OpenResty XRay 简介

对您的在线应用进行全天候、无死角的分析和诊断

试用 OpenResty XRay 产品

https://openresty.com.cn/cn/xray/

info@openresty.com

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

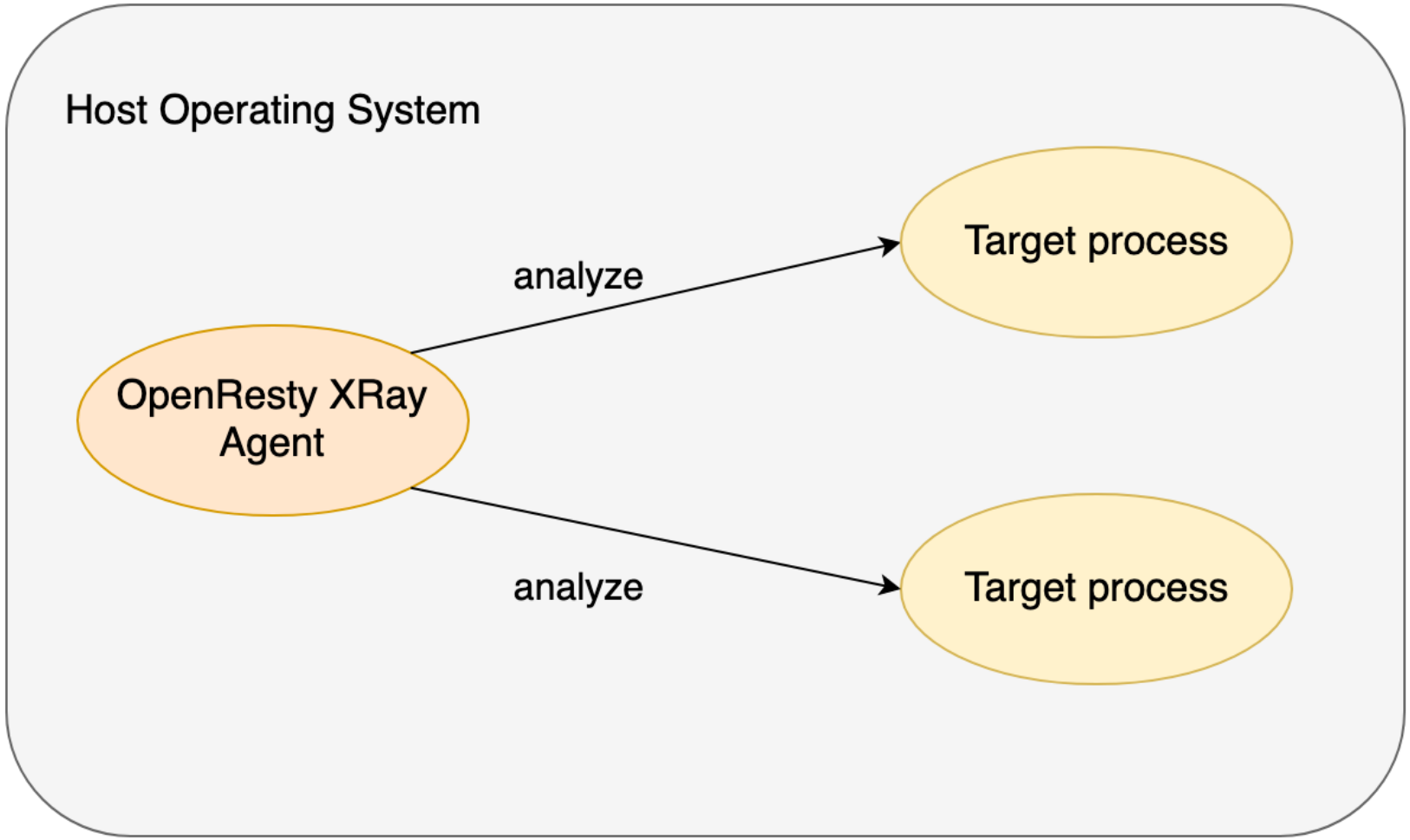
传统方法的缺点

