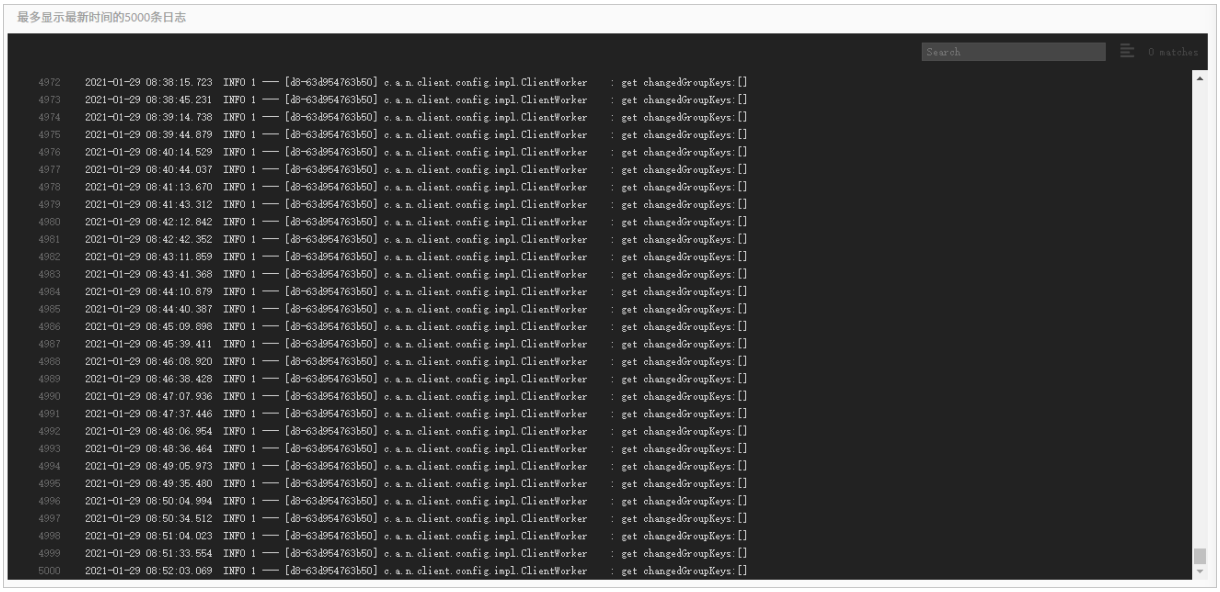
1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用详情**。
6. 在**应用详情**页面，选择Pod实例，设置时间段，单击**日志**页签。



日志

日志页签下显示该应用的Pod实例的最新时间的日志。

> **说明** 最多显示最新时间的5000条日志。



1. 在**日志**页签下的搜索框，您可以输入关键字，单击图标，筛选日志。

3.5.13. 内存快照

JVM监控可以直观展示指定时间段内的多项内存指标，虽然图表能体现出内存使用量过大的情况，但无法显示具体信息，因此不能帮助您排查问题产生的原因。此时您可以创建内存快照，通过详细的日志查看内存占用的详细信息，帮助您排查内存泄漏和内存浪费等内存问题。

创建内存快照

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用名称。
4. 在左侧导航栏单击**应用详情**，在页面右侧单击**JVM监控**页签。

5. 在**JVM监控**页签右上角，单击**创建内存快照**。

> **说明** 如果单击**创建内存快照**时，上一个快照任务仍在运行，则系统会弹出错误消息。请您耐心等待上一个快照任务运行完毕。目前仅支持为Linux系统新建内存快照。

6. 在**创建内存快照**对话框中选择一个IP，并单击**保存**。

> **警告** 快照任务的运行时间从几分钟到半小时不等。应用进程在转储期间会停止响应，请谨慎使用。

> **说明** 如果在**应用详情**页面左侧已选择目标实例，则IP字段会默认选中该实例的IP地址。

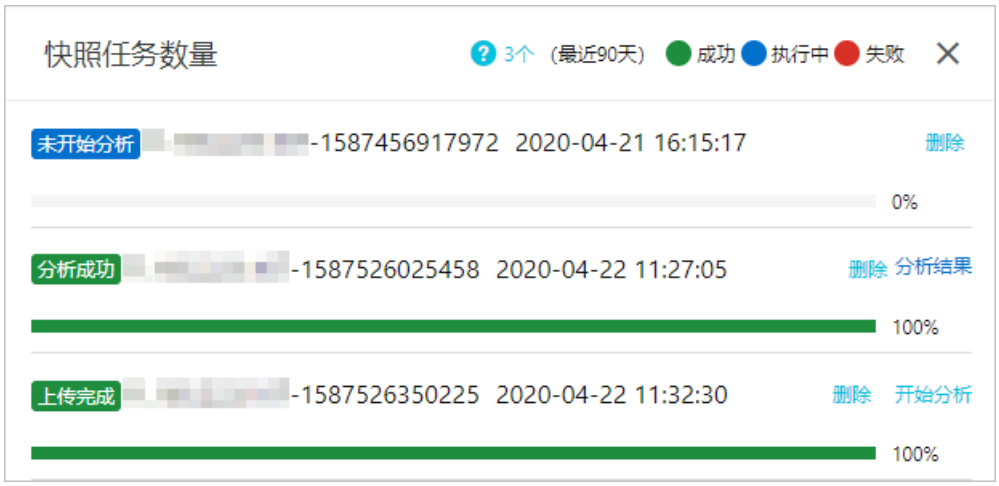
查看内存快照详情

1. 在**JVM监控**页签右上角，单击**历史快照**。

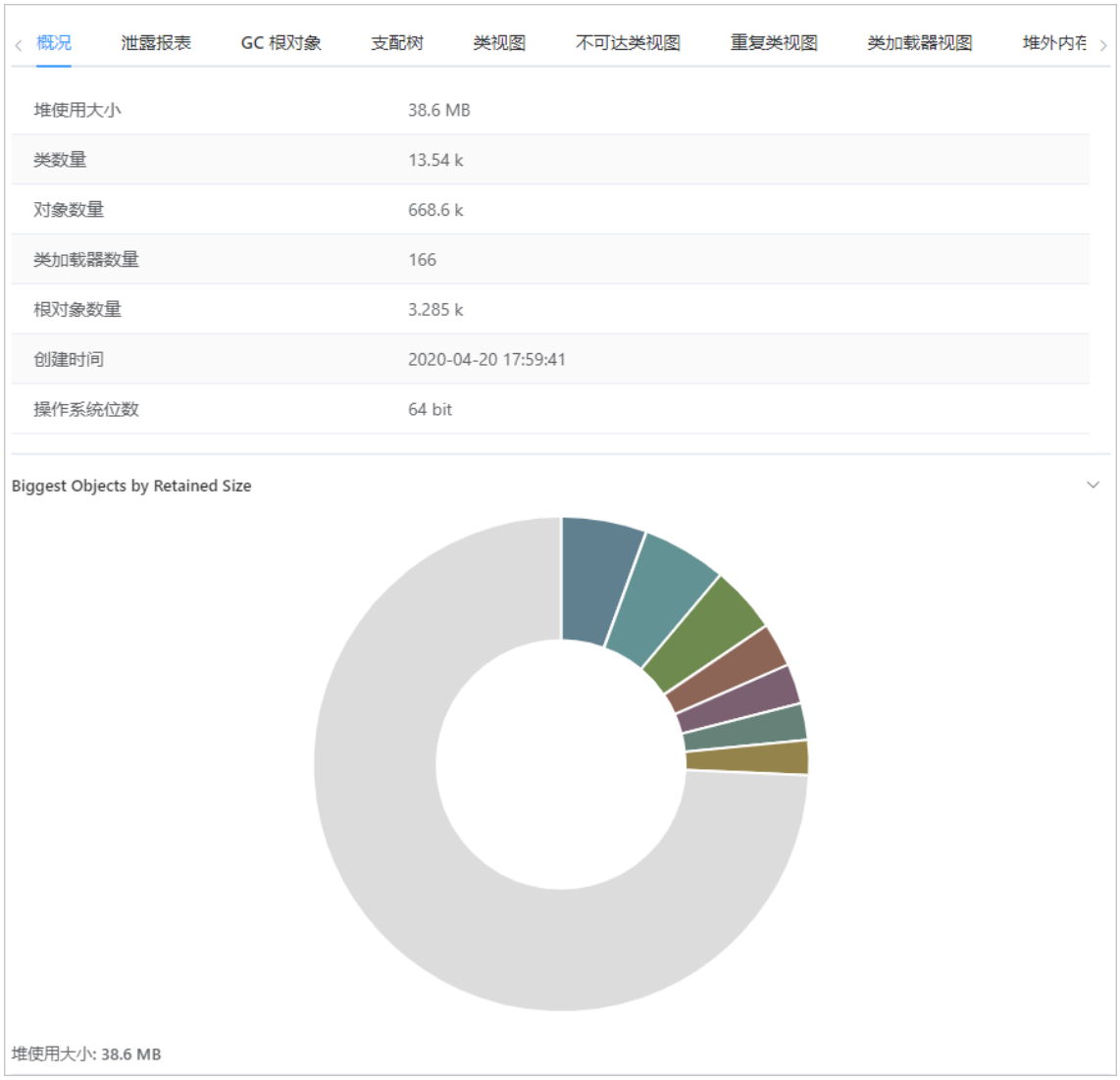
   在**快照任务数量**面板展示了任务执行状态：绿色表示快照任务执行成功，蓝色表示快照任务执行中，红色表示快照任务执行失败。
   快照任务名称的信息依次为：

   - 内存分析状态
   - 由IP地址和时间戳组成的快照任务ID

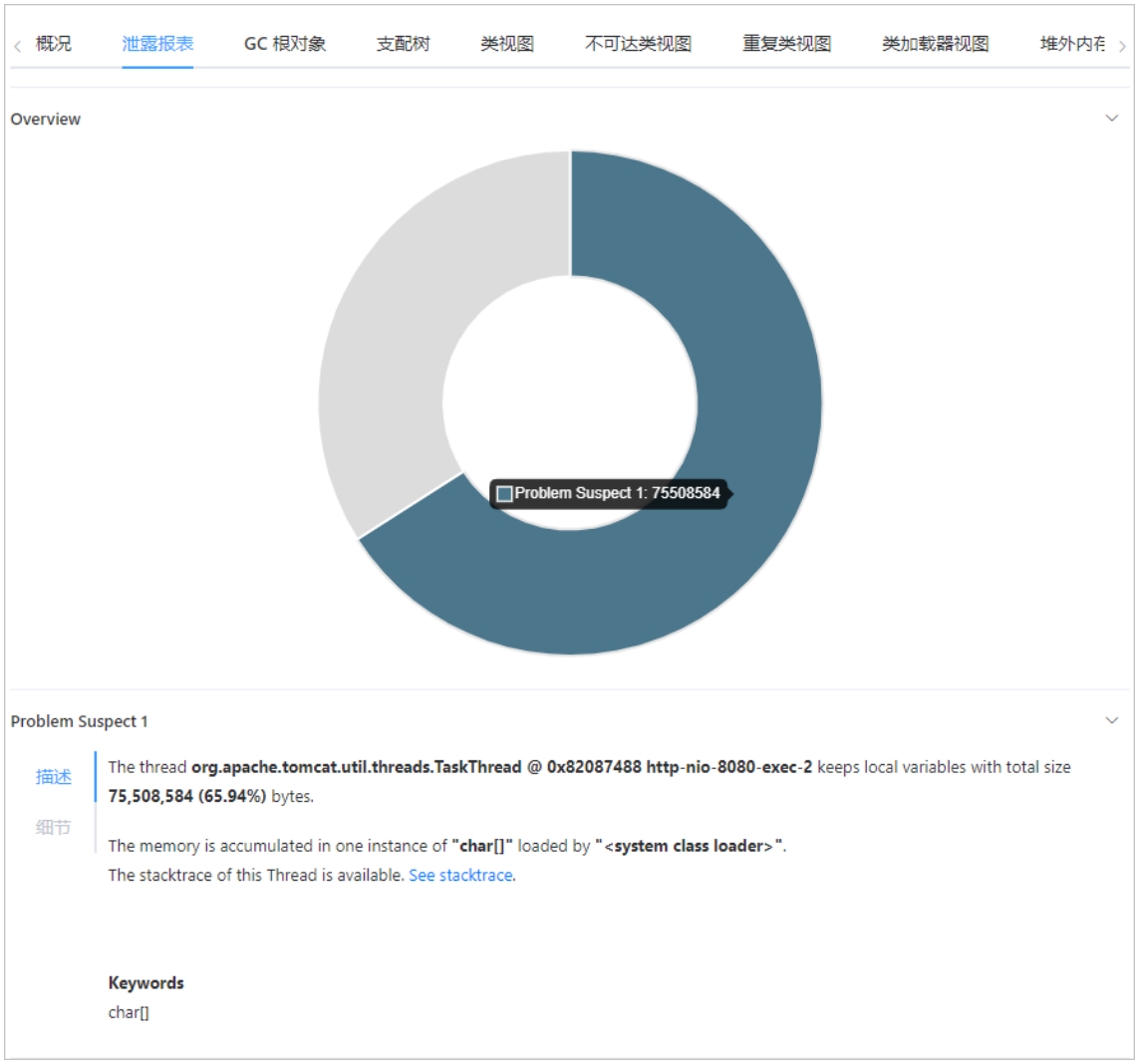
- 快照创建时间



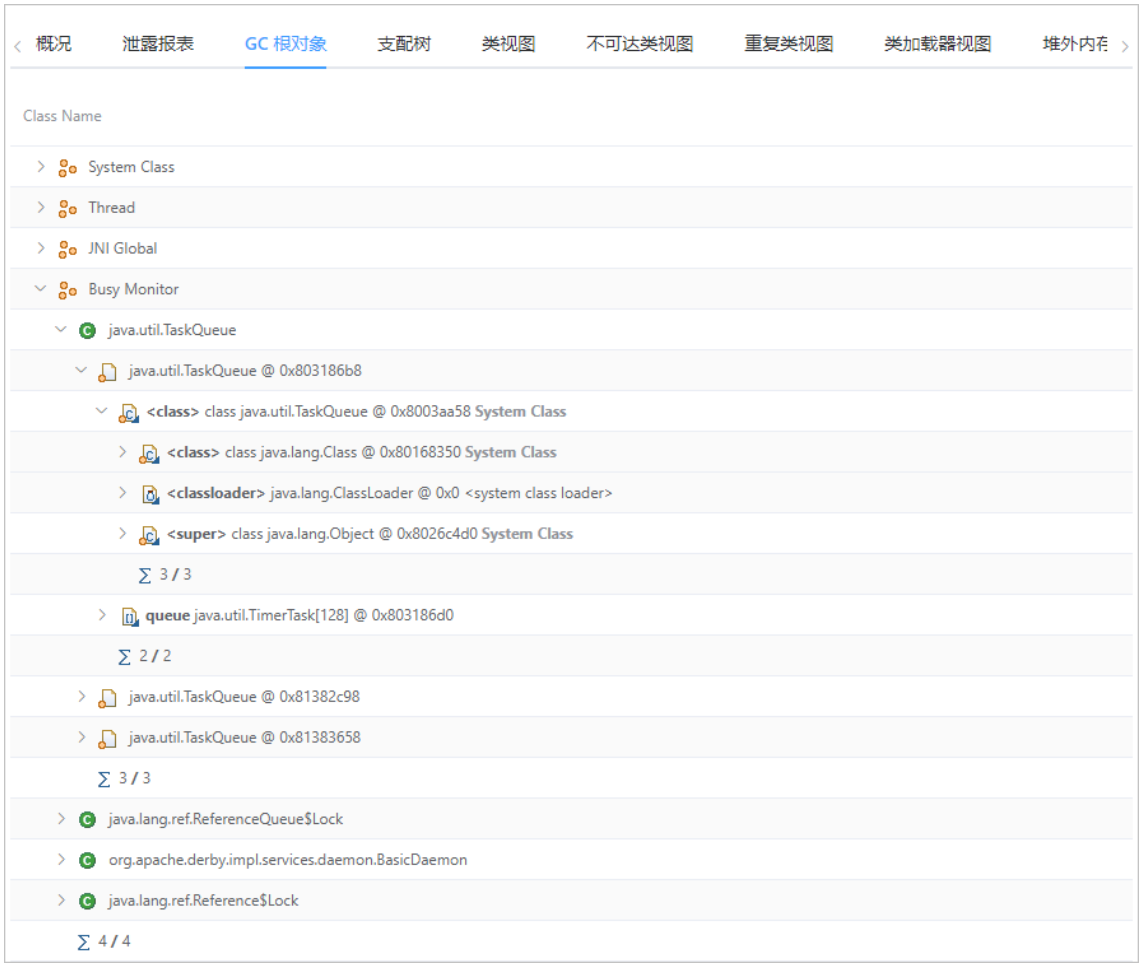
2. 单击**开始分析**，弹出**提示**对话框，单击**确认**。
3. 单击**分析结果**，打开内存分析页面查看内存分析详情，帮助您查找内存泄漏信息，用于减少内存消耗。
   - 单击**概况**页签，查看堆使用大小、类数量、对象数量、类加载器数量和根对象数量等统计信息，以及以环形图显示的内存占用情况。



- 单击**泄露报表**页签，可以查看消耗过多内存的可疑对象。单击页面下方的**Problem Suspect**查看可疑对象的具体实例、内存占用的情况以及类加载信息。



- 单击**GC根对象**页签，查看按根类型和Java类类型分类的所有根对象（被静态变量或线程栈等GC根引用的对象）。



- 单击**支配树**页签，查看堆中对象之间的支配关系，可以识别出占用了大量内存的对象和它们之间的依赖关系。

- 单击**类视图**页签，查看各个类型的堆使用情况和对应实例的数量。

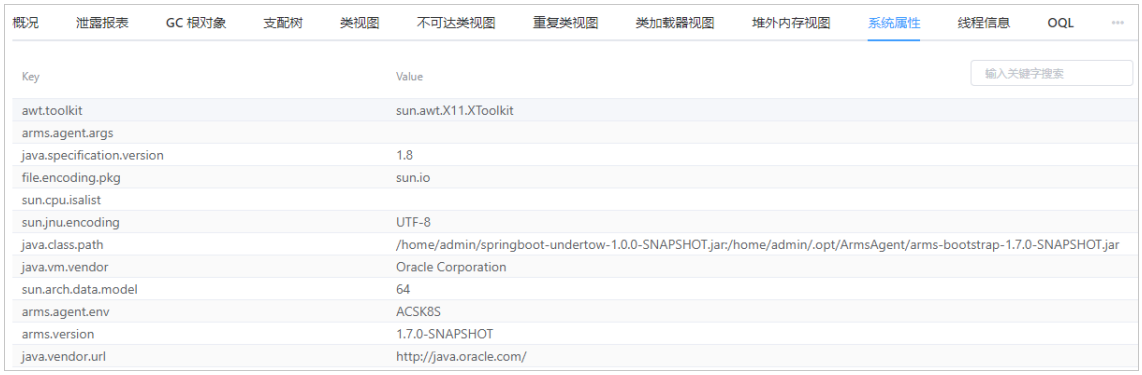
- 单击**不可达类视图**页签，查看堆中未被引用的对象的大小及类型情况。

- 单击**重复类视图**页签，查看被多个类加载器加载的类型情况。

- 单击**类加载器视图**页签，查看应用当前使用到的所有类加载，以及每个类加载器加载类的情况。例如加载了哪些类，这些类有多少实例等。

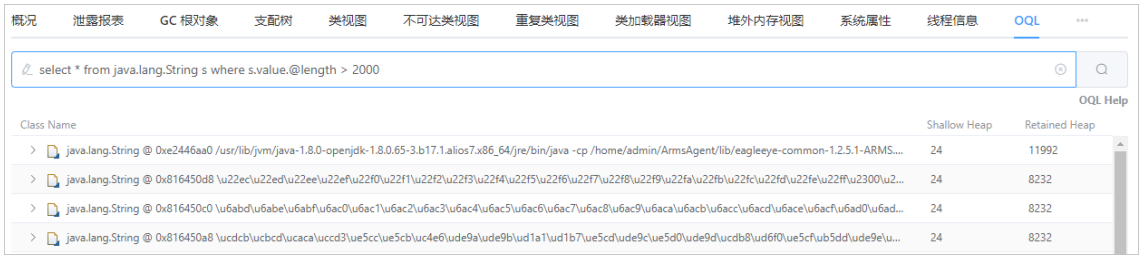
- 单击**堆外内存视图**页签，查看应用目前使用的所有java.nio.DirectByteBuffer对象以及对应的堆外内存信息，用于排查由堆外内存导致物理内存消耗过多的问题。

- 单击**系统属性**页签，查看系统参数和环境变量。



- 单击**线程信息**页签，查看线程的相关信息，例如线程名、堆占用情况、调用栈信息、局部变量等。通过该视图您可以分析线程过多、死锁、调用栈过深等问题。

- 单击**OQL**页签，查询堆中的各种信息，例如所有长度大于2000的字符串。



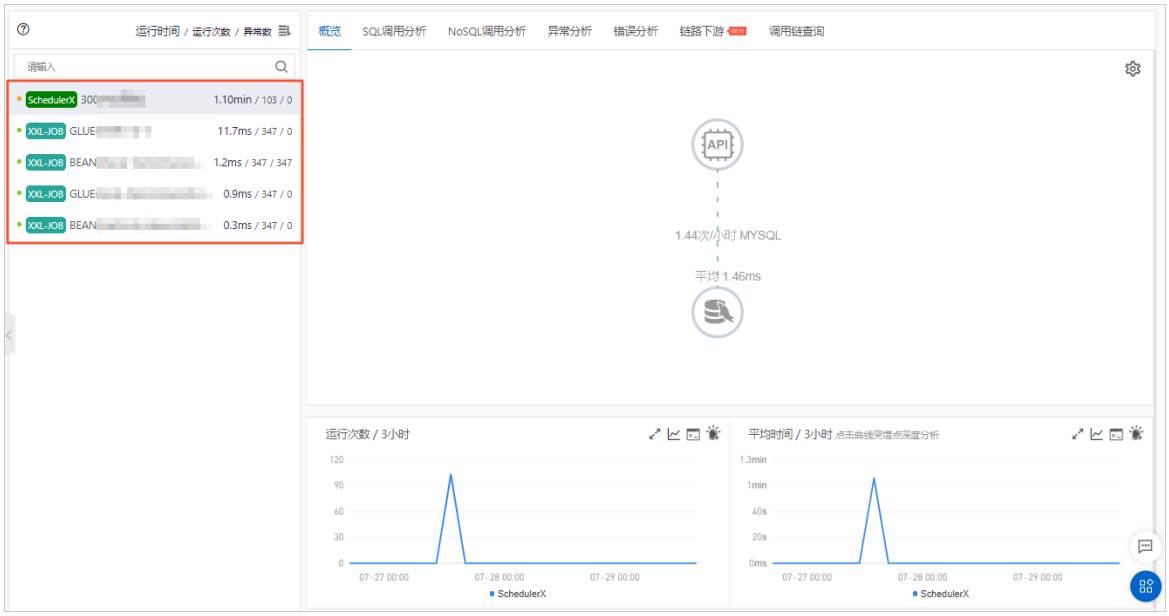
3.6. 定时任务

定时任务可以在固定的间隔时间执行指定的程序或者指令，应用监控的定时任务功能可以帮助您了解定时任务的详细情况，包括概览、SQL调用分析、NoSQL调用分析、异常分析、错误分析、链路下游和调用链查询。

> **说明** ARMS应用监控目前仅支持展示XXL-JOB和SchedulerX类型的定时任务。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**定时任务**。
   在定时任务页面左侧列表显示了当前应用下的所有定时任务。



6. 单击目标定时任务，可以查看对应的定时任务详情。

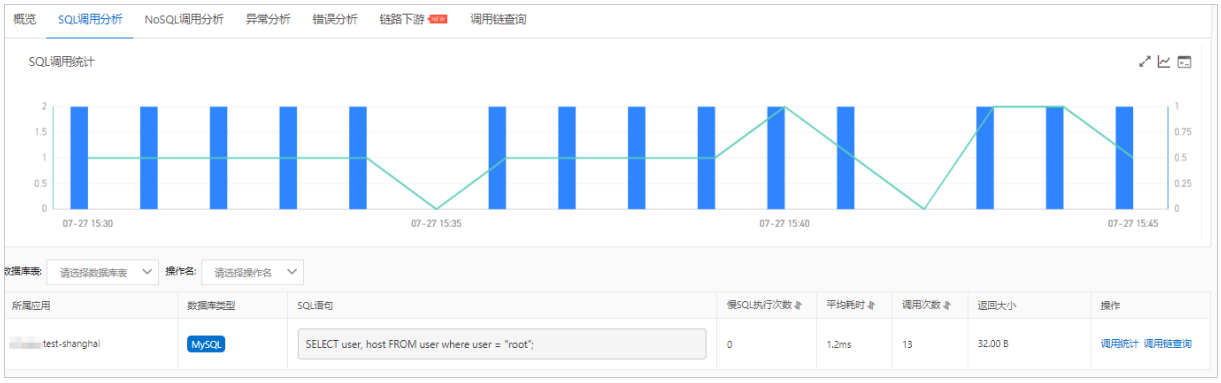
> **注意** 定时任务功能要求Agent版本升级至2.7.1.3及以上。升级Agent的具体操作，请参见升级探针。

查看概览信息

概览页签可以查看目标定时任务的详细调用拓扑，以及运行次数、平均时间、运行异常次数和调度延迟时间的时序曲线。

查看SQL和NoSQL调用分析

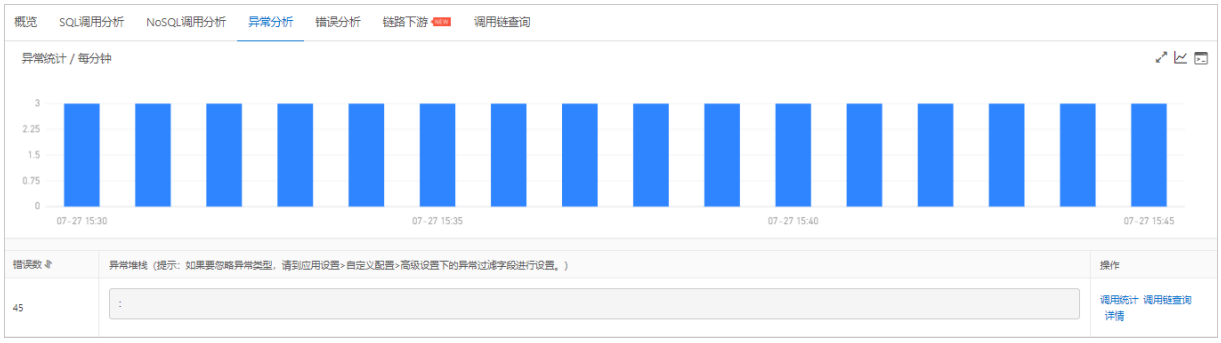
SQL调用分析页签和**NoSQL调用分析**页签展示了左侧选中的定时任务所发起的SQL和NoSQL请求列表。借助此页签，您可以找出是哪一个SQL或NoSQL造成某个服务过慢。



1. （可选）在SQL或NoSQL语句列表，您可以执行以下操作：
   - 单击**操作**列**调用统计**，查看该SQL或NoSQL语句的调用时序曲线。
   - 单击**操作**列**调用链查询**，查看该SQL或NoSQL语句的调用链。更多信息，请参见调用链查询。

查看异常分析

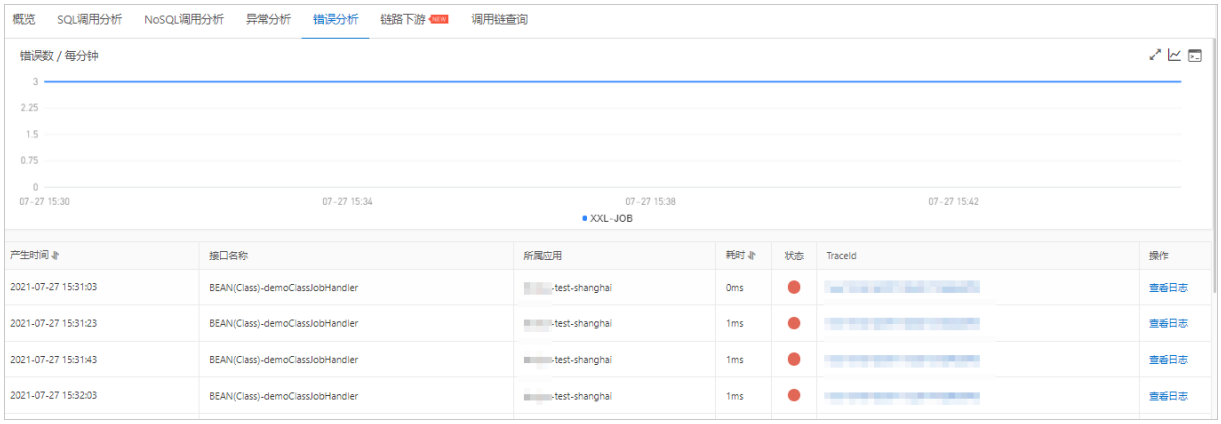
异常分析页签展示了左侧选中定时任务的代码段内所造成的Java异常。



1. （可选）在异常列表，您可以执行以下操作：
   - 单击**操作**列**调用统计**，查看该异常的时序曲线。
   - 单击**操作**列**调用链查询**，查看该异常的调用链。更多信息，请参见调用链查询。

查看错误分析

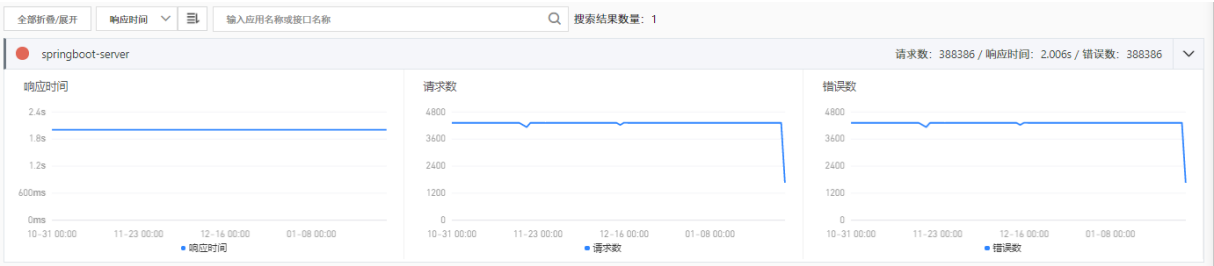
错误分析页签展示了目标定时任务产生错误的统计数据。



1. （可选）在错误列表，您可以执行以下操作：
   - 在错误右侧的**TraceId**列，单击TraceId名称查看该错误的调用链。更多信息，请参见调用链路查询。
   - 在错误右侧的**操作**列，单击**查看日志**查看该错误的日志。

查看链路下游的调用情况

链路下游页签列出了应用下游（被应用调用的一方）的接口及其调用性能指标，包括响应时间、请求数和错误数。

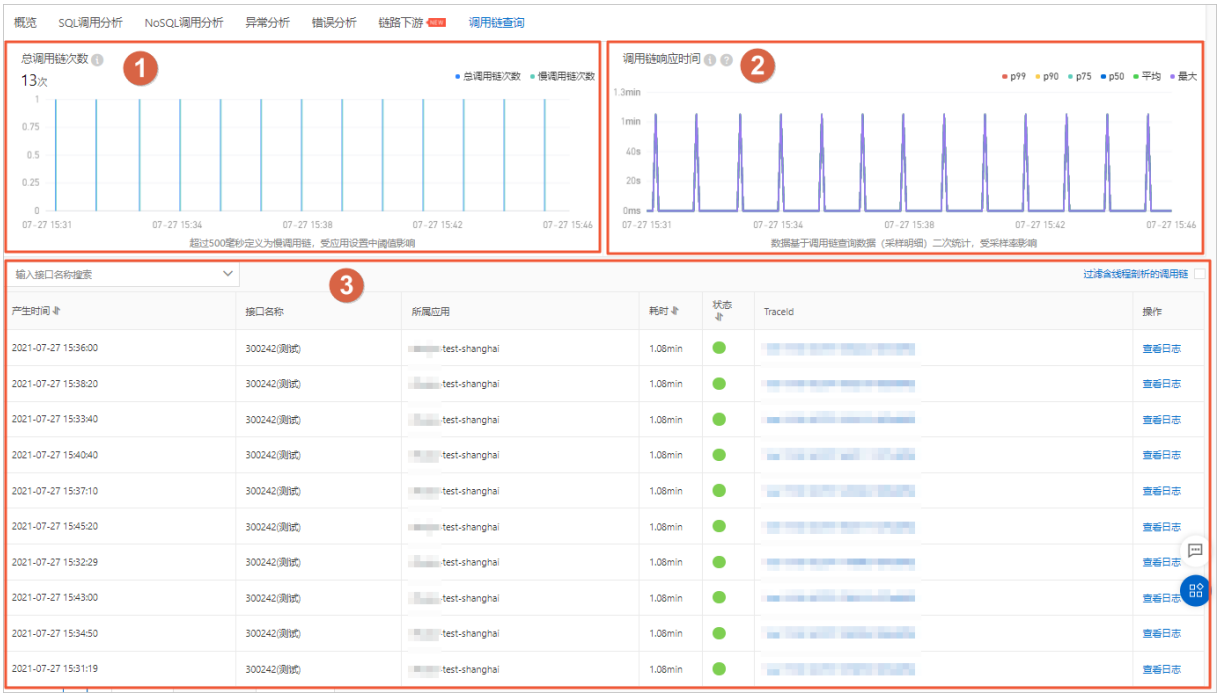


在**链路下游**页签上，可按需执行以下操作：

- 在页签顶部单击**全部折叠/展开**，即可折叠或展开下方的所有接口。
- 在页签顶部的搜索框内输入应用名称或接口（Span）名称的关键字，并单击搜索图标，即可筛选出符合条件的接口。
- 单击接口信息所在的折叠面板，或者单击行末的上箭头或下箭头，即可展开或折叠该接口的性能指标信息。

调用链查询

调用链查询页签展示了目标定时任务的调用链详情，以及根据该任务生成的总调用链次数和调用链响应时间两个图表。



- 在总调用链次数图表中（图示中①），超过500毫秒被定义为慢调用链。慢调用链受应用设置中阈值影响，但不包含因为异步产生的调用本地API的子调用链。
- 在调用链响应时间图表中（图示中②），其展示的数据是基于接口调用链数据的二次统计，并受采样率影响。
- 在调用链详情表格中（图示中③），单击TraceId可以查看调用链路和业务轨迹。单击**查看日志**可以查看定时任务的调用日志。

3.7. 接口调用

本功能用于监控应用下的接口调用详情，包括SQL调用分析、NoSQL调用分析、异常分析、错误分析、链路上下游和调用链查询。

功能入口

按照以下步骤进入应用接口调用监控功能。

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面选择您想查看的应用。
4. 在左侧导航栏中单击**接口调用**。

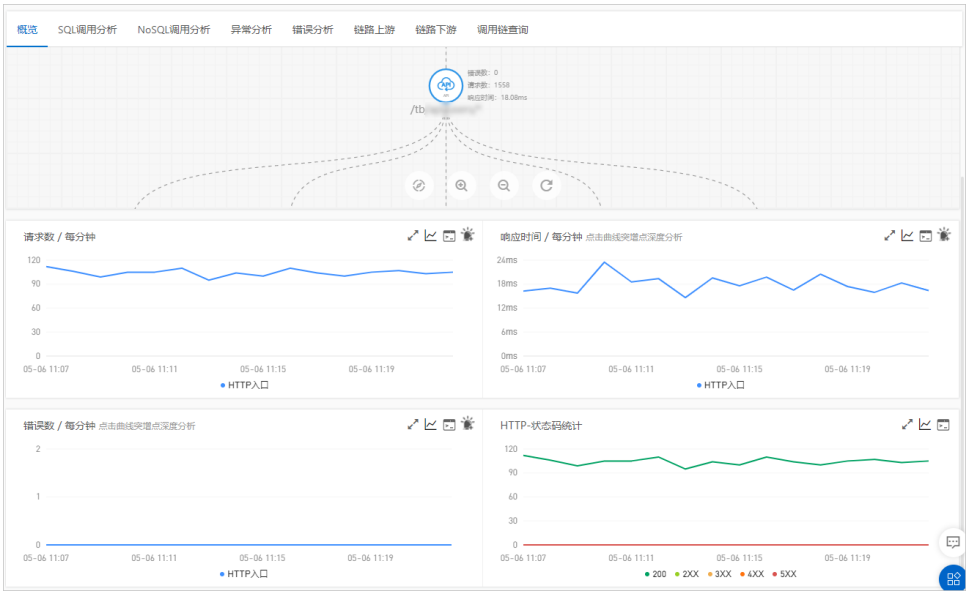
支持的框架

本功能模块可自动发现和监控以下Web框架和RPC框架中提供的接口：

- Tomcat 7+
- Jetty 8+
- Resin 3.0+
- Undertow 1.3+
- WebLogic 11.0+
- SpringBoot 1.3.0+
- HSF 2.0+
- Dubbo 2.5+

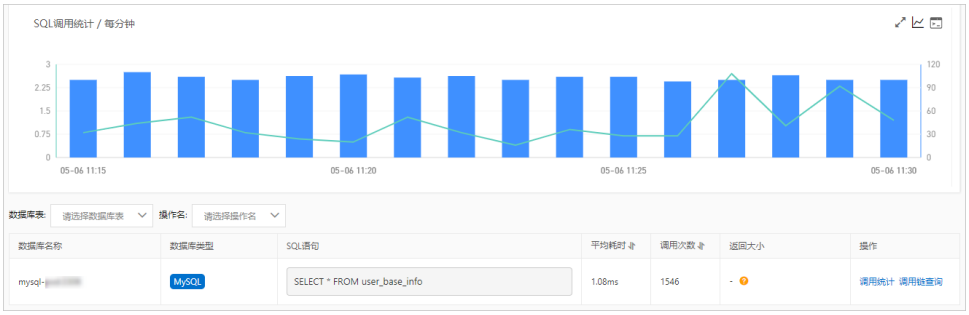
查看接口概览

在**概览**页签上可以查看目标接口的详细调用拓扑，以及请求数、响应时间、错误数和HTTP-状态码统计的时序曲线。



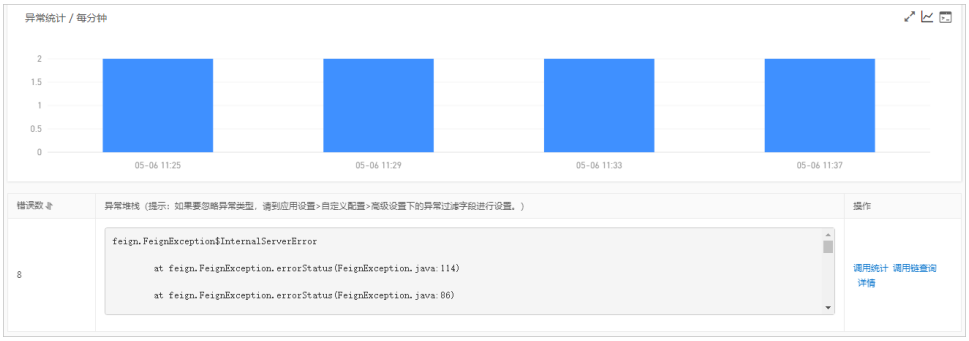
查看SQL和NoSQL调用分析

在**SQL调用分析**页签和**NoSQL调用分析**展示了左侧选中服务的代码段内所发起的SQL和NoSQL请求列表。借助此页签，您可以找出是哪一个SQL或NoSQL造成某个服务过慢。您还可以单击某个SQL或NoSQL中的**调用链查询**来查看一个SQL或NoSQL执行逻辑所处的完整代码链路。



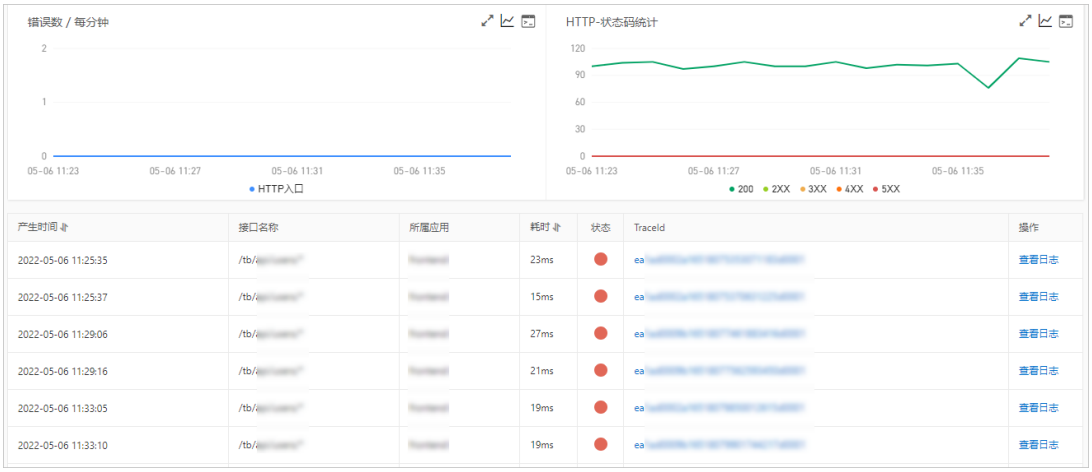
查看异常分析

在**异常分析**页签展示了左侧选中服务的代码段内所抛出的Java异常。您还可以单击某个异常中的**调用链查询**来查看一个异常堆栈所处的完整代码链路。



查看错误分析

在**错误分析**页签展示了应用的错误和HTTP状态码统计数据。您还可以单击需要查看的TraceId，即可在新页面查看调用链路相关信息。



查看链路上游和链路下游的接口调用情况

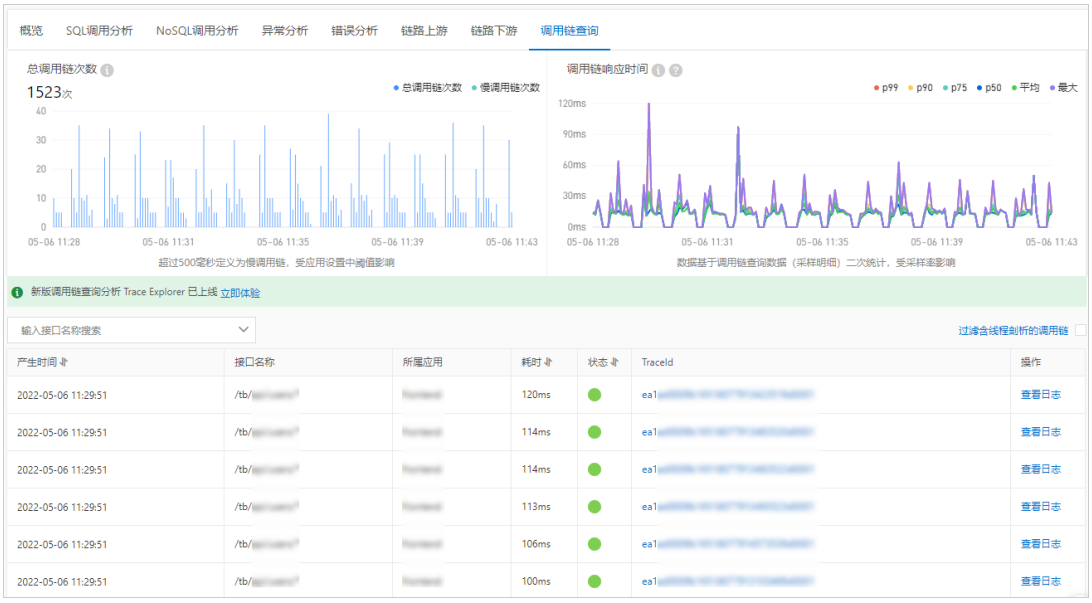
链路上游和**链路下游**页签分别列出了应用上游（调用应用的一方）和应用下游（被应用调用的一方）的接口及其调用性能指标，包括响应时间、请求数和错误数。

在**链路上游**和**链路下游**页签上，可按需执行以下操作：

- 在页签顶部单击**全部折叠/展开**，即可折叠或展开下方的所有接口。
- 在页签顶部的搜索框内输入应用名称或接口（Span）名称的关键字，并单击搜索图标，即可筛选出符合条件的接口。
- 单击接口信息所在的折叠面板，或者单击行末的上箭头或下箭头，即可展开或折叠该接口的性能指标信息。

查看调用链查询

调用链查询页签展示了该服务接口中的参数详情，以及根据该参数详情生成的调用链次数和调用链响应时间两个图表。



- 总调用链次数图表显示了总调用链次数和慢调用链次数。调用时间超过500毫秒则被定义为慢调用链，受应用设置中阈值影响，不含因为异步产生的子调用链（调用本地API）。
- 调用链响应时间图表数据基于调用链查询数据（采样明细）二次统计，受采样率影响。
- 在调用链列表中，单击TraceId可以查看调用链路和业务轨迹。单击**查看日志**可以查看该接口的调用日志。

3.8. 事件中心

事件中心将云产品所生成的事件数据进行统一管理、存储、分析和展示，已接入EDAS的变更事件、ARMS的报警事件、0-1事件（如死锁、OOM和应用启动等）、MSE的微服务管控事件和K8s集群事件。当您的应用使用了相关的产品，对应的事件会自动接入事件中心进行统一的分析展示，方便您查看与分析。

事件模型

事件中心的一个事件主要由以下参数来定义：

参数	是否必须	描述
source	是	事件来源
type	是	事件类型
level	是	事件等级
time	是	事件发生时间
data	是	事件体（一般为JSON格式）
PID	否	Pod ID
IP	否	IP地址
ClusterId	否	集群ID
PodName	否	Pod名称

