穿透 K8s 容器的深度应用分析

使用动态追踪分析应用的前沿技术

OpenResty.com

章亦春
OpenResty 创始人

2023.2

软件世界的挑战

- 业务快速迭代
- 团队成员水平不一
- 测试不充分
- 业务复杂度不断提高

- CPU 资源占用过高
- 内存资源占用过多（含内存泄漏）
- 硬盘 IO 资源不足
- 存在长延时响应
- 很难线下复现的异常和错误（含进程崩溃）

K8s/Docker 容器时代的挑战

很多容器，很多应用，很多发行版，很多技术栈

容器最小化，缺乏最基本的调试工具

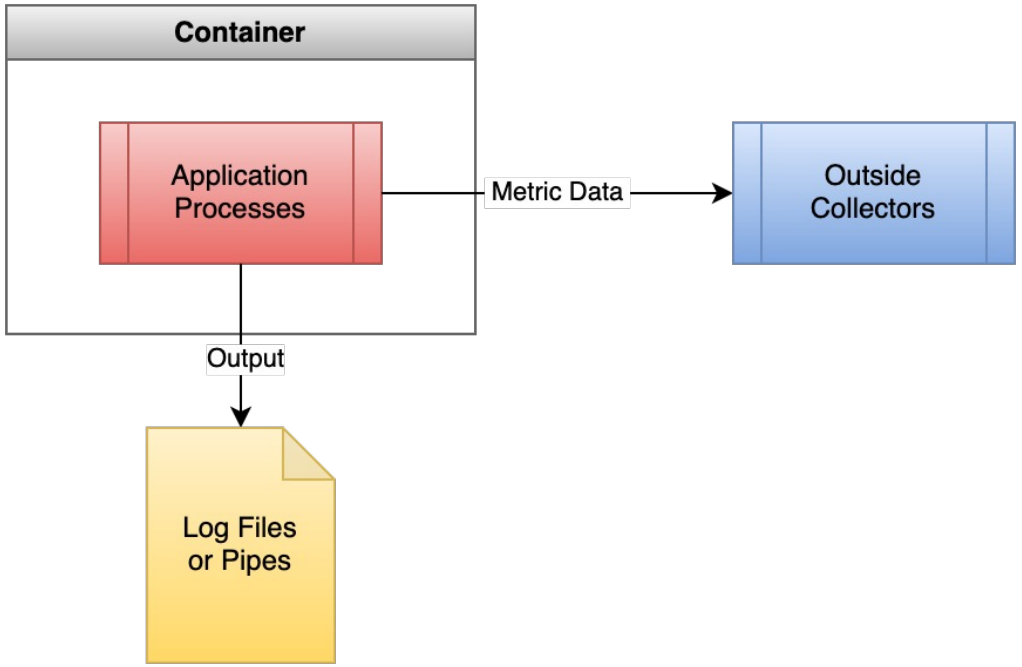
容器权限集合最小化

出问题时自动丢弃和重启容器，软件 Bug 容易被掩盖

容器虚拟化、微服务——软件复杂度进一步提高

K8s/Docker 容器的监控方法

- 日志文件采集和分析
- 埋点指标的输出接口



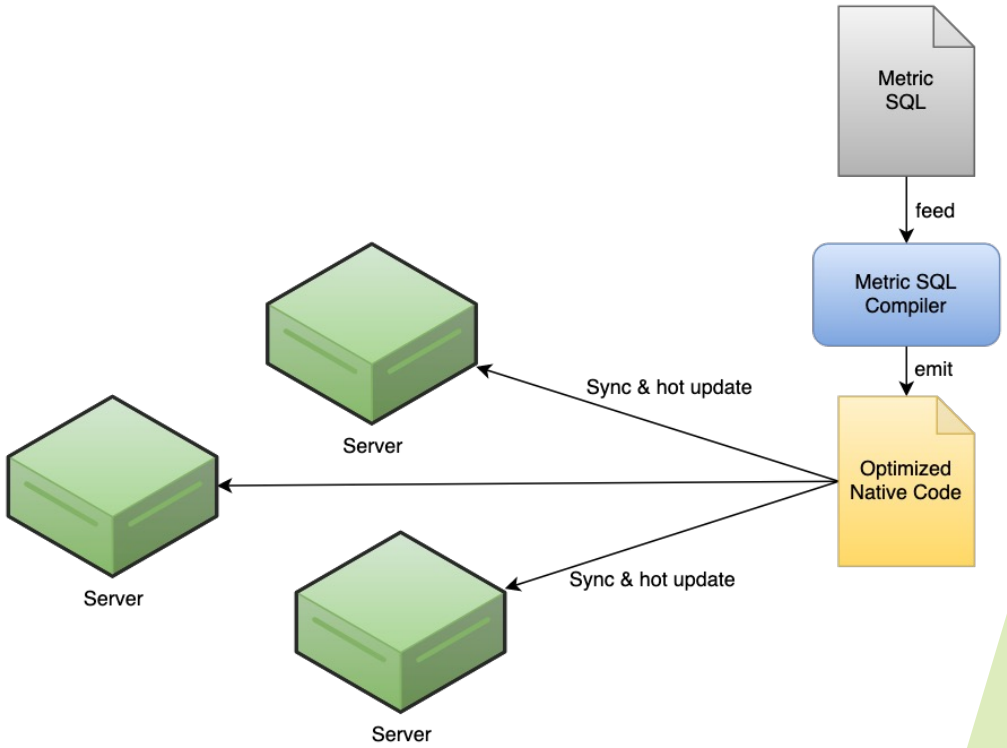
传统方法的缺点

- 侵入式需要修改应用
- 响应需求慢
- 需要大数据存储和分析
- 仅限表面指标
- 只有现象，没有原因
- 缺少深度全技术栈分析和诊断
- 数据采集和处理流程复杂，开销大，易出错