- 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

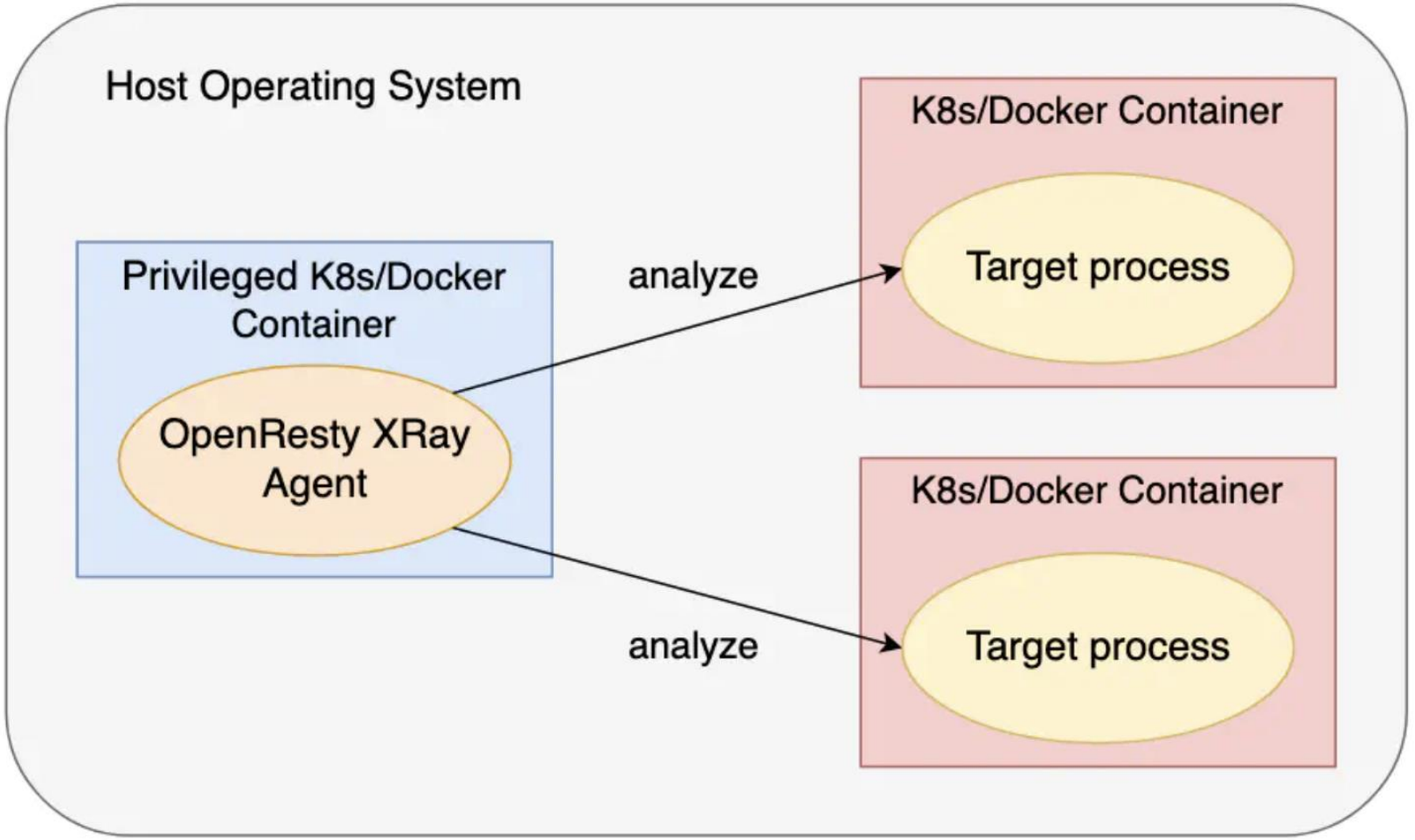
OpenResty XRay

- OpenResty XRay 是一款动态追踪产品
- 可以实时分析各种云和服务器应用程序
- 将运行中的进程和容器视为只读数据库，并提取必要的信息来解决性能问题、异常、错误和安全漏洞
- 拥有知识库、推理引擎和数百个高级分析器
- 可以在不改变或影响目标应用程序的情况下诊断和缩小深层问题的根源