进入事件中心

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 事件中心**，并在顶部菜单栏选择目标地域。

事件中心模块介绍

事件中心主要包含典型事件和四个页签（分别是**普通视图**，**拓扑视图**、**集群视图**和**订阅规则**）。

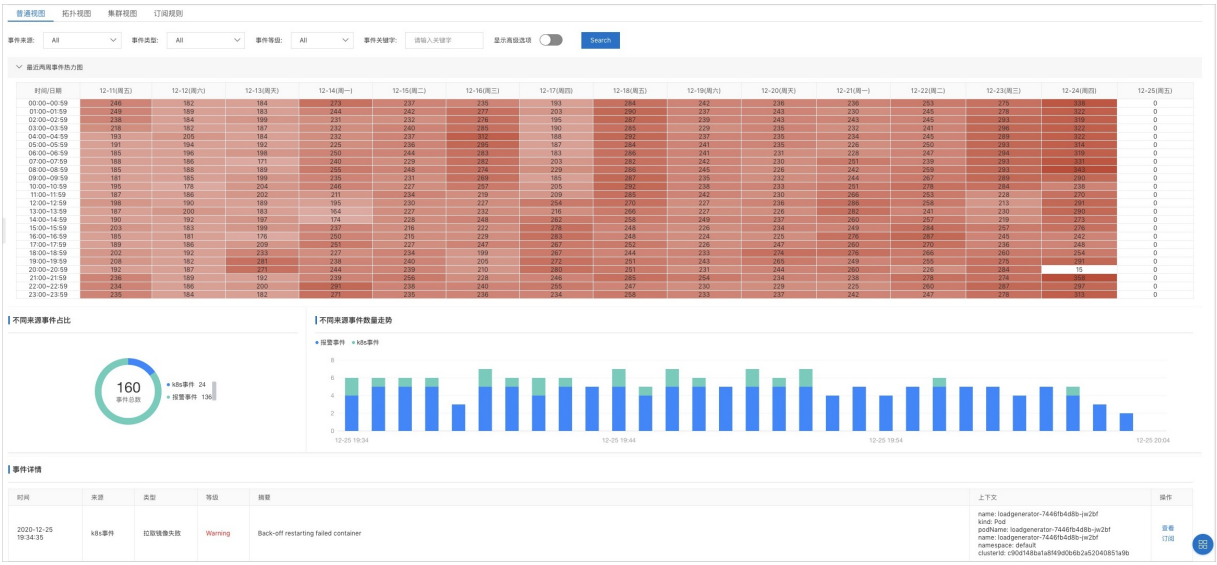
- 典型事件：展示系统预置的典型事件的数量。更多信息，请参见典型事件。
- **普通视图**将与当前应用关联的所有事件进行简单的多维度分析与展示。更多信息，请参见普通视图。
- **拓扑视图**将当前应用关联的事件和与当前应用的资源拓扑图进行结合展示。更多信息。请参见拓扑视图。
- **集群视图**将K8s集群关联的所有事件进行简单的多维度分析与展示。更多信息。请参见集群视图。
- **订阅规则**以列表形式展示了当前您创建的订阅规则。更多信息，请参见订阅规则。

典型事件

典型事件：展示系统预置的典型事件类型在最近30分钟（可在右上角调整时间范围）内发生的次数。在事件下方单击**订阅**，可以编辑该事件的订阅规则。订阅规则的操作，请参见订阅规则。

普通视图

普通视图可以按照您指定的检索条件进行搜索，搜索的结果会以四个视图进行展示，分别是：**最近两周事件热力图**、**不同来源事件占比**、**不同来源事件数量走势**和**事件详情**。



- **最近两周事件热力图**展示近两周内满足过滤条件的事件发生次数按小时统计的热力分布。颜色越深，说明该小时内发生的事件数量越多。
- **不同来源事件占比**展示不同来源事件的数量占比。
- **不同来源事件数量走势**展示不同来源事件在选定时段内的走势。
- **事件详情**展示当前所有事件的详情列表。

普通视图查看说明：

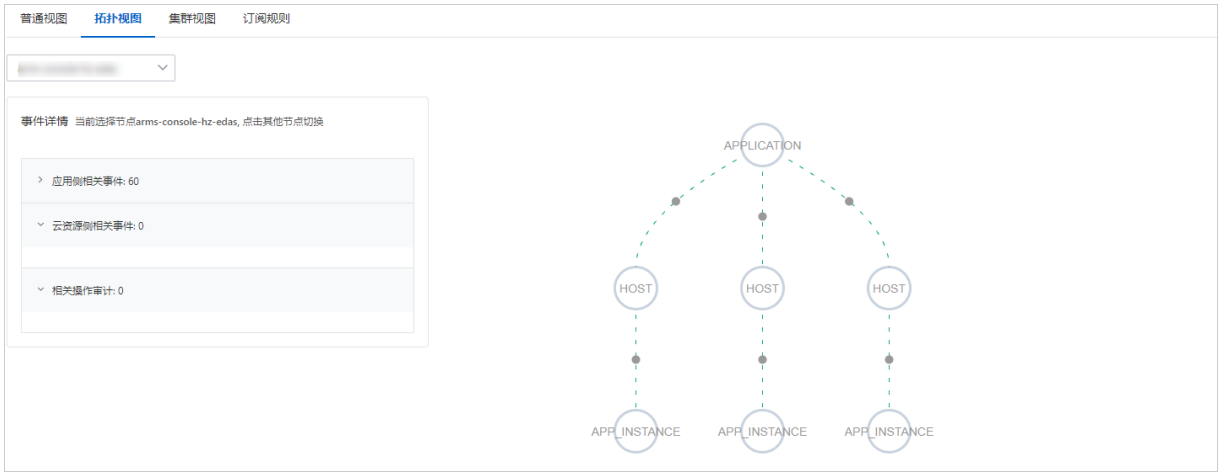
- 单击**最近两周事件热力图**中的热力方块，可查看该小时内的事件详情。
- 单击**不同来源事件数量走势**中的立柱，可在**事件详情**区域查看对应时段内的所有事件列表。
- 单击**事件详情**右侧的**查看**，可以查看事件的具体内容。
- 单击**事件详情**右侧的**订阅**，可以订阅指定事件。更多信息，请参见订阅规则。

拓扑视图

拓扑视图首先会绘制出该应用的资源拓扑，包含该应用使用的ECS，该应用部署的实例Pod，该应用使用的所有RDS和Redis等中间件资源，以及该应用挂载的SLB和NAT等。之后会将获取到的关联事件、操作审计、云监控事件关联到对应的拓扑节点上，单击相应节点，会在左上角的**事件详情**区域展示与该节点关联的事件：

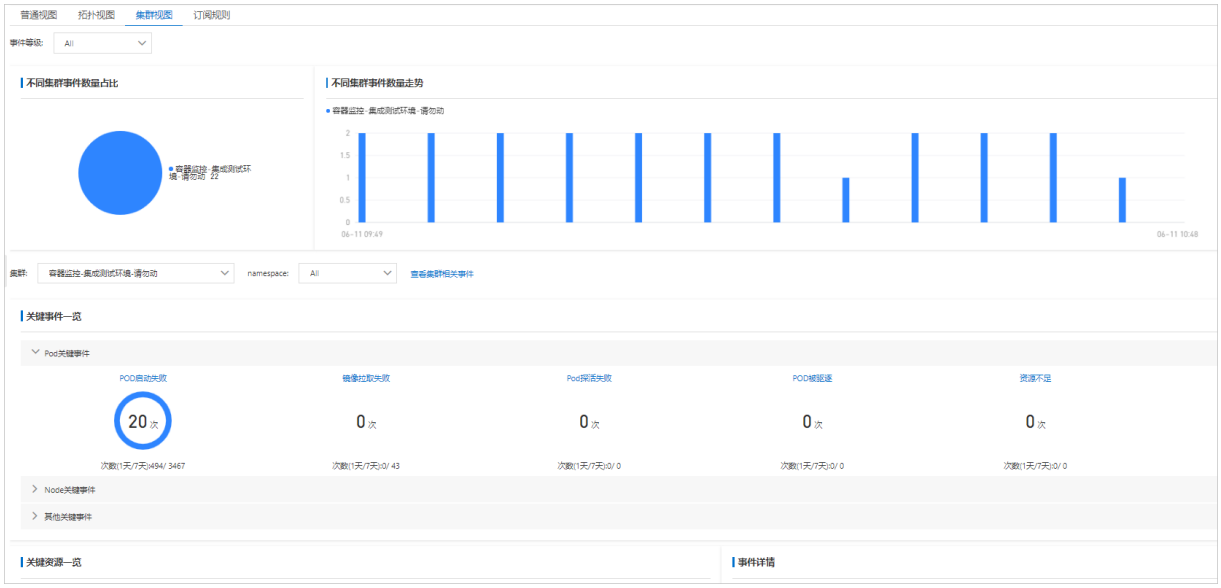
- **应用侧相关事件**：即事件中心的事件。
- **云资源侧相关事件**：即云监控的相关事件。
- **相关操作审计**：即来自于操作审计的审计记录。

该视图可以帮助您在应用出现故障时，快速排查关联的资源各自发生了什么问题。例如在大型企业中，由于某个员工的误操作，重启了生产环境的RDS，导致线上业务故障，利用该视图，可以快速的发现应用访问的RDS出现了重启操作。



集群视图

集群视图可以按照您指定的检索条件进行搜索，搜索的结果会以四个视图进行展示，分别是：**不同集群事件数量占比**、**不同集群事件数量走势**、**关键事件一览**、**关键资源一览**和**事件详情**。



- **不同集群事件数量占比**：展示不同集群事件的数量占比。
- **不同集群事件数量走势**：展示不同集群事件在选定时段内的走势。
- **关键事件一览**：展示指定集群下系统预置的典型事件类型在一段时间内发生的次数。
- **关键资源一览**：展示指定集群下关键资源的详情列表。
- **事件详情**：展示指定集群下所有事件的详情列表。

集群视图查看说明：

- 单击**不同集群事件数量走势**中的立柱，可在**事件详情**区域查看对应时段内的所有事件列表。
- 单击**查看集群相关事件**，可在**普通视图**页签查看所有集群相关事件详情。
- 单击**关键事件一览**中的事件名称，可在**普通视图**页签查看对应事件详情。
- 单击**关键资源一览**中的**查看生命周期相关事件**，可在**事件详情**区域查看对应时段内的资源对应的事件列表。
- 单击**事件详情**右侧的**查看**，可以查看事件的具体内容。
- 单击**事件详情**右侧的**订阅**，可以订阅指定事件。更多信息，请参见订阅规则。

订阅规则

订阅规则页签展示您当前所有的订阅规则，您可以在该页面启用、禁用或者修改规则。

> 注意 您只能修改您自己在控制台创建的规则，无法修改系统内部自动创建的规则。

订阅规则是一条表示您订阅满足指定条件的事件，并将该事件发送到指定Webhook的规则。创建订阅规则有两种方式：

- 方式一：在**订阅规则**页签右上角单击**新建订阅规则**。
- 方式二：在**普通视图**页签的事件列表区域单击事件的**订阅**。

1. 单击**订阅规则**页签，在页面右上角单击**新建订阅规则**。
2. 在**创建订阅规则**面板的**填写基本信息**页面输入**规则名称**和**规则描述**，然后单击**下一步**。
3. 在**选择事件模式**页面设置事件规则参数，然后单击**下一步**。

参数	描述
事件来源	在下拉列表中选择事件来源。
事件类型	在下拉列表中选择事件类型。
事件等级	在下拉列表中选择事件等级。
事件关键字	在文本框中输入事件关键字。
显示高级过滤选项	默认关闭，开启后可设置**关联应用ID**、**主机IP**、**集群ID**和**POD名**过滤选项。
自定义过滤条件	自定义过滤条件一般用来指定事件体JSON的某个字段需要满足的条件，根节点是data，以 . 的形式下钻事件体JSON的某个字段。请输入自定义的过滤条件，最多可以设置6条过滤条件。
选择有效字段	以data.x.y的形式选择有效字段，选择完成后，请给选择的字段输入一个别名，该别名可用于在填写Webhook信息的Post请求体中以占位符的形式出现。最多可以设置6条有效字段。
消息通知模板	消息通知模板会作为当指定消息发生时通知给您的内容。如果通知对象为钉钉机器人Webhook，请注意在消息模板中包含创建钉钉机器人Webhook时的关键字。

4. 在**选择联系人**页面选择联系人，然后单击**提交**。

   如果**选择联系人**的列表中没有联系人信息，在右侧单击**新建联系人**，创建联系人后再在列表中选择。

1. 在**普通视图**页签的**事件详情**区域，单击事件详情右侧的**订阅**。

   以该种方式创建的订阅规则，会根据您选择的事件来自动选择事件来源、事件类型和事件等级三个过滤条件。

2. 单击事件JSON文件中某个字段*Value*的形式来选择自定义过滤条件和有效字段。每次单击会自动生成一个过滤条件和选择字段，同时您还可以手动修改或者删除字段来调整规则，完成事件模式设置后单击**下一步**。

3. 在**选择联系人**页面选择联系人，然后单击**提交**。

   如果**选择联系人**的列表中没有联系人信息，在右侧单击**新建联系人**，创建联系人后再在列表中选择。

新建订阅规则

在普通视图中订阅事件

3.9. 数据库调用

本文说明如何查看数据库调用，从而了解应用的数据库调用情况，包括概况、SQL调用、异常、调用来源、调用链列表等信息。

前提条件

接入应用监控

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**数据库调用**。
6. 在显示的页面，选择数据库，设置时间段。

7. 完成设置后，您可以执行以下操作：
   - 单击**概览**页签，查看应用的数据库调用概况。
   - 单击**SQL调用分析**页签，查看应用的SQL调用。
   - 单击**异常分析**页签，查看应用的数据库调用异常。
   - 单击**调用来源**页签，查看应用的数据库调用来源。
   - 单击**调用链查询**页签，查看应用的调用链列表。

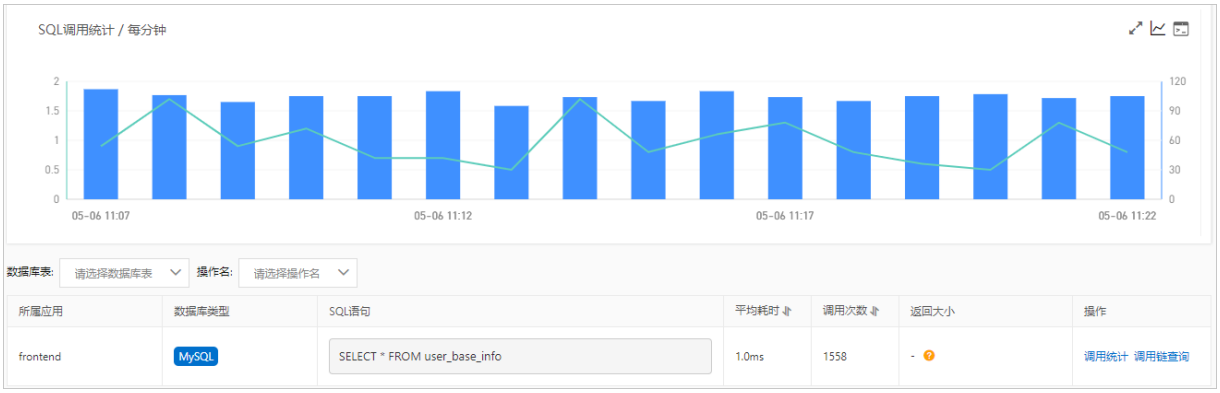
概览

概览页签下显示数据库的调用关系拓扑、请求数、响应时间、错误数、返回大小和性能一览。

1. （可选）在**概览**页签下，您可以执行以下操作：
   - 单击 图标或向上滑动鼠标滚轮，放大应用拓扑图。
   - 单击 图标或向下滑动鼠标滚轮，缩小应用拓扑图。
   - 单击 图标，将拓扑图恢复至默认大小。
   - 单击 图标，可以将拓扑图调整为适应页面大小。
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。
   - 单击图标，为该指标创建报警。具体操作，请参见创建报警。

SQL调用分析

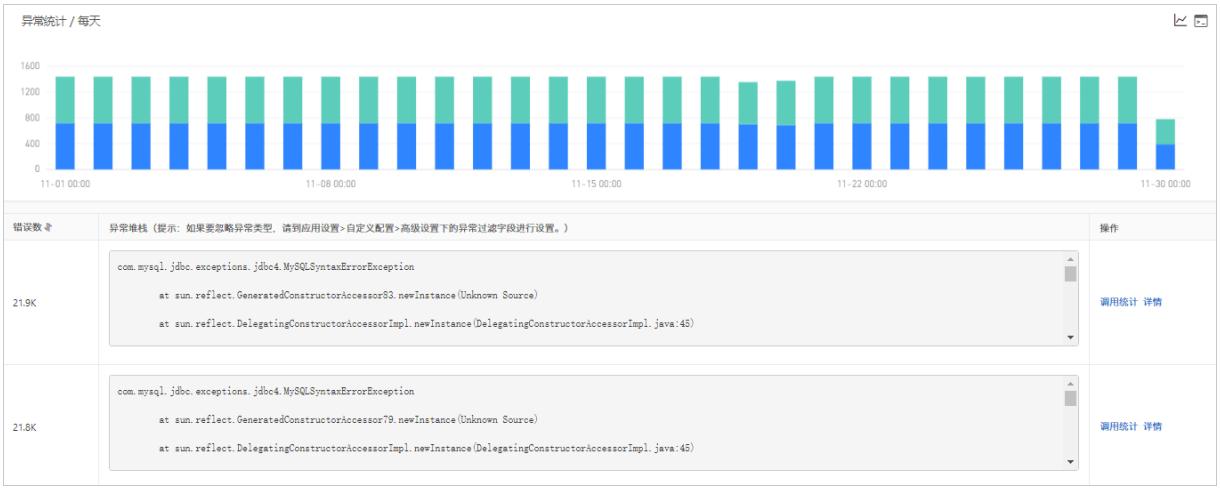
SQL调用分析页签下显示数据库的SQL调用数柱状图、响应时间时序曲线、SQL语句列表等信息。



1. （可选）在**SQL调用分析**页签下，您可以执行以下操作：
   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。
   - 在SQL语句的**操作**列，单击**调用统计**，查看该SQL语句的SQL调用统计。
   - 在SQL语句的**操作**列，单击**调用链查询**，查看该SQL语句的调用链列表。

异常分析

异常分析页签下显示数据库的异常情况。



1. （可选）在**异常分析**页签下，您可以执行以下操作：

   > **说明** 如果要过滤异常，在**应用设置** > **自定义设置** > **高级设置**的**异常过滤**文本框设置。

   - 将光标移到统计图上，查看统计情况。
   - 使用光标选中一段时间，查看指定时间段的统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。
   - 单击图标，查看该指标的API详情。
   - 在异常的**操作**列，单击**调用统计**，查看该异常的统计情况。

- 在异常的**操作**列，单击**详情**，查看该异常的详情。

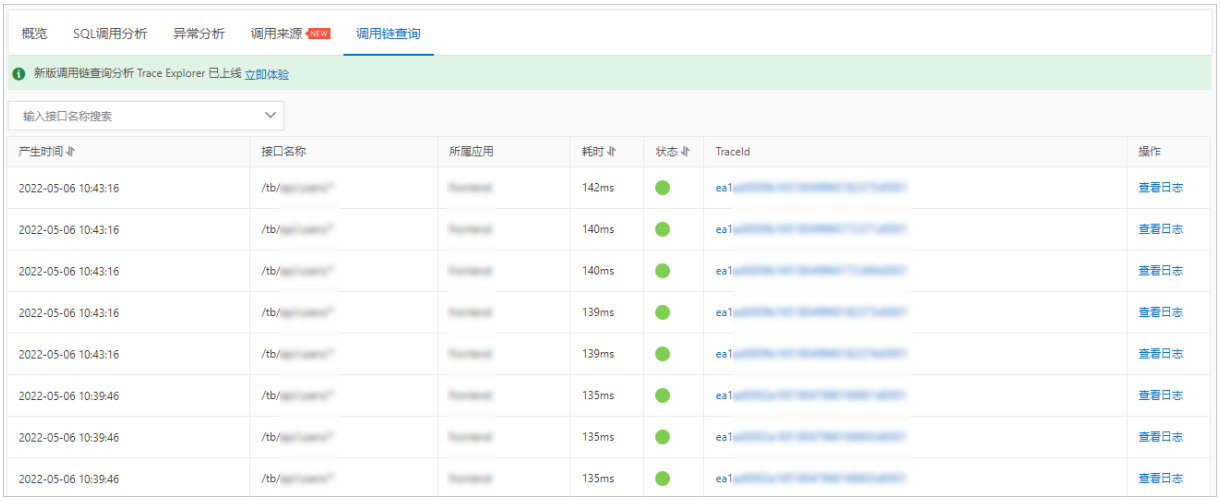
调用来源

调用来源页签下显示数据库的调用来源情况。

1. （可选）在**调用来源**页签下，您可以执行以下操作：
   - 在搜索框，输入应用或接口名称，单击图标，查看应用或接口的情况。
   - 在调用来源应用的接口右侧，单击**查看详情**，查看该接口的调用链列表。
   - 将光标移到统计图上，查看统计情况。
   - 单击图标，查看该指标在某个时间段的统计情况或对比不同日期同一时间段的统计情况。

调用链查询

调用链查询页签下显示数据库的所有调用链列表。



1. （可选）在**调用链查询**页签下，您可以执行以下操作：
   - 在搜索框，输入接口名称，然后单击图标，查看指定接口的调用链列表。
   - 在TraceId列，单击TraceId名称，查看调用链路详情。
   - 在**操作**列，单击**查看日志**，查看该调用链的日志信息。

3.10. 外部调用

当您需要定位应用外部调用缓慢或出错的问题时，可以使用ARMS应用监控的外部调用功能实现问题定位。

功能入口

1. 登录ARMS控制台。

2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏中单击**外部调用**。

   外部调用页面的左侧列表展示了应用的所有外部调用。您可以按照响应时间、请求数、错误数或异常数对该列表排序。

概览

在左侧列表选中其中一个外部调用，可在**概览**页签上查看该外部调用的请求数、响应时间、错误数以及HTTP-状态码的时序曲线。

调用来源

在左侧列表选中其中一个外部调用，可在**调用来源**页签上查看该外部调用所有接口的响应时间、请求数和错误数的时序曲线。

在**调用来源**页签上，可按需执行以下操作：

- 在页签顶部单击**全部折叠/展开**，即可折叠或展开下方的所有接口。
- 在页签顶部的搜索框内输入应用名称或接口（Span）名称的关键字，并单击搜索图标，即可筛选出符合条件的接口。
- 单击接口信息所在的折叠面板，或者单击行末的上箭头或下箭头，即可展开或折叠该接口的性能指标信息。

接口快照

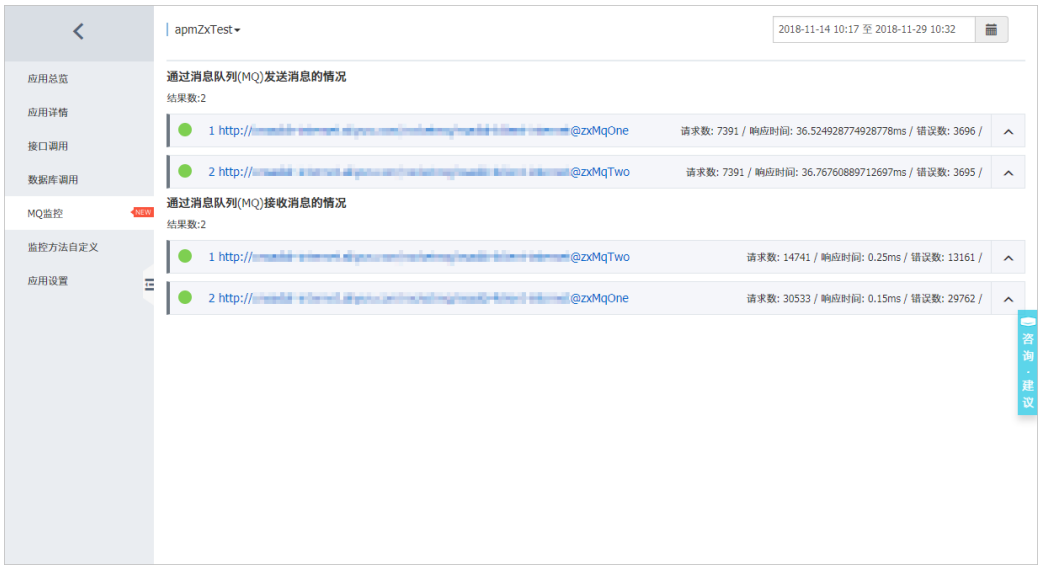
在左侧列表选中其中一个外部调用，可在**接口快照**页签上查看该外部调用的参数详情，单击TraceId可以查看调用链路和业务轨迹。

3.11. MQ监控

ARMS应用监控的MQ监控可展示消息队列RocketMQ版的Topic发布和订阅消息的情况。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用名称。
4. 在左侧导航栏中单击**MQ监控**。
5. 在页面右侧单击查询结果的链接。



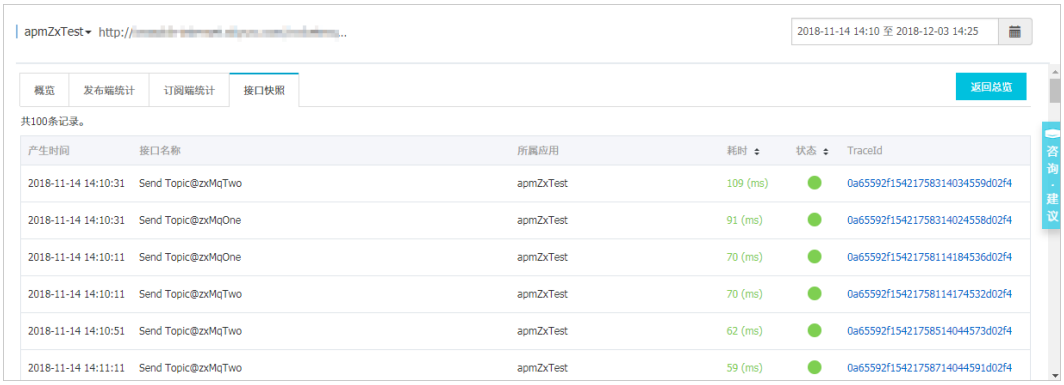
功能介绍

- 在**概览**页签中以拓扑图展示应用与MQ数据源之间的消息发布和订阅关系。

- 在**发布端统计**页签展示消息发布的统计数据，包括请求数、响应时间和错误数。

- 在**订阅端统计**页签展示消息订阅的统计数据，包括消息请求数、响应时间和错误数。

- 在**接口快照**页签提供关于消息发布和订阅的接口快照，您可以通过TraceId链接查看完整调用链以及诊断问题原因。



3.12. 主动剖析

ARMS应用监控的主动剖析功能可以根据您的实际需求，手动触发线程剖析监听任务。

前提条件

接入应用监控

> **说明** 该功能需要升级探针版本至2.7.1.1及以上，如何升级探针，请参见升级探针。

新增主动剖析任务

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**主动剖析**。
6. 在**新增任务**区域设置所有参数，然后单击**完成**。

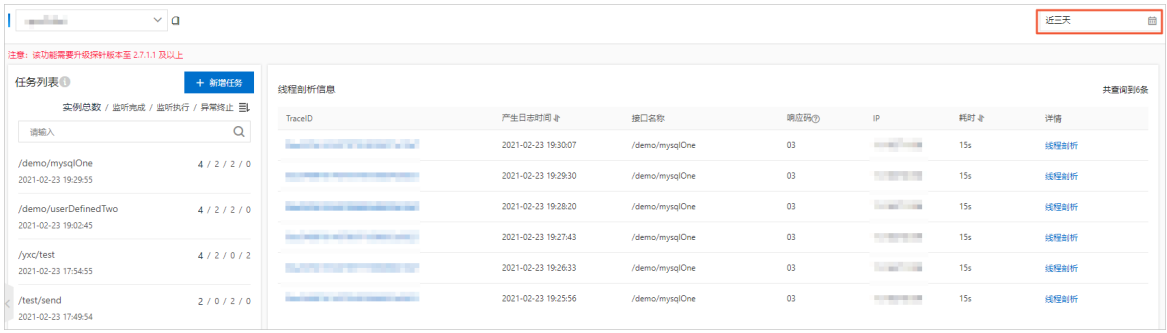
参数	描述
接口名称	接口名称。

参数	描述
监控持续时间	一次监控任务有效的生命周期。选项： ○ 5min ○ 10min ○ 15min
剖析开始阈值（ms）	超过剖析阈值的慢调用才会被监听。例如：一个持续时间为1.4s的请求，如果剖析开始阈值设置为100ms，那么只有100ms~1400ms的时间段会被监听。取值范围：50ms~30000ms。
线程剖析频率（ms）	线程剖析过程中，接口快照的打印频率。取值越小，精度越高，相应的性能开销越大，建议频率设置为100ms。取值范围：5ms~1000ms。
最大采样数	最多监听的慢调用数量。取值范围：1~100。

查看线程剖析信息

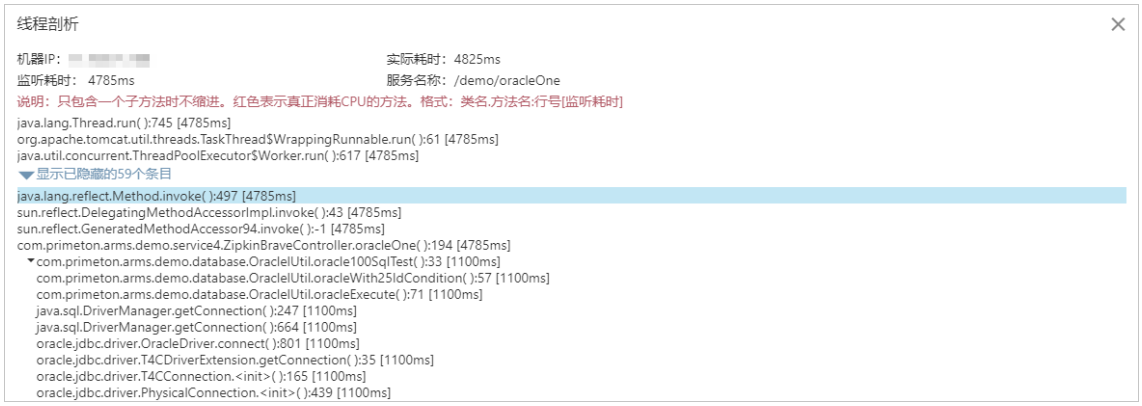
1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**主动剖析**。
6. 在**新增任务**区域单击**查看历史任务**。
7. 在页面右上角选择需要查看的时间段，并在在**任务列表**区域单击需要查看的主动剖析任务。

   在**任务列表**区域显示所选时间段内所有的主动剖析任务。



在右侧**线程剖析信息**区域显示当前选中接口的剖析信息，包括：TraceID、产生日志时间、接口名称、响应码、IP和耗时。

- 单击TraceID，可以查看调用链路详情。更多信息，请参见调用链路查询。
- 单击**详情**列的**线程剖析**，查看目标链路的线程剖析详情。



查看线程剖析任务详情

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**主动剖析**。
6. 在**新增任务**区域单击**查看历史任务**。
7. 单击**任务列表**区域目标剖析任务右下角的**详情**。

   在**日志**页签可以查看该任务的详细信息。

8. 单击**日志**页签。

   在**日志**页签可以查看该任务的日志信息。

相关文档

- 使用线程剖析诊断代码层面的问题

3.13. 应用诊断

3.13.1. 实时诊断

实时诊断功能适用于在短时间内密切监控应用性能和定位问题原因的场景。本文介绍实时诊断功能的使用方法。

背景信息

当您需要密切监控一小段时间内的应用性能时，例如发布应用或者对应用进行压测时，可以使用实时诊断功能。开启实时诊断后，本功能会持续监控应用5分钟，并在这5分钟内全量上报调用链数据。接下来，您就能以出现性能问题的调用链路为起点，通过方法栈瀑布图和线程剖析等功能定位问题原因。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏选择**应用诊断 > 实时诊断**。

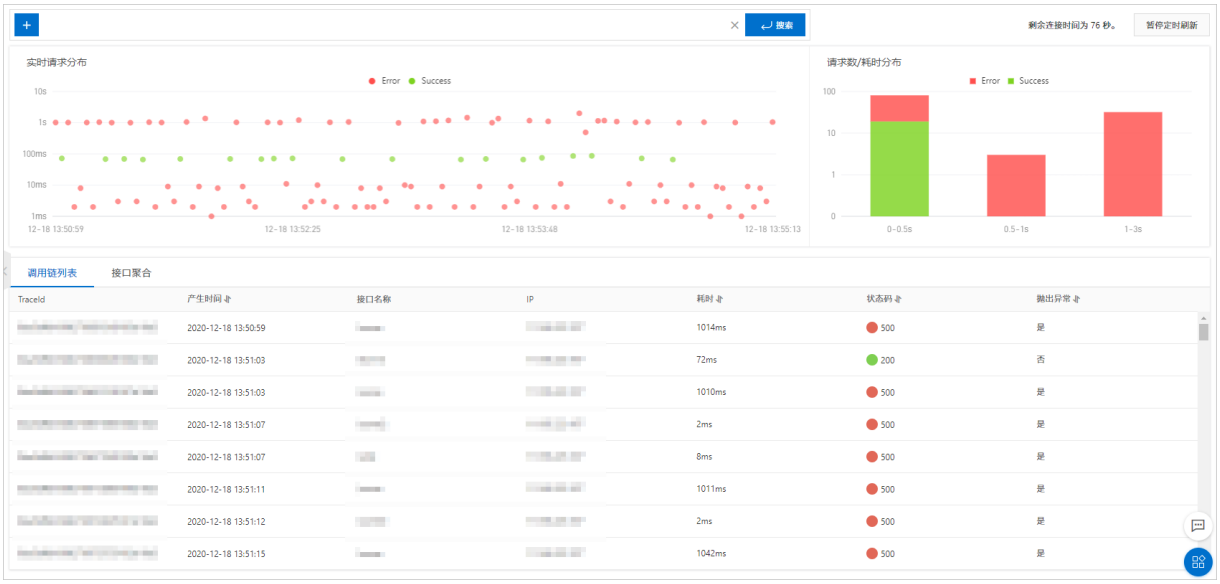
开启和终止实时诊断

首次进入**实时诊断**页面时，默认自动开启实时诊断。其他情况下，如需开启实时诊断，请单击页面右上角的**开启实时诊断**。

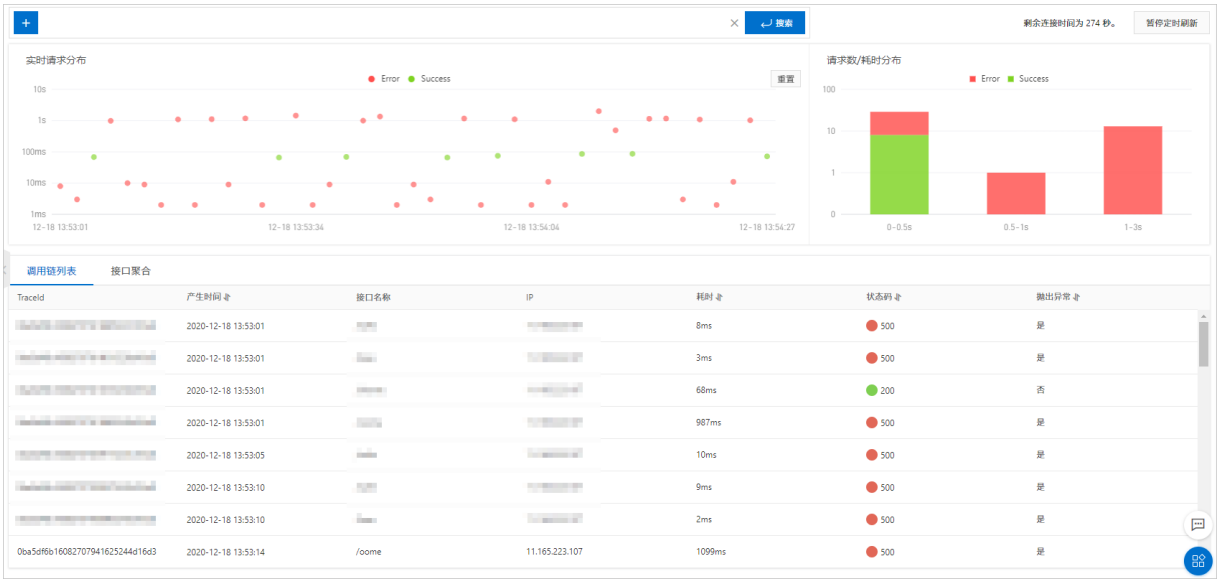
实时诊断将于自动开启5分钟后自动终止。如需提前终止实时诊断，请单击页面右上角的**暂停定时刷新**。

查看实时监控数据

在**实时请求分布**和**请求数/耗时分布**区域，您可以查看截至当前时间点捕捉到的最后1000次请求统计数据。



在**实时请求分布**区域的图表中，用鼠标框选一段时间区间，即可将所选时间区间设为数据可视时间区间，即图表中仅显示该时间区间内的数据。此后，单击图表右上角的**重置**即可恢复为默认视图。



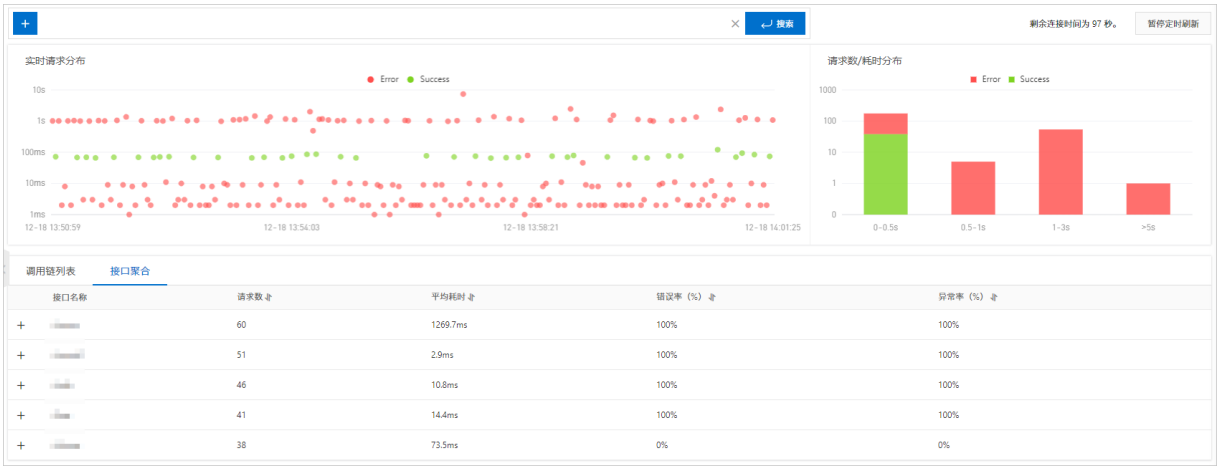
筛选监控数据

您可以按照接口名称和IP筛选页面上显示的请求监控数据。

1. 在**实时请求分布**区域上方单击+图标。
2. 在下拉框中选择一个API或IP，并单击**搜索**。
   仅选中接口的请求监控数据会显示在页面上。

查看调用链信息

在**调用链列表**和**接口聚合**页签上，您可以查看相应时间区间内捕捉到的全部调用链信息。单击一个TraceId，即可进入**调用链路**页面，并借助本地方法栈瀑布图和线程剖析等功能定位问题原因。



相关文档

- 调用链路查询
- 使用线程剖析诊断代码层面的问题

3.13.2. 线程分析

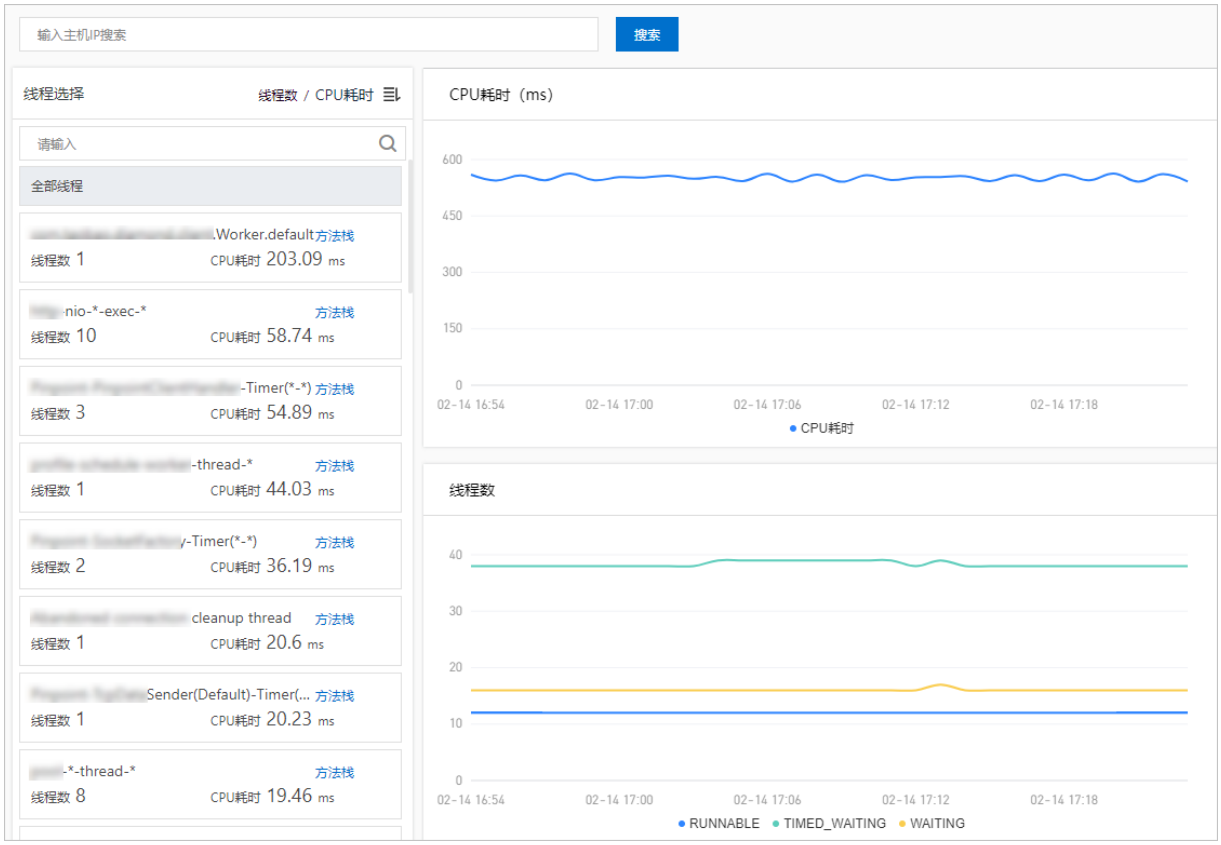
线程分析功能提供线程粒度的CPU耗时和每类线程数量的统计，并且每5分钟记录一次线程的方法栈并聚合，可真实还原代码执行过程，帮助您快速定位线程问题。当发现集群的CPU使用率过高，或者出现大量慢方法时，可以通过线程分析功能找到消耗CPU最多的线程或方法。

功能入口

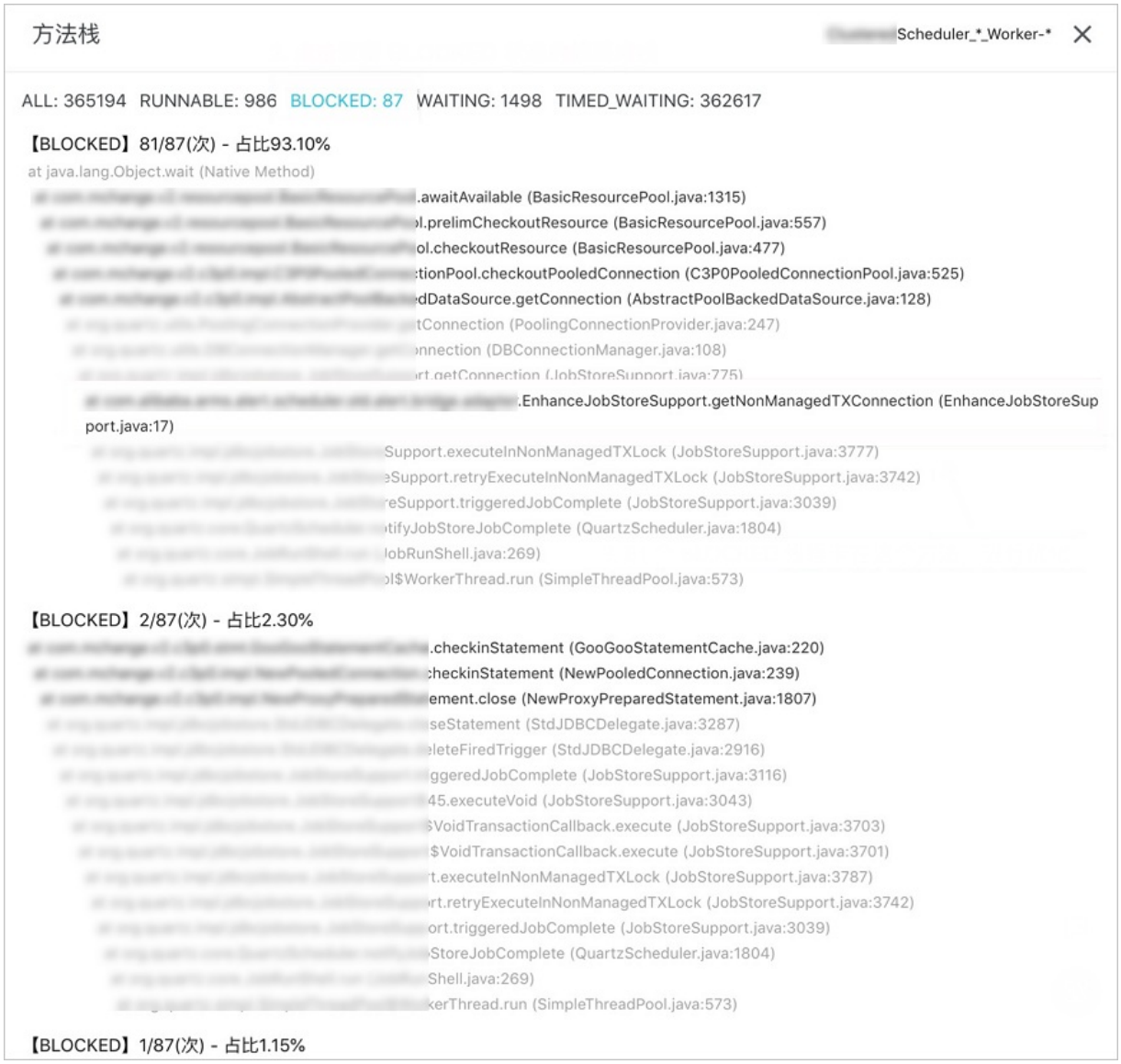
1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏中选择**应用诊断 > 线程分析**。

