



GOTC 2023

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE, INTO THE FUTURE

专场

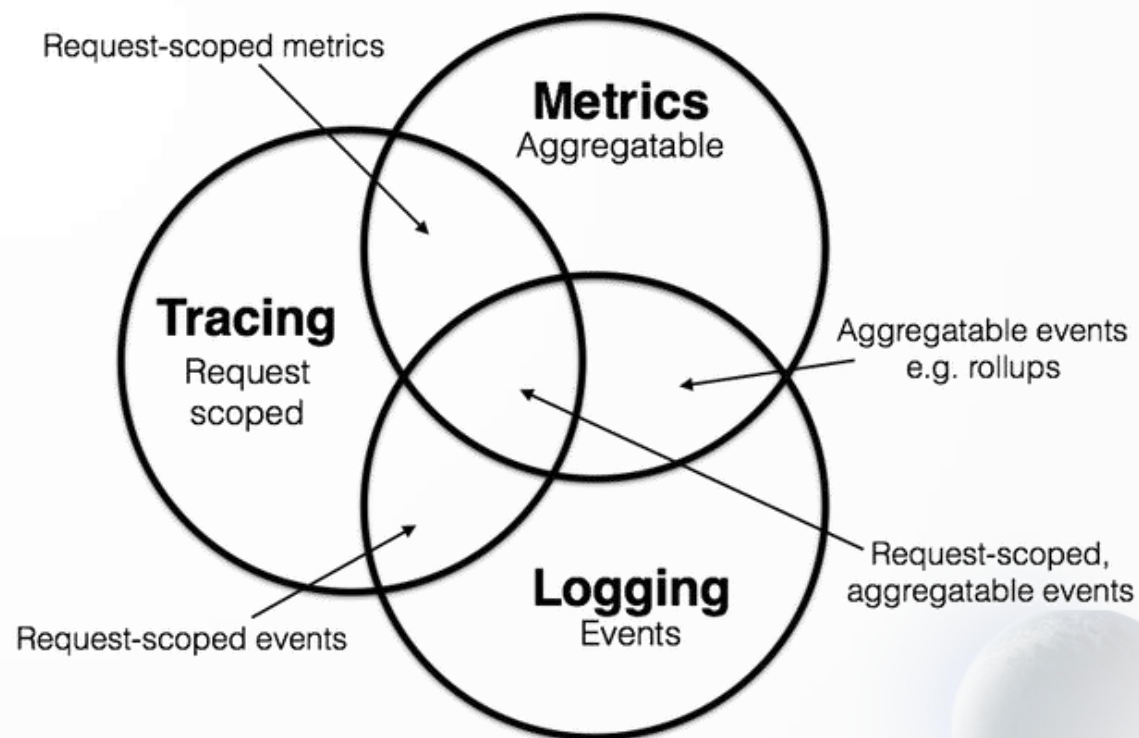
本期议题：基于eBPF程序摄像头构建的运维北极星指标体系

嘉宾姓名 苒程 2023年05月

tracing、metrics、logging已经成为绝大多数云原生业务的标配

1-5-10仍然很难

- 1分钟发现问题
- 5分钟找到故障初因
- 10分钟恢复业务



故障初发阶段

工单体系：XXX业务时快时慢

运维工程师：利用tracing系统快速找到疑似的service或者pod

运维工程师：检查Pod与Node的资源饱和度

运维工程师：各种资源利用率看上去还好，日志好像也没有很严重的异常

运维工程师：放大招重启POD

运维工程师：ε(┏┓┗┓)3 问题没有解决

架构师与研发介入

架构师：日志体系正常，先看看tracing中哪段代码慢

架构师：拉入这段代码负责人，这段代码为啥突然慢了

业务研发：我去仔细分析下源码，晚点回复。。。

架构师：之前我遇到一个问题，request与limit配置不对，导致资源在申请过程中申请不到资源。抓紧时间检查下request与limit的配置

架构师：之前遇到过GC导致的问题，重点看下GC日志，是不是GC导致的