OpenResty XRay 直接分析非容器应用进程



OpenResty XRay 穿透容器分析应用

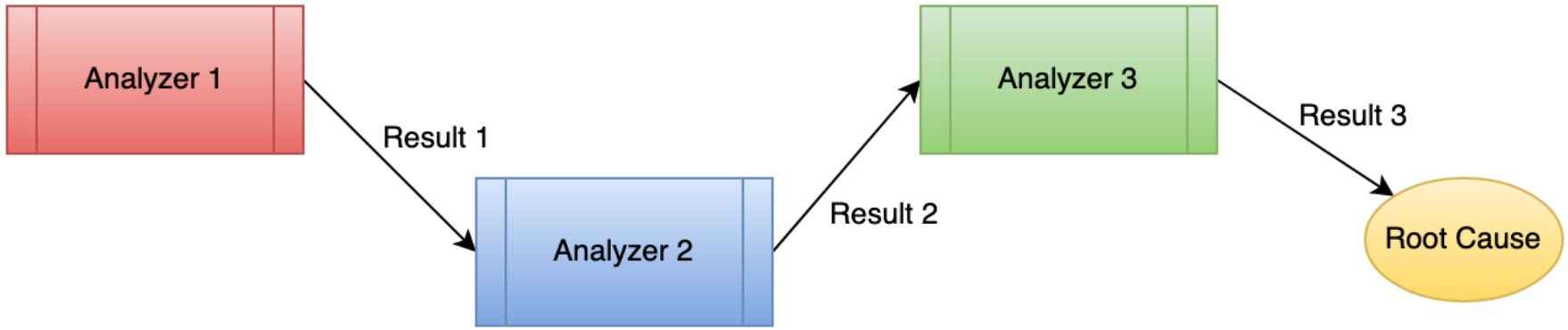


100% 非侵入式

- 无需修改您的应用
- 无需加装新的插件、模块或补丁到您的应用
- 无需注入任何代码到您的应用
- 无需重启您的应用进程
- 无需在您的应用中使用特殊的启动或编译选项
- 无需重建您现有的应用容器或应用软件包