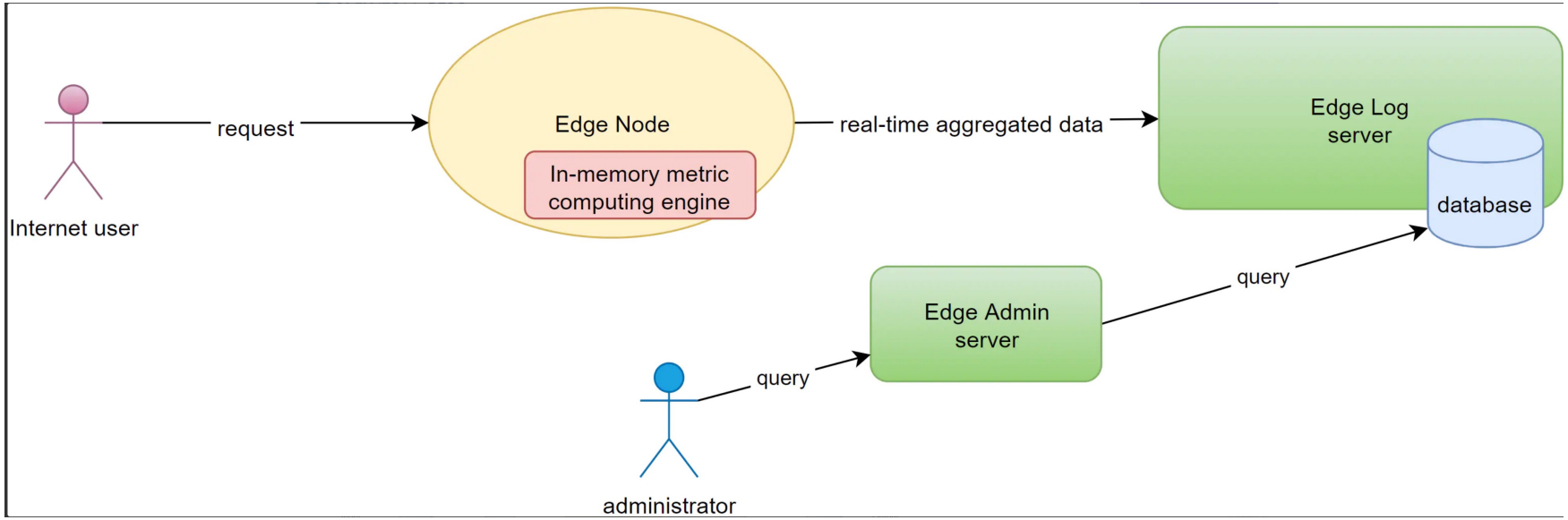
传统方法的优点

- 适合全量简单统计（如针对所有流量，但复杂统计开销增加显著）
- 适合基础指标监控

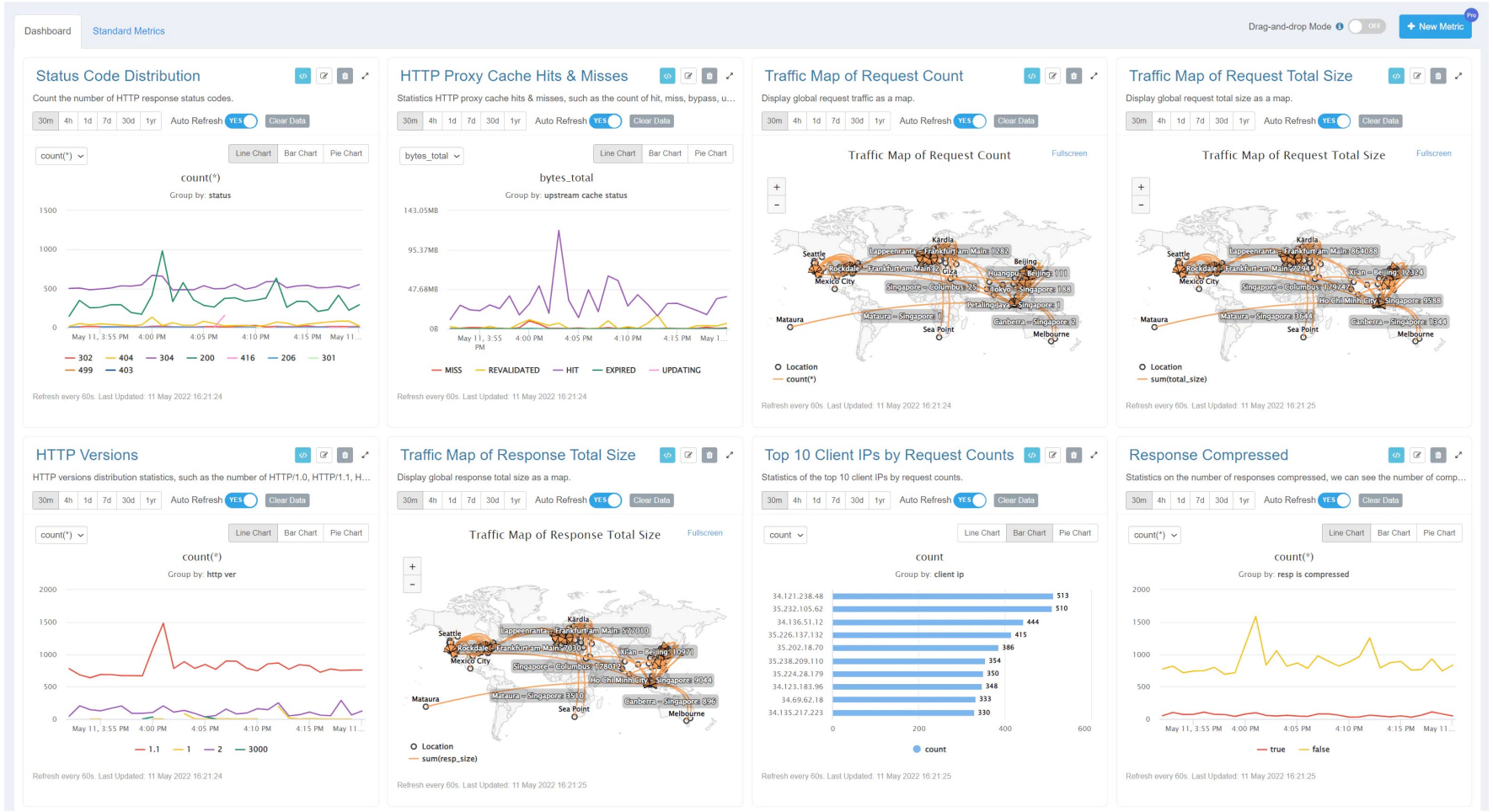
指标方式的改进：OpenResty Edge 动态指标



- 边缘机器内存内流式实时统计
- 全动态代码推送和热替换
- 使用类 SQL 语言表达统计指标和聚合汇总逻辑
- 常量内存使用
- 热插拔，随时增删改


request
Edge Node
In-memory metric computing engine
real-time aggregated data
Edge Log server
database
Internet user
query
Edge Admin server
query
administrator

流式在线多级聚合

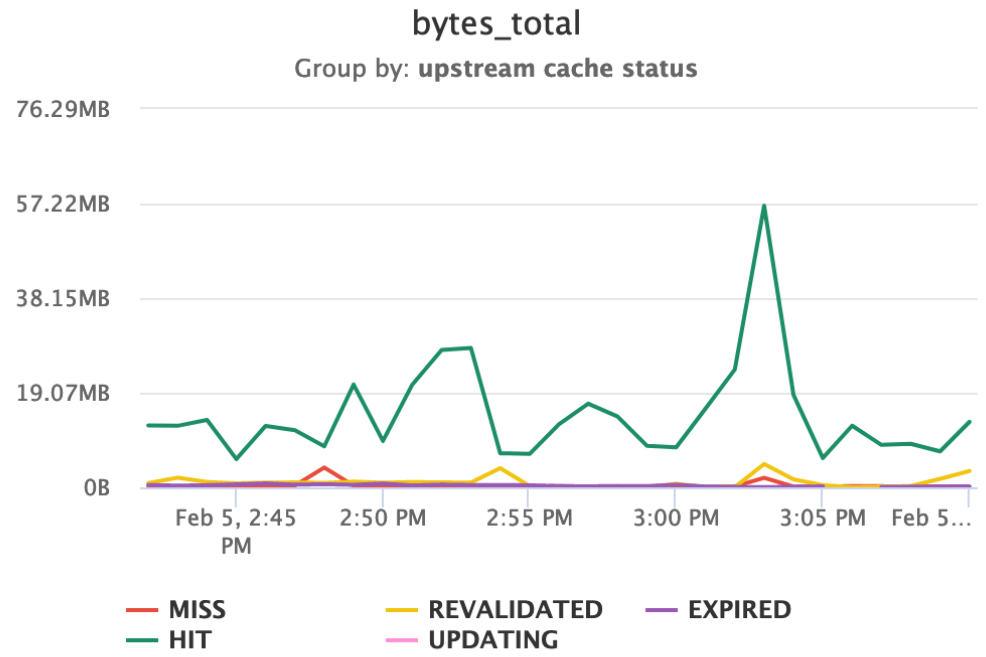

Dashboard
Standard Metrics
Drag-and-drop Mode
New Metric
Status Code Distribution
Count the number of HTTP response status codes.
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
count(*)
Line Chart Bar Chart Pie Chart
count(*)
Group by: status
1500
1000
500
0
May 11, 3:55 PM 4:00 PM 4:05 PM 4:10 PM 4:15 PM May 11...
302 404 304 200 416 206 301 499 403
Refresh every 60s. Last Updated: 11 May 2022 16:21:24
HTTP Proxy Cache Hits & Misses
Statistics HTTP proxy cache hits & misses, such as the count of hit, miss, bypass, u...
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
bytes_total
Line Chart Bar Chart Pie Chart
bytes_total
Group by: upstream cache status
143.05MB
95.37MB
47.68MB
0B
May 11, 3:55 PM 4:00 PM 4:05 PM 4:10 PM 4:15 PM May 1...
MISS REVALIDATED HIT EXPIRED UPDATING
Refresh every 60s. Last Updated: 11 May 2022 16:21:24
Traffic Map of Request Count
Display global request traffic as a map.
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
Traffic Map of Request Count
Fullscreen
Seattle
Rockdale
Mexico City
Mataura
Sea Point
Melbourne
Kärdla
Beijing
Giza
Tokyo
Location
count(*)
Refresh every 60s. Last Updated: 11 May 2022 16:21:24
Traffic Map of Request Total Size
Display global request total size as a map.
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
Traffic Map of Request Total Size
Fullscreen
Seattle
Rockdale
Mexico City
Mataura
Sea Point
Melbourne
Kärdla
Location
sum(total_size)
Refresh every 60s. Last Updated: 11 May 2022 16:21:25
HTTP Versions
HTTP versions distribution statistics, such as the number of HTTP/1.0, HTTP/1.1, H...
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
count(*)
Line Chart Bar Chart Pie Chart
count(*)
Group by: http ver
2000
1500
1000
500
0
May 11, 3:55 PM 4:00 PM 4:05 PM 4:10 PM 4:15 PM May 11...
1.1 1 2 3000
Refresh every 60s. Last Updated: 11 May 2022 16:21:24
Traffic Map of Response Total Size
Display global response total size as a map.
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
Traffic Map of Response Total Size
Fullscreen
Seattle
Rockdale
Mexico City
Mataura
Sea Point
Melbourne
Kärdla
Location
sum(resp_size)
Refresh every 60s. Last Updated: 11 May 2022 16:21:25
Top 10 Client IPs by Request Counts
Statistics of the top 10 client IPs by request counts.
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
count
Line Chart Bar Chart Pie Chart
count
Group by: client ip
34.121.238.48 513
35.232.105.62 510
34.136.51.12 444
35.226.137.132 415
35.202.18.70 386
35.238.209.110 354
35.224.28.179 350
34.123.183.96 348
34.69.62.18 333
34.135.217.223 330
0 200 400 600
count
Refresh every 60s. Last Updated: 11 May 2022 16:21:25
Response Compressed
Statistics on the number of responses compressed, we can see the number of comp...
30m 4h 1d 7d 30d 1yr Auto Refresh YES Clear Data
count(*)
Line Chart Bar Chart Pie Chart
count(*)
Group by: resp is compressed
2000
1500
1000
500
0
May 11, 3:55 PM 4:00 PM 4:05 PM 4:10 PM 4:15 PM May 11...
true false
Refresh every 60s. Last Updated: 11 May 2022 16:21:25

HTTP Proxy Cache Hits & Misses

Statistics HTTP proxy cache hits & misses, such as the count of hit, miss, byp…

30m | 4h | 1d | 7d | 30d | 1yr Auto Refresh YES Clear Data

bytes_total Line Chart | Bar Chart | Pie Chart



Refresh every 60s. Last Updated: 5 Feb 2023 15:11:45

Edit Metric

Metric Name * HTTP Proxy Cache Hits & Misses

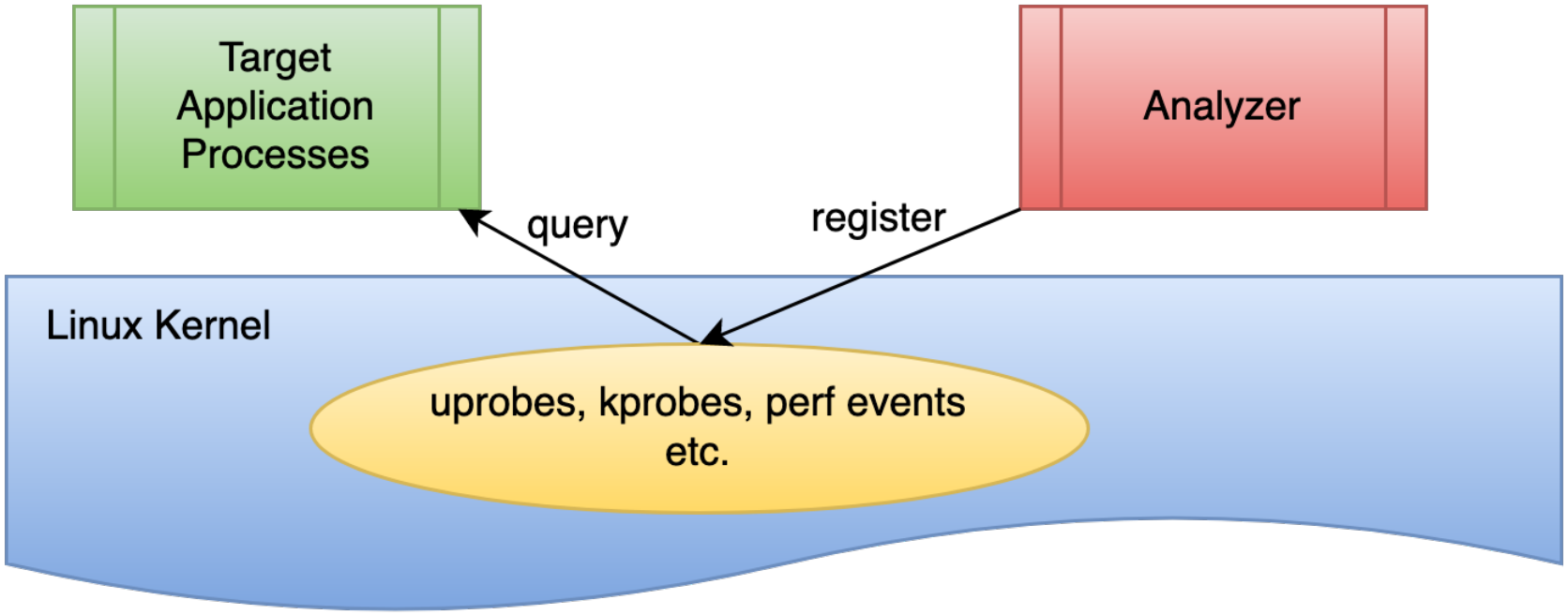
Report Interval * 60 seconds

Metric SQL * Metric SQL User Manual

```
1 select sum(total_size) as bytes_total,
  upstream_cache_status, count(*)
2 from reqs
3 where upstream_cache_status != ''
4 group by upstream_cache_status;
```

Cancel Submit

动态追踪技术


Target
Application
Processes
Analyzer
query
register
Linux Kernel
uprobes, kprobes, perf events
etc.

