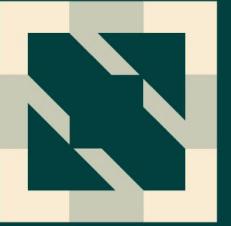


KubeCon



CloudNativeCon

# S OPEN SOURCE SUMMIT

---

China 2023

---



KubeCon



CloudNativeCon



OPEN SOURCE SUMMIT

China 2023

# Trace-Profiling——a New Way About How to Tracking Application Behavior

*CHANG, Cheng*

# About Me

姓名: 范程

Name: CHANG,Cheng

From 2010, RA at Zhejiang University SEL Lab focusing on cloud native and distributed computing fields.

Founded Harmonycloud Technology as a technical co-founder in 2016.

Founder of the Kindling open source project based on eBPF in 2022.

<https://github.com/KindlingProject/kindling>

Founded a new startup focusing on root cause analysis of faults as CEO in 2023.

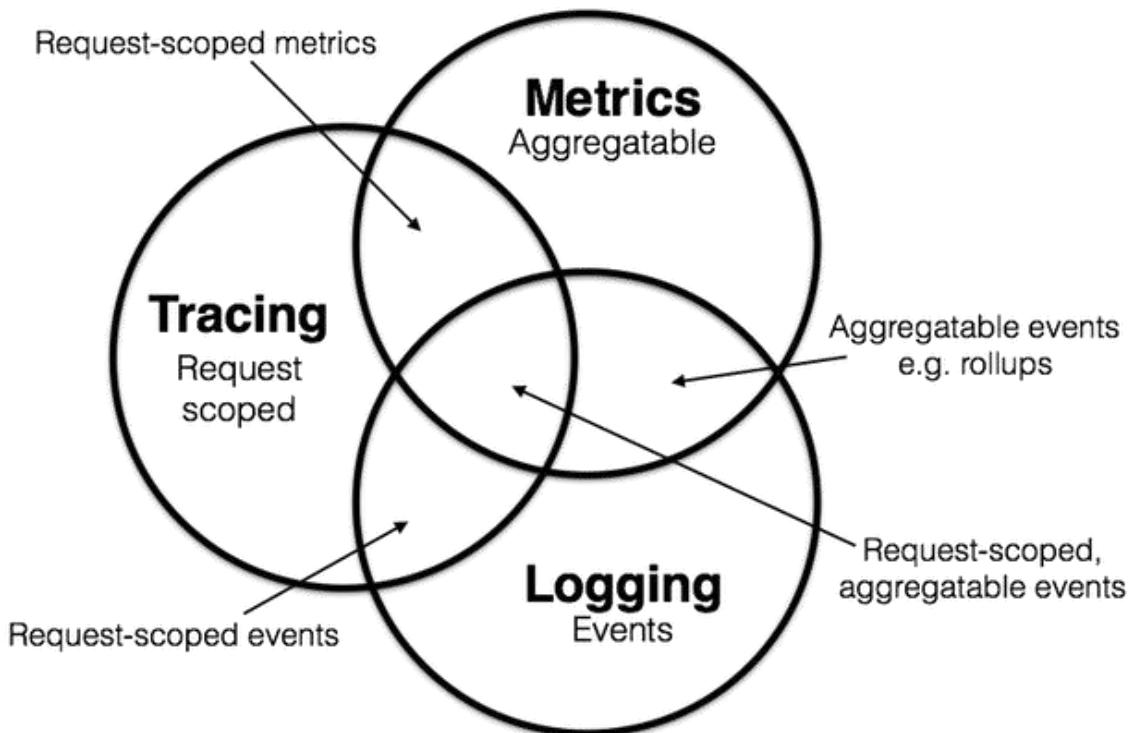


范程.

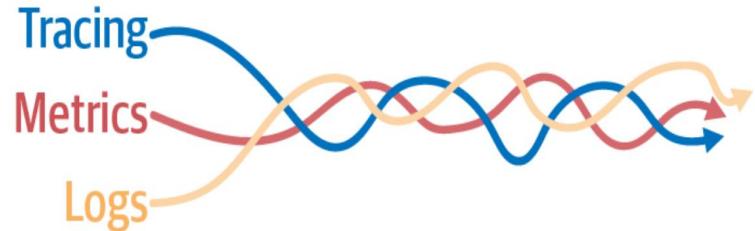


扫一扫上面的二维码图案，加我为朋友。

# Observability current status



OpenTelemetry project trying to integrate the different data from backend



# why find app abnormal root-cause so hard

If we know what happened  
root-cause can be found easily.

observability tools helps understanding  
the internal black box of the system.

But still many blind spots

Unknow-Unknow cause  
the difficulties

Known

Unknown

Knowns

Unknowns



Things we are aware of  
and understand



Things we are aware of  
but **don't** understand



Things we are **not** aware of  
but understand



Things we are **not** aware of  
and **don't** understand

# How to open the blind spots in old way

heavliy rely on Expert Experience

how about so many issue never meet before?

what happed if the expert is not on site?



cycle to add customize metric

## heuristic troubleshooting for blind spot

The ability to perform interactive exploration and analysis of multiple telemetry types (such as traces, metrics, logs) to detect “unknown unknowns”