架构师：之前我遇到过网络问题，重点排查下网络问题，有没有丢包，有没有重传

故障定位当下是个盲人摸象的过程

依赖人工

依赖经验



排障周期不可预期

Kindling的解题之道

——构建排障北极星指标体系与标准化的步骤

有没有可能创建一套类似于北极星指标指引排障过程



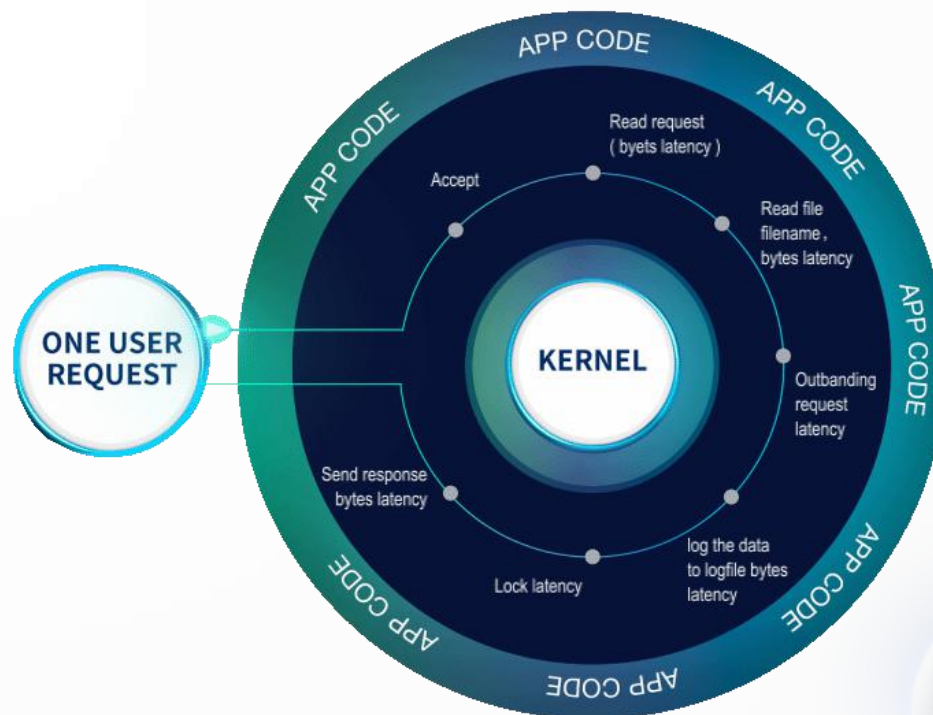
Kindling

eBPF-based Cloud Native Monitoring tool

Understanding the app behavior from kernel to app code

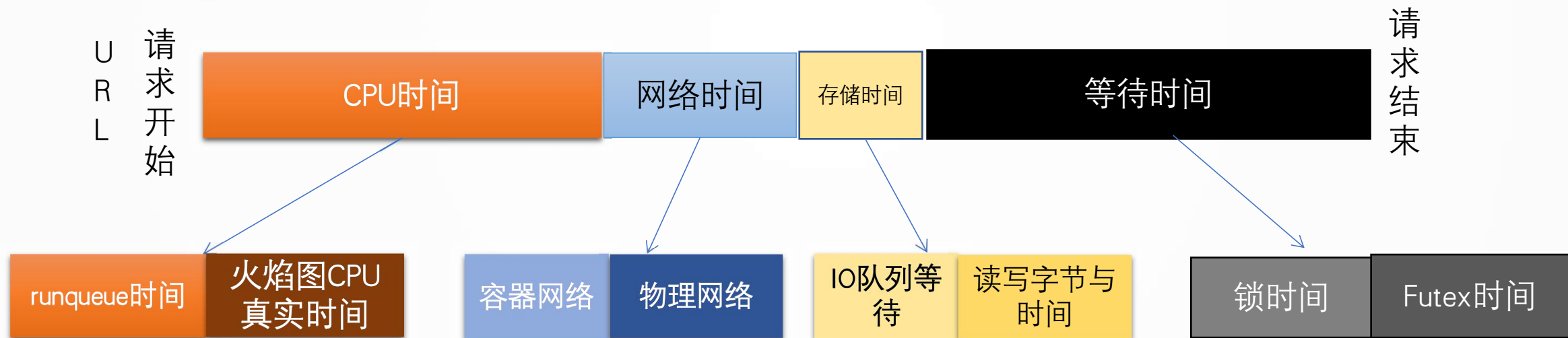
Get Started

Watch the Demo



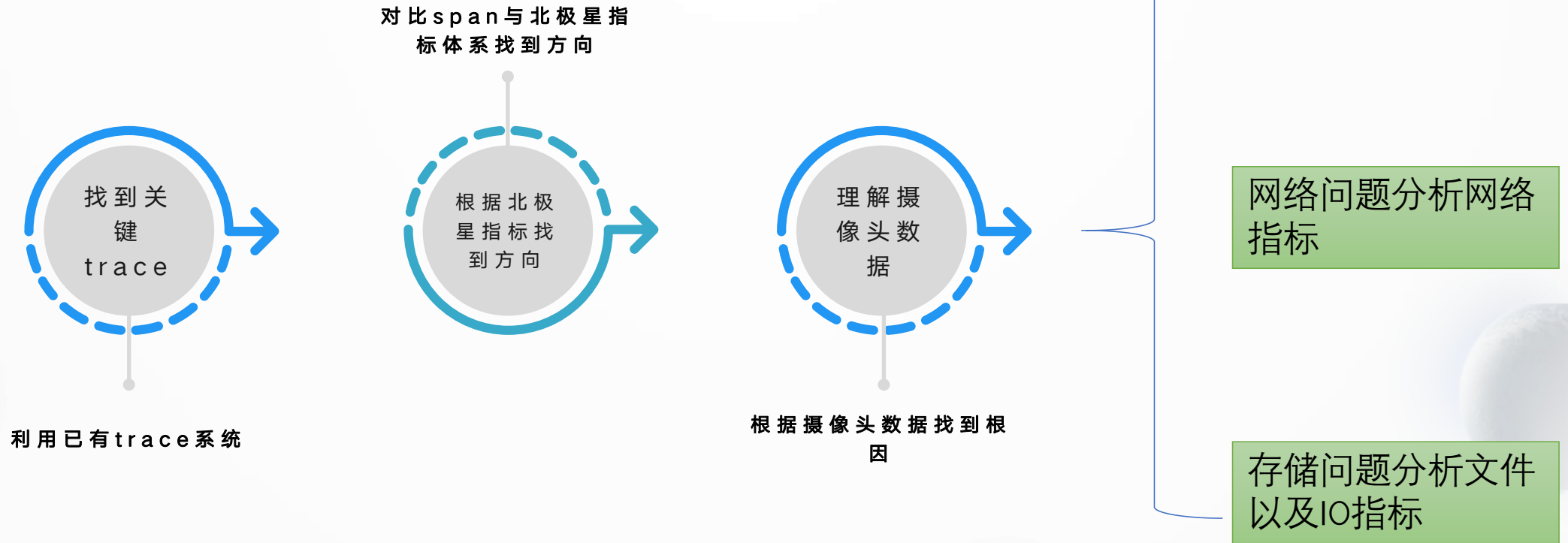
Kindling程序摄像头trace-profiling的排障北极星指标体系

对程序摄像头数据进行分析得到程序执行过程的北极星指标



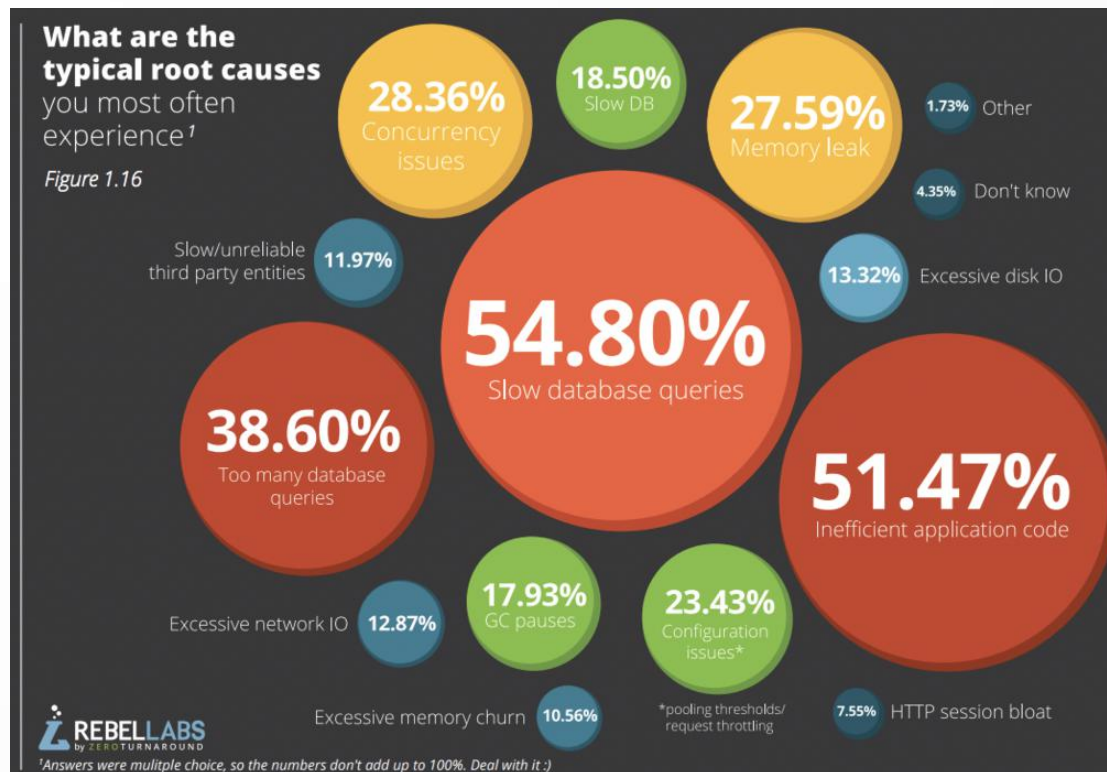
Trace_profiling程序摄像头的核心价值

核心价值：以标准化流程，分钟级定位全资源种类故障的根因



Trace_profiling程序摄像头的能够覆盖的场景

