
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

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

CHANG, Cheng

About Me

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2023

姓名：苌程
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.

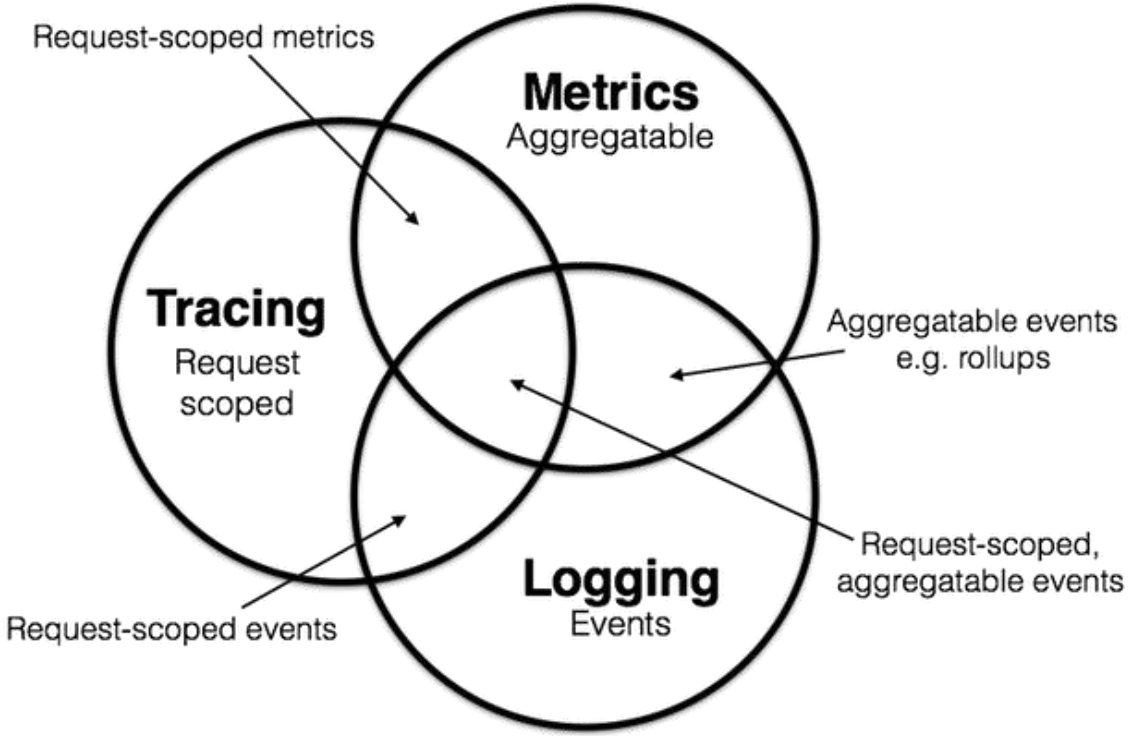
Founder of the Kindling open source project based on eBPF in 2022.
https://github.com/KindlingProject/kindling

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

苌程.

扫一扫上面的二维码图案，加我为朋友。

Observability current status



Opentelemetry project tring to integrate the different data from backend

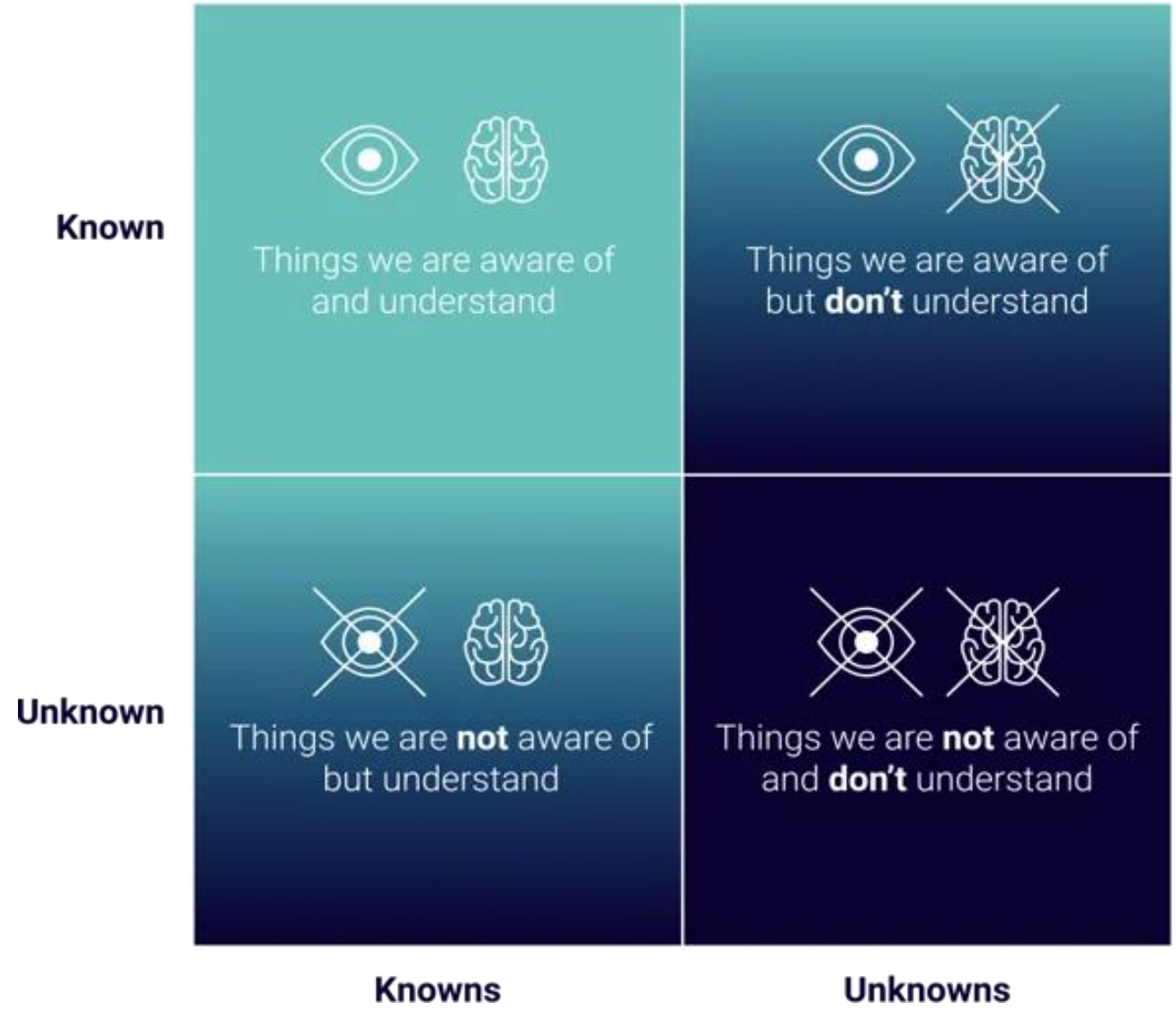
why find app abnormal root-cause so hard