Lucky

Expert Experience

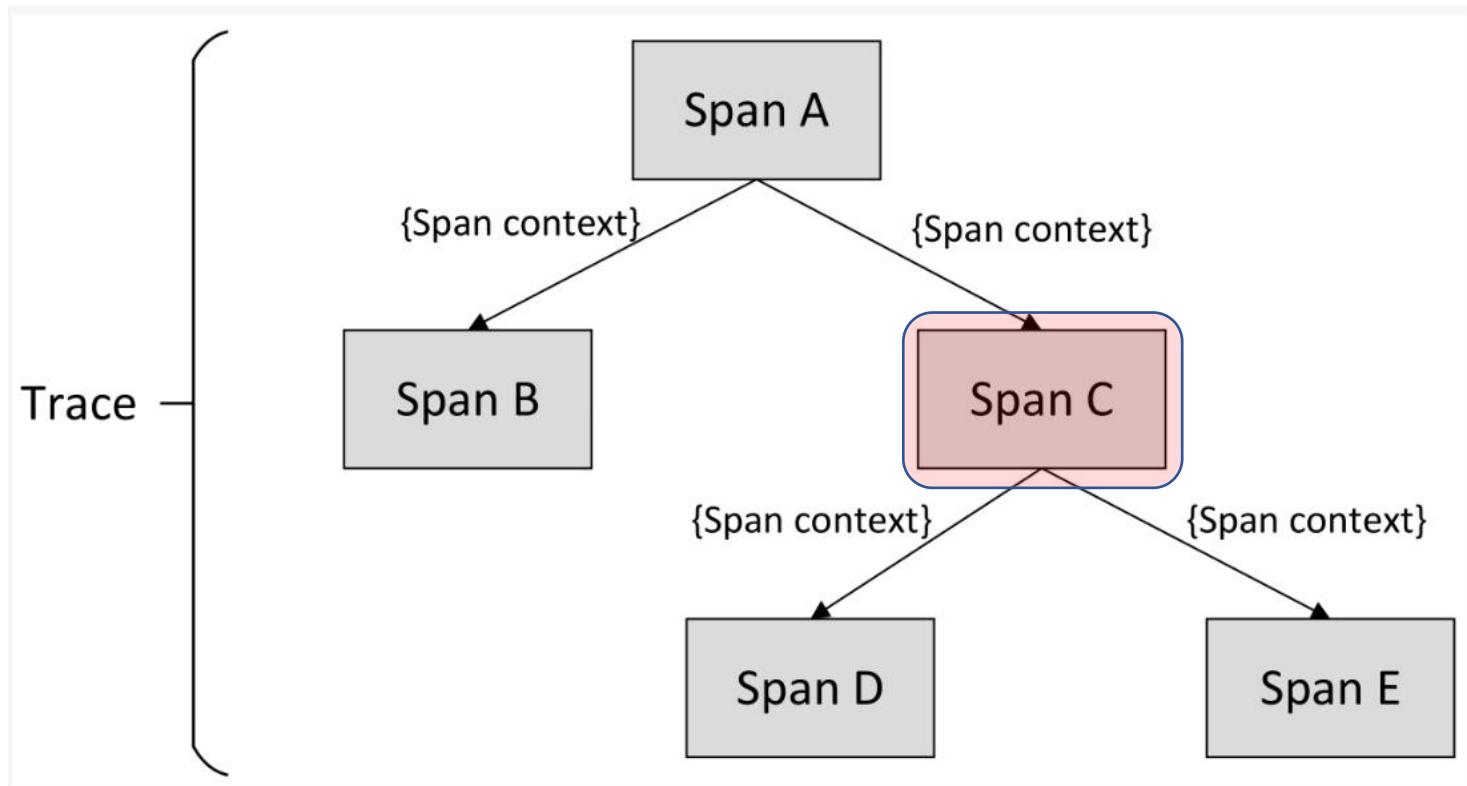


HIGH MTTR

# where is the blind spots come from

Response time (specific URL) higher than usual

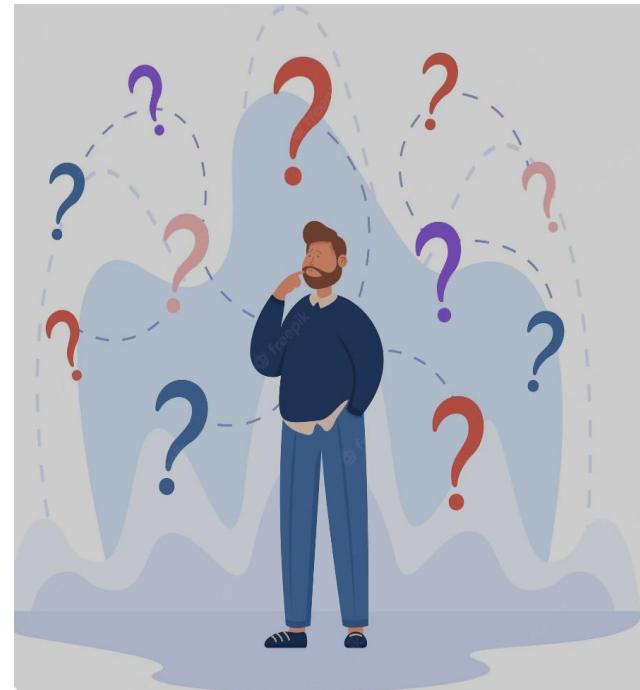
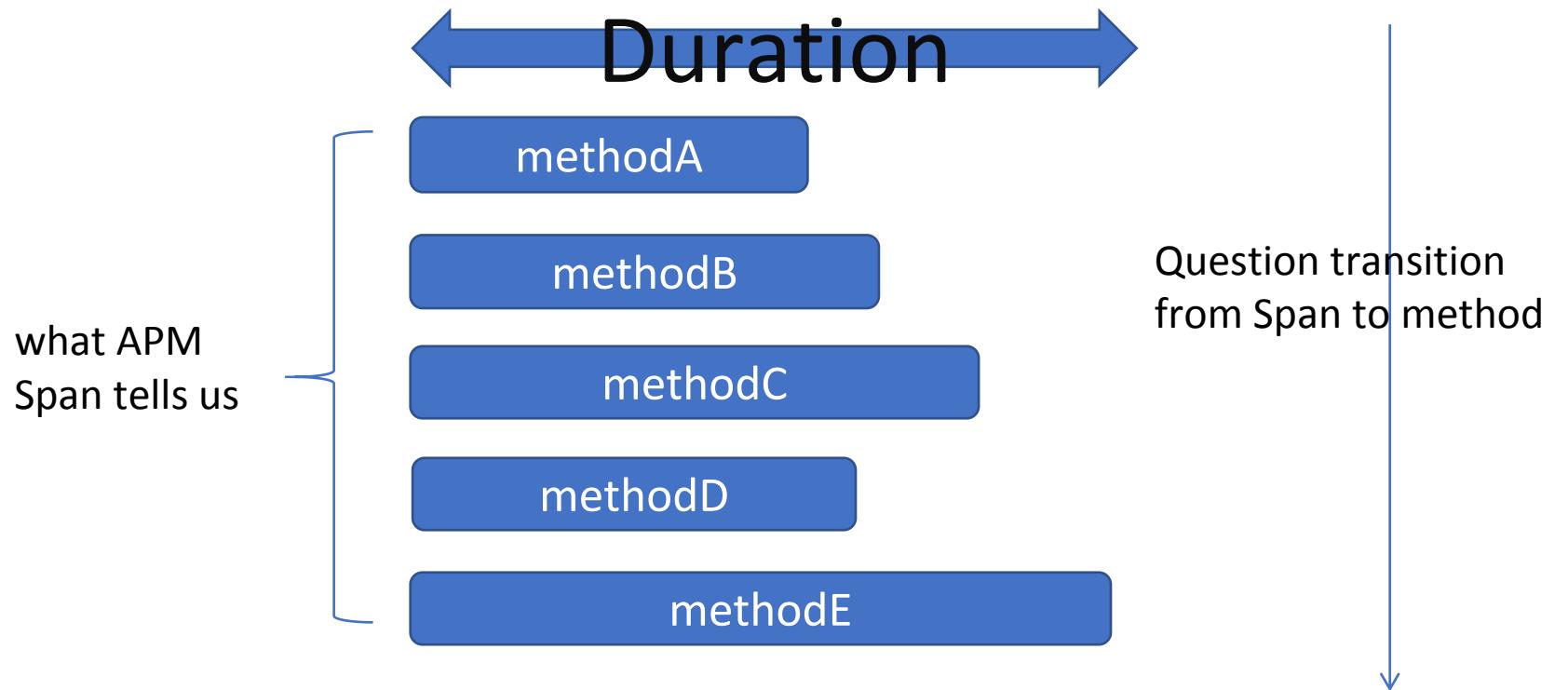
one slower trace