动态追踪技术

- Solaris Dtrace (FreeBSD, Oracle Linux, macOS)
- SystemTap
- eBPF -> clang/llvm/bcc/bpftrace
- GDB
- OpenResty XRay

动态追踪的优点

- 非侵入式，无需修改应用
- 热插拔，通常无需应用配合（很多开源动态追踪工具有时还是要求应用配合）
- 开销通常较低，可以在数据源进行聚合汇总（开源工具不注意仍然可能过载目标应用）
- 事发后的在线实时调试能力
- 全技术栈分析无死角
- 按需采样
- 不采样时严格 0 损耗

SystemTap 的缺点

- 分析工具的开发成本高
- 提供的 Stap 脚本语言和 C 或其他通用目的语言相差大
- 通常依赖目标机器安装有 Linux 内核头文件、Linux 内核调试符号、C 编译器工具链等等很多依赖项
- 缺少直接穿透容器的支持

eBPF 工具链的缺点

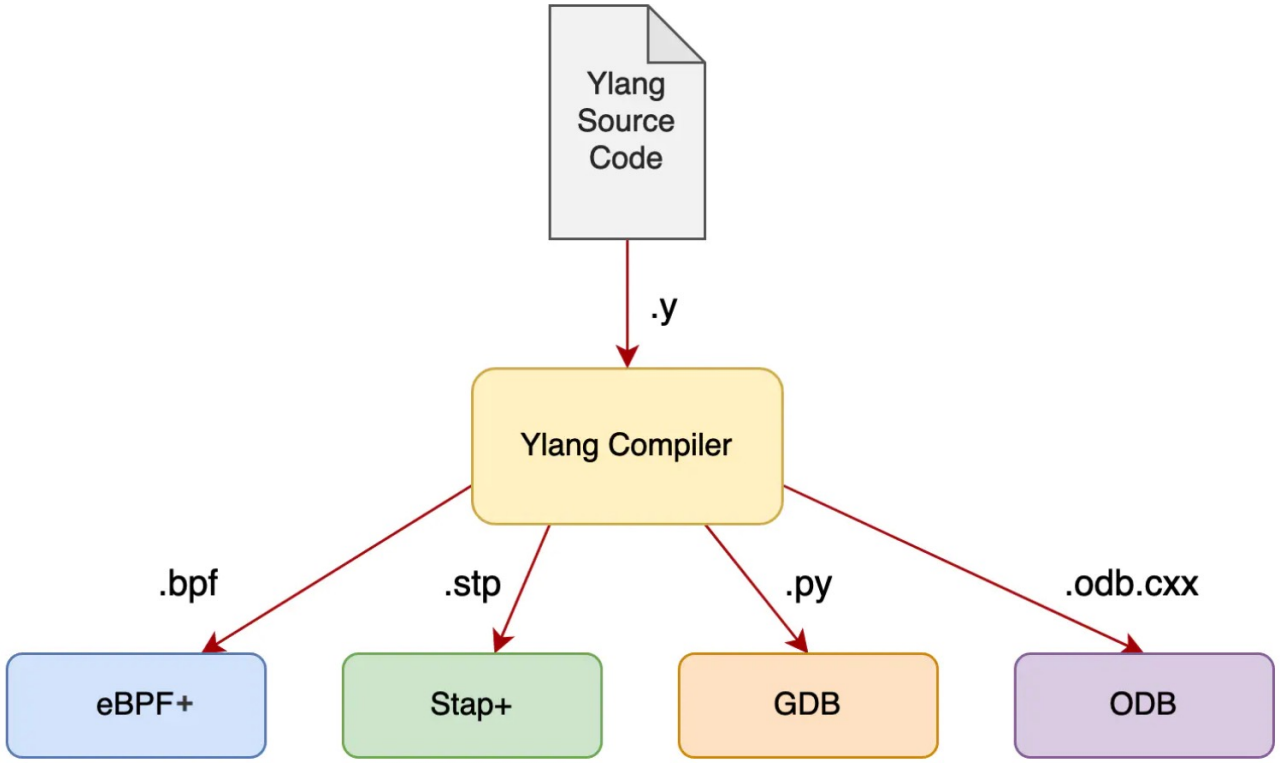
- 要求很新的 Linux 内核
- BCC 等工具链也像 SystemTap 那样依赖目标机器安装有 Linux 内核头文件、Clang/LLVM 编译器工具链等等很多依赖项
- Bpftrace 的脚本语言的限制比 SystemTap 的脚本语言多得多
- eBPF C 语言只支持极简单的程序逻辑
- eBPF C 语言没有原生字符串支持
- eBPF C 语言不支持一般的循环、递归函数调用和 goto 控制流
- eBPF C 语言的自定义函数只支持最多 5 个参数
- eBPF C 语言不支持一般字符串或其他复合类型数据的常量（依赖 .rodata 段）
- 加载较复杂的 eBPF 程序开销极大（指数时间复杂度）
- eBPF verifier 对用户程序的校验过于严苛
- 缺少直接穿透容器分析用户态代码的支持

GDB 的缺点

- 依赖 Linux 的 ptrace 系统调用
- 断点开销大，容易有历史 bug
- 会改变应用进程的父子关系
- 启动慢
- 运行 GDB 原生命令语言和 Python 代码的效率很低
- 很臃肿

OpenResty XRay

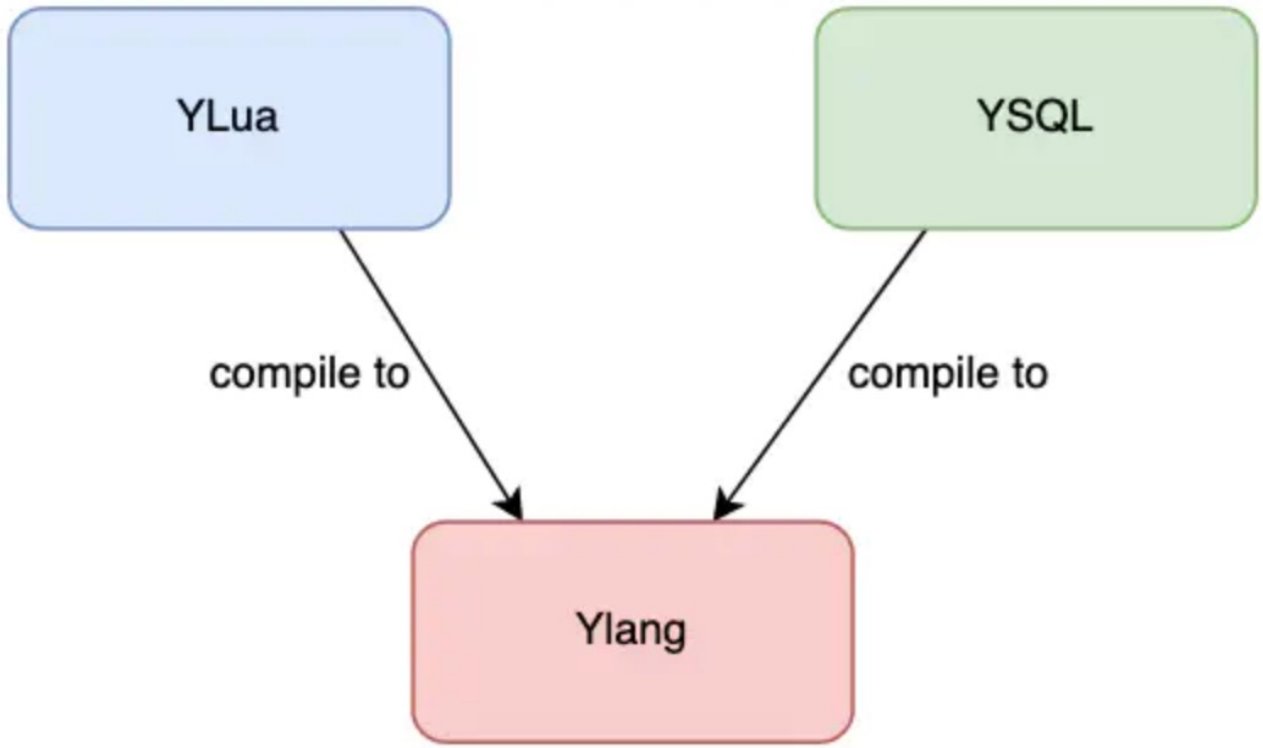
- Y 语言（Ylang）编译器（支持 GNU C 和标准 C 语言的绝大部分语法）
- Ylua 语言
- YSQL 语言
- Stap+ 显著改进了 SystemTap
- eBPF+ 显著改进了 eBPF（同时 LLVM+ 也大大改进了开源 LLVM)
- ODB 是超轻量版的 GDB
- Ylang 编译器也能生成高度优化的 GDB 的 Python 扩展代码
- 针对线上生产环境的严格性能损耗控制