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 differculties

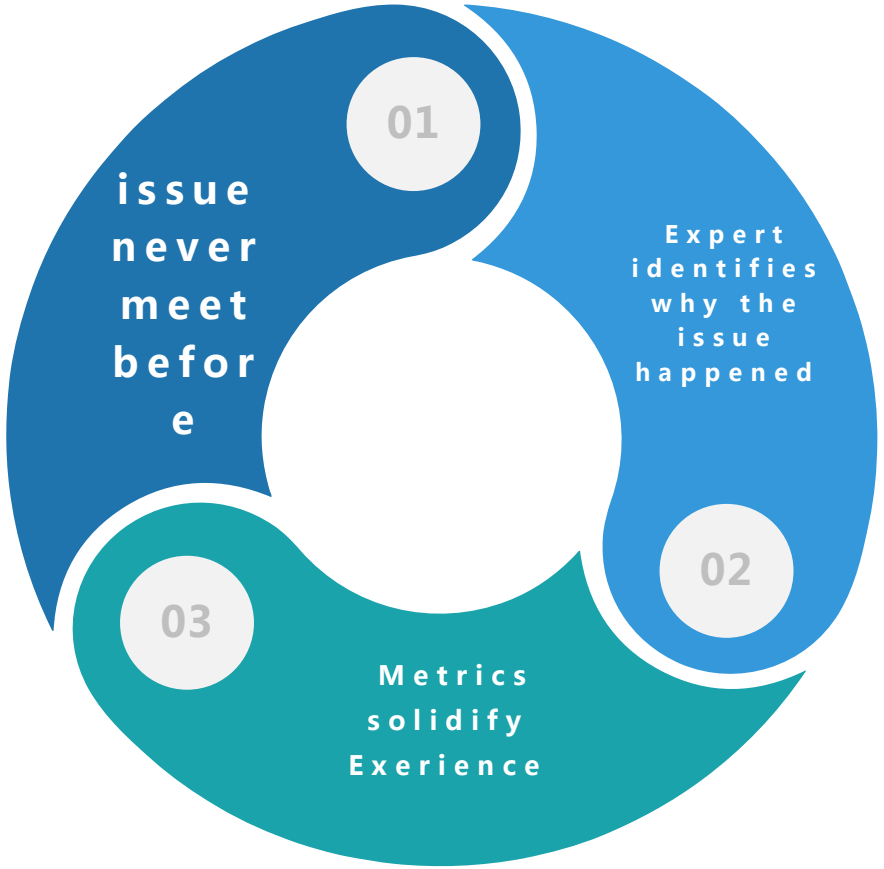


How to open the blind spots in old way

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

Gartner perspective for uncovering blind spots

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

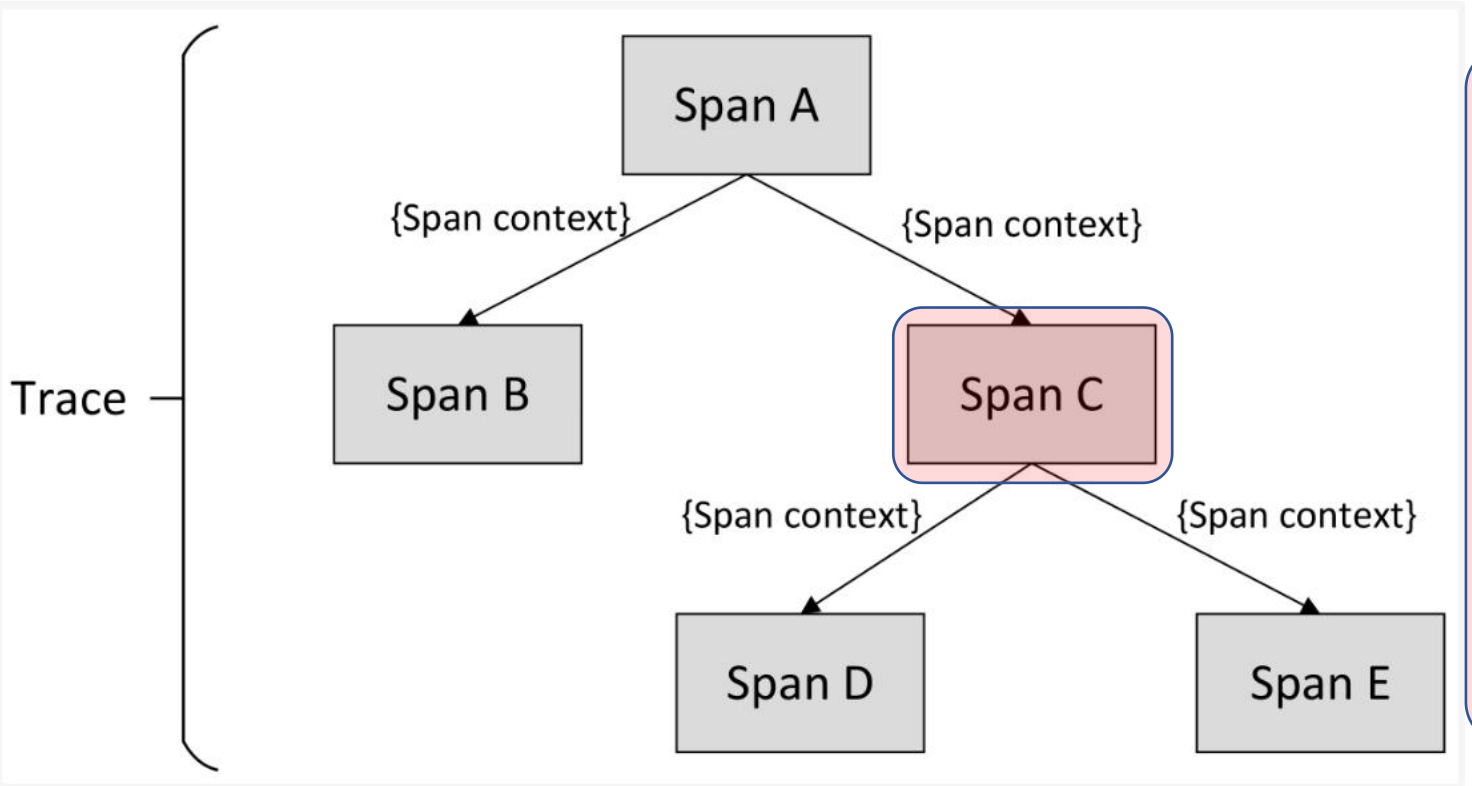
Expert Experience

HIGH MTTR

