Apusic 金蝶天燕

生成时间：2026-05-19

用户手册

金蝶Apusic应用性能管理v1.0

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 前言

2 产品简介

金蝶Apusic应用性能管理软件实现对数据中心、云计算 (公有云、私有云和混合云)、以及容器云等环境中各类单体架构应用、分布式应用以及微服务架构应用进行非侵入式的实时观测，获取服务端性能数据,并通过对业务应用从代码层和环境层帮助用户定位和诊断性能问题，并进行深入分析和优化，改善用户体验。

3 范围和读者

本文面向AAPM产品使用用户、应用开发工程师及应用系统运维开发工程师。

阅读本文第二章和第三章，您可以从宏观上对金蝶Apusic应用性能管理产品有所认识，如果您想全面了解金蝶Apusic应用性能管理产品的全貌，建议您通篇阅读全文。

4 文档导航

- 快速开始

> *以Java应用性能监控为例，快速上手使用AAPM产品。*

- 服务

> *简要介绍当前所有被监控的服务（应用）列表信息。*

- 服务概览

> *详细介绍服务的概要性能指标。*

- 链路拓扑

> *详细介绍服务全局链路拓扑的功能及使用。*

- 性能监控

> *详细介绍端点、关键端点、数据库、JVM性能监控的功能及使用。*

- 性能剖析

> *实时针应用服务的端点进行线程性能剖析，可以参考该节内容的使用说明。*

- 探针下载及管理

> *简要介绍各种语言的探针多版本集中管理。*

5 约定与术语

一些约定的缩略词诠释：

- APM

> *应用性能管理（Application Performance Management）*

- AIOps

> *智能运维（Artificial Intelligence for IT Operations）*

- Apdex

> *性能指数，（Application Performance Index）*

- DEM

> *数字化体验监控*

6 快速开始

本章介绍如何快速开始使用金蝶Apusic应用性能管理软件。

以启动监控一个java应用监控的过程为例。在探针下载模块选择Java探针（目前提供语言安装包有java、php、net），点击下载对应的探针并解压，利用探针自动收集所需的指标，对服务、实例、端点三个维度进行追踪的原理。在服务程序中添加 apusic-apm-agent探针，设置启动参数的方式检测服务，添加成功后重启服务。

访问APM服务可以查看监控的服务，服务实例，端点指标分析，服务拓扑图分析、服务，服务实例和端点依赖关系分析，检测到慢速服务和端点，性能优化，分布式跟踪和上下文传播、数据库访问指标、检测慢速数据库访问语句。

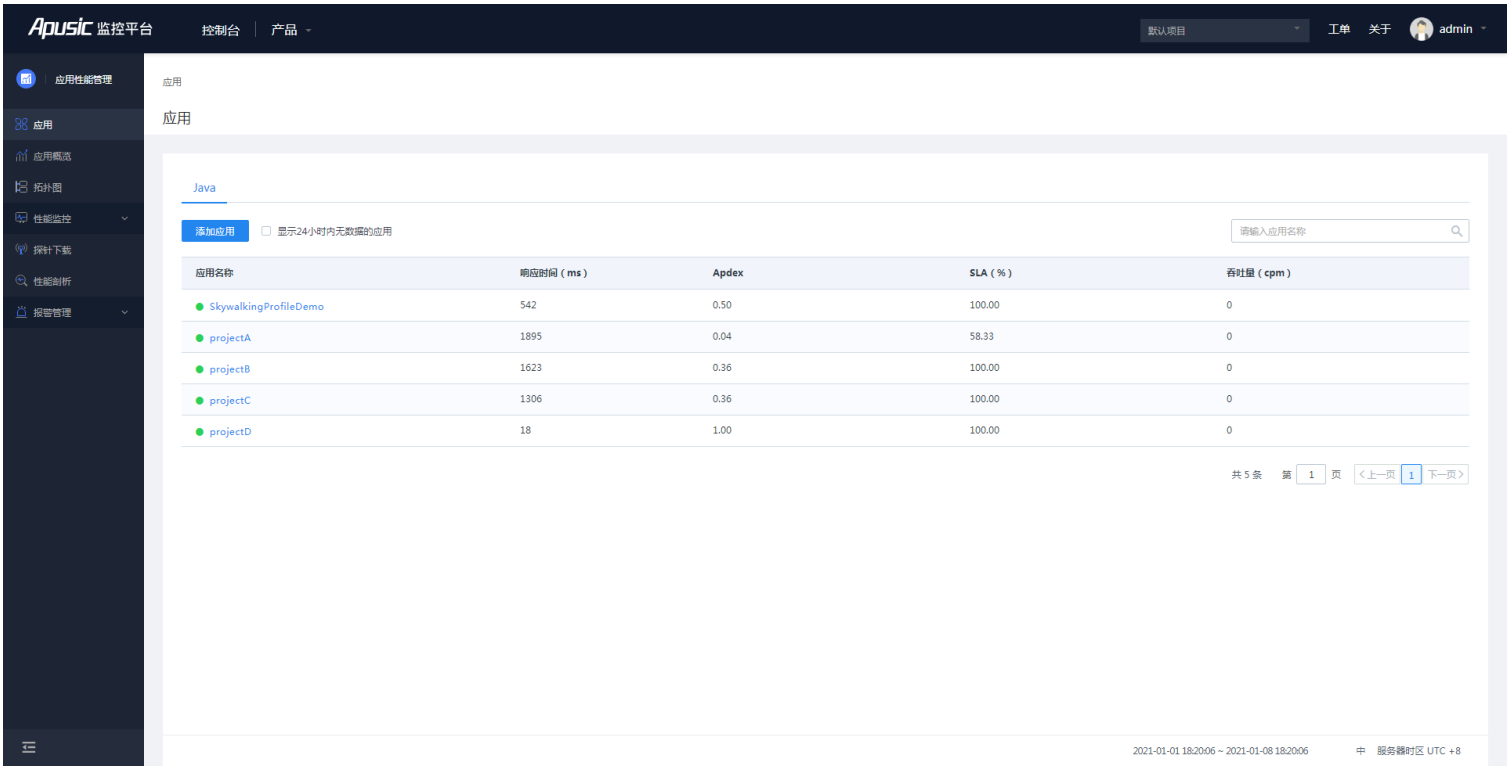

图 2-1快速安装

6.1 服务

选择查看查询的服务和时间区间（选择时间区间不能大于60天），即可查询对应服务的指标。列表服务（如图所示）主要指标数据包括：

响应时间：ResponseTime，即在选定时间内，服务所有请求的平均响应时间(ms)。

性能指数： Apdex(Application Performance Index)是一个国际通用标准，Apdex 是用户对应用性能满意度的量化值。它提供了一个统一的测量和报告用户体验的方法，把最终用户的体验和应用性能作为一个完整的指标进行统一度量，其中最高为1最低为0。

服务级别: SLA（service level agreement），即服务等级协议，指每分钟内响应成功请求的占比。

吞吐量: Throughput，即在选定时间内，每分钟服务响应的请求量(cpm)。

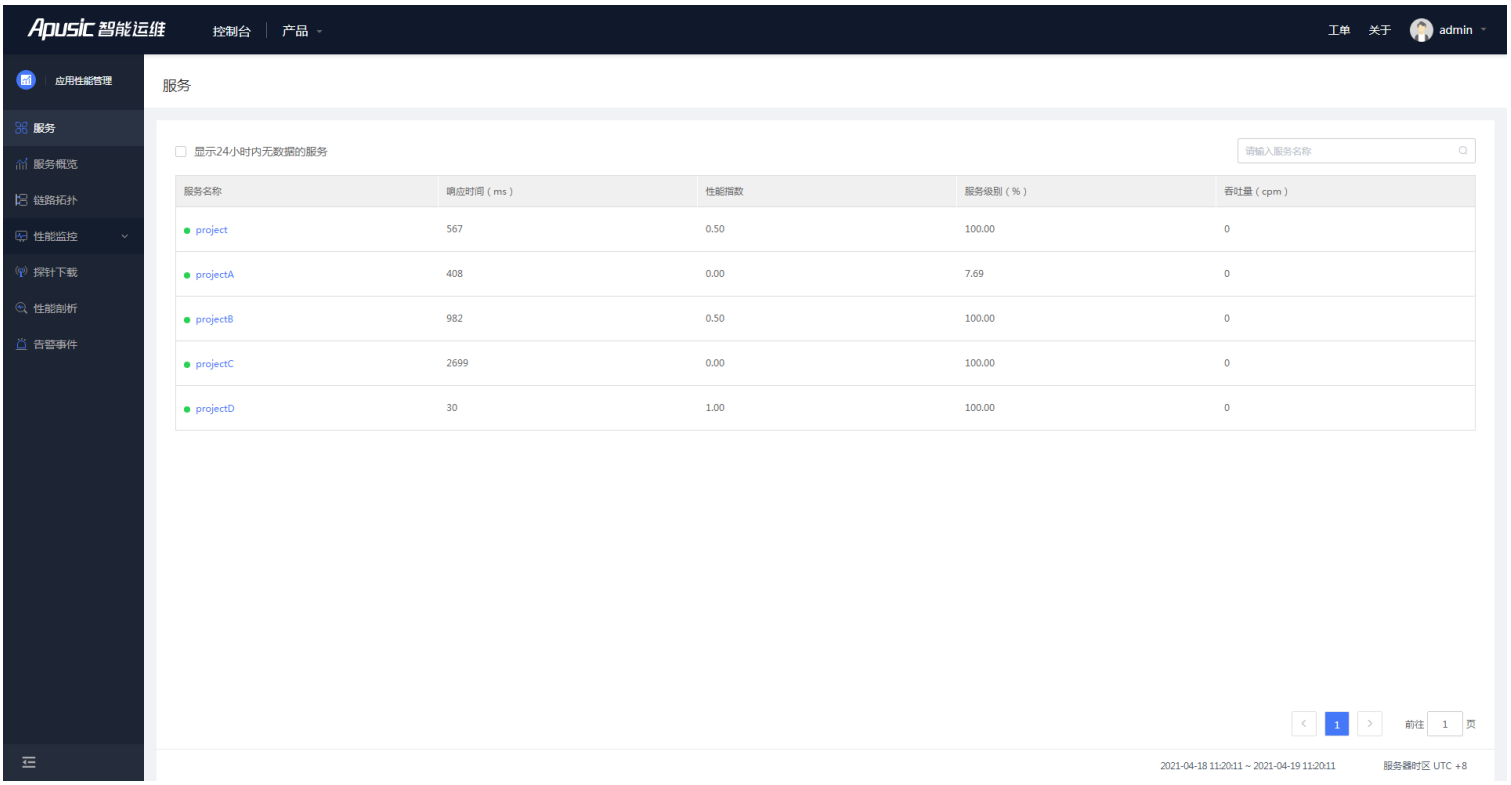

图 3-1服务

6.2 服务概览

点击左侧导航栏【服务概览】，进入服务概览如图所示。服务概览选择服务和对应实例通过数字或图表展示性能指标，响应时间、响应时间分布、吞吐量、SLA、慢SQL等详细信息。这里要理解三个概念：

- 服务：表示对请求提供相同行为的一系列或一组工作负载。
- 实例：服务实例（Service Instance）：上述的一组工作负载中的每一个工作负载称为一个实例。
- 端点：Endpoint，对于特定服务所接收的请求路径，如 HTTP 的 URI 路径和 gRPC 服务的类名 + 方法签名。