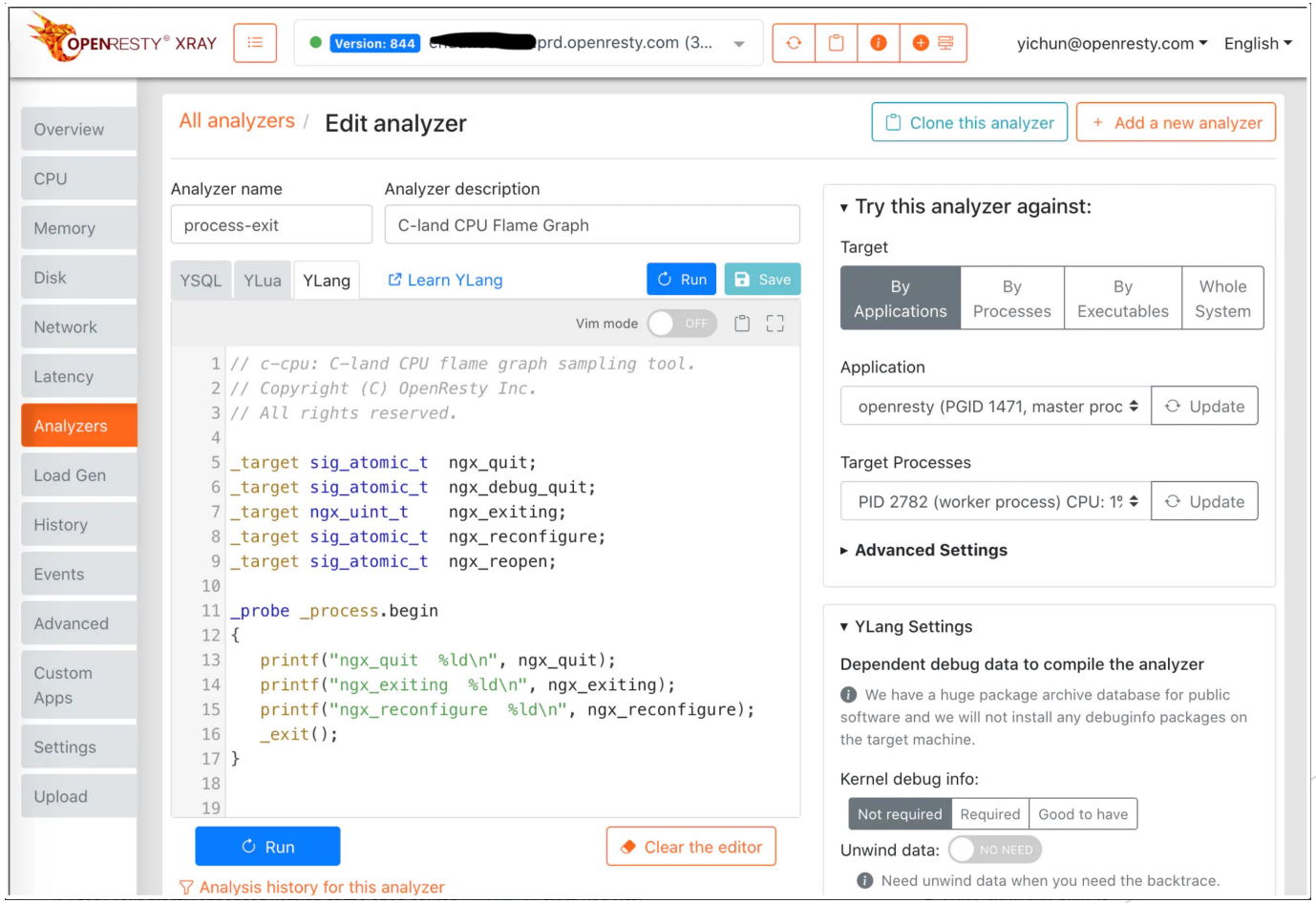
“一次编写，到处运行”

从 Y 语言代码到各种不同运行时的代码

Y 语言之上更抽象的语言



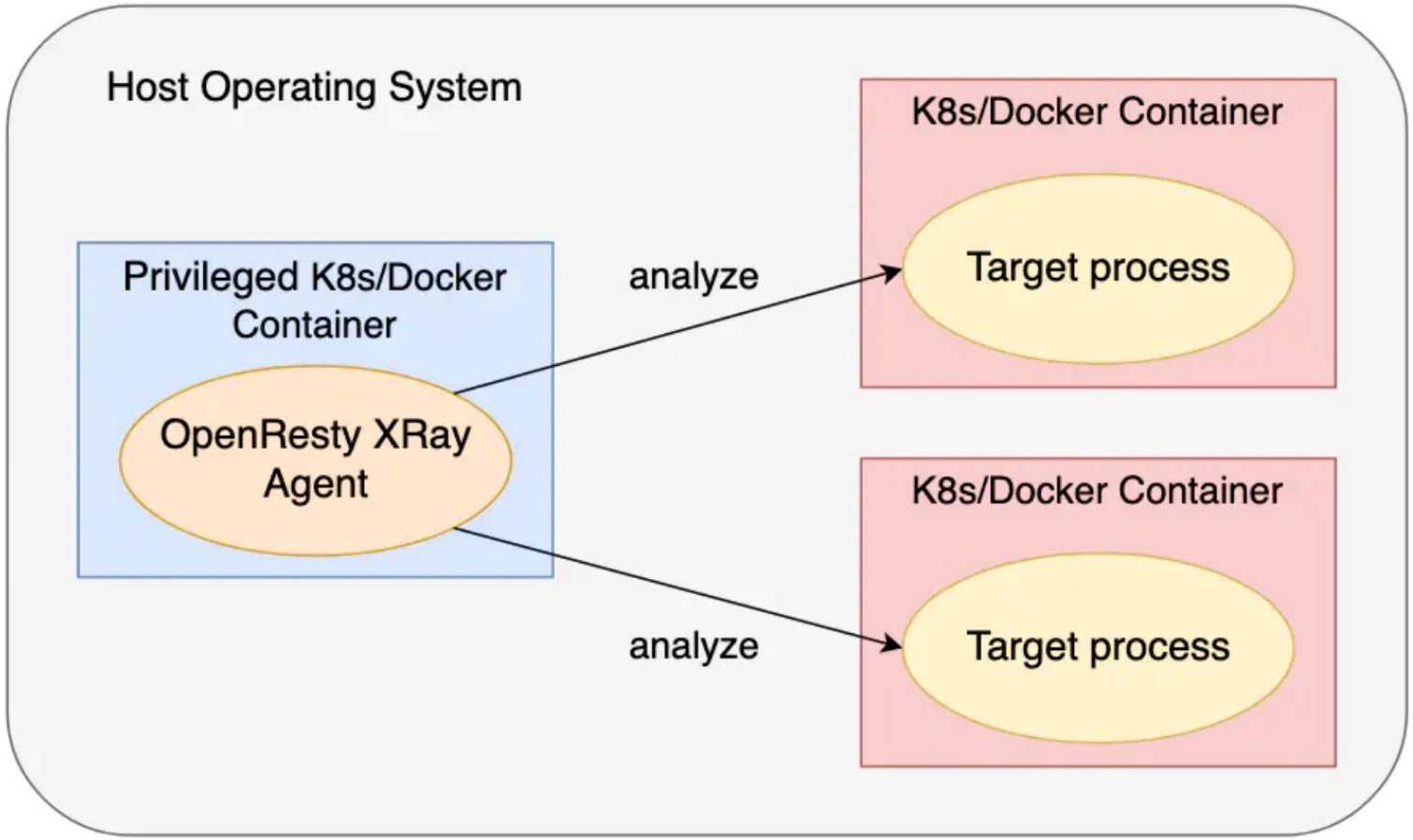
在 OpenResty XRay 界面上编写和调试 Ylang/YLua/YSQL 等语言的分析工具



OpenResty XRay 数百种标准分析器

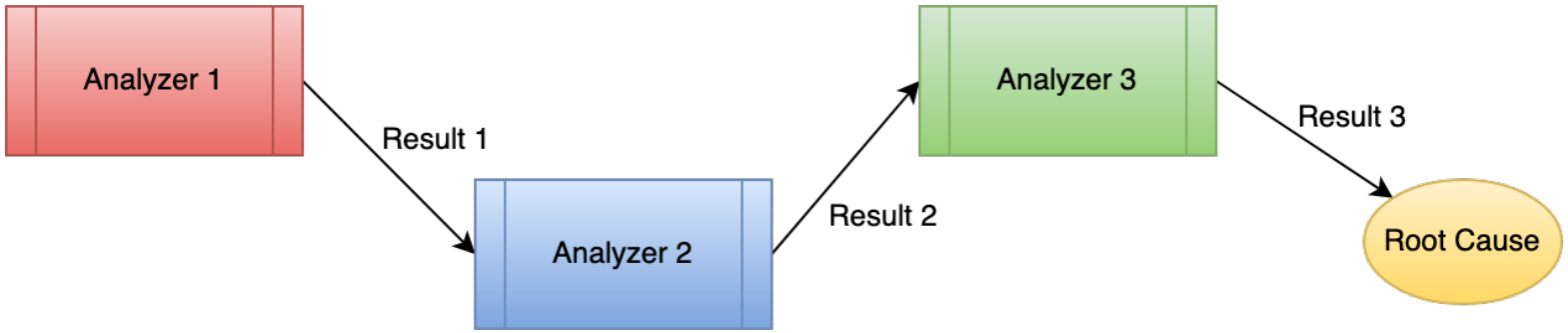
- OpenResty XRay Standard Analyzers
  - c-alloc-fgraph
  - c-count-alloc-free
  - c-memory
  - c-memory-leak-fgraph
  - c-off-cpu
  - c-on-cpu
  - collect-luajit-ffnames
  - cpu-hogs
  - epoll-loop-blocking-distr
  - epoll-sched-latency-distr
  - epoll-wait-ret-distr
  - epoll-wait-timers
  - epoll-wait-timers-fgraph
  - file-system-fgraph
  - func-latency-distr
  - glibc-chunks
  - jemalloc-bins
  - kernel-on-cpu
  - lj-add-timer-lua-fgraph
  - lj-alloc-stats
  - lj-c-memory-leak-fgraph
  - lj-c-off-cpu
  - lj-c-on-cpu
  - lj-config
  - lj-dump-loaded-mods
  - lj-err-mem
  - lj-excep-lua-fgraph
  - lj-free-stats
  - lj-gc-step-calls
  - lj-lua-exception
  - lj-gco-ref
  - lj-gco-stat
  - lj-lua-err-msg
  - lj-lua-new-timer-errors
  - lj-lua-newcdata
  - lj-lua-newfunc
  - lj-lua-newgco
  - ngx-add-timer-event-fgraph
  - lj-trace-stats
  - ngx-add-timer-event-timer-distr
  - lj-vm-states
  - mmap-leaks
  - musl-libc-chunks
  - ngx-access-log-buffer-size
  - ngx-config
  - ngx-config-servers
  - ngx-cpu-hottest-hosts
  - ngx-cpu-hottest-uris
  - ngx-downstream-keepalive-stats
  - ngx-dump-req
  - ngx-dump-timers
  - ngx-epoll-wait-timers
  - ngx-err-log-lvl-distr