where is the blind spots come from

Response time（specific URL）higher than usual

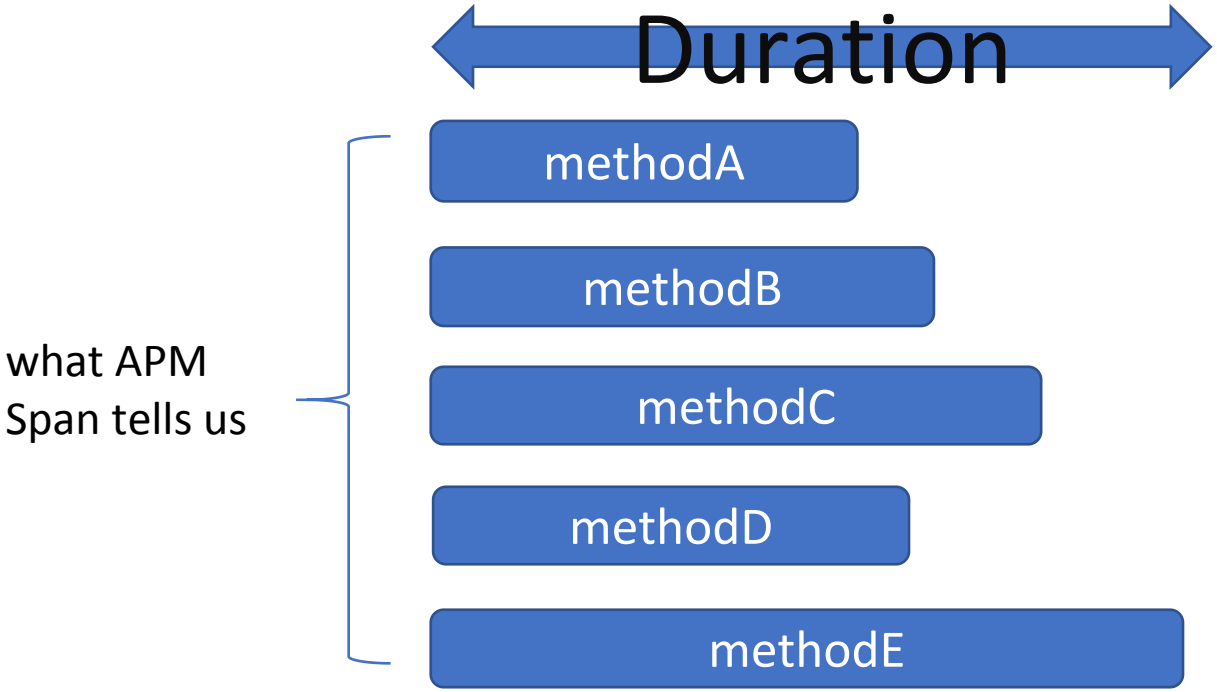
one slower trace



with tracing which span cause the problem

blind spots come from error span

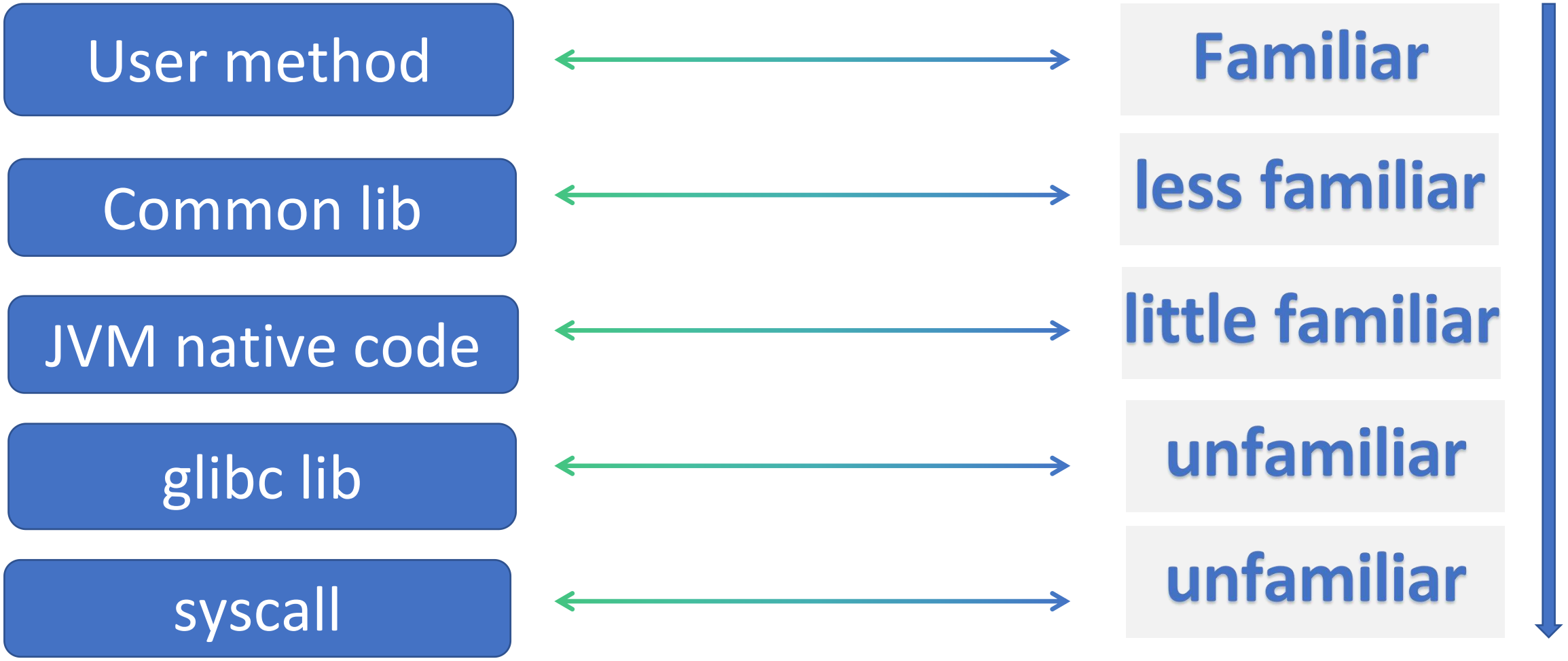
why span is slower than usual?



Question transition
from Span to method

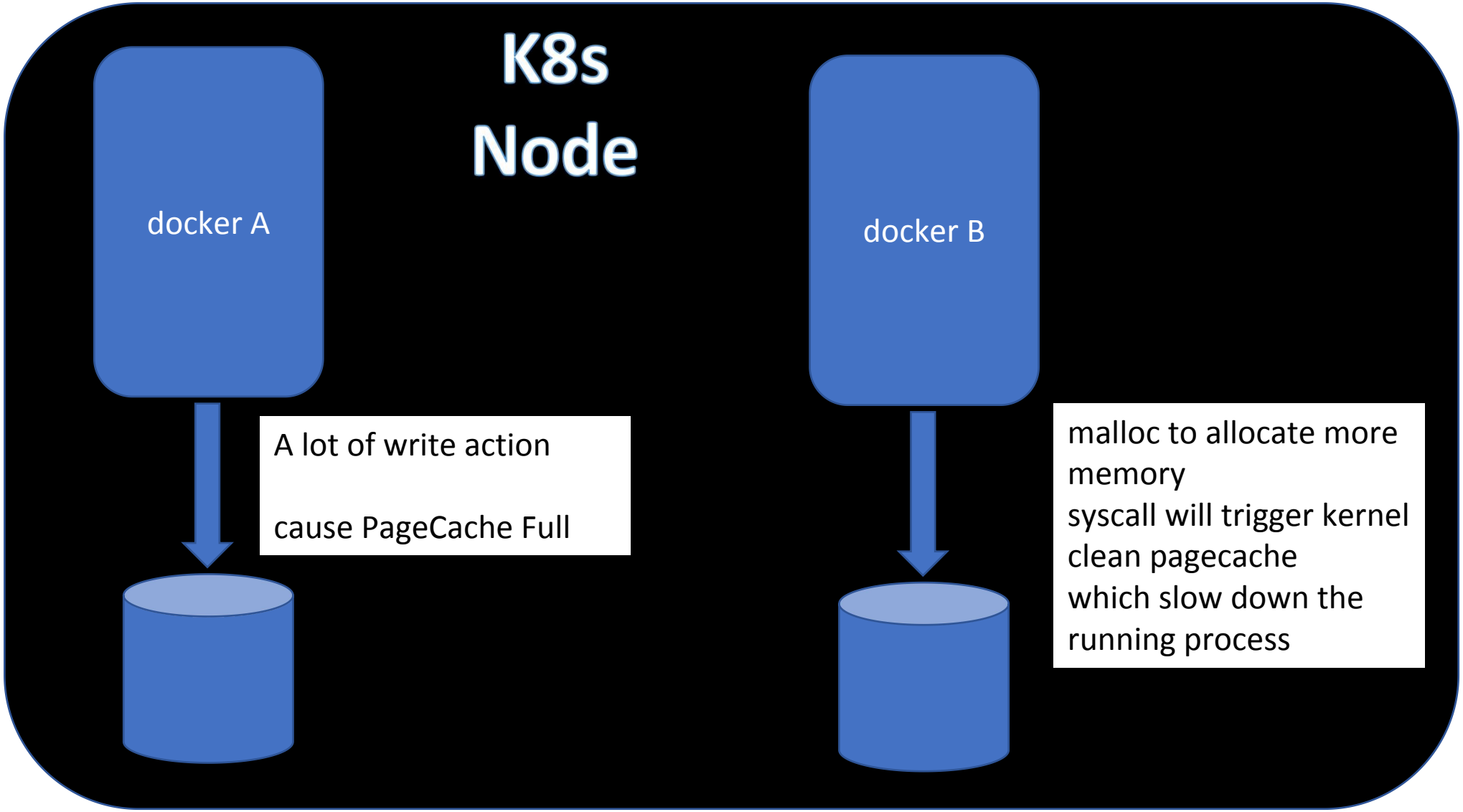
why methodE is slower than usual?