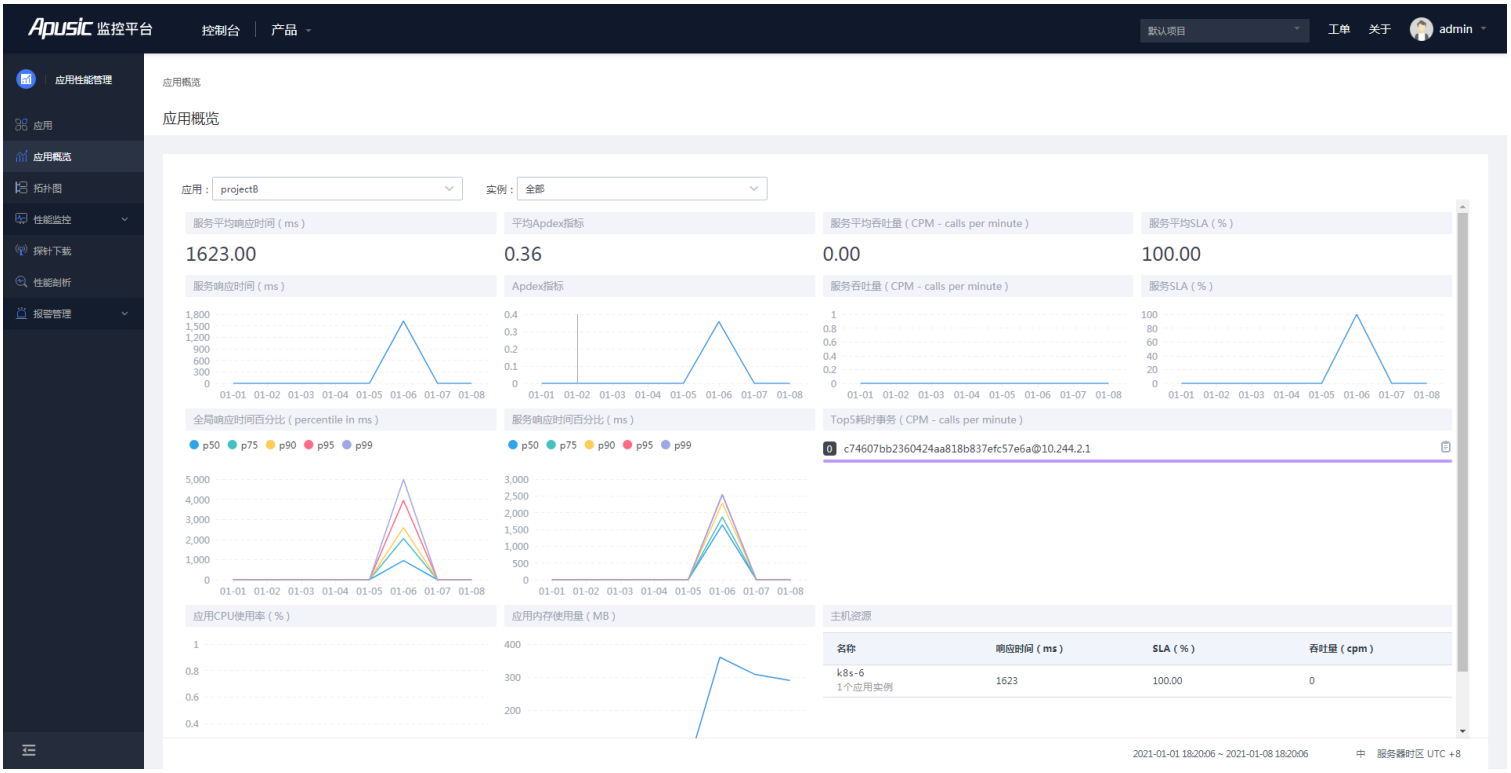

图 4-1服务概览

服务概览展示性能指标有：

- 服务平均吞吐量（数字）：每分钟请求数；
- 服务响应吞吐量（折线图）：不同时间的每分钟请求数；
- 服务性能指数（数字）:当前服务的评分；
- 服务性能指数（折线图）：不同时间的Apdex评分；
- 服务平均吞吐量（数字）：当前服务响应的请求量(cpm)；
- 服务吞吐量（折线图）：不同时间的服务响应的请求量(cpm)；
- 服务SLA（数字）：当前服务响应成功请求的占比；
- 服务SLA（折线图）：不同时间的响应成功请求占比；
- 服务响应时间百分比（折线图）：列举P50，P75，P90，P95，P99统计百分位数口径，在一个样本数据集合中，通过某个样本值，可以得到小于这个样本值的数据占整体的百分之多少，这个样本值的值就是这个百分数对应的百分位数；
- 全局响应时间百分比（折线图）：同样列举P50，P75，P90，P95，P99统计口径，百分位数；
- Top5耗时端点：列举出当前服务响应时间最大的端点Top5；
- 服务CPU使用率： CPU占用的百分比；
- 服务内存使用量：内存占用的大小；

- 主机资源：包括响应时间、SLA、吞吐量。

6.3 链路拓扑

7 链路拓扑

链路拓扑是来展示服务和服务之间的依赖关系。用户可以选择单一服务查询，也可以将多个服务设定为一组同时查询。SW能够根据获取的数据自动绘制服务之间的调用关系图，并能识别常见的服务显示在图标上，如 DB和中间件等（如图所示）。点击依赖线上的圆点，显示服务之间的依赖情况，如每分钟吞吐量、平均响应时间、平均SLA、和侦察端模式（client/Server）。当点中某个服务时，会自动显示当前的服务指标信息， 表示服务告警信息、 表示服务端点追踪信息、 表示服务实例性能信息、 表示api信息面板。

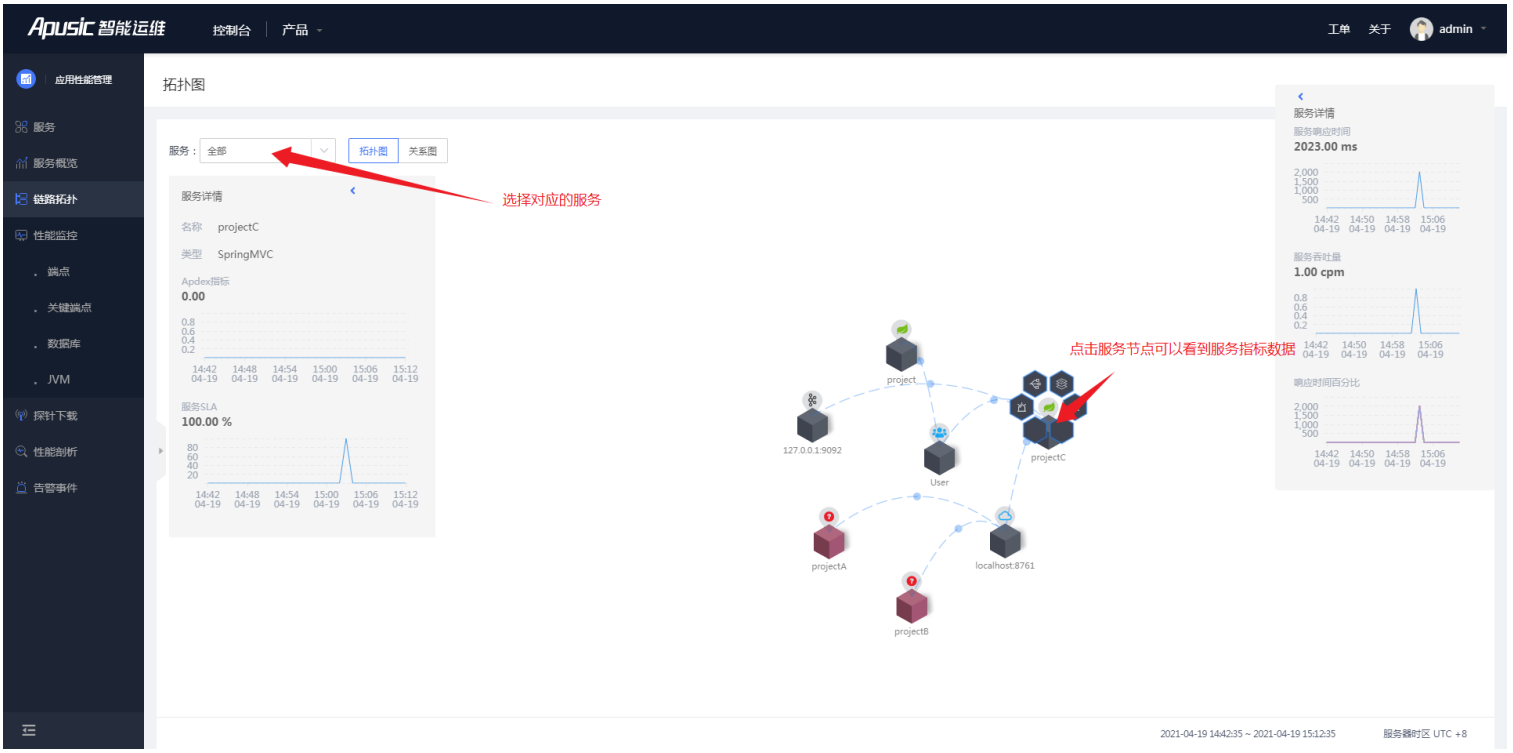

图 5-1链路拓扑

8 告警信息

在链路拓扑界面，点击服务告警信息icon图标，展开服务告警信息（如图所示）,告警是从三个维度来监控，分别为服务、实例,端点触发告警。

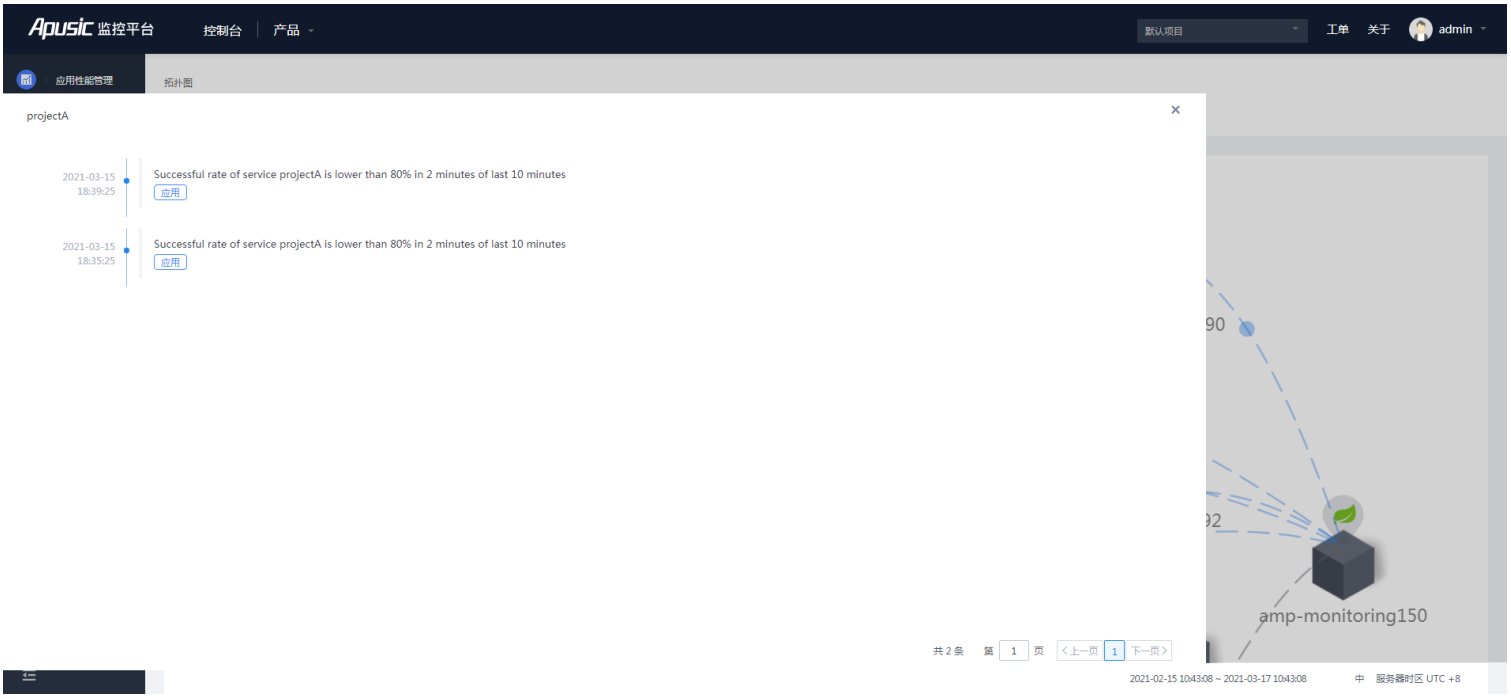

图 5-2告警信息

9 链路追踪

在链路拓扑界面，点击服务端点追踪信息icon图标，展开端点追踪信息（如图所示），界面以接口列表的方式展现，追踪接口内部调用过程。当用户发现服务的SLA降低，或者某个具体的端口响应时间上扬明显，可以使用追踪功能查询具体的请求记录。

用户可以指定搜索条件，如隶属于哪个服务、哪个实例、哪个端点，或者请求是成功还是失败；点击更多按钮，选择追踪ID和时间间区可用精确查询。整个调用链上每一个跨度的耗时和执行结果都会被列出（默认是列表，也可选择树形结构和表格的形式）；如图所示，左侧：api接口列表，红色-异常请求，蓝色-正常请求，右侧表示api追踪列表，api请求连接各端点的先后顺序和时间。

