应用性能管理
2.0
常见问题

文档版本 01

发布日期 2025-01-20

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
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。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 APM 探针和其他产品探针（如 Pinpoint）是否兼容?

APM 探针和其他产品探针都不兼容。

APM大多是基于ASM框架进行字节码插桩实现的，同时安装两个探针相当于对您的代码插桩两次，而不同产品的插桩代码实现不同，代码冲突可能造成严重的性能问题。

因此，不要将其他产品的探针与APM 探针同时安装，以确保应用的稳定性。

2 APM 指标数据采样策略是什么?

在使用APM服务过程中用户开启APM数据采集开关后，APM仅采集应用性能指标及调用链相关数据，不涉及个人隐私数据，详细内容请参见**数据采集**。

APM可以通过非侵入方式采集APM 探针提供的应用数据、基础资源数据、用户体验数据等多项指标。

指标数据周期性完整采集，默认采集周期为1分钟。

3 APM 指标数据未采集上来，可能有哪些原因?

1. 如果APM探针刚接入，可以稍微等待几分钟后，就能看到数据。
2. 如果显示数据采集被停止，可能的原因有：
   - 实例级别停止，APM探针管理中采集被停止。
   - 监控项级别停止，监控项状态列表某些监控项采集被人为停止。
   - APM控制台的“全局配置（系统管理 > 通用配置）”中字节码方式采集被停止。
3. 如果长时间未采集到指标数据，可能的原因有：
   - java9启动提示找不到sql.time类

     原因分析：APM探针开发环境为jdk1.7，而java 9模块化后，sql包不会默认提供，需要应用引入模块。

     出现概率：有条件出现。

     规避措施：如果出现该问题，组件在module-info.java主动引入java.sql。
   - java11提示找不到Caused by: java.lang.NoClassDefFoundError: sun/misc/Unsafe类

     原因分析：APM探针开发环境为jdk1.7，而java 11Unsafe类已经重新归到其他包下，需要使用兼容模式。

     出现概率：必然出现。

     规避措施：如果出现该问题，应用在module-info.java主动引入jdk.unsupported。
   - java9提示反射使用告警，后期针对java9以上版本会避免这个问题

     规避措施：设置illegal-access = warn （仅提示）或者删除该选项。

4 为什么 CCE 开启 java 探针后，APM 无监控数据?

CCE开启java探针后，APM无监控数据，可能是由于用户使用的java探针版本过低或者用户使用Tomcat服务启动的java探针。

解决改问题的方法如下：

1. 在APM控制台，免费开通APM 2.0（免费版可以使用10个探针），具体操作参见**开通APM 2.0**。
2. 购买APM企业版，具体操作参见**“应用列表 -> 订购APM企业版”**。
3. 请用户使用最新版本的java探针，重启容器后APM界面显示正常。

5 微信小程序接入有哪些方式?

微信小程序接入方式，与其他小程序有些不同。目前支持两种方式接入。

方式一：采用 npm 方式集成 SDK

1. 确保项目有“package.json”文件，如果项目中没有“package.json”文件，可以在项目的根目录下，使用以下命令来创建。
```
npm init
```
2. 运行安装SDK的命令 ，安装SDK软件包。
```
npm i apm-mini-sdk
```
3. 单击开发者工具菜单栏中的“工具 > 构建npm”，构建当前工程的npm库文件。

图 5-1 构建 npm

4. 在app.js文件中使用import agent from ‘apm-mini-sdk’。

方式二：使用文件引入方式

1. 运行安装SDK的命令，安装SDK软件包。
   ```
   npm i apm-mini-sdk
   ```
2. 找到SDK文件夹中的app.js文件“node_modules > apm-mini-sdk > app.js”，将app.js文件从node_module复制到根路径并改名。