OpenResty XRay 全自动采样
无人值守的使用模式

- 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

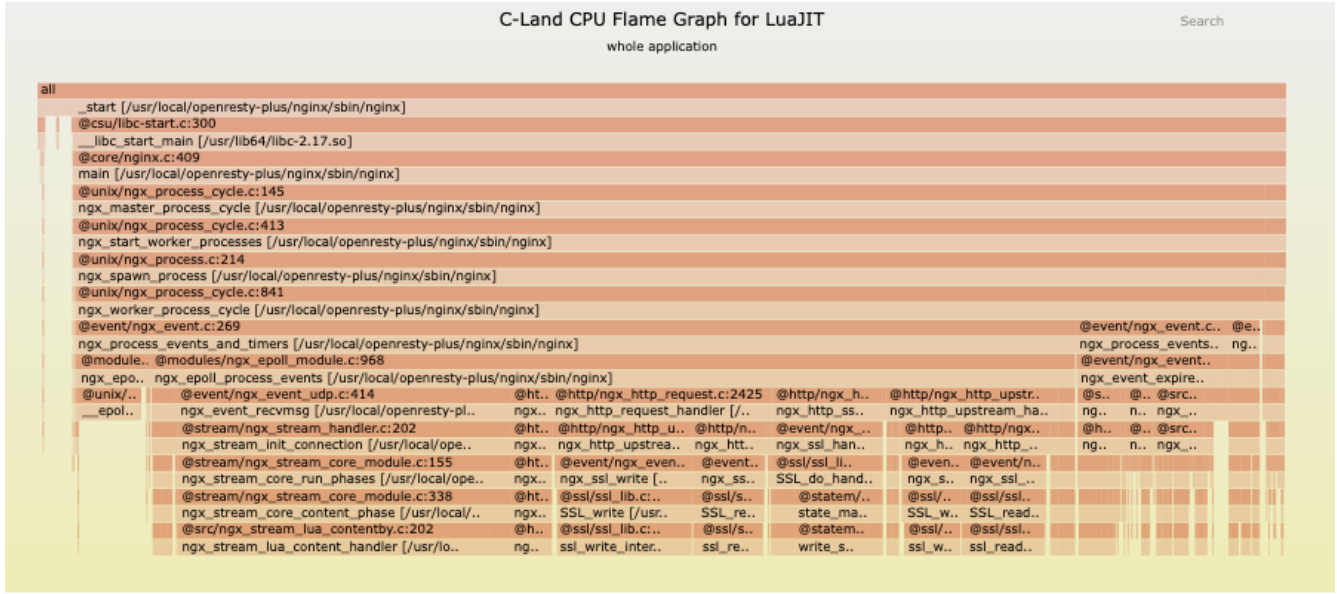


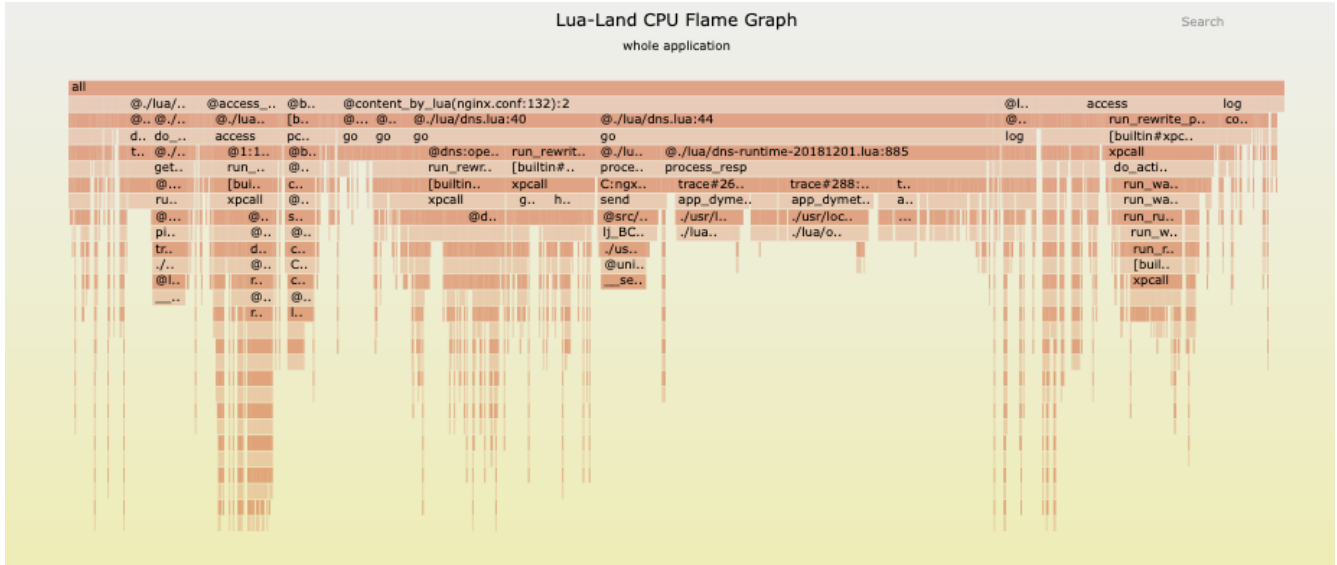
极低的性能损耗

- 未采样时的性能开销严格为 0
- 采样时的性能开销通常不易觉察

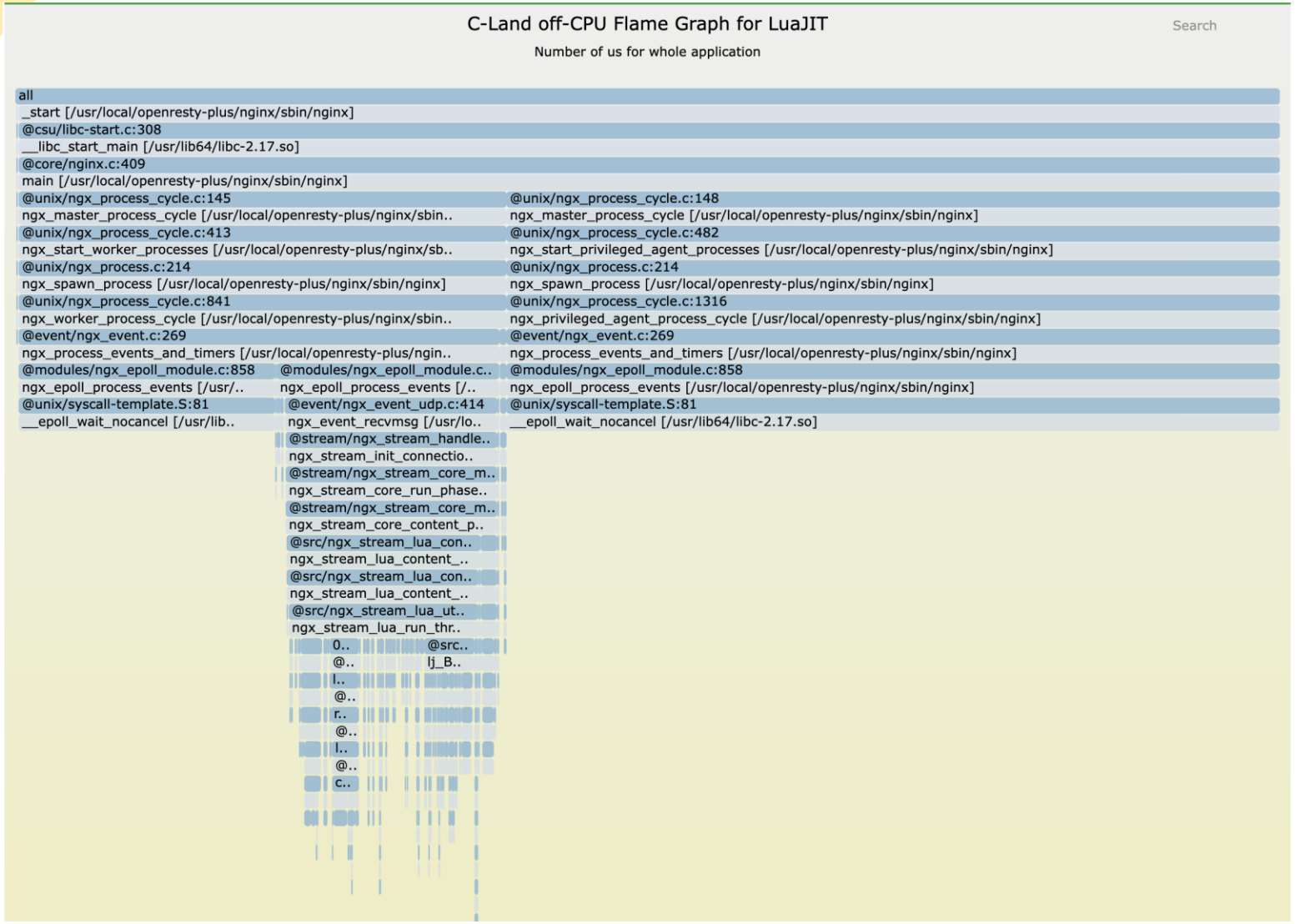
OpenResty XRay CPU 性能分析

- 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
Search
whole application
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]
@module..
@modules/ngx_epoll_module.c:968
ngx_epo..
ngx_epoll_process_events [/usr/local/openresty-plus/nginx/sbin/nginx]
@unix/..
__epol..
@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..
@ht..
ngx..
@http/ngx_http_request.c:2425
ngx_http_request_handler [/..
@http/ngx_http_u..
ngx_http_upstrea..
@event/ngx_even..
ngx_ssl_write [..
@ssl/ssl_lib.c:..
SSL_write [/usr..
@ssl/ssl_lib.c:..
ssl_write_inter..
@http/ngx_h..
ngx_htt..
@event/ngx_..
@ssl/ssl_li..
ngx_ss..
@ssl/s..
SSL_re..
@ssl/s..
ssl_re..
@http/ngx_http_upstr..
ngx_http_ss..
@http/n..
ngx_htt..
ngx_ssl_han..
@event/ngx_..
@ssl/ssl_han..
SSL_do_hand..
@statem/..
state_ma..
@statem..
write_s..
@http/ngx_http_upstream_ha..
@http..
ngx_h..
@even..
ngx_s..
@ssl/..
SSL_w..
@ssl/..
ssl_w..
@s..
@..
@src..
ng..
n..
ngx_..
@h..
@..
@src..
ng..
n..
ngx_..
@http/ngx..
ngx_http_..
@event/n..
ngx_ssl_..
@ssl/ssl..
@ssl/ssl..
SSL_read..
@ssl/ssl..
ssl_read..
@event/ngx_event.c..
@e..
ng..
ngx_process_events..
@event/ngx_event..
ngx_event_expire..