why method has the blind spots



example for syscall behavior cause probelm

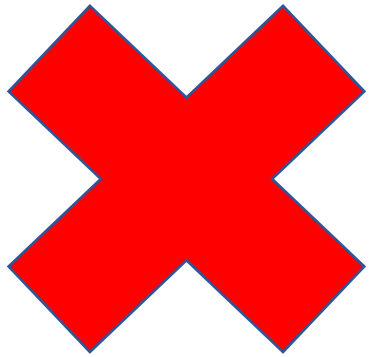
cloud native share the same resource



blind spots come from unfamiliar

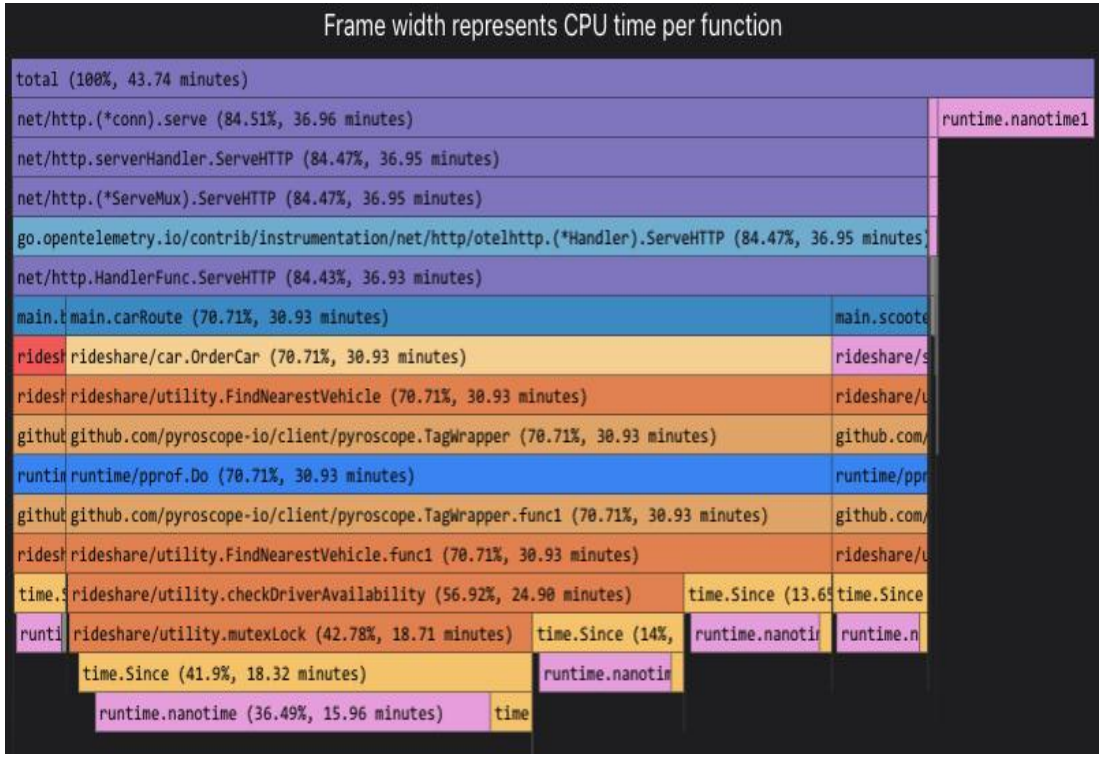
Layer design helps the dev be more productive