with tracing which span cause the problem

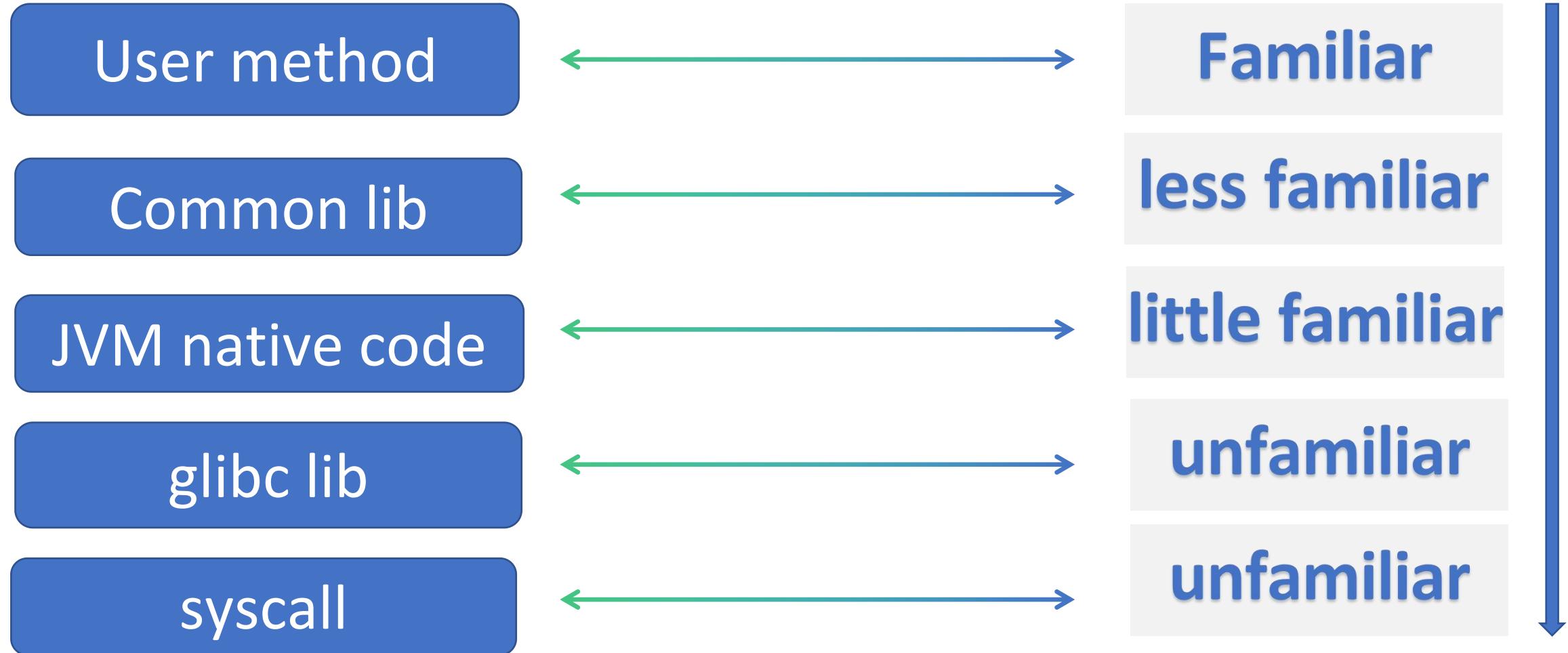
# blind spots come from error span

## why span is slower than usual?



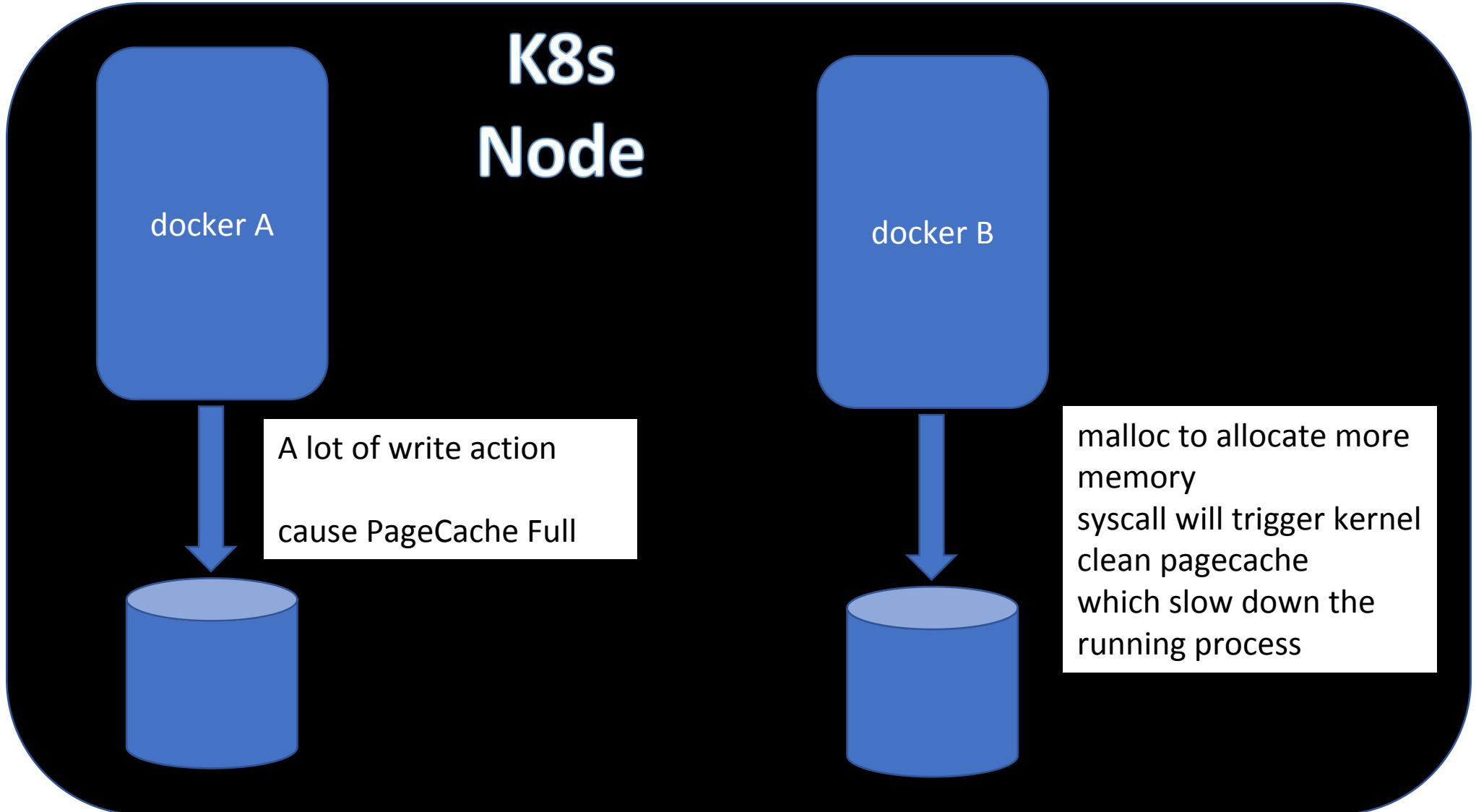
## why methodE is slower than usual?