进行线程分析

线程分析页面的左侧列表展示了应用的全部线程，您可以根据**CPU耗时**统计快速发现异常线程。选中某一异常线程后，再根据右侧的**CPU耗时**和**线程数**曲线图分析CPU耗时与线程数变化，例如分析每分钟的线程总数是否过多。



您还可以单击异常线程的**方法栈**，查看指定时间内的真实运行方法栈，例如查看处于BLOCKED状态的线程对应的方法，从而优化指定代码段，以便降低CPU使用率。



说明 如果单击**方法栈**后，显示无数据，则需要单击左侧导航栏的**应用设置**，在**自定义配置**页签**线程设置**区域查看**线程诊断方法栈**开关是否开启。如未开启，则无法记录方法栈信息；如已开启，则每5分钟采集一次方法栈信息。

相关文档

- 使用线程剖析诊断代码层面的问题

3.13.3. 日志分析（直接采集）

当应用出现业务异常问题时，您可以通过分析业务日志，精准定位业务异常。日志分析功能支持分析日志服务SLS或直接采集的日志，本文介绍如何开通日志分析功能并分析直接采集的日志。

前提条件

ARMS Agent版本为v2.7.1.4或以上。登录ARMS控制台，在**应用监控 > Agent列表**页面的**Java版本发布说明**页签获取2.7.1.4版本的Agent安装包。

日志源说明

日志分析功能支持分析的日志源包括：日志服务SLS的日志、直接采集的日志。本文仅介绍如何查询并分析直接采集的日志，日志服务SLS的日志分析详情，请参见日志分析（日志服务SLS）。

- 日志分析（直接采集）：通过ARMS探针采集日志框架的输出并直接推送到ARMS的日志分析中心，通过一键开启后无需其他操作即可在ARMS控制台查询分析应用日志。

  > **说明** 日志分析（直接采集）功能需要将探针升级到v2.7.1.4及以上版本。

- 日志分析（日志服务SLS）：
  您需要将应用的日志采集到日志服务SLS，并在ARMS应用配置中配置相应的Project和Logstore，ARMS会内嵌日志服务的页面方便您进行日志分析。

功能开通说明

开通日志采集功能

1. 在ARMS控制台左侧导航栏，选择**应用监控 > 应用列表**。
2. 在顶部菜单栏，选择地域，然后单击应用名称。
3. 在左侧导航栏中单击**应用设置**，并在右侧单击**自定义配置**页签。
4. 在**自定义配置**页签的**应用日志关联配置**区域，选择日志源为默认日志，打开**日志框架自动采集**开关，选择需要采集的日志级别，并设置单条日志显示的最大长度和日志上报限制。

   > **说明** 日志分析功能不会采集日志级别低于应用中配置的日志输出级别的日志。例如应用中配置的日志输出级别为Warn，即使在ARMS控制台配置采集了Debug级别的日志，也不会生效。如果想不重启应用修改应用日志输出级别，您可以使用ARMS的Arthas诊断中的Arthas Shell功能动态修改日志框架的日志输出级别，更多信息，请参见Arthas Shell。

5. 在**自定义配置**页签左下角单击**保存**。

日志采集原理

1. 通过ARMS探针拦截日志框架的日志打印方法，获取日志内容。
2. 为日志增加部分上下文（例如traceId、spanName、threadName）。
3. 将日志推送至ARMS日志分析中心存储。
4. 在ARMS控制台的日志分析页面，查询该条日志。

> **说明** 整个日志采集链路时延大约为10s。

功能额外开销

- CPU额外开销：0.01核

- 内存额外开销：20 M以内
- 带宽额外开销：最大为每秒配置的日志上报条数上限×配置最长日志字符数（个）

单条日志结构

一条日志包含日志附带的Label以及日志自身解析出的字段。

日志附带的Label字段

字段	说明
majorVersion	探针主版本号。
minorVersion	探针子版本号。
serverIp	探针所在主机IP。
userId	应用所属用户ID。
appId	应用ID。
job	日志来源，固定为arms-agent。
logType	日志类型，固定为userLog。

日志解析出的字段

字段	说明
level	日志级别。
log	包括格式化时间和日志原文。时间和日志中间使用短划线（-）连接。
loggerName	Logger名称。
parentAppId	本条日志关联的调用链中上游应用ID。
spanName	本条日志关联的SpanName。
parentSpanName	本条日志关联的上游SpanName。
threadName	线程名称。
traceId	关联的TraceID。
ts	日志上报时间。

常用查询语句示例

下面以appId为a2n80plglh@89f2dd21b561bdc的应用为例说明。

关键字查询

例如搜索包含关键字error的日志，查询语句为：

```
{job="arms-agent", appId="a2n80plglh@89f2dd21b561bdc"} |= "error"
```

例如搜索traceId为eaac105afb16540713955671006d0009的日志，查询语句为：

```
{job="arms-agent", appId="a2n80plglh@89f2dd21b561bdc"} |= "eaac105afb16540713955671006d0009
"
```

多条件查询

例如搜索同时包含关键字error和Exception的日志，查询语句为：

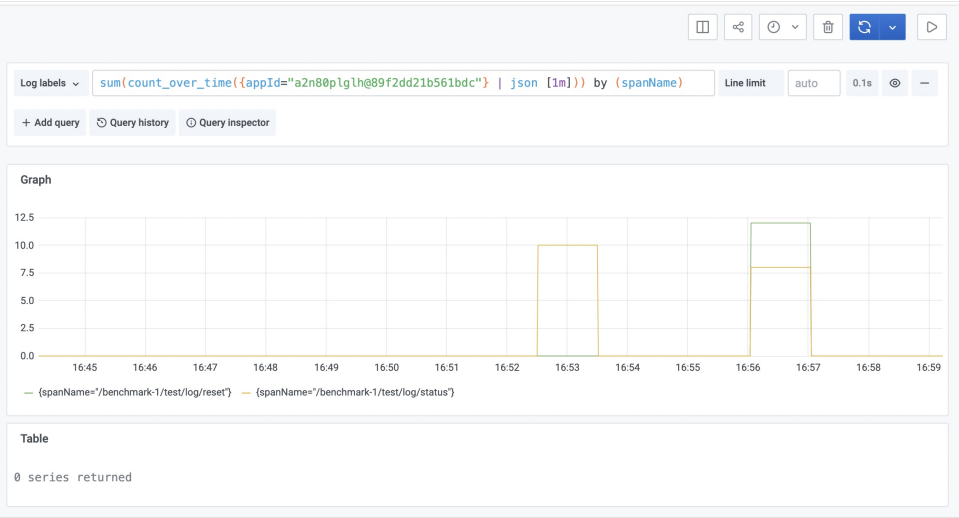
```
{job="arms-agent", appId="a2n80plglh@89f2dd21b561bdc"} |= "error" |= "Exception"
```

分析日志

例如想要分析不同spanName error日志输出的数量变化趋势，查询语句为：

```
sum(count_over_time({job="arms-agent", appId="a2n80plglh@89f2dd21b561bdc"} | json [1m])) by
(spanName)
```

查询结果如下：



同理可以查询不同loggerName、threadName、level的日志输出数量变化。

3.13.4. 日志分析（日志服务SLS）

当应用出现业务异常问题时，您可以分析业务日志，精准定位业务异常。日志分析功能支持分析日志服务SLS或直接采集的日志，本文介绍如何开通日志分析功能并分析日志服务SLS的日志。

前提条件

- 已接入应用监控。具体操作，请参见应用监控接入概述。
- 已开通日志服务SLS。登录日志服务控制台时，根据页面提示开通日志服务。
- 已创建Project。具体操作，请参见创建Project。
- 已创建Logstore，具体操作，请参见创建Logstore。

日志源说明

日志分析功能支持分析的日志源包括：日志服务SLS的日志、直接采集的日志。本文仅介绍如何查询并分析日志服务SLS的日志，直接采集的日志分析详情，请参见日志分析（直接采集）。

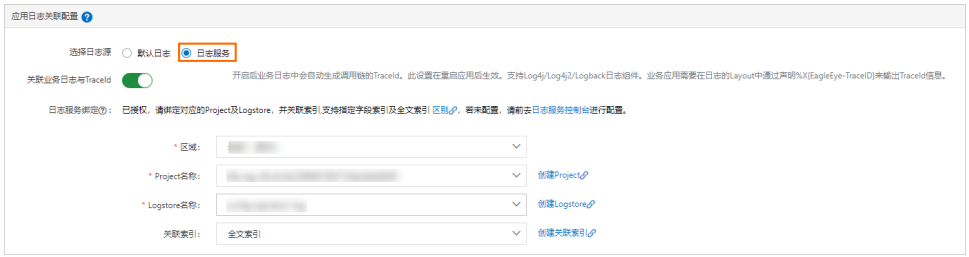
- 日志分析（直接采集）：通过ARMS探针采集日志框架的输出并直接推送到ARMS的日志分析中心，通过一键开启后无需其他操作即可在ARMS控制台查询分析应用日志。

  > **说明** 日志分析（直接采集）功能需要将探针升级到v2.7.1.4及以上版本。

- 日志分析（日志服务SLS）：
  您需要将应用的日志采集到日志服务SLS，并在ARMS应用配置中配置相应的Project和Logstore，ARMS会内嵌日志服务的页面方便您进行日志分析。

步骤一：关联业务日志

1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用列表**，在顶部菜单栏选择目标地域，然后在**应用列表**页面单击目标应用的名称。
3. 在左侧导航栏中单击**应用设置**，并在右侧单击**自定义配置**页签。
4. 在**自定义配置**页签的**应用日志关联配置**区域，选择日志源为日志服务，打开**关联业务日志与TraceId**开关，选择日志服务所在地域，然后绑定Project和Logstore。



5. 在**自定义配置**页签左下角单击**保存**。

步骤二：查询并分析日志

1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用列表**，在顶部菜单栏选择目标地域，然后在**应用列表**页面单击目标应用的名称。
3. 在左侧导航栏，选择**应用诊断 > 日志分析**。
4. 在右侧页面，执行以下操作：
   i. 在搜索框中输入查询分析语句。

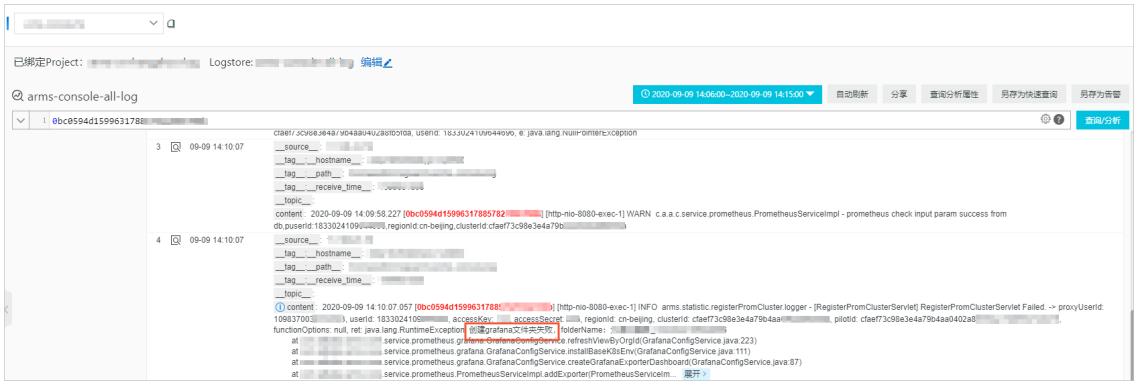
      查询分析语句由查询语句和分析语句构成，格式为查询语句|分析语句，查询分析语句语法请参见查询语法、SQL分析语法。

   ii. 设置查询分析的时间范围。

      您可以设置相对时间、整点时间和自定义时间。

      > **说明** 查询结果有1分钟以内的误差。

   iii. 单击**查询/分析**，查看查询分析结果。



3.13.5. 日志分析（旧版）

当应用出现业务异常问题时，您可以通过分析业务日志，精准定位业务异常。本文介绍如何开通并使用ARMS日志分析功能。

前提条件

- ARMS Agent版本为v2.7.1.3或以上。登录ARMS控制台，在**应用监控 > Agent列表**页面的**Java版本发布说明**页签获取2.7.1.3版本的Agent安装包。
- 日志分析功能目前仅支持对阿里云容器服务Kubernetes版（简称ACK）集群下应用的日志进行分析，且Kubernetes集群应用需要通过Annotations方式接入ARMS应用监控。通过Annotations方式接入ARMS应用监控的具体操作，请参见为容器服务Kubernetes版Java应用安装探针。

版本说明

当前版本的日志分析功能已不支持开通，如果已开通当前版本的日志服务，则仍可以使用。

您可以根据需求重新开通新版日志分析功能。

> **注意** 开通新版日志分析功能后无法返回旧版的日志分析功能。

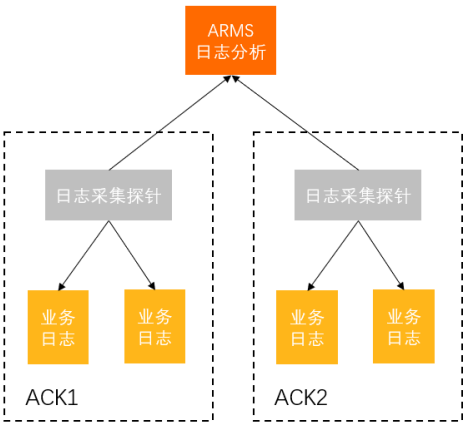
新版日志分析功能支持分析日志服务SLS或直接采集的日志：

- 日志分析（直接采集）：通过ARMS探针采集日志框架的输出并直接推送到ARMS的日志分析中心，通过一键开启后无需其他操作即可在ARMS控制台查询分析应用日志。更多信息，请参见日志分析（直接采集）。

  > **说明** 日志分析（直接采集）功能需要将探针升级到v2.7.1.4及以上版本。

- 日志分析（日志服务SLS）：
  您需要将应用的日志采集到日志服务SLS，并在ARMS应用配置中配置相应的Project和Logstore，ARMS会内嵌日志服务的页面方便您进行日志分析。更多信息，请参见日志分析（日志服务SLS）。

日志采集原理

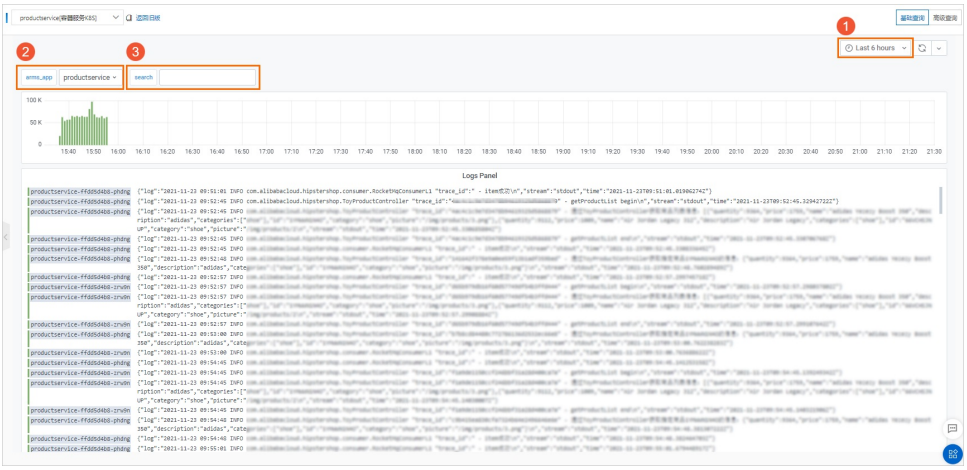


1. 应用开通日志采集功能后，采集到的业务日志将会保存至本地。
2. 安装日志采集探针后，日志采集探针将读取本地业务日志并发送至ARMS控制台。

> **说明** 同一个ACK集群下仅需安装一次日志采集探针。

3. 在ARMS控制台的日志分析页面，您可以通过关键字对业务日志进行查询、筛选、分析等操作。例如通过搜索订单ID、Trace ID、应用名称等参数，可以查看相关的异常日志信息。

查询日志

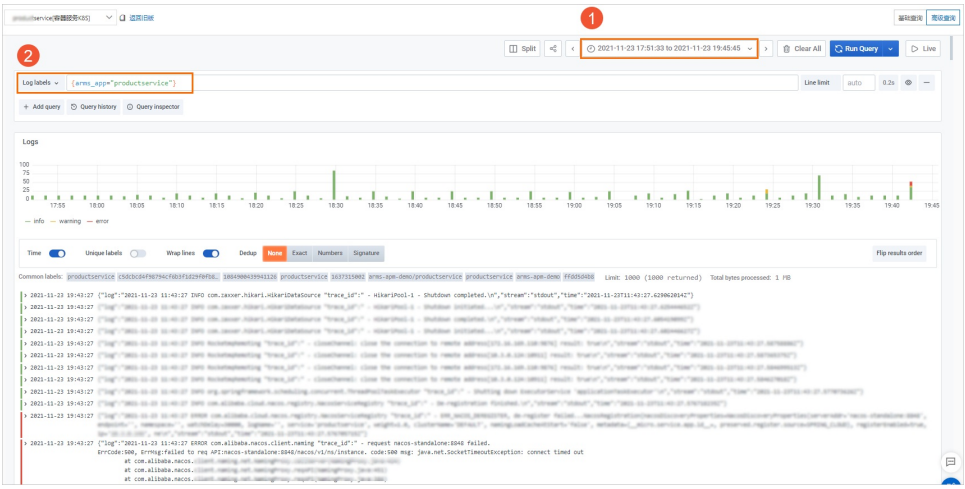


基础查询支持以下筛选操作：

- 在基础查询页面右上角（图示中①），设置日志需要查看的时间段。
- 单击arms_app右侧下拉框（图示中②）可以切换已开通日志采集的其他应用。
- 在search右侧文本框（图示中③）输入关键字，按回车或单击**search**可以搜索包含对应关键词的日志。例如输入*WARN*，可以查看所有Warn级别的日志信息。

> **说明** 关键字区分大小写。

单击页面右上角的**高级查询**，在高级查询页面，您可以通过LogQL语句查询日志信息。



高级查询支持以下筛选操作：

- 在高级查询页面右上角（图示中①），设置日志需要查看的时间段。
- 在查询文本框（图示中②）输入LogQL语句，单击右上角的**Run Query**筛选业务日志信息。
  LogQL语句支持通过日志标签和关键字查询日志信息，LogQL语句中必须包含日志标签。
  - 日志标签格式：{label键值对}。您可以单击**Log labels**，在下拉框中选择日志标签和标签值。
    label键值对支持的运算符如下：
    - `=`：等于
    - `!=`：不等于
    - `=~`：正则匹配
    - `!~`：正则不匹配

    例如：{arms_app="productservice"}，查询名称为productservice的应用的业务日志。
  - 关键字：在日志标签后添加关键字可以进一步过滤业务日志。
    关键字支持的运算符如下：
    - `|=`：包含关键字
    - `!=`：不包含关键字
    - `|~`：正则匹配
    - `!~`：正则不匹配

    > **说明** 关键字区分大小写。

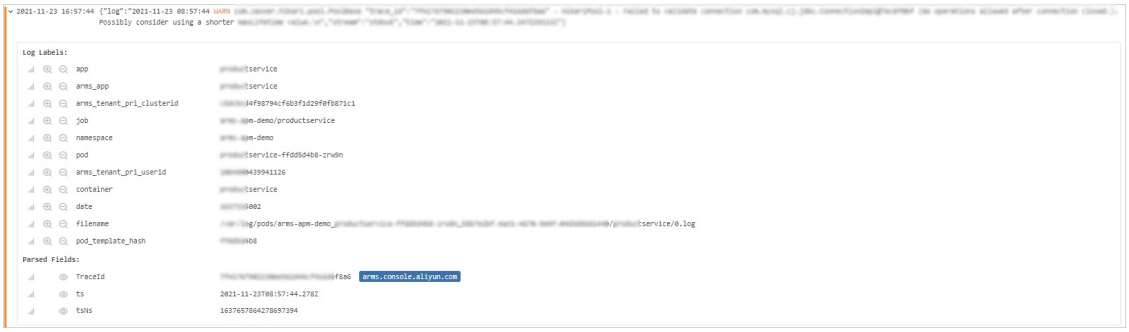
    例如：{arms_app="productservice"}|="WARN"，查询名称为productservice的应用下Warn级别的业务日志。

高级查询

查看日志

在业务日志区域，您可以执行以下操作：

单击日志所在行，可以查看目标日志包含的日志标签和关键字。

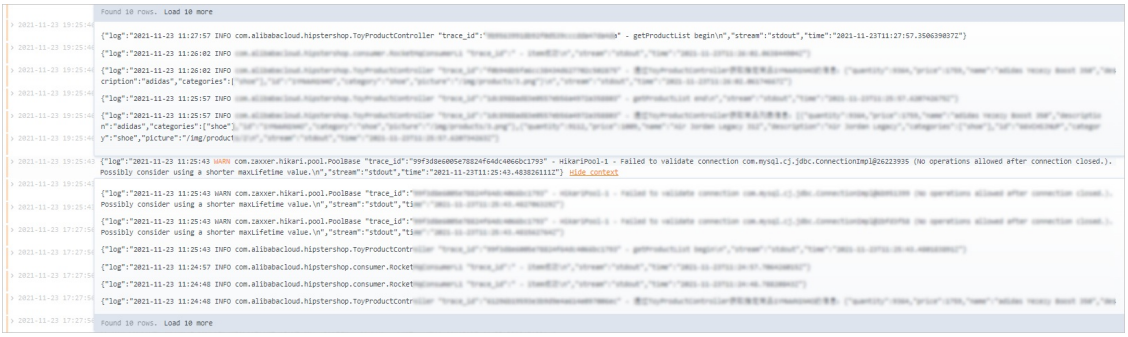


在日志展开信息区域，您可以执行以下操作：

- 单击标签前面的图标，可以筛选包含当前标签和标签值的日志。
- 单击标签前面的图标，可以筛选不包含当前标签和标签值的日志。
- 单击Trace ID右侧的控制台链接，可以查看对应的调用链路和业务轨迹。

如果您需要查看经过筛选后的某条日志的上下文，将鼠标悬浮于日志所在行，单击日志信息尾部的**Show context**。

展开的上下文默认前后各显示5条日志，如果您需要查看更多上下文，单击上下日志区域的**Load 10 more**。



查看日志详情

查看日志上下文

3.13.6. Arthas诊断（新版）

Arthas是诊断Java领域线上问题的利器，利用字节码增强技术，可以在不重启JVM进程的情况下，查看程序的运行情况。

前提条件

> **说明** 仅应用监控专家版支持Arthas诊断功能。

- 新开通ARMS的账号或15天内未使用旧版Arthas诊断功能的账号仅支持使用新版Arthas诊断功能。
  如果您在使用Arthas诊断中有任何问题，请搜索钉钉群号35396829进入ARMS应用诊断答疑群获取帮助。
- ARMS Agent版本为v2.7.1.3或以上。
- 已接入应用监控。具体操作，请参见应用监控概述。
- 应用的编程语言需要为Java。
- 3658端口未被占用。

背景信息

ARMS提供的Arthas诊断功能主要用于补齐ARMS在实时诊断方面的能力。ARMS的Arthas诊断功能包括以下几种类型：

- JVM概览：查看当前JVM进程实时的内存使用情况、系统信息、系统变量和环境变量。
- 线程分析：查看当前JVM进程的线程耗时情况以及指定线程的实时方法栈。
- 方法执行分析：抓取任意方法（非JDK方法）满足指定条件的一次执行记录，记录该方法的参数、异常、返回值以及方法内部各个方法执行耗时。
- 对象查看器：查看任意类的某个实例实时的属性取值情况。
- 实时看板：常见组件的实时看板，例如，Druid连接池的实时看板可以看到连接池的配置、使用情况以及SQL执行耗时情况。
- 性能分析：对CPU耗时、内存分配等对象进行一定时间的采样并生成相应的火焰图。
- Arthas Shell：通过命令行方式使用Arthas诊断。

开通Arthas诊断功能

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏中单击**应用设置**，并在右侧单击**自定义配置**页签。
6. 在**自定义配置**页签的**Arthas监控**区域，打开**Arthas开关**，根据需求选择是否仅对部分IP进行Arthas诊断，并添加目标IP。

7. 在**自定义配置**页签左下角单击**保存**。

查看Arthas诊断信息

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，选择**应用诊断 > Arthas诊断**。
6. 在Arthas诊断页面，在顶部下拉列表选择待诊断的应用实例。
   - 若该实例的Agent版本未升级至2.7.1.3或以上，页面会提示您需要先升级Agent。
   - 若该实例的Agent版本已升级至2.7.1.3或以上，页面会显示该实例的Arthas诊断信息。

JVM概览

JVM概览支持查看应用的JVM相关信息，包括JVM内存、操作系统信息、变量信息等，帮助您了解JVM的总体情况。

Arthas诊断页面默认显示**JVM概览**页签，您可以在**JVM概览**页签查看以下信息：

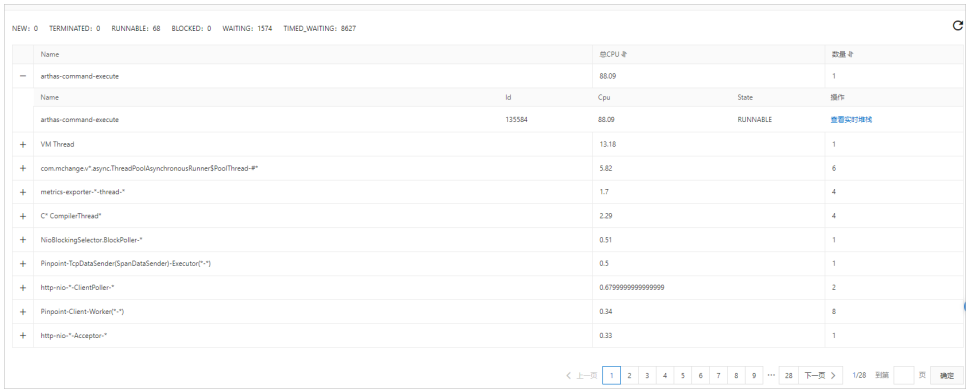
- **JVM内存**：JVM内存的相关信息，包括堆内存使用情况、非堆内存使用情况、GC情况等。

- **操作系统信息**：操作系统的相关信息，包括平均负载情况，操作系统名称、操作系统版本、Java版本等。
- **变量信息**：变量的相关信息，包括系统变量和环境变量。

线程耗时分析

线程耗时分析支持显示该应用的所有线程和查看线程的堆栈信息，帮助您快速定位耗时较高的线程。

1. 在Arthas诊断页面，单击**线程耗时分析**页签。
   线程耗时分析页签会实时获取当前JVM进程的线程耗时情况，并将相似线程聚合。

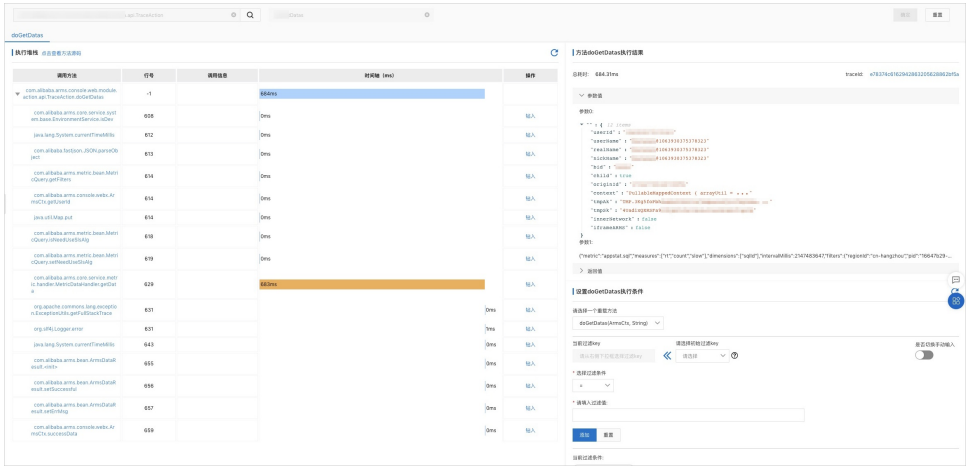


2. 单击线程左侧的+图标展开线程明细，可以查看线程的ID、CPU使用率和状态。
3. 如需查看目标线程的堆栈信息，您可以在目标线程右侧的**操作**列，单击**查看实时堆栈**。

方法执行分析

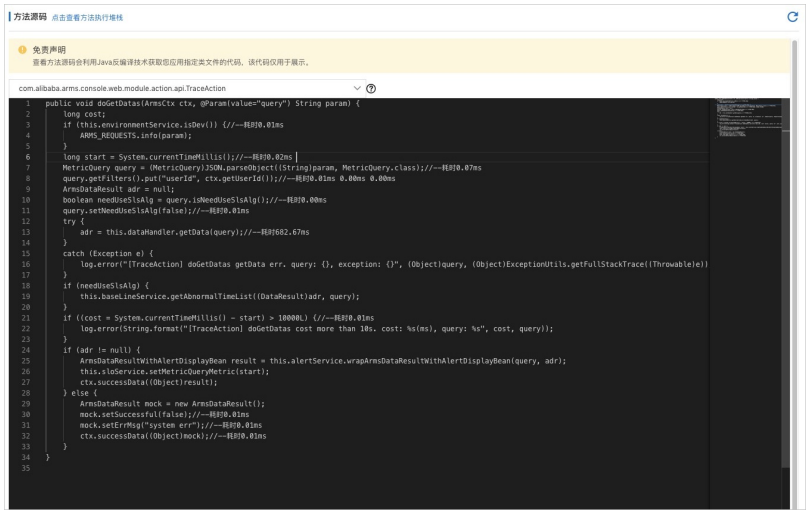
方法执行分析支持抓取方法的某一次执行的耗时、入参、返回值等信息和钻入，帮助您快速定位导致慢调用的根本原因，以及问题线下无法复现或日志缺失等场景。

1. 在Arthas诊断页面，单击**方法执行分析**页签。
2. 在**方法执行分析**页签的搜索框中输入类名的关键词，然后单击图标。
3. 在搜索到的类中选择需要诊断的类，然后在右侧方法选择框选取该类的某个方法，单击**确定**。
   页面将会显示ARMS随机抓取的该方法的某一次执行的信息。



   - 左侧**执行堆栈**区域显示诊断方法的内部执行记录。
     - 如需钻入某个内部方法，在其右侧**操作**列，单击**钻入**。

- 如需查看方法源码，单击**执行堆栈**区域顶部的**查看方法源码**。
  如下图所示，每一次内部方法的执行耗时都会以注释的方式显示在源代码中。

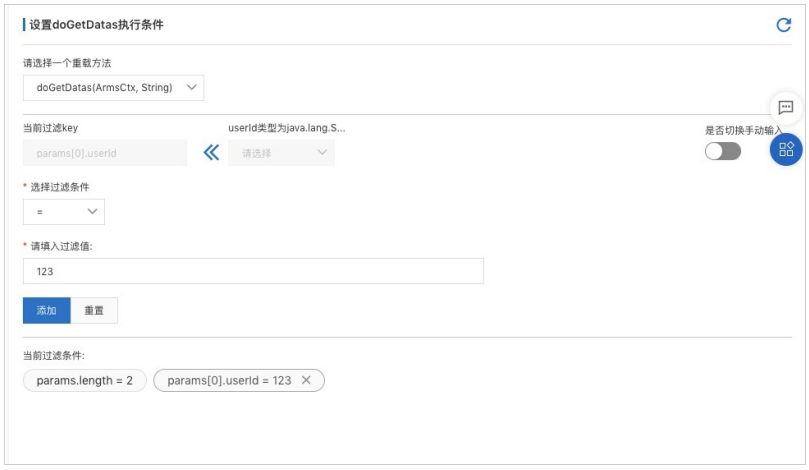


  - 右侧**方法执行结果**区域显示方法执行的参数值、返回值、异常、成员变量以及此次方法执行的TraceID。
  - 右侧**设置执行条件**区域执行以下步骤，可以设置方法执行条件来抓取满足条件的方法执行记录。
    a. 选择当前方法中的一个重载方法。
    b. 在**请选择初始过滤key**下拉框选择初始过滤Key的类型，单击<<图标。
       初始过滤Key的类型：
       - params[n]：方法的第n个参数。
       - returnObj：方法的返回值。
       - 方法执行耗时：方法执行的耗时。
       - 是否抛出异常：方法执行时是否抛出异常。

       > **说明** 如果选取的初始过滤Key为嵌套类型，则还需要继续选择该嵌套类型的内部字段，直到选择字段为基础类型。

    c. 选择过滤条件。
    d. 输入过滤值。

e. 单击**添加**。
在**当前过滤条件**区域会显示已添加的过滤条件。

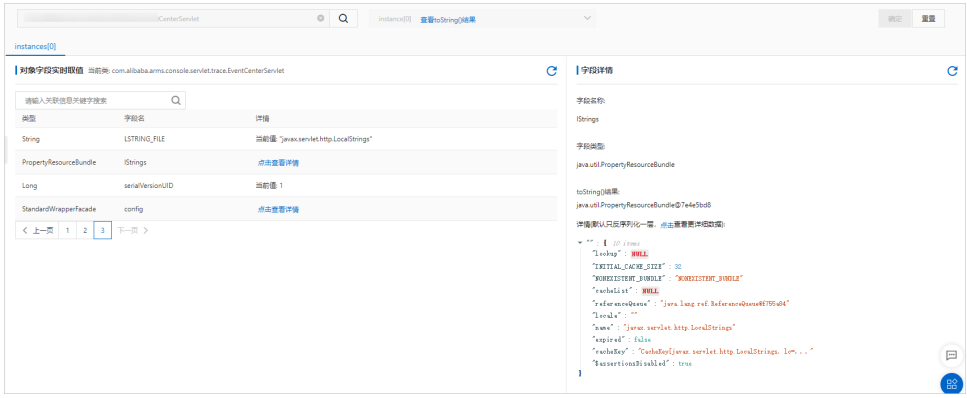


f. 在左侧**执行堆栈**区域右上角单击图标，系统会按照设置的条件重新抓取一次方法执行。

对象查看器

对象查看器用于查看一些单例对象当前的状态，用于排查应用状态异常问题，例如应用配置、黑白名单、成员变量等。

1. 在Arthas诊断页面，单击**对象查看器**页签。
2. 在**对象查看器**页签的搜索框中输入类名的关键词，然后单击**搜索**图标。
3. 在搜索到的类中选择需要诊断的类，然后在右侧实例选择框选择该类的某个实例，单击**确定**。
页面则会显示该实例中当前字段的实时取值。



- 对于简单类型的字段，在左侧**对象字段实时取值**区域的**详情**列会是字段的取值。
- 对于复制类型的字段，在左侧**对象字段实时取值**区域的**详情**列单击**点击查看详情**，在右侧**字段详情**区域查看字段取值详情。
字段详情区域仅支持将复杂类型字段反序列化一层进行展示，如果需要查看更具体的数据，单击**点击**查看字段更详细的数据。

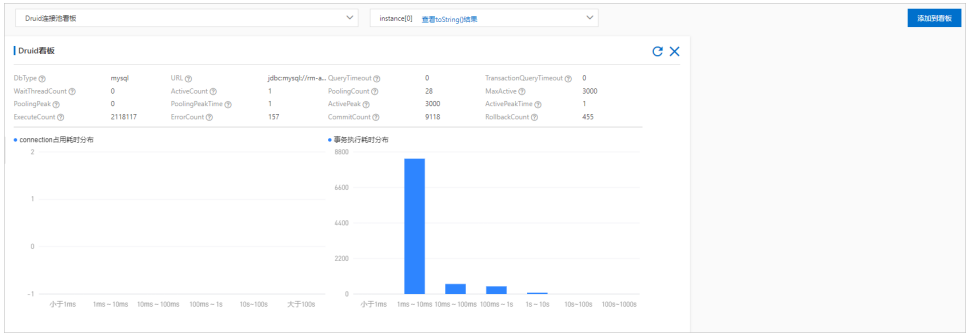
实时看板

实时看板用于查看系统中用到的关键组件的实时状态，例如查看数据库连接池的使用情况、HTTP连接池的使用情况等，有利于排查资源类型的问题。

1. 在Arthas诊断页面，单击**实时看板**页签。
2. 在**实时看板**页签的下拉列表中选择一个看板，然后在右侧实例选择框选择该看板的一个实例，单击**添**

加到看板。

页面显示该看板的实时情况。如下图显示为一个Druid连接池的实时状态信息，包括基础配置、连接池状态、执行耗时分布等。



性能分析

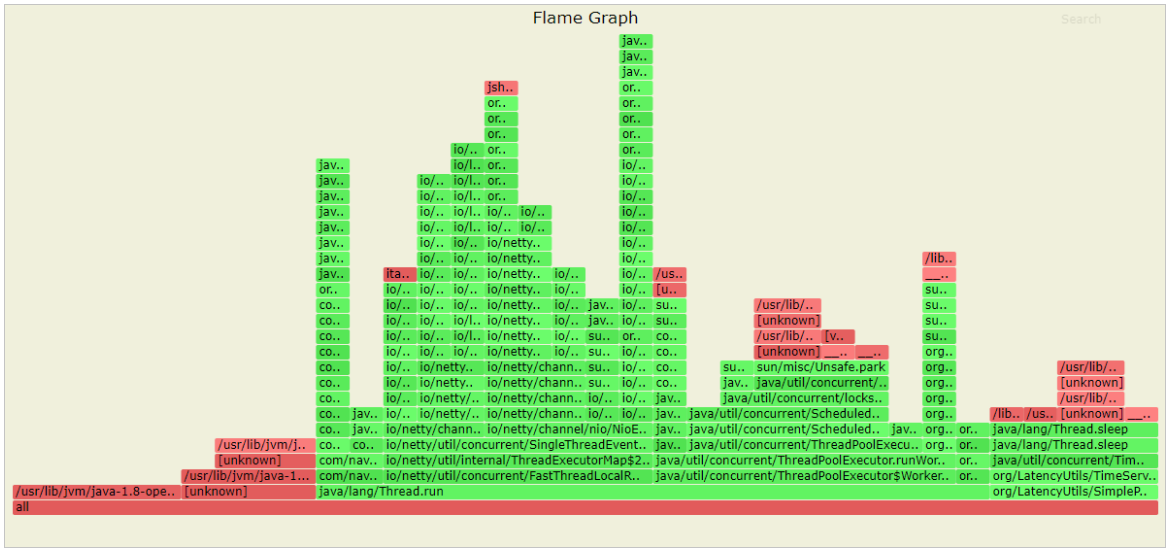
性能分析支持对CPU耗时、内存分配等对象进行一定时间的采样并生成相应的火焰图，帮助您快速定位应用的性能瓶颈。

1. 在Arthas诊断页面，单击**性能分析**页签。
2. 在**性能分析**页签下方，单击**新建火焰图**。
3. 在**新建火焰图**对话框，选择火焰图类型，输入采样时间和备注信息，然后单击**确认**。

参数	描述	示例值
火焰图类型	采样对象的类型。取值： ○ *cpu耗时* ○ *内存分配* ○ *锁耗时* ○ *itimer*	*cpu耗时*
输入采样时间（单位：秒）	采样的时长。取值：10~1800。	*30*

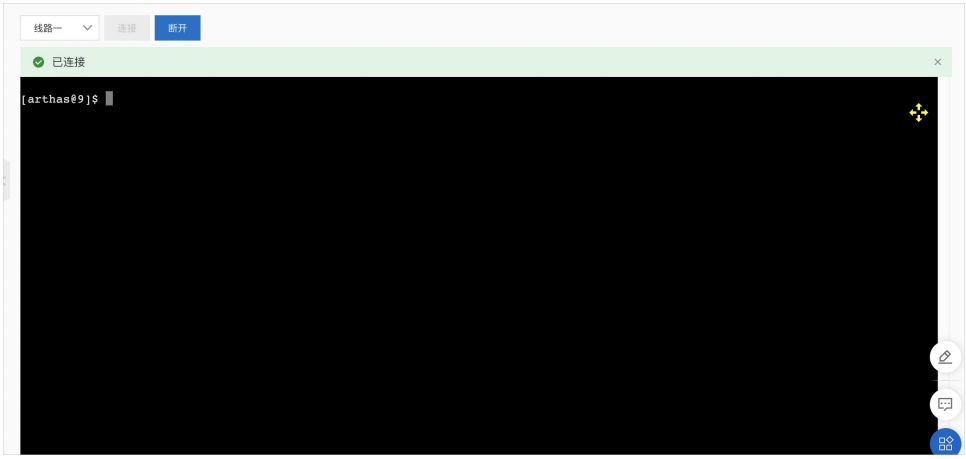
性能分析页签下方显示已创建的火焰图的任务信息，包括开始时间、采样时间、备注信息、火焰图类型和任务状态。

4. 在**性能分析**页签下方，找到任务记录，在其右侧**任务状态**，单击**查看火焰图**，根据页面提示下载SVG格式的火焰图文件，然后在浏览器中打开。



Arthas Shell

通过命令行方式自定义Arthas诊断。



3.14. 应用设置

3.14.1. 自定义配置

应用监控的一些常用设置，例如调用链采样率、Agent开关、慢SQL阈值等，可直接在**自定义配置**页签上配置。

前提条件

创建应用监控任务

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏中单击**应用设置**，并在右侧页面单击**自定义配置**页签。
5. 设置自定义配置参数，设置完毕后，在页面底部单击**保存**。

配置调用链采样设置

在**调用链采样设置**区域，可以打开或关闭调用链采样开关，并设置采样率。**采样率设置**字段输入百分比的数字部分即可，例如输入*10*代表采样10%。

> **注意** 修改即时生效，无需重启应用。如果关闭采样，则调用链数据将不会被采集，请谨慎操作。

ARMS基础版支持设置客户端采样策略，并按照采集数据行数进行计费。ARMS默认免费为您账号下所有接口的每个探针每分钟采集一条调用链。除此之外，您还可以添加自定义的采样策略。

客户端采样策略设置项

设置项	说明
策略名称	自定义采样策略名称。
采样类型和**采样值**	• **固定比例采样**：按照设置的固定比例进行调用链采样。若选择此选项，则**采样值**设置为固定比例，如*10%*。 • **流量限额**：若选择此选项，**采样值**需设置每个探针在设定时间范围内采集的调用链条数，如*5*条Trace/每个探针每*1*秒，表示每个探针每1秒钟采集5条调用链。
适用接口	设置采样策略生效的接口范围，可选择**每个接口**或**指定接口**并输入指定的接口名称。 **说明** 目前每个采样策略在选择指定接口时，仅支持输入一个接口名。若需对多个接口进行调用链采样，则需设置多个采样策略。

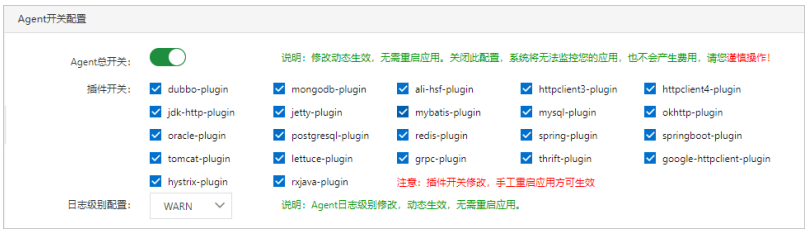
完成采样策略设置后，您可以在控制台选择是否开启该策略。多个采样策略会同时生效，但存在优先级如下：默认采样（免费）> 单接口流量限额 > 单接口固定比例采样 > 全部接口流量限额 > 全部接口固定比例采样。您也可以编辑已添加的采样策略，或删除不需要的采样策略。

ARMS基础版

配置Agent（探针）开关和日志级别

在**Agent开关配置**区域，可以打开或关闭探针总开关以及各插件开关，并配置日志级别。

> **注意** 探针总开关和日志级别的修改即时生效，无需重启应用。如果关闭探针总开关，则系统将无法监控您的应用，请谨慎操作。要使对各插件开关的修改生效，必须手动重启应用。



配置阈值设置

在**阈值设置**区域，可以设置慢SQL查询阈值、接口响应时间阈值和限流阈值。

配置高级设置

在**高级设置**区域，可以设置需过滤的接口、方法堆栈最大长度等。

- **无效接口调用过滤**：输入不需要查看调用情况的接口，从而将其从**接口调用**页面隐去。
- **方法堆栈最大长度**：默认为128条，最大值为400条。
- **同类异常堆栈区分深度**：同类型异常，用于作为不同异常区分的堆栈深度，一般设置为第一个差异调用的深度。
- **采集SQL最大长度**：默认为1024个字符，最小值为256，最大值为4096。
- **采集SQL绑定值**：捕获PrepareStatement参数绑定的变量值，无需重启应用即可生效。