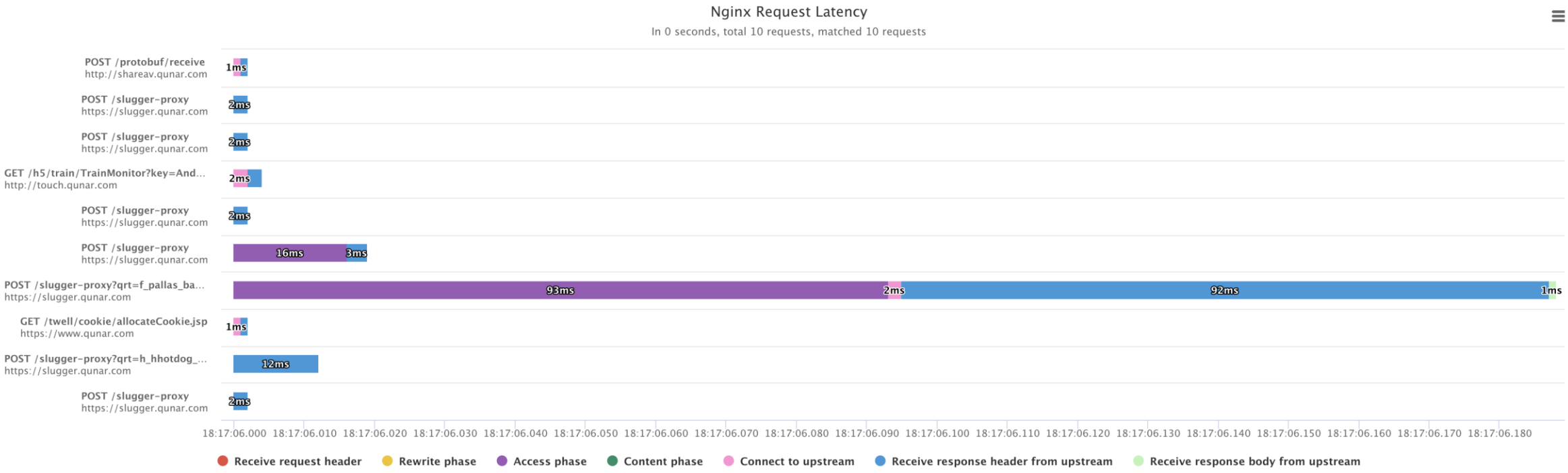

Lua-Land CPU Flame Graph
Search
whole application
all
@./lua/..
@access_..
@b..
@content_by_lua(nginx.conf:132):2
@l..
access
log
@..
@./..
@./lua..
[b..
@...
@..
@./lua/dns.lua:40
@./lua/dns.lua:44
run_rewrite_p..
co..
d..
do_..
access
pc..
go
go
go
go
log
[builtin#xpc..
t..
@./..
@1:1..
@b..
@dns:ope..
run_rewrit..
@./lu..
@./lua/dns-runtime-20181201.lua:885
xpcall
get..
run_..
@..
run_rewr..
[builtin#..
proce..
process_resp
do_acti..
@...
[bui..
c..
[builtin..
xpcall
C:ngx..
trace#26..
trace#288:..
t..
run_wa..
ru..
xpcall
@..
xpcall
g..
h..
send
app_dyme..
app_dymet..
a..
run_wa..
@...
@..
s..
@d..
@src/..
./usr/l..
./usr/loc..
...
run_ru..
pi..
@..
@..
lj_BC..
./lua..
./lua/o..
run_w..
tr..
d..
c..
./us..
run_r..
./..
@..
C..
@uni..
[buil..
@l..
r..
c..
__se..
xpcall
__..
@..
@..
r..
l..