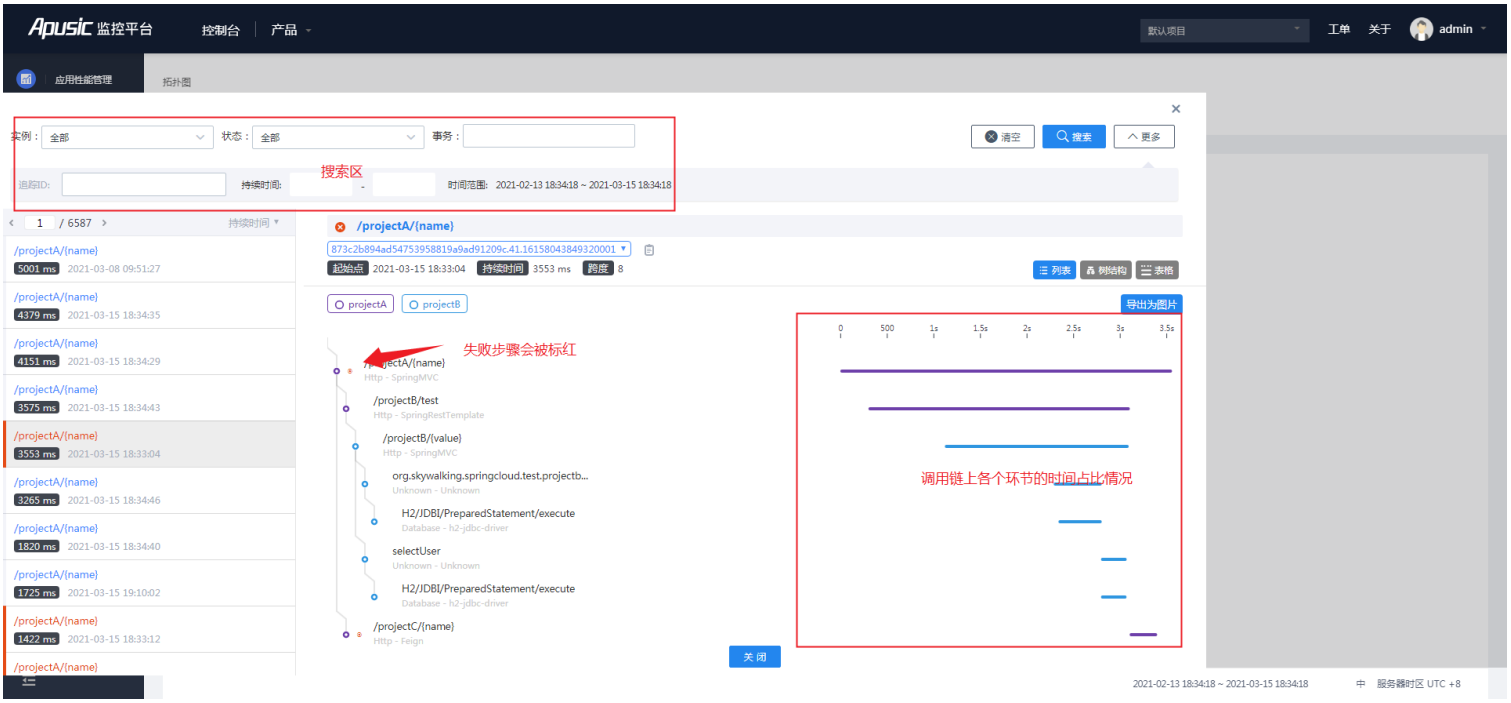

图 5-3请求追踪

点击跨度（如图所示）,显示跨度详情，如果有异常发生，异常的种类、信息和堆栈都会被自动捕获。

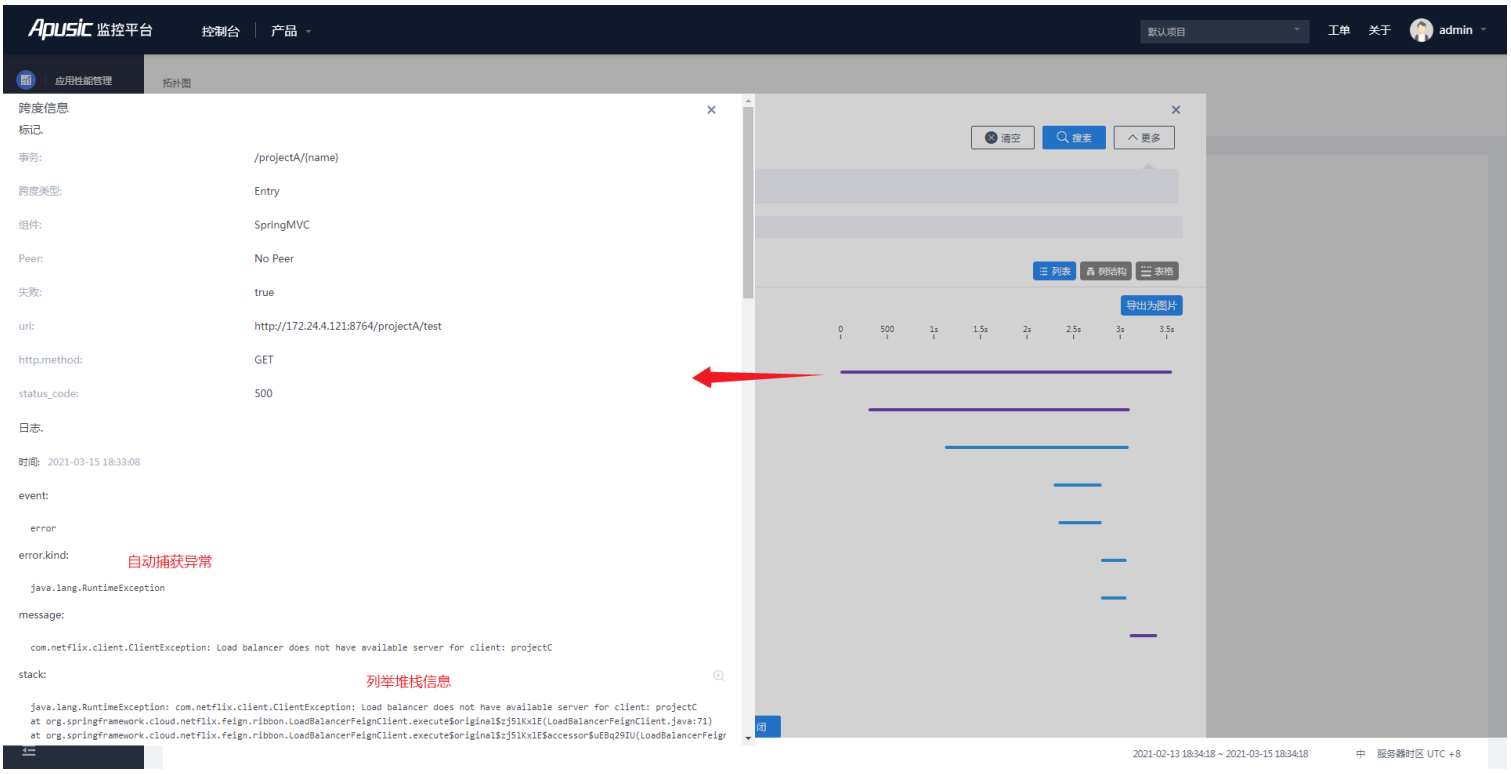

图 5-4跨度信息

如果跨度为数据库操作（如图所示）,，执行的SQL也会被自动记录。

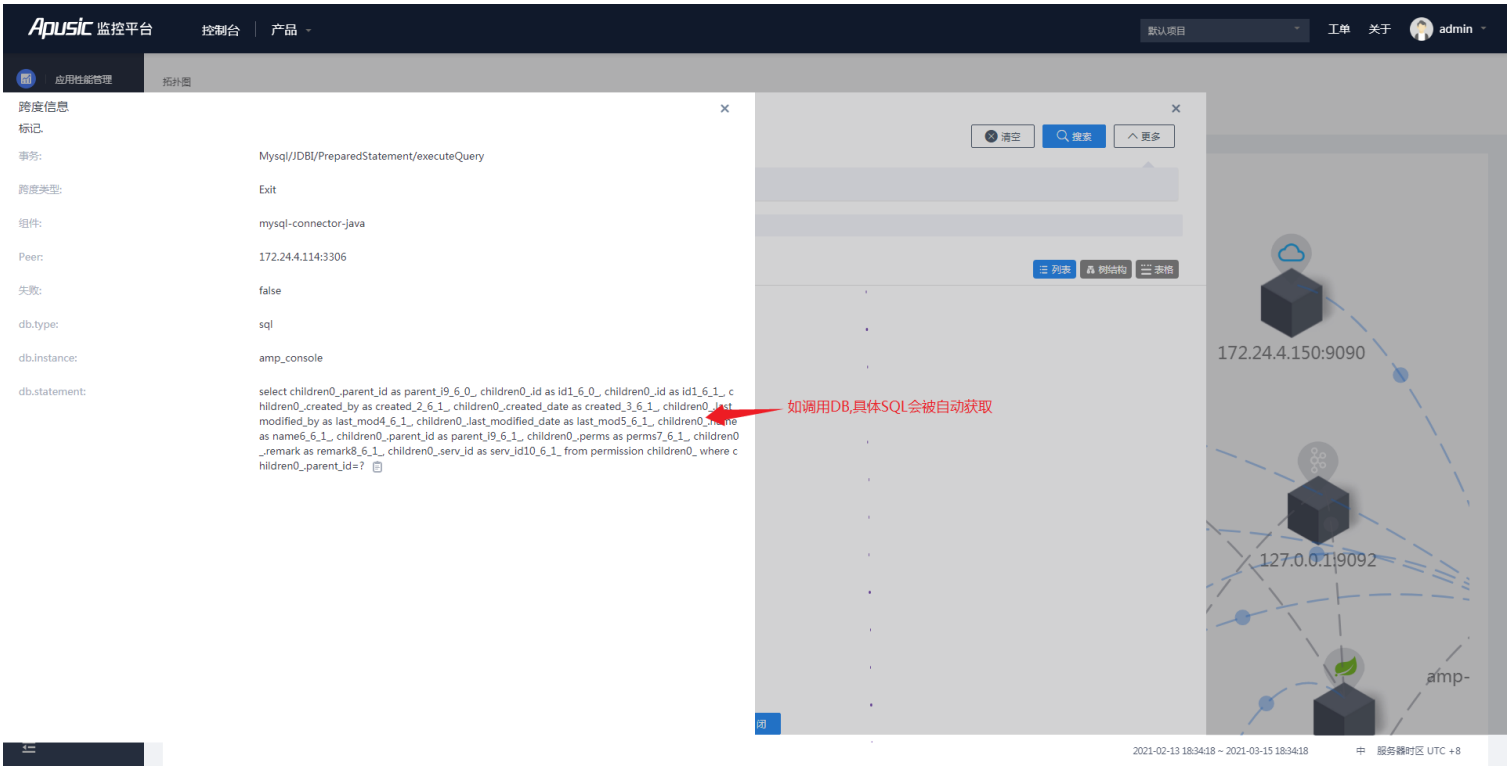

图 5-5数据库跨度信息

10 实例性能信息

在链路拓扑界面，点击服务实例性能信息icon图标，展开服务实例性能信息（如图所示），包括响应时间、Apdex指数、吞吐量、SLA指标等详细信息。

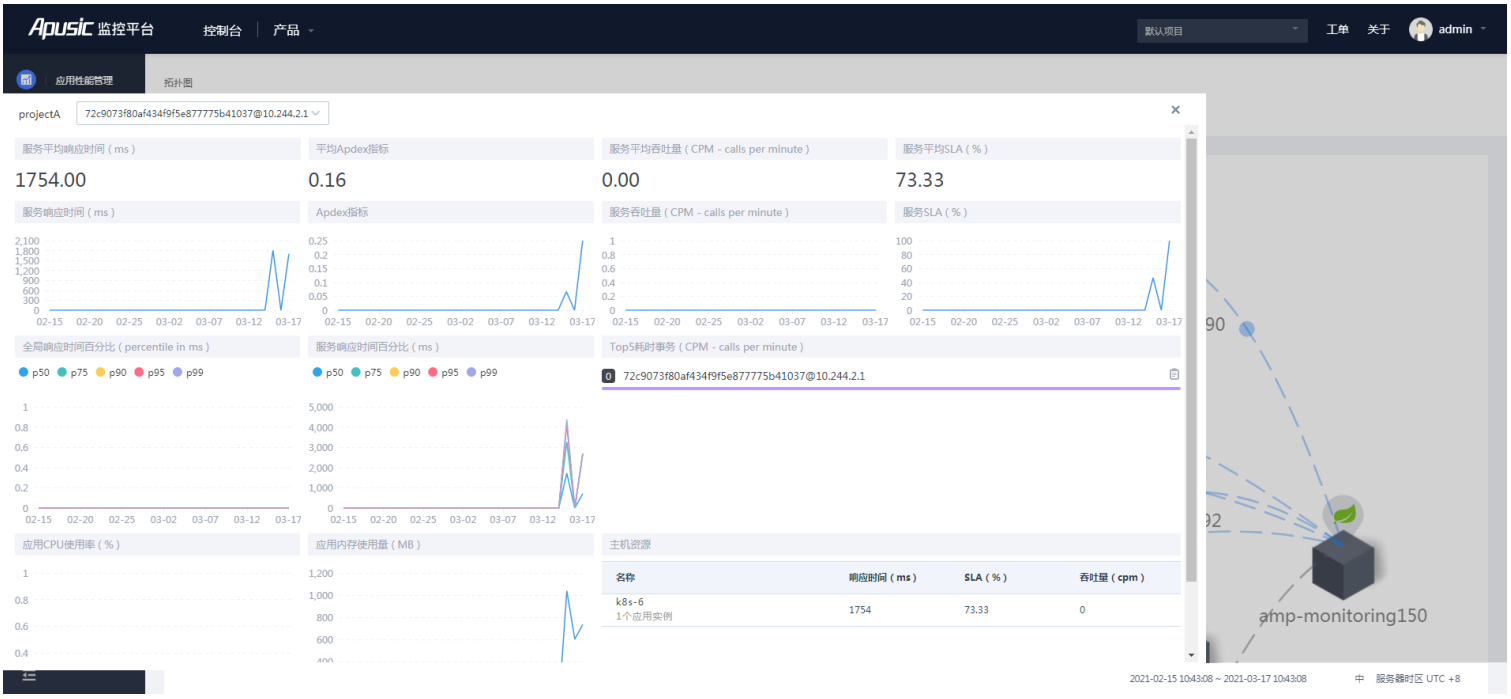

图 5-6实例性能信息

11 API信息面板

在链路拓扑界面，点击api信息面板图标，展开端点信息数据展示，内容包括端点性能和性能对比，具体内容参考第7.1.3章节的端点的介绍，这里不再赘述。

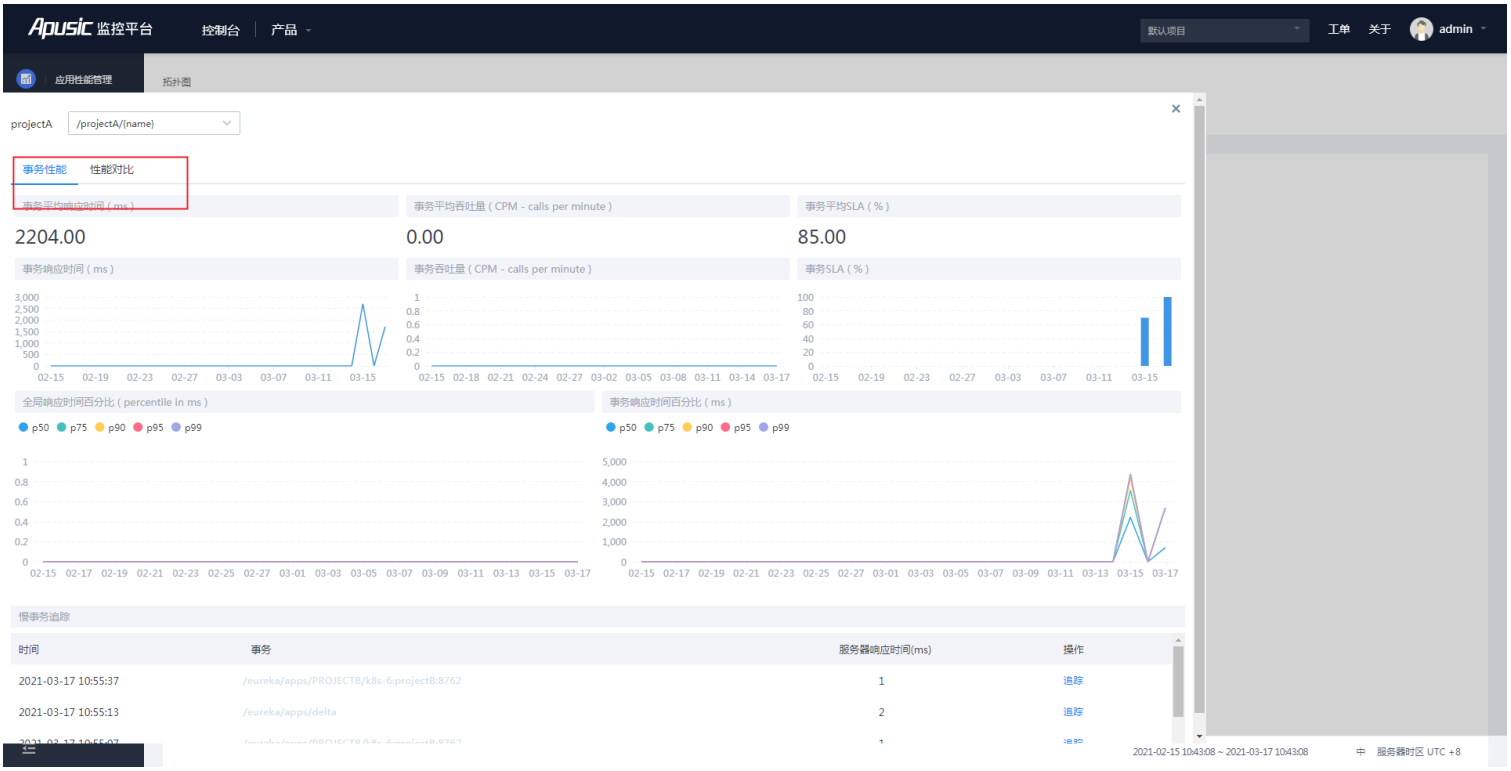

图 5-7 API信息面板

11.1 性能监控

12 端点

12.0.1 全局

在点击ApusicAPM的左侧导航栏 【端点】，进入查看端点的性能指标信息（如图所示）。和服务指标类似，端点指标也包括响应时间、吞吐量、SLA等指标。根据响应时间、吞吐量、SLA性能指标排序，展示端点全局的性能信息。