- 理论上所有程序上的故障最终都会在资源层面得以表现
 - CPU (cpu上执行了死循环或者递归太多)
 - 网络 (网络系统调用先出现问题)
 - 存储 (VFS接口明确存储接口是否有问题)
 - 内存 (应用层的垃圾回收, 内核层的pagecache等回收)



用例讲解

UseCase1：生产环境CPU不定时飙高

常规解题方法：

1. 登录上CPU不定时飙高的机器
2. 排查占用CPU的进程
3. 找出实际占用最高CPU的线程
4. 用jstack获取对应线程的堆栈信息，找出耗CPU的代码位置对应修复

不足之处：

1. 现场并未保留，难以重现，CPU在某些地方卡住比较适用常规解法
2. 耗时，由于不定时飙高，需要找CPU飙高的规律，但是规律难寻
3. 人有误操作可能
4. Jstack性能消耗大，没有办法一直常开
5. 不同业务URL都会占用一定的CPU使用量，很难定位到底是哪个业务导致的CPU使用量比较高，从而会被误导排查很多正常使用的CPU代码

UseCase1：生产环境CPU不定时飙高

预备知识：

1. 序列化非常消耗CPU

案例说明：

对一个6M大小的对象用不同的序列化框架进行序列化操作，并捕捉了请求Trace，耗时如下

- Fastjson: 285.00ms
- Jackson: 255.29ms
- Gson : 283.08ms
- net.sf.json: 655.06ms