# why method has the blind spots



# example for syscall behavior cause problem

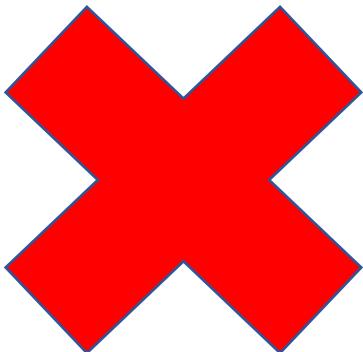
cloud native share the same resource



# blind spots come from unfamiliar

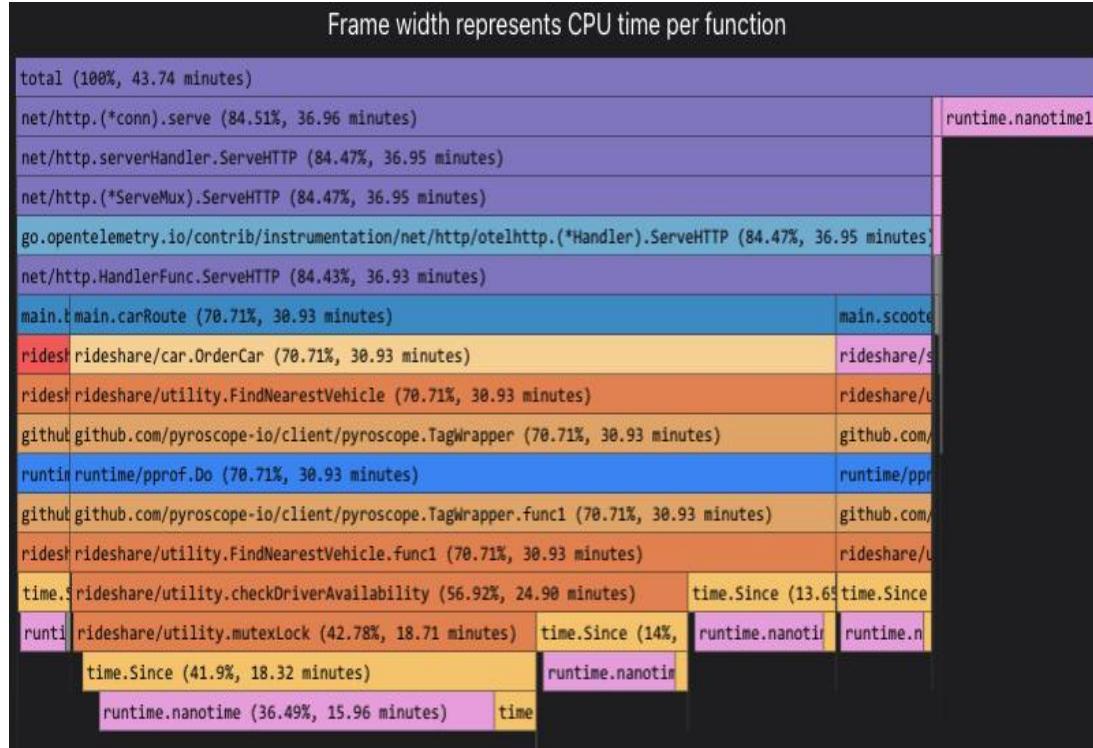
Layer  
design  
helps the  
dev be  
more  
productive