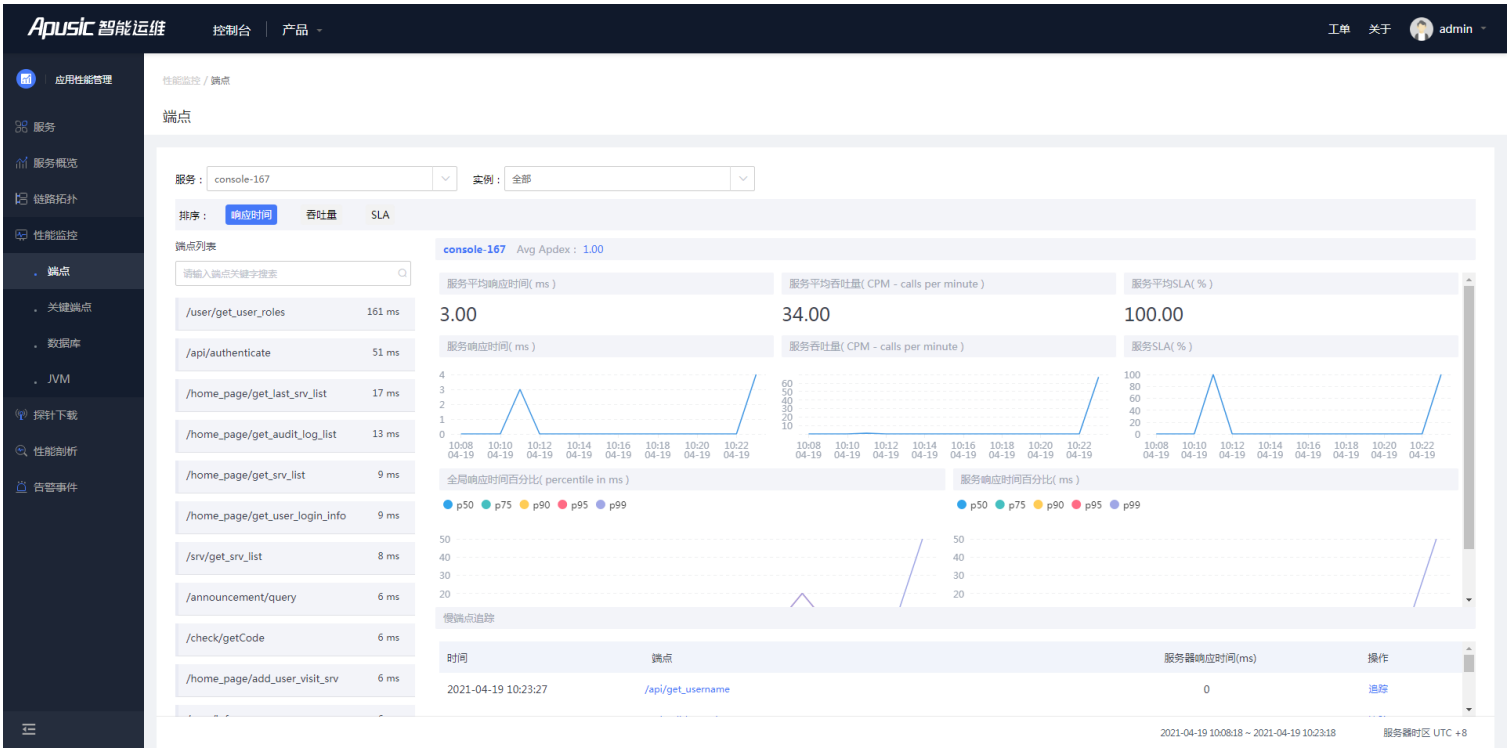

图 6-1端点

12.0.2 追踪

在ApusicAPM的端点全局指标信息界面，点击端点或追踪，跳转到端点追踪信息界面。利用Trace功能进行链路追踪，可以跟着请求穿透整个系统。如图所示可以了解每个跨度(Spans)的耗时情况，哪些耗时长。默认是列表，也可选择树形结构和表格的形式；而树形结构的图可以看清层次关系。而点到具体的跨度上，可以看到明细信息，如点到DB上可以看到具体执行的SQL。

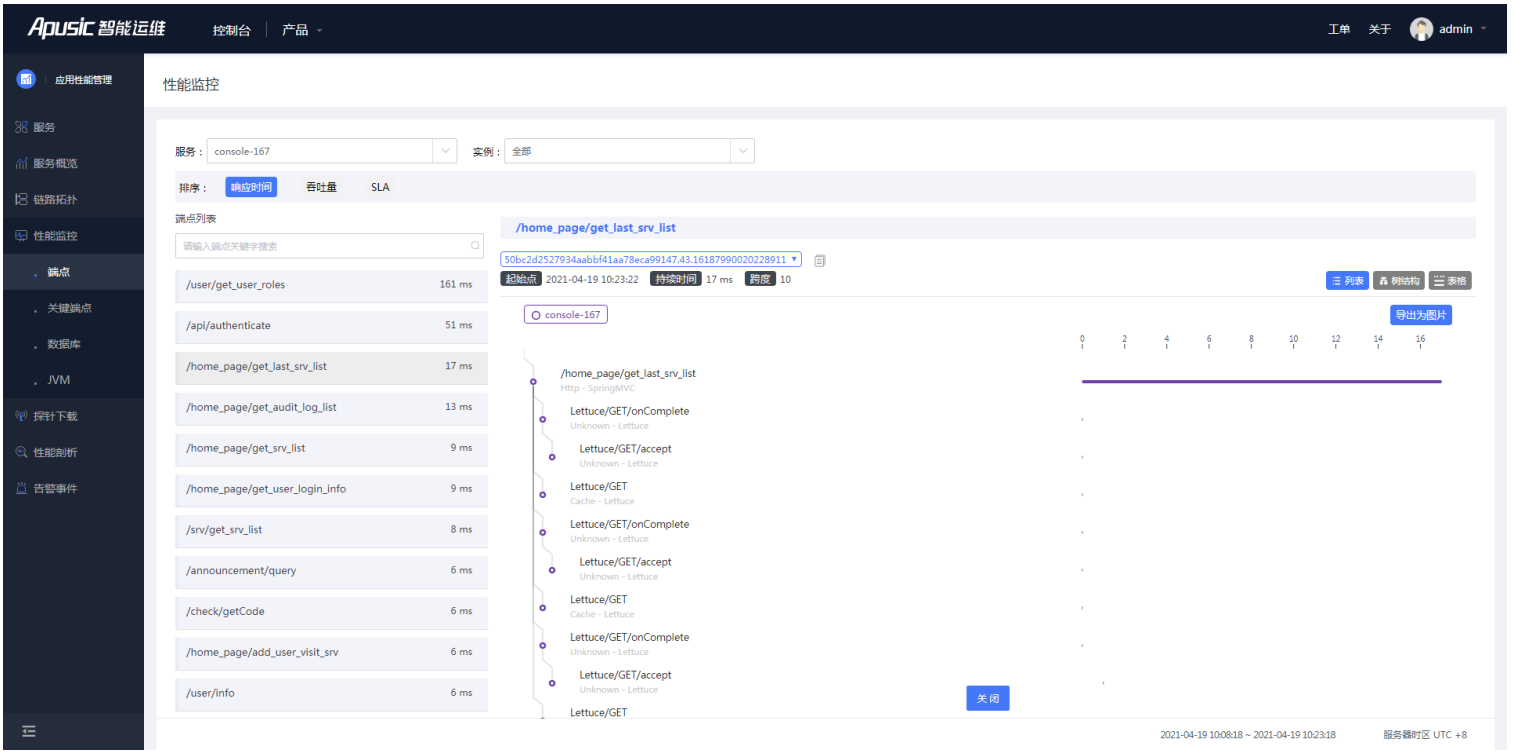

图 6-2追踪

12.0.3 端点

在ApusicAPM的端点全局指标信息界面，点击端点列表的某个端点，右侧展示该端点性能指标信息和性能指标对比信息。端点性能指标也包括响应时间、吞吐量、SLA等指标。而性能指标对比指对比昨天、今天、一周的响应时间、吞吐量、SLA的性能指标（如图所示）。

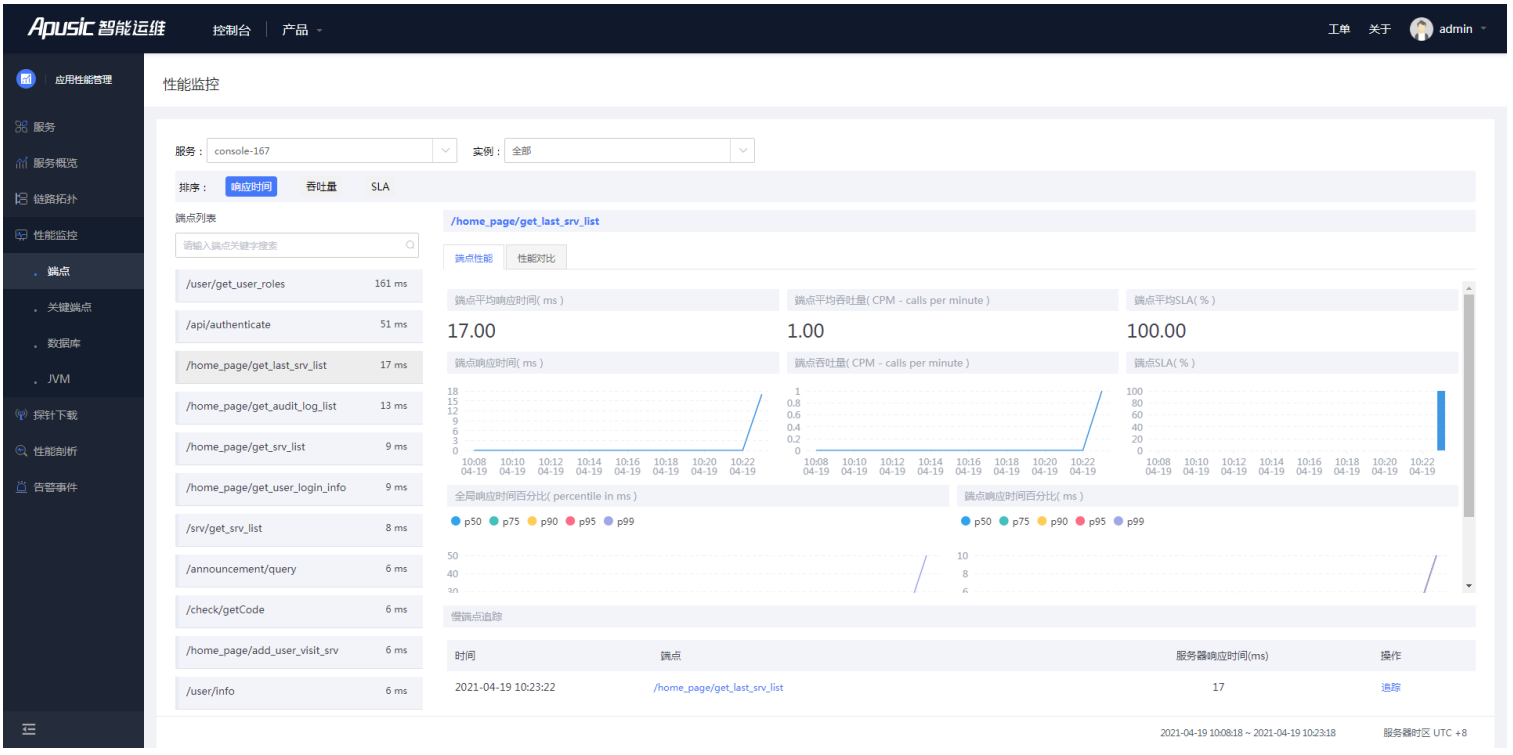

图 6-3性能对比

13 关键端点

在ApusicAPM点击左侧导航栏【关键端点】，进入查看关键端点界面（如图所示）。列表展示端点名称，端点所属服务、响应时间、吞吐量、SLA指标。点击列表的修改可修改关键端点的端点别名和ApdexT值。点击列表的删除按钮并确认可成功删除添加的关键端点。