- **原始SQL**：仅对SQL截断，不做额外处理。
- **异常过滤**：此处输入的异常不会显示在应用详情和异常分析页面的图表中。
- **错误数过滤**：默认情况下，大于400的状态码会计入错误数，您可以自定义大于400但不计入的HTTP状态码。
- **调用链新格式**：采用支持调用链时间排序的新存储格式（默认打开）。
- **调用链压缩**：是否将重复调用（比如for循环）简化，无需重启应用即生效。
- **请求入参最大长度**：默认为512字符，支持的最大长度为2048字符。
- **分位数统计**：是否开启分位数统计功能。
- **RabbitMQ自定义消费者**：通过设置自定义消费者的类名或者包含匿名内部消费者的类名，可以看到该消费者的调用链路。多个消费者以英文逗号（,）分隔。
- **线程池监控**：支持Tomcat/Dubbo/HSF等框架的线程池指标监控，需升级Agent探针至最新版本。该设置需重启应用后生效。
- **异步透传扫描包名**：添加异步透传扫描包实现异步任务监控。异步透传扫描包中的Runnable、Callable和Supplier接口在创建新对象时会自动捕获当前线程调用链的上下文，并在异步线程中执行时使用该调用链上下文，完成串联。探针版本必须为v2.7.1.3及以上。
- **请求中返回TraceId**：仅针对HTTP类型的请求，在Response Header中返回字段为 `eagleeye-traceid` 。

配置线程设置

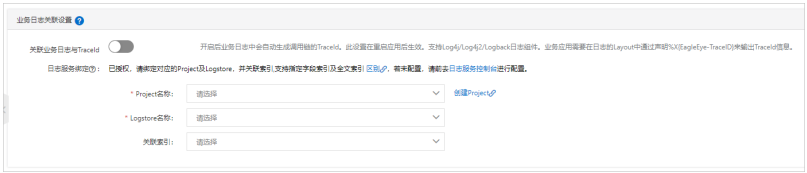
在**线程设置**区域，可以打开或关闭线程诊断方法栈开关、线程剖析总控开关，并设置慢调用监听触发阈值。

> **说明** 服务调用耗时超过慢调用监听触发阈值（默认值为1000毫秒）时才会启动监听，并一直持续到该次调用结束或超过15秒。建议将此阈值设为调用耗时的第99百分位数。假设有100次调用，则按耗时从小到大排序，排在第99位的耗时就是第99百分位数。

设置关联业务日志与TraceId

在**业务日志关联设置**区域，可以设置是否在应用的业务日志中关联调用链的TraceId信息。更多信息，请参考业务日志关联调用链的TraceId信息。

> **说明** 仅应用监控专家版支持该功能。



配置URL收敛规则

在**URL收敛设置**区域，可以打开或关闭收敛功能的开关，并设置收敛阈值、收敛规则和排查规则。URL收敛是指将具有相似性的一系列URL作为一个单独的个体展示，例如将前半部分都为/service/demo/ 的一系列URL集中展示。收敛阈值是指要进行URL收敛的最低数量条件，例如当阈值为100时，则符合规则正则表达式的URL达到100时才会对它们进行收敛。

设置业务监控

在**业务监控设置**区域，可以打开或关闭业务监控开关，并设置HTTP编码。

设置Arthas监控

在**Arthas监控**区域，可以打开或关闭Arthas诊断功能，并设置生效IP。更多信息，请参见Arthas诊断（新版）。

> **说明** 仅应用监控专家版支持该功能。

设置日志分析

在**日志分析配置**区域，可以打开或关闭日志分析功能，并设置日志采集级别和长度。更多信息，请参见日志分析（旧版）。

相关文档

- 调用链路查询
- 接口调用
- 使用线程剖析诊断代码层面的问题
- 内存快照

3.14.2. 标签管理

您可以在**应用列表**页面上管理应用标签，包括添加应用标签、通过标签筛选应用和删除应用标签。

添加标签

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面，将鼠标悬浮于目标应用右侧的 图标上，然后单击**编辑标签**。
4. 在**编辑标签**对话框可以通过以下方式为目标应用添加标签。
   - 单击**绑定**右侧的下拉框，选择已有的标签键值对添加为当前应用的标签。
   - 单击**新建标签**，设置新的标签键值对后，单击**确定**。

   > **说明** 单击标签右侧的×图标，可以为当前应用删除该标签。

5. 单击**确定**。

通过标签筛选应用

1. 在**应用列表**页面，单击**标签**下拉框。
2. 在**标签**下拉框选择需要查看的应用对应的标签键值对。

3.14.3. 监控方法自定义

除了被ARMS探针自动发现的所有方法和接口外，如果您需要监控应用中的其他方法和接口，则可在ARMS应用监控中自定义监控方法。

前提条件

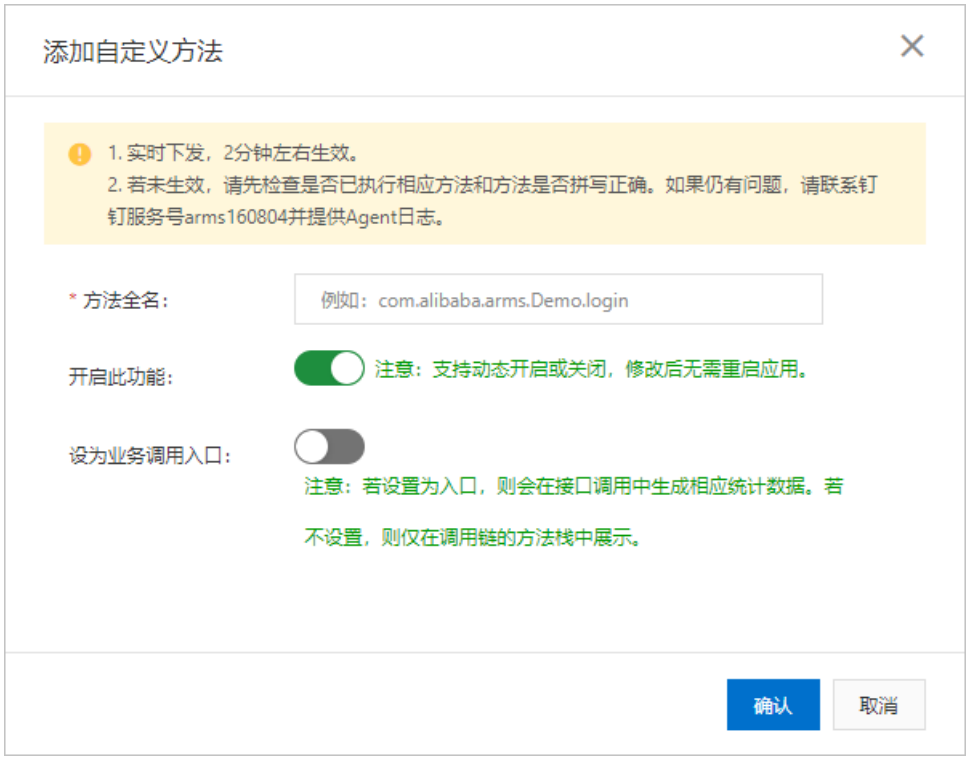
创建应用监控任务

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏中单击**应用设置**，并在右侧单击**监控方法自定义**页签。

自定义监控方法

1. 在**监控方法自定义**页签，单击右上角的**添加方法**。
2. 在**添加自定义方法**对话框中，设置以下参数，并单击**确认**。



参数	描述
方法全名	需要监控的方法的名称，具有唯一性。
开启此功能	开启此功能，即可监控此方法，并且可在本地方法栈中展示此方法，更多信息，请参见调用链路查询。默认为开启状态。 **说明** ARMS支持动态开启或关闭此功能，并且无需重启应用。
设为业务调用入口	设置为业务调用入口后，则可通过调用链对其进行业务查询，并且可在**接口调用**模块中展示对应接口，更多信息，请参见接口调用。默认为关闭状态。

自定义监控方法添加成功后，自动显示在方法列表中。

相关文档

- 调用链路查询
- 接口调用

3.14.4. 删除应用

当您不需要再使用ARMS监控您的应用，并且需要在ARMS中删除您的应用时，可以在应用设置页面彻底删除。

操作步骤

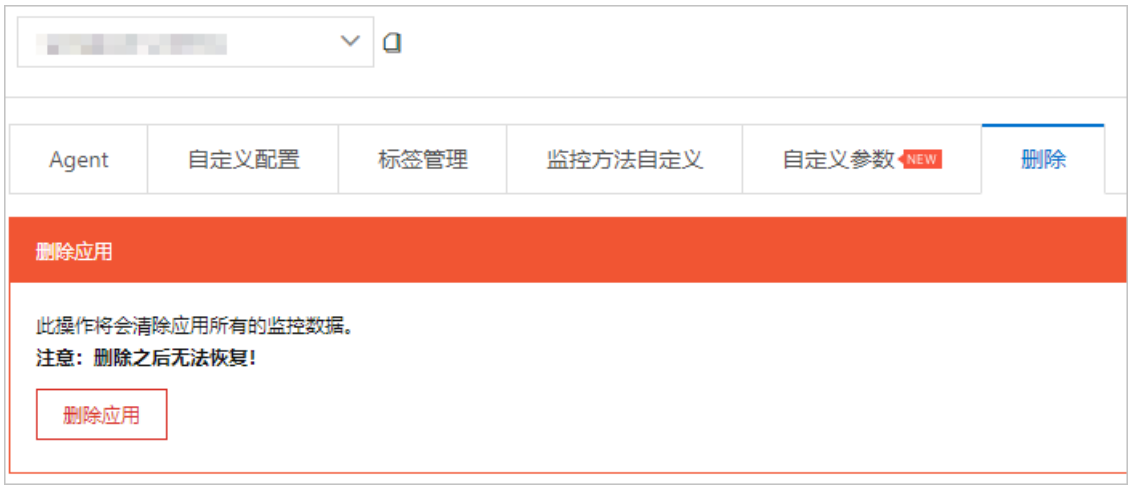
1. 登录ARMS控制台。

2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏中单击**应用设置**，并在右侧页面单击**自定义配置**页签。
5. 在**自定义配置**页签的**Agent开关配置**区域，关闭**Agent总开关**，然后单击**保存**。

> 注意 开关修改动态生效，无需重启应用。关闭此开关后，系统将无法监控您的应用，同时也不会产生费用，请您谨慎操作。

6. 卸载ARMS Agent，具体操作，请参见卸载Agent的常见问题。
7. Agent卸载完成后，在**应用设置**页面单击**删除**页签。

> 警告 此操作将会清除应用所有的监控数据，并且删除之后无法恢复。



8. 在**删除**页签单击**删除应用**，在弹出的**删除应用**对话框选择**删除原因**，单击**确定**彻底删除应用。

执行结果

返回**应用列表**页面，您可以查看到**应用列表**不再显示已删除的应用。

相关文档

- 应用监控常见问题

3.15. 应用监控告警规则（新版）

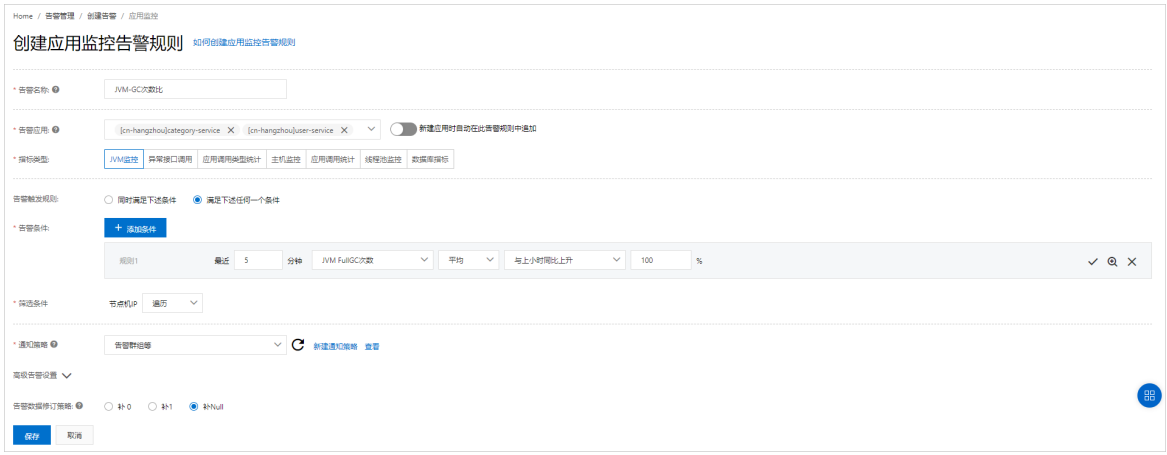
通过创建应用监控告警规则，您可以制定针对特定应用监控的告警规则。当告警规则被触发时，系统会以您指定的通知方式向告警联系人或钉群发送告警信息，以提醒您采取必要的解决措施。

前提条件

已接入应用监控，请参见应用监控接入概述。

操作步骤

1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用监控告警规则**。
3. 在**告警规则**页面的右上角单击**创建应用监控告警规则**。
4. 在**创建应用监控告警规则**页面输入所有必填信息，完成后单击**保存**。



参数	说明
告警名称	告警的名称。例如：JVM-GC次数同比告警。
告警应用	选择需要设置告警的应用。可以选择多个应用。
新建应用时自动在此告警规则中追加	是否将之后接入的应用自动接入当前告警。
指标类型	选择监控指标的类型： ○ JVM监控 ○ 异常接口调用 ○ 应用调用类型统计 ○ 主机监控 ○ 应用调用统计 ○ 线程池监控 ○ 数据库指标 **说明** 不同的指标类型，告警规则的条件字段和筛选条件不同。
告警触发规则	○ 同时满足下述规则：需满足所有告警条件才会触发告警。 ○ 满足下述一条规则：满足任意一条告警条件就会触发告警。

参数	说明
告警条件	单击**+添加条件**，设置告警规则表达式。例如：最近5分钟JVM FullGC次数平均与上小时同比上升100%。然后单击右侧图标。 **说明** ○ 单击规则右侧图标，可以修改告警规则表达式。 ○ 单击规则右侧图标，可以预览应用在当前规则下的指标走势图。 ○ 单击规则右侧图标，可以删除该告警规则表达式。 ○ 若需设置多条告警规则，单击**+添加条件**，即可编辑第二条告警规则。
筛选条件	告警指标的维度： ○ 无：告警内容中透出这个维度所有数值的和。 ○ = ：告警中只透出当前设置维度的内容。 ○ 遍历：会在告警内容中透出实际触发告警的维度内容。
通知策略	○ 不指定通知规则：告警被触发时不会发送告警，仅当通知策略的分派规则被触发时才会发送告警。 ○ 指定通知规则发送告警：告警被触发时，ARMS通过指定通知策略的通知方式发送告警信息。您可以选择已有的通知策略，也可以新建一个通知策略。更多信息，请参见通知策略。 **说明** 单击查看，可以查看选中的通知策略详情。
高级告警设置	
告警数据修订策略	用于无数据、复合指标和环比同比等异常数据的修复。当告警指标没有达到设置的条件时，告警数据补0、补1或不补充数据。 ○ 补零：将被判断的数值修复为0。 ○ 补一：将被判断的数值修复为1。 ○ 补Null：不会触发报警。 更多详细信息，请参见告警管理名词解释。

管理告警

创建的应用监控告警规则在**告警规则**页面上，您可以对告警规则执行启动、停止、编辑、删除、查看告警详情等操作。

1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用监控告警规则**。
3. （可选）在**告警规则**页面的搜索框中输入告警名称，并单击搜索图标。

   > **说明** 您可以输入告警名称的一部分内容进行模糊搜索。

4. 在搜索结果列表的**操作**列中，按需对目标告警规则采取以下操作：

   - 如需编辑告警规则，请单击**编辑**，在**编辑告警**页面中编辑告警规则，并单击**保存**。
   - 如需删除告警规则，请单击**删除**，并在**提示**对话框中单击**确认**。
   - 如需启动已停止的告警规则，请单击**启动**，并在**提示**对话框中单击**确认**。
   - 如需停止已启动的告警规则，请单击**停止**，并在**提示**对话框中单击**确认**。
   - 如需查看告警事件历史和告警发送历史，请单击**告警历史**，在**告警事件历史**和**告警发送历史**页面上查看相关记录。

相关文档

- 查看告警发送历史
- 查看告警事件历史

4.最佳实践

4.1. 使用调用链采样策略

本文介绍ARMS支持的多种调用链采样策略，帮助您根据自身场景选择合适的调用链采样策略，从而以较低成本获取想要的调用链数据。

调用链采样适用于访问量比较高的大流量应用，能够帮助您以较低的成本和性能开销记录最有价值的链路数据。调用链采样的基本原则是优先记录您最关心、最有可能访问的调用链。ARMS提供的调用链采样策略如下：

- 基于链路特征采样
- 基于业务特征采样
- 基于运维特征采样
- 基于时间特征采样

> **说明** 您可以组合使用以上调用链采样策略，以充分满足个性化的采样需求。

基于链路特征采样

基于链路特征采样是指根据调用链本身的属性进行采样，例如耗时、状态等。ARMS支持按照固定比例采样和线程剖析慢调用采样。

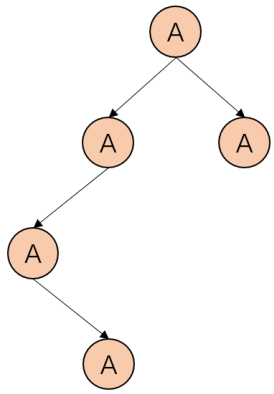
固定比例采样就是根据TraceId顺序号记录一定比例的调用链数据。例如，固定比例为10%，则每10条调用链数据记录1条。固定比例采样不会导致调用链数据本身不完整，要么保留整条链路数据，要么丢弃整条链路数据。

基于固定比例采样适用于以下场景：

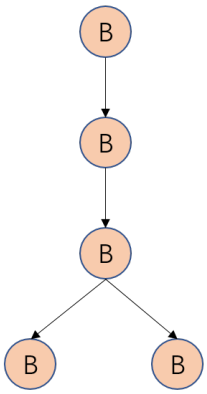
- 压测或大促峰值期间流量过大，为了避免调用链全量日志上报影响客户端性能，建议调低固定采样率比例至1%~10%。
- 日常态全量调用链日志上报导致网络带宽成本高，可以考虑按需调整固定采样率的比例。

基于固定比例采样的基本原理如下：

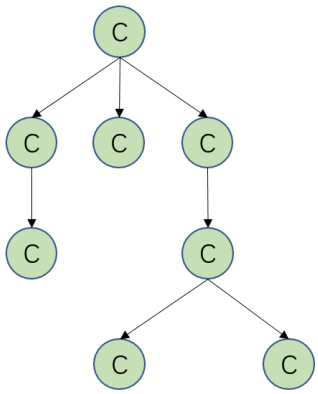
TraceId A不符合采样率全部丢弃



TraceId B不符合采样率全部丢弃



TraceId C符合采样率全部保留



设置固定比例采样的操作步骤如下：

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。

3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏中单击**应用设置**，并在右侧页面单击**自定义配置**页签。
5. 在**调用链采样设置**区域，可以打开或关闭调用链采样开关，并设置采样率。**采样率设置**字段输入百分比的数字部分即可，例如输入*10*代表采样10%。

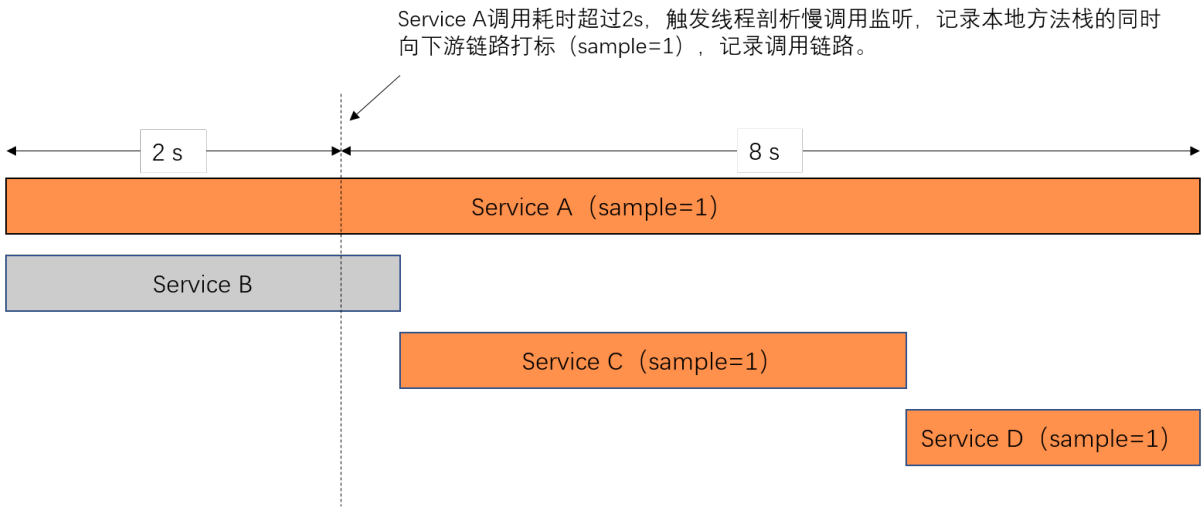
> **注意** 修改即时生效，无需重启应用。如果关闭采样，则调用链数据将不会被采集，请谨慎操作。

线程剖析慢调用采样就是记录触发线程剖析慢调用监听的调用链数据。开启线程剖析后，ARMS在记录慢调用的同时会向下传递一个采样标记，并保留该次慢调用的下游链路。不过，为了避免影响客户端性能，线程剖析监听数是有限的，如果有多个慢调用同时发生，仅会记录部分满足条件的慢调用及其下游调用链路。

线程剖析慢调用采样适用于以下场景：

- 系统出现偶发性慢调用，例如夜间服务响应耗时从0.5s突增到10s。如果提前打开线程剖析（默认触发阈值为2s），ARMS会自动记录该次请求2s至10s之间的慢调用本地方法栈，及其下游调用链路。
- 秒杀或周期性大促峰值期间系统响应慢，线程剖析会记录满足触发阈值的慢调用本地方法栈，及其下游调用链路。

线程剖析慢调用采样的基本原理如下：



设置线程剖析慢调用采样的操作步骤如下：

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏中单击**应用设置**，并在右侧页面单击**自定义配置**页签。
5. 在**线程设置**区域，可以打开或关闭线程诊断方法栈开关、线程剖析总控开关，并设置慢调用监听触发阈值。

> **说明** 服务调用耗时超过慢调用监听触发阈值（默认值为1000毫秒）时才会启动监听，并一直持续到该次调用结束或超过15秒。建议将此阈值设为调用耗时的第99百分位数。假设有100次调用，则按耗时从小到大排序，排在第99位的耗时就是第99百分位数。

固定比例采样

线程剖析慢调用采样

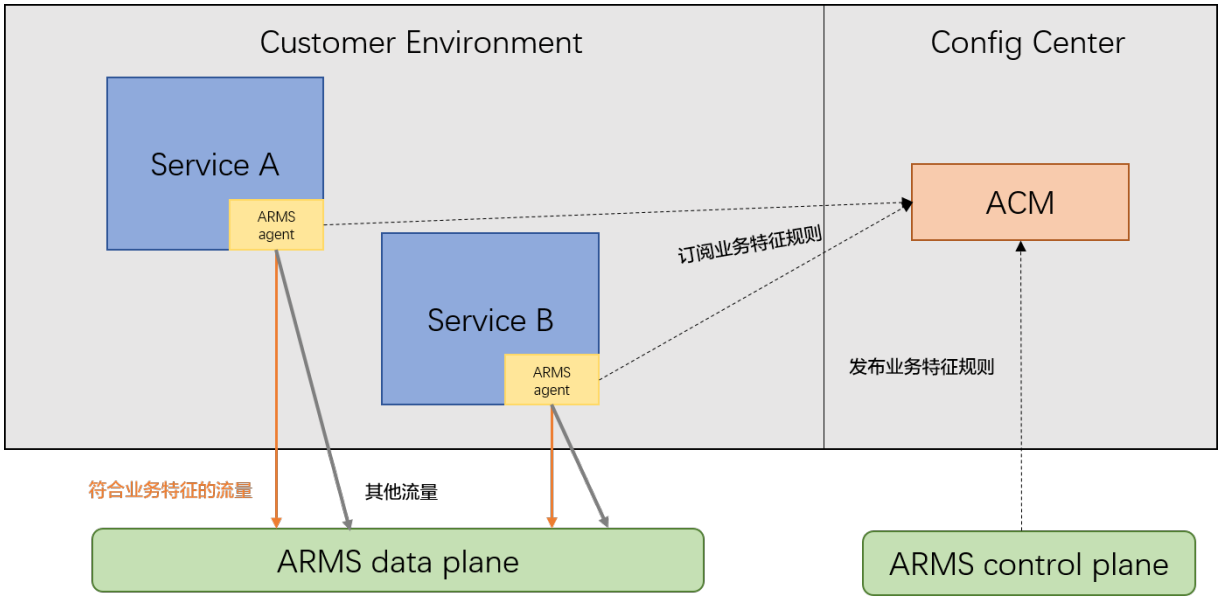
基于业务特征采样

基于业务特征采样是指根据应用的业务流量特征进行采样。ARMS支持针对入口应用的HTTP流量特征进行规则配置。业务特征提取支持针对Header/Method/Cookie/Parameter信息进行过滤筛选，满足多场景业务特征匹配条件。开启全量开关后满足业务条件的调用链数据可以优先全量采集。配置后及时生效并支持运行期动态修改。

基于业务特征采样适用于以下场景：

- 业务人员需要监控所关注的业务，但已有的监控系统无法表达业务语义。
- 应用系统包含很多业务语义，运维人员需要快速配置并监控各个业务的流量数据。
- 新业务接口上线后，接口不稳定导致出现异常或错误，开发人员需要针对此接口的每个调用分析定位问题。
- 业务负责人需要梳理某个业务的依赖情况，并基于业务依赖进行相关业务优化，保障重点业务的稳定性。

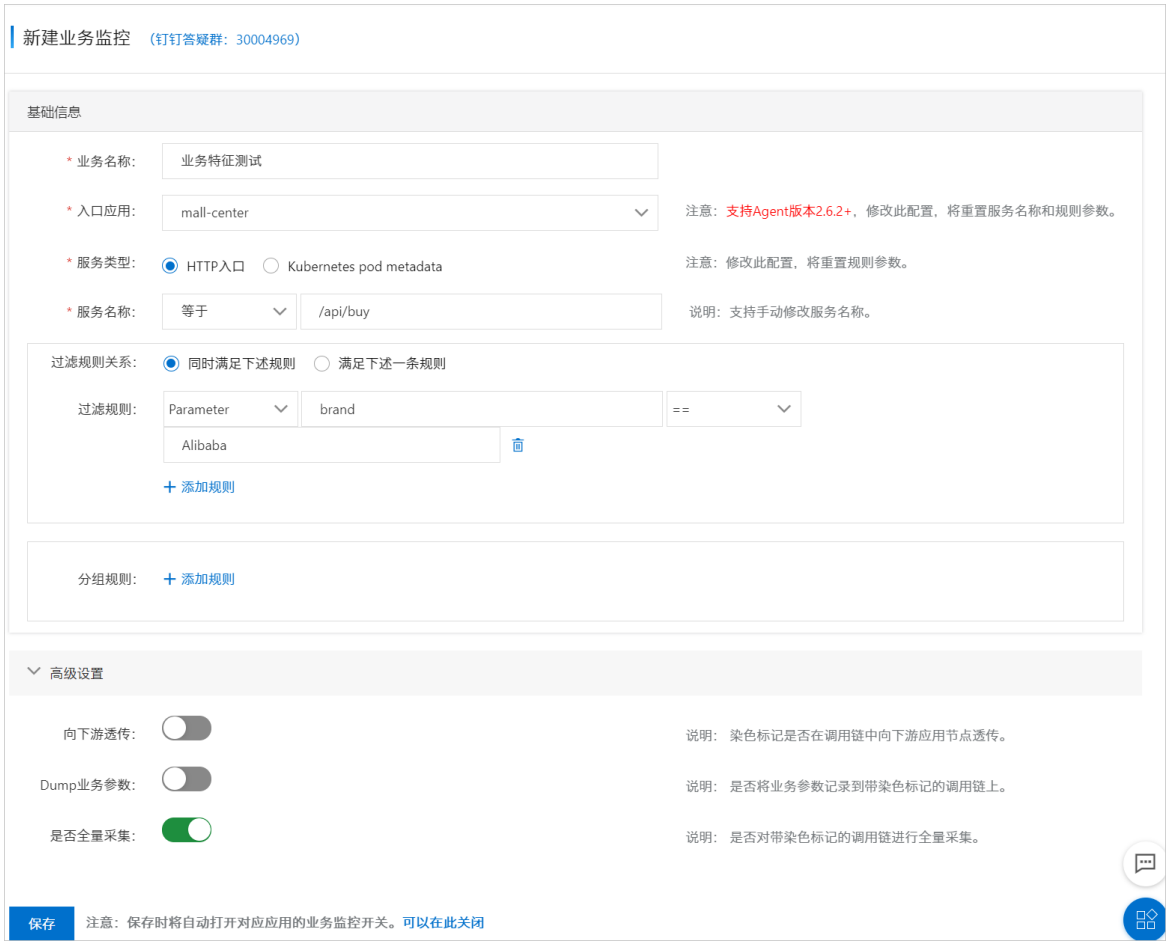
基于业务特征的基本原理如下：



设置业务特征采样的操作步骤如下：

1. 登录ARMS控制台。
2. 在左侧导航栏选择**业务监控 > 应用接入**，并在页面顶部选择目标地域。
3. 在**应用接入**页面的右上角单击**创建业务监控**。

4. 在**新建业务监控**页面的**基础信息**区域，设置以下信息，单击**高级设置**，在**高级设置**区域，设置以下信息，然后单击**保存**。



参数	描述	示例
业务名称	业务监控任务的名称，必填。	业务特征测试。
入口应用	**入口应用**列表显示所有已安装应用监控探针的Java应用，必选。选中所需应用后，ARMS自动检测其应用探针的版本号。 **说明** 应用探针升级至2.6.2+版本才能使用业务监控功能，若检测到探针版本非2.6.2+时，请您先升级探针，具体操作，请参见升级探针。	mall-center。
服务类型	设置服务类型，必选。ARMS目前仅支持**HTTP入口**，适用于按HTTP流量特征进行业务链路染色场景。	**HTTP入口**。

参数	描述	示例
服务名称	即应用提供的接口名称，必填。ARMS会根据您设置的**入口应用**自动匹配出该应用最近提供的接口列表以便您选择。如果推荐的接口不满足需求时，您也可以编辑修改。**服务名称**的匹配模式支持以下4种类型： ○ **等于**：完全匹配**服务名称**的业务接口，默认匹配模式。 ○ **开始等于**：匹配以**服务名称**为前缀的业务接口。若您需要监控具备相同前缀的业务接口时，可选择此模式。 ○ **包含**：匹配包含**服务名称**的业务接口。若您的应用提供大量业务接口时，可选择此模式快速监控您所需的业务接口。 ○ **结束等于**：匹配以**服务名称**为后缀的业务接口。以“.do”和“.action”配置结尾的典型Web框架比较适合使用此模式。 ○ **模式匹配**：匹配动态URI Path，支持Ant-style路径模式匹配规则，可实现对某一类模式的URI的监控及分析。	**等于** /api/buy。
过滤规则关系	可以选择**同时满足下述规则**或**满足下述一条规则**。	**同时满足下述规则**。
过滤规则	对配置的业务接口进一步筛选过滤，选填。**过滤规则**需要设置匹配参数（**Parameter**、**Cookie**、**Method**、**PathVariable**和**Header**）、匹配Key值、匹配方式（**==**、**!=**和**contains**）和阈值，其中，只有在**服务名称**选择**模式匹配**时，且输入的字符串中包含占位符大括号（{}），则匹配参数会出现**PathVariable**选项。支持设置多条过滤规则，多条过滤规则之间逻辑关系由您设置的**过滤规则关系**决定。	假设您的应用提供 /api/buy?brand=*** 这样的URL，您希望监控brand=Alibaba的接口调用时，可在表单中设置**过滤规则**为**Parameter** brand == Alibaba。
是否全量采集	是否对带染色标记的调用链进行全量采集。	是

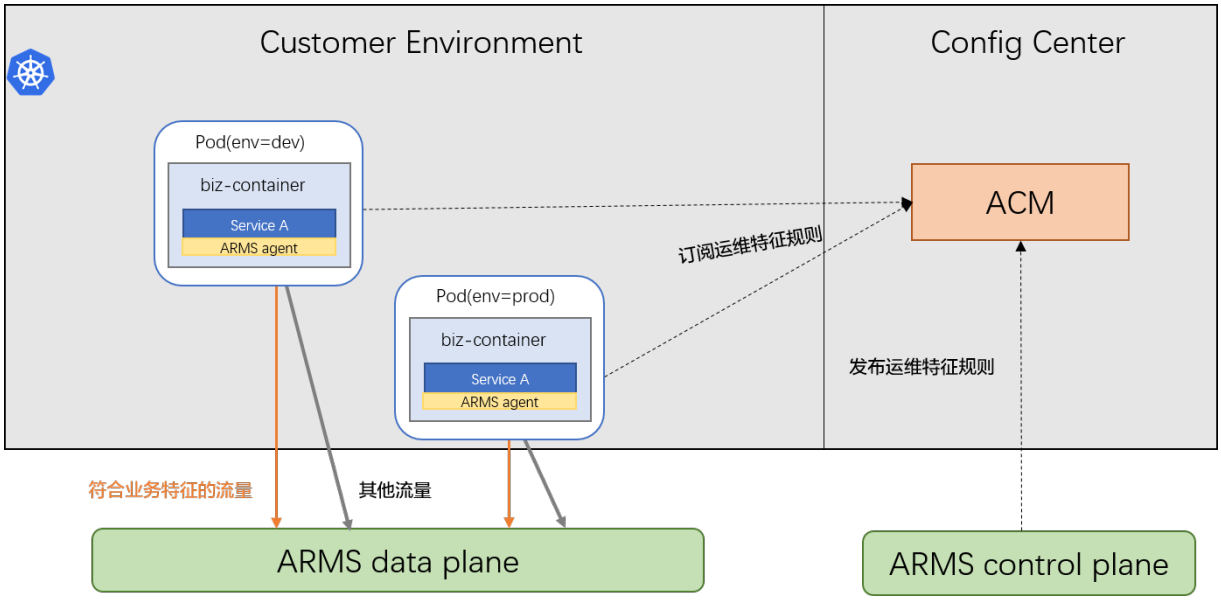
基于运维特征采样

基于运维特征采样是指根据应用的运维特征进行采样，例如部署环境、网络环境等。ARMS支持根据应用的Kubernetes Pod Metadata特征进行采样。ARMS支持针对入口应用Kubernetes Pod Metadata特征进行运维特征规则配置，能够自动识别应用Pod实例元数据信息（如label/annotation/pod name/namespace等）并进行过滤筛选，满足多场景运维特征匹配条件。开启全量开关后满足运维特征条件的调用链数据可以优先全量采集。配置后及时生效并支持运行期动态修改。

基于运维特征采样适用于以下场景：

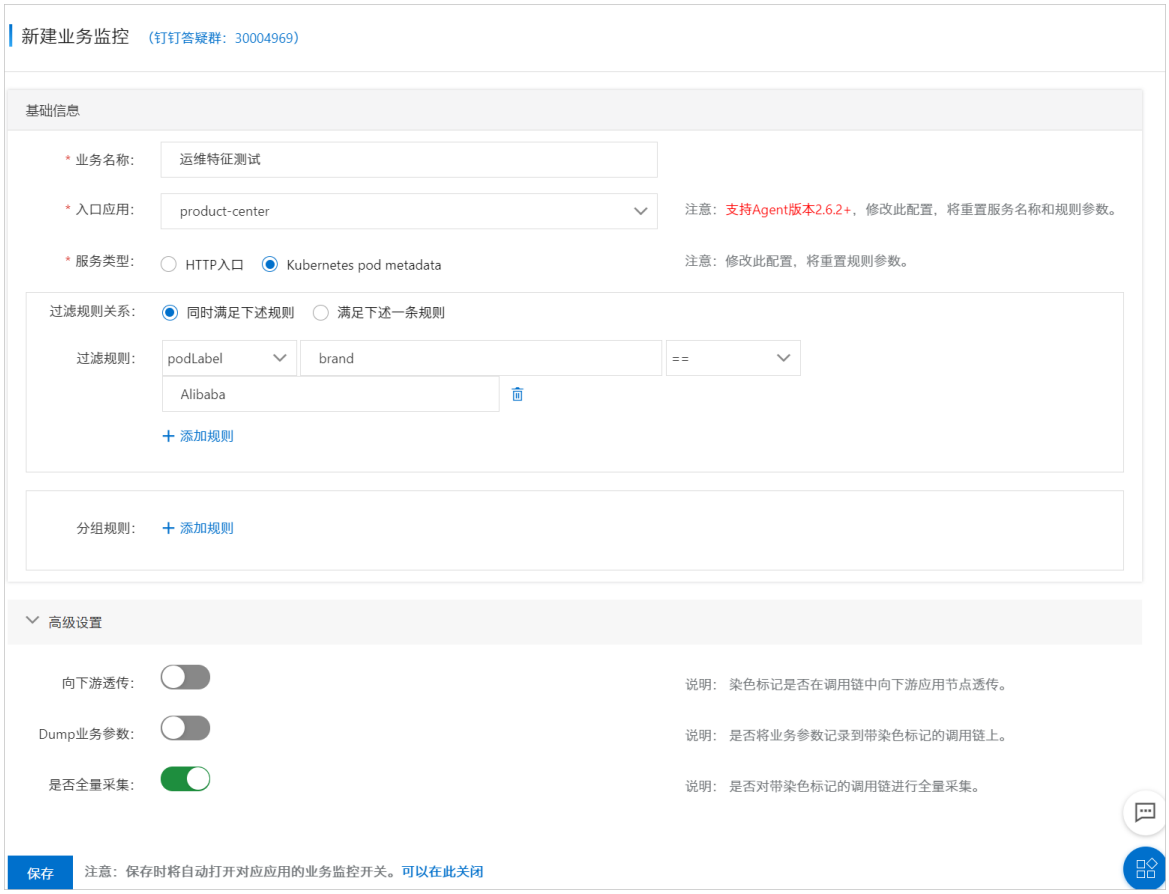
- 应用探针差异如语言、版本、环境变量、启动参数等信息需要针对性采样设置。
- 应用部署机器差异如规格、机型、区域、可用区等信息需要针对性采样设置。
- 应用部署环境差异如开发、测试、预发、生产隔离性需求需要针对性采样设置。
- 应用所处的网络环境如物理机、虚拟机、容器等部署形态差异或经典、VPC网络形式差异需要针对性采样设置。

基于运维特征采样的基本原理如下：



设置运维特征采样的操作步骤如下：

1. 登录ARMS控制台。
2. 在左侧导航栏选择**业务监控 > 应用接入**，并在页面顶部选择目标地域。
3. 在**应用接入**页面的右上角单击**创建业务监控**。
4. 在**新建业务监控**页面的**基础信息**区域，设置以下信息，单击**高级设置**，在**高级设置**区域，设置以下信息，然后单击**保存**。



参数	描述	示例
业务名称	业务监控任务的名称，必填。	运维特征测试。
入口应用	**入口应用**列表显示所有已安装应用监控探针的Java应用，必选。选中所需应用后，ARMS自动检测其应用探针的版本号。 **说明** 应用探针升级至2.6.2+版本才能使用业务监控功能，若检测到探针版本非2.6.2+时，请您先升级探针，具体操作，请参见升级探针。	product-center。
服务类型	设置服务类型，必选。ARMS目前仅支持**Kubernetes Pod Metadata**，适用于按照K8s Pod环境特征进行业务链路染色场景。	**Kubernetes Pod Metadata**。
过滤规则关系	可以选择**同时满足下述规则**或**满足下述一条规则**。	**同时满足下述规则**。

参数	描述	示例
过滤规则	对配置的业务接口进一步筛选过滤，选填。**过滤规则**需要设置匹配参数（podLabel、podAnnotation、podName、podNamespace、podUID、podIp、nodeName、hostIp和podServiceAccount）、匹配方式（==、!=和contains）和阈值。支持设置多条过滤规则，多条过滤规则之间逻辑关系由您设置的**过滤规则关系**决定。	假设您的应用在dev/test/stagging/prod等环境均有部署，通过Pod label stge标签进行区分，您希望监控dev环境并对调用链全量采集，可在表单中设置**过滤规则**为podLabel stage == dev。
是否全量采集	是否对带染色标记的调用链进行全量采集。	是

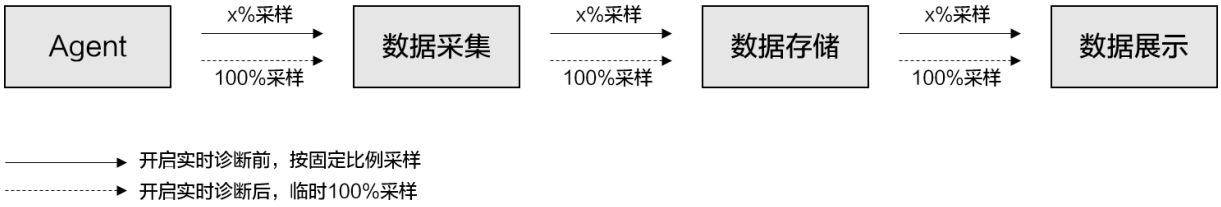
基于时间特征采样

基于时间特征采样是指根据应用的时间特征进行采样，例如在线诊断和离线分析。不同的诊断场景对于采样率的要求也不尽相同。ARMS支持在线诊断场景临时全量采集，方便您快速定位线上问题。开启实时诊断后，ARMS会持续监控应用5分钟，并在这5分钟内全量上报调用链数据。接下来，您就能以出现性能问题的调用链路为起点，通过方法栈瀑布图和线程剖析等功能定位问题原因。

基于时间特征采样适用于以下场景：

当您需要密切监控一小段时间内的应用性能时，例如发布应用或者对应用进行压测时。

基于时间特征采样的基本原理如下：



设置时间特征采样的操作步骤如下：

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面单击目标应用的名称。
4. 在左侧导航栏选择**应用诊断 > 实时诊断**。

   首次进入**实时诊断**页面时，默认自动开启实时诊断。其他情况下，如需开启实时诊断，请单击页面右上角的**开启实时诊断**。
   实时诊断将于自动开启5分钟后自动终止。如需提前终止实时诊断，请单击页面右上角的**暂停定时刷新**。

4.2. 使用线程剖析诊断代码层面的问题

ARMS线程剖析是代码级的诊断工具，能够自动捕获慢调用的堆栈快照，真实还原代码执行的第一现场。

使用场景

- 当促销活动出现慢调用时，ARMS线程剖析可为您快速定位问题代码。
- 当系统出现大量慢调用时，ARMS线程剖析可为您自动保存第一现场。
- 当业务太复杂，偶发性慢调用无法复现时，ARMS线程剖析可为您还原代码真实执行轨迹。

设置线程剖析参数

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
3. 在**应用列表**页面，单击目标应用名称。
4. 在左侧导航栏中单击**应用设置**，然后单击**自定义配置**页签。
5. 在**线程设置**区域，可以打开或关闭线程剖析总控开关，并设置慢调用监听触发阈值。

> **说明**
>
> - 服务调用耗时超过该阈值（默认值为2000ms）时才会启动监听，监听一直持续到此次调用结束但不超过15s。
> - 建议将此阈值设为调用耗时的第99分位值。假设有100次调用，则按耗时从小到大排序，排在第99位的耗时就是第99分位值。

通过接口快照查看线程剖析详情

1. 在控制台左侧导航栏选择**应用监控 > 应用列表**，并在顶部菜单栏选择目标地域。
2. 在**应用列表**页面，单击目标应用的名称。
3. 在左侧导航栏中单击**接口调用**，并在页面右侧选择目标接口，然后单击**接口快照**页签。
4. 在**接口快照**页签上单击目标TraceId链接。
   进入**调用链路**页签。
5. 在**线程剖析**列中单击放大镜图标。
   弹出**线程剖析**对话框。

> 说明
> - 实际耗时是服务调用的实际执行时间，不受线程剖析影响。
> - 监听耗时是能够被线程剖析监听到的耗时。通常情况下，监听耗时≈实际耗时-慢调用触发阈值。

通过调用链路查询查看线程剖析详情

1. 在控制台左侧导航栏中选择**应用监控 > 调用链路查询**。
2. 在**调用链路查询**页面的**参数类型**下拉列表中选择**仅含线程剖析快照**，并单击**添加到查询条件**。

3. 在搜索结果中单击目标TraceId链接。
   进入**调用链路**页签。
4. 在**线程剖析**列中单击放大镜图标。
   弹出**线程剖析**对话框。

常见问题

- 实际耗时是什么？
  答：实际耗时是服务调用的真实执行时间，不受线程剖析影响。
- 监听耗时是什么？
  答：监听耗时是指能够被线程剖析监听到的调用执行时间。为了尽量降低监听压力，线程剖析只会对每次调用超过慢调用监听触发阈值（默认为2s）后的执行时间进行监听。例如一次实际耗时为5s的慢调用，前2s不会监听，只监听3s~5s这一区间。如果一次调用耗时只有1.8s，那它将不会被监听。
- 为什么监听耗时小于实际耗时？甚至有些超过监听触发阈值的慢调用也没有被监听？
  答：
  - 由于线程剖析只监听一次调用超过触发阈值后的执行时间，因此，通常情况下，监听耗时≈实际耗时-慢调用监听触发阈值。
  - 如果系统在同一时间内出现大量慢调用，由于监听线程有限，无法保证每个慢调用在满足触发阈值的第一时间就被监听。此时，就可能出现监听耗时远小于实际耗时的情况，甚至不会被监听。
  - 为了保证超慢调用被监听，线程剖析针对5s以上的超慢调用设置了独立的监听线程，因此会出现：监听耗时≈实际耗时-5s。