## contradictions

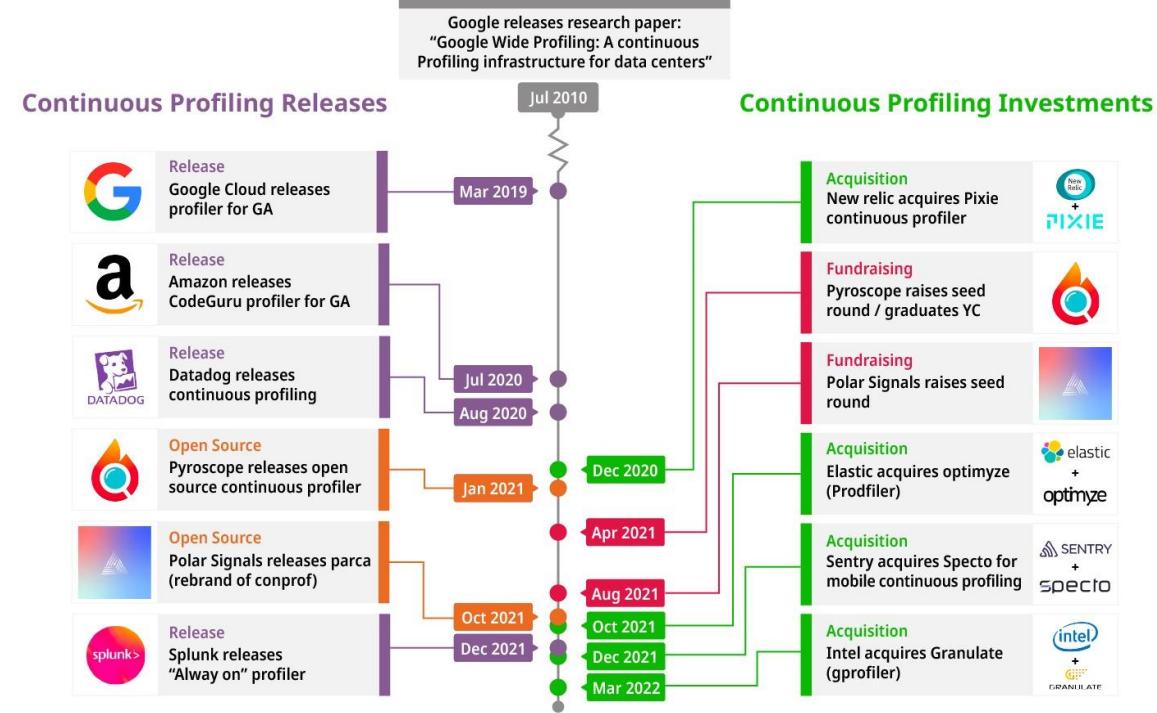


Layer  
design  
encapsulates  
details

# how to know what happened for the unfamiliar method

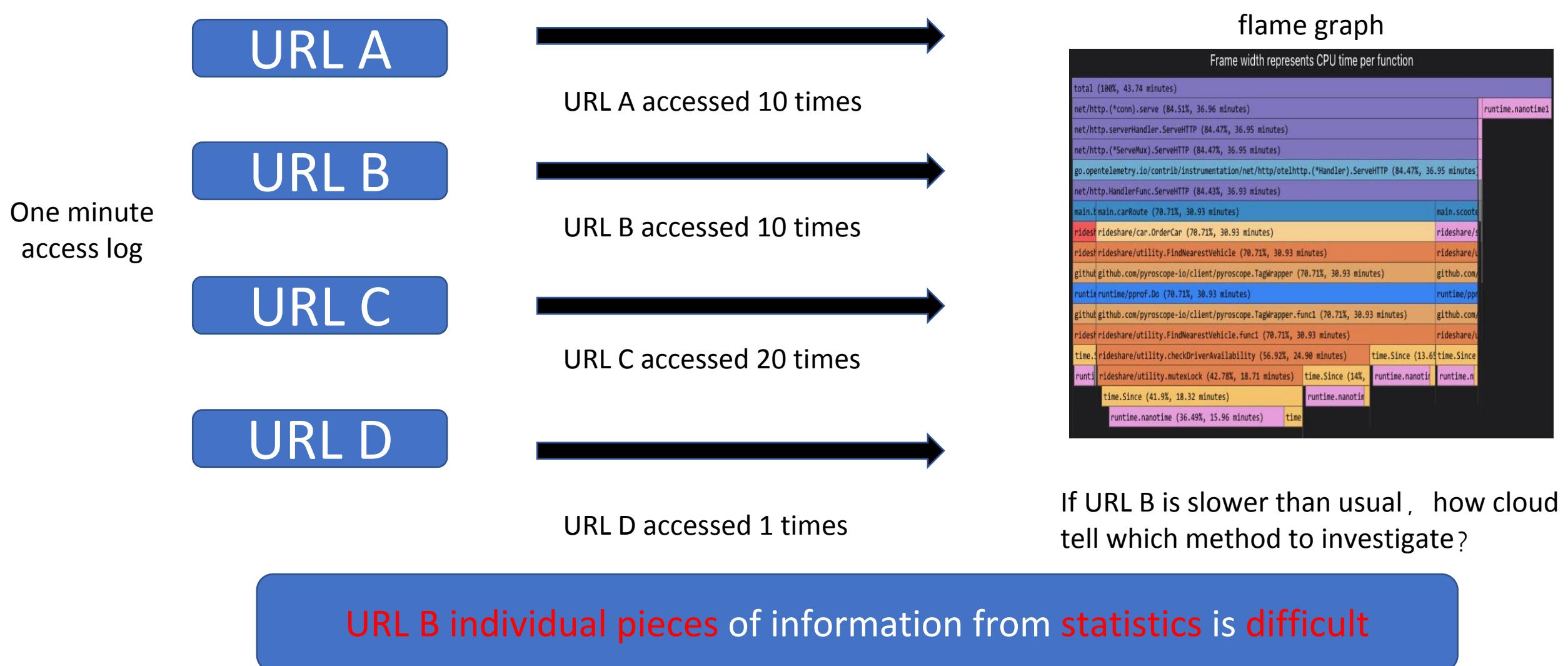


onCpu flame-graph help

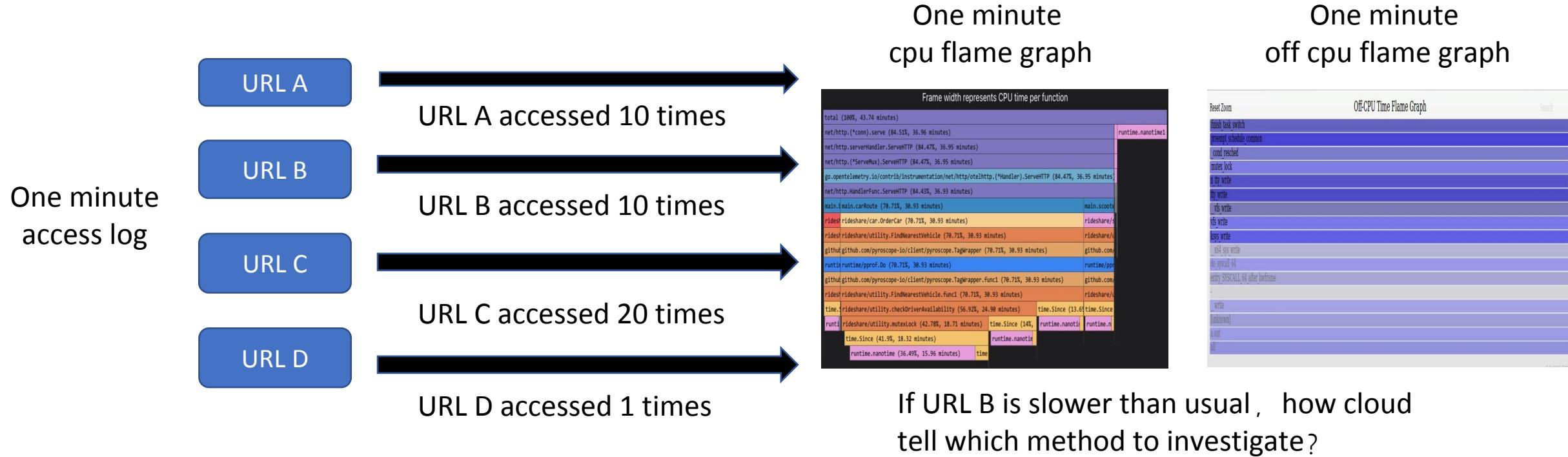


Continuous profiling

# One of Continuous profiling limitation



# Another of Continuous profiling limitation



most Continuous profiling **do not have off cpu flame graph**  
Even, with onCPU flame graph and off cpu flame graph, how to  
connect onCpu information with offCpu information

# how to know what happened for the unfamiliar method——online debug

# Arthas

Java 应用诊断利器



User method

Common lib

JVM native code

glibc lib

syscall

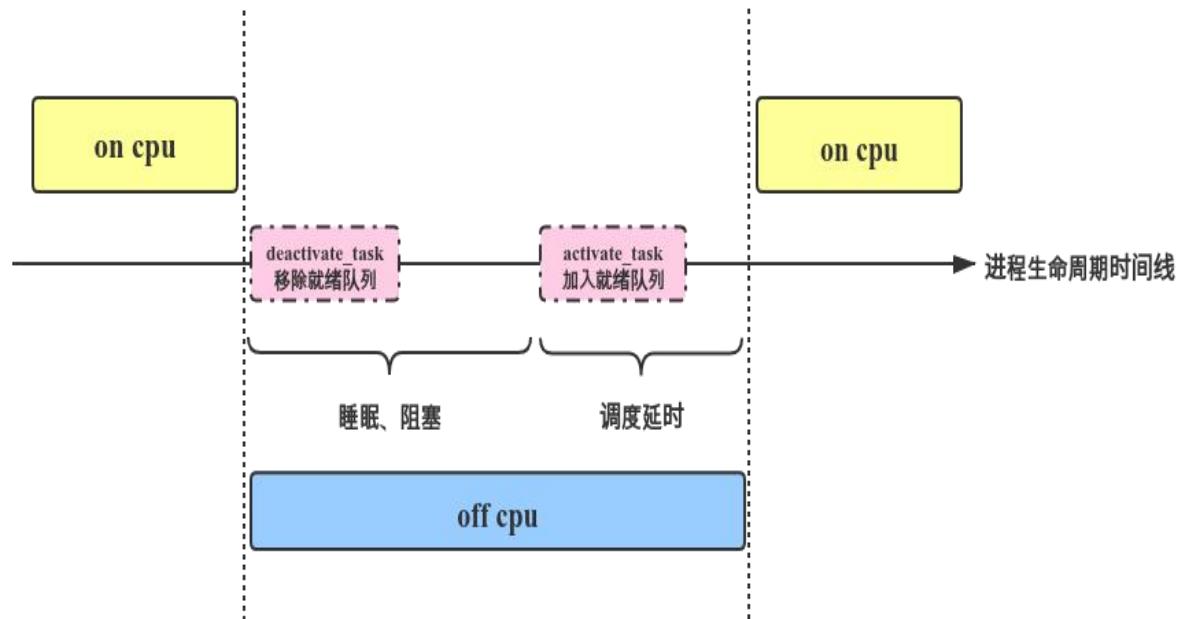
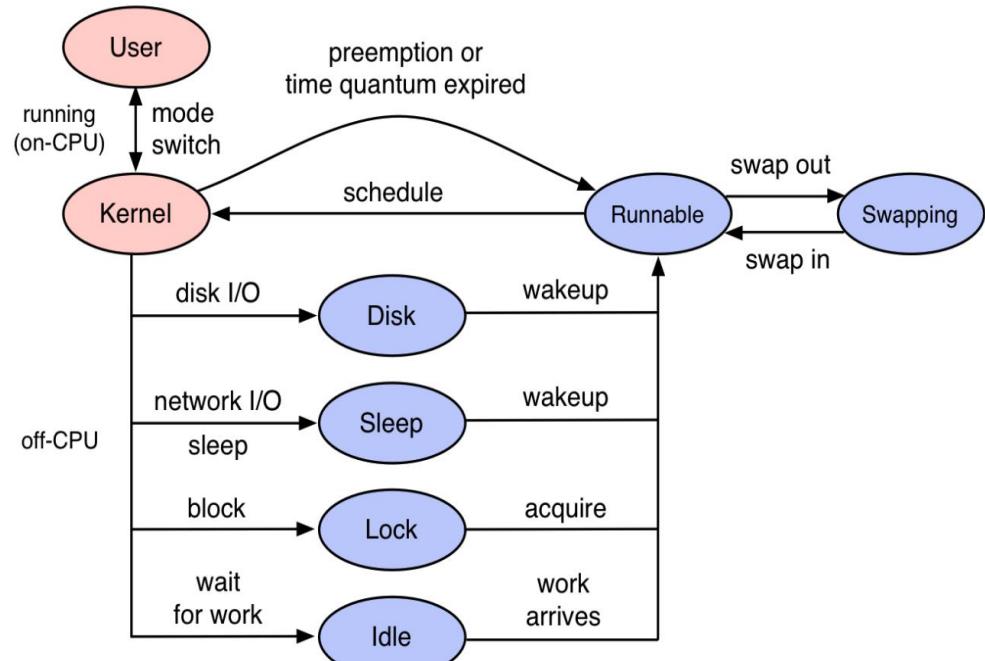
low level code can not be debug online easily