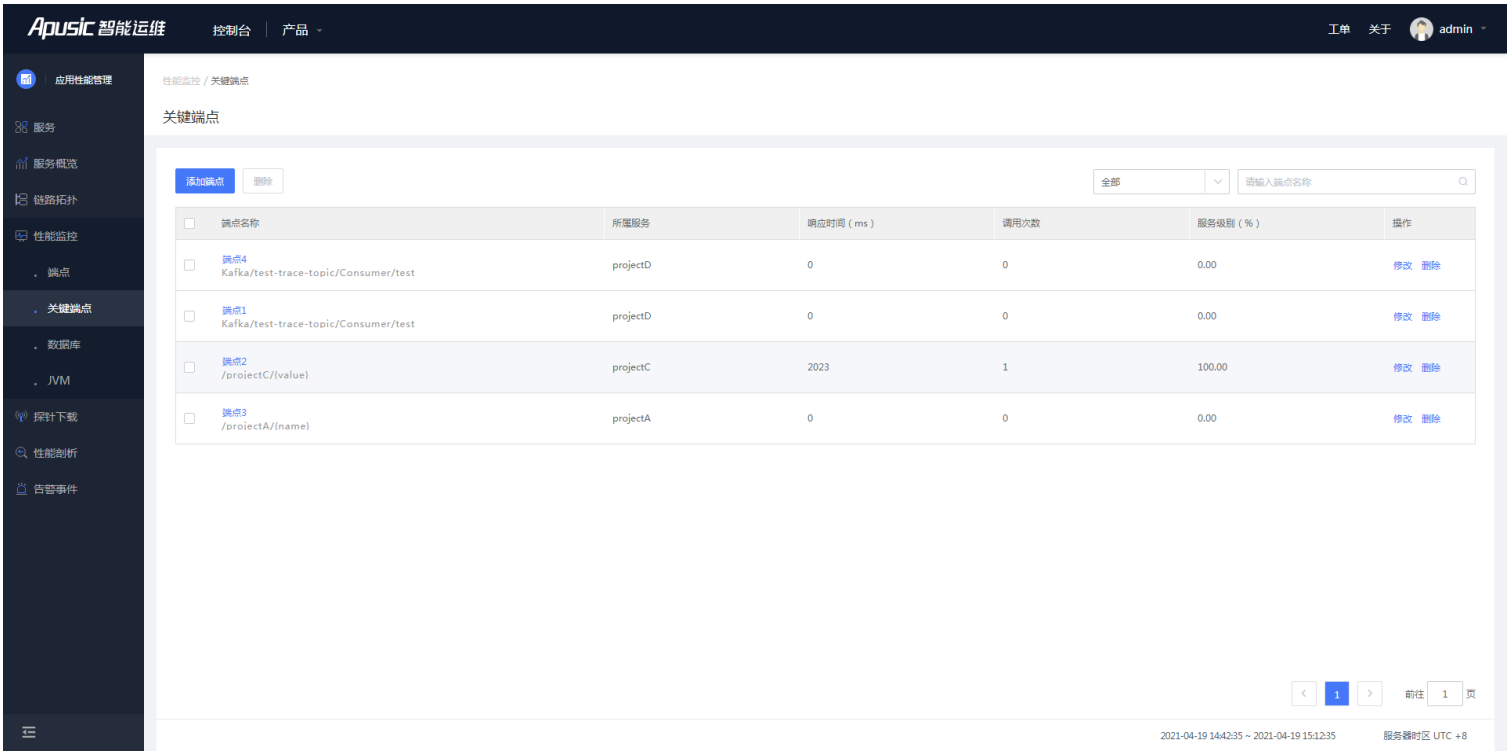

图 6-4关键端点列表

在ApusicAPM的关键端点界面点击添加端点按钮，选择监控服务和关键端点，输入端点别名，配置ApdexT值，点击完成按钮，即可成功添加关键端点任务（如图所示）。

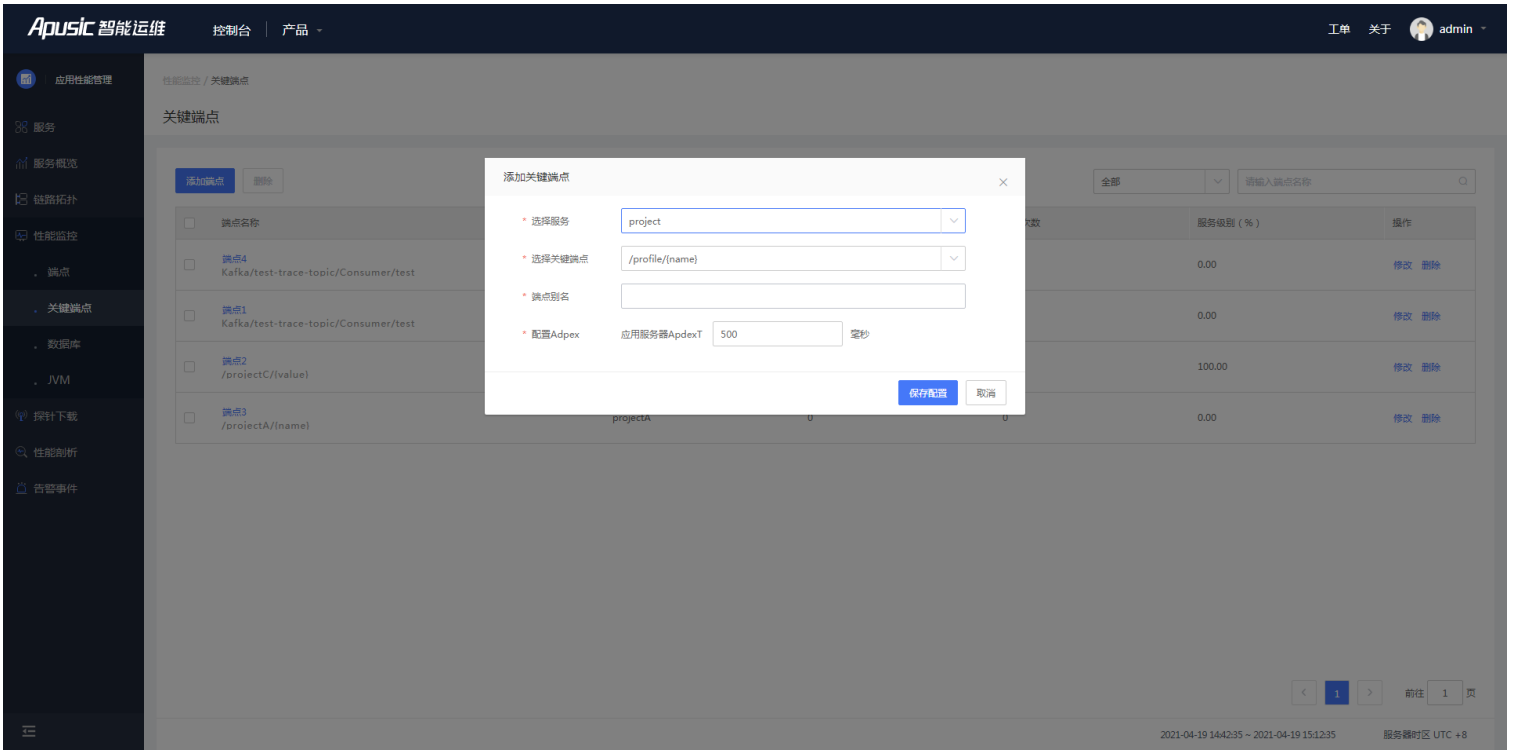

图 6-5添加关键端点

在应用性能管理的关键端点界面，点击列表端点名称，可查看关键端点详情，关键端点详情显示响应时间、吞吐量、SLA性能指标、慢端点追踪、依赖图。

依赖图：Dependency Map，代表哪些服务在依赖（调用）该端点，如果是前端直接调用，会显示为用户（User）依赖中。

慢端点追踪：SW会自动列出当前时间段内端点最慢的调用记录，点击追踪可以在追踪功能找到具体的调用链信息，便于定位。

图 8-6依赖图

14 数据库

除了服务本身的指标，APM也监控了服务依赖的DB指标。在APM的数据库界面，就可以看到从服务角度（client）来查看当前数据库事件平均响应时间（单位ms）、当前数据库访问成功率（单位%）、当前数据库每分钟请求数、数据库不同比例的响应时间（单位ms）、前N个慢查询（单位ms）、所有数据库中CPM排名、所有数据库不健康排名（请求成功率排名）。

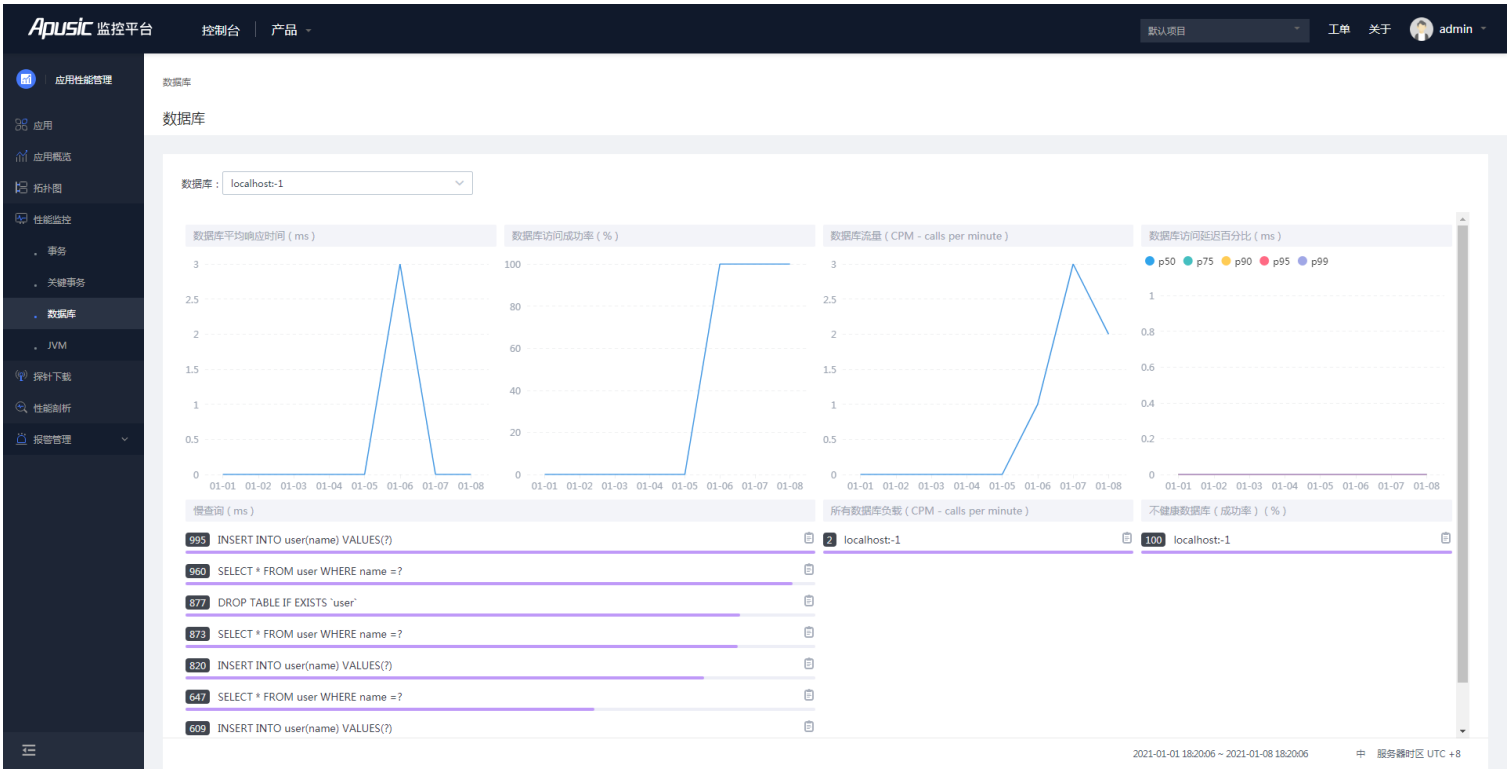

图 8-7数据库

15 JVM

监控到Service运行时的CPU、堆内存、非堆内存使用率、以及GC情况。这些信息来源于JVM。在ApusicAPM的JVM界面，选择服务的实例，即可查看服务某个实例的指标数据。除了常规的吞吐量、SLA、响应时间等指标外，实例信息中还会给出JVM的信息，如堆栈使用量，GC耗时和次数等。