contradictions

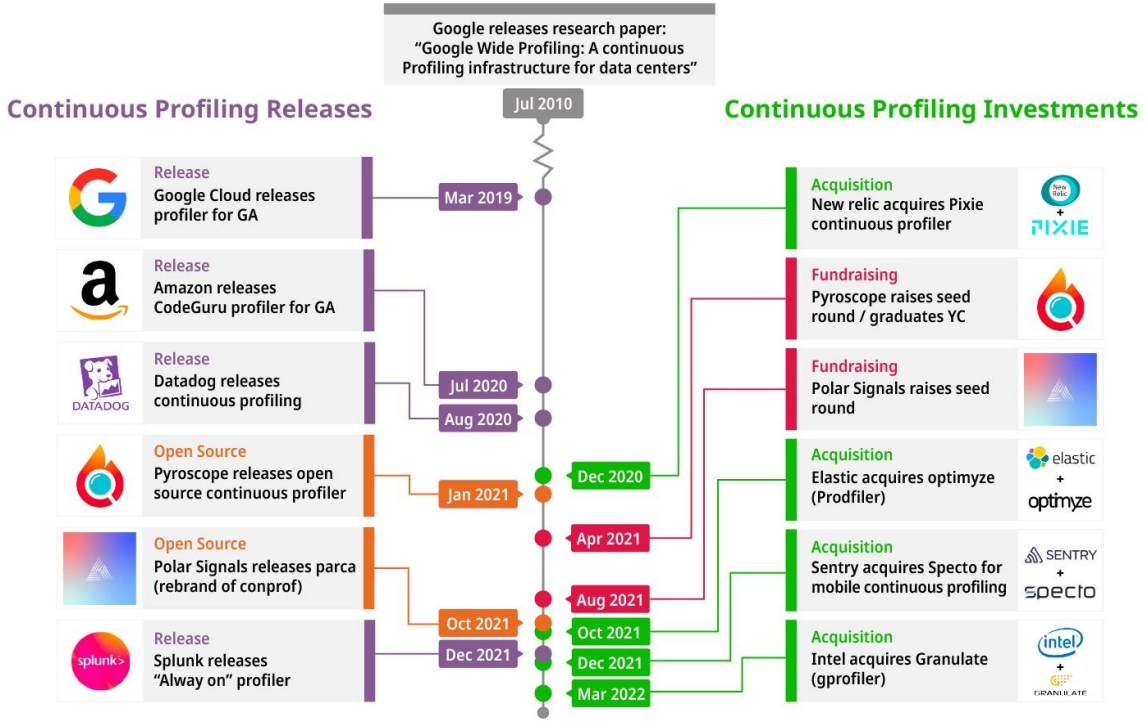


Layer design encapsulates details

how to know what happened for the unfamiliar method



onCpu flame-graph help



Continuous profiling

One of Continuous profiling limitation

One minute access log

URL A

URL A accessed 10 times

URL B

URL B accessed 10 times

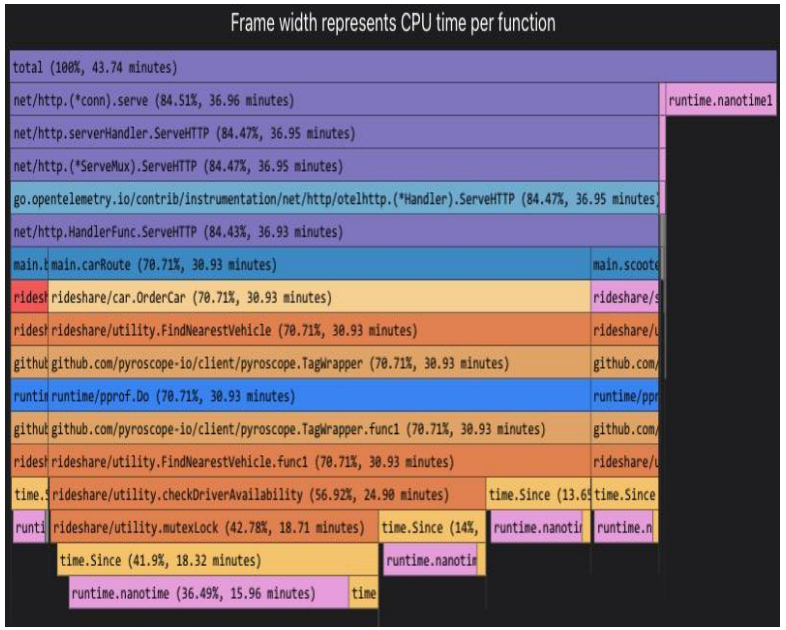
URL C

URL C accessed 20 times

URL D

URL D accessed 1 times

One minute flame graph



If URL B is slower than usual， how cloud tell which method to investigate？

URL B individual pieces of information from statistics is difficult

Another of Continuous profiling limitation

One minute access log

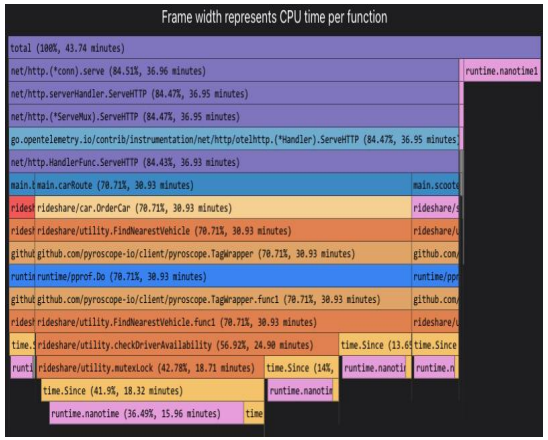
URL A → URL A accessed 10 times

URL B → URL B accessed 10 times

URL C → URL C accessed 20 times

URL D → URL D accessed 1 times

One minute cpu flame graph



One minute off cpu flame graph

If URL B is slower than usual， how cloud tell which method to investigate？

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

Lightrun

User method

Common lib

JVM native code

glibc lib

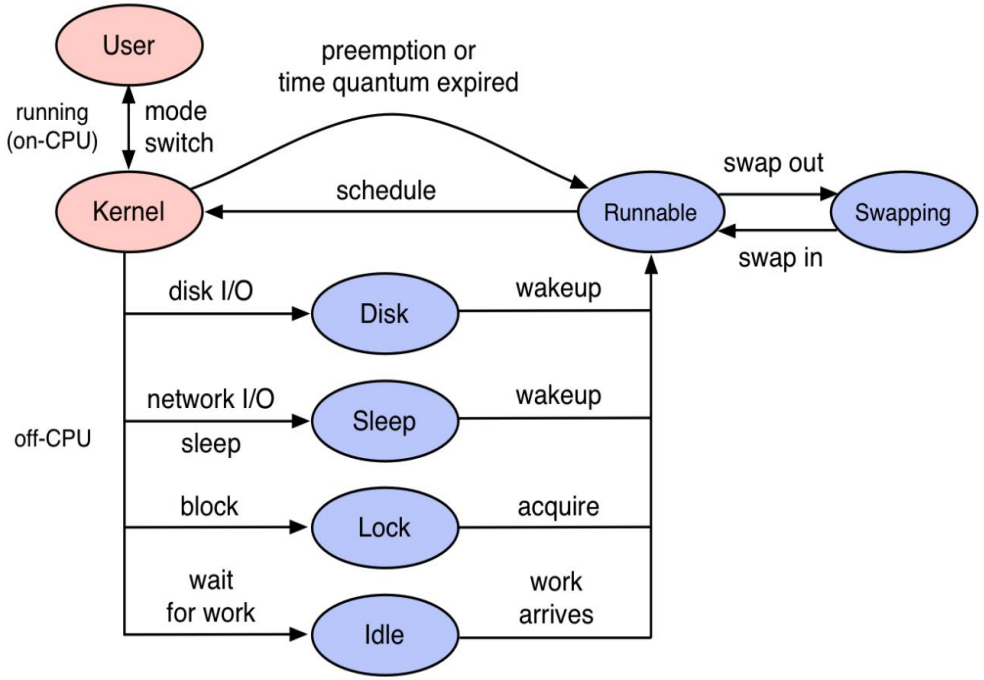
syscall

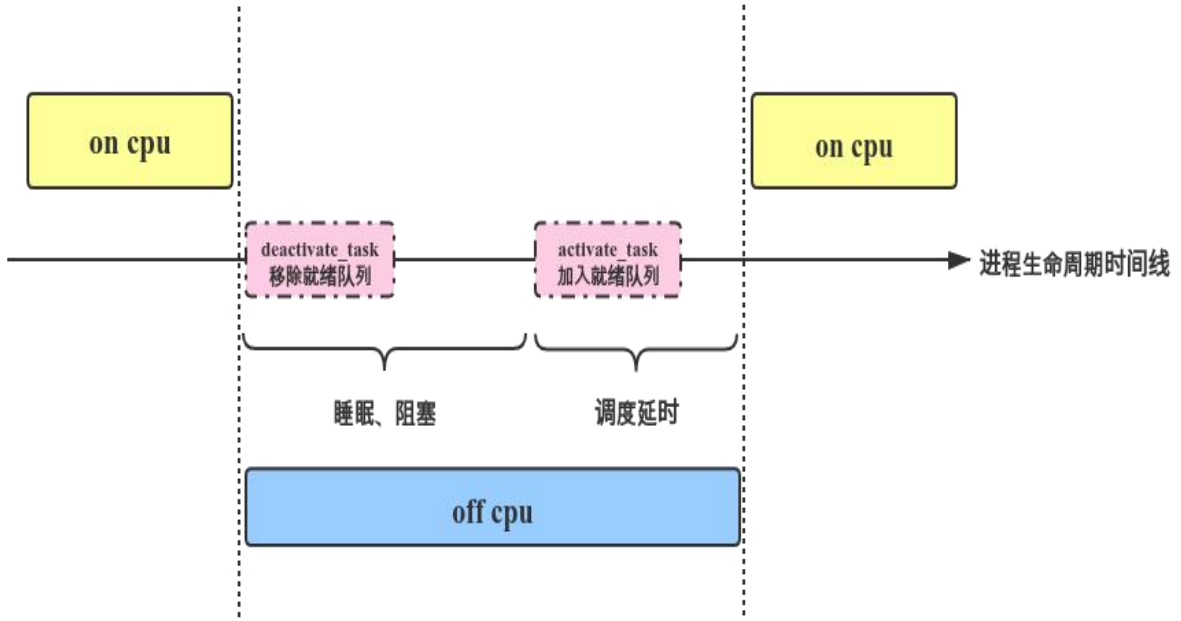
low level code can not be debug online easily