UseCase1：生产环境CPU不定时飙升——fastjson



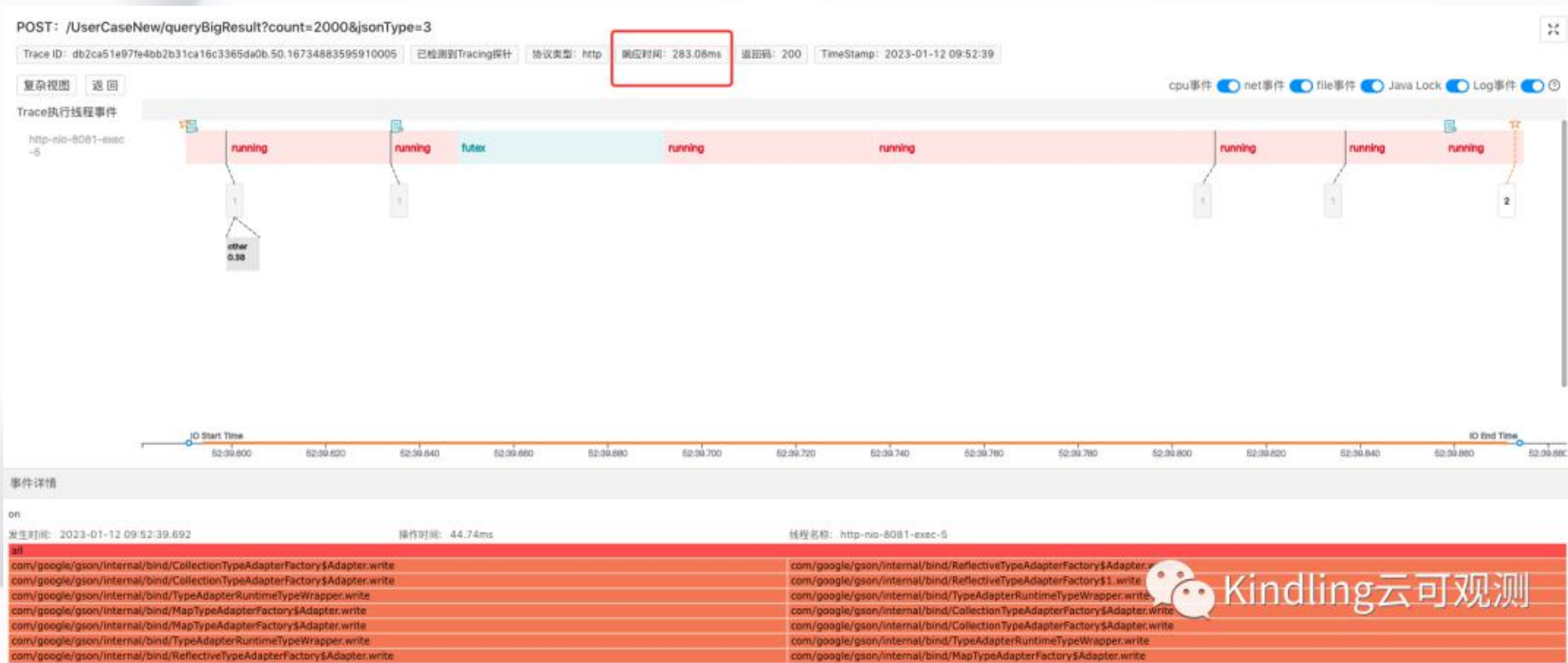
UseCase1：生产环境CPU不定时飙升——Jackson 255.29ms



全球开源技术峰会

THE GLOBAL OPEN SOURCE TECHNOLOGY CONFERENCE

UseCase1: 生产环境CPU不定时飙升——Gson 283.08ms



UseCase1：生产环境CPU不定时飙升——net.sf.json

POST: /UserCaseNew/queryBigResult?count=2000&jsonType=4



Trace ID: db2ca51e97fe4bb2b31ca16c3385da0b.48.16734884866590007

已检测到Tracing探针

协议类型: http

响应时间: 655.06ms

返回码: 200

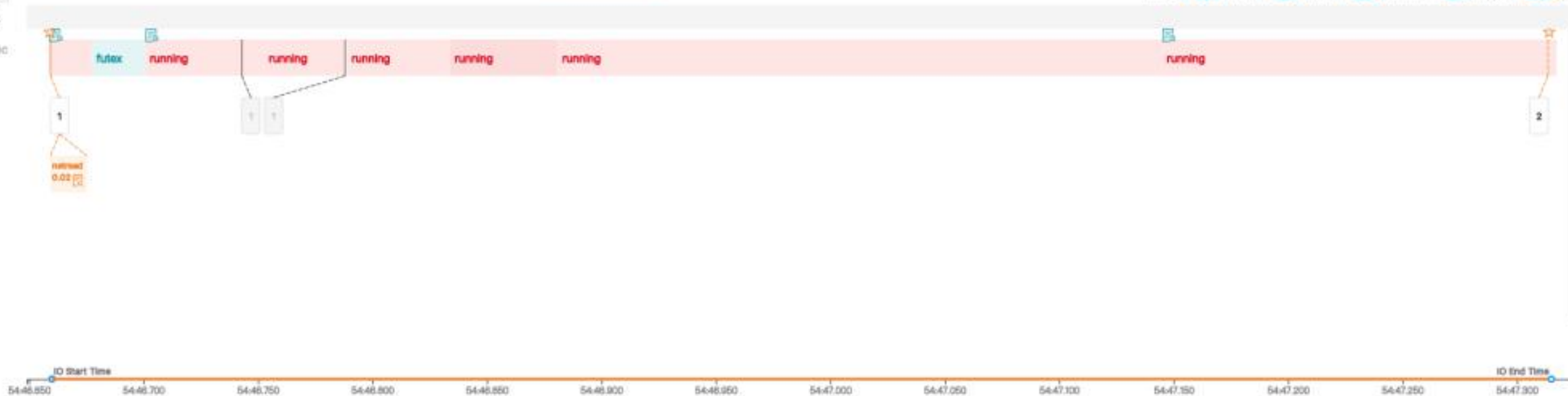
TimeStamp: 2023-01-12 09:54:46

复杂视图 返回

cpu事件 net事件 file事件 Java Lock Log事件

Trace执行线程事件

http-nio-8081-exec-3



事件详情

on

发生时间: 2023-01-12 09:54:46.833

操作时间: 46.96ms

线程名称: http-nio-8081-exec-3

all

[unknown_Java]

net/sf/json/JSONObject._processValue

net/sf/json/AbstractJSON._processValue

net/sf/json/JSONObject._fromObject

net/sf/json/JSONObject._fromBean

net/sf/json/JSONObject.defaultBeanProcessing

net/sf/json/JSONObject.setValue

net/sf/json/JSONObject.setInternal

net/sf/json/JSONObject.processValue

net/sf/json/JSONArray.addValue

net/sf/json/JSONArray.processValue

net/sf/json/JSONArray._processValue

net/sf/json/AbstractJSON._processValue

net/sf/json/JSONObject._fromObject

net/sf/json/JSONObject._fromBean

net/sf/json/AbstractJSON.removeInstance

net/sf/json/JSONObject.setValue

net/sf/json/JSONObject.setInternal

net/sf/json/JSONObject._processValue

net/sf/json/AbstractJSON._processValue

net/sf/json/JSONObject._fromObject

net/sf/json/JSONObject._fromMap



全球开源技术峰会

THE GLOBAL OPEN SOURCE TECHNOLOGY CONFERENCE

UseCase2: k8s环境使用servicename 进行远程调用

常规解题方法:

1. 用tracing系统串联客户端与服务端数据
2. 直接排查网络质量

不足之处:

1. 当用tracing系统串联客户端与服务端数据之后, 发现Client延时数据过大, 而server端延时过小之时, 问题比较难以解释
2. 想当然的怀疑网络问题, 经过一番排查发现网络质量好像没有问题, 接下来束手无策

UseCase2: k8s环境使用servicename 进行远程调用

标准化排障步骤:

1. 找关键Trace, 通过Trace系统, 结合时间点, 找出可能存在问题的关键Trace
2. 查Span信息
3. 对比分析Span信息, 找出与预期不符的地方

GET: /UserCaseNew/dnsTest?url=http%3A%2F%2Fkindling.freehk.svipss.top%2Fsleep%2F1