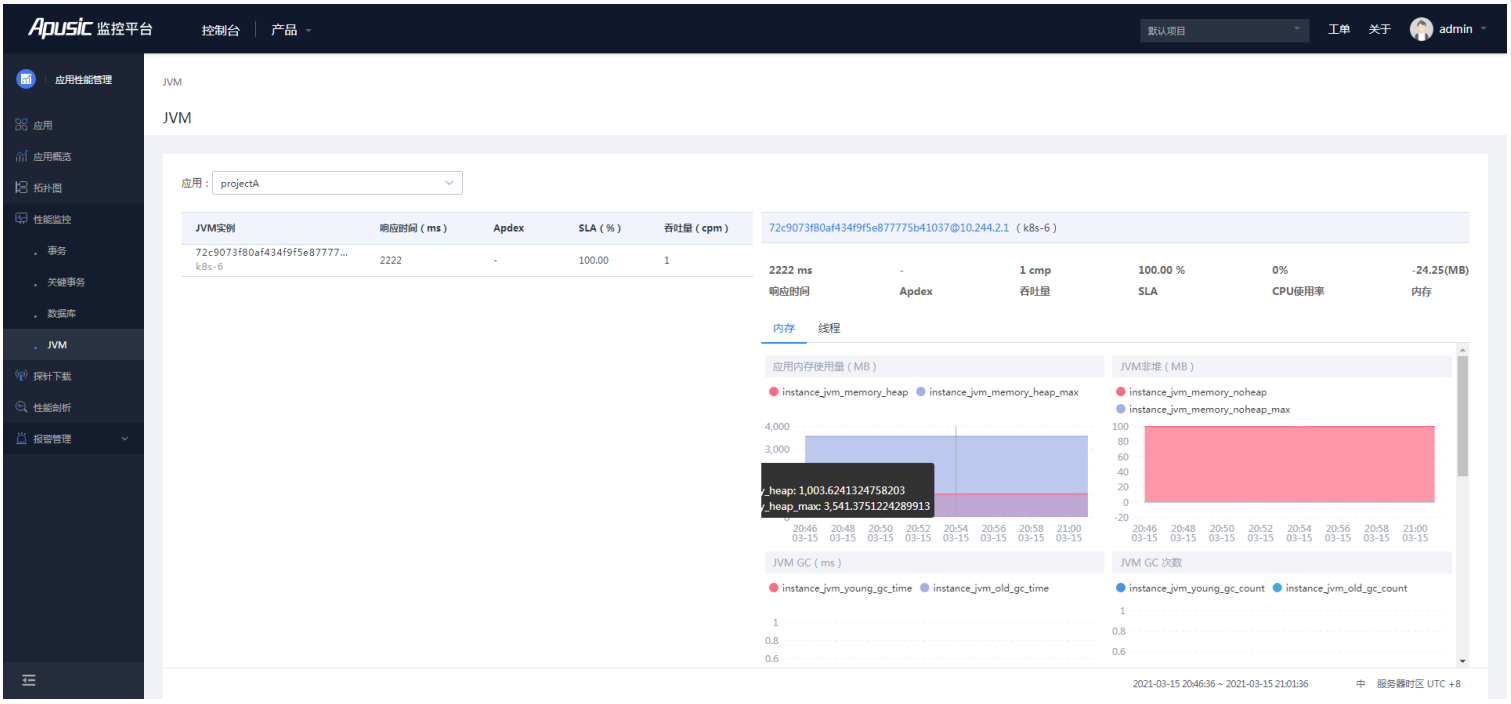

图 8-8端点

15.1 性能剖析

- 新建任务

单独端点进行采样分析，并可查看堆栈信息，在ApusicAPM的性能剖析界面，点击新建按钮按钮，新建需要分析的端点（如图所示），选择服务和端点，设置监控时间和采样数，点击完成按钮。

- 选择服务：需要分析的服务
- 选择端点：链路监控中端点的名称，可以再链路追踪中查看端点名称
- 监控时间：采集数据的开始时间
- 监控持续时间：监控采集多长时间
- 起始监控时间：多少秒后进行采集
- 采样时间间隔：多少毫秒采集一次

- 最大采集数：最大采集多少样本

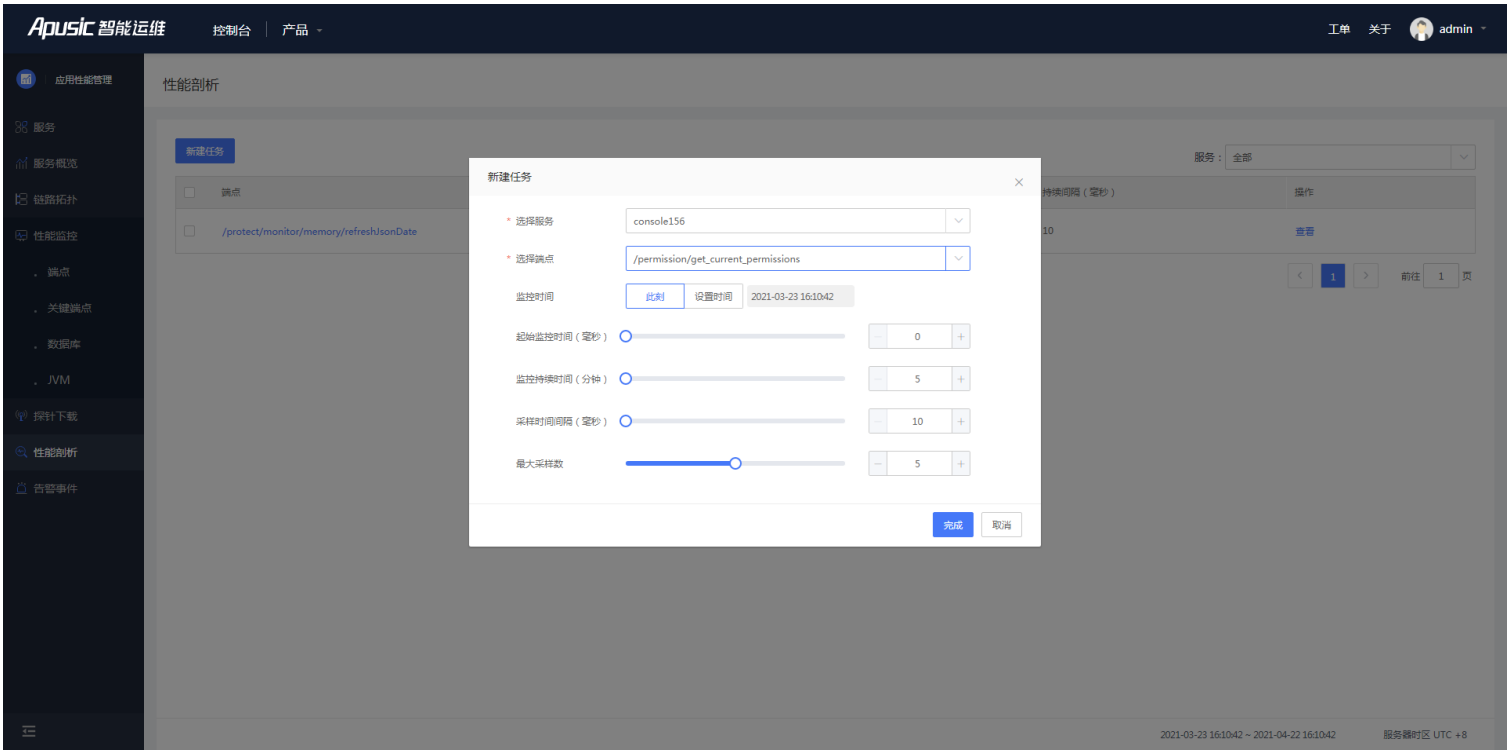

图 7-1新建性能任务

- 性能剖析

> *新建任务后，APM将开始采集服务的实时堆栈信息。采样结束后，用户点击分析即可查看具体的堆栈信息（如图所示）。左侧列表表示任务及对应的采样请求，右侧：端点链路及每个端点的堆栈信息。*

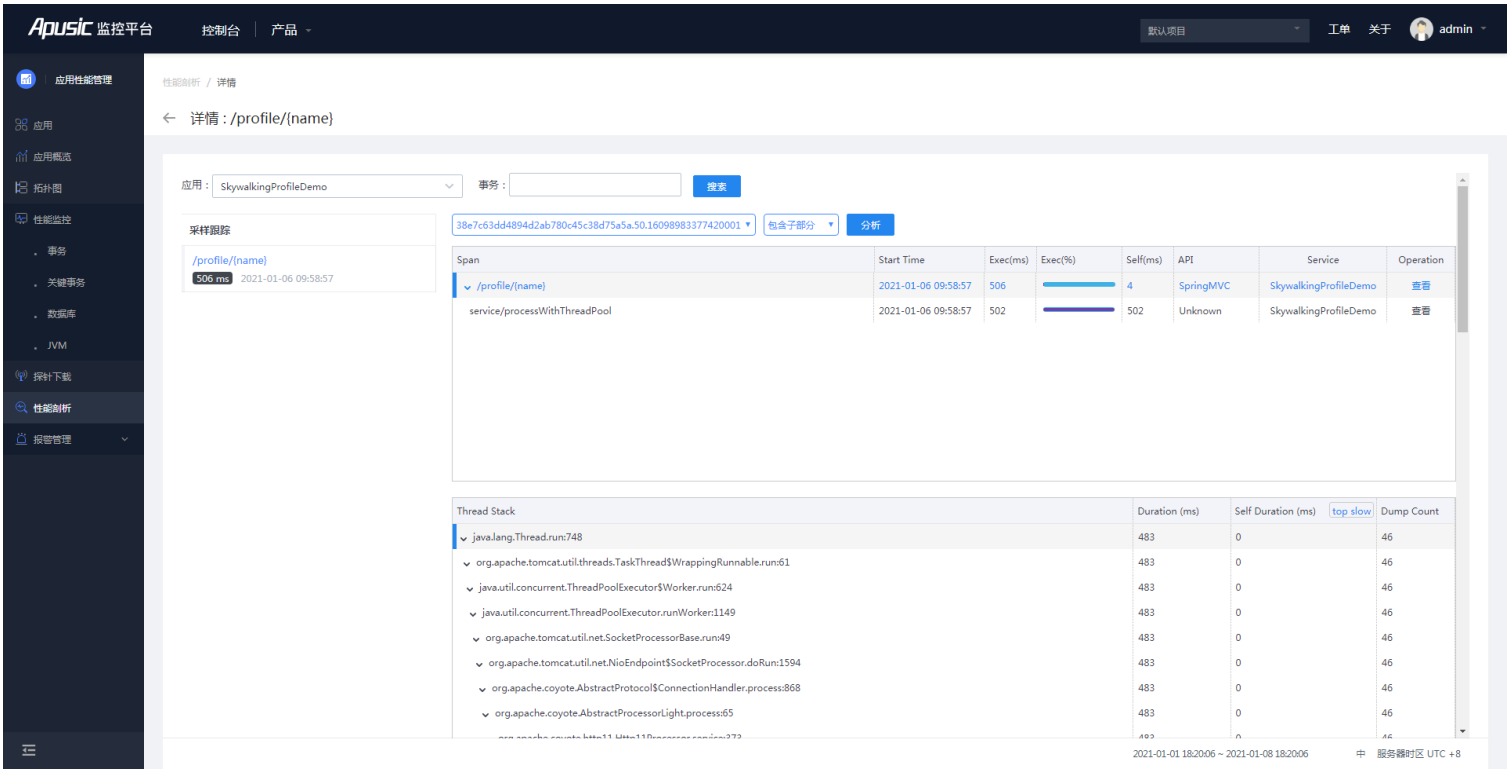

图 7-2性能剖析分析

15.2 探针下载与安装

探针下载界面，提供java、php、.net探针下载并介绍各探针的功能和修复缺陷。探针(agent)负责进行数据的收集，包含了Tracing和Metrics的数据，agent会被安装到服务所在的服务器上，以方便数据的获取。基于不同的来源探针是不一样的，但作用都是收集数据，将数据格式化为适用的格式。

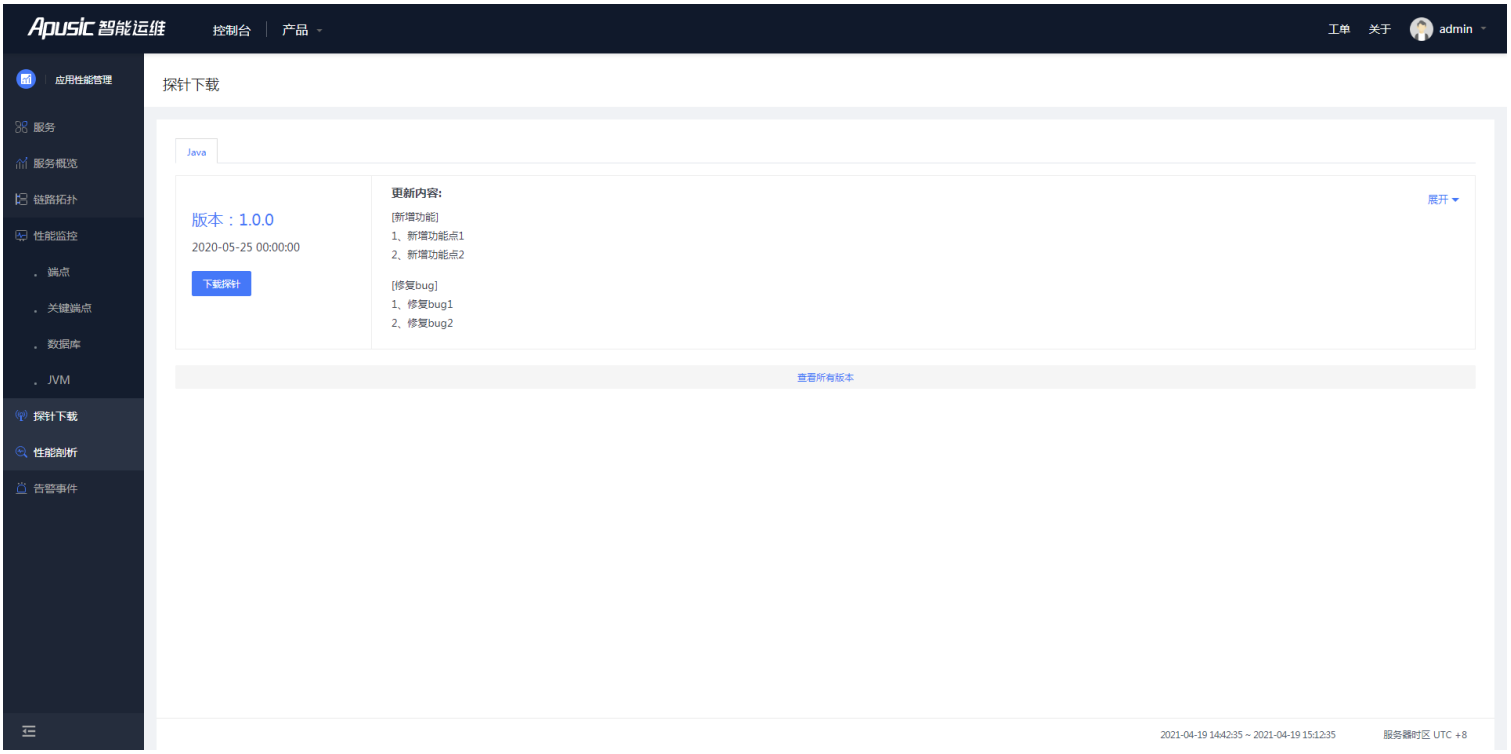

图 9-1探针下载与安装

本章介绍目前ApusicAPM支持的Java、.NET Core、PHP 等不同语言探针的安装部署及配置方式。

16 Java探针安装

16.0.1 Java探针通用安装配置

apusic-skywalking-apm-agent目录	说明
bin	启动脚本
config	为客户端代理配置文件
logs	agent相关运行情况日志
activations	插件包
bootstrap-plugins	插件包
optional-reporter-plugins	插件包
skywalking-agent.jar	agent代理jar包