OpenResty XRay 穿透容器分析应用



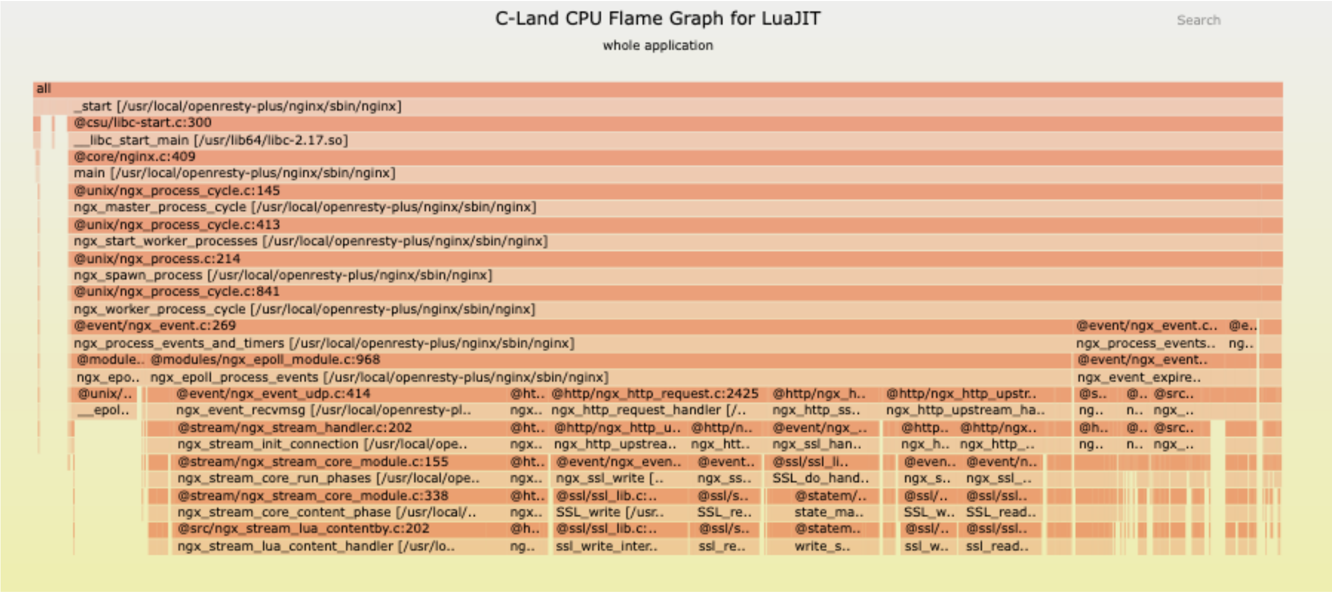
OpenResty XRay 全自动采样无人职守的使用模式

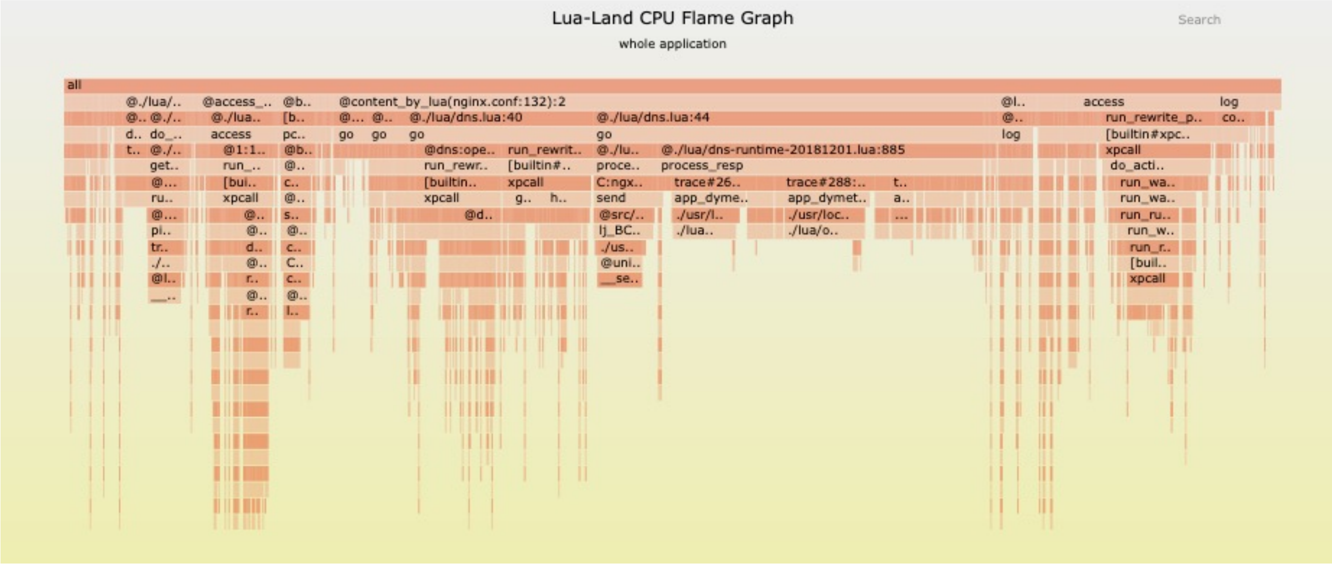
- 定时采样
- 事件驱动采样（CPU 变化，内存变化，IO 变化，异常错误）
- 链式推理驱动



OpenResty XRay CPU 性能分析

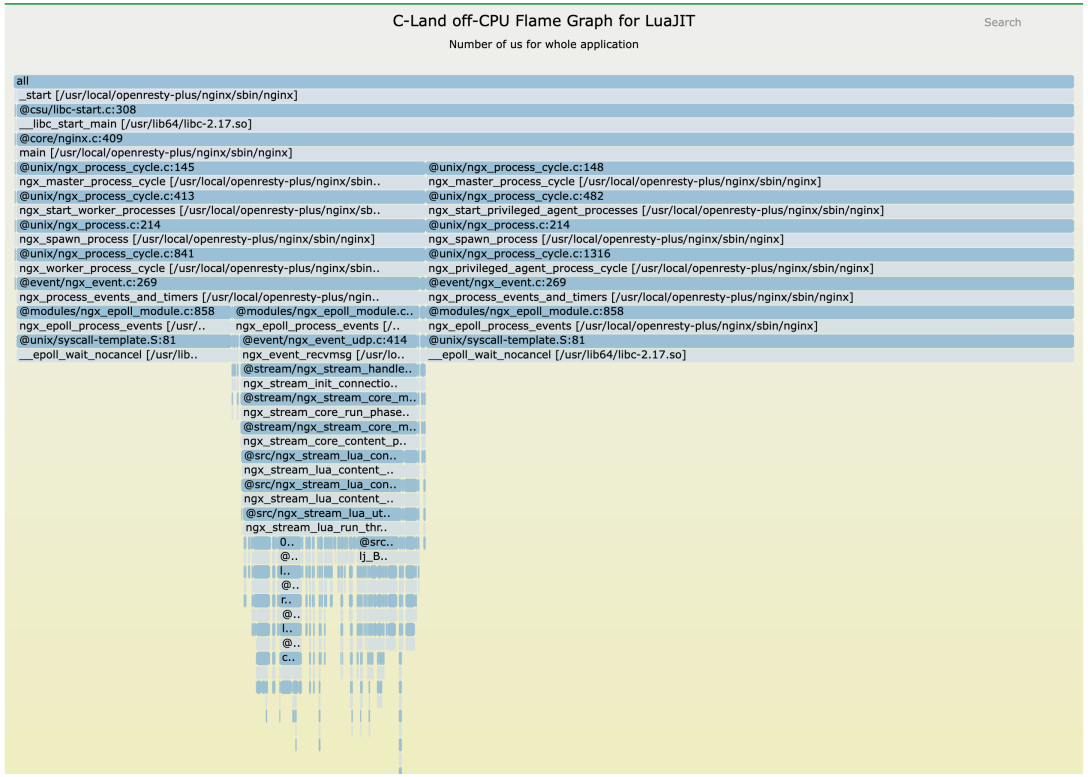
- 高 CPU 使用率会降低系统稳定性和服务质量，甚至导致服务不可用
- CPU 时间在不同场景下，是如何分布在不同代码路径上的（火焰图，火焰图自动解释器）
- 覆盖不同软件层面的代码路径：业务编程语言层面（Lua/Python/PHP/Perl/Go/等等），系统编程语言层面（C/C++/Rust），操作系统内核层面（网络协议栈/进程调度器/内存管理/系统调用）
- 常见的 CPU 瓶颈举例：重复计算（缺少缓存）、SSL 握手相关、垃圾回收（GC）开销，动态内存分配开销、序列化与反序列化、意外的密集系统调用、死循环、病态正则表达式匹配、实现低效的（第三方）软件库、自旋锁竞争


C-Land CPU Flame Graph for LuaJIT
whole application
Search
all
_start [/usr/local/openresty-plus/nginx/sbin/nginx]
@csu/libc-start.c:300
__libc_start_main [/usr/lib64/libc-2.17.so]
@core/nginx.c:409
main [/usr/local/openresty-plus/nginx/sbin/nginx]
@unix/ngx_process_cycle.c:145
ngx_master_process_cycle [/usr/local/openresty-plus/nginx/sbin/nginx]
@unix/ngx_process_cycle.c:413
ngx_start_worker_processes [/usr/local/openresty-plus/nginx/sbin/nginx]
@unix/ngx_process.c:214
ngx_spawn_process [/usr/local/openresty-plus/nginx/sbin/nginx]
@unix/ngx_process_cycle.c:841
ngx_worker_process_cycle [/usr/local/openresty-plus/nginx/sbin/nginx]
@event/ngx_event.c:269
ngx_process_events_and_timers [/usr/local/openresty-plus/nginx/sbin/nginx]
@modules/ngx_epoll_module.c:968
ngx_epoll_process_events [/usr/local/openresty-plus/nginx/sbin/nginx]
@event/ngx_event_udp.c:414
ngx_event_recvmsg [/usr/local/openresty-pl..
@stream/ngx_stream_handler.c:202
ngx_stream_init_connection [/usr/local/ope..
@stream/ngx_stream_core_module.c:155
ngx_stream_core_run_phases [/usr/local/ope..
@stream/ngx_stream_core_module.c:338
ngx_stream_core_content_phase [/usr/local/..
@src/ngx_stream_lua_contentby.c:202
ngx_stream_lua_content_handler [/usr/lo..
@http/ngx_http_request.c:2425
ngx_http_request_handler [/..
@http/ngx_http_upstr..
ngx_http_upstream_ha..
SSL_do_hand..
ngx_event_expire..