6 Debugging 诊断的方法分析功能，不支持重载嵌套调用的下钻。

现象

步骤1 登录管理控制台。

步骤2 单击左侧 ，选择“管理与监管 > 应用性能管理 APM”，进入APM服务页面。

步骤3 在左侧导航栏选择“应用监控 > 指标”。

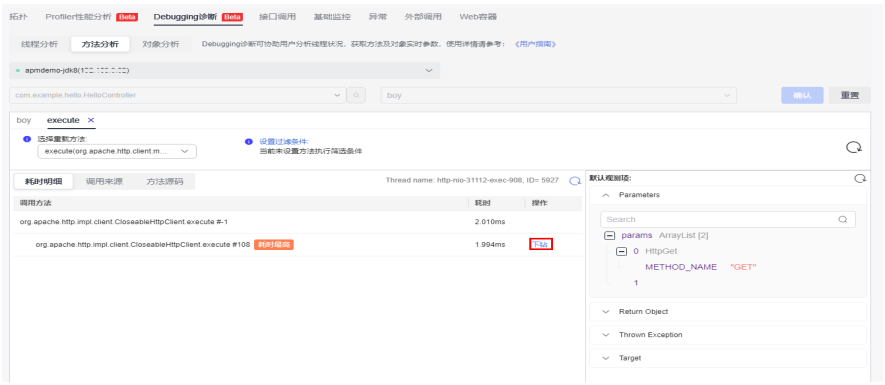
步骤4 在界面左侧树，单击环境后的 。

步骤5 单击“Debugging诊断”，切换至Debugging诊断页签。

步骤6 单击“方法分析”，进入方法分析页面。

步骤7 输入类名称：com.example.hello.helloController，并选择方法名为boy，单击“确认”。

步骤8 单击“org.apache.http.impl.client.CloseableHttpClint.execute #-1”对应的“下钻”。



步骤9 单击“org.apache.http.impl.client.CloseableHttpClint.execute #108”对应的“下钻”，没有任何反应。

----结束

触发场景

当Debugging的观测类中存在方法重载时，即类中存在多个同名的函数且存在嵌套调用。只能追踪到首个被调用的方法，即execute方法中又调用了execute方法，则无法下钻。

根因及约束

前端watch、trace命令没有指定重载的方法参数列表，所以Debugging无法返回精确的方法信息，只能返回首个方法。问题中execute方法内部又调用了execute方法，当前的实现无法支持这种场景。

因此，Debugging暂时不支持重载方法嵌套调用的下钻功能。

7 Profiler 性能分析常见问题

在Profiler性能分析的火焰图无数据或其他异常情况下，可以查看profiler日志确定问题原因。

日志路径：

```
cd ~/apm/instances/<应用名称>/logs/profiler/
```

7.1 perf_event_open 被限制导致的 No access to perf events 报错问题

问题现象

CPU Profiler依赖perf_event_open的系统调用，但因为Linux kernel的Syscall安全策略（**seccomp**）控制，可能会禁止进程调用特定Syscall。

错误提示如下：

```
[ERROR] xxxx Failed to execute
'start,jfr=7,jstackdepth=100,threads=true,event=cpu,interval=50ms,alloc=512k,wall=50ms,file=xxxx.jfr'
[ERROR] xxxx Failed to start Continuous Profile Collector
[ERROR] xxxx No access to perf events. Try --fdtransfer or --all-user option or 'sysctl
kernel.perf_event_paranoid=1'
```

解决方案

- Docker环境：执行以下命令运行容器。如需配置更精细化的系统调用控制，请参见**官方文档**。开启特权容器存在容器逃逸风险，请评估后使用。
  ```
  docker run --security-opt seccomp=unconfined  XXX
  ```
- Kubernetes环境：配置特权容器参数privileged: true，特权容器始终保持为**Unconfined**。如果无法配置特权容器参数privileged: true，可以通过修改k8s的yaml文件中的securityContext配置，使用默认的系统调用控制，请参见**官方文档**。开启特权容器存在容器逃逸风险，请评估后使用。
  ```
  apiVersion: v1
  kind: Pod
  metadata:
    name: default-pod
    labels:
      app: default-pod
  spec:
    containers:
      - name: test-container
  ```



```
image: hashicorp/http-echo:1.0
args:
  - "-text=just made some more syscalls!"
securityContext:
  seccompProfile:
    type: RuntimeDefault
  capabilities:
    add: [ "SYS_ADMIN" ]
  privileged: false
```