表： Java探针目录结构

- Java Agent的配置agent.config 文件属性：

> *agent.service_name：一个服务（项目）的唯一标识，这个字段决定了在sw的UI上的关于service的展示名称*
> *collector.backend_service：探针需要同collector进行数据传输的IP和端口*

- Java Agent配置方式：

1、系统属性(-D)

使用 -Dskywalking. + agent.config配置文件中的key 即可。例如：

agent.config 文件中有一个属性名为 agent.service_name ， 那么如果使用系统属性的方式，则可以写成java -javaagent:/opt/agent/skywalking-agent.jar -Dskywalking.agent.service_name=你想设置的值 -jar somr-spring-boot.jar

2、代理选项

在JVM参数中的代理路径之后添加属性即可。

格式：

-javaagent:/path/to/skywalking-agent.jar=[option1]=[value1],[option2]=[value2]

例如：

java -javaagent:/opt/agent/skywalking-agent.jar=agent.service_name=服务名称 -jar somr-spring-boot.jar

3、系统环境变量

agent.config 文件中默认的大写值，都可以作为环境变量引用。例如，agent.config 中有如下内容

agent.service_name=${SW_AGENT_NAME:Your_ApplicationName}

这说明Skywalking会读取名为 SW_AGENT_NAME 的环境变量。

几种配置优先级：

代理选项 > 系统属性（-D） > 系统环境变量 > 配置文件

16.0.2 ApusicAS V9.0安装探针

1. 下载Java探针apusic-skywalking-apm-agent.tar.gz,解压探针到目标服务器

```
tar -zxvf apusic-skywalking-apm-agent.tar.gz -C {PATH}
```

apusic-skywalking-apm-agent/config/agent.config位置文件修改如下参数：

```
-Dskywalking.collector.backend_service= localhost:11800
-Dskywalking.agent.service_name=service_name"
```

2. 在ApusicASV9应用的启动脚本/path/domains/mydomain/bin/startapusic启动脚本,在JVM_OPTS添加如下参数：

```
-javaagent:{PATH}/agent/skywalking-agent.jar
```

3. 执行/path/domains/mydomain/bin/startapusic重启ApusicASV9.0应用服务器，登录AAMP控制台，访问应用性能管理服务。

16.0.3 ApusicAS V10.0安装探针

1. 下载Java应用探针apusic-skywalking-apm-agent.tar.gz并解压探针到目标服务器:

```
tar -zxvf apusic-skywalking-apm-agent.tar.gz -C {PATH}
```

2. 访问金蝶 Apusic应用服务器管理控制台，在登录界面输入用户名和密码，进行登录，在配置->服务配置->JVM配置->JVM选项，添加如下JVM选项参数：

```
-javaagent:{PATH}/agent/skywalking-agent.jar
-Dskywalking.collector.backend_service= localhost:11800
-Dskywalking.agent.service_name=service_name"
```

3. 重启Apusic ASV10应用，登录AMP，访问应用性能管理。

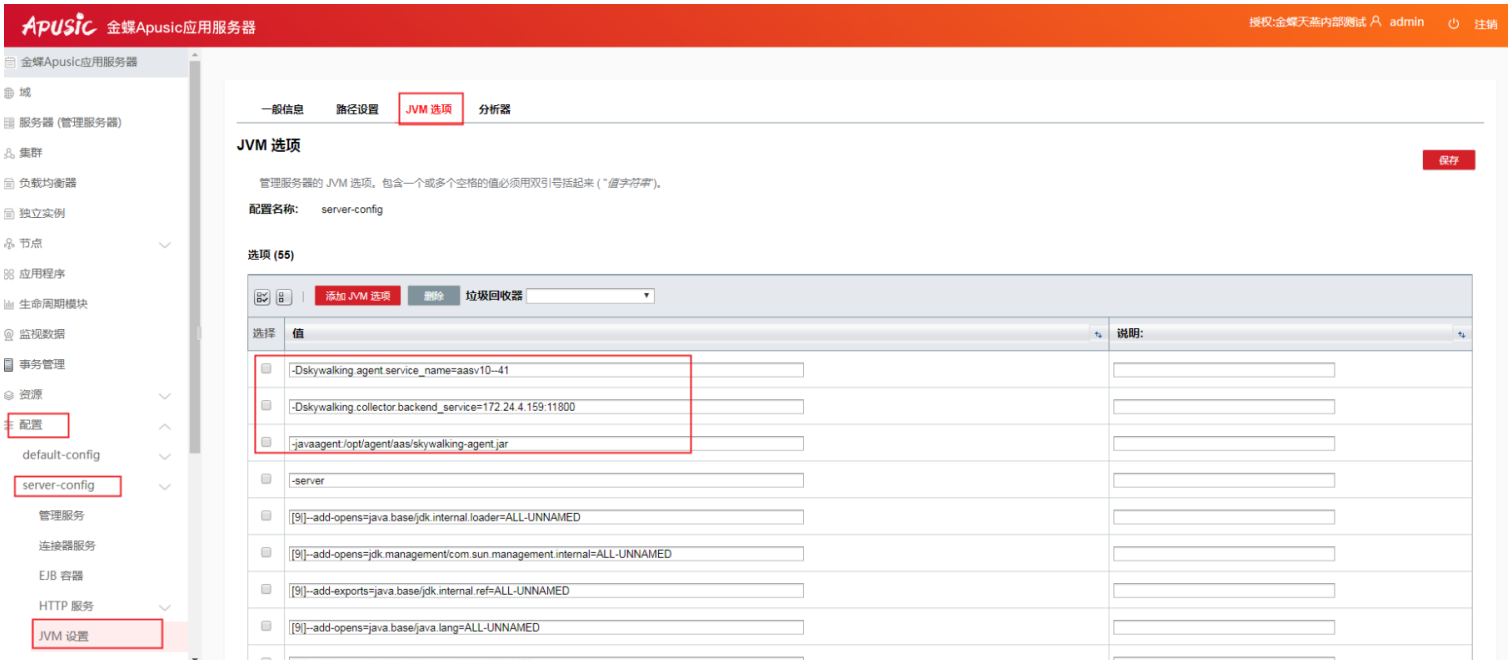

图 9-2 JVM选项配置

16.0.4 Apache Tomcat服务器器安装探针

1. 下载Java探针apusic-skywalking-apm-agent.tar.gz,解压探针到目标服务器tar -zxvf apusic-skywalking-apm-agent.tar.gz -C /path

2. 在 tomcat/bin/catalina.sh在第一行加入，如下设置：

```
CATALINA_OPTS="$CATALINA_OPTS -javaagent:{你的路径}/agent/skywalking-
agent.jar"; -Dskywalking.collector.backend_service=localhost:11800";
-Dskywalking.agent.service_name=service_name
export CATALINA_OPTS
```

3. 重启Tomcat应用服务器，访问应用性能管理服务，即可看到部署在Tomcat中的应用服务实例。

16.0.5 TongWeb应用服务器配置探针

1. 下载Java探针apusic-skywalking-apm-agent.tar.gz,解压探针到目标服务器tar -zxvf apusic-skywalking-apm-agent.tar.gz -C /path

2. 访问TongWeb管理控制台，在登录界面输入用户名和密码，进行登录，在启动参数配置界面，在其他jvm参数添加如下配置:

```
-Dskywalking.agent.service_name=Your_ApplicationName
-Dskywalking.collector.backend_service=localhost: 11800
```

```
-javaagent:/path/skywalking-agent.jar
```

3. 重启TongWeb服务，访问应用性能管理，即可看到部署在TongWeb服务器中的应用实例。

16.0.6 InforSuite AS应用服务器安装探针

1. 下载Java探针apusic-skywalking-apm-agent.tar.gz,解压探针到目标服务器

```
tar -zxvf apusic-skywalking-apm-agent.tar.gz -C /path
```

2. 访问InforSuite AS管理控制台，在登录界面输入用户名和密码，进行登录，在配置-JVM设置界面，在jvm选项添加如下配置：

```
-Dskywalking.agent.service_name=Your_ApplicationName
-Dskywalking.collector.backend_service=localhost: 11800
-javaagent:/path/skywalking-agent.jar
```

3. 进入\path\InforSuite\AppServer\as\config目录，修改osgi.properties文件，在org.netbeans.lib.profiler, org.netbeans.lib.profiler.*末尾添加

```
org.apache.skywalking.apm.agent.core,org.apache.skywalking.apm.agent.core.*
```

4. 重启InforSuite AS服务，访问应用性能管理服务即可。

17 PHP 探针安装

针对PHP应用的性能监控及链路追踪，ApusicAPM提供PHP语言探针用于采集链路信息并上报到服务端。PHP探针基于源码包方式，用户需要在目标应用节点进行编译安装及构建，当前仅支持Linux环境部署和安装。

注意：当前PHP探针仅支持PHP7.1及以上版本。

支持的PHP框架类库及中间件列表如下：

- CURL
- PDO
- Mysqli
- Yar Client
- GRPC Client
- Predis Client
- Redis Extension (Redis Extension)
- Memcache Extension
- RabbitMQ
- Swoole
- Hyperf
- Tars-php
- LaravelS

> *注意：以下使用x86_64环境下的centos7.6 版本服务器操作系统为示例环境进行说明*

17.0.1 编译环境准备

> *需要安装gRPC和Protobuf，编译器使用gcc4.8.5+版本、CMake V3.1及以上版本。*

- 检查gcc 编译器版本

```
# gcc -v
 gcc version 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)
```

- 安装CMake编译工具

> *探针编译安装依赖CMake，可以根据平台架构需要选择x86_64、linux-aarch64、macos等平台的发行包。*

```
#wget https://github.com/Kitware/CMake/releases/download/v3.20.1/cmake-3.20.1-linux-x86_64.tar.gz
#tar -zxvf cmake-3.20.1-linux-x86_64.tar.gz
#cp -R cmake-3.20.1-linux-x86_64 /usr/local
#ln -s /usr/local/cmake-3.20.1-linux-x86_64/bin/* /usr/bin
```

> *验证CMake编译工具安装结果*

```
#cmake --version
cmake version 3.20.1
```

- 安装编译依赖的包

```
#yum install -y gcc-c++ autoconf libtool
#yum groupinstall -y "Development Tools"
```

> *到此，编译安装环境准备完毕，可以开始安装PHP探针依赖的gRPC框架及PHP探针。*

17.0.2 安装gRPC

gRPC 是一个高性能、开源和通用的 RPC 框架，面向移动和 HTTP/2 设计。目前提供 C、Java 和 Go 语言版本，分别是：grpc, grpc-java, grpc-go. 其中 C 版本支持 C, C++, Node.js, Python, Ruby, Objective-C, PHP 和 C# 支持,PHP语言探针使用gRPC 的C版本。

> *从产品包分发目录中拷贝gRPC完整的源码包文件grpc-v1.31.x.tar.gz 。*

- 编译安装protobuf

```
# tar -zxvf  grpc-v1.31.x.tar.gz
# cd grpc-v1.31.x/third_party/protobuf
# ./autogen.sh
# ./configure
# make -j$(nproc)
# make install
$ ldconfig
$ make clean
```

> *验证安装结果*

```
# protoc --version
libprotoc 3.5.1
```

- 编译安装gRPC

```
# cd ..
# make -j$(nproc)
# sudo make install
# make clean
# ldconfig
# make clean
```

- 配置环境变量

```
# vim /etc/profile
LD_LIBRARY_PATH=/usr/local/lib:/usr/local/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

> *执行source命令，环境变量在当前会话立即生效*

```
# source /etc/profile
```

> *再次执行ldconfig，确保共享链接库可以正常使用*

```
# ldconfig
```

- 验证安装

> *可以访问gRPC源码包目录下的的example示例程序route_guide，验证是否安装成功*

```
# cd examples/cpp/route_guide
# make
# ./route_guide_server &
# ./route_guide_client
```

执行客户端程序

```
# ./route_guide_client
DB parsed, loaded 100 features.
-------------- GetFeature --------------
GetFeature rpc failed.
GetFeature rpc failed.
-------------- ListFeatures --------------
```

> *终端出现如上信息，则表示gRPC运行环境就绪。*

17.0.3 安装PHP探针

PHP探针包采用源码包封发方式，由用户自行在目标环境进行编译安装，后续将考虑分发不同架构平台的二进制版本。

下载探PHP探针安装包文件 apusicapm-php-sdk-4.1.1.tar.gz到目标PHP服务器运行环境，解压编译安装。

- 安装探针