Lua-Land CPU Flame Graph
whole application
Search
all
@content_by_lua(nginx.conf:132):2
@./lua/dns.lua:40
@./lua/dns.lua:44
@./lua/dns-runtime-20181201.lua:885
process_resp
access
run_rewrite_p..
xpcall
log

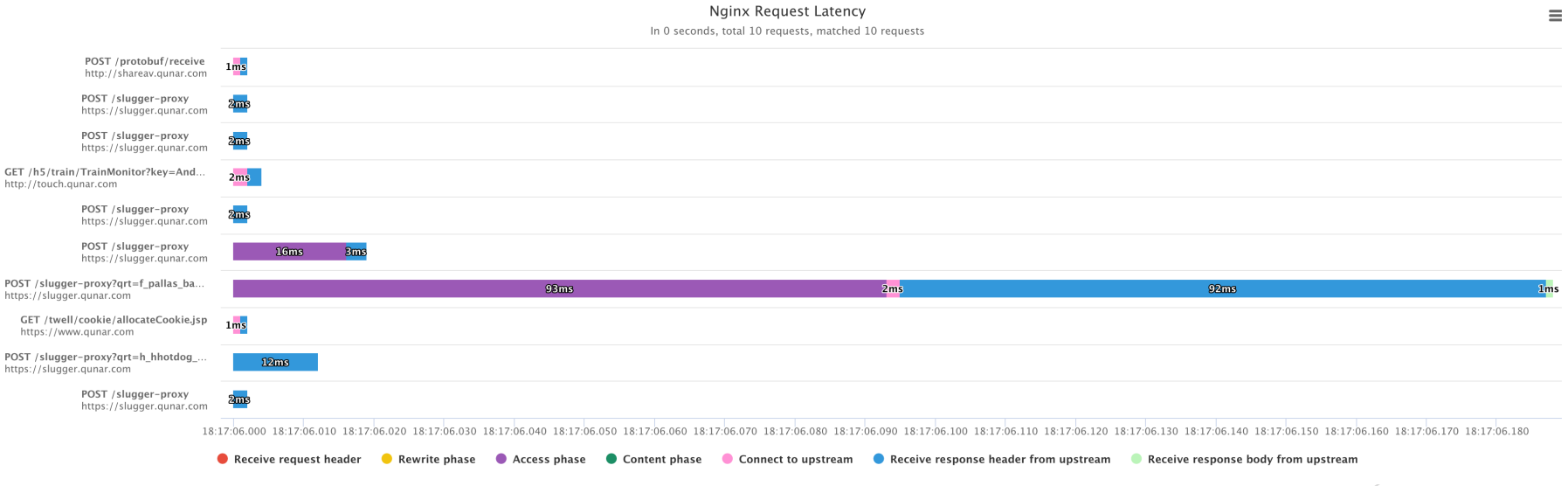
OpenResty XRay CPU 阻塞（off-CPU）分析

- 长时间阻塞的系统调用
- 系统锁竞争（semaphore），含死锁
- 操作系统进程调度器过载（过多的进程或线程争用 CPU 资源）



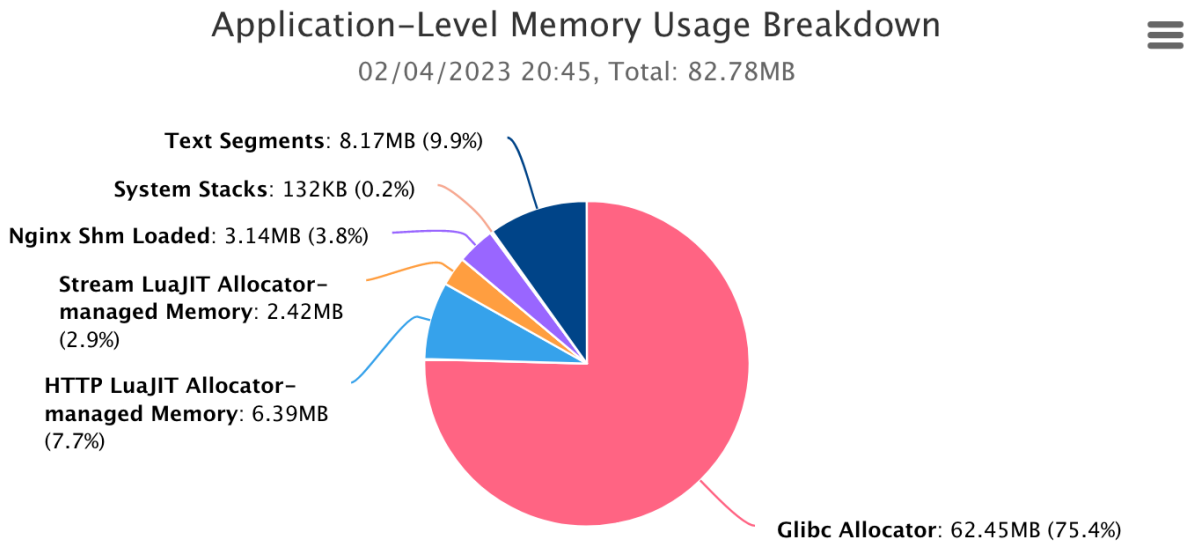
OpenResty XRay
请求延时分析

- 分解延时到不同应用操作和处理阶段
- 精准抓包，只抓实际有问题的连接上的网络包（问题包括延时较高、超时、连接错误、上层应用报错等等）
- 异步非阻塞 IO 的延时统计（如 Lua 协程 yield 的时间在 Lua 代码路径上的分布）

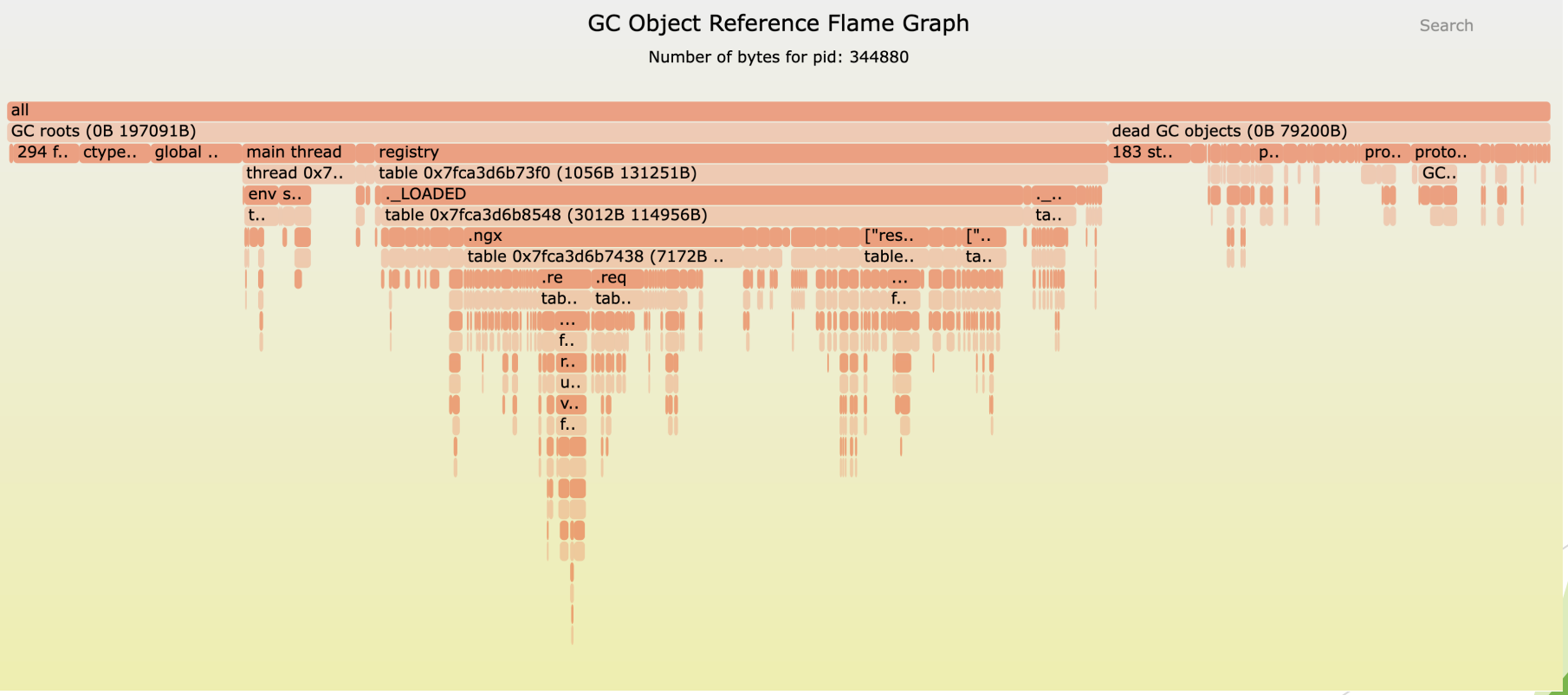


OpenResty XRay
内存使用分析

- Glibc/Jemalloc 等 C 内存分配器的内存使用（含 Glibc 内存碎片）
- 内存如何定量分布在所有的 GC 对象上（比如 Lua 对象、Python 对象、PHP 对象等等），按 GC 对象之间的引用关系。
- 内存泄漏，内存碎片，还是延迟释放？