Trace ID: 294a88963a4f472098b2106571eff3e6.53.16727351571500009

协议类型: http

响应时间: 1.28s

返回码: 200

TimeStamp: 2023-01-03 16:39:17

复杂视图



Trace中span执行消耗



Span信息提供不了深入的更多的线索

UseCase2: k8s环境使用servicename 进行远程调用



GET: /UserCaseNew/dnsTest?url=http%3A%2F%2Fkindling.free.svipss.top%2Fsleep%2F1



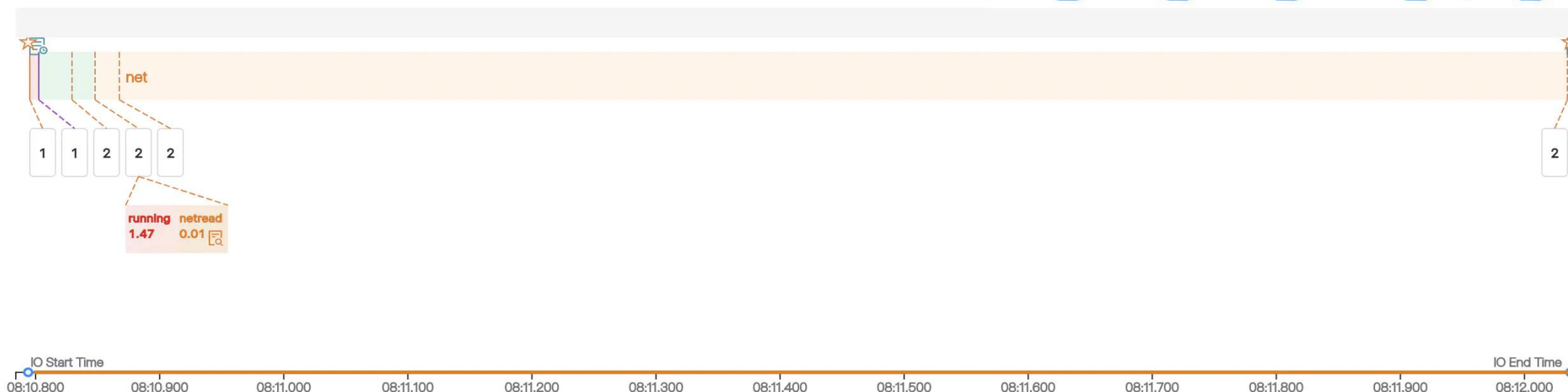
Trace ID: 81d10e659fad4cf98d1ab978d8fc083a.49.16722328907950031 协议类型: http 响应时间: 1.24s 返回码: 200 TimeStamp: 2022-12-28 21:08:10

复杂视图 返回

cpu事件 net事件 file事件 Java Lock Log事件 ?

Trace执行线程事件

http-nio-9999-exec-4



事件详情

net

发生时间: 2022-12-28 21:08:10.847

操作时间: 0.01ms

线程名称: http-nio-9999-exec-4

操作文件类型: read

操作大小: 58byte

连接信息: 10.10.103.149:60388->8.8.8.8:53

报文信息: --

相关请求信息

未来丢包, 重传, 带宽, 0窗口
等信息会关联起来

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

UseCase2: k8s环境使用servicename 进行远程调用

如果DNS有问题, Trace_profiling能够很好告诉用户

GET: /UserCaseNew/dnsTest?url=http%3A%2F%2Fkindling.free.svipss.top%2Fsleep%2F1



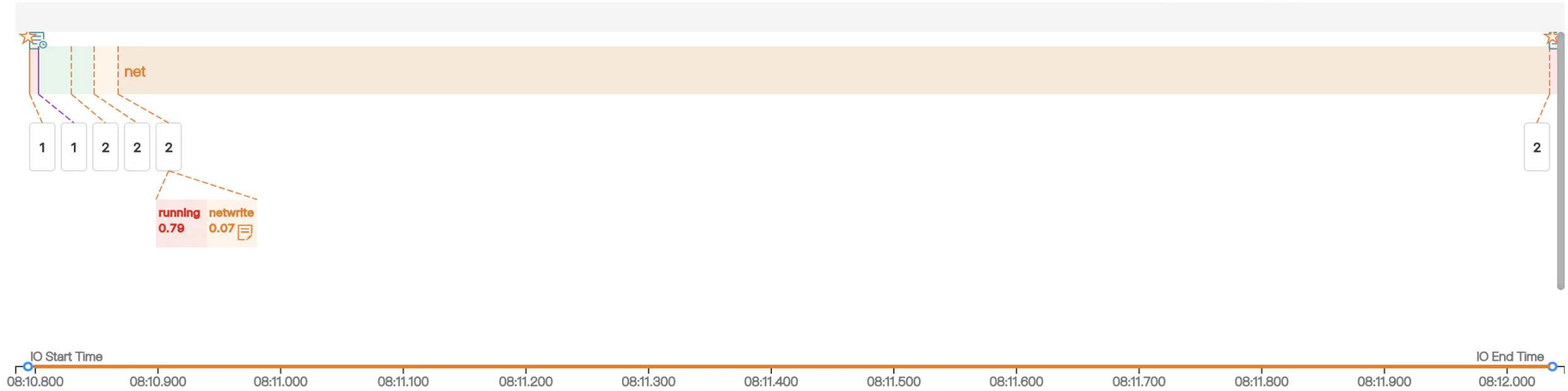
Trace ID: 81d10e659fad4cf98d1ab978d8fc083a.49.16722328907950031 协议类型: http 响应时间: 1.24s 返回码: 200 TimeStamp: 2022-12-28 21:08:10

复杂视图 返回

cpu事件 net事件 file事件 Java Lock Log事件

Trace执行线程事件

http-nio-9999-exec-4



事件详情

net
发生时间: 2022-12-28 21:08:10.867 操作时间: 1166.02ms 线程名称: http-nio-9999-exec-4 操作文件类型: read
操作大小: 202byte 连接信息: 10.10.103.149:48166->150.242.98.19:80
报文信息: HTTP/1.1 200 Content-Type: application/json Transfer-Encoding: chunked Date: Wed, 28 Dec 2022 13:07:59 GMT Keep-Alive: timeout=60 Connection: keep-alive 22 ("success":true,"data":{"Sleep 1s"})