Trace-profiling open the blind spots

Align onCpu event with offCpu event from tread level

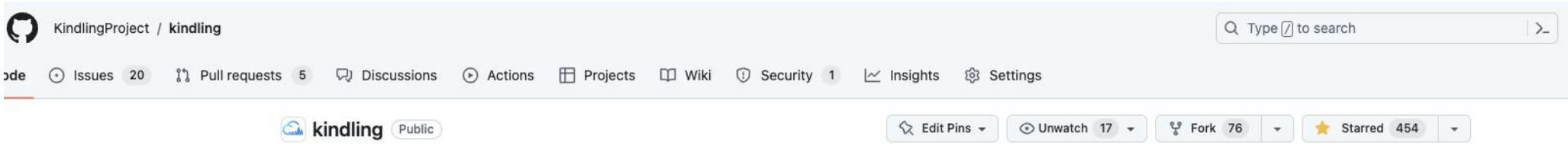
integration with distributed tracing





Kindling open source projct

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2023

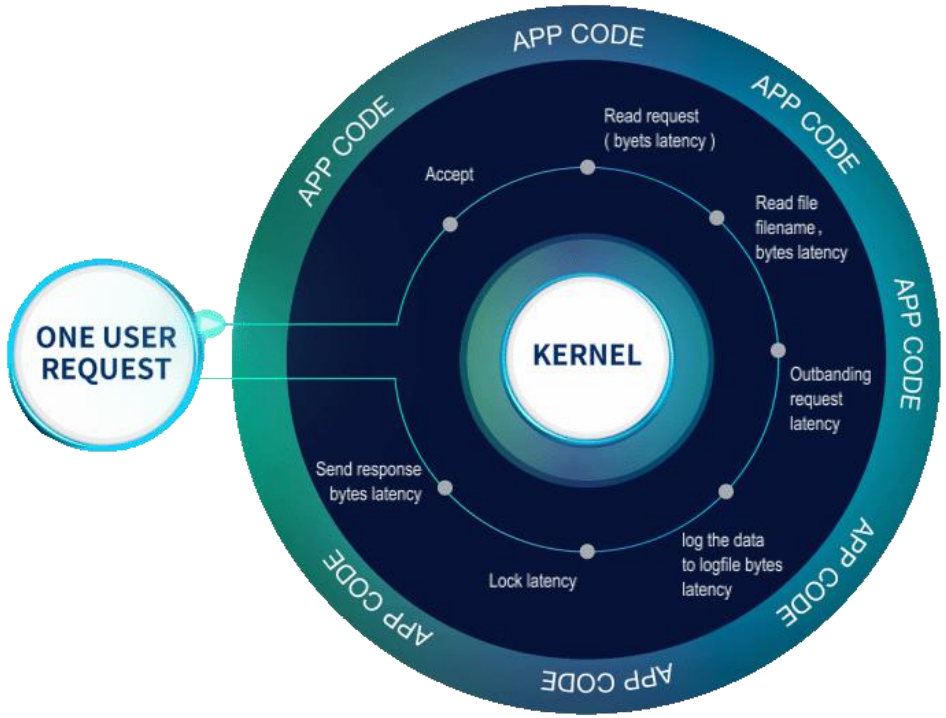


Kindling

eBPF-based Cloud Native Monitoring tool

Understanding the app behavior from kernel to app code

Get Started Watch the Demo



With trace-profiling

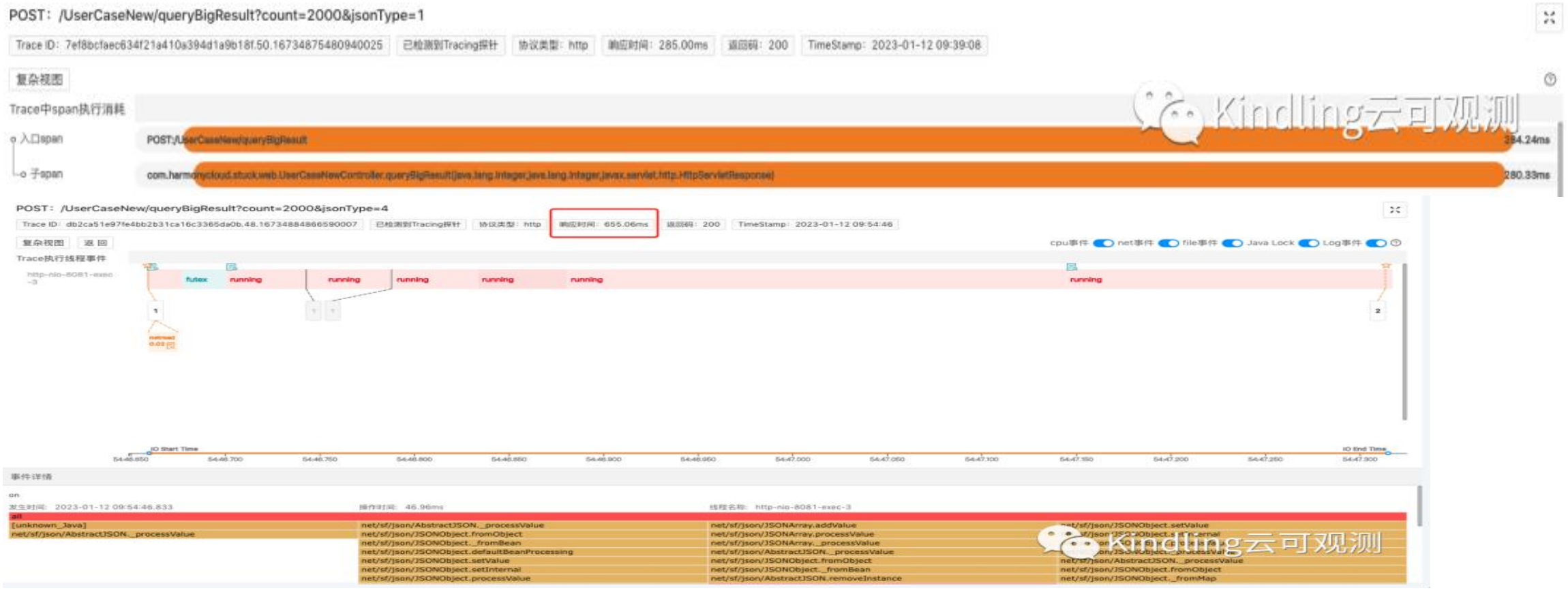
- How all threads were executed is recorded and can be displayed.
- The exact thread which executed the trace span is highlighted.
- The code execution flame graph is correlated to the time series where CPU is busy.
- The network-related metrics are correlated to the time series where the network syscalls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