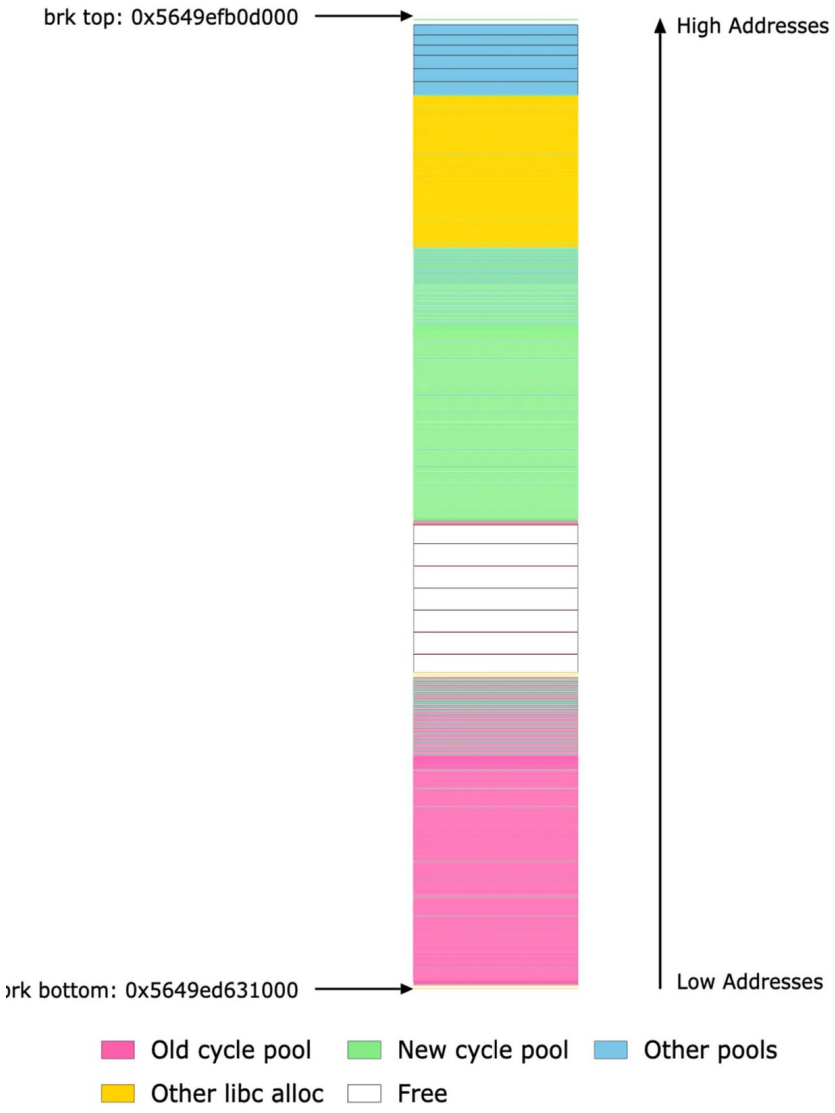


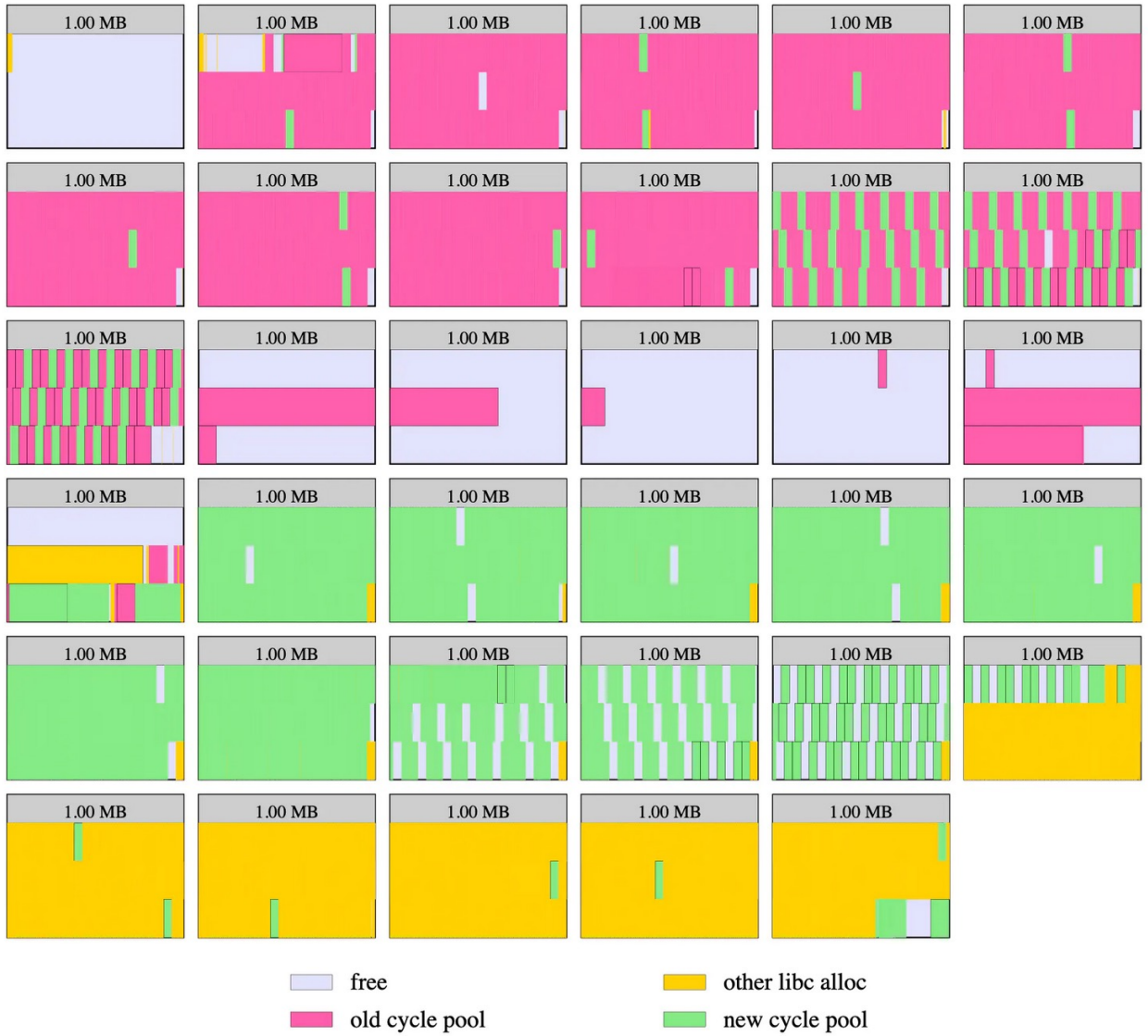
GC 对象引用关系火焰图
定量显示内存是如何在所有的对象引用路径上分布的



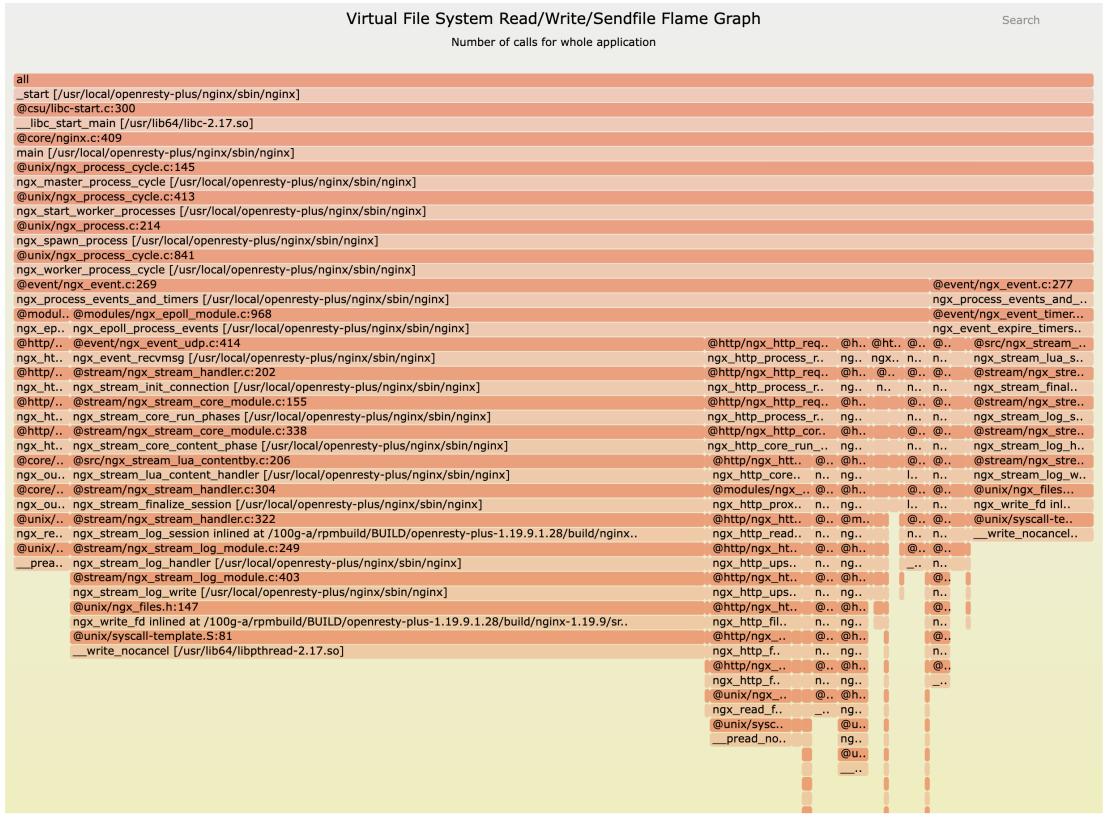
Nginx memory pool allocatons via the brk syscall

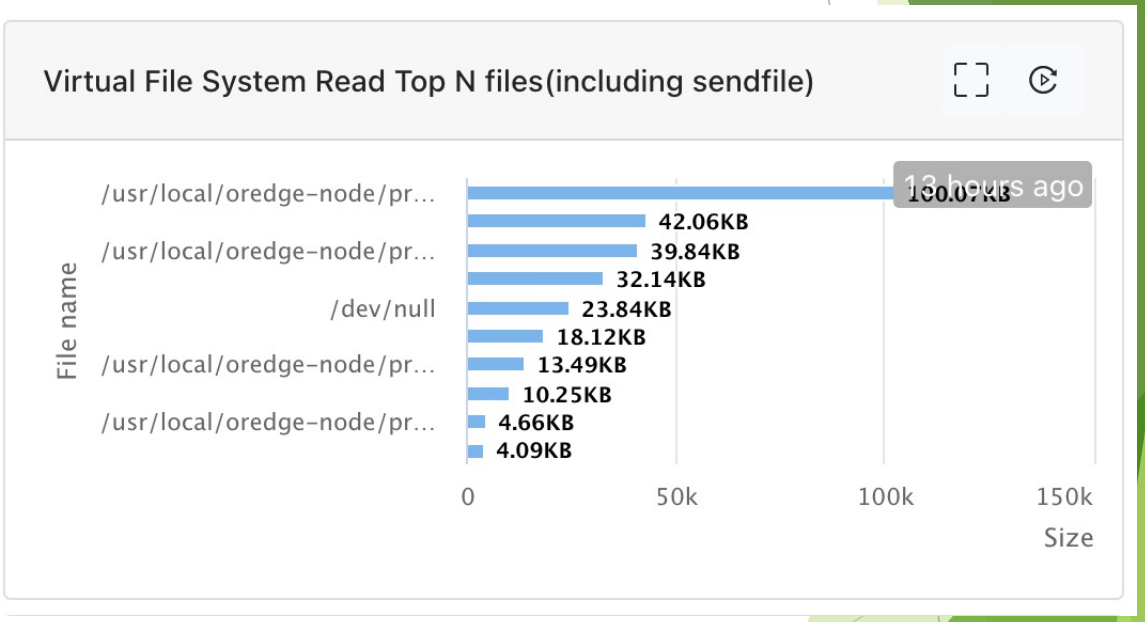
When the old and new nginx cycles coexist