7.2 No AllocTracer symbols found .Are JDK debug symbols installed?报错问题

常见问题

内存Profiler依赖JDK的符号信息，如果JDK内无符号信息，则会遇到如下问题：

[ERROR] xxxx Failed to start Continuous Profile Collector

[ERROR] xxxx No AllocTracer symbols found. Are JDK debug symbols installed?

解决方案

如果Java进程运行在容器环境，出现以上报错或者该功能无数据，一般都是由于使用了Alpine基础镜像导致。Alpine基础镜像为了控制体积而去除了JDK调试符号（debug symbols），影响Profiler功能正常使用。建议在基础镜像中为JDK安装调试符（部分JDK版本缺乏对应的调试符包，会导致无法安装）或使用非Alpine基础镜像。

如果是CentOS物理机环境部署应用可通过以下步骤安装调试符：

步骤1 执行以下命令，确认是否已经配置了debuginfo的源。

```
yum repolist all | grep -i debug
```

通过返回信息确认源配置信息是否如下：

```
debuginfo/7/x86_64 CentOS-7 - debuginfo - mirrors.huaweicloud.com 启用: 8,760
```

如果未配置debuginfo的源，则执行**步骤2**增加配置；如果已经配置，则执行**步骤3**。

步骤2 在Yum配置文件中增加debuginfo的配置。

```
vi /etc/yum.repos.d/CentOS-Base.repo
[debuginfo]
name=CentOS-$releasever - debuginfo - mirrors.huaweicloud.com
failovermethod=priority
baseurl=http://mirrors.huaweicloud.com/centos-debuginfo/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=http://mirrors.huaweicloud.com/centos/RPM-GPG-KEY-CentOS-Debug-7
```

保存后执行如下命令：

```
yum clean all
yum makecache
yum-config-manager --enable debuginfo
```

步骤3 安装JDK版本对应的符号信息。

```
JDK 8
yum list installed|grep openjdk    #查询JDK版本。
java-1.8.0-openjdk.x86_64          1:1.8.0.332.b09-1.el7_9          @updates
debuginfo-install -y java-1.8.0-openjdk      #安装符号信息。
JDK 11
```

```
yum list installed|grep openjdk    #查询JDK版本。
java-11-openjdk.x86_64              1:11.0.15.0.9-2.el7_9              @updates

debuginfo-install -y java-11-openjdk      #安装符号信息。
```

步骤4 检查JDK的符号信息是否已经安装成功。

```
gdb $JAVA_HOME/lib/server/libjvm.so -ex 'info address UseG1GC'
gdb /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.332.b09-1.el7_9.x86_64/jre/lib/amd64/server/libjvm.so -ex 'info
address UseG1GC'
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-120.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.332.b09-1.el7_9.x86_64/jre/lib/amd64/server/
libjvm.so...Reading symbols from /usr/lib/debug/usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.332.b09-1.el7_9.x86_64/jre/lib/amd64/server/libjvm.so.debug...done.
done.
```

如果安装正确，返回信息如下：

```
Symbol "UseG1GC" is static storage at address 0x107f993.
```

如果安装失败，返回信息如下：

```
No symbol "UseG1GC" in current context.
```

----结束

7.3 perf_event mmap failed...报错问题

问题现象

此错误一般出现在JVM的标准输出中。Profiler功能进行CPU热点采样时，会同时采集Native（Linux Kernel + JVM + C/C++）以及Java栈，采集Native栈需要对Java中每个线程的perf_event的fd进行MMap，Linux内核中限制了进程perf_event相关的MMap的总内存大小（默认516 K Bytes）。当Java中线程数较多时，会触发限制并在Java标准输出中打印警告信息perf_event mmap failed...。出现这个告警信息，对Java的运行没有副作用，对业务也没有影响，实际的影响是火焰图中看不到Native的栈。一般来说定位CPU热点问题时，只看Java方法栈就够了，您可以忽略此告警。