OpenResty XRay CPU 阻塞（off-CPU）分析



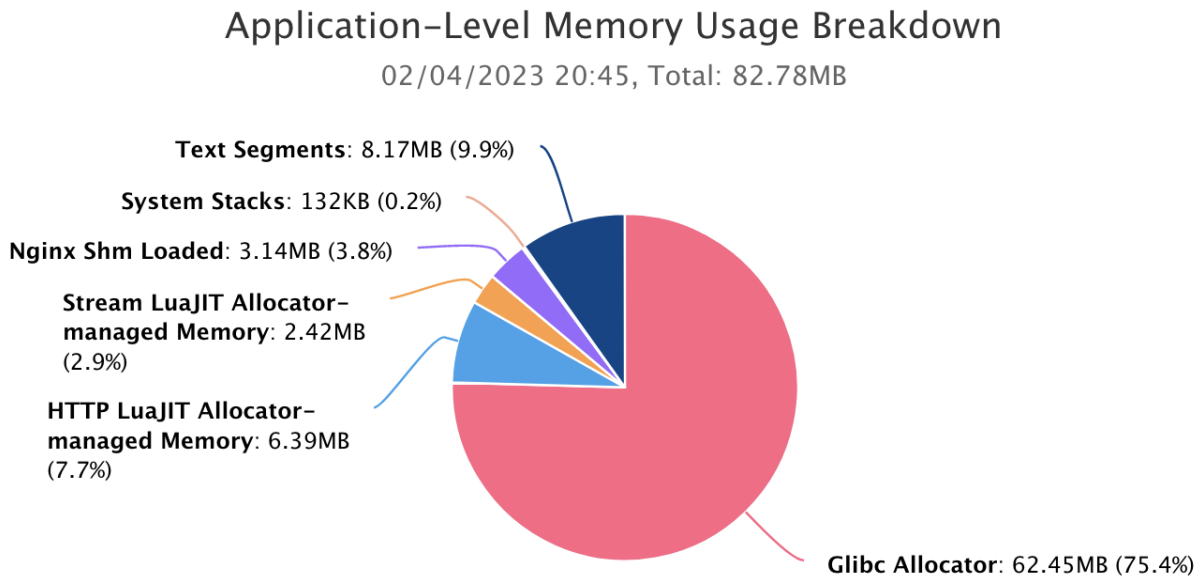
OpenResty XRay 请求延时分析

- 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）



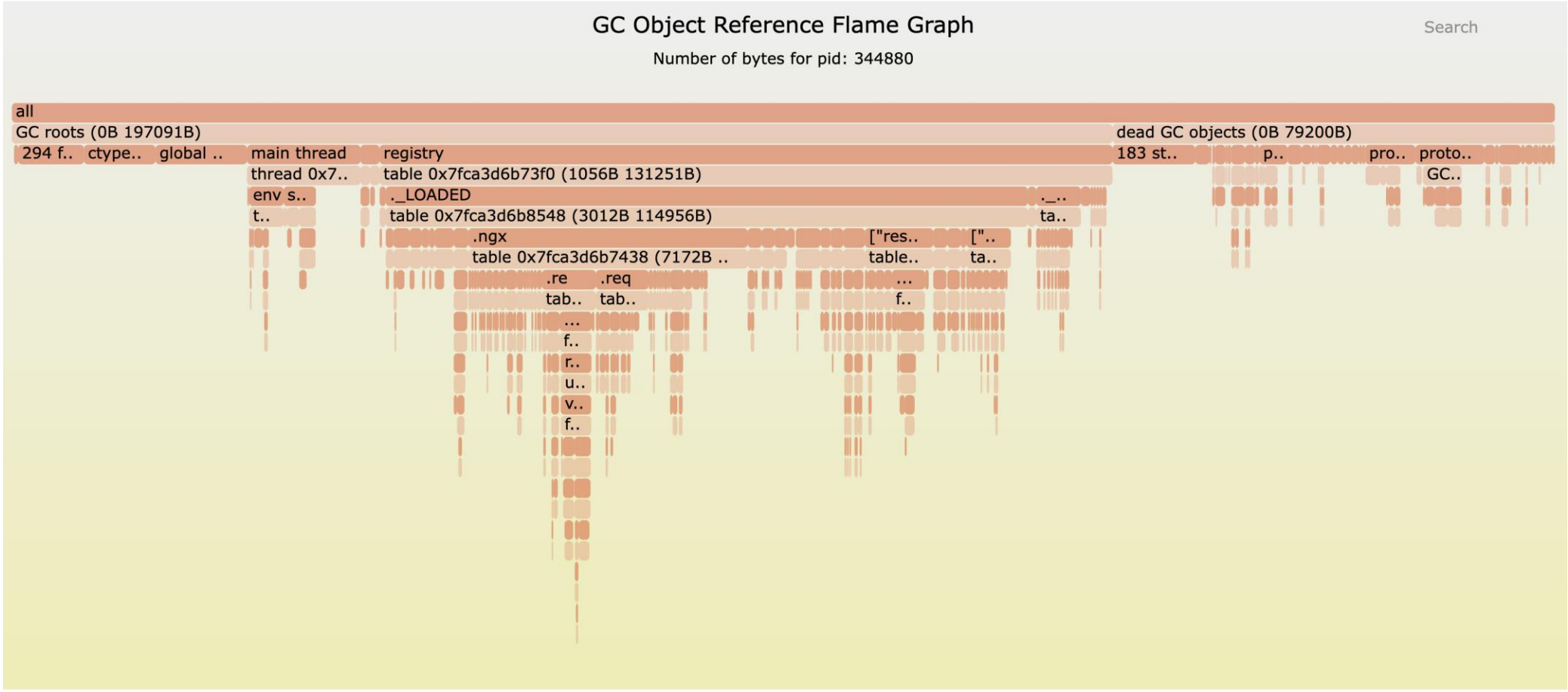
OpenResty XRay 内存使用分析