未来丢包, 重传, 带宽, 0窗口
等信息会关联起来

UseCase3: 云原生存储出现问题

常规解题方法:

1. 查询日志从中找到蛛丝马迹

不足之处:

1. 蛛丝马迹的发现需要对代码, 对环境非常熟悉。
2. 同样的日志, 不同的人看能得到不同的理解。
3. 常规指标与监控很难发现问题

UseCase3: 云原生存储出现问题

标准化排障步骤:

1. 找关键Trace, 通过Trace系统, 结合时间点, 找出可能存在问题的关键Trace
2. 查Span信息
3. 对比分析Span信息, 找出与预期不符的地方

GET: /UserCaseNew/fileIO?filePath=%2Fopt%2FgrhTest%2Fdemo2.jar&useBuffer=true

Trace ID: 726daf9e5abe47a0acce166040202291.46.16751443101150001 已检测到Tracing探针 协议类型: http 响应时间: 1.47s 返回码: 200 TimeStamp: 2023-01-31 13:51:50

复杂视图

Trace中span执行消耗

入口span
子span

GET:/UserCaseNew/fileIO

com.harmonycloud.stuck.web.UserCaseNewController.fileIO(java.lang.Boolean,java.lang.String)



Span信息提供不了深入的更多的线索

UseCase3: 云原生存储出现问题



文件读写占了非常多的时间

文件读写占了非常多的时间

文件读写占了非常多的时间

UseCase4: 云原生环境网络TCP窗口配置有问题 ——导致1M左右报文返回延时大

常规解题方法:

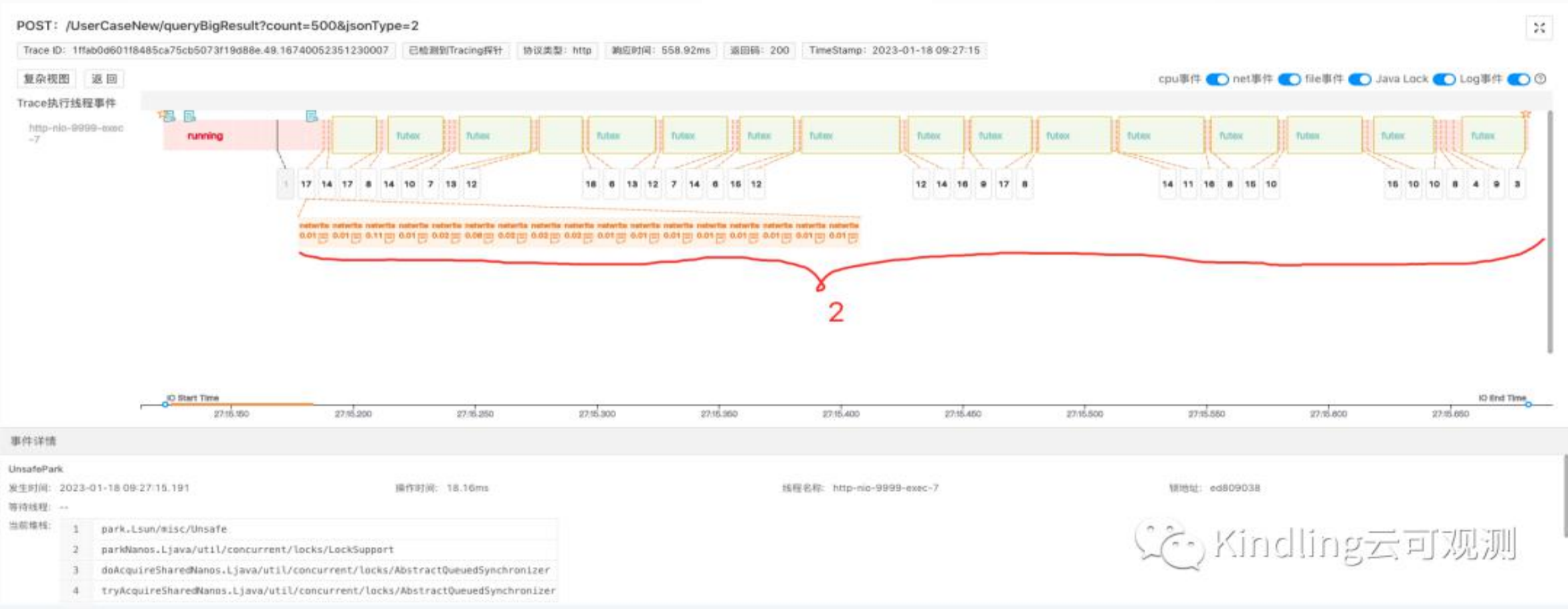
1. 查看网络带宽
2. 不断调整日志点位, 通过日志确认网络发生或者接收时间

不足之处:

1. 不同框架适配周期长。
2. 对网络理解不深的人很难意识到网络传输会有问题, 日志都不会往这个方向打

UseCase4: 云原生环境网络TCP窗口配置有问题

程序后续加锁时间较长



UseCase4: 云原生环境网络TCP窗口配置有问题

程序后续加锁时间较长，通过分析锁的代码，定位出是在网络传输时间较长，但是实际上返回报文与带宽比，不需要如此时间。

事件详情

4	<code>tryAcquireSharedNanos.Ljava/util/concurrent/locks/AbstractQueuedSynchronizer</code>
5	<code>await.Ljava/util/concurrent/CountDownLatch</code>
6	<code>awaitLatch.Lorg/apache/tomcat/util/net/NioEndpoint\$NioSocketWrapper</code>
7	<code>awaitWriteLatch.Lorg/apache/tomcat/util/net/NioEndpoint\$NioSocketWrapper</code>
8	<code>write.Lorg/apache/tomcat/util/net/NioBlockingSelector</code>
9	<code>write.Lorg/apache/tomcat/util/net/NioSelectorPool</code>
10	<code>doWrite.Lorg/apache/tomcat/util/net/NioEndpoint\$NioSocketWrapper</code>