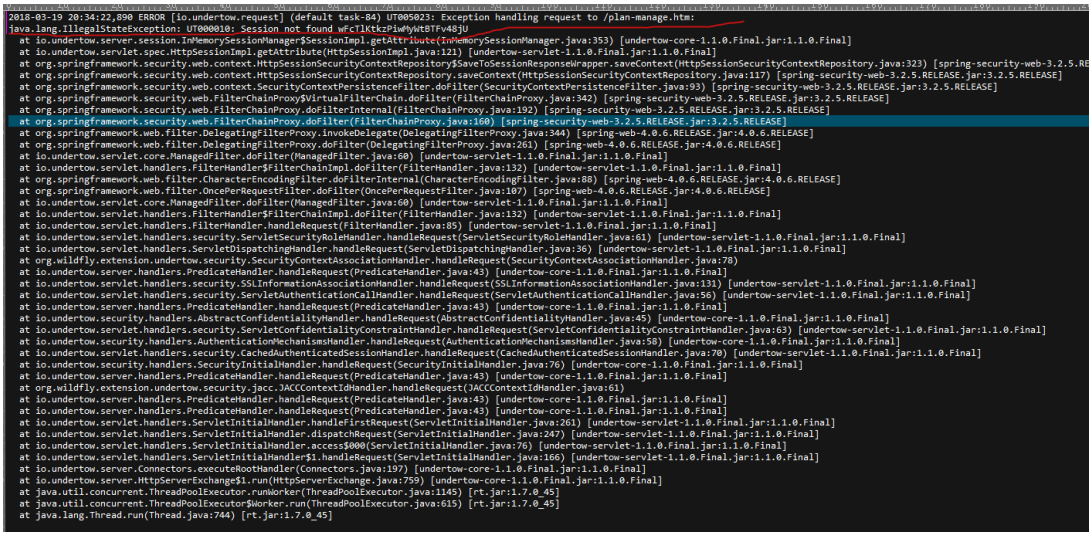
4.3. 诊断服务端报错问题

网页抛错是互联网应用最常见的问题之一，但其错因分析是一个难点。为应用安装ARMS探针后，就能在不改动应用代码的情况下，借助ARMS应用监控的异常自动捕捉、收集、统计和溯源等功能，准确定位应用中所有异常并进行线上诊断。

问题描述

网页抛错，尤其是5xx错误是互联网应用最常见的问题之一。5xx错误通常发生于服务端。服务端是业务逻辑最复杂，也是整条网络请求链路中最容易出错、出了错之后最难诊断原因的地方。运维工程师或研发工程师往往需要登录机器查看日志来定位问题。

示例：常见的Java应用错误日志



对于逻辑不太复杂、上线时间不长的应用来说，登录机器查看日志的方式能够解决大部分网站抛错的问题。但在以下场景中，传统的问题诊断方式往往没有用武之地。

- 在一个分布式应用集群中，需知道某一类错误的发生时间和频率。
- 某系统已运行了很长时间，但是不想关心遗留的异常，只想知道今天和昨天相比、发布后和发布前相比多了哪些异常。
- 查看一个异常对应的Web请求和相关参数。
- 客服人员提供了一个用户下单失败的订单号，分析该用户下单失败的原因。

解决方案

为应用安装ARMS探针后，即可在不改动应用代码的情况下，利用ARMS应用监控的异常自动捕捉、收集、统计和溯源等能力，全面掌握应用的各种错误信息。

步骤一：安装ARMS探针

为应用安装ARMS探针后，才能对应用进行全方位监控。请根据实际需求选择一种方式来安装探针。具体操作，请参见应用监控接入概述。

步骤二：查看关于应用异常的统计信息

为应用安装ARMS探针后，ARMS会收集和展示选定时间内应用的总请求量、平均响应时间、错误数、实时实例数、FullGC次数、慢SQL次数、异常次数和慢调用次数，以及这些指标和上一天的环比、上周的同比升降幅度。请按以下步骤查看应用异常的统计信息。

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域，然后单击目标应用名称。
3. 在**应用总览**页面的**概览分析**页签下方，查看异常的总数、周同比和日同比数据。

   异常次数统计

4. 滑动页面至**概览分析**页签底部的**统计分析**区域的**异常类型**，查看各类型异常出现的次数。

   各类型异常出现的次数

5. 在左侧导航栏，单击**应用详情**，然后在页面右侧单击**异常分析**页签，查看异常统计图、错误数、异常堆栈等。

   异常分析页签

步骤三：诊断异常出现的原因

掌握应用异常的统计信息还不足以诊断异常出现的原因。虽然日志中异常堆栈包含调用的代码片段，但并不包含这次调用的完整上下游信息和请求参数。ARMS探针采用了字节码增强技术，让您能够以很小的性能消耗捕获异常上下游的完整调用快照，进而找出导致异常出现的具体原因。

1. 在**异常分析**页签下，找到要诊断的异常类型，在其右侧**操作**列，单击**调用链查询**。
   调用链查询页签下显示与该异常类型相关的调用链路信息。
2. 在**调用链查询**页签下，单击某个错误调用的TraceId。

> **说明** 如何查找目标调用链路，请参见调用链路查询。

接调用链查询页签

3. 在弹出的页面，查看异常的调用链路信息，在**详情**列，单击放大镜图标，查看调用的方法栈，从而获得异常的上下文信息。

异常的完整调用链路信息

操作至此，您已发现了应用异常的原因，这将有效地帮助您进行下一步的代码优化工作。您还可以返回**调用链查询**页签，查看列表中其他异常，逐一解决。

后续操作

为避免在出现问题后被动诊断错误原因，您还可以使用ARMS的告警功能针对一个接口或全部接口创建告警，即可在出现问题的第一时间向运维团队发送通知。如何创建告警，请参见应用监控告警规则（新版）。

相关文档

- 创建应用监控任务
- 接口调用
- 调用链路查询
- 使用线程剖析诊断代码层面的问题
- 应用监控告警规则（新版）

4.4. 诊断应用卡顿问题

定位、排查应用卡顿问题的原因有诸多难点。针对这类问题，ARMS提供线程剖析、调用链路诊断、接口监控等一套解决方案，帮助您快速准确定位应用中所有慢调用，进而解决应用卡顿问题。

问题分析

网站卡顿、页面加载过慢是互联网应用最常见的问题之一。排查、解决网站卡顿、页面加载过慢等问题过程复杂，耗时较长，原因如下：

- 应用链路太长
  - 从前端页面到后台网关，从Web应用服务器到后台数据库，任何一个环节出现故障都有可能导致整体卡顿。
  - 采用微服务架构的应用，链路更加复杂，而且不同组件可能由不同的团队和人员维护，加剧了问题排查的难度。
- 日志不全或质量欠佳
  应用日志是排查线上问题的主要方法，但出现问题的位置往往无法预期，而且“慢”通常是偶发现象，要真正找到“慢”的原因，需要在每个可能出现问题的地方打印日志，记录每一次调用，但是成本太高。
- 监控不足
  业务发展过快、应用快速迭代导致应用频繁修改接口、增加依赖等情况，进而导致代码质量恶化。应用需要一个完善的监控体系来自动监控应用的每一个接口，自动记录出现问题的调用。

解决方案

为应用安装ARMS探针后，即可在不改动应用代码的情况下，使用ARMS应用监控的线程剖析、调用链路诊断、接口监控等功能，全方位监控应用中所有慢调用。

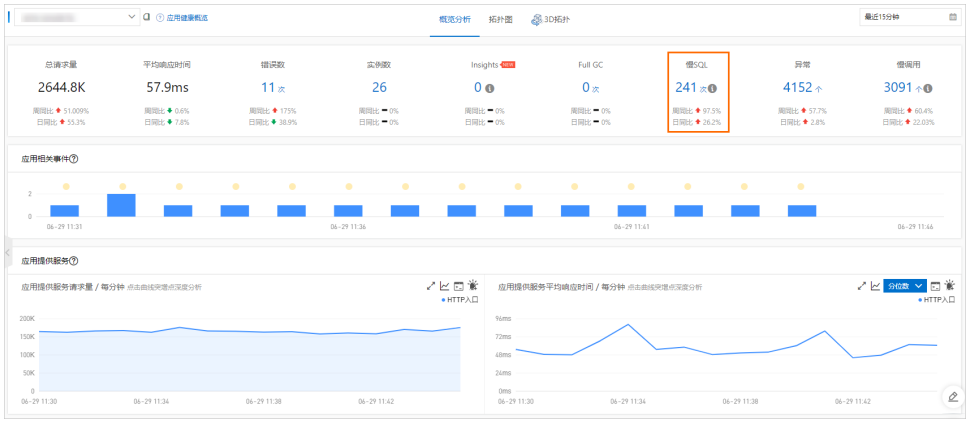
步骤一：安装ARMS探针

为应用安装ARMS探针后，才能对应用进行全方位监控。请根据实际需求选择一种方式来安装探针。具体操作，请参见应用监控接入概述。

步骤二：查看慢SQL的统计信息

为应用安装ARMS探针后，ARMS会收集和展示选定时间内应用的总请求量、平均响应时间、错误数、实时实例数、Full GC次数、慢SQL次数、异常次数和慢调用次数，以及这些指标的周同比和日同比。请按以下步骤查看慢SQL的统计信息。

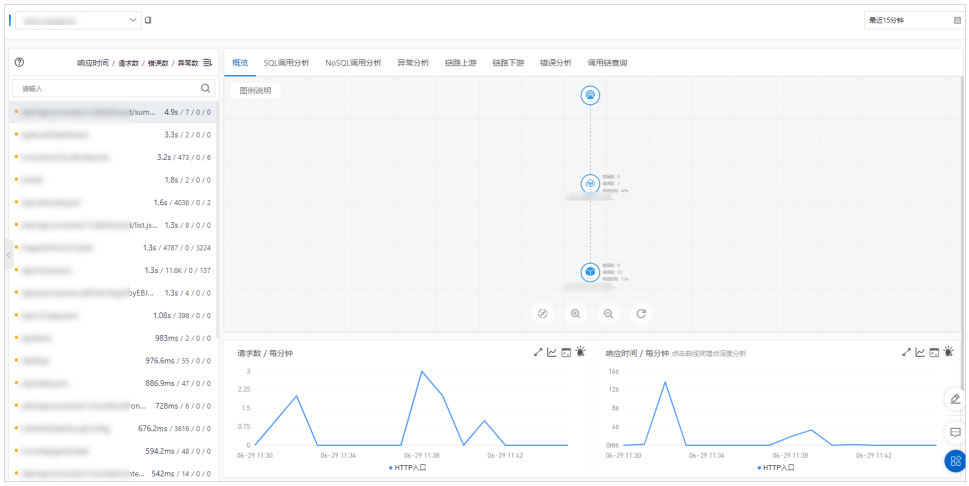
1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域，然后单击目标应用名称。
3. 在**应用总览**页面的**概览分析**页签下，查看慢SQL的总数、周同比和日环比数据。



步骤三：发现并锁定慢调用

ARMS在**接口调用**页面展示了被监控的应用提供的所有接口以及这个接口的调用次数和耗时，慢接口会被标注出来，帮助您发现和锁定慢接口。

1. 在左侧导航栏，单击**接口调用**。
2. 在**接口调用**页面的左侧，单击调用次数最多的慢接口，在右侧查看慢接口的详细信息。



步骤四：查看并锁定问题代码

锁定慢接口后，需要找到问题代码来解决问题。快照是对一次调用的全链路调用的完整记录，包括每一次调用所经过的代码及耗时，可以精准定位问题代码。

1. 在**接口调用**页面右侧，单击**调用链查询**页签。
   调用链查询页签下显示该接口的所有调用链。
2. 在**调用链查询**页签下，单击某个调用链路的TraceId。
3. 在弹出的页面，查看异常的调用链路信息，在**方法栈**列，单击放大镜图标，查看调用的方法栈，从而获得异常的上下文信息。

   > **说明** 如何查找目标调用链路，请参见调用链路查询。

   操作至此，您已发现了系统中的某个慢调用的原因，这将有效地帮助您进行下一步的代码优化工作。您还可以返回**接口调用**页面，查看列表中其他慢调用，逐一解决。

后续操作

为避免在出现问题后被动诊断错误原因，您还可以使用ARMS的告警功能针对一个接口或全部接口创建告警，即可在出现问题的第一时间向运维团队发送通知。如何创建告警，请参见应用监控告警规则（新版）。

相关文档

- 创建应用监控任务
- 接口调用
- 调用链路查询
- 使用线程剖析诊断代码层面的问题
- 应用监控告警规则（新版）

4.5. 通过诊断报告排查异常情况

定位、排查异常需要对多项指标逐一排查，过程漫长且复杂。针对此类问题，ARMS应用监控提供主动诊断功能，帮助您快速准确地定位应用中各类异常，提供诊断报告，进而解决应用响应时间过长等问题。

步骤一：安装ARMS探针

为应用安装ARMS探针后，才能对应用进行全方位监控。请根据实际需求选择一种方式来安装探针。具体操作，请参见应用监控接入概述。

步骤二：查看诊断报告

为应用安装ARMS探针后，ARMS会收集和展示选定时间内应用的总请求量、平均响应时间、错误数、实时实例数、FullGC次数、慢SQL次数、异常次数和慢调用次数等指标。ARMS支持查看包含所有排查指标的诊断报告，也可以通过**应用总览**页面查看单个指标的诊断报告。

查看包含所有指标的诊断报告

1. 登录ARMS控制台，在左侧导航栏选择**应用监控 > 应用列表**。
2. 在**应用列表**页面顶部选择目标地域。

   在**应用列表**页面，若应用存在异常，则**状态**列显示为红色。
3. 在**应用列表**页面上，将鼠标悬浮于目标应用所在行，出现**检查到异常，单击查看详情**的提示，单击红点，加载诊断报告。加载完成后，将鼠标悬浮于红点上，单击**诊断报告**，进入**诊断报告**页面。

也可以在**应用列表**页面单击目标应用名称，进入**应用总览**页面，将鼠标悬浮于**应用健康概览**右侧的图标，出现**检查到异常，单击查看详情**的提示，单击图标，加载诊断报告。加载完成后，再次单击图标进入**诊断报告**页面。

4. 在**诊断报告**页面查看诊断应用名称、诊断时间、故障现象、故障定界、根因分析和所有指标的检测结果。

   诊断报告


RT诊断报告

诊断应用：nan

诊断时间：10/10 14:06:00 ~ 10/10 15:06:00

故障现象：应用nan 提供的RT在10/10 14:46:00出现突增，突增至1672.5306 ms

故障定界

应用nan 的主机 的RT相较其他主机较高，RT高调用主要集中在调用服务 com. 上

根因分析

(1)可能的原因：应用存在慢调用，关键耗时方法栈为
inner.business.diff.impl.CommodityDiffServiceImpl.diffPrice(diff.ConfigDiffRequest request, .diff.dto.CommodityDiffConditionDTO diffCondition)
inner.business.diff.impl.CommoidityBizConfigDiffServiceImpl.diffConfig(diff.ConfigDiffRequest request, com.alibaba.fastjson.TypeReference diffRtnType)
facade.SubDomainConfigApiFacade.diffConfig diff.ConfigDiffRequest request, TypeReference diffRtnType)
.component.MethodXrayAspectInnerComponent.around(org.aspectj.lang.ProceedingJoinPoint jp)
.ProviderProcessor.handleRequest(.Invocation invocation, .io.Output output)
(2)可能的原因：应用的主机 发生FullGC，GC耗时为359.0
(3)可能的原因：应用当前jvm参数配置可优化，请将配置 -Xms4096m，-Xmx4096m，-Xmn2048m 添加到java进程启动参数中
(4)可能的原因：主机 上的线程hugePriceDetailQry-pool-*耗时较高

查看单个指标的诊断报告

1. 在**应用列表**页面单击目标应用名称，进入**应用总览**页面。
2. 在**应用总览**页面各指标的曲线图上拖动鼠标选取目标时间段，单击**查看所选时间段的诊断报告**，进入**详情**面板。

3. 在**详情**面板查看该指标所选时间段的检测结果。

后续操作

为避免在出现问题后被动诊断错误原因，您还可以使用ARMS的告警功能针对一个接口或全部接口创建告警，即可在出现问题的第一时间向运维团队发送通知。

创建告警操作步骤请参见应用监控告警规则（新版）。

相关文档

- 创建应用监控任务
- 接口调用
- 调用链路查询
- 使用线程剖析诊断代码层面的问题
- 应用监控告警规则（新版）

4.6. 业务日志关联调用链的TraceId信息

您可以在应用的业务日志中关联调用链的TraceId信息，从而在应用出现问题时，能够通过调用链的TraceId快速关联到业务日志，及时定位分析、解决问题。

前提条件

> **说明** 仅应用监控专家版支持该功能。

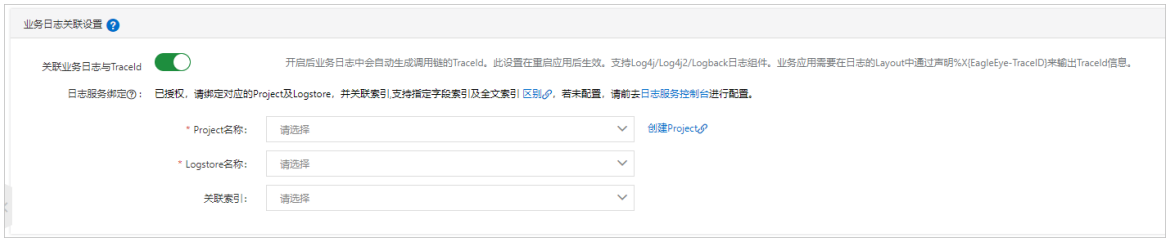
您已将Agent版本升级至2.6.1.2及以上版本，具体操作，请参见更新Java探针版本。

背景信息

ARMS在业务日志中关联调用链TraceId的功能基于MDC（Mapped Diagnostic Context）机制实现，支持主流的Log4j、Log4j2和Logback日志框架。

业务日志关联调用链的TraceId信息

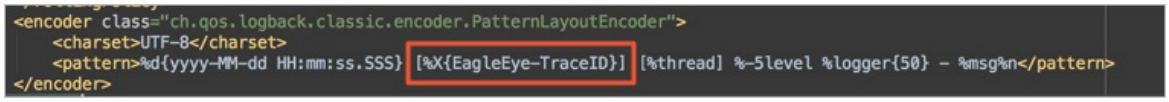
1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用列表**，在顶部菜单栏选择目标地域，然后在**应用列表**页面单击目标应用的名称。
3. 在左侧导航栏中单击**应用设置**，并在右侧单击**自定义配置**页签。
4. 在**自定义配置**页签的**业务日志关联设置**区域，打开**关联业务日志与TraceId**。



> **说明**
>
> - 开启此开关后，会在业务日志中自动生成调用链的TraceId。

5. 在您业务日志的Layout的Pattern属性中添加 `%X{EagleEye-TraceID}` 配置。以Logback组件添加此配置为例，如下图所示。

> **说明** 如何在业务代码中获取{EagleEye-TraceID}，请参见ARMS SDK使用说明。



6. 重启应用。
   在应用的业务日志中成功打印出TraceId信息，则说明业务日志关联调用链的TraceId关联成功，如下图所示。

4.7. 非阿里云环境下部署ARMS

本文介绍如何在非阿里云环境下将应用部署到ARMS。

非阿里云环境存在以下3种场景，请根据环境的实际情况选择部署ARMS的操作。

- 场景一：有专线连通到阿里云Region。
  等同于在对应Region下的正常部署，安装Java Agent的操作，请参见为Java应用手动安装Agent。
- 场景二：没有专线，机器环境可以开放部分公网。
  确认如下网络环境是否可以开放。
  ARMS端口：
  - 8442（元数据）
  - 8443（统计数据）

- 8883（明细数据）
- 8848（微服务相关数据）
- 9092（应用诊断功能）

ARMS域名：

> **说明** 此处以北京地域为例，不同地域的网络域名不同，请确认需要接入的地域对应的网络域名是否可以访问。各地域的网络域名，请参见各地域网络域名。

- 公网：arms-dc-bj.aliyuncs.com
- 内网（VPC网络）：arms-dc-bj-internal.aliyuncs.com
- 公网：acm.aliyuncs.com

链路追踪端口：

- 80（Jaeger HTTP）
- 1883（Jaeger gRPC）

链路追踪域名：

> **说明** 此处以北京地域为例，不同地域的网络域名不同，请确认需要接入的地域下的网络域名是否可以访问。各地域的网络域名，请参见各地域网络域名。

- 公网：tracing-analysis-dc-bj.aliyuncs.com
- 内网（VPC网络）：tracing-analysis-dc-bj-internal.aliyuncs.com

各地域网络域名

地域	ARMS公网	ARMS内网（VPC网络）	链路追踪公网	链路追踪内网（VPC网络）
华东1（杭州）	arms-dc-hz.aliyuncs.com	arms-dc-hz-internal.aliyuncs.com	tracing-dc-hz.aliyuncs.com	tracing-dc-hz-internal.aliyuncs.com
华北2（北京）	arms-dc-bj.aliyuncs.com	arms-dc-bj-internal.aliyuncs.com	tracing-dc-bj.aliyuncs.com	tracing-dc-bj-internal.aliyuncs.com
华东2（上海）	arms-dc-sh.aliyuncs.com	arms-dc-sh-internal.aliyuncs.com	tracing-dc-sh.aliyuncs.com	tracing-dc-sh-internal.aliyuncs.com
华北1（青岛）	arms-dc-qd.aliyuncs.com	arms-dc-qd-internal.aliyuncs.com	tracing-dc-qd.aliyuncs.com	tracing-dc-qd-internal.aliyuncs.com
华南1（深圳）	arms-dc-sz.aliyuncs.com	arms-dc-sz-internal.aliyuncs.com	tracing-dc-sz.aliyuncs.com	tracing-dc-sz-internal.aliyuncs.com

地域	ARMS公网	ARMS内网（VPC网络）	链路追踪公网	链路追踪内网（VPC网络）
华北3（张家口）	arms-dc-zb.aliyuncs.com	arms-dc-zb-internal.aliyuncs.com	tracing-dc-zb.aliyuncs.com	tracing-dc-zb-internal.aliyuncs.com
中国（香港）	arms-dc-hk.aliyuncs.com	arms-dc-hk-internal.aliyuncs.com	tracing-dc-hk.aliyuncs.com	tracing-dc-hk-internal.aliyuncs.com
亚太东南（新加坡）	arms-dc-sg.aliyuncs.com	arms-dc-sg-internal.aliyuncs.com	tracing-dc-sg.aliyuncs.com	tracing-dc-sg-internal.aliyuncs.com
政务云	arms-dc-gov.aliyuncs.com	arms-dc-gov-internal.aliyuncs.com	tracing-dc-gov.aliyuncs.com	tracing-dc-gov-internal.aliyuncs.com
华东1金融云（杭州）	arms-dc-hz-finance.aliyuncs.com	arms-dc-hz-finance-internal.aliyuncs.com	tracing-dc-hz-finance.aliyuncs.com	tracing-dc-hz-finance-internal.aliyuncs.com

> **说明** 仅以上地域的网络环境支持通过此方式接入应用。

如果上述网络环境可以开放，参考以下文档安装Java Agent：

- 非Kubernetes集群下的应用：为Java应用手动安装Agent
- Kubernetes集群下的应用：为开源Kubernetes环境中的应用安装探针

- 场景三：没有专线，大部分机器环境不可以开放公网。
  在堡垒机上开放以上网络端口和对应Region的网络域名后，通过Gateway组件上传数据到ARMS应用监控中。具体操作，请参见在用户专有网络中部署ARMS。

4.8. 在用户专有网络中部署ARMS

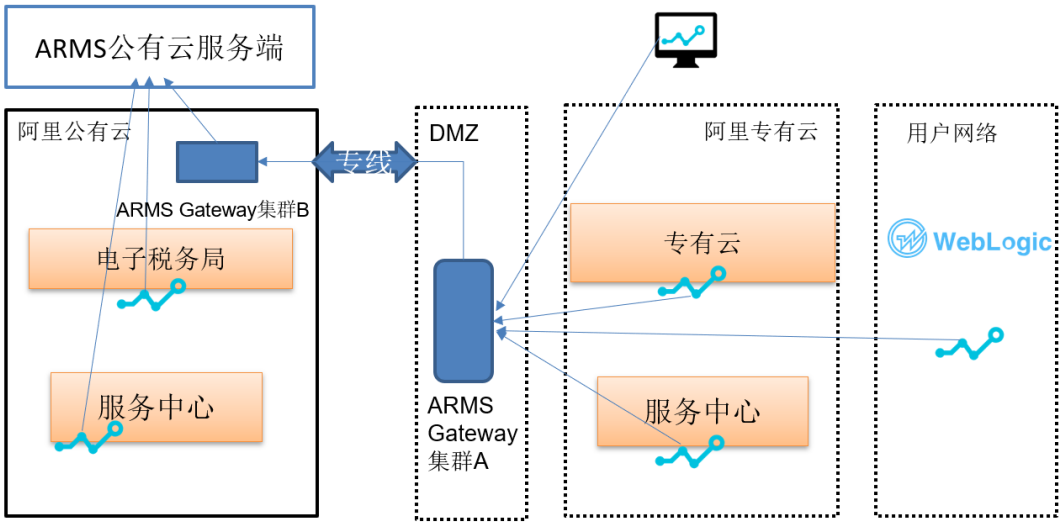
本文介绍如何通过Gateway组件将用户网络中的应用接入公有云ARMS应用监控服务中。

使用场景

Gateway组件主要解决混合云场景下，由于网络不通原因，导致应用无法接入公有云ARMS应用监控服务问题，将其部署在阿里云环境（经典网络或VPC网络）充当代理角色。

部署原理

ARMS部署图



1. 部署在DMZ中的ARMS Gateway集群A对阿里云专有云和用户网络暴露内网地址，部署在专有云中的ARMS探针和WebLogic上的ARMS探针把采集的监控数据发送到Gateway上。
2. DMZ通过专线和阿里云公有云打通，部署在公有云VPC内的ARMS Gateway集群B通过专线连接Gateway集群A，起桥接作用。
3. ARMS Gateway集群B把收集到的ARMS监控数据发送到公有云服务端。
4. 阿里云公有云上的ARMS探针，包括部署在页面上的前端探针，将数据发送到ARMS公有云服务端。

ARMS混合云监控部署要求

- 按照采集专有云和用户网络500个节点来规划，需要6台规格为2核8 G的虚拟机，部署2个包含3台Gateway组件的集群。
- Gateway向阿里公有云传输的监控数据量比较小，可以控制在1 M/s内，所以利用现有的专线带宽，专有云EDAS和WebLogic都可以通过安装ARMS探针开启应用监控，无需修改代码。

安装ARMS

1. 下载Gateway软件包，此处以杭州地域为例。
2. 将Gateway部署在代理机器上，通过如下命令启动。

```
java -jar arms-gateway-1.7.0.jar
```

> **说明** JDK版本需要为JDK 1.7+。

Gateway默认连接杭州地域的ARMS服务，如需连接其它地域，可通过-D参数修改地域域名。连接到北京地域的示例命令如下：

```
java -jar -Darms.server.endpoint=arms-dc-bj.aliyuncs.com arms-gateway-1.7.0.jar
```

地域	域名
华东1（杭州）	arms-dc-hz.aliyuncs.com
华东2（上海）	arms-dc-sh.aliyuncs.com

地域	域名
华北1（青岛）	arms-dc-qd.aliyuncs.com
华北2（北京）	arms-dc-bj.aliyuncs.com
华北3（张家口）	arms-dc-zb.aliyuncs.com
华北5（呼和浩特）	dc-cn-huhehaote.arms.aliyuncs.com
华北6（乌兰察布）	dc-cn-wulanchabu.arms.aliyuncs.com
华南1（深圳）	arms-dc-sz.aliyuncs.com
华南2（河源）	dc-cn-heyuan.arms.aliyuncs.com
华南3（广州）	dc-cn-guangzhou.arms.aliyuncs.com
西南1（成都）	dc-cn-chengdu.arms.aliyuncs.com
中国（香港）	arms-dc-hk.aliyuncs.com
亚太东南1（新加坡）	arms-dc-sg.aliyuncs.com
亚太东南2（悉尼）	dc-ap-southeast-2.arms.aliyuncs.com
亚太东南3（吉隆坡）	dc-ap-southeast-3.arms.aliyuncs.com
亚太东南5（雅加达）	arms-dc-indonesia.aliyuncs.com
亚太东北1（东京）	arms-dc-jp.aliyuncs.com
欧洲中部1（法兰克福）	arms-dc-frankfurt.aliyuncs.com
欧洲西部1（伦敦）	dc-eu-west-1.arms.aliyuncs.com
美国东部1（弗吉尼亚）	dc-us-east-1.arms.aliyuncs.com
美国西部1（硅谷）	arms-dc-usw.aliyuncs.com
亚太南部1（孟买）	dc-ap-south-1.arms.aliyuncs.com
政务云	arms-dc-gov.aliyuncs.com
杭州金融云	arms-dc-hz-finance.aliyuncs.com
上海金融云	arms-dc-sh-finance.aliyuncs.com
深圳金融云	arms-dc-sz-finance.aliyuncs.com

3. 下载Agent。
   可以在ARMS控制台的**接入中心**页面下载，具体操作，请参见为Java应用手动安装Agent。
4. 解压Agent安装包。

5. 在arms-agent.config文件中，修改profiler.collector.ip指定的服务地址为代理机器地址。

```
profiler.collector.ip={代理机器IP}
```

6. 启动应用，在ARMS控制台的**应用监控 > 应用列表**页面查看监控数据是否正常上报，如果有数据上报，则说明接入成功。

4.9. 通过调用链路和日志分析定位业务异常问题

定位业务异常问题难度大、效率低，一直是ARMS应用监控的性能瓶颈。ARMS应用监控通过结合调用链路和日志分析，可以快速、准确地定位业务异常问题，提升微服务框架下的开发诊断效率。

前提条件

- 已开通日志服务SLS。登录日志服务控制台时，根据页面提示开通日志服务。
- 已创建Project，详情请参见创建Project。
- 已创建Logstore，详情请参见创建Logstore。

背景信息

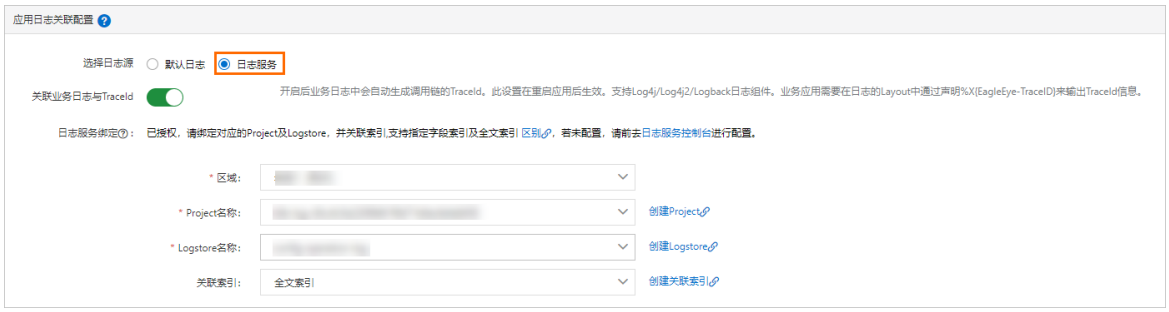
在使用调用链路和日志分析定位业务异常问题前，需要先了解Metrics、Tracing和Logging三个概念。

- Metrics：应用的关键性能指标，如应用提供服务请求量、应用提供服务平均响应时间、应用依赖服务请求量等。
- Tracing：调用链路，应用的任何接口调用、请求响应等动作都会绑定到完整的链路。
- Logging：业务日志，应用的任何接口调用、请求响应等动作都会输出完整的业务日志。

当应用出现业务异常问题时，应用指标统计图会出现明显波动，您可据此粗略地分析异常问题；通过完整的调用链路和业务日志分析，可以精准定位业务异常问题。

关联业务日志与TraceId

1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用列表**，在顶部菜单栏选择目标地域，然后在**应用列表**页面单击目标应用的名称。
3. 在左侧导航栏中单击**应用设置**，并在右侧单击**自定义配置**页签。
4. 在**自定义配置**页签的**业务日志关联设置**区域，打开**关联业务日志与TraceId**开关，然后绑定Project和Logstore。



5. 在**自定义配置**页签左下角单击**保存**。

从应用指标的角度排查业务异常问题

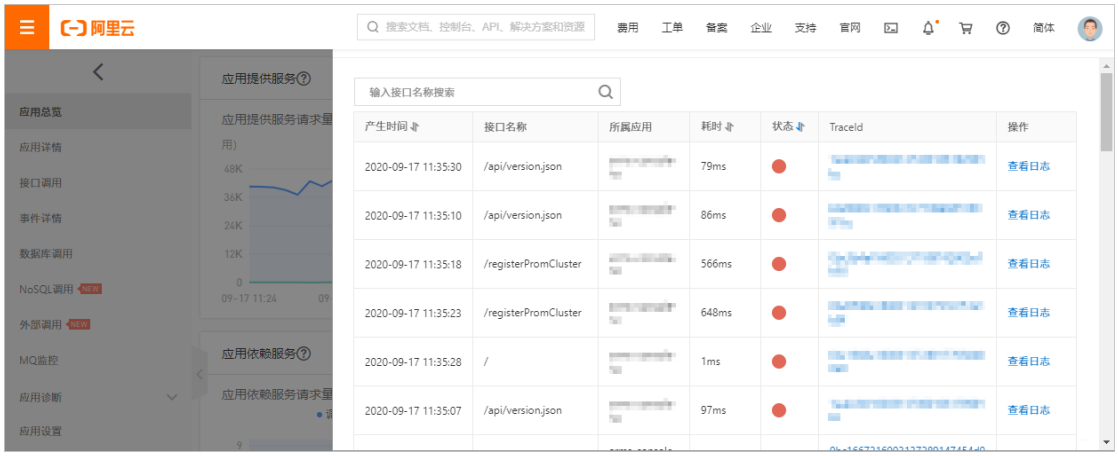
1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用列表**，在顶部菜单栏选择目标地域，然后在**应用列表**页面单击目标应用的名称。
3. 在左侧导航栏单击**应用总览**，在顶部选择**概览分析**，然后在右上角选择或自定义设置目标时间段。

   概览分析页面展示目标应用的关键指标，如**应用提供服务请求量**、**应用提供服务平均响应时间**、**应用依赖服务请求量**等。
4. 在**概览分析**页面选择某个应用指标，在该指标的曲线图上拖动鼠标选取目标时间段。

   本示例以**应用提供服务平均响应时间**指标为例。

5. 查看步骤4所选时间段的调用链路。
   i. 单击**查看所选时间段的调用链路**。
   ii. 在调用链路列表面板选择**状态**异常（显示为）的调用链路记录，单击该调用链路记录TraceId列下的TraceId值。

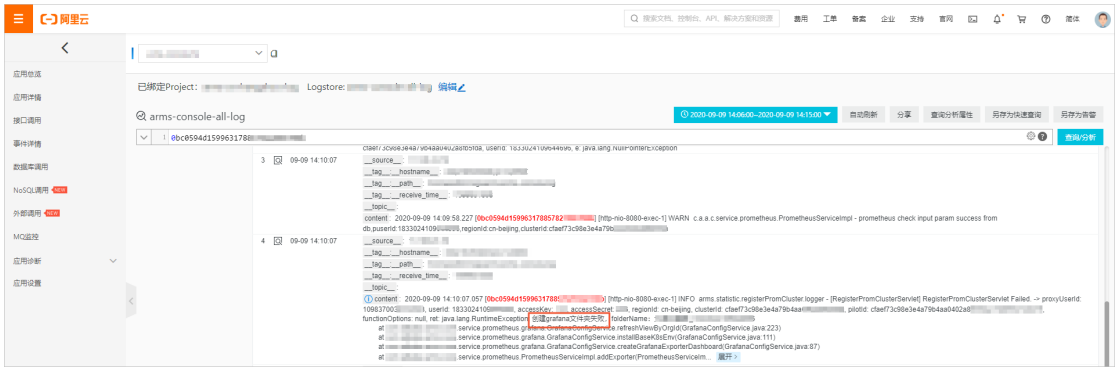
   您也可以在该调用链路记录**操作**列下单击**查看日志**，查看该时间点的业务日志，分析业务异常原因。



   iii. 单击**调用链路**页签，然后在**方法栈**列下单击图标。

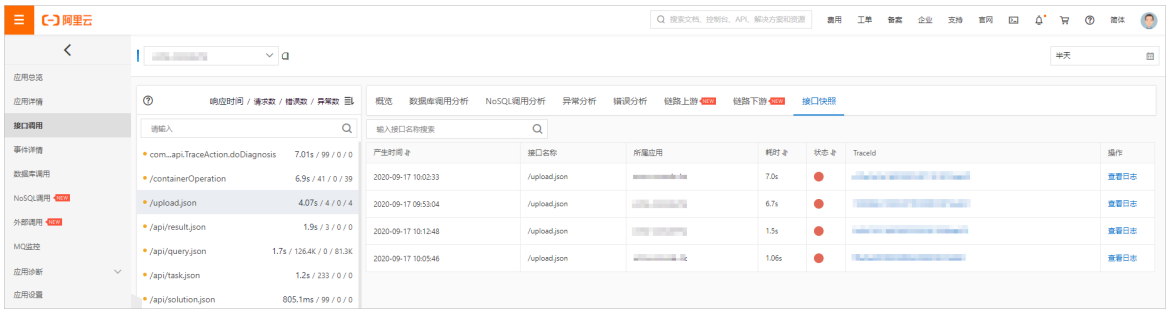
iv. 在链路详情信息页面查找错误信息，鼠标悬停在错误信息上可查看异常原因。

6. 查看步骤4所选时间段的业务日志。
   i. 单击**查看所选时间段的日志**。
   ii. 在**日志分析**页面选择异常错误信息，查看日志并定位业务异常原因。



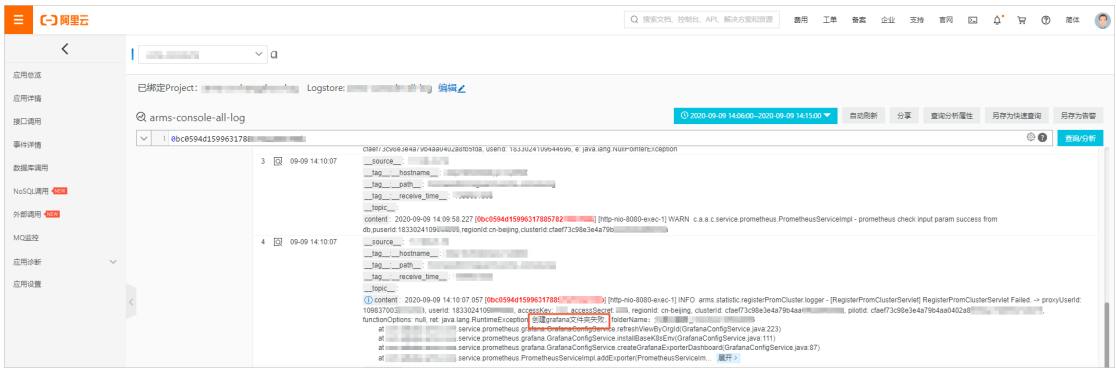
从接口调用的角度排查业务异常问题

1. 登录ARMS控制台。
2. 在左侧导航栏中选择**应用监控 > 应用列表**，在顶部菜单栏选择目标地域，然后在**应用列表**页面单击目标应用的名称。
3. 在左侧导航栏单击**接口调用**。
4. 在**接口调用**页面的接口列表区域单击目标接口，然后单击右侧的**接口快照**页签。
5. 在**接口快照**页签选择**状态**异常的接口调用记录，异常状态显示为。



6. 查看接口调用的调用链路。
   i. 在目标接口调用记录的**TraceId**列下单击TraceId的值。
   ii. 单击**调用链路**页签，然后在**方法栈**列下单击图标。
   iii. 在链路详情信息页面查找错误信息，鼠标悬停在错误信息上可查看异常原因。

7. 查看接口调用的日志。
   i. 在目标接口调用记录的**操作**列下单击**查看日志**。
   ii. 在**日志分析**页面选择异常错误信息，查看日志并定位业务异常原因。



4.10. 将ARMS页面嵌入自建Web应用

如需在自建Web应用中免登录查看ARMS控制台的页面，您可将ARMS控制台嵌入自建Web应用，以此避免系统间的来回切换。

背景信息

将ARMS控制台嵌入自建Web应用可以实现以下效果：

- 可登录自有系统并浏览嵌入的ARMS控制台的应用列表、应用详情、调用查询等页面。
- 可隐藏ARMS控制台的顶部菜单栏和左侧导航栏。具体操作，请参见隐藏导航栏。
- 可通过访问控制RAM控制操作权限，例如将完整权限改为只读权限。具体操作，请参见借助RAM用户实现分权。

示例代码

如需快速体验将ARMS页面嵌入自建Web应用的效果，您可以下载并使用示例代码。

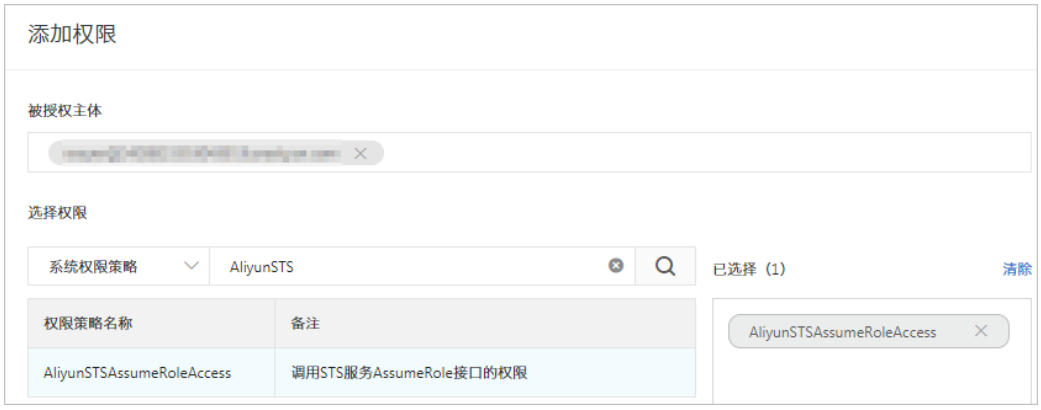
步骤一：创建RAM用户并添加权限

使用阿里云账号创建RAM用户并为其添加调用STS服务扮演RAM角色的权限。

1. 登录RAM控制台。
2. 在左侧导航栏，选择**人员管理 > 用户**。
3. 在**用户**页面，单击**创建用户**。
4. 在**创建用户**页面的**用户账号信息**区域，输入**登录名称**和**显示名称**，在**访问方式**区域，选中**编程访问**，然后单击**确定**。

   > 注意 RAM会自动为RAM用户创建AccessKey（API访问密钥）。出于安全考虑，RAM控制台只提供一次查看或下载AccessKeySecret的机会，即创建AccessKey时，因此请务必将AccessKeySecret记录到安全的地方。

5. 在**手机验证**对话框，单击**获取验证码**，输入收到的手机验证码，然后单击**确定**。
6. 在**用户**页面，找到创建好的用户，单击**操作**列的**添加权限**。
7. 在**添加权限**面板的**选择权限**区域，通过关键字搜索需要添加的权限策略AliyunSTSAssumeRoleAccess，并单击权限策略将其添加至右侧的**已选择**列表，然后单击**确定**。



8. 在**添加权限**的**授权结果**页面，查看授权信息摘要，然后单击**完成**。

步骤二：创建RAM角色并添加权限

创建RAM角色并为其添加访问控制台的权限。RAM用户将会扮演该RAM角色访问控制台。

1. 登录RAM控制台。
2. 在左侧导航栏，单击**RAM角色管理**。
3. 在**RAM角色管理**页面，单击**创建RAM角色**。
4. 在**创建RAM角色**面板，执行以下操作：
   i. 在**选择类型**页面的**当前可信实体类型**区域，选择**阿里云账号**，然后单击**下一步**。
   ii. 在**配置角色**页面的**角色名称**文本框，输入RAM角色名称，然后单击**完成**。
   iii. 在**创建完成**页面，单击**为角色授权**。
5. 在**添加权限**面板的**选择权限**区域，搜索需要添加的权限策略，单击权限策略将其添加至右侧的**已选择**列表，然后单击**确定**。

   您可以根据需要为RAM角色赋予以下ARMS权限：
   - AliyunARMSFullAccess：ARMS的完整权限。
   - AliyunARMSReadOnlyAccess：ARMS的只读权限。
6. 在**添加权限**的**授权结果**页面，查看授权信息摘要，然后单击**完成**。

步骤三：获取临时AccessKey和Token

登录自建Web，然后在Web服务端调用STS的AssumeRole接口获取临时AccessKey和Token，即临时身份。AssumeRole接口详情，请参见AssumeRole。

您可以选择以下方式调用AssumeRole接口：

- 通过OpenAPI开发者门户调用。
- 通过Java SDK调用。

本文以通过Java SDK调用为例。

您在使用Java SDK时需要注意填写以下参数：

```
String accessKey = "<accessKeyId>"; //RAM用户的AccessKeyId。
String accessSecret = "<accessKeySecret>"; //RAM用户的AccessKeySecret。
String roleArn = "<roleArn>"; //RAM角色的标识ARN。
```

RAM用户的AccessKeyId和AccessKeySecret为创建RAM用户时获取。

获取RAM角色的roleArn的操作步骤如下：

1. 登录RAM控制台。
2. 在左侧导航栏，单击**RAM角色管理**。
3. 在**RAM角色管理**页面下方的RAM角色列表，单击目标RAM角色名称。
4. 在RAM角色详情页面的**基本信息**区域，复制**ARN**信息。

步骤四：获取登录Token

通过STS的AssumeRole接口获取临时AccessKey和Token后，调用登录服务接口获取登录Token。

> **注意** STS返回的安全Token中可能包含特殊字符，请对特殊字符进行URL编码后再输入。

请求样例：

```
http://signin.aliyun.com/federation?Action=GetSigninToken
    &AccessKeyId=<STS返回的临时AccessKeyId>
    &AccessKeySecret=<STS返回的临时AccessKeySecret>
    &SecurityToken=<STS返回的安全Token>
    &TicketType=mini
```

步骤五：生成免登录链接

利用获取到的登录Token与待嵌入的ARMS控制台页面链接生成免登录访问链接，以最终实现在自建Web中免登录访问ARMS控制台页面的目的。

> **说明** 由于Token有效期为3小时，建议在自建Web应用中将链接设置为每次请求时生成新登录Token。

1. 在ARMS控制台获取待嵌入的控制台页面的URL链接。

   例如上海地域应用页面的URL链接为：

   ```
   https://arms.console.aliyun.com/apm?iframeMode=true&pid=${pid}&regionId=${regionId}#/${pid}/home
   ```