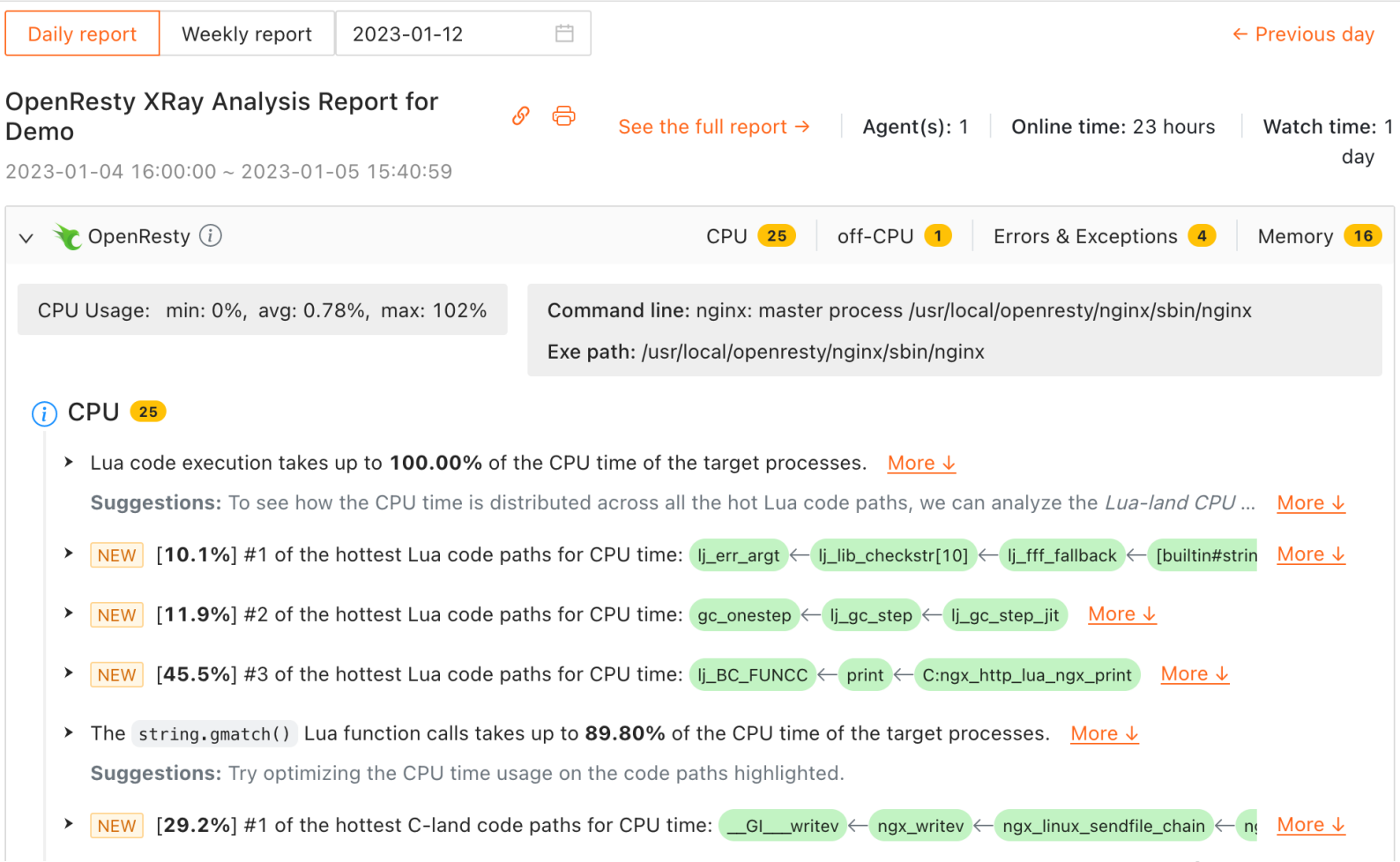

Nginx memory pool allocatons via the mmap syscall
For total 35.00 MB in 35 memory mappings with 49,464 chunks
1.00 MB
free
old cycle pool
other libc alloc
new cycle pool

OpenResty XRay
文件 IO 性能分析





OpenResty XRay
自动分析诊断报告

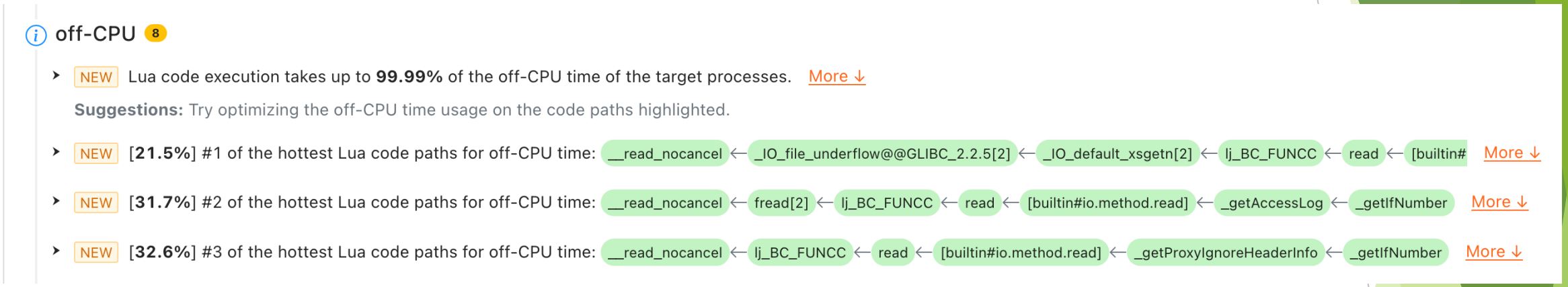


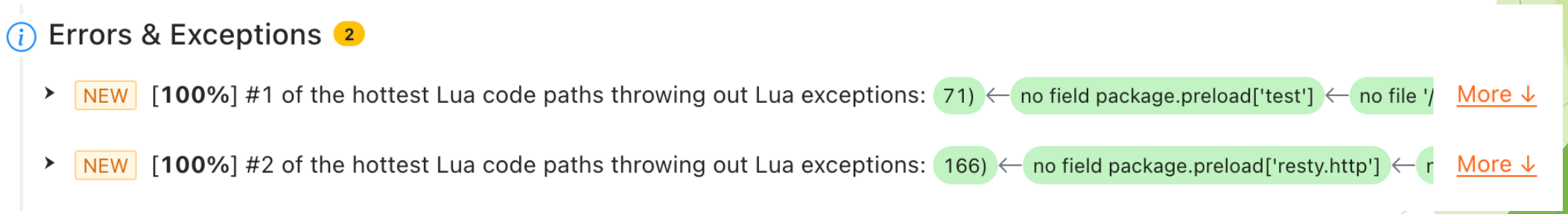
自动内存问题诊断报告

Memory 15

- NEW Glibc allocator takes up to **72.68 MB** in a target process. More ↓
- NEW Glibc arena takes up to **72.68 MB** in a target process. More ↓
- NEW In-use total memory in glibc arena takes up to **67.06 MB** in a target process. More ↓
- NEW Reserved free memory in glibc arena takes up to **5.67 MB** in a target process. More ↓
- NEW Free memory reserved by the LuaJIT allocator takes up to **12.19 MB** in a target process. More ↓
- NEW In-use total memory by the LuaJIT allocator takes up to **4.26 MB** in a target process. More ↓
- NEW Lua GC size of all types takes up to **1.45 MB** in a target process. More ↓
- NEW [**10.7%**] #1 of the hottest reference paths for LuaJIT GC object: table 0x7f950c0a0828 (584B 524.40KB) ← [light userdat More ↓
- NEW [**12.4%**] #2 of the hottest reference paths for LuaJIT GC object: trace 0x7f950a93a8e8 (2.36KB 161.13KB) ← next side ← More ↓

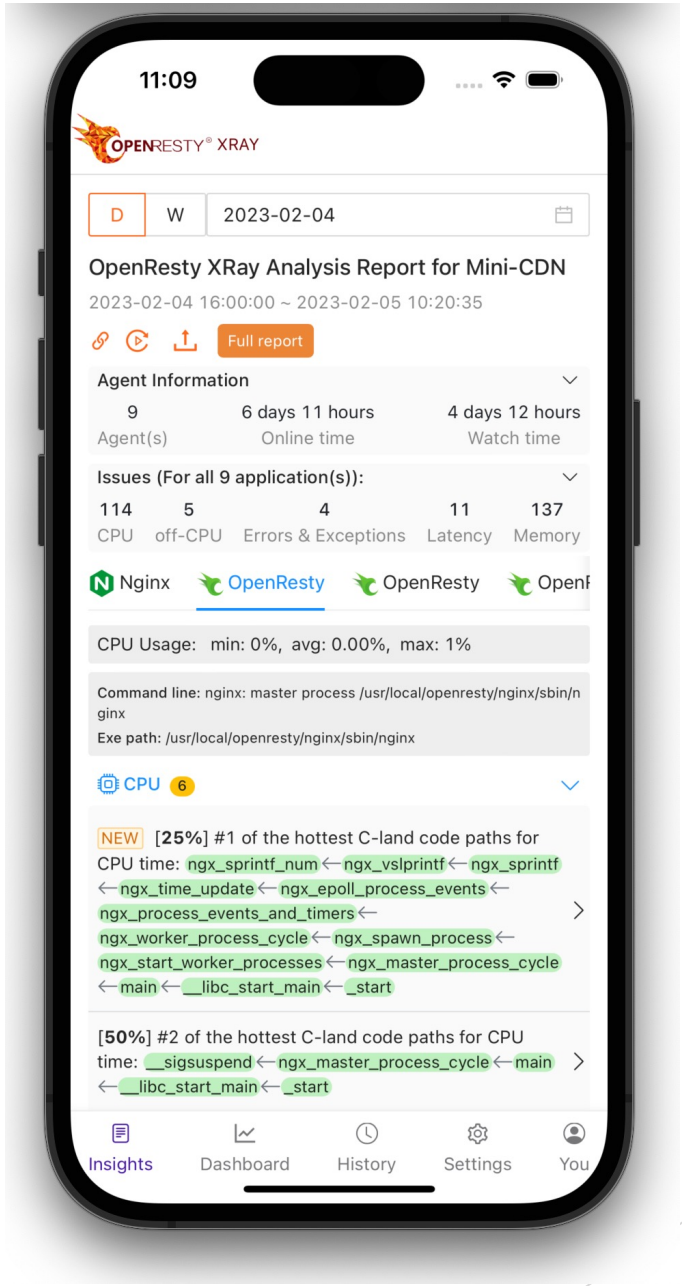
Off-CPU & 异常错误自动诊断报告





OpenResty XRay 移动端 App

- Android
- iOS



OpenResty XRay 并非只针对 OpenResty 应用

- Nginx, LuaJIT, OpenResty
- Python, PHP, Perl, Redis, PostgreSQL
- Go, Ruby, NodeJS, Java

调试符号

- OpenResty XRay 有一个中央包数据库，索引了几百 TB 的公开包的调试符号，仍在快速增长
- 目标机器无需安装或保存调试符号，只要调试符号曾被 OpenResty XRay 中央包数据库索引
- 无法找到调试符号或编译时未生成调试符号的程序，OpenResty XRay 将有能力自动重建调试符号（已有可工作的机器学习算法的原型实现）

感谢大家！

欢迎提问

中文 FAQ:
https://openresty.com.cn/cn/xray/faq/