解决方案

如果想消除这个错误信息，可以执行以下步骤。

步骤1 在宿主机上执行以下命令。

```
echo 1028 > /proc/sys/kernel/perf_event_mlock_kb
```

默认阈值是516，可以逐渐增加，直到不出现告警。该值尽量满足8*N + 4，N是自然数。例如516 = 512 + 4, 1028 = 1024 + 4。

步骤2 重启Docker，即可消除错误。

----结束

7.4 libz.so.1: version `ZLIB_1.2.9' not found

问题现象

报错信息：java.lang.UnsatisfiedLinkError: /jre/lib/amd64/libfontmanager.so: /apm-javaagent-profiler/apm-javaagent/native-agent/x86/libz.so.1: version `ZLIB_1.2.9' not found (required by /usr/lib64/libpng16.so.16)

这个错误出现在服务日志中，服务程序调用JDK的libfontmanager.so中的方法。出现该错误的原因是由于该so依赖libz.so.1的1.2.9版本，与javaagent中的libz.so.1版本不兼容。

解决方案

1. 手动删除javaagent下的libz.so.1文件即可，这样服务程序将使用系统目录下的libz.so.1。
2. 升级探针版本，2.4.14-profiler后的探针版本已修复该问题。

8 APM Android SDK 和其他同类产品是否兼容?

APM工具通常基于ASM框架进行字节码插桩。这种技术允许开发者动态修改应用程序的字节码，以便在不改变源代码的情况下监控性能。

然而，若同时安装多个APM工具，会导致代码多次插桩。不同产品的实现可能相互冲突，进而引发编译错误和性能问题。例如，一个工具可能会修改某个方法的字节码，而另一个工具则可能尝试在同一位置进行修改，这种冲突会导致运行时异常或不一致的行为。此外，频繁的插桩可能还会增加应用的启动时间和运行开销，给性能带来额外负担。

因此，建议在一个项目中只选择安装一个APM工具，以确保应用的稳定性和优化性能。

9 Java 增强探针性能对用户性能是否会产生影响?

应用性能管理（APM）探针利用字节码增强技术动态采集性能数据，包括方法调用、异常信息、分布式追踪等，帮助开发和运维团队实时监控和优化系统性能。探针集成不可避免地引入了一定量的性能损耗，通过评估探针对Java应用的性能开销（如CPU、内存、延迟），高负载场景下的稳定性和数据采集完整性，可以确保探针在生产环境中运行可靠，性能影响可控，为优化和部署提供重要依据。

测试环境

工具/服务	版本/规格	说明
JVisualVM	1.8.0_216	JVisualVM是Java自带性能监控工具，监视和管理控制台JConsole，它可以提供 Java 某个进程的内存、线程、类加载、jvm 概述以及的实时信息。
JMeter	5.3	Apache JMeter是Apache组织开发的基于Java的压力测试工具，在本次测试中主要是用于模拟多用户并发调用APM查询图表接口。
JavaAgent	2.4.11-profiler	Java Agent稳定版本
ECS服务器 Demo应用	2u4g benchmark.jar	通用计算增强型 | 2vCPUs | 4GiB | c7.large.2，节点的操作系统版本为CentOS 7.9。 根据压测源发起请求，会同时访问MySQL和Redis服务，并返回查询值，使用Spring Cloud、Dubbo实现。