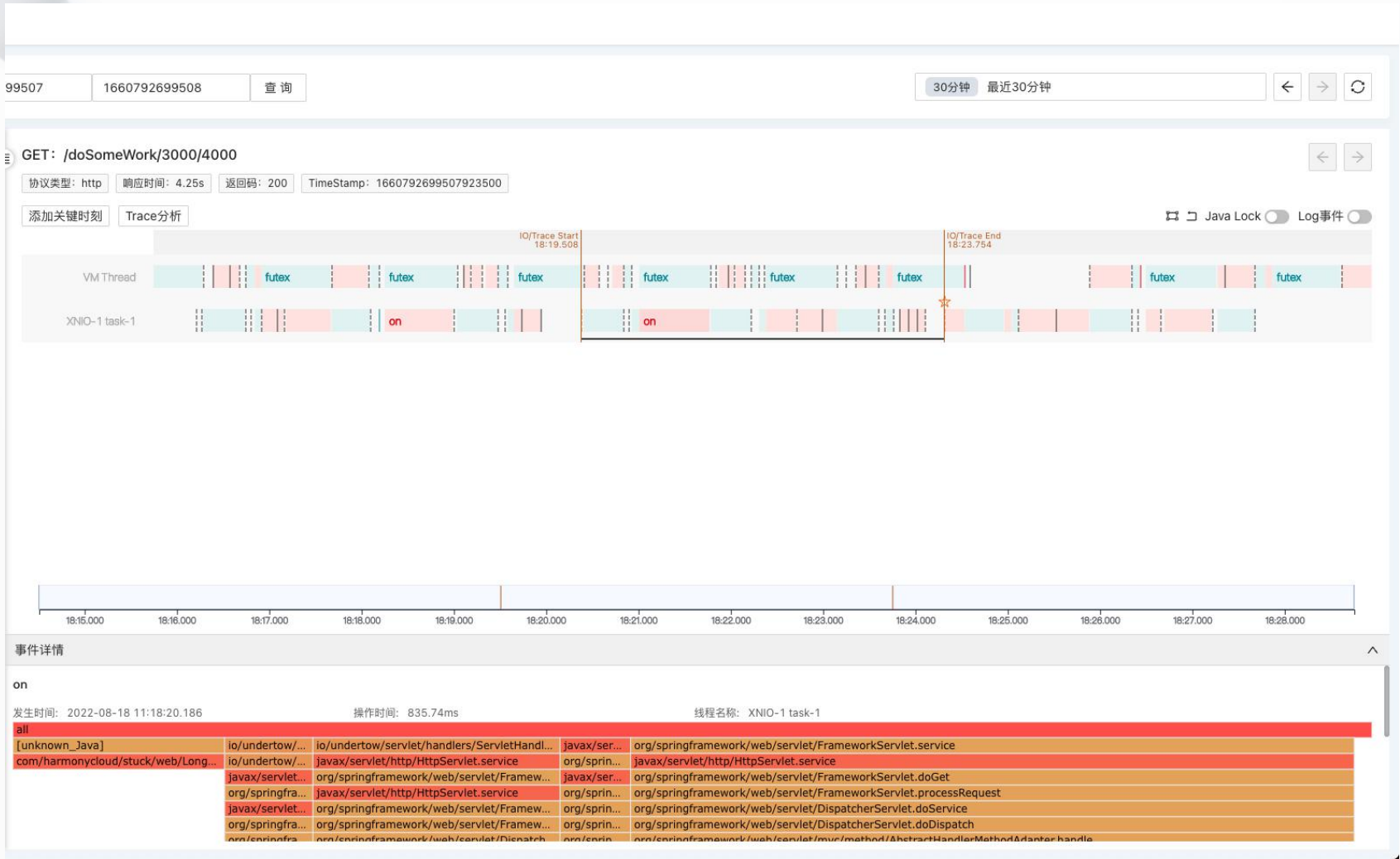
Kindling云可观测

UseCase4: 云原生环境网络TCP窗口配置有问题

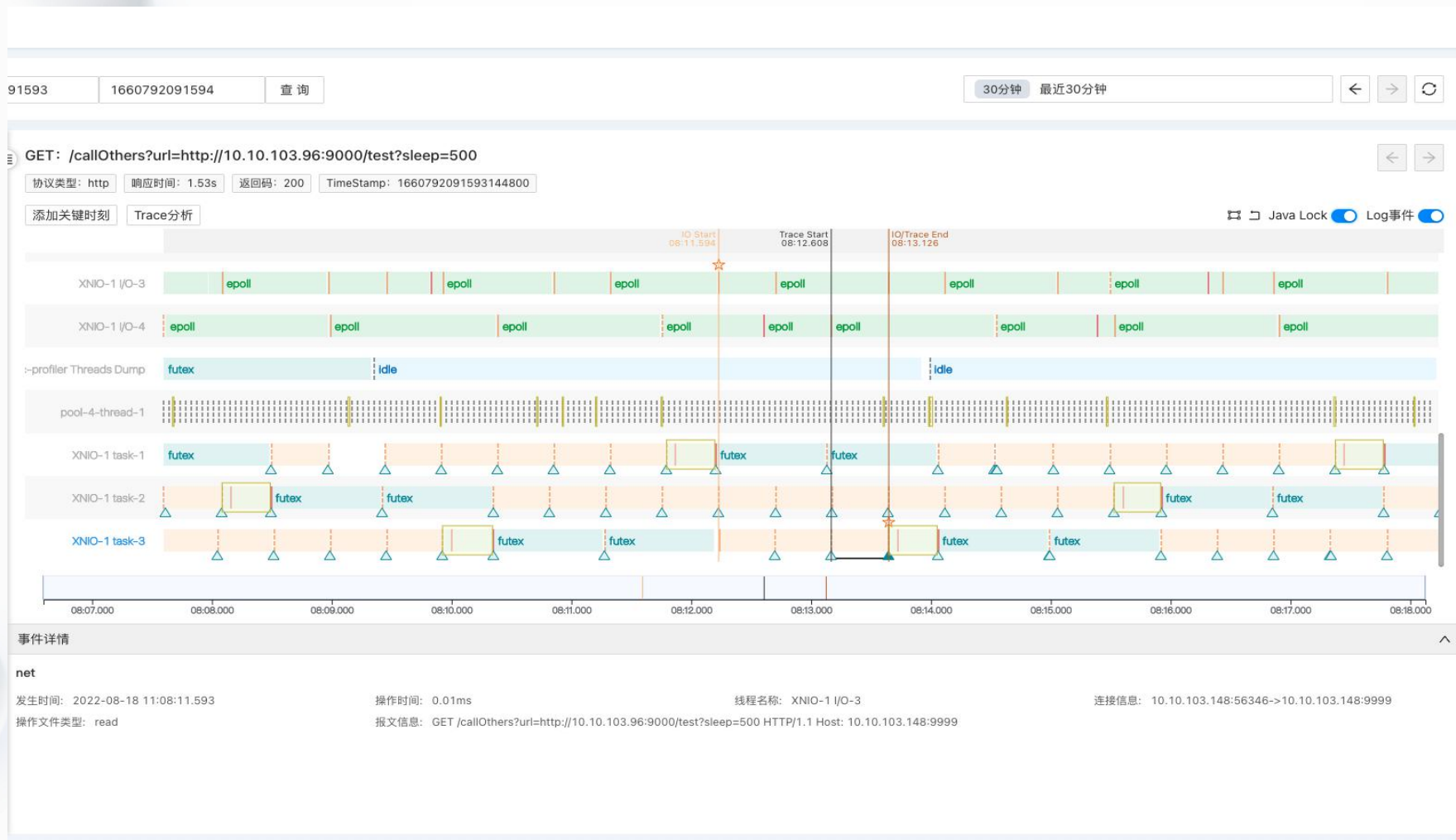
在测试环境, 通过调整TCP 窗口优化性能

接收方buffer大小	85Kb	512Kb	1Mb	2Mb	4Mb