# Trace-profiling open the blind spots

Align onCpu event with offCpu event from tread level

integration with distributed tracing



# Kindling open source project

KindlingProject / kindling

Type  to search

Code Issues 20 Pull requests 5 Discussions Actions Projects Wiki Security 1 Insights Settings

kindling Public Edit Pins Unwatch 17 Fork 76 Starred 454

**Kindling**

eBPF-based Cloud Native Monitoring tool

Understanding the app behavior from kernel to app code

Get Started Watch the Demo

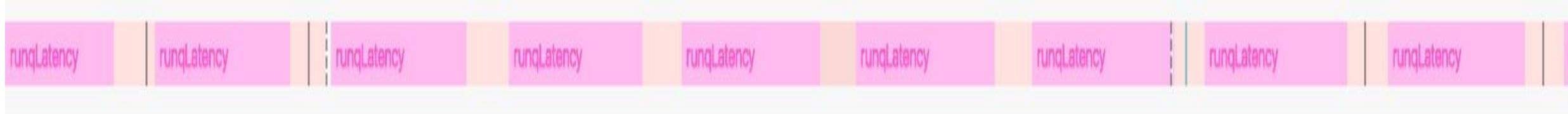
The diagram illustrates the flow of a user request through the system. A blue circle labeled "ONE USER REQUEST" is shown on the left, connected by a teal arrow to a circular diagram on the right. The circular diagram is divided into three concentric rings. The innermost ring is labeled "KERNEL". The middle ring is labeled "APP CODE" and contains several points representing latency measurements: "Accept", "Read request (bytes latency)", "Read file filename, bytes latency", "Outbanding request latency", "Send response bytes latency", "Lock latency", and "log the data to logfile bytes latency". The outermost ring is also labeled "APP CODE".

# With trace-profiling

- How all threads were executed is recorded and can be displayed.
- The exact thread which executed the trace span is highlighted.
- The code execution flame graph is correlated to the time series where CPU is busy.
- The network-related metrics are correlated to the time series where the network syscalls are executing.
- The file-related metrics are correlated to the time series where the file syscalls are executing.

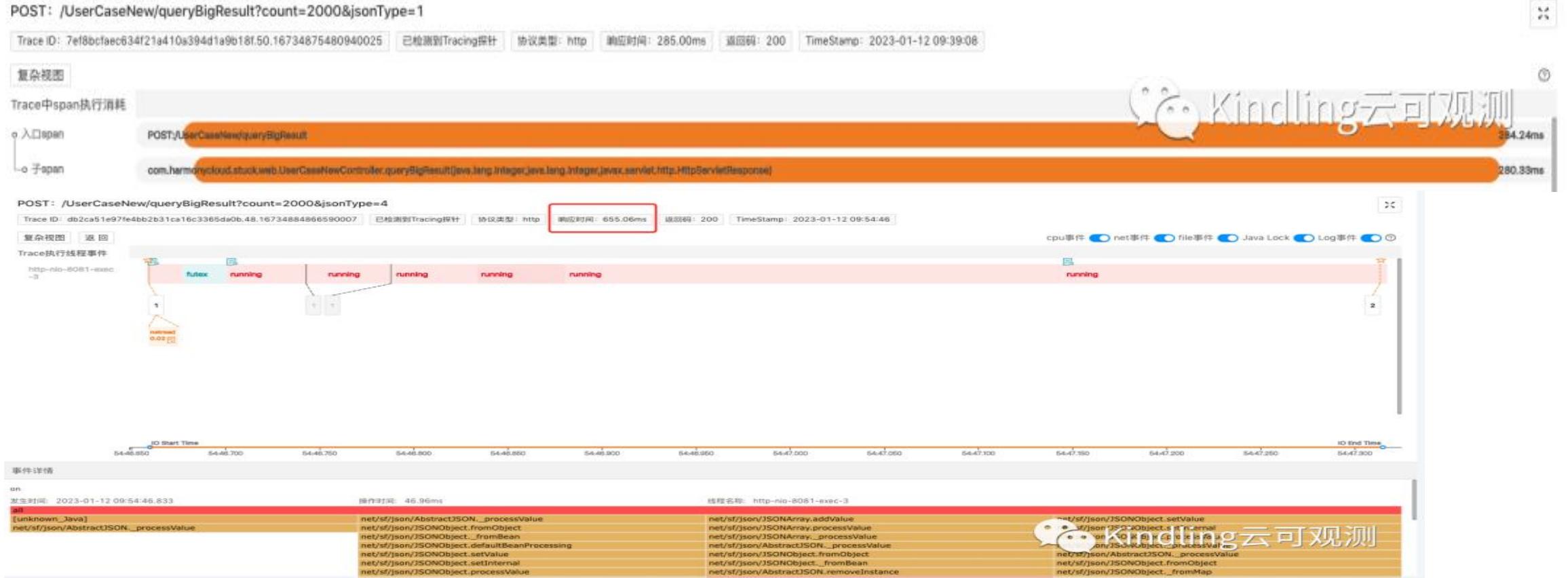
# UseCase 1—— CPU restricted

- no appropriate configuration for CPU limit and request
- some pod use high CPU resource, no available CPU resource



# UseCase 2—— High CPU usage

- User's method cost high cpu



# UseCase 3—— storage issue

- User method need access storage issue

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

Trace ID: 726da9e5abe47a0acce166040202291.46.16751443101150001 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

o 入口span

GET/UserCaseNew/fileIO

com.harmonycloud.stuck.web.UserCaseNewController.getFileOrwsLangBooker.java.lang.String)



Kindling云可观测

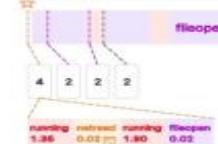
GET: /UserCaseNew/fileIO?filePath=%2Fopt%2FgrhTest%2Fdemo2.jar&useBuffer=true

Trace ID: 726da9e5abe47a0acce166040202291.46.16751443101150001 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 | 返回

Trace执行线程事件

http-nio-9999-exec-4



事件详情

file  
发生时间: 2023-01-31 13:52:02,213  
操作文件: /opt/grhTest/demo2.jar  
操作文件类型: read