```
# tar -zxvf apusicapm-php-sdk-4.1.1.tar.gz
# cd apusicapm-php-sdk-4.1.1
# phpize
# ./configure --with-php-config=/usr/local/php/bin/php-config
# make
# make install
```

安装后显示：

```
# make install
Installing shared extensions:     /usr/local/php/lib/php/extensions/no-debug-non-zts-20170718/
```

编译将生成skywalking.so库扩展文件。

- 配置PHP探针

在php.ini文件末尾增加如下配置启用探针，并在前台运行运行php-fpm

```
;Loading extensions in PHP
extension=skywalking.so

;enable skywalking
skywalking.enable = 1

;Set skyWalking collector version
skywalking.version = 8

;Set app code e.g. MyProjectName
skywalking.app_code = php7app
```

```
;Set grpc address
skywalking.grpc=127.0.0.1:11800
```

配置说明：

配置项	默认值	说明
extension	skywalking.so	扩展库文件名称，默认值
Skywalking.enable	1	1 启用，0 停用
skywalking.version	8	ApusicAPM后端skywalking Collector的版本，这里ApusicAPM使用的V8
skywalking.app_code	服务名称	指定服务名称，APM服务中显示的被监控的服务名
skywalking.grpc	服务地址及端口	ApusicAPM后端接收链路数据的服务IP地址及端口

- 验证探针安装

```
# .php -m | grep skywalking
skywalking
pure virtual method called
terminate called without an active exception
# php -i | grep skywalking
skywalking
Directive => Local Value => Master Value
skywalking.app_code => php7app => php7app
skywalking.authentication => no value => no value
skywalking.enable => On => On
skywalking.grpc => 127.0.0.1:11800 => 127.0.0.1:11800
skywalking.grpc_tls_enable => 0 => 0
skywalking.grpc_tls_pem_cert_chain => no value => no value
skywalking.grpc_tls_pem_private_key => no value => no value
skywalking.grpc_tls_pem_root_certs => no value => no value
skywalking.log_enable => 0 => 0
```

以上，在目标PHP应用所在的服务器PHP探针已经安装就绪。

现在可以通过重启Apache或Nginx，以及php-fpm，并访问php 应用产生请求，在ApusicAPM的web 管控台查看http请求链路追踪情况。

17.0.4 容器化部署

> *当从企业云盘的共享文件下载apusic-skywalking-php-v4.1.1.tar,具体目录结构为：*

> *内部共享->金蝶天燕知识共享->产品安装包介质->智能运维产品->产品介质->APM-agent-sdk*

- 加载docker镜像

```
$ docker load –i apusic-skywalking-php-v4.1.1.tar
$ docker images
```

- 如何使用

- 运行

```
$ docker run -d  -e SW_OAP_ADDRESS=192.168.0.1:11800 -e
SW_SERVICE_NAME=hello_skywalking -p8080:8080 apusic/skywalking-php:v4.1.1
```

其中环境变量SW_OAP_ADDRESS为APM后端服务地址，SW_SERVICE_NAME为APM代理服务名，可根据需要修改。如需挂载自己的应用，通过-v参数将自己应用所在宿主机目录挂载到docker容器目录/var/www/html即可。

- 访问

```
$ curl http://localhost:8080/index.php
```

- 验证

登录AMP控制台，访问应用性能管理，看hello_skywalking服务是否注册上去，拓扑图是否显示hello_skywalking，调用链里面是否有hello_skywalking的调用链。

18 C#/.NET代理构建部署

SkyAPM-dotnet** 提供了c#和. net标准平台的本地支持代理。

- 支持
- 这个项目目前支持针对netcoreapp2.0/netframework4.6.1或更高版本的应用程序。
- 支持的中间件、框架和库。
  - ASP.NET Core
  - .NET Core BCL types (HttpClient and SqlClient)
  - EntityFrameworkCore
  - EntityFrameworkCore.Sqlite
  - Npgsql.EntityFrameworkCore.PostgreSQL
  - Pomelo.EntityFrameworkCore.MySql]
  - CAP
  - ASP.NET
- 特征
- SkyWalking .NET Core 代理功能快速列表
  - 应用拓扑
  - 分布式跟踪
  - ASP.NET Core 诊断
  - HttpClient 诊断
  - EntityFrameworkCore 诊断

18.0.1 先决条件

- 安装 SDK
- windows上

> *下载安装文件(https://dotnet.microsoft.com/download/dotnet/thank-you/sdk-3.1.408-windows-x64-installer)并安装。*

- Linux上

```
$ wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
$ sudo dpkg -i packages-microsoft-prod.deb
$ sudo apt-get update; \
  sudo apt-get install -y apt-transport-https && \
```

```
  sudo apt-get update && \
  sudo apt-get install -y dotnet-sdk-3.1
```

- 安装Nuget包管理器
- windows上

> *下载可执行文件(https://dist.nuget.org/win-x86-commandline/latest/nuget.exe)并设置环境变量Path。*

- Linux上

```
$ sudo apt install nuget
```

18.0.2 快速开始

- 需求

> *-从v1.0.0开始，SkyAPM . net Core Agent只支持SkyWalking 8.0或更高版本。*

- 安装.NET Core 代理

> *可以运行以下命令在项目中安装SkyWalking . net Core代理。*

```
$ dotnet add package SkyAPM.Agent.AspNetCore
```

> *也可以手动在c#项目文件.csproj添加如下依赖。*

```
<PackageReference Include="SkyAPM.Agent.AspNetCore" Version="1.2.0" />
```

- 如何使用

> *设置环境变量 `ASPNETCORE_HOSTINGSTARTUPASSEMBLIES` 来激活SkyAPM .NET Core代理。*

- 添加装配名称"SkyAPM.Agent.AspNetCore"到'"aspnetcore_hostingstartupassemables"环境变量。
- 例子

```
$ dotnet new mvc -n sampleapp
$ cd sampleapp
$ dotnet add package SkyAPM.Agent.AspNetCore
```

> *- windows上*

```
$ set ASPNETCORE_HOSTINGSTARTUPASSEMBLIES=SkyAPM.Agent.AspNetCore
$ set SKYWALKING__SERVICENAME=sample_app
```

```
$ dotnet run
```

> *- Linux上*

```
$ export ASPNETCORE_HOSTINGSTARTUPASSEMBLIES=SkyAPM.Agent.AspNetCore
$ export SKYWALKING__SERVICENAME=sample_app
$ dotnet run
```

- 配置
- 安装 `SkyAPM.DotNet.CLI`

```
$ dotnet tool install -g SkyAPM.DotNet.CLI
```

- 使用`dotnet skyapm config [your_service_name] [your_servers]`生成配置文件。

```
$ dotnet skyapm config sample_app 192.168.0.1:11800
```

- 也可以手动在项目根目录添加skyapm.json。

```
{
  "SkyWalking": {
    "ServiceName": "sample_app",
    "Namespace": "",
    "HeaderVersions": [
      "sw8"
    ],
    "Sampling": {
      "SamplePer3Secs": -1,
      "Percentage": -1.0
    },
    "Logging": {
      "Level": "Information",
      "FilePath": "logs\\skyapm-{Date}.log"
    },
    "Transport": {
      "Interval": 3000,
      "ProtocolVersion": "v8",
      "QueueSize": 30000,
      "BatchSize": 3000,
```

```
      "gRPC": {
        "Servers": "192.168.0.1:11800",
        "Timeout": 10000,
        "ConnectTimeout": 10000,
        "ReportTimeout": 600000,
        "Authentication": ""
      }
```

18.0.3 容器化构建部署

> ***容器****具有很多特性和优点，如具有不可变的基础结构、提供可移植的体系结构和实现可伸缩性。*

- 安装 Docker - windows上

下载安装文件(https://desktop.docker.com/win/stable/amd64/Docker%20Desktop%20Installer.exe?utm_source=docker&utm_medium=webreferral&utm_campaign=dd-smartbutton&utm_location=header)并安装。

- Linux上

```
$ sudo apt-get update
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
$ echo \
  "deb [arch=arm64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- 快速开始
- 发布 .Net Core 应用

> *- 在将 .NET Core 应用添加到 Docker 映像之前，必须先发布该应用。*

```
$ cd sampleapp
$ dotnet publish -c Release
```

> - *windows上*

```
$ dir .\bin\Release\netcoreapp3.1\publish
```

> - *Linux上*

```
$ ls bin/Release/ netcoreapp3.1/publish
```

- 创建 Dockerfile
- docker build 命令使用 Dockerfile 文件来创建容器映像。此文件是名为"Dockerfile" 的文本文件，它没有扩展名。
  - 在包含 .csproj的目录中创建名为"Dockerfile"的文件，并在文本编辑器中将其打开 。 1）Dockerfile

```
FROM mcr.microsoft.com/dotnet/aspnet:3.1-buster-slim-arm64v8

ENV ASPNETCORE_HOSTINGSTARTUPASSEMBLIES=SkyAPM.Agent.AspNetCore

COPY bin/Release/netcoreapp3.1/publish/ App/
WORKDIR /App
ENTRYPOINT ["./service.sh"]
```

2）service.sh

```
#!/usr/bin/env sh
grpc=$SW_OAP_ADDRESS
if [ ! $grpc ]; then
    grpc="127.0.0.1:11800"
fi
echo "sw oap address:" $grpc
sed -i "s/127.0.0.1:11800/$grpc/g" skyapm.json
donet sampleapp.dll
```

> - *在终端中运行以下命令：*

```
$ docker build -t sampleapp-image -f Dockerfile .
```

- 运行

```
$ docker run -d -p5001:5001 -e SW_OAP_ADDRESS=192.168.0.1:11800 sampleapp-image
```

- 访问

```
$ curl http://localhost:5001
```

- 验证

登录AMP控制台，访问应用性能管理，看sampleapp服务是否注册上去，拓扑图是否显示sampleapp，调用链里面是否有sample的调用链。

18.1 告警事件

19 告警原理

金蝶Apusic应用性能管理如眼见发送告警的基本原理是每隔一段时间轮询后端采集器收集到的链路追踪的数据，再根据所配置的告警规则（如服务响应时间、服务响应时间百分比）进行决策判断，如果达到阈值则发送响应的告警事件信息。

发送告警信息是以线程池异步的方式调用webhook接口完成，可以无缝对接AAlarm智能告警平台支持更多高级通知策略和通知方式进行告警事件通知。

20 告警相关配置

开启AAPM相关告警配置，编辑 config/alarm-settings.yml，打开之后如下所示：rules即为需要配置的告警规则的列表；第一个规则'endpoint_percent_rule'，是规则名，不能重复且必须以'_rule'为结尾。

告警规则配置属性：

属性	说明
metrics-name	指定的规则（与规则名不同，这里是对应的告警中的规则map，具体可查看 https://github.com/apache/skywalking/blob/master/docs/en/setup/backend/backend-alarm.md#list-of-all-potential-metrics-name ，其中一些常见的，endpoint_percent_rule——端点相应半分比告警，service_percent_rule——服务相应百分比告警）
threshold	阈值，与metrics-name和下面的比较符号相匹配
op	比较操作符，可以设定>,<,=，即如metrics-name: endpoint_percent, threshold: 75，op: < ,表示如果相应时长小于平均75%则发送告警
period	多久检查一次当前的指标数据是否符合告警规则
counts	达到多少次告警后，发送告警消息
silence-period	在多久之内，忽略相同的告警消息
message	告警消息内容

webhook接口url的定义（地址自定义），除了规则制定之外，还有达到告警规则后，需要skywalking调用的webhook接口，如下所示的配置，一定要注意url的缩进， 配置完成之后，重启ApusicAPM生效。

```
rules:
  # Rule unique name, must be ended with `_rule`.
  service_resp_time_rule:
    metrics-name: service_resp_time
    op: ">"
    threshold: 1000
    period: 10
    count: 3
    silence-period: 5
    message: Response time of service {name} is more than 1000ms in 3 minutes
of last 10 minutes.
  service_sla_rule:
    # Metrics value need to be long, double or int
    metrics-name: service_sla
```

```
    op: "<"
    threshold: 8000
    # The length of time to evaluate the metrics
    period: 10
    # How many times after the metrics match the condition, will trigger
alarm
    count: 2
    # How many times of checks, the alarm keeps silence after alarm
triggered, default as same as period.
    silence-period: 3
    message: Successful rate of service {name} is lower than 80% in 2 minutes
of last 10 minutes
  service_resp_time_percentile_rule:
    # Metrics value need to be long, double or int
    metrics-name: service_percentile
    op: ">"
    threshold: 1000,1000,1000,1000,1000
silence-period: 5
message: Percentile response time of service {name} alarm in 3 minutes of
last 10 minutes, due to more than one condition of p50 > 1000, p75 > 1000,
p90 > 1000, p95 > 1000, p99 > 1000
service_instance_resp_time_rule:
metrics-name: service_instance_resp_time
    op: ">"
    threshold: 1000
    period: 10
    count: 2
    silence-period: 5
    message: Response time of service instance {name} is more than 1000ms in
2 minutes of last 10 minutes
  database_access_resp_time_rule:
    metrics-name: database_access_resp_time
    threshold: 1000
    op: ">"
    period: 10
    count: 2
    message: Response time of database access {name} is more than 1000ms in 2
minutes of last 10 minutes
  endpoint_relation_resp_time_rule:
```

```
    metrics-name: endpoint_relation_resp_time
    threshold: 1000
    op: ">"
    period: 10
```

图9-1告警配置文件

21 告警列表

告警列表分别从服务、实例,端点维度展示ApusicAPM监控服务报警情况。

图 9-2报警管理

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com