> 说明
>
> - URL链接必须为ARMS应用监控或前端监控的控制台地址，链接中的 `{pid}` 和 `{regionId}` 请从实际URL中获取。
> - 在URL链接的search部分添加 `iframeMode=true` ，可以隐藏ARMS控制台原有的顶部菜单栏和左侧导航栏。
> - 在URL链接的search部分添加 `hideTopbar=true` 可以隐藏顶部菜单栏，添加 `hideSidebar=true` 可以隐藏左侧导航栏，同时使用等同于添加 `iframeMode=true` 。

2. 利用登录Token与ARMS控制台页面链接生成免登录访问链接。

```
http://signin.aliyun.com/federation?Action=Login
    &LoginUrl=<登录失效跳转的地址，一般配置为自建Web配置302跳转的URL>
    &Destination=<ARMS控制台页面链接>
    &SigninToken=<获取到的登录Token>
```

3. 在浏览器访问该免登录访问链接。

4.11. 使用ARMS监控异步任务

对于通过Spring @Async标签实现的异步任务，ARMS默认支持对其进行监控。此外，ARMS还支持通过添加异步透传扫描包和使用ARMS SDK进行手动透传对自定义异步任务实现监控。若您的异步任务出现接口超时等异常，可以通过调用链路查看异步任务上下游以便及时处理潜在问题。

前提条件

该功能要求ARMS探针版本为公测版本v2.7.1.3及以上。请联系ARMS钉钉服务账号arms160804，为您进行探针新版本升级。

方式一：默认支持Spring @Async标签

ARMS默认支持监控使用Spring @Async标签实现的异步任务。对于下列Spring框架中默认的Executor和Task，ARMS会自动完成增强：

- Executor:
  - org.springframework.scheduling.concurrent.ConcurrentTaskExecutor
  - org.springframework.core.task.SimpleAsyncTaskExecutor
  - org.springframework.scheduling.quartz.SimpleThreadPoolTaskExecutor
  - org.springframework.core.task.support.TaskExecutorAdapter
  - org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor
  - org.springframework.scheduling.concurrent.ThreadPoolTaskScheduler
  - org.springframework.jca.work.WorkManagerTaskExecutor
  - org.springframework.scheduling.commonj.WorkManagerTaskExecutor
- Task:
  - org.springframework.aop.interceptor.AsyncExecutionInterceptor$1
  - org.springframework.aop.interceptor.AsyncExecutionInterceptor$$Lambda$

方式二：添加异步透传扫描包

您可以选择在应用设置中添加异步透传扫描包实现异步任务监控。异步透传扫描包中的Runnable、Callable和Supplier接口在创建新对象时会自动捕获当前线程调用链的上下文，并在异步线程中执行时使用该调用链上下文，完成串联。

1. 登录ARMS控制台。
2. 在左侧导航栏，选择**应用监控 > 应用列表**。
3. 在顶部菜单栏，选择地域。
4. 在**应用列表**页面，单击应用名称。
5. 在左侧导航栏，单击**应用设置**，然后单击**自定义配置**页签。
6. 在**高级设置**区域的**异步透传扫描包名**对话框中，添加异步透传扫描包。

7. 在页面底部，单击**保存**。

> **注意** 配置后需重启应用才能使功能生效。如需添加多个异步透传扫描包，可以使用半角逗号（,）分隔。

例如，对于通过以下示例代码创建的异步任务，您可以使用添加异步透传扫描包的方式对此异步任务进行监控。此示例代码中，异步透传包名为com.alibaba.arms.brightroar.console.service。

> **注意** 在配置时，您可以按需调整异步透传包名的范围。若需监控的异步任务过多，您可以缩小异步透传包名的范围。在本示例中，除了输入完整的异步透传包名com.alibaba.arms.brightroar.console.service以外，您还可以输入更短的前缀包名com.alibaba.arms来自动扫描此目录下的所有子透传包。但需注意的是，若前缀包名范围过大，会对性能产生影响，请谨慎操作。

```
package com.alibaba.arms.brightroar.console.service;
@Service
public class NameService {
    private ExecutorService es = Executors.newFixedThreadPool(5);
    public void name() {
        es.submit(new Runnable() {
            @Override
            public void run() {
                System.out.println(System.currentTimeMillis()+ ": my name is john, " + Thre
ad.currentThread().getId());
            }
        });
    }
}
```

方式三：使用ARMS SDK手动透传

当您的使用方式比较复杂，上述场景无法满足需求时，您还可以选择使用ARMS SDK进行手动透传。通过手动增强Runnable接口、Callable接口和Executor，在异步线程中完成调用链串联，实现异步任务监控。

1. 在Maven项目的pom.xml中添加以下依赖。

```
<dependency>
 <groupId>com.alibaba.arms.apm</groupId>
 <version>1.7.5</version>
 <artifactId>arms-sdk</artifactId>
</dependency>
```

2. 增强Runnable接口、Callable接口和Executor。您可以按需选择以下任意一种增强方式。
   - **方式一：增强Runnable接口和Callable接口**
     使用 `TraceRunnable.asyncEntry()` 增强Runnable接口，使用 `TraceCallable.asyncEntry()` 增强Callable接口。示例代码如下：

```
public class AsyncEntryExample {
    private final ExecutorService executor = Executors.newSingleThreadExecutor();
    @GetMapping(value = "/sdk-async-plugin/asyncEntry-propagation")
    public String asyncEntryAndExecute() throws Exception {
        CompletableFuture<String> future = new CompletableFuture<>();
        Runnable command = TraceRunnable.asyncEntry(() -> future.complete("asyncEntry
-execute"));
        executor.execute(command);
        Thread.sleep(1000);
        return future.get();
    }
}
```

   - **方式二：增强Executor**
     使用 `TraceExecutors.wrapExecutorService(executr, true)` 增强Executor。示例代码如下：

```
public class AutoExample {
    private final ExecutorService contextPropagationExecutor
            = TraceExecutors.wrapExecutorService(Executors.newSingleThreadExecutor(),
true);
    @GetMapping(value = "/sdk-async-plugin/auto-context-propagation")
    public String autoWrapAndExecute() throws Exception {
        CompletableFuture<String> future = new CompletableFuture<>();
        contextPropagationExecutor.execute(() -> future.complete("auto-execute"));
        Thread.sleep(1000);
        return future.get();
    }
}
```

   - **方式三：同时增强Runnable接口、Callable接口和Executor**
     示例代码如下：

```
public class ManualExample {
    private final ExecutorService traceExecutor
            = TraceExecutors.wrapExecutorService(Executors.newSingleThreadExecutor())
;
    @GetMapping(value = "/sdk-async-plugin/manual-context-propagation")
    public String manualWrapAndExecute() throws Exception {
        CompletableFuture<String> future = new CompletableFuture<>();
        traceExecutor.execute(TraceRunnable.wrap(() -> future.complete("manual-execut
e")));
        traceExecutor.execute(() -> "Not captured");
        Thread.sleep(1000);
        return future.get();
    }
}
```

执行结果

配置完成后，您可以在调用链路详情页查看异步任务的调用链详情。具体详情，请参见调用链路查询。

4.12. 通过TraceExplorer实时分析链路数据

本文将介绍如何通过链路分析快速定位五种经典线上问题，更直观的了解链路分析的用法与价值。

背景信息

除了使用调用链排查单次请求的异常，或者使用预聚合的链路统计指标进行服务监控与告警之外，链路追踪还支持基于明细链路数据的后聚合分析，简称链路分析（Trace Explorer）。相比调用链，链路分析能够更快的定界问题；相比预聚合的监控图表，链路分析可以更灵活的实现自定义诊断。

链路分析是基于已存储的全量链路明细数据，自由组合筛选条件与聚合维度进行实时分析，可以满足不同场景的自定义诊断需求。例如，查看耗时大于3秒的慢调用时序分布，查看错误请求在不同机器上的分布，或者查看VIP客户的流量变化等。

问题一：流量不均

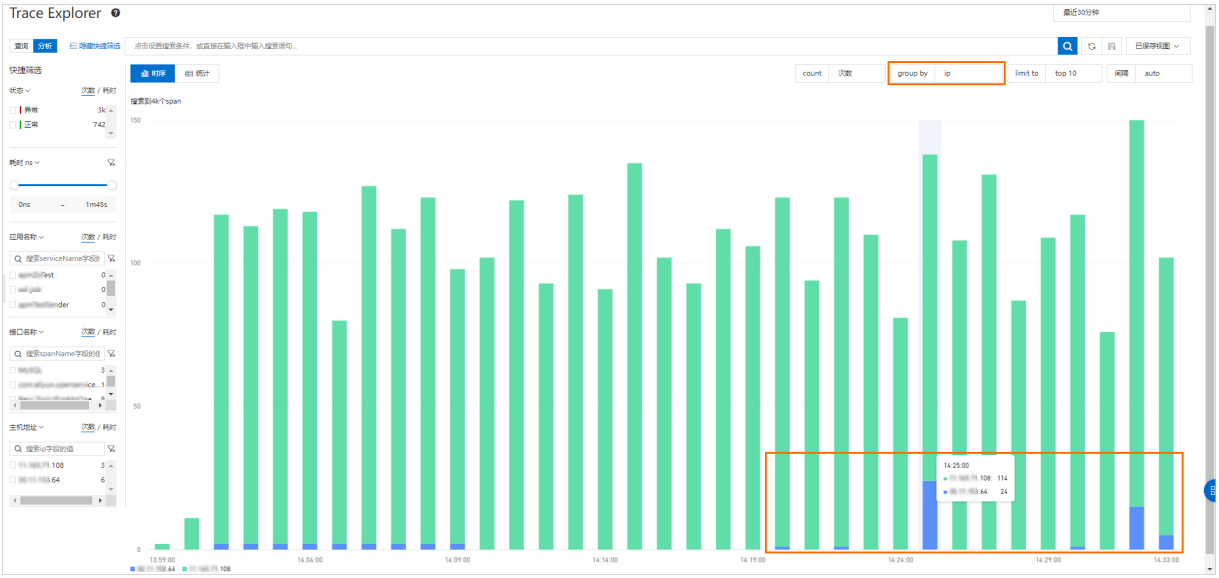
负载均衡配置错误，导致大量请求打到少量机器，造成“热点”影响服务可用性，怎么办？

流量不均导致的“热点击穿”问题，很容易造成服务不可用。在生产环境中出现过多起这样的案例，比如因负载均衡配置错误，注册中心异常导致重启节点的服务无法上线，DHT哈希因子异常等。

流量不均的最大风险在于能否及时发现“热点”现象。它的问题表象更多是服务响应变慢或报错，传统的监控无法直观的反映热点现象，所以大部分运维人员都不会第一时间考虑这个因素，从而浪费了宝贵的应急处理时间，造成故障影响面不断扩散。

通过链路分析按IP分组统计链路数据，可以直观地看到调用请求分布在哪些机器上，特别是问题发生前后的流量分布变化。如果大量请求突然集中在一台或少量机器，很可能是流量不均导致的热点问题，然后再结合问题发生点的变更事件，快速定位造成故障的错误变更，及时回滚。

在Trace Explorer页面设置按IP聚合，如下图所示，可以发现大部分流量集中在XX.XX.XX.108这台机器上。



问题二：单机故障

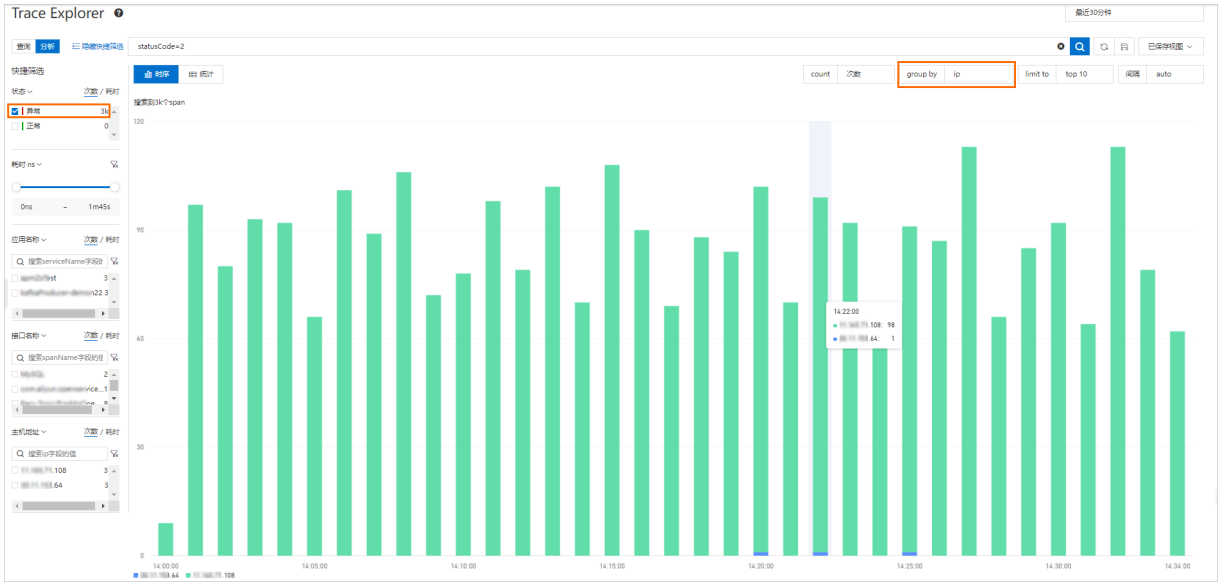
网卡损坏、CPU超卖、磁盘打满等单机故障，导致部分请求失败或超时，如何排查？

单机故障每时每刻都在频繁发生，特别是核心集群由于节点数量比较多，从统计概率来看几乎是一种“必然”事件。单机故障不会造成服务大面积不可用，但是会造成少量的用户请求失败或超时，持续影响用户体验和答疑成本，需要及时处理。

单机故障可以分为宿主机故障和容器故障两类（在Kubernetes环境可以分为Node和Pod）。例如CPU超卖、硬件故障等都是宿主机级别，会影响所有容器；而磁盘打满、内存溢出等故障仅影响单个容器。因此，在排查单机故障时，可以根据宿主机IP和容器IP两个维度分别进行分析。

面对这类问题，可以通过链路分析先筛选出异常或超时请求，然后再根据宿主机IP或容器IP进行聚合分析，可以快速判断是否存在单机故障。如果异常请求集中在单台机器，可以尝试替换机器进行快速恢复，或者排查该机器的各项系统参数：例如磁盘空间是否已满、CPU Steal Time是否过高等。如果异常请求分散在多台机器，那么大概率可以排除单机故障因素，可以重点分析下游依赖服务或程序逻辑是否异常。

在Trace Explorer页面筛选错误调用或慢调用，并设置按IP进行分组统计，如果异常调用集中出现在特定机器，则有较大概率是机器故障。



问题三：慢接口治理

新应用上线或大促前性能优化，如何快速梳理慢接口列表，解决性能瓶颈？

新应用上线或大促备战时通常需要做一次系统性的性能调优。第一步就是分析当前系统存在哪些性能瓶颈，梳理出慢接口的列表和出现频率。

此时，可以通过链路分析筛选出耗时大于一定阈值的调用，再根据接口名称进行分组统计，这样就可以快速定位慢接口的列表与规律，然后对出现频率最高的慢接口逐一进行治理。

找到慢接口后，可以结合相关的调用链、方法栈和线程池等数据定位慢调用根因。常见原因包括以下几类：

- 数据库或微服务连接池过小，大量请求处于获取连接状态。可以调大连接池最大线程数解决。
- N+1问题。例如一次外部请求内部调用了上百次的数据库调用，可以将碎片化的请求进行合并，降低网络传输耗时。
- 单次请求数据过大，导致网络传输和反序列化时间过长，而且容易导致Full GC。可以将全量查询改为分页查询，避免一次请求过多数据。
- 日志框架“热锁”。可以将日志同步输出改为异步输出。

在Trace Explorer页面筛选大于5秒的慢调用，并设置按接口名进行分组统计，发现慢接口的规律。

问题四：业务流量统计

如何分析重保客户或渠道的流量变化和服务质量？

在实际生产环境中，服务通常是标准的，但业务却是分类分级的。同样的订单服务，我们需要按照类目、渠道、用户等维度进行分类统计，实现精细化运营。例如，对于线下零售渠道而言，每一笔订单、每一个POS机的稳定性都可能会触发舆情，线下渠道的SLA要求要远高于线上渠道。那么，应该如何在通用的电商服务体系中，精准的监控线下零售链路的流量状态和服务质量呢？

这里可以使用链路分析的自定义Attributes过滤和统计实现低成本的业务链路分析。例如，在入口服务针对线下订单打上 `{"attributes.channel": "offline"}` 的标签，然后再针对不同门店、用户客群和商品类目分别打标。最后，通过对 `attributes.channel = offline` 进行过滤，再对不同的业务标签进行group by来分组统计调用次数、耗时或错误率等指标，就可以快速地分析出每一类业务场景的流量趋势与服务质量。

问题五：灰度发布监控

500台机器分10批发布，如何在第一批灰度发布后，就能快速判断是否有异常？

变更三板斧“可灰度、可监控、可回滚”是保障线上稳定性的重要准则。其中，分批次灰度变更是降低线上风险，控制爆炸半径的关键手段。一旦发现灰度批次的服务状态异常，应及时进行回滚，而不是继续发布。然而，生产环境很多故障的发生都是由于缺乏有效的灰度监控导致的。

例如，当微服务注册中心异常时，重启发布的机器无法进行服务注册上线。由于缺乏灰度监控，前几批重启机器虽然全部注册失败，导致所有流量都集中路由到最后一批机器，但是应用监控的总体流量和耗时没有显著变化，直至最后一批机器也重启注册失败后，整个应用进入完全不可用状态，最终导致了严重的线上故障。

在上述案例中，如果使用 `{"attributes.version": "v1.0.x"}` 对不同机器流量进行版本打标，通过链路分析对 `attributes.version` 进行分组统计，可以清晰的区分发布前后或不同版本的流量变化和服务质量，不会出现灰度批次异常被全局监控掩盖的情况。

链路分析的约束限制

链路分析虽然使用灵活，可以满足不同场景的自定义诊断需求，但是它也有几点使用约束限制：

- 基于链路明细数据进行分析的成本较高。

链路分析的前提是尽可能完整的上报并存储链路明细数据。如果采样率比较低导致明细数据不全，链路分析的效果就会大打折扣。为了降低全量存储成本，可以在用户集群内部署边缘数据节点，进行临时数据缓存与处理，降低跨网络上报开销。或者，在服务端进行冷热数据分离存储，热存储进行全量链路分析，冷存储进行错慢链路诊断。

- 后聚合分析的查询性能开销大，并发小，不适合用于告警。
  链路分析是实时的进行全量数据扫描与统计，查询性能开销要远大于预聚合统计指标，所以不适合进行高并发的告警查询。需要结合自定义指标功能将后聚合分析语句下推至客户端进行自定义指标统计，以便支持告警与大盘定制。
- 结合自定义标签埋点，才能最大化释放链路分析价值。
  链路分析不同于标准的应用监控预聚合指标，很多自定义场景的标签需要用户手动埋点打标，这样才能最有效的区分不同业务场景，实现精准分析。

4.13. 通过OpenTelemetry Java SDK进行手工埋点

本文主要介绍如何通过OpenTelemetry开源SDK手动埋点监控您的应用。

前提条件

该功能要求ARMS探针版本为公测版本v2.7.1.3及以上。请联系ARMS钉钉服务账号 `arms160804` ，为您进行探针新版本升级。

背景信息

云原生背景下，由OpenTracing演进而来的OpenTelemetry提供了多达数十种语言的SDK，基于一套规范的API和统一的数据格式，成为可观测的事实标准。

ARMS Java Agent从版本v2.7.1.3开始，内置对OpenTelemetry Java SDK的支持。如果您的应用已经通过OpenTelemetry Java SDK手动埋点，则无需任何改动，ARMS会将相关的Span记录作为独立入口或者内部方法栈，从而监控您的应用。如果您的应用尚未埋点，请先通过依赖OpenTelemetry Java SDK，实现手动埋点。

手动埋点

如果您的应用尚未埋点，请先通过引入以下Maven依赖，实现手动埋点。更多信息，请参见OpenTelemetry官方文档。

```
<dependencies>
    <dependency>
        <groupId>io.opentelemetry</groupId>
        <artifactId>opentelemetry-api</artifactId>
    </dependency>
    <dependency>
          <groupId>io.opentelemetry</groupId>
        <artifactId>opentelemetry-sdk-trace</artifactId>
    </dependency>
    <dependency>
          <groupId>io.opentelemetry</groupId>
        <artifactId>opentelemetry-sdk</artifactId>
    </dependency>
    <dependency>
          <groupId>io.opentelemetry</groupId>
        <artifactId>opentelemetry-exporter-logging</artifactId>
    </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.opentelemetry</groupId>
      <artifactId>opentelemetry-bom</artifactId>
      <version>1.2.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

ARMS对OpenTelemetry埋点的兼容

ARMS对OpenTelemetry埋点的兼容的介绍涉及以下名词，OpenTelemetry相关的其他名称解释，请参见OpenTelemetry Specification。

- Span：一次请求的一个具体操作，比如远程调用入口或者内部方法调用。
- SpanContext：一次请求追踪的上下文，用于关联该次请求下的具体操作。
- Attribute：Span的附加属性字段，用于记录关键信息。

OpenTelemetry的Span可以分为三类：

- 入口Span：会创建新的SpanContext，例如Server、Consumer。

  > **说明** 对于此类Span，ARMS的埋点入口多位于框架内部，手动埋点时链路上下文已存在，ARMS会将OpenTelemetry的入口Span作为内部Span处理。对于ARMS没有埋点的入口，则OpenTelemetry的入口Span保持不变，例如异步调用或者自定义RPC框架入口，同时ARMS会在客户端聚合，生成相关统计数据。

- 内部Span：会复用已经创建的SpanContext，作为内部方法栈记录。
- 出口Span：会将SpanContext透传下去，例如Client、Producer。

目前，ARMS对于入口Span和内部Span做了兼容。对于出口Span，ARMS暂不支持按照OpenTelemetry标准透传，而是按照ARMS自定义格式透传。

例如以下代码：

```
package com.alibaba.arms.brightroar.console.controller;
import io.opentelemetry.api.OpenTelemetry;
import io.opentelemetry.api.common.AttributeKey;
import io.opentelemetry.api.common.Attributes;
import io.opentelemetry.api.trace.Span;
import io.opentelemetry.api.trace.SpanKind;
import io.opentelemetry.api.trace.StatusCode;
import io.opentelemetry.api.trace.Tracer;
import io.opentelemetry.api.trace.propagation.W3CTraceContextPropagator;
import io.opentelemetry.context.Scope;
import io.opentelemetry.context.propagation.ContextPropagators;
import io.opentelemetry.exporter.logging.LoggingSpanExporter;
import io.opentelemetry.sdk.OpenTelemetrySdk;
import io.opentelemetry.sdk.trace.SdkTracerProvider;
import io.opentelemetry.sdk.trace.export.BatchSpanProcessor;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;
import javax.annotation.PostConstruct;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
@RestController
@RequestMapping("/ot")
public class OpenTelemetryController {
    private Tracer tracer;
    private ScheduledExecutorService ses = Executors.newSingleThreadScheduledExecutor();
    @PostConstruct
    public void init() {
        SdkTracerProvider sdkTracerProvider = SdkTracerProvider.builder()
                .addSpanProcessor(BatchSpanProcessor.builder(new LoggingSpanExporter()).bui
ld())
                .build();
        OpenTelemetry openTelemetry = OpenTelemetrySdk.builder()
                .setTracerProvider(sdkTracerProvider)
                .setPropagators(ContextPropagators.create(W3CTraceContextPropagator.getInst
ance()))
                .buildAndRegisterGlobal();
        tracer = openTelemetry.getTracer("manual-sdk", "1.0.0");
        ses.scheduleAtFixedRate(new Runnable() {
            @Override
            public void run() {
                Span span = tracer.spanBuilder("schedule")
                        .setAttribute("schedule.time", System.currentTimeMillis())
                        .startSpan();
                try (Scope scope = span.makeCurrent()) {
                    System.out.println("scheduled!");
                    Thread.sleep(500L);
                    span.setAttribute("schedule.success", true);
                } catch (Throwable t) {
                    span.setStatus(StatusCode.ERROR, t.getMessage());
                } finally {
```

```
                    span.end();
                }
            }
        }, 10, 30, TimeUnit.SECONDS);
    }
    @ResponseBody
    @RequestMapping("/parent")
    public String parent() {
        Span span = tracer.spanBuilder("parent").setSpanKind(SpanKind.SERVER).startSpan();
        try (Scope scope = span.makeCurrent()) {
            child();
            span.setAttribute("http.method", "GET");
            span.setAttribute("http.uri", "/parent");
        } finally {
            span.end();
        }
        return "parent";
    }
    private void child() {
        Span span = tracer.spanBuilder("child").startSpan();
        try (Scope scope = span.makeCurrent()) {
            span.addEvent("Sleep Start");
            Thread.sleep(1000);
            Attributes attr = Attributes.of(AttributeKey.longKey("cost"), 1000L);
            span.addEvent("Sleep End", attr);
        } catch (Throwable e) {
            span.setStatus(StatusCode.ERROR, e.getMessage());
        } finally {
            span.end();
        }
    }
}
```

以上示例代码中通过OpenTelemetry SDK创建了三个Span：

- `parent`：按照OpenTelemetry标准是HTTP入口，但是由于ARMS在Tomcat内置代码中已经创建了链路上下文，因此这里会作为一个内部方法记录在ARMS方法栈上。
- `child`：`parent` Span的内部Span，作为内部方法记录在方法栈上。
- `schedule`：独立线程入口Span，默认情况下ARMS没有为此类入口创建上下文，因此这里会作为一个自定义方法入口，并生成相应的统计数据。

在ARMS控制台查看 `parent` 和 `child`

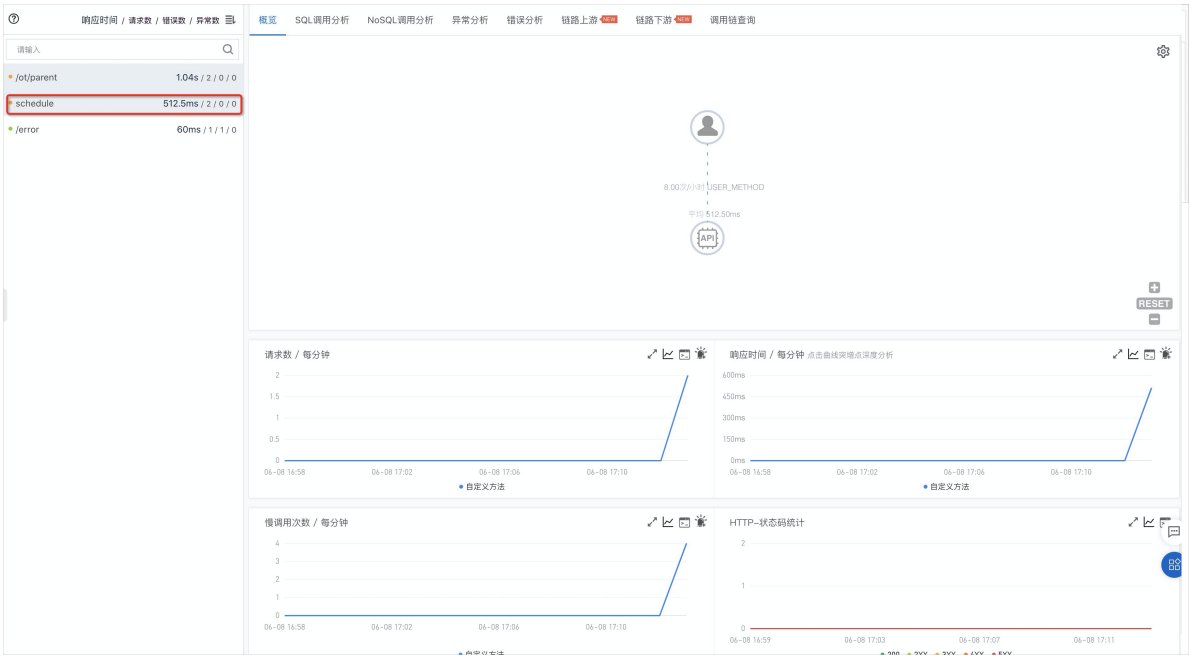
在ARMS控制台找到/ot/parent的HTTP入口的内部方法栈，可以看到多了以下Span的展示。更多信息，请参见调用链路查询。

在ARMS控制台查看 `schedule`

ARMS控制台支持通过以下几个页面查看：

- 可以在ARMS控制台**应用总览**页面看到自定义入口的统计。更多信息，请参见应用总览。

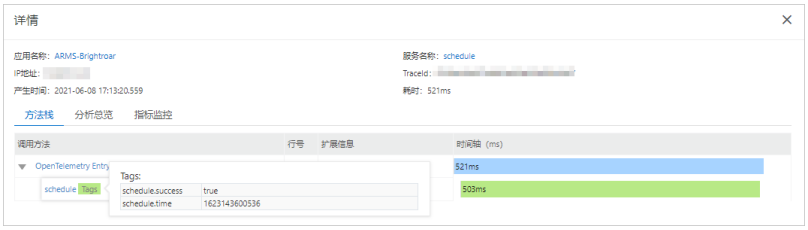
- 可以在**接口调用**页面查看Span详细信息。更多信息，请参见接口调用。



- 可以在**调用链路**页面看到独立的入口。更多信息，请参见调用链路查询。

单击**详情**列的放大镜图标，可以查看入口的详细信息。

目前ARMS支持将OpenTelemetry Span的Attribute作为Tags展示，将鼠标悬浮于Span名称后面的**Tags**即可看到Span的Attribute。



4.14. 在阿里云Prometheus监控下查看应用监控大盘

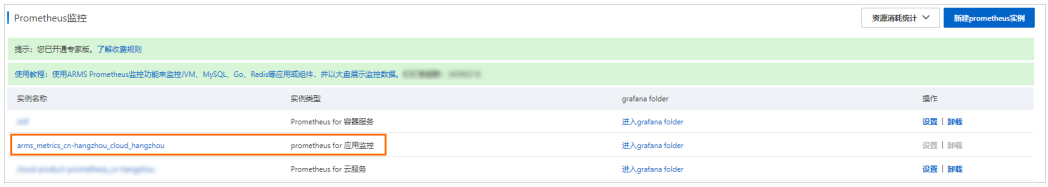
阿里云Prometheus默认集成了ARMS应用监控数据源，您可以直接在Prometheus监控下查看应用监控大盘。

前提条件

已开通应用监控和Prometheus监控。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏选择**Prometheus监控 > Prometheus实例列表**。
3. 在**Prometheus监控**页面的顶部菜单栏，选择应用监控所在的地域。
   在**Prometheus监控**页面，实例类型为prometheus for 应用监控的实例即为默认集成的应用监控。



4. 单击应用监控实例名称，可以查看预置大盘并获取Remote Read地址、Remote Write地址、HTTP API地址和Token信息。

查看预置大盘

在**大盘列表**页面可以查看应用监控预置大盘。大盘内各指标含义，请参见应用监控指标说明。

获取Remote Read地址等信息

在**设置**页面获取Remote Read地址、Remote Write地址、HTTP API地址和Token信息。

5.参考信息

5.1. ARMS Java探针性能压测报告

本文为ARMS Java探针的性能压测报告。在本报告中，测试人员对同一个应用以不同的压测流量进行了多轮压测，以此来验证ARMS Java探针的性能。

背景信息

压测环境基本信息

- ARMS Java探针版本：v2.7.3
- 压测平台：阿里云性能测试PTS
- 测试应用所在ECS实例的硬件配置：2 Intel(R) Xeon(R) Platinum 8369B CPU @ 2.70GHz，4G memory
- 操作系统版本：CentOS Linux release 7.6.1810 (Core)

测试应用基本信息

该测试应用基于Spring，包含Spring Boot、Spring MVC，模拟的Redis客户端和JDBC连接池（用于模拟MySQL客户端）。在监控该应用的过程中，针对每个事务，ARMS Java探针会抓取7个Span，包括1个Tomcat，1个Spring MVC，1个Method，2个Jedis和2个MySQL。

下载测试应用。

基线指标

基线是测试应用在不安装探针的情况下进行压测得到的基准指标。指标包括：

- CPU：CPU总体百分比
- MEM：所占用的内存
- TPS：每秒事务数
- RT：响应时间，单位为毫秒

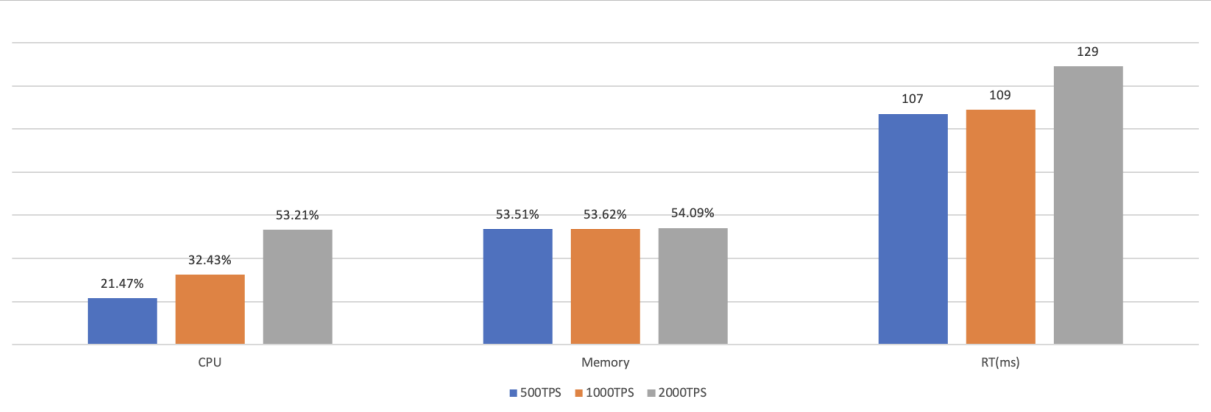
探针性能开销计算方法

在得到基线指标后，为测试应用安装ARMS Java探针，并在安装探针的情况下再次进行压测，然后观察CPU、MEM、TPS和RT指标的情况，以此来计算探针性能开销。

探针性能开销 =（安装探针后的压测结果 - 基线指标）÷ 基线指标

基线指标

测试人员测试了应用在500、1,000以及2,000 TPS下的性能状况，并将得到的结果作为后续对比实验的基线指标。基线指标结果如下图所示：



ARMS Java探针压测结果

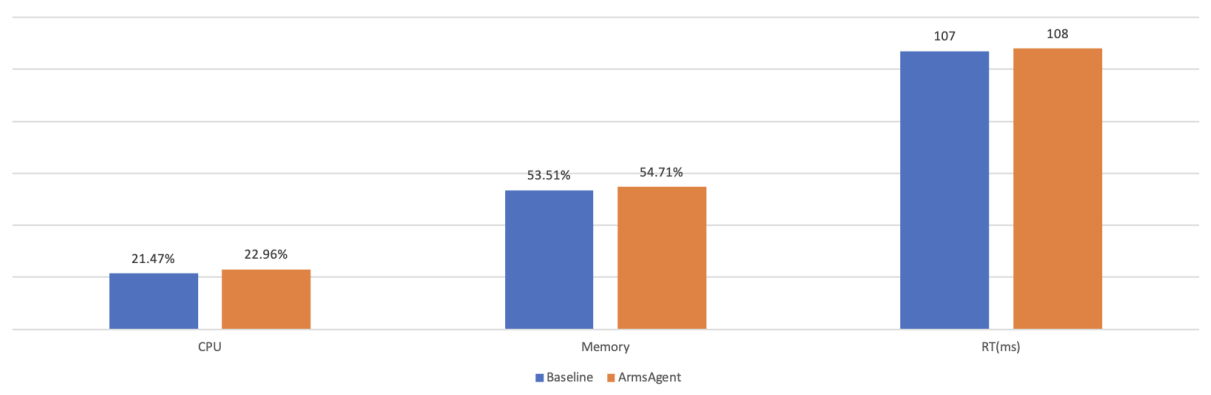
本压测报告中使用的测试场景如下，每个场景分别使用500、1,000以及2,000 TPS进行压测：

- 场景一：使用ARMS Java探针默认配置，探针采样率为10%。
- 场景二：使用ARMS Java探针默认配置，探针采样率为100%。

> **说明** ARMS Java探针默认配置指的是在ARMS控制台的**应用设置** > **自定义配置**中默认开启的配置项。

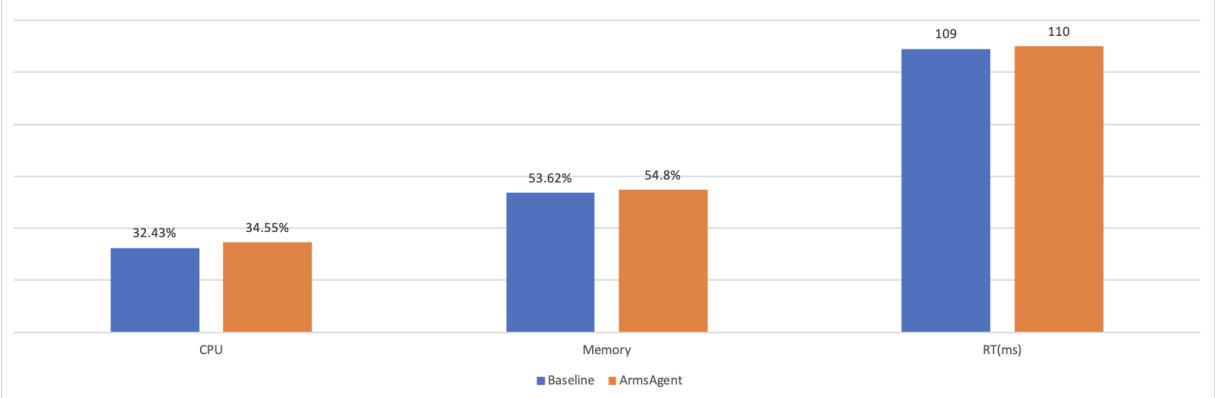
场景一压测详情

500 TPS下与基线指标对比：



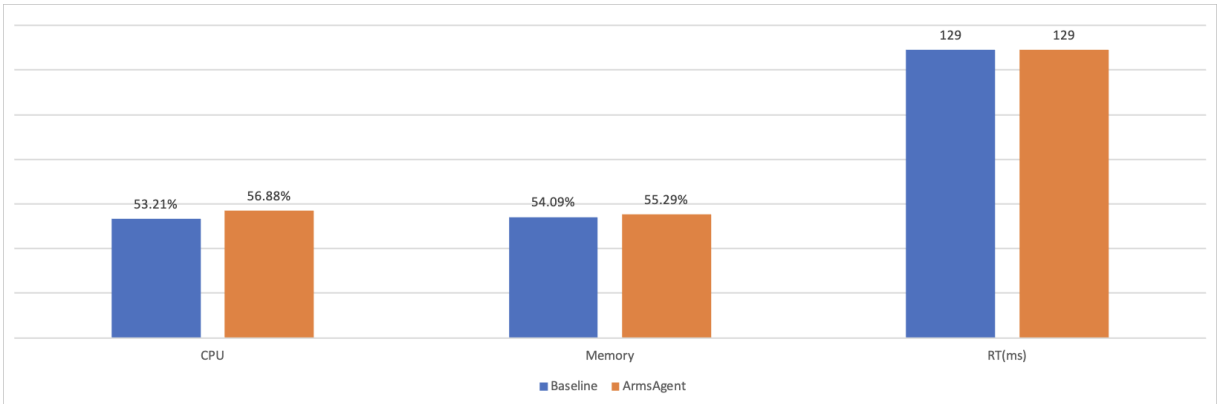
可以看到在500 TPS的压测条件下，ARMS Java探针的额外CPU开销为1.49%（相对基线指标额外损耗6.94%），额外内存开销为1.2%，RT损耗约为1毫秒。

1,000 TPS下与基线指标对比：



可以看到在1,000 TPS的压测条件下，ARMS Java探针的额外CPU开销为2.12%（相对基线指标额外损耗6.54%），额外内存开销为1.18%，RT损耗约为1毫秒。

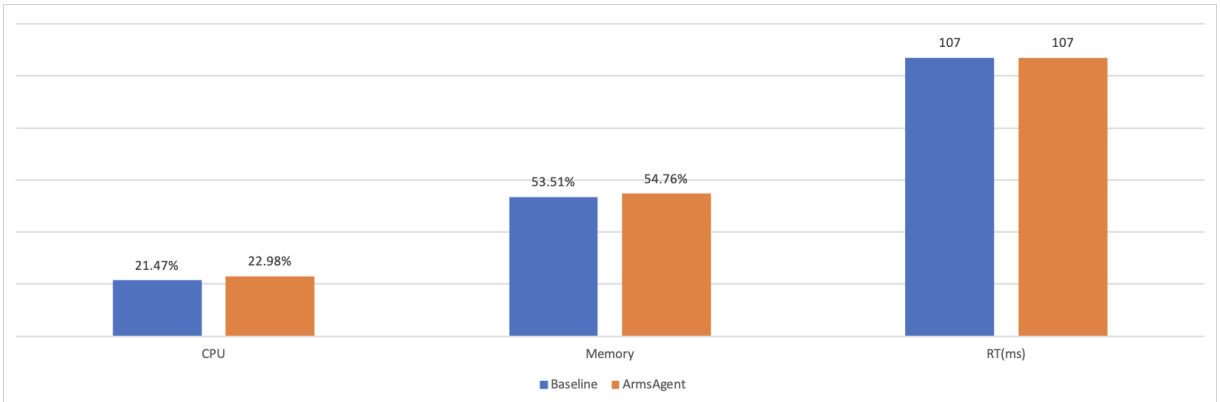
2,000 TPS下与基线指标对比：



可以看到在2,000 TPS的压测条件下，ARMS Java探针的额外CPU开销为3.67%（相对基线指标额外损耗6.90%），额外内存开销为1.2%，RT损耗为0毫秒。

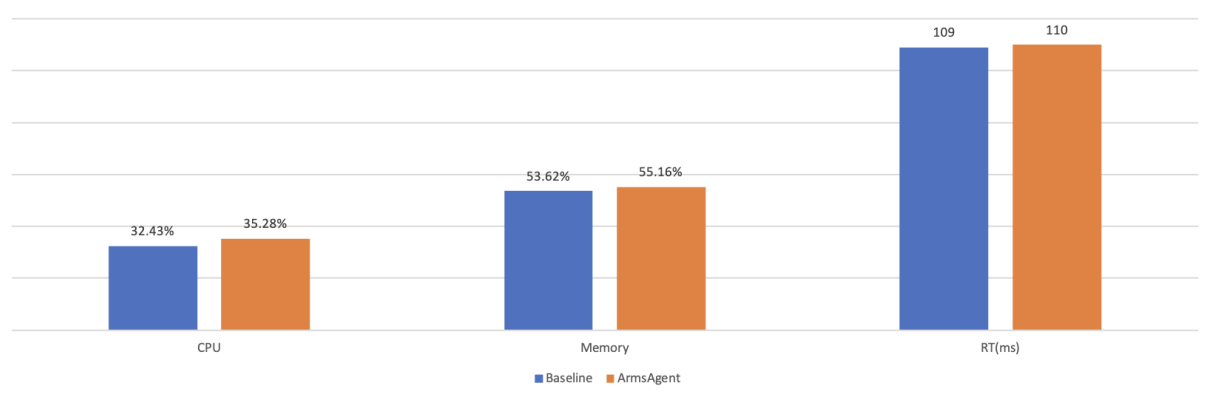
场景二压测详情

500 TPS下与基线指标对比：



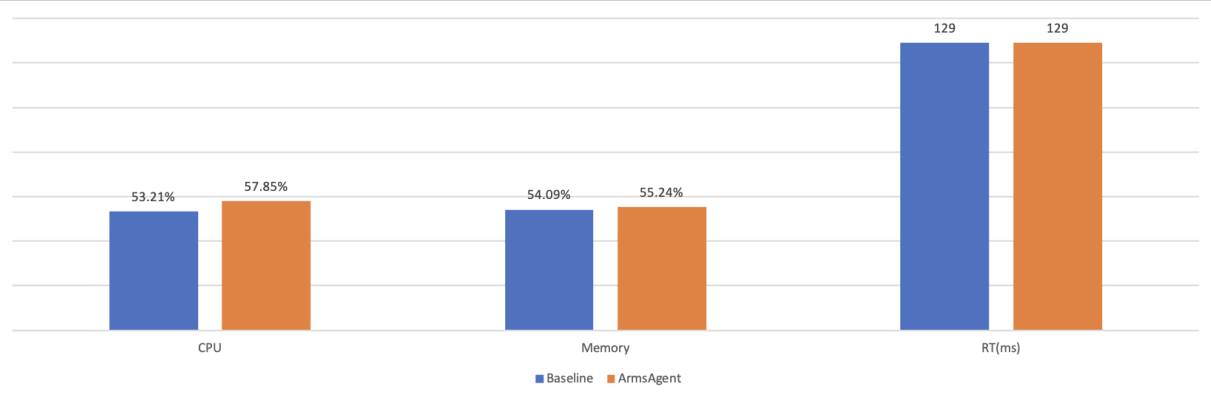
可以看到在500 TPS的压测条件下，ARMS Java探针的额外CPU开销为1.51%（相对基线指标额外损耗7.03%），额外内存开销为1.25%，RT损耗为0毫秒。

1,000 TPS下与基线指标对比：



可以看到在1,000 TPS的压测条件下，ARMS Java探针的额外CPU开销为2.85%（相对基线指标额外损耗8.79%），额外内存开销为1.54%，RT损耗为1毫秒。

2,000 TPS下与基线指标对比：



可以看到在2,000 TPS的压测条件下，ARMS Java探针的额外CPU开销为4.64%（相对基线指标额外损耗8.72%），额外内存开销为1.15%，RT损耗为0毫秒。

ARMS Java探针性能总结

CPU

对比项	500 TPS	1,000 TPS	2,000 TPS
基线	21.47%	32.43%	53.21%
场景一	22.96%	34.55%	56.88%
场景二	22.98%	35.28%	57.85%

Memory（MEM）

对比项	500 TPS	1,000 TPS	2,000 TPS
基线	53.51%	53.62%	54.09%
场景一	54.71%	54.80%	55.29%
场景二	54.76%	55.16%	55.24%

RT

对比项	500 TPS	1,000 TPS	2,000 TPS
基线	107 ms	109 ms	129 ms
场景一	108 ms	110 ms	129 ms
场景二	107 ms	110 ms	129 ms

在不同TPS下，经过对相同默认配置不同采样率的ARMS Java探针进行性能压测和分析，可以看到ARMS Java探针能够在CPU额外开销5%~10%以内，内存开销100M以内的情况下（其中Metaspace会带来30M左右的增长），正常提供大部分服务。

5.2. ARMS应用监控支持的Java组件和框架

本文列出了ARMS应用监控支持的Java第三方组件和框架。如果待监控应用使用的组件或框架不在支持范围内，则需要通过配置通用Filter拦截器的方式进行监控数据采集。

ARMS应用监控支持的JDK版本

- JDK 1.7.0+
- JDK 1.8.0_25+

  > **说明** Kubernetes集群应用部署建议：JDK 1.8.0_191+。

- JDK 11.0.8+

ARMS应用监控支持的Java组件和框架

> **说明** ARMS对于JDK 11版本组件的支持目前处于公测期，如需体验，请联系ARMS服务钉钉账号 `arms160804` 申请使用。

组件	JDK 1.7	JDK 1.8	JDK 11
Aliware MQ	4.1.X+	4.1.X+	4.1.X+
Druid	1.0.0+	1.2.4+	1.2.4+
DSF（Daojia Service Framework）	不支持	2.1.X+	2.1.X+
Dubbo	2.5.X+	2.5.X+	2.5.X+
Elasticsearch Rest Client	5.X+	7.X+	7.X+
Elasticsearch Rest High Level Client	5.X+	7.X+	7.X+
Feign	不支持	9.X+	9.X+
Google HTTP Client	1.10.X+	1.10.X+	1.10.X+

组件	JDK 1.7	JDK 1.8	JDK 11
GRPC-Java	1.15+	1.15+	1.15+
HikariCP	2.3.0+	2.3.13+	2.3.13+
Ali-HSF（High Speed Framework）	2.2.4.6-TLS+	2.2.4.6-TLS+	2.3.13+
HttpClient 3	3.X+	3.X+	3.X+
HttpClient 4	4.X+	4.X+	4.X+
Hystrix	1.5.X+	1.5.X+	1.5.X+
JDK HTTP	1.7.X+	1.7.X+	1.7.X+
Jedis	2.X+	2.X+	2.X+
Jetty	8.X+	8.X+	8.X+
Lettuce	不支持	4.0+	4.0+
MariaDB	1.3+	1.3+	1.3+
MemCached	2.8+	2.8+	2.8+
MongoDB	3.7+	3.7+	3.7+
MyBatis	3.X+	3.X+	3.X+
MySQL JDBC	5.0.X+	5.0.X+	5.0.X+
OKHttp	2.X+	2.X+	2.X+
Oracle JDBC	10.2.X+	10.2.X+	10.2.X+
Play Framework	不支持	1.4.X	1.4.X
PostgreSQL JDBC	9.4+	9.4+	9.4+
RabbitMQ	不支持	2.1.X+	2.1.X+
Reactor	不支持	3.X+	3.X+
Reactor Netty	不支持	0.9+	0.9+
Redisson	不支持	3.10.0+	3.10.0+
Resin	3.0+	3.0+	3.0+
RxJava	2.X+	2.X+	2.X+
SchedulerX	1.2.0+	1.2.0+	1.2.0+

组件	JDK 1.7	JDK 1.8	JDK 11
SofaRPC	5.4.X+	5.4.X+	5.4.X+
Spring	4.X+	4.X+	4.X+
Spring Boot	1.3.X+	1.3.X+	1.3.X+
Spring Cloud Gateway	不支持	2.1.0.RELEASE+	2.1.0.RELEASE+
Spring WebMVC	不支持	5.0.6.RELEASE+	5.0.6.RELEASE+
Spring WebFlux	不支持	5.0.0.RELEASE+	5.0.0.RELEASE+
SQLServer JDBC	6.4+	6.4+	6.4+
Thrift	0.8+	0.8+	0.8+
Tomcat	7.X+	7.X+	7.X+
Undertow	1.3X+	1.3X+	1.3X+
WebLogic	12.X+	12.X+	12.X+

配置通用Filter拦截器以采集数据

如果应用的组件或框架不在支持范围内，可以通过配置通用Filter拦截器的方式采集数据。配置步骤：

1. 在工程的*pom.xml*文件中引入*arms-sdk-1.7.1.jar*。