操作时间: 338.11ms

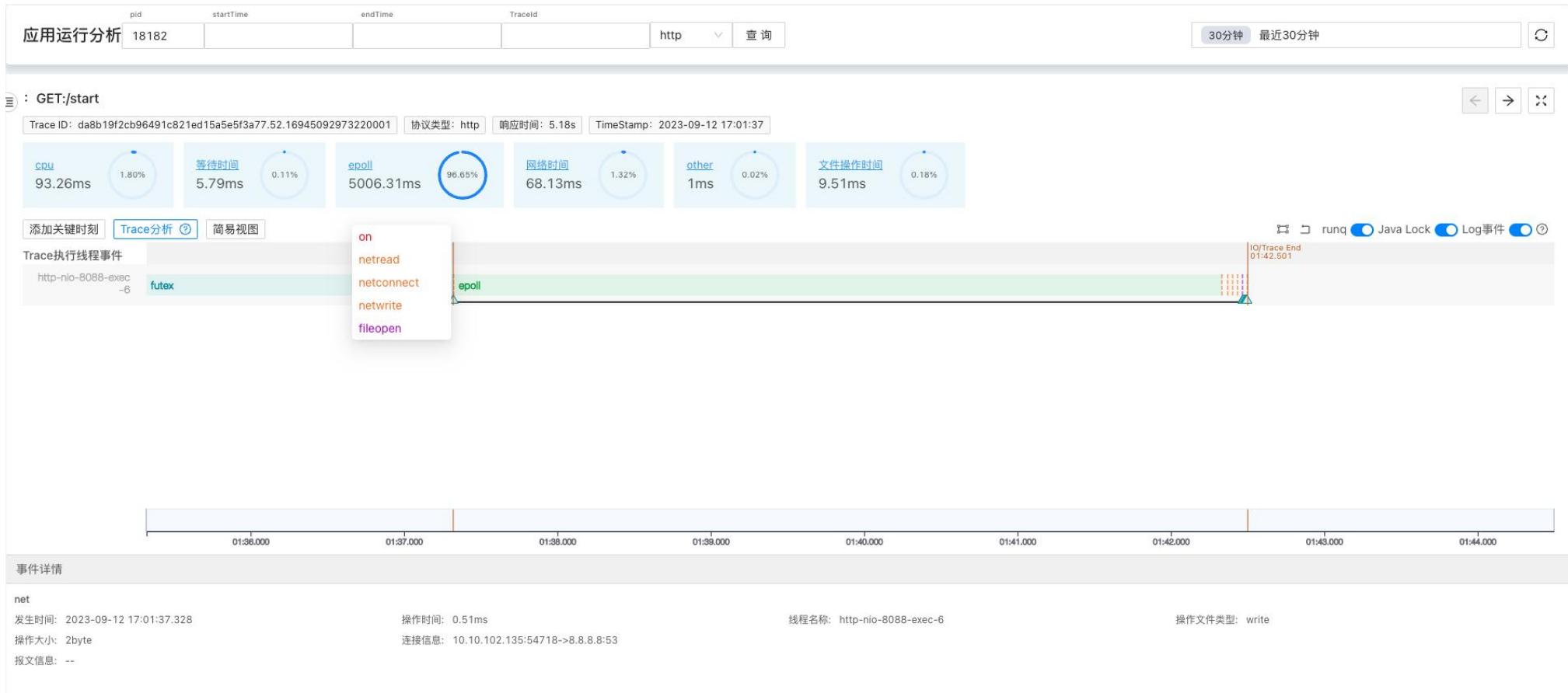
线程名称: http-nio-9999-exec-8

操作文件: /opt/grhTest/demo2.jar

Kindling云可观测

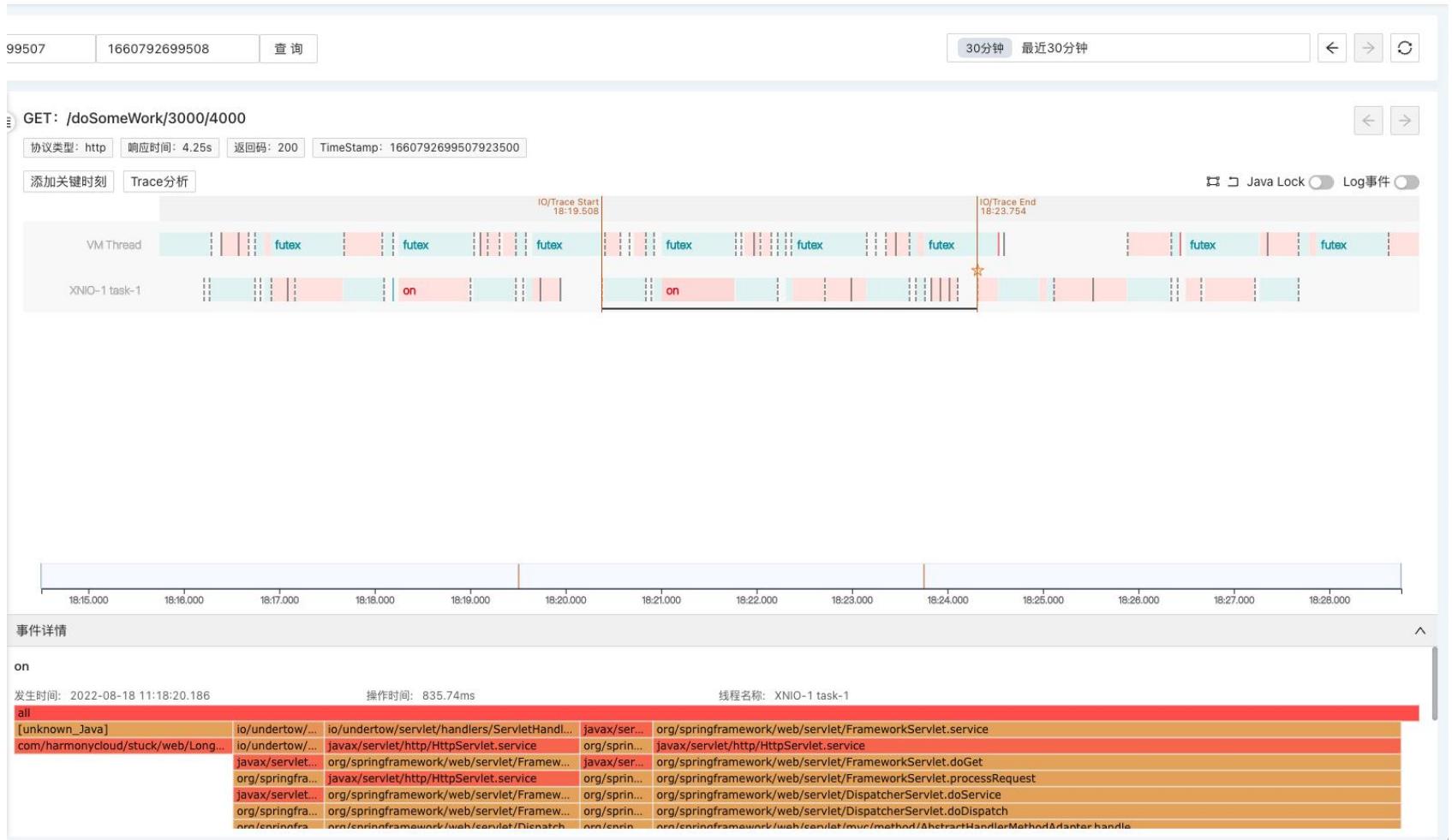
# UseCase 4—— DNS issue

- The epoll syscall is waiting for x.x.x.x->8.8.8.8:53 for long time
- 8.8.8.8 is the DNS server
- 53 is the DNS server port