约束限制

1. Profiler探针版本需要2.4.5版本及以上，请参见**JavaAgent更新说明**。
2. 采样策略设置探针版本需要2.4.11版本及以上，请参见**JavaAgent更新说明**。

测试流程

步骤1 在不安装探针的情况下，分别使用1TPS、500TPS、1000TPS、2000TPS压测样本，每次的持续时长为30分钟，压测结果将作为基线性能指标。

步骤2 安装探针，采样策略设置为智能采样和100%采样两种情况下，重复步骤1的压测过程，对比CPU、内存、RT上的差异。

步骤3 安装带Profiler探针，性能剖析设置为关闭状态，采样策略设置为智能采样和100%采样两种情况下，重复步骤1的压测过程，对比CPU、内存、RT上的差异。

----结束

未安装探针性能极限指标

序号	压测样本	RT（ms）	cpu（%）	内存（MB）
1	1TPS	77.05	0.4	200
2	500TPS	77.42	11	250
3	1000TPS	79.17	23	300
4	2000TPS	83.19	45	350

安装探针性能极限指标

表 9-1 智能采样

序号	压测样本	RT（ms）	CPU（%）	内存（MB）
1	1TPS	78.36	0.4	250
2	500TPS	79.05	18	300
3	1000TPS	81.84	31	350
4	2000TPS	86.81	55	400

表 9-2 100%采样

序号	压测样本	RT（ms）	CPU（%）	内存（MB）
1	1TPS	78.38	0.4	250
2	500TPS	80.39	20	300
3	1000TPS	84.51	33	450
4	2000TPS	88.72	65	500

探针性能开销对比

序号	压测样板	智能采样对比			100%采样率对比		
-		RT	CPU	内存	RT	CPU	内存
1	1TPS	+1.31ms	+0%	+50MB	+1.33ms	+0%	+50MB
2	500TPS	+1.63ms	+7%	+50MB	+2.97ms	+9%	+50MB
3	1000TPS	+2.67ms	+8%	+50MB	+5.34ms	+10%	+150MB
4	2000TPS	+3.62ms	+10%	+50MB	+5.53ms	+20%	+150MB

报告结论

1. JAVA增强型探针对于RT（请求响应时间）影响非常小。
2. JAVA增强型探针默认智能采样会额外造成的CPU和内存开销增长，增长幅度小于主机总量的10%，实际还会随客户应用复杂性有所增加。
3. JAVA增强型探针在100%采样率的情况下，性能开销比智能采样略有上升，在高负载情况下不建议开启100%采样。

10 什么是 Apdex?

Apdex全称是Application Performance Index，是由Apdex联盟开发的用于评估应用性能的工业标准。Apdex标准从用户的角度出发，将对应用响应时间的表现，转为用户对于应用性能的可量化范围为0-1的满意度评价。

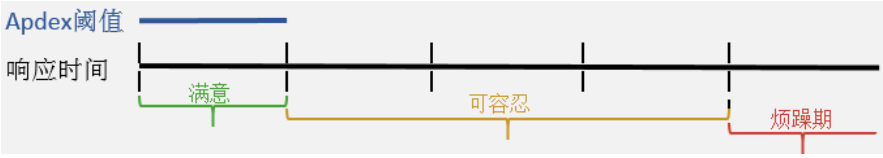
- Apdex的原理

  Apdex定义了应用响应时间的门槛为T（即Apdex阈值，T由性能评估人员根据预期性能要求确定），然后根据应用响应时间结合T定义了三种不同的性能表现：

  Satisfied（满意）：应用响应时间低于或等于T，比如T为1.5s，则一个耗时1s的响应结果则可以认为是satisfied的。

  Tolerating（可容忍）：应用响应时间大于T，但同时小于或等于4T。假设应用设定的T值为1s，则4*1=4s为应用响应时间的容忍上限。

  Frustrated（烦躁期）：应用响应时间大于4T。

- APM如何计算Apdex

  APM中，Apdex阈值即请求响应达到满意程度的最大时间。应用响应时延即服务时延，Apdex取值范围为0～1，计算公式如下：

  Apdex=（满意样本+可容忍样本*0.5）/样本总数