```
<dependency>
    <groupId>com.alibaba.arms.apm</groupId>
    <artifactId>arms-sdk</artifactId>
    <version>1.7.1</version>
</dependency>
```

> **说明** 如果无法获取*pom.xml*文件，请下载arms-sdk-1.7.1.jar。

2. 在*web.xml*中配置ARMS的Filter拦截器。

```
<filter>
    <filter-name>EagleEyeFilter</filter-name>
    <filter-class>com.alibaba.arms.filter.EagleEyeFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>EagleEyeFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

3. 登录ARMS控制台。
4. 将Java应用接入ARMS。具体操作，请参见为Java应用安装探针。
5. 重启应用令配置生效。

5.3. ARMS应用监控支持的Kafka版本

2.7.1.2及以上版本的ARMS Agent增加了Kafka插件，使ARMS应用监控可以对应用中特定版本和接入方式的Kafka进行Trace跟踪。本文列出了ARMS应用监控支持的Kafka版本，以及不同接入方式的注意事项。

ARMS Agent版本要求

仅v2.7.1.2及以上版本的ARMS Agent增加了Kafka插件。如果你需要使用包含Kafka插件的ARMS Agent，请联系ARMS钉钉服务账号 `arms160804` ，为您进行探针版本升级。

Kafka版本要求

ARMS Agent的Kafka插件实现Trace跟踪需要依赖Kafka消息协议（Message Protocol）所支持的Header的能力，Kafka消息协议共有以下3个版本。

消息版本	Magic Number	Kafka版本	是否支持Header
v0	0	<v0.10.0.x	否
v1	1	[v0.10.0.x, v0.11.0.x)	否
v2	2	≥v0.11.0.x	是

由于仅v0.11.0及以上版本的Kafka支持Header，因此，ARMS应用监控仅支持v0.11.0及以上版本的Kafka。

Kafka的客户端（Producer和Consumer）和服务端（Broker）版本要求

在理想情况下，客户端和服务端的版本应该是相同的，即Kafka Broker如果使用X版本，那么Producer和Consumer应该使用的也是X版本，这样可减少很多运行中的负担。

但是实际上，很多用户经常会采用不匹配的版本，这不可避免的带来了一些额外的复杂度。

因此，ARMS Agent的Kafka插件对不同版本的客户端和服务端设置了以下约束，即不论Kafka客户端（Producer和Consumer）和服务端（Broker）版本是否相同，只要都大于v0.11.0，ARMS Agent的Kafka插件就会注入Trace数据。

服务端（Borker）	客户端（Provider和Consumer）	Kafka插件是否会注入Trace数据
[v0.10.0.x, v0.11.0.x)	<v0.10.0.x	否
	[v0.10.0.x, v0.11.0.x)	否
	≥v0.11.0.x	否
≥v0.11.0.x	<v0.10.0.x	否
	[v0.10.0.x, v0.11.0.x)	否
	≥v0.11.0.x	是

Kafka接入方式

ARMS对以下两种Kafka接入方式进行了验证。

Producer

Apache Kafka Provider API无需额外操作即可通过官方接入方式接入ARMS。更多信息，请参见Apache Kafka官方文档。

Consumer

Apache Kafka Consumer API并不支持Consumer逐个消费消息，而是采用Pull模式不间断的轮询来消费消息，因此ARMS Agent的Kafka插件没办法直接将Trace注入，需要一个额外的地方进行Consumer消息的注入。如果直接采用Apache Kafka客户端进行Consumer端的接入，ARMS需要进行以下操作。

Consumer端示例文件：

```
package arms.test.kafka;
public class KafkaConsumeTest {
       public void testConsumer(){
           Properties props = new Properties();
           props.put("bootstrap.servers", "PLAINTEXT://XXXX");
           props.put("group.id",  UUID.randomUUID().toString());
           props.put("enable.auto.commit", "true");
           props.put("auto.offset.reset", "earliest");
           props.put("auto.commit.interval.ms", "1000");
           props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDese
rializer");
           props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDe
serializer");
           KafkaConsumer<String,String> kafkaConsumer = new KafkaConsumer<String, String>(
props);
           kafkaConsumer.subscribe(Arrays.asList("test"));
           while (true) {
               ListAdapter consumer;
               ConsumerRecords<String, String> records = kafkaConsumer.poll(100);
               for (ConsumerRecord<String, String> record : records)
                    handler(record);
           }
        }
        public void handler(ConsumerRecord<String, String> record){
               LOGGER.info( Utils.string(record));
        }
}
```

对于以上示例文件，对应的在ARMS Agent安装包的*arms-agent.config*文件中，添加以下内容。

```
profiler.kafka.consumer.entryPoint=arms.test.kafka.KafkaConsumeTest.handler
```

其中，`arms.test.kafka.KafkaConsumeTest.handler`为示例文件中的包、类和方法名。

Spring Kafka在Apache Kafka客户端之上做了一层封装，因此，引入的Spring Kafka版本和Apache Kafka客户端版本需要满足以下映射关系。更多信息，请参见Spring Kafka官方文档。

Spring for Apache Kafka Version	Spring Integration for Apache Kafka Version	`kafka-clients`	Spring Boot
v2.7.0	v5.4.x	v2.7.0或v2.8.0	v2.4.x或v2.5.x
v2.6.x	v5.3.x或v5.4.x	v2.6.0	v2.3.x或v2.4.x
v2.5.x	v3.3.x	v2.5.1	v2.3.x

Spring for Apache Kafka Version	Spring Integration for Apache Kafka Version	kafka-clients	Spring Boot
v2.4.x	v3.2.x	v2.4.1	v2.2.x
v2.3.x	v3.2.x	v2.3.1	v2.2.x

如果您使用的是间接接入Spring Kafka的方式接入Kafka Broker，那么您不需要其他任何配置，ARMS Agent Kafka插件默认支持这种方式的Trace链路追踪。

间接接入Spring Kafka

Overhead Trace数据

ARMS Agent的Kafka插件默认为Kafka Message Header添加以下数据，ARMS应用监控即可通过以下数据实现Trace跟踪。

- EagleEye-TraceID
- EagleEye-RpcID
- EagleEye-SpanID
- EagleEye-pSpanID
- EagleEye-pAppName
- EagleEye-pRpc
- EagleEye-IP
- EagleEye-ROOT-APP
- EagleEye-Sampled
- uber-trace-id
- X-B3-TraceId
- X-B3-SpanId
- X-B3-ParentSpanId
- X-B3-Flags
- X-B3-Sampled
- traceparent

5.4. Agent版本说明

本文主要介绍了ARMS应用监控Java Agent的版本更新历史。

Java Agent版本

版本	发布时间	发布说明

版本	发布时间	发布说明
2.7.1.3	2021年10月18日	• 支持XXL-JOB、SchedulerX、ElasticSearch和VertX。 • 支持Arthas诊断、客户端自定义采样、应用安全攻击防护以及危险组件检测。 • 支持JDBC返回大小、支持Redis命中率、调整了池化监控、优化了部分异步场景的支持。
2.7.1.2	2021年09月17日	• 支持Kafka、RocketMQ、Sofa等插件，支持池化监控，增加调用链展示信息，支持FC场景，支持基于Spring注解的URL收敛。 • 支持Dubbo、SpringCloud全链路灰度，支持SpringCloudGateway金丝雀发布，支持Nacos、Eureka注册中心迁移，服务降级功能，Dubbo、Spring Cloud支持同可用区优先路由。 • 优化启动时间、内存使用率，修复内存快照、线程诊断、数据上报以及SDK等部分Bug，修复安全漏洞。
2.7.1.1	2020年08月14日	• 支持NoSQL监控。 • 支持微服务标签路由。 • 支持N+1调用压缩。 • 修复金融云网络连接问题，优化内存占用。
2.7.1	2020年07月16日	• 支持新版本Jedis插件，解决拓扑图Redis集群不识别问题。
2.7.0	2020年05月20日	• 支持微服务子产品功能。
2.6.2	2020年05月20日	• 支持业务监控。
2.6.1.2	2020年03月19日	• 支持微服务鉴权。 • 支持微服务优雅下线
2.6.1.1	2020年03月16日	• 支持SpringCloud Gateway及Spring Webflux等组件。

版本	发布时间	发布说明
2.6.1	2020年02月14日	• 支持获取微服务元数据等相关功能。
2.6.0.2	2020年01月02日	• 支持新版异常分析。 • 修复Thrift插件问题。
2.6.0	2019年12月17日	• 支持异步调用链。 • Dubbo/HSFProvider调用参数将被记录。 • 修复现有插件若干问题。
2.5.9.5	2019年11月28日	• 支持jfinal-undertow插件。 • 若干错误修复，包括获取不到Dubbo线程剖析数据等问题。
2.5.9.3	2019年11月25日	• 支持ARMS和链路追踪产品打通。 • 若干错误修复和性能优化。
2.5.9	2019年09月06日	• 修复FastJson拒绝服务漏洞。 • 修改获取网卡IP逻辑。
2.5.8	2019年08月02日	• 支持二元状态报警功能，即针对仅具有是和否、有和无这两种状态的指标设置报警规则。 • 支持国产达梦数据库插件。
2.5.7.2	2019年07月30日	• 支持JVM Metaspace指标。 • 支持自定义要忽略的HTTP状态码。默认情况下，大于400的状态码会计入错误数，您可以自定义大于400但不计入的HTTP状态码。[相关文档]
2.5.7	2019年07月11日	升级依赖的有安全漏洞的FastJson版本。
2.5.6.1	2019年06月28日	• 支持Dubbo/MariaDB插件。 • 自定义配置支持获取SQL绑定值：捕获PrepareStatement参数绑定的变量值，无需重启应用即可生效。[相关文档] • 优化内存和修复若干错误。 • 去除Log4j日志依赖，避免冲突。

版本	发布时间	发布说明
2.5.6	2019年06月07日	• 支持分位数统计功能。 • 优化功能和修复若干错误。
2.5.5	2019年06月03日	• 支持HSF-HTTP调用。 • 优化功能和修复若干错误。
2.5.3	2019年03月15日	• 支持应用运行过程中的线程指标上报。 • 支持Spring-Data-Redis插件。 • 支持Druid数据库连接池插件。
2.5.2	2019年02月21日	• 增加文件句柄数采集。 • 支持GC时间及次数瞬时值上报。 • 支持自定义配置请求入参最大长度。[相关文档]
2.5.1	2019年01月14日	• 支持调用链压缩。[相关文档] • 支持不通过控制台创建应用监控任务的方式。 • 优化功能和修复若干错误。
2.5.0	2018年12月28日	• 支持一键接入，无需重启应用。 • 完善主机监控，支持Windows系统。 • 支持Spring-webflux。 • 优化功能和修复若干错误。
2.4.6	2018年10月26日	• 支持GRPC、THRIFT、XMemcached插件。 • 支持接口调用拓扑展示。 • 支持覆盖前后端的拓扑展示。
2.4.5	2018年09月17日	• 支持Lettuce插件（JRE 1.8+）。 • 支持MongoDB插件。 • 采集异常详细信息。
2.4.4	2018年08月06日	• 支持应用线程剖析数据上报。 • 支持Memcached缓存。 • 支持自定义配置异常过滤。[相关文档]

版本	发布时间	发布说明
2.4.3.1	2018年06月29日	• 支持WebLogic服务器。 • 支持Undertow服务器。 • 优化Agent内存占用。 • 优化Agent启动加载时间。 • 解决JVM监控/主机监控指标可能无法上报问题。
2.4.3	2018年05月18日	• 支持采集消息队列RocketMQ监控指标。 • 支持监控方法自定义。 • 解决限流场景下频繁输出日志的问题。 • 支持自定义配置本地方法堆栈最大长度。[相关文档] • 优化采样功能，不对异常调用链进行采样。
2.4.2	2018年04月19日	• 支持自定义配置信息读取。 • 支持通过SDK方式实时获取链路信息。 • 支持线程、GC次数/耗时等JVM指标采集。 • 支持HSF方法级调用监控。 • 支持主机监控（CPU/物理内存/网络/磁盘）等指标采集。 • 解决Tomcat环境下通过./shutdown.sh停止进程时可能卡住的问题。
2.4.1	2018年03月24日	• 支持JVM监控，如堆内存、非堆内存等指标上报。 • 支持PlayFrameWork 1.4.4版本。 • 支持自定义配置采样率、Agent开关、日志级别、阈值参数等。[相关文档]
2.4.0	2018年02月14日	• 支持PostgreSQL数据库。 • 支持阿里云各地域的ECS与ARMS服务器进行内网通讯。 • 支持ARMS应用监控正式商用。

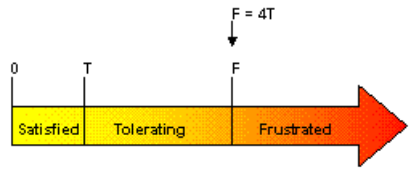
5.5. 关键统计指标说明

本文说明了ARMS应用监控各页面的关键统计指标的含义。

基本概念

本文涉及以下基本概念：

- APDEX性能指数
  APDEX性能指数（Application Performance Index）是一个国际通用的应用性能计算标准。该标准将用户对应用的使用感受定义为三个等级：
  - 满意（0~T）
  - 可容忍（T~4T）
  - 不满意（大于4T）



图片来源：apdex.org
计算公式为：

```
Apdex=(满意数+可容忍数/2)/总样本量
```

ARMS取应用的平均响应时间作为计算指标，并将T定义为500毫秒。

- 实例
  实例是指被监控的应用所部署的机器，以JVM为粒度。例如在下图中，“a3”是一个应用，下方的每一行都是该应用所部署的一台机器，即一个实例。

相关统计页面

- **应用列表**页面
  在控制台左侧导航栏中选择**应用监控 > 应用列表**，即可看到各应用的健康度、请求数、错误数、响应时间、异常数、状态和最近10分钟响应时间曲线。
- **应用总览**页面
  在应用监控列表页面单击应用名称，即可进入**应用总览**页面。在页面顶部选择相应菜单，可以查看其他维度的统计信息。
  - **概览分析**页签
    - 应用提供的服务：请求量和平均响应时长
    - 应用依赖的服务：请求量、平均响应时长、实例数和HTTP-状态码

- 系统信息：CPU、内存和负载
- 统计分析：慢接口调用分析和平均响应时间、异常类型和出现次数

- **拓扑图**页签
  - 应用拓扑图
  - 实例健康： 绿色表示正常，黄色表示警告，红色表示严重。
  - 调用类型：

调用类型	描述	备注
调用本地API	对本地API进行操作的调用	API调用
HTTP入口	客户端使用HTTP协议调用该应用的入口	服务入口调用
调用Dubbo	Dubbo的消费者产生的调用	服务入口调用
调用HSF	HSF服务的消费者产生的调用	服务入口调用
调用HTTP	该调用为该应用对其他服务发起的HTTP调用	服务间调用
提供HSF	HSF的生产者产生的调用	服务间调用
提供Dubbo	Dubbo的生产者产生的调用	服务间调用
调用MySQL	对MySQL进行操作的调用	数据库调用
调用Oracle	对Oracle进行操作的调用	数据库调用
调用Redis	对Redis进行操作的调用	数据库调用

  - 实例IP： 显示该应用下所有实例的IP。
  - 应用每分钟的请求数、响应时间和错误率。

- **3D拓扑**页签
  - QPS：Query Per Second（每秒查询数）
  - RT(ms)：Response Time（响应时间，单位为毫秒）
  - Error：Error QPS（每秒错误查询数）

- **应用详情**页面
  此页面展示当前应用的调用详细信息。选择不同页签，可切换展示实例响应时间、请求数、错误数统计，以及实例概览、SQL分析、异常分析、接口快照等维度的详细分析。
- **接口调用**页面
  此页面展示当前应用所开放的接口的统计信息。选择不同页签，可切换展示实例响应时间、请求数、错误数统计，以及实例概览、SQL分析、异常分析、接口快照等维度的详细分析。
- **数据库调用**页面
  该部分展示应用所关联的数据库调用情况。选择不同页签，可切换展示实例响应时间、请求数、错误数统计，以及实例概览、SQL分析、异常分析等维度的详细分析。

相关页签的关键统计指标说明

- 响应时间：应用、实例调用的平均响应时间，或数据库操作的平均执行响应时间。
- 请求数：应用、实例调用的请求调用次数，或数据库操作的执行次数。
- 错误数：应用、实例调用的错误调用次数，或数据库操作中异常执行次数。
- **概览**页签

上报字段	描述
请求数	应用、实例调用的请求调用次数，或数据库操作的执行次数。
响应时间	应用、实例调用的平均响应时间，或数据库操作的平均执行响应时间。
错误率	（应用、实例调用的异常调用次数，或数据库操作的异常次数）/请求数。

- **SQL分析**页签

上报字段	描述
SQL调用统计	柱状图与左Y轴为数据库请求数统计，折线图与右Y轴为数据库响应时间。
平均耗时	本次数据库调用的平均耗时。
调用次数	该应用此类型数据库调用次数。

- **异常分析**页签

上报字段	描述
异常统计	柱状图为该应用、实例、数据库的异常次数。
异常类型	采集到的抛错类型。
异常详细信息	抛错的详细信息。
平均耗时	本次错误调用的平均耗时。
错误数	该异常类型的错误出现的次数。

- **接口快照**页签

上报字段	描述
耗时	应用、实例的接口的调用耗时。
状态	应用、实例的接口的调用返回状态，绿色表示正常返回，红色表示抛异常。
TraceId	应用、实例调用的索引ID，单击可以跳转到该调用链详情。

5.6. ARMS SDK使用说明

借助ARMS提供的SDK，您可以在业务代码中动态获取TraceId及相关调用链属性。

> **说明** ARMS已支持通过OpenTelemetry Java SDK埋点监控您的应用，建议您优先选择使用OpenTelemetry Java SDK进行埋点。更多信息，请参见通过OpenTelemetry Java SDK进行手工埋点。

前提条件

- 已在ARMS控制台上创建应用监控，并已在Java程序中挂载和启动应用监控的Agent。具体操作，请参见为Java应用手动安装Agent。
- 程序中已引入arms-sdk-1.7.3.jar依赖。

```
<dependency>
    <groupId>com.alibaba.arms.apm</groupId>
    <artifactId>arms-sdk</artifactId>
    <version>1.7.3</version>
</dependency>
```

> **说明** 如果无法获取pom.xml中的依赖，请直接下载arms-sdk-1.7.3.jar。

获取TraceId与RpcId

您可通过以下代码获取TraceId与RpcId。

```
Span span = Tracer.builder().getSpan();
String traceId = span.getTraceId();
String rpcId = span.getRpcId();
```

透传业务自定义标签baggage

若您要透传业务自定义标签，则需要在代码中写入添加和获取自定义标签，具体操作步骤如下：

1. 在业务代码中添加自定义标签baggage。

```
Map<String, String> baggage = new HashMap<String, String>();
baggage.put("key-01", "value-01");
baggage.put("key-02", "value-02");
baggage.put("key-03", "value-03");
Span span = Tracer.builder().getSpan();
span.withBaggage(baggage);
```

2. 在业务代码中获取自定义标签baggage。

```
Span span = Tracer.builder().getSpan();
Map<String, String> baggage = span.baggageItems();
```

为Span设置自定义标签tag

为Span设置自定义标签只会在当前Span中有效，并不会透传。您需要在代码中写入添加和获取自定义的标签，具体操作步骤如下：

1. 在业务代码中为Span添加自定义标签tag，可以添加多个标签。

```
Span span = Tracer.builder().getSpan();
// Add a tag to the Span.
span.setTag("tag-key1", "tag-value1");
span.setTag("tag-key2", "tag-value2");
```

2. 在业务代码中获取Span自定义标签tag。

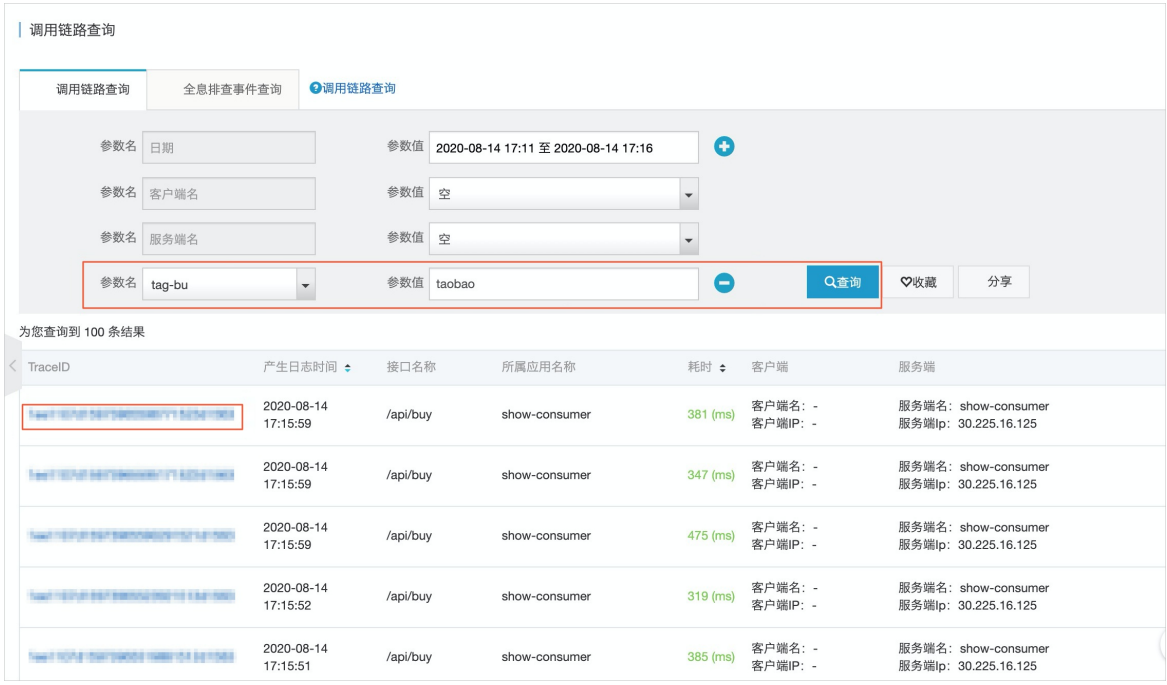
```
Span span = Tracer.builder().getSpan();
// Inspect the Span's tags.
Map<String, String> tags = span.tags();
```

根据自定义标签baggage和tag查询调用链

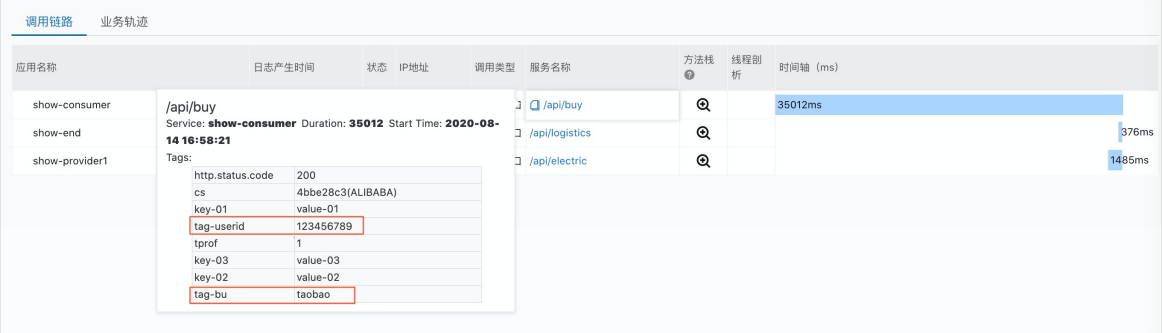
通过Span设置的自定义标签baggage和tag可以用来按标签维度查询调用链。

- baggage上的标签具有透传到下游效果，一般用于业务染色，标签项不建议设置过多。
- tag上的标签只在本Span作用域内有效，可以设置多个业务项。

1. 登录ARMS控制台。
2. 在左侧导航栏选择**应用监控 > 调用链路查询**，并在顶部菜单栏选择目标地域。
3. 在**调用链路查询**页面选择**参数名**，在**参数值**中填入自定义标签，单击**查询**。
4. 在**调用链路查询**列表中单击目标链路的**TraceID**。



5. 在**调用链路**详情页面，鼠标移动至**服务名称**，会显示当前Span对应的Tags信息（baggage内容会自动加入到每个Span的Tags上）。



5.7. Trace Explorer参数说明

本文介绍了Trace Explorer功能常见的参数说明。

此处仅介绍了ARMS应用监控Trace Explorer功能中常见的参数，更多信息，请参见OpenTelemetry的Tracing数据模型。

参数	说明
attributes	用于索引的属性。
duration	耗时，单位为纳秒。
event	事件，包含时间戳、名称、属性等非索引字段。
hostname	主机名。
ip	主机IP地址。
kind	SPAN类型，SPAN类型包括以下几种： • SPAN_KIND_UNSPECIFIED：未指定 • SPAN_KIND_INTERNAL：内部方法 • SPAN_KIND_SERVER：服务端 • SPAN_KIND_CLIENT：客户端 • SPAN_KIND_PRODUCER：生产者 • SPAN_KIND_CONSUMER：消费者
links	当前SPAN和同一Trace或者不同Trace的关系。
parentSpanId	父SPAN分配的ID，表示SPAN之间的关系。
resources	SPAN的资源信息，包括进程ID、JVM版本号等。
serviceName	服务名，用于多租户。
spanId	SPAN分配的ID，表示SPAN之间关系。
spanName	SPAN的名称。
startTime	开始时间，单位为纳秒。

参数	说明
statusCode	SPAN状态，SPAN状态包括以下几种： • STATUS_CODE_UNSET：未设置 • STATUS_CODE_OK：正常 • STATUS_CODE_ERROR：错误
statusMessage	状态附加信息。
traceId	调用链的唯一标识。

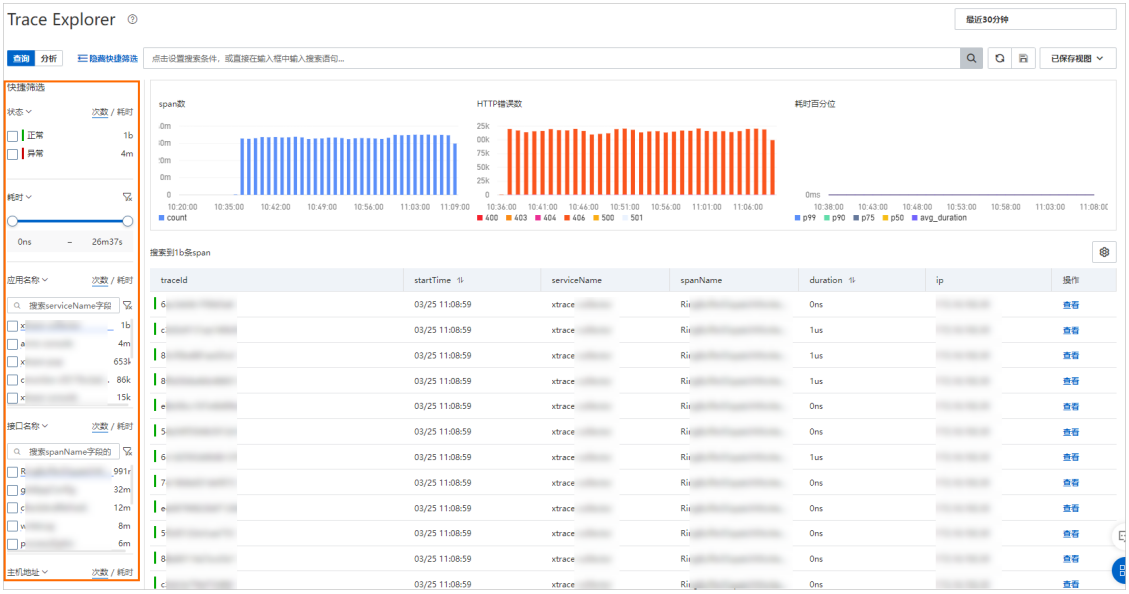
5.8. Trace Explorer查询用法说明

Trace Explorer支持三种查询方式：快捷筛选、查询面板和查询框。查询条件会互相联动，您可以根据个人习惯选择合适的查询方式。

快捷筛选

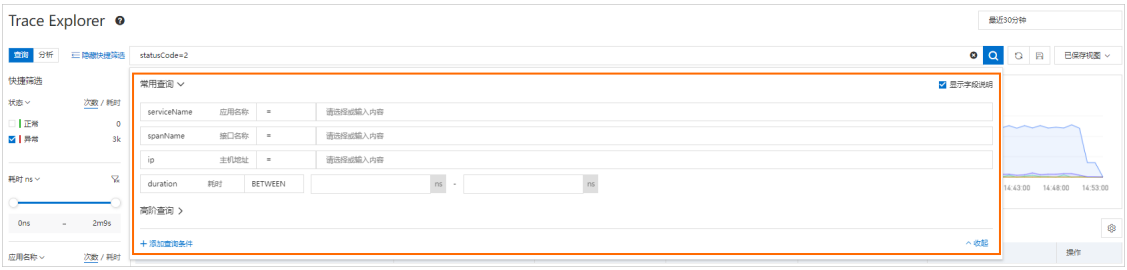
在Trace Explorer页面左侧**快捷筛选**区域，通过状态、耗时、应用名称、接口名称和主机地址维度快速筛选链路。例如选中异常，可以快速筛选出 `statusCode=2` 的异常链路；选中某个应用或接口，可以快速筛选该应用或接口下的链路。

快捷筛选的优势不仅是操作方便，还可以进行多级筛选。通过实时显示当前组合条件下的链路数据分布，帮助您发现潜在的异常特征。



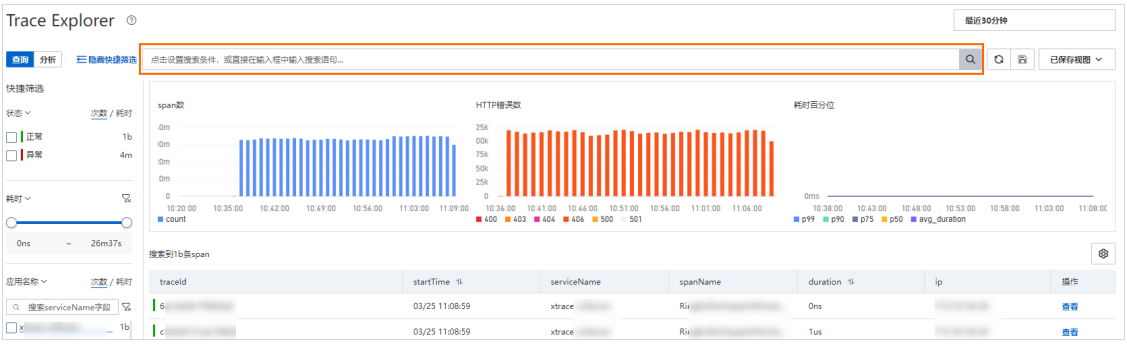
查询面板

在Trace Explorer页面单击右侧顶部文本框，在下拉查询面板中可以指定筛选条件的具体值，并支持添加自定义Attributes筛选条件。



查询框

在Trace Explorer页面查询框中直接输入筛选条件。



Trace Explorer支持简单的查询语法，常用的运算符说明如下，查询参数请参见Trace Explorer参数说明。

运算符	说明
AND	AND运算符，用于组合多个查询条件，表示多重过滤。例如 `serviceName="arms-demo" AND statusCode=2` 表示查询arms-demo应用下的异常链路。
=	等于运算符，查询某字段等于某数值的链路，如果值为字符串类型则需要加上半角双引号（""），数值类型则不需要添加。例如 `serviceName="arms-demo", statusCode=2` 。
>	大于运算符，仅可用于数值类型判断。例如 `duration>10000000` 。
<	小于运算符，仅可用于数值类型判断。例如 `duration<30000000` 。
IN	多选运算符。例如 `ip IN ("127.0.0.1","192.168.0.1")` 表示IP等于127.0.0.1或者192.168.0.1，满足任一条件即可返回。 IN的候选值需要包含在半角圆括号内（()），每个值用半角双引号（""）包裹，多个候选值之间用半角逗号（,）分隔。IN的候选值至少有一个，不能为空。

查询语法示例：

- 查询arms-demo应用下异常链路：

```
serviceName="arms-demo" AND statusCode=2
```

- 查询/createOrder/pc和/createOrder/mobile两个接口大于3s的链路：

```
spanName IN ("/createOrder/pc","/createOrder/mobile") AND duration>3000000000
```

5.9. 应用监控指标说明

本文介绍了ARMS应用监控中常见的指标说明。

业务类指标

公共维度

维度名称	维度Key
服务名称	service
服务PID	pid
机器IP	serverIp
接口	rpc

指标列表

所有访问类型都包含下列指标，执行查询操作时，只需要将 `$callType` 替换为具体的访问类型即可。详细的访问类型，请参见服务访问类型及可用维度。

例如：需要查询HTTP服务的请求数时，只需将 `arms_$callType_requests_count` 修改为 `arms_http_requests_count` 。

指标名称	指标	指标类型	采集间隔	单位	维度
总请求数	arms_$callType_requests_count	Gauge	15s	无	不同服务访问类型维度不同，详细信息，请参见服务访问类型及可用维度。
错误请求数	arms_$callType_requests_error_count	Gauge	15s	无	
总请求耗时	arms_$callType_requests_seconds	Gauge	15s	秒	
慢请求数	arms_$callType_requests_slow_count	Gauge	15s	无	
总请求耗时分位数	arms_$callType_requests_latency_seconds	Summary	15s	秒	仅当服务访问类型为HTTP且开启分位数统计的情况下存在。 quantile分数位维度： • 0.5 • 0.75 • 0.90 • 0.99

JVM指标

公共维度

维度名称	维度Key
服务名称	service
服务PID	pid
机器IP	serverIp

指标列表

指标名称	指标	指标类型	采集间隔	单位	维度
累计GC发生次数	arms_jvm_gc_total	Counter	15s	无	gen gc发生区域： • young：年轻代 • old：老年代
累计GC耗时	arms_jvm_gc_seconds_total	Counter	15s	秒	
两次采集间隔之间的GC次数	arms_jvm_gc_delta	Gauge	15s	无	
两次采集间隔之间的GC耗时	arms_jvm_gc_seconds_delta	Gauge	15s	秒	
JVM线程数	arms_jvm_threads_count	Gauge	15s	无	state线程状态： • blocked：阻塞状态 • live：活跃状态 • daemon：守护状态 • new：初始状态 • dead-lock：死锁状态 • runnable：运行状态 • terminated：终止状态 • timed-wait：超时等待状态 • wait：等待状态

指标名称	指标	指标类型	采集间隔	单位	维度
JVM内存区域初始大小	arms_jvm_mem_init_bytes	Gauge	15s	字节	area区域： • heap：堆区 • nonheap：非堆区 • total：总计 id区域细分： • eden：伊甸区 • old：老年代 • survivor：幸存者区 • metaspace：元数据区 • code cache：代码缓存区 • compressed class space • total：总计
JVM内存区域最大大小	arms_jvm_mem_max_bytes	Gauge	15s	字节	
JVM内存区域使用大小	arms_jvm_mem_used_bytes	Gauge	15s	字节	
JVM内存区域已提交大小	arms_jvm_mem_committed_bytes	Gauge	15s	字节	
JVM内存区域使用比例	arms_jvm_mem_usage_ratio	Gauge	15s	比例（0~1）	
JVM已加载类	arms_class_load_loaded	Counter	15s	无	无
JVM已卸载类	arms_class_load_un_loaded	Counter	15s	无	无
JVM缓存池大小	arms_jvm_buffer_pool_total_bytes	Gauge	15s	字节	id区域： • direct • mapped
JVM缓存池已使用大小	arms_jvm_buffer_pool_used_bytes	Gauge	15s	字节	
JVM缓存池个数	arms_jvm_buffer_pool_count	Gauge	15s	无	
文件描述符打开个数	arms_file_desc_open_count	Gauge	15s	无	无
文件描述符打开比例（已打开数/最大允许打开数）	arms_file_desc_open_ratio	Gauge	15s	比例（0~1）	无

系统指标

公共维度

维度名称	维度Key
服务名称	service
服务PID	pid
机器IP	serverIp

指标列表

指标名称	指标	指标类型	采集间隔	单位
空闲CPU占比	arms_system_cpu_idle	Gauge	15s	百分数
IO等待CPU占比	arms_system_cpu_io_wait	Gauge	15s	百分数
系统CPU占比	arms_system_cpu_system	Gauge	15s	百分数
用户态CPU占比	arms_system_cpu_user	Gauge	15s	百分数
系统负载（1分钟）	arms_system_load	Gauge	15s	无
磁盘空闲大小	arms_system_disk_free_bytes	Gauge	15s	字节
磁盘总大小	arms_system_disk_total_bytes	Gauge	15s	字节
磁盘使用率	arms_system_disk_used_ratio	Gauge	15s	比例（0~1）
内存Buffer大小	arms_system_mem_buffers_bytes	Gauge	15s	字节
内存缓存大小	arms_system_mem_cached_bytes	Gauge	15s	字节
内存空闲大小	arms_system_mem_free_bytes	Gauge	15s	字节
内存交换区空闲大小	arms_system_mem_swap_free_bytes	Gauge	15s	字节
内存交换区大小	arms_system_mem_swap_total_bytes	Gauge	15s	字节

指标名称	指标	指标类型	采集间隔	单位
内存大小	arms_system_mem_total_bytes	Gauge	15s	字节
已用内存大小	arms_system_mem_used_bytes	Gauge	15s	字节
网络接收流量大小	arms_system_net_in_bytes	Gauge	15s	字节
网口发送流量大小	arms_system_net_out_bytes	Gauge	15s	字节
网络入口错误数	arms_system_net_in_err	Gauge	15s	无
网络出口错误数	arms_system_net_out_err	Gauge	15s	无

线程池指标

公共维度

维度名称	维度Key
服务名称	service
服务PID	pid
机器IP	serverIp
线程池名称	name
线程池类型	type

指标列表

指标名称	指标	指标类型	采集间隔	维度
线程池核心线程数	arms_threadpool_core_size	Gauge	15s	无
线程池最大线程数	arms_threadpool_max_size	Gauge	15s	无
线程池活跃线程数	arms_threadpool_active_size	Gauge	15s	无
线程池队列大小	arms_threadpool_queue_size	Gauge	15s	无

指标名称	指标	指标类型	采集间隔	维度
线程池当前大小	arms_threadpool_current_size	Gauge	15s	无
线程池不同状态任务数	arms_threadpool_task_total	Gauge	15s	• status：任务状态 • scheduled：已调度 • completed：完成 • rejected：拒绝

服务访问类型及可用维度

客户端类

- 访问类型
  - http_client
  - hsf_client
  - mq_client
  - kafka_send
  - notify_client
  - dubbo_client
  - grpc_client
  - thrift_client
- 维度
  - parent：上游服务的名称
  - ppid：上游服务PID
  - destId：请求对端扩展信息
  - endpoint：请求对端地址
  - excepType：异常ID
  - excepInfo：异常ID编码规则
  - stackTraceId：异常栈ID

DB类

- 访问类型
  - mysql
  - oracle
  - mariadb
  - postgresql
  - ppas
  - sqlserver
  - mongodb

- dmdb

- 维度
  - parent：上游服务的名称
  - ppid：上游服务PID
  - destId：数据库名称
  - endpoint：数据库地址
  - excepType：异常ID
  - excepInfo：异常ID编码规则
  - stackTraceId：异常栈ID
  - sqlId：SQL语句ID

服务端类

- 访问类型
  - http
  - hsf
  - dubbo
  - user_method
  - grpc
  - thrift

- 维度
  - prpc：上游接口
  - parent：上游服务的名称
  - ppid：上游服务PID
  - endpoint：服务地址
  - excepType：异常ID
  - excepInfo：异常ID编码规则
  - stackTraceId：异常栈ID

客户端类

- 访问类型
  - xxl
  - schedulerx

- 维度
  - prpc：上游接口
  - parent：上游服务的名称
  - ppid：上游服务PID

6.升级探针

对于在ARMS中监控的EDAS应用、容器服务应用和其他类型应用，请按照本文所述的相应方法更新Java探针版本。

如何为EDAS应用更新探针？

如果您需要更新EDAS应用的Java探针版本，重新部署EDAS应用即可。

如何为容器服务应用更新探针？

如果您需要更新容器服务应用的Java探针版本，重启应用Pod即可。

如何为其他类型的应用更新探针？

如果您需要更新除了EDAS应用和容器服务应用之外的其他类型应用的Java探针版本，先卸载探针再重新安装即可。

如需卸载探针，请参见卸载Agent的常见问题。

7.应用监控常见问题

本章节汇总了使用ARMS应用监控时的常见问题。

本页目录

- 如何卸载Docker中Java应用的Agent？

- 其他问题
  - 使用OpenFeign组件的应用在ARMS中数据不完整怎么办？

ARMS Agent和其他APM产品Agent（例如Pinpoint）是否兼容？

ARMS Agent和其他APM产品Agent都不兼容。APM大多是基于ASM框架进行字节码插桩实现的，同时安装两个Agent相当于对您的代码插桩两次，而不同商家的插装代码实现不同，代码冲突可能造成严重的性能问题，因此强烈建议您不要同时安装多个APM Agent。

[回到顶部]

安装Agent后应用启动时报OutOfMemoryError错误怎么办？

请在Java启动命令后面加上堆内存大小配置项，以适当调大JVM参数。以下示例配置项表示堆内存初始值（Xms）为512 M，堆内存最大值（Xmx）为2 G。

> **说明** 请根据实际情况适当调节。对于Tomcat等其他环境，请在配置文件的JAVA_OPTS中加入此参数。

```
-Xms512M
-Xmx2048M
```