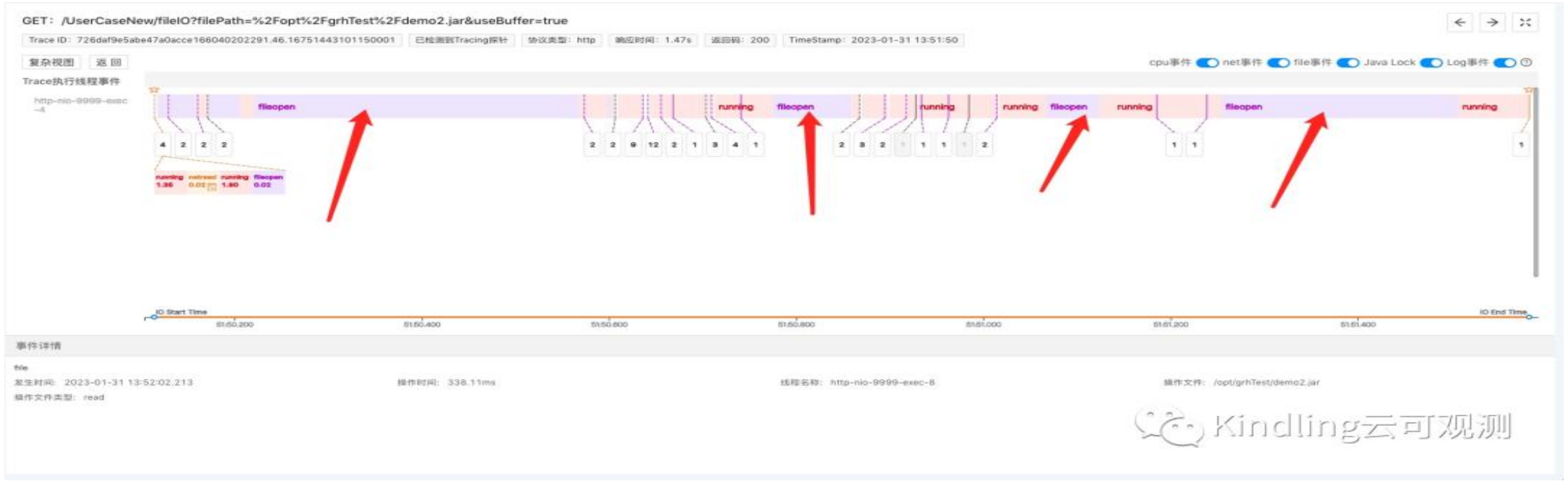
UseCase 2—— High CPU usage

- User's method cost high cpu



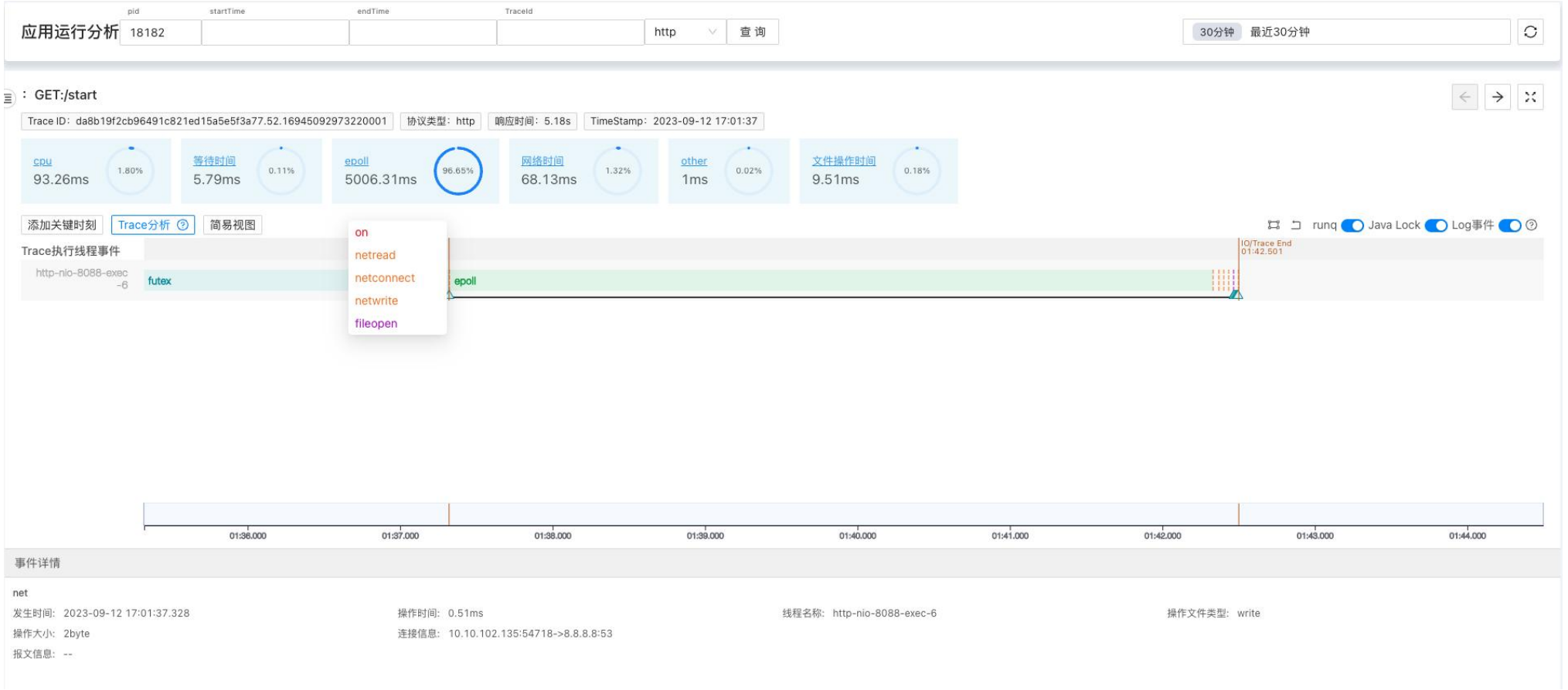
UseCase 3—— storage issue

- User method need access storage issue



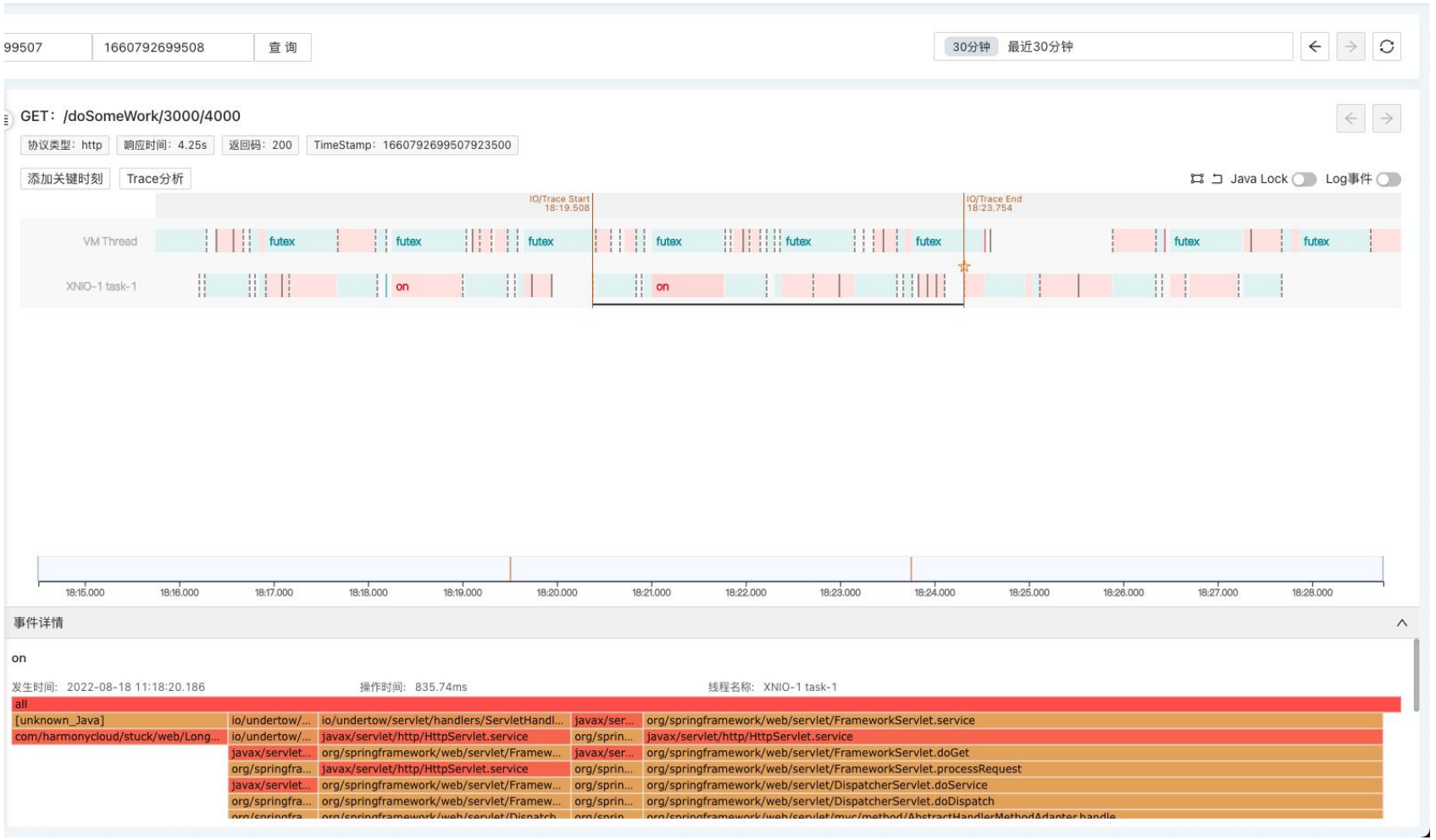
UseCase 4—— DNS issue