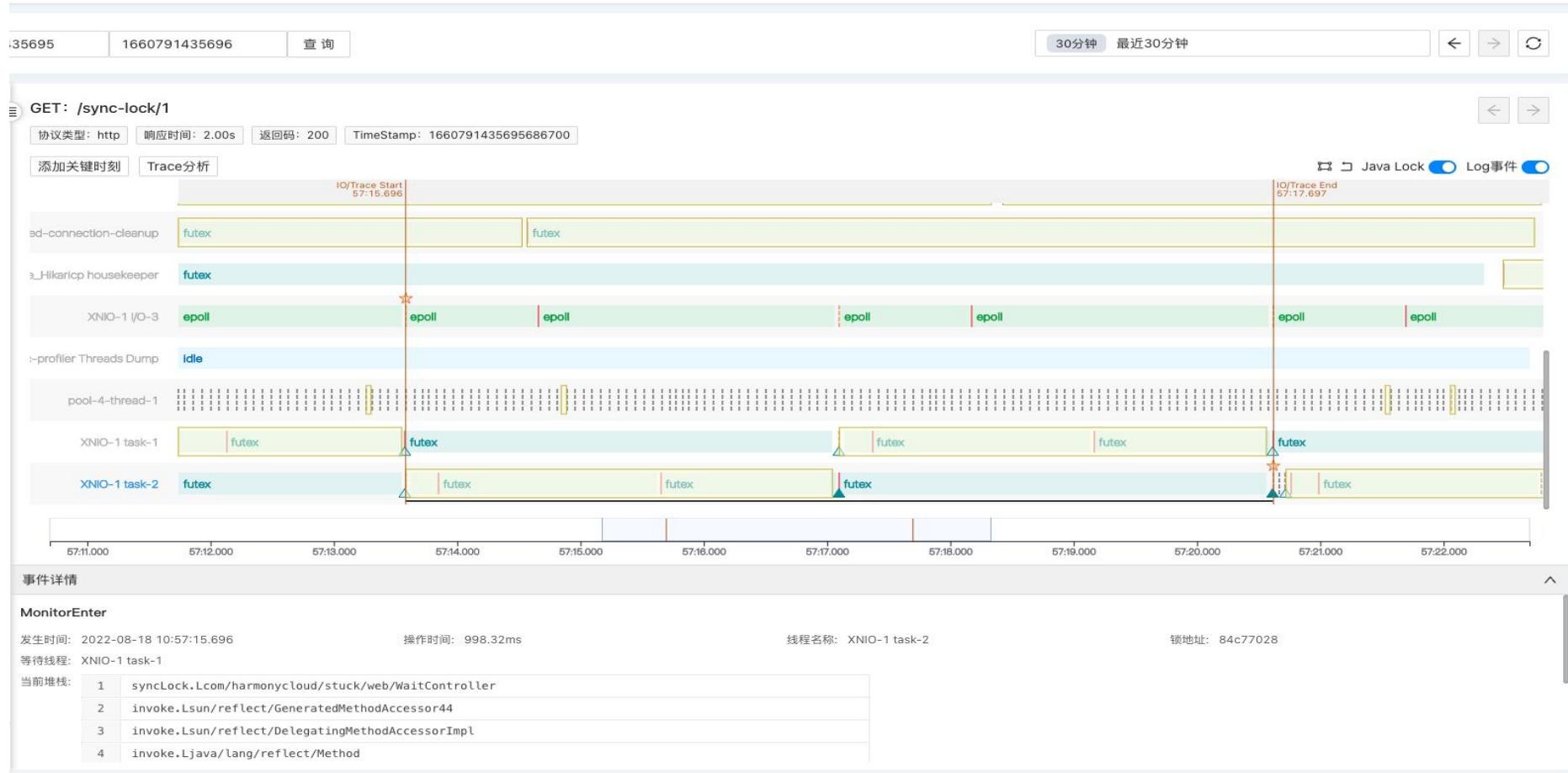
# UseCase 5—— Java GC issue

- The user method has been stoped by GC



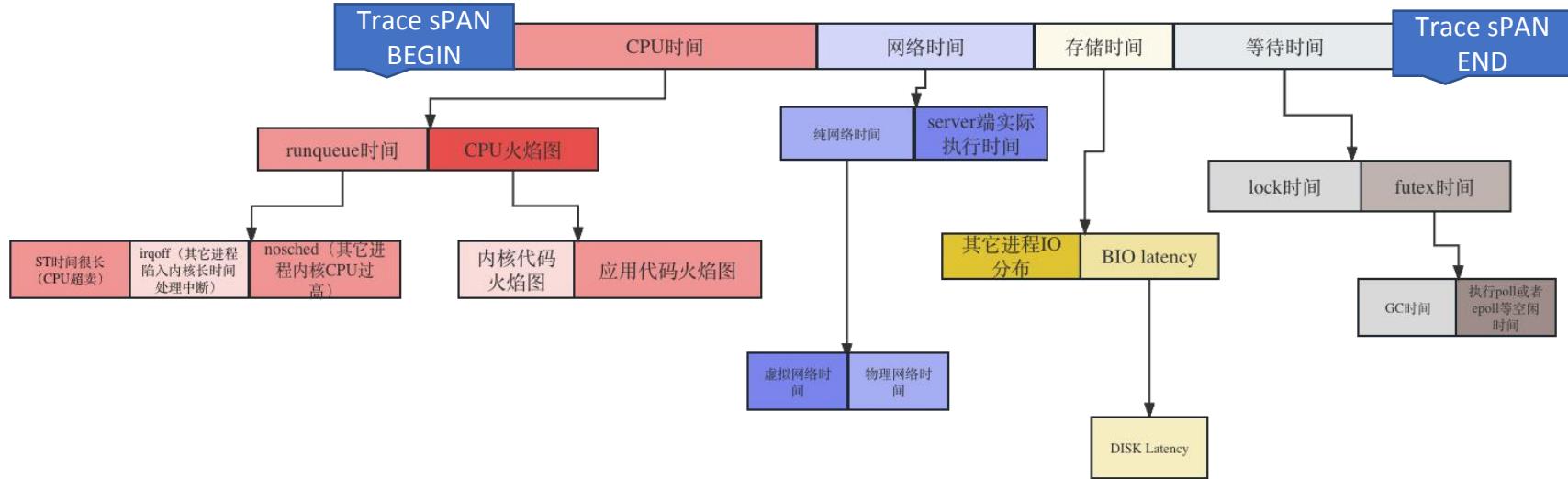
# UseCase 6—— Lock issue

- The user method has been stoped by Lock



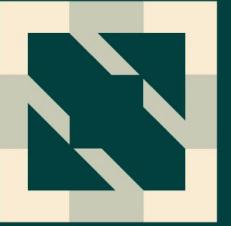
# Polaris trouble-shooting metrics

- OpenAnolis and Kindling released **Polaris trouble-shooting metrics**
- By Analysis of trace profiling data get **Polaris trouble-shooting metrics**, which guide how to troubleshooting





KubeCon



CloudNativeCon

# S OPEN SOURCE SUMMIT

---

China 2023

---