如果出现 `OutOfMemoryError: PermGen space` 错误，请添加以下配置。

```
-XX:PermSize=256M
-XX:MaxPermSize=512M
```

如果出现 `OutOfMemoryError: metaspace` 错误，请添加以下配置。

```
-XX:MetaspaceSize=256M
-XX:MaxMetaspaceSize=512M
```

[回到顶部]

如何测试网络连通性？

部署ARMS Agent时，必须确保8883、8443、8442这三个端口的可连通性，Agent才能正常工作。可使用Telnet命令测试目标主机与ARMS服务器网络是否连通。例如，以测试深圳地域的连通性为例，请登录应用所部署的机器，并输入以下命令：

```
telnet arms-dc-sz.aliyuncs.com 8883
telnet arms-dc-sz.aliyuncs.com 8443
telnet arms-dc-sz.aliyuncs.com 8442
```

> **说明** 请根据所在地域替换服务器地址，且服务器地址区分经典网络和公网与VPC网络。

各地域ARMS应用监控服务器地址

地域	经典网络和公网	VPC网络
华东1（杭州）	arms-dc-hz.aliyuncs.com	arms-dc-hz-internal.aliyuncs.com
华东2（上海）	arms-dc-sh.aliyuncs.com	arms-dc-sh-internal.aliyuncs.com
华北1（青岛）	arms-dc-qd.aliyuncs.com	arms-dc-qd-internal.aliyuncs.com
华北2（北京）	arms-dc-bj.aliyuncs.com	arms-dc-bj-internal.aliyuncs.com
华北3（张家口）	arms-dc-zb.aliyuncs.com	arms-dc-zb-internal.aliyuncs.com
华北5（呼和浩特）	dc-cn-huhehaote.arms.aliyuncs.com	dc-cn-huhehaote-internal.arms.aliyuncs.com
华北6（乌兰察布）	dc-cn-wulanchabu.arms.aliyuncs.com	dc-cn-wulanchabu-internal.arms.aliyuncs.com
华南1（深圳）	arms-dc-sz.aliyuncs.com	arms-dc-sz-internal.aliyuncs.com
华南2（河源）	dc-cn-heyuan.arms.aliyuncs.com	dc-cn-heyuan-internal.arms.aliyuncs.com
华南3（广州）	dc-cn-guangzhou.arms.aliyuncs.com	dc-cn-guangzhou-internal.arms.aliyuncs.com
西南1（成都）	dc-cn-chengdu.arms.aliyuncs.com	dc-cn-chengdu-internal.arms.aliyuncs.com
中国（香港）	arms-dc-hk.aliyuncs.com	arms-dc-hk-internal.aliyuncs.com
亚太东南1（新加坡）	arms-dc-sg.aliyuncs.com	arms-dc-sg-internal.aliyuncs.com
亚太东南2（悉尼）	dc-ap-southeast-2.arms.aliyuncs.com	dc-ap-southeast-2-internal.arms.aliyuncs.com
亚太东南3（吉隆坡）	dc-ap-southeast-3.arms.aliyuncs.com	dc-ap-southeast-3-internal.arms.aliyuncs.com
亚太东南5（雅加达）	arms-dc-indonesia.aliyuncs.com	arms-dc-indonesia-internal.aliyuncs.com
亚太东北1（东京）	arms-dc-jp.aliyuncs.com	arms-dc-jp-internal.aliyuncs.com
欧洲中部1（法兰克福）	arms-dc-frankfurt.aliyuncs.com	arms-dc-frankfurt-internal.aliyuncs.com
欧洲西部1（伦敦）	dc-eu-west-1.arms.aliyuncs.com	dc-eu-west-1-internal.arms.aliyuncs.com
美国东部1（弗吉尼亚）	dc-us-east-1.arms.aliyuncs.com	dc-us-east-1-internal.arms.aliyuncs.com

地域	经典网络和公网	VPC网络
美国西部1（硅谷）	arms-dc-usw.aliyuncs.com	dc-us-west-1-internal.arms.aliyuncs.com
亚太南部1（孟买）	dc-ap-south-1.arms.aliyuncs.com	dc-ap-south-1-internal.arms.aliyuncs.com
政务云	arms-dc-gov.aliyuncs.com	arms-dc-gov-internal.aliyuncs.com
杭州金融云	arms-dc-hz-finance.aliyuncs.com	arms-dc-hz-finance-internal.aliyuncs.com
上海金融云	arms-dc-sh-finance.aliyuncs.com	arms-dc-sh-finance-internal.aliyuncs.com
深圳金融云	arms-dc-sz-finance.aliyuncs.com	arms-dc-sz-finance-internal.aliyuncs.com

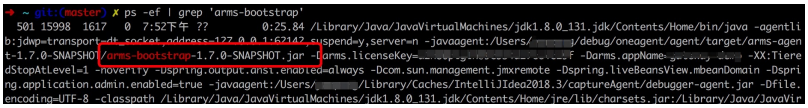
[回到顶部]

如何检查ARMS Agent是否安装成功？

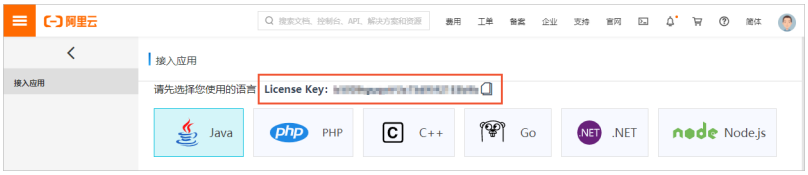
使用**ps**命令查看命令行启动参数中是否成功安装ARMS Agent。

```
ps -ef | grep 'arms-bootstrap'
```

成功安装时，如下图所示：



命令行中的Darms.licenseKey的值必须与ARMS**接入应用**页面显示的License Key一致。



[回到顶部]

ARMS Agent安装成功，为什么控制台上仍无监控数据？

1. 如果Agent日志中出现 `send agent metrics. no metrics.` ，请确认您的应用是否有持续的外部访问请求，包括HTTP请求、HSF请求和Dubbo请求，并确认开发框架是否在ARMS Agent的支持范围内。关于ARMS应用监控对第三方组件和框架的支持情况，请参见应用监控概述。
2. 确认选择的查询时间范围是否正确。请您将查询时间条件设为最近15分钟，然后再次确认是否有监控数据。
3. 如果是通过 `-jar` 命令行启动的，请检查命令行设置，确保-javaagent参数在 `-jar` 之前。以下是一个正确示例。

```
java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.li
censeKey=xxx -Darms.appName=xxx -jar demoApp.jar
```

4. 如果*ArmsAgent/log/*下的日志中出现LicenseKey is invalid.异常，请检查应用所属地域与Agent所属地域是否一致。
5. 如果应用启动之后*ArmsAgent*目录下没有log子目录，则表明*arms-bootstrap-1.7.0-SNAPSHOT.jar*未被成功加载，请检查ArmsAgent安装目录的权限是否正确。
6. 若应用启动时出现以下报错，请您检查*arms-bootstrap-1.7.0-SNAPSHOT.jar*软件包及相应权限是否正确。

```
Error opening zip file or JAR manifest missing: /root/ArmsAgent/arms-bootstrap-1.7.0-SN
APSHOT.jar
Error occurred during initialization of VM
agent library failed to init: instrument
```

7. 如果仍无监控数据，请将Java Agent日志（路径：*ArmsAgent/log*）保存为压缩文件，并联系钉钉服务账号 `arms160804` 解决问题。
8. 检查JDK版本。如果JDK版本为1.8.0_25或者1.8.0_31，可能会出现无法安装Agent的情况，建议您升级对应的JDK版本，或咨询钉钉服务账号 `arms160804` 。

[回到顶部]

ARMS Agent安装成功，但IP显示不正确或不显示怎么办？

1. 首先通过**ifconfig -a**命令确认当前机器的网络配置情况，若为多网卡环境，则Agent采集到的IP地址可能会不符合预期（与网络配置环境有关）。
2. 选择以下方式之一来解决问题。
   - 配置JVM的 `-DEAGLEEYE.LOCAL.IP=10.XX.XX.XX` 参数。

     > **说明** 将 `10.XX.XX.XX` 替换为实际IP地址。

   - 配置Agent获取JVM的 `-DNETWORK.INTERFACE=eth0` 参数，其中 `eth0` 为网卡名称。

[回到顶部]

对于ARMS Agent日志（路径：ArmsAgent/log）错误，有哪些常用排查方法？

LicenseKey is invalid错误：

1. 确保在ARMS中已成功创建该应用，并且确认安装Agent时填写的LicenseKey正确。
2. 由于ARMS产品是多地域（Region）环境，所以还需要确认Agent的下载地址所属的地域与应用所属地域是否一致。

[回到顶部]

如何支持单机多实例？

如需在同一台机器上部署同一应用的多个实例，可以通过-Darms.agentId（逻辑编号，如001、002）参数来区分接入的JVM进程，例如：

```
java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licens
eKey=<LicenseKey> -Darms.appName=<AppName> -Darms.agentId=001 -jar demoApp.jar
```

[回到顶部]

运行接入脚本时出现getcwd相关错误，该如何解决？

若您在运行一键接入Java应用的接入脚本时遇到以下错误：

```
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory Error occurred during initialization of VM java.lang.Error: Properties init: Could not determine current working directory. at java.lang.System.initProperties(Native Method) at java.lang.System.initializeSystemClass(System.java:1119)
```

可能是因为执行脚本过程中当前目录被误删，解决方法为执行 `cd` 后重新运行脚本。

[回到顶部]

使用一键接入方式安装Agent后，在哪里查看日志？

日志的默认目录为*/root/.arms/supervisor/logs/*，若此目录下没有日志，请执行命令 `ps -ef |grep arms` 查看日志所在目录。

[回到顶部]

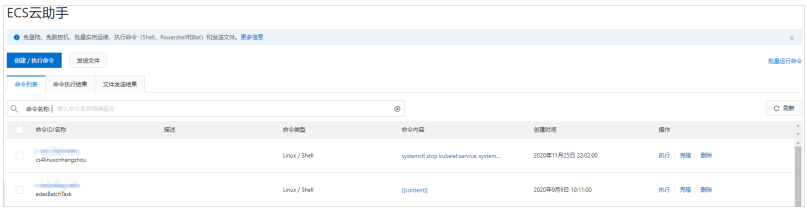
Agent安装失败怎么处理？

1. 确保您的ECS实例可以访问公网，且能够访问所在地域的Agent下载链接。

地域	公网地址
华东1（杭州）	arms-apm-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/ArmsAgent.zip
华东2（上海）	arms-apm-cn-shanghai.oss-cn-shanghai.aliyuncs.com/ArmsAgent.zip
华北1（青岛）	arms-apm-cn-qingdao.oss-cn-qingdao.aliyuncs.com/ArmsAgent.zip
华北2（北京）	arms-apm-cn-beijing.oss-cn-beijing.aliyuncs.com/ArmsAgent.zip
华北3（张家口）	arms-apm-cn-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/ArmsAgent.zip
华北5（呼和浩特）	arms-apm-cn-huhehaote.oss-cn-huhehaote.aliyuncs.com/ArmsAgent.zip
华北6（乌兰察布）	arms-apm-cn-wulanchabu.oss-cn-wulanchabu.aliyuncs.com/ArmsAgent.zip
华南1（深圳）	arms-apm-cn-shenzhen.oss-cn-shenzhen.aliyuncs.com/ArmsAgent.zip
华南2（河源）	arms-apm-cn-heyuan.oss-cn-heyuan.aliyuncs.com/ArmsAgent.zip
华南3（广州）	arms-apm-cn-guangzhou.oss-cn-guangzhou.aliyuncs.com/ArmsAgent.zip
西南1（成都）	arms-apm-cn-chengdu.oss-cn-chengdu.aliyuncs.com/ArmsAgent.zip
中国（香港）	arms-apm-cn-hongkong.oss-cn-hongkong.aliyuncs.com/ArmsAgent.zip
亚太东南1（新加坡）	arms-apm-ap-southeast-1.oss-ap-southeast-1.aliyuncs.com/ArmsAgent.zip
亚太东南2（悉尼）	arms-apm-ap-southeast-2.oss-ap-southeast-2.aliyuncs.com/ArmsAgent.zip

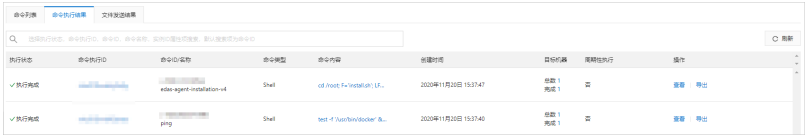
地域	公网地址
亚太东南3（吉隆坡）	arms-apm-ap-southeast-3.oss-ap-southeast-3.aliyuncs.com/ArmsAgent.zip
亚太东南5（雅加达）	arms-apm-ap-southeast-5.oss-ap-southeast-5.aliyuncs.com/ArmsAgent.zip
亚太东北1（东京）	arms-apm-ap-northeast-1.oss-ap-northeast-1.aliyuncs.com/ArmsAgent.zip
欧洲中部1（法兰克福）	arms-apm-eu-central-1.oss-eu-central-1.aliyuncs.com/ArmsAgent.zip
欧洲西部1（伦敦）	arms-apm-eu-west-1.oss-eu-west-1.aliyuncs.com/ArmsAgent.zip
美国东部1（弗吉尼亚）	arms-apm-us-east-1.oss-us-east-1.aliyuncs.com/ArmsAgent.zip
美国西部1（硅谷）	arms-apm-us-west-1.oss-us-west-1.aliyuncs.com/ArmsAgent.zip
亚太南部1（孟买）	arms-apm-ap-south-1.oss-ap-south-1.aliyuncs.com/ArmsAgent.zip
政务云	arms-apm-cn-north-2-gov-1.oss-cn-north-2-gov-1.aliyuncs.com/ArmsAgent.zip

2. 确保您的ECS实例可以访问控制台。
   - 中国ARMS控制台
   - 新加坡ARMS控制台
3. 登录ECS控制台，并完成以下检查工作。
   i. 在左侧导航栏中选择**运维与监控 > 发送命令/文件（云助手）**。
   ii. 在**ECS云助手**页面**命令列表**页签的搜索框中选择**命令名称**，输入 `InstallJavaAgent` 命令并按Enter键。



> **说明** 若查找结果不存在，请联系ARMS钉钉服务账号 `arms160804` 。

   iii. 在**ECS云助手**页面的**命令执行结果**页签的搜索栏输入 `InstallJavaAgent` 命令的命令ID，在查找结果中单击该记录右侧**操作**列的**查看**，即可查看 `InstallJavaAgent` 命令是否执行成功。若未执行成功，请根据详细执行结果排查问题。如ECS磁盘满、未安装Agent等问题，可以通过清理磁盘或安装Agent来解决。若不能解决，请将详细执行结果反馈给ARMS钉钉服务账号 `arms160804` 。



[回到顶部]

安装Agent后ECS实例进程信息不准确怎么办？

若您在ECS上成功安装Agent后没有出现ECS实例进程信息，或者ECS实例进程信息不准确，请单击ECS实例左侧的-号并单击+号刷新数据。若不能解决，请联系ARMS钉钉服务账号 `arms160804` 。

[回到顶部]

ECS实例中的进程启动应用监控失败怎么办？

请检查ECS实例的 `/root/.arms/supervisor/logs/arms-supervisor.log` 日志中是否有日志级别为Error的信息，并根据此信息排查。若不能解决，请联系ARMS钉钉服务账号 `arms160804` 。

[回到顶部]

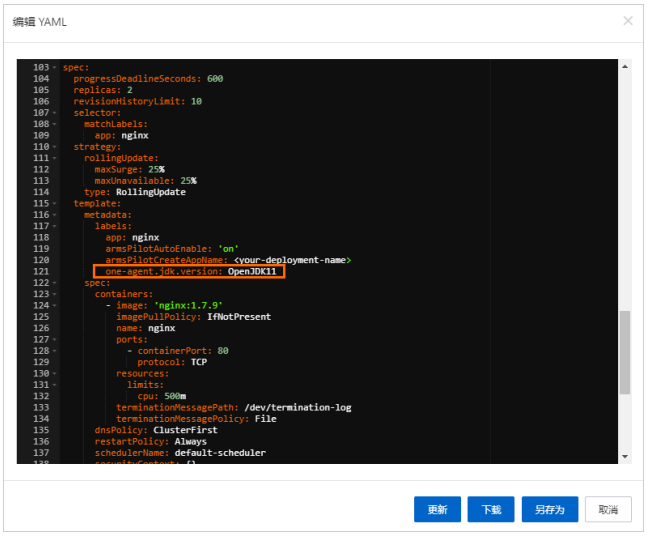
为什么容器服务K8s集群中JDK11版本的Java应用安装Agent后，应用监控没有数据呢？

可能原因：应用的YAML文件中未添加JDK版本参数。

解决方案：执行以下步骤在应用的YAML文件中添加JDK版本参数。

1. 登录容器服务管理控制台。
2. 在左侧导航栏选择**集群**，在**集群列表**页面上的目标集群右侧操作列单击**应用管理**。
3. 在**无状态**页面顶部选择您的应用所在的命名空间，然后在目标应用右侧选择**更多 > 查看Yaml**。
4. 在**编辑YAML**对话框的*spec > template > metadata > labels*层级下添加以下参数。

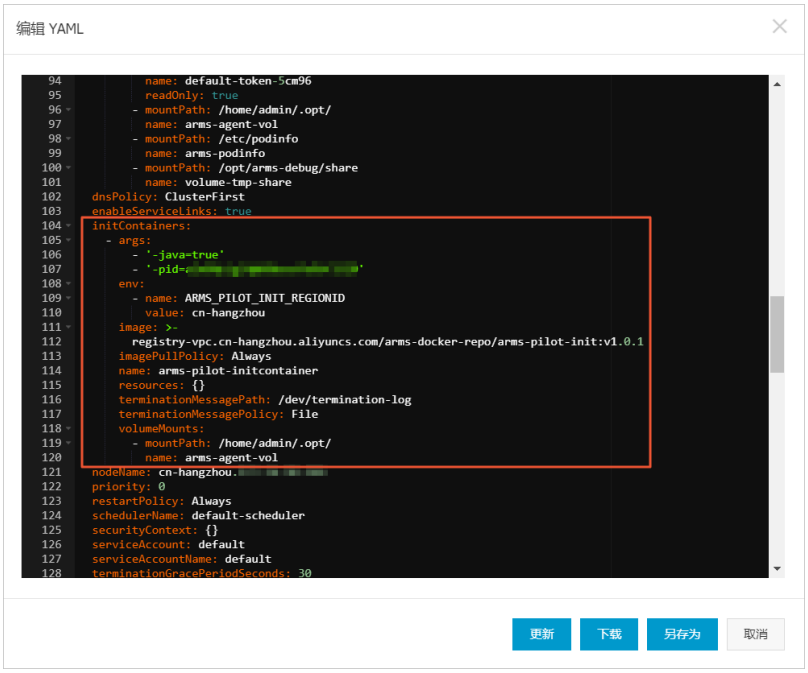
```
one-agent.jdk.version: "OpenJDK11"
```



[回到顶部]

为什么容器服务K8s集群中的Java应用安装Agent后，应用监控没有数据呢？

1. 登录容器服务管理控制台。
2. 在左侧导航栏选择**集群**，在**集群列表**页面上的目标集群右侧操作列单击**应用管理**。
3. 在**容器组**页签顶部选择您的应用所在的命名空间，然后单击目标应用右侧单击**编辑**。
4. 在**编辑YAML**对话框中查看YAML文件中是否存在initContainers。



- 如果不存在，则说明未被注入arms-init-container，执行步骤5。
- 如果存在，则说明已被注入arms-init-container，执行步骤8。

5. 在**容器组**页签顶部选择命名空间为**ack-onepilot**。查看Pod列表中是否存在名称前缀为**ack-onepilot**的Pod。
   - 如果存在，则执行步骤6。
   - 如果不存在，则在应用市场中安装ack-onepilot。具体操作，请参见安装ARMS应用监控组件。
6. 在**无状态**或**有状态**页签目标应用右侧**操作**列中选择**更多 > 查看Yaml**，在**编辑YAML**对话框查看YAML文件中是否存在以下Labels注解。

```
labels:
  armsPilotAutoEnable: "on"
  armsPilotCreateAppName: "<your-deployment-name>"    //请将<your-deployment-name>替换为您的应用名称。
  one-agent.jdk.version: "OpenJDK11"    //如果应用的JDK版本是JDK11，则需要配置此参数。
```

   - 如果存在，则执行步骤7。
   - 如果不存在，则在**编辑YAML**对话框中的*spec.template.metadata*层级下添加以上Labels注解，然后单击**更新**。
7. 在**容器组**页签目标应用右侧单击**日志**，查看ack-onepilot的Pod日志是否报STS错误，即提示`"Message":"STS错误"`。
   - 如果报STS错误，则需为应用所在集群授权，并重启应用所在Pod。具体操作，请参见为容器服务Kubernetes版授权。
   - 如果未报STS错误，则联系ARMS钉钉服务账号：arms160804。
8. 在**容器组**页签目标应用右侧单击**编辑**，在**编辑YAML**对话框中查看YAML文件中是否存在以下javaagent参数。

```
-javaagent:/home/admin/.opt/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar
```

   - 如果存在，则单击**容器组**页签右侧的**终端**进入**命令行**页面，执行以下命令查看是否存在以.log为后缀

的日志文件，然后联系ARMS钉钉服务账号：arms160804。

```
cd /home/admin/.opt/ArmsAgent/logs
```

- 如果不存在，则联系ARMS钉钉服务账号：arms160804。

[回到顶部]

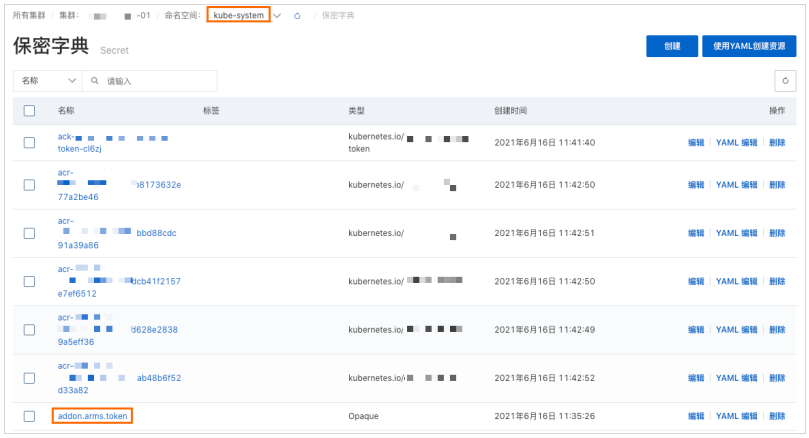
容器服务K8s集群中的应用安装Agent失败怎么处理？

可能原因：目标集群中不存在ARMS Addon Token。

解决方案：

步骤一：查看集群是否存在ARMS Addon Token。

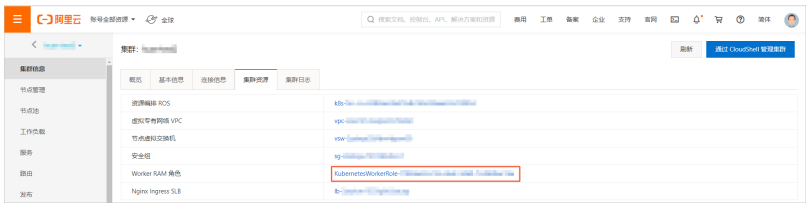
1. 登录容器服务管理控制台，在**集群列表**页面，单击目标集群名称进入集群详情页。
2. 在左侧导航栏选择**配置管理 > 保密字典**。
3. 在顶部选择**命名空间**为**kube-system**，查看addon.arms.token是否存在。



步骤二：为容器服务Kubernetes版授予ARMS资源的访问权限。

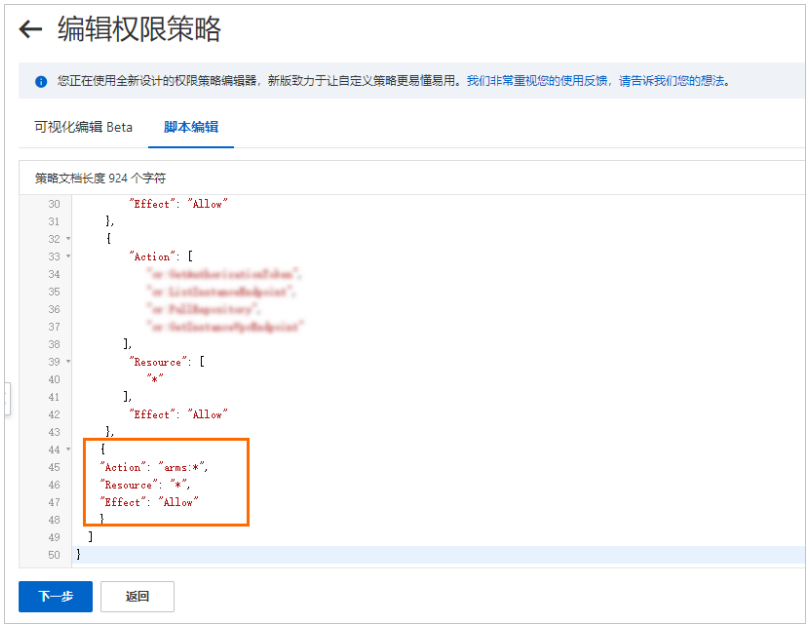
1. 登录容器服务管理控制台。
2. 在左侧导航栏选择**集群**，在**集群列表**页面，单击目标集群名称或其右侧**操作**列的**详情**。

3. 在目标集群的**集群信息**页面上单击**集群资源**页签，然后单击**Worker RAM角色**右侧的链接。



4. 在访问控制RAM控制台的RAM角色管理页面上，单击**权限管理**页签上的权限策略名称。
5. 在**策略内容**页签上单击**修改策略内容**，并将以下内容添加到**策略内容**中，最后单击**确定**。

```
{
   "Action": "arms:*",
   "Resource": "*",
   "Effect": "Allow"
}
```



[回到顶部]

为什么开源K8s集群中JDK11版本的Java应用安装Agent后，应用监控没有数据呢？

可能原因：应用的YAML文件中未添加JDK版本参数。

解决方案：在YAML文件中的*spec > template > metadata > labels*层级下添加以下内容。

```
one-agent.jdk.version: "OpenJDK11"
```

[回到顶部]

ACK集群如何跨区域上报数据？

1. 在容器集群arms-pilot Namespace下的Deployment中添加ARMS_REPORT_REGION环境变量，值为ARMS所支持的RegionId（例如cn-hangzhou、cn-beijing）。
2. 重启现有应用或部署一个新的应用，完成跨区域上报。

> **说明** 当添加环境变量后，该集群下所有应用均会跨区域上报到步骤一中指定的地域。

应用无法正常启动怎么办？

请执行以下命令查看ack-onepilot的日志信息来排查问题。

```
k logs -f ack-onepilot-ack-onepilot-XXXX-XXXX -n ack-onepilot
```

> **说明** XXXX-XXXX为随机信息，您可通过执行命令 `k get pod -n ack-onepilot` 来获取。

[回到顶部]

如何查看Agent日志？

在Kubernetes集群Worker机上查看*/home/admin/.opt/ArmsAgent/logs/xxxx.log*。

[回到顶部]

以通用方式安装Agent的普通Java应用如何更改应用名称？

普通Java应用是指除了部署在阿里云ECS实例上的应用以外的Java应用。如果您是以通用方式安装Agent的，则Agent目录就是您自定义的位置。

您可以在Agent目录的Version文件中查看Agent版本。例如，`2.5.8_cf020486_20190816150025` 表示Agent版本为2.5.8，Agent发布时间为2019年8月16日。

- 如果您的Agent版本低于2.5.8.1，请先卸载Agent再重新安装，并在重新安装时使用新的应用名称。
  - 手动安装的Agent的卸载方法，请参见如何卸载以通用方式安装的Agent。
  - 以脚本自动安装的Agent的卸载方法，请参见如何卸载以快速方式安装的Agent。
  - ECS应用的Agent卸载方法，请参见如何卸载ECS中Java应用的Agent。
- 如果您的Agent版本为2.5.8.1或以上，则可在不重装Agent的情况下更改Java应用名称，请参见以下操作步骤：

  > **说明** 2019年8月20日之后下载的Agent默认都支持该功能。

  1. 请使用以下方法之一下载并解压Supervisor。

     > **说明** 请根据您的应用所在的地域替换下载地址中的地域。

     - 公网地址

       ```
       wget http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/ArmsSupervisor.zip -O Arms
       Supervisor.zip
       unzip ArmsSupervisor.zip
       ```

     - VPC地址（公网地址无法下载时使用）

       ```
       wget http://arms-apm-hangzhou.oss-cn-hangzhou-internal.aliyuncs.com/ArmsSupervisor.zi
       p -O ArmsSupervisor.zip
       unzip ArmsSupervisor.zip
       ```

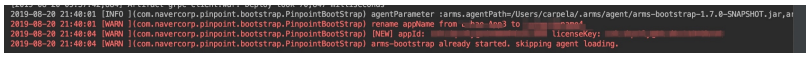
  2. 运行以下命令更改应用名称。

     ```
     cd ArmsSupervisor
     ./attach.sh </path/to/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar> <PID> <NewAppName>
     <LicenseKey>
     ```

     - *</path/to/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar>*：指向Agent下同名文件的路径。
     - *<PID>*：目标进程ID，可通过 `jps/ps` 命令获取。
     - *<NewAppName>*：新的应用名称。

- *<LicenseKey>*：ARMS应用监控中应用的LicenseKey，可从控制台获取。

如果应用的标准输出打印以下日志，则说明应用名称更改成功。



[回到顶部]

以快速方式安装Agent的普通Java应用如何更改应用名称？

如果您是以快速方式安装Agent的，则Agent目录为*~/.arms/supervisor/agent*。注意，使用的账号必须与应用相同。

请按照以下步骤更改应用名称。

1. 登录应用所在的机器，并运行以下命令。

```
cd ~/.arms/supervisor
./cli.sh <LicenseKey> <NewAppName>
```

   - *<LicenseKey>*：ARMS应用监控中应用的LicenseKey，可从控制台获取。
   - *<NewAppName>*：新的应用名称。

2. 从程序列出的所有Java进程中选择正确的进程。

> **说明** 如果只有一个应用进程则默认选中该进程。

3. 打开*~/.arms/attach.info*文件，将需要修改的旧应用名称修改为新应用名称并保存文件。

> **警告** 修改文件时切勿多加空格等内容，否则会因为与前述步骤中修改的新应用名称不符而导致修改失败。

   应用名称更改成功后稍等片刻，旧名称的应用下将不再有监控数据上报，且新名称的应用下将有监控数据上报。

部署在ECS中的Java应用如何更改应用名称？

如果是部署在ECS实例中的Java应用，则Agent目录为*/.arms/agent*。

请按照以下步骤更改应用名称。

1. 登录应用所在的机器，并使用root账号运行以下命令。

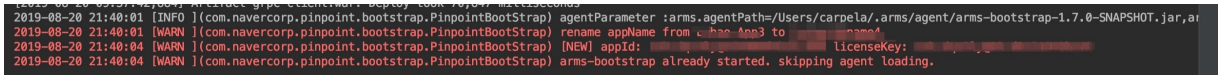
```
su <USER> -c "./attach.sh /.arms/agent/arms-bootstrap-1.7.0-SNAPSHOT.jar <PID> <NewAppName> <LicenseKey>"
```

   - *<PID>*：目标进程ID，可通过`jps/ps`命令获取。
   - *<NewAppName>*：新的应用名称。
   - *<LicenseKey>*：ARMS应用监控中应用的LicenseKey，可从控制台获取。

2. 打开*~/.arms/attach.info*文件，将需要修改的旧应用名称修改为新应用名称并保存文件。

> **警告** 修改文件时切勿多加空格等内容，否则会因为与前述步骤中修改的新应用名称不符而导致修改失败。

如果应用的标准输出打印以下日志，则说明应用名称更改成功。



部署在容器服务K8s集群中的Java应用如何更改应用名称？

您可以在Agent目录的Version文件中查看Agent版本。例如，`2.5.8_cf020486_20190816150025`表示Agent版本为2.5.8，Agent发布时间为2019年8月16日。

- 如果您的Agent版本低于2.5.8.1，请先卸载Agent再重新安装，并在重新安装时使用新的应用名称。
  - 手动安装的Agent的卸载方法，请参见如何卸载以通用方式安装的Agent。
  - 以脚本自动安装的Agent的卸载方法，请参见如何卸载以快速方式安装的Agent。
  - ECS应用的Agent卸载方法，请参见如何卸载ECS中Java应用的Agent。
- 如果您的Agent版本为2.5.8.1或以上，则可在不重装Agent的情况下更改Java应用名称，请参见以下操作步骤：

> **说明** 2019年8月20日之后下载的Agent默认都支持该功能。

修改Deployment内的armsPilotCreateAppName参数并重启Pod即可。

应用名称更改成功后稍等片刻，旧名称的应用下将不再有监控数据上报，且新名称的应用下将有监控数据上报。

[回到顶部]

部署在EDAS上的Java应用如何更改应用名称？

目前不支持为部署在EDAS上的Java应用更改应用名称。

[回到顶部]

如何卸载以通用方式安装的Agent？

以通用方式安装Agent是指使用手动方式为Java应用安装Agent，具体操作，请参见为Java应用手动安装Agent。卸载以通用方式安装的Agent的操作步骤如下所示：

1. 当您不需要使用ARMS监控您的Java应用时，删除上述安装文档步骤8添加的AppName、LicenseKey相关的所有参数。
2. 重启Java应用。

[回到顶部]

如何卸载以快速方式安装的Agent？

以快速方式安装Agent是指使用一键接入脚本为Java应用安装Agent，具体操作，请参见使用脚本为Java应用快速安装探针。卸载以快速方式安装的Agent的操作步骤如下所示：

1. 当您不需要使用ARMS监控您的Java应用时，执行`jps -l`命令查看所有进程，并在执行结果中找到`com.alibaba.mw.arms.apm.supervisor.daemon.Daemon`对应的进程号。

   在本示例中，`com.alibaba.mw.arms.apm.supervisor.daemon.Daemon`对应的进程号为：62857。

```
➜ ~ jps -l
62800 org.apache.catalina.startup.Bootstrap
62857 com.alibaba.mw.arms.apm.supervisor.daemon.Daemon
5411
62799 org.jetbrains.jps.cmdline.Launcher
67809 sun.tools.jps.Jps
```

2. 执行命令 `kill -9 <进程号>` 。

   例如： `kill -9 62857` 。

3. 执行 `rm -rf /.arms /root/.arms` 。
4. 重启您的应用。

[回到顶部]

如何卸载ECS中Java应用的Agent？

1. 当您不需要使用ARMS监控您的Java应用时，执行 `jps -l` 命令查看所有进程，并在执行结果中找到 `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` 对应的进程号。

   在本示例中， `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` 对应的进程号为：62857。

   ```
   ➜ ~ jps -l
   62800 org.apache.catalina.startup.Bootstrap
   62857 com.alibaba.mw.arms.apm.supervisor.daemon.Daemon
   5411
   62799 org.jetbrains.jps.cmdline.Launcher
   67809 sun.tools.jps.Jps
   ```

2. 执行命令 `kill -9 <进程号>` 。

   例如： `kill -9 62857` 。

3. 执行 `rm -rf /.arms /root/.arms` 。
4. 重启您的应用。

[回到顶部]

如何卸载容器服务K8s集群中Java应用的Agent？

当您不需要使用ARMS监控容器服务Kubernetes版集群中的Java应用时，可按照以下步骤卸载Agent：

1. 登录容器服务管理控制台。
2. 在左侧导航栏单击**集群**，在**集群列表**页面上的目标集群右侧**操作**列单击**应用管理**。
3. 在左侧导航栏选择**应用** > **Helm**。
4. 在**Helm**页面，ARMS Agent对应的发布名称**arms-pilot**右侧**操作**列，单击**删除**。
5. 在**删除应用**对话框单击**确定**。
6. 重启您的业务Pod。

[回到顶部]

如何卸载开源K8s环境中Java应用的Agent？

1. 当您不需要再监控开源Kubernetes环境中的Java应用时，可执行以下命令卸载arms-pilot。

   ```
   helm del --purge arms-pilot
   ```

2. 重启您的业务Pod。

[回到顶部]

如何卸载Docker中Java应用的Agent？

1. 当您不需要再监控Docker集群中的Java应用时，可以删除安装文档的步骤1所编辑的*Dockerfile*内容。

2. 运行**docker build**命令来构建镜像。
3. 运行**docker run**命令来启动镜像。

[回到顶部]

使用OpenFeign组件的应用在ARMS中数据不完整怎么办？

若您的OpenFeign应用接入ARMS应用监控后，出现数据不完整、看不到下游应用的数据等情况，可能的原因是OpenFeign组件默认开启了使用RxJava异步框架的Hystrix，而ARMS不支持异步框架。

> **说明** 本文仅限于ARMS应用监控Java Agent版本低于2.6.0的场景，2.6.0及以上版本已支持异步框架。

您可以通过关闭Hystrix并配置OkHttp请求类来解决此类问题：

1. 在*pom.xml*文件中添加以下依赖。

```
<!-- OKHttp对Feign支持 -->
<dependency>
    <groupId>io.github.openfeign</groupId>
    <artifactId>feign-okhttp</artifactId>
</dependency>
```

2. 在SpringCloud配置文件中添加以下配置。

```
feign.okhttp.enabled: true
feign.hystrix.enabled: false
```

3. 配置OkHttp请求类。

```
@Configuration
@ConditionalOnClass(Feign.class)
@AutoConfigureBefore(FeignAutoConfiguration.class)
public class FeignClientOkHttpConfiguration {
    @Bean
    public OkHttpClient okHttpClient() {
        return new OkHttpClient.Builder()
                // 连接超时
                .connectTimeout(20, TimeUnit.SECONDS)
                // 响应超时
                .readTimeout(20, TimeUnit.SECONDS)
                // 写超时
                .writeTimeout(20, TimeUnit.SECONDS)
                // 是否自动重连
                .retryOnConnectionFailure(true)
                // 连接池
                .connectionPool(new ConnectionPool())
                .build();
    }
```

[回到顶部]

8.故障排除

8.1. ARMS Agent安装成功，为什么控制台上仍无监控数据？

Condition

ARMS Agent安装成功，控制台上仍无监控数据。

Cause

应用无持续的外部请求访问、应用所属地域和ARMS Agent所属地域不一致、或者ARMS Agent安装目录权限不正确等原因都有可能导致控制台无监控数据。

操作步骤

1. 如果Agent日志中出现“send agent metrics. no metrics.”，请确认您的应用是否有持续的外部请求访问，包括HTTP请求、HSF请求和Dubbo请求，并确认开发框架是否在ArmsAgent的支持范围内。关于ARMS应用监控对第三方组件和框架的支持情况，请参见ARMS应用监控支持的Java组件和框架。
2. 确认选择的查询时间范围是否正确。请您将查询时间条件设为最近15分钟，然后再次确认是否有监控数据。
3. 如果是通过 `-jar` 命令行启动的，请检查命令行设置，确保-javaagent参数在 `-jar` 之前。

```
java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey=xxx -Darms.appName=xxx -jar demoApp.jar
```

4. 如果ArmsAgent/log/ 下的日志中出现LicenseKey is invalid.异常，请您检查应用所属地域与Agent所属地域是否一致。
5. 如果应用启动之后ArmsAgent目录下无log子目录，是由于arms-bootstrap-1.7.0-SNAPSHOT.jar未被成功加载导致的，请您检查ArmsAgent安装目录的权限是否正确。
6. 若应用启动时出现以下报错，请您检查arms-bootstrap-1.7.0-SNAPSHOT.jar软件包及相应权限是否正确。

```
Error opening zip file or JAR manifest missing : /root/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar
Error occurred during initialization of VM
agent library failed to init: instrument
```

7. 如果仍无监控数据，请打包Java探针日志（路径：ArmsAgent/log），并联系钉钉服务账号 `arms160804` 解决问题。
8. 检查JDK版本。

   如果JDK版本为1.8.0_25或者1.8.0_31，可能会出现无法安装探针的情况，建议您升级对应的JDK版本，或咨询钉钉服务账号 `arms160804` 。

8.2. 为什么容器服务Kubernetes版集群中的Java应用安装Agent后，应用监控没有数据呢？

Condition

阿里云容器服务Kubernetes版集群中的Java应用安装Agent后，应用监控没有数据。

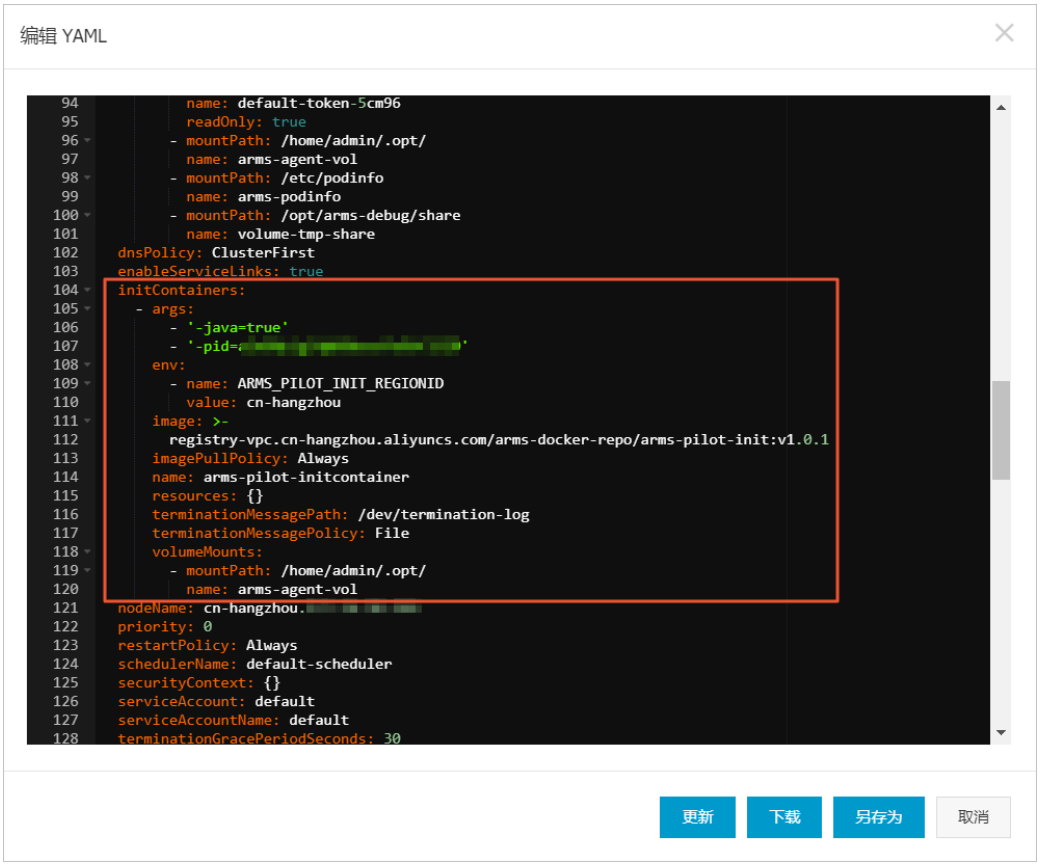
Cause

应用所在Pod未被注入arms-init-container、应用YAML文件中无Labels注解以及STS服务未正确授权等原因都可能导致应用监控无数据。

Remedy

操作步骤

1. 登录容器服务管理控制台。
2. 在左侧导航栏选择**集群**，在**集群列表**页面上的目标集群右侧**操作**列单击**应用管理**。
3. 在**容器组**页签顶部选择您的应用所在的命名空间，然后单击目标应用右侧单击**编辑**。
4. 在**编辑YAML**对话框中查看YAML文件中是否存在initContainers。



   - 如果不存在，则说明未被注入arms-init-container，执行步骤5。
   - 如果存在，则说明已被注入arms-init-container，执行步骤8。
5. 在**容器组**页签顶部选择命名空间为**ack-onepilot**。查看Pod列表中是否存在名称前缀为**ack-onepilot**的Pod。
   - 如果存在，则执行步骤6。
   - 如果不存在，则在应用市场中安装ack-onepilot。具体操作，请参见安装ARMS应用监控组件。
6. 在**无状态**或**有状态**页签目标应用右侧**操作**列中选择**更多 > 查看Yaml**，在**编辑YAML**对话框查看YAML文件中是否存在以下Labels注解。

```
labels:
  armsPilotAutoEnable: "on"
  armsPilotCreateAppName: "<your-deployment-name>"    //请将<your-deployment-name>替换为您的应用名称。
  one-agent.jdk.version: "OpenJDK11"    //如果应用的JDK版本是JDK11，则需要配置此参数。
```

- 如果存在，则执行步骤7。
- 如果不存在，则在**编辑YAML**对话框中的*spec > template > metadata*层级下添加以上Labels注解，然后单击**更新**。

7. 在**容器组**页签目标应用右侧单击**日志**，查看ack-onepilot的Pod日志是否报STS错误，即提示 `"Message":"STS错误"` 。
   - 如果报STS错误，则需为应用所在集群授权，并重启应用所在Pod。具体操作，请参见为容器服务Kubernetes版授权。
   - 如果未报STS错误，则联系ARMS钉钉服务账号：arms160804。
8. 在**容器组**页签目标应用右侧单击**编辑**，在**编辑YAML**对话框中查看YAML文件中是否存在以下javaagent参数。

   ```
   -javaagent:/home/admin/.opt/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar
   ```

   - 如果存在，则单击**容器组**页签右侧的**终端**进入**命令行**页面，执行以下命令查看是否存在以.log为后缀的日志文件，然后联系ARMS钉钉服务账号：arms160804。

     ```
     cd /home/admin/.opt/ArmsAgent/logs
     ```

   - 如果不存在，则联系ARMS钉钉服务账号：arms160804。