- 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



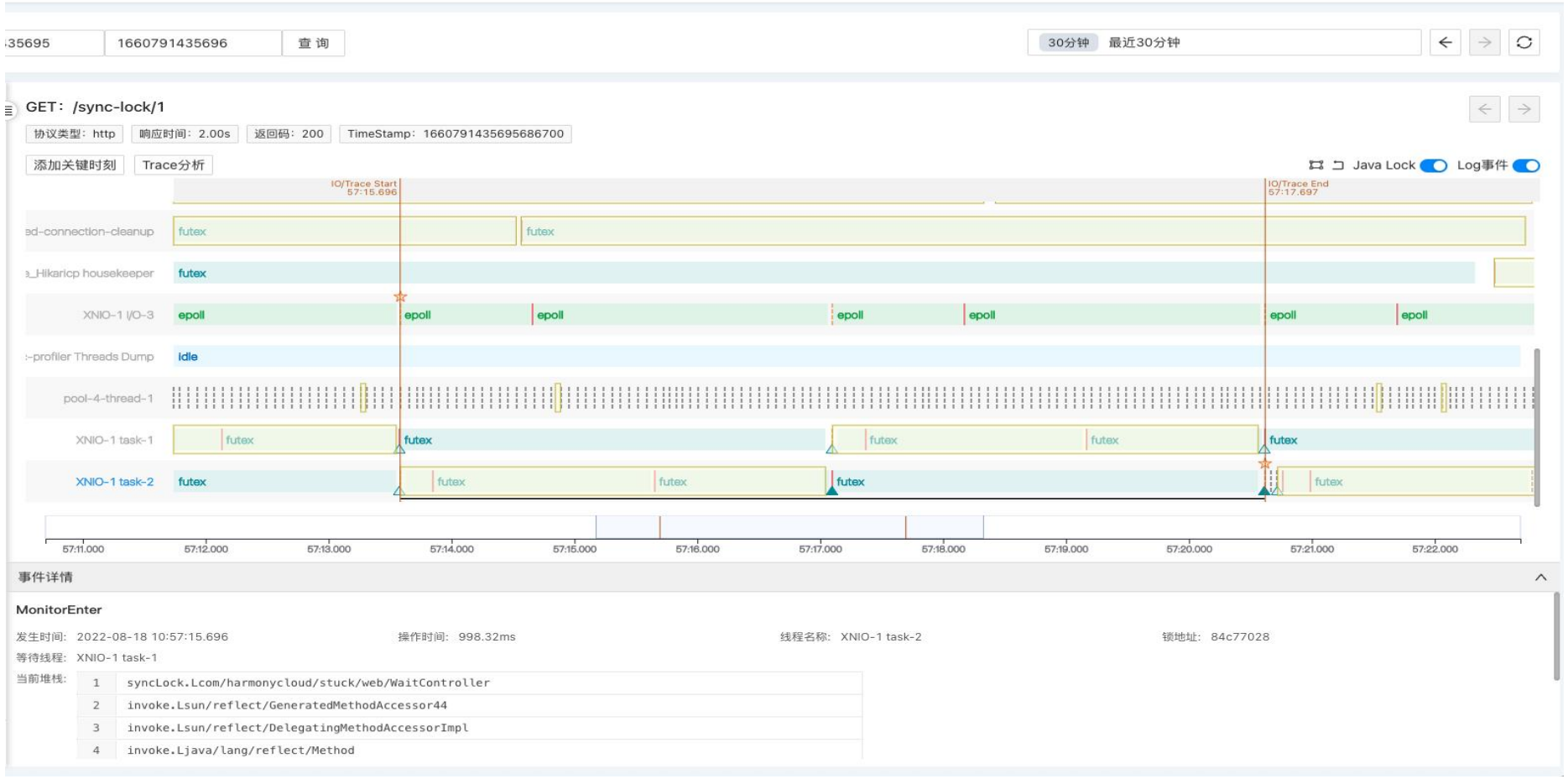
UseCase 5—— Java GC issue

- The user method has been stoped by GC



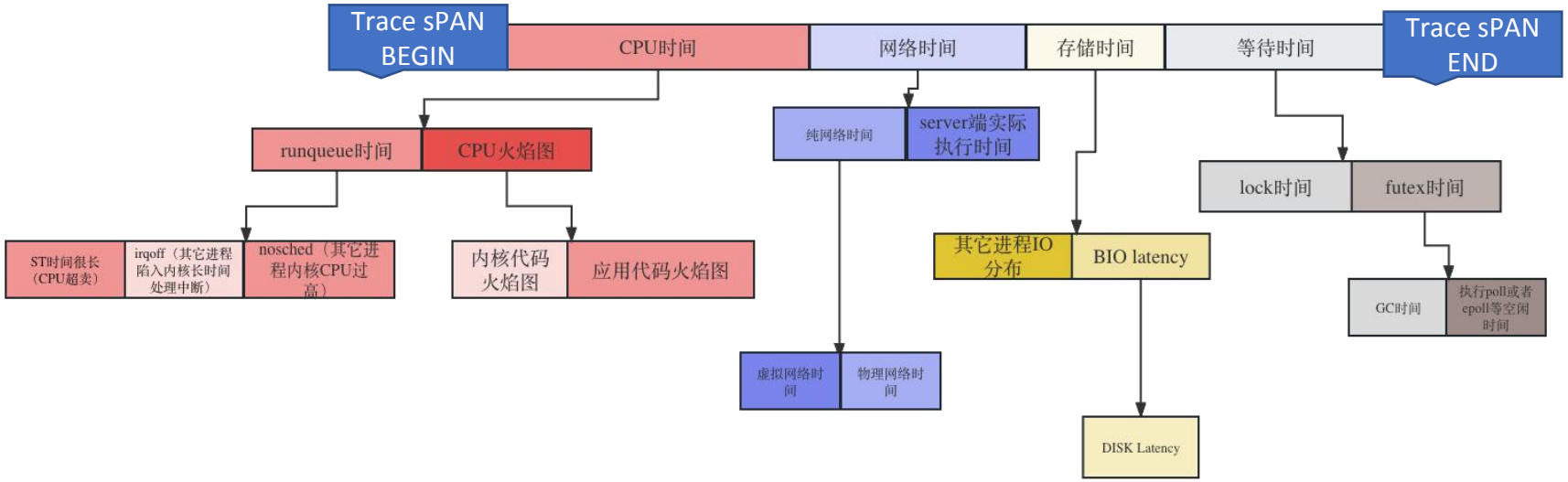
UseCase 6—— Lock issue

- 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
OPEN SOURCE SUMMIT
China 2023