响应时间(ms)	127.1	137.1	129.1	98	77.4

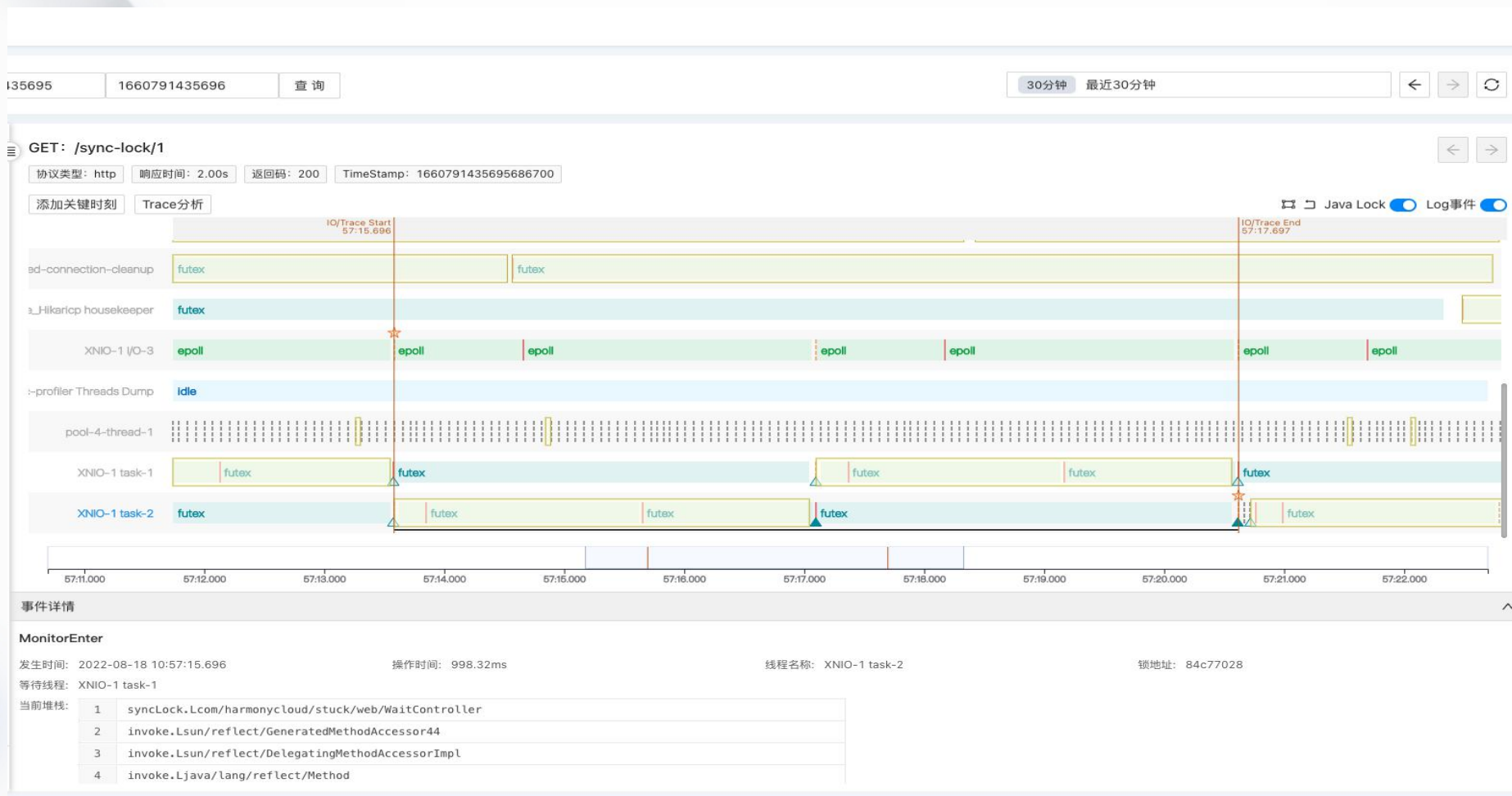
UseCase5: Java程序由于GC的原因被暂停执行



UseCase6: 程序并发过高, 线程池不够用



UseCase7: 程序锁的呈现



THANKS