- 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？



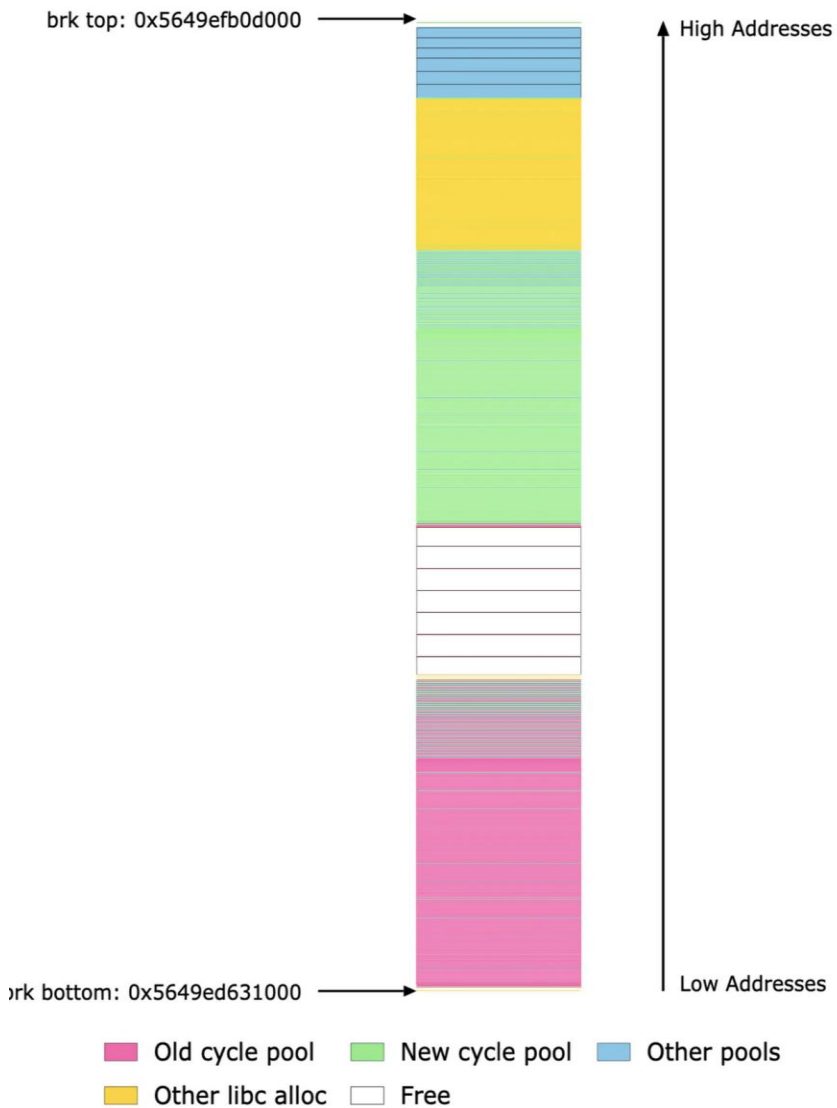
GC 对象引用关系火焰图

定量显示内存是如何在所有的对象引用路径上分布的



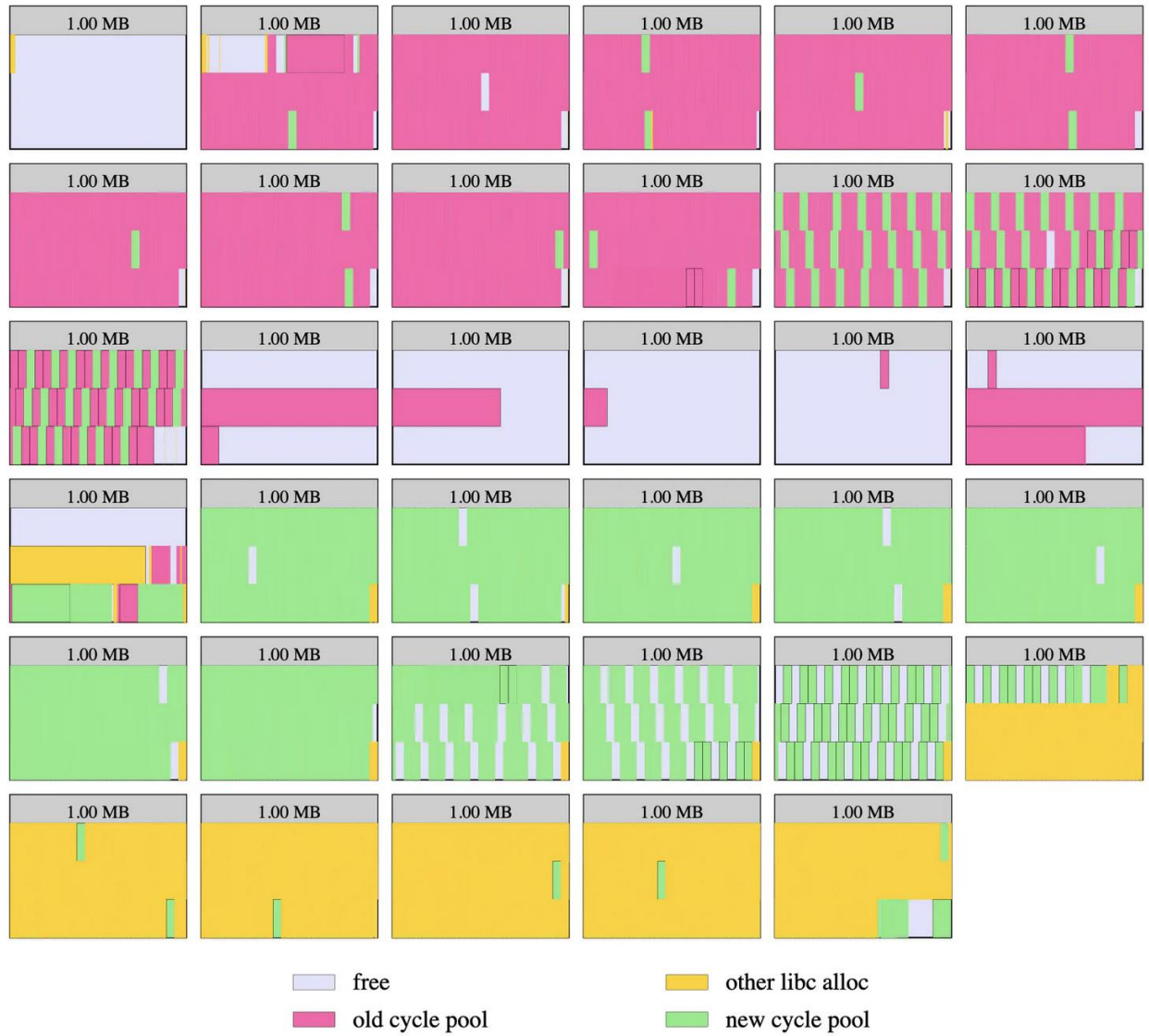
Nginx memory pool allocatons via the brk syscall

When the old and new nginx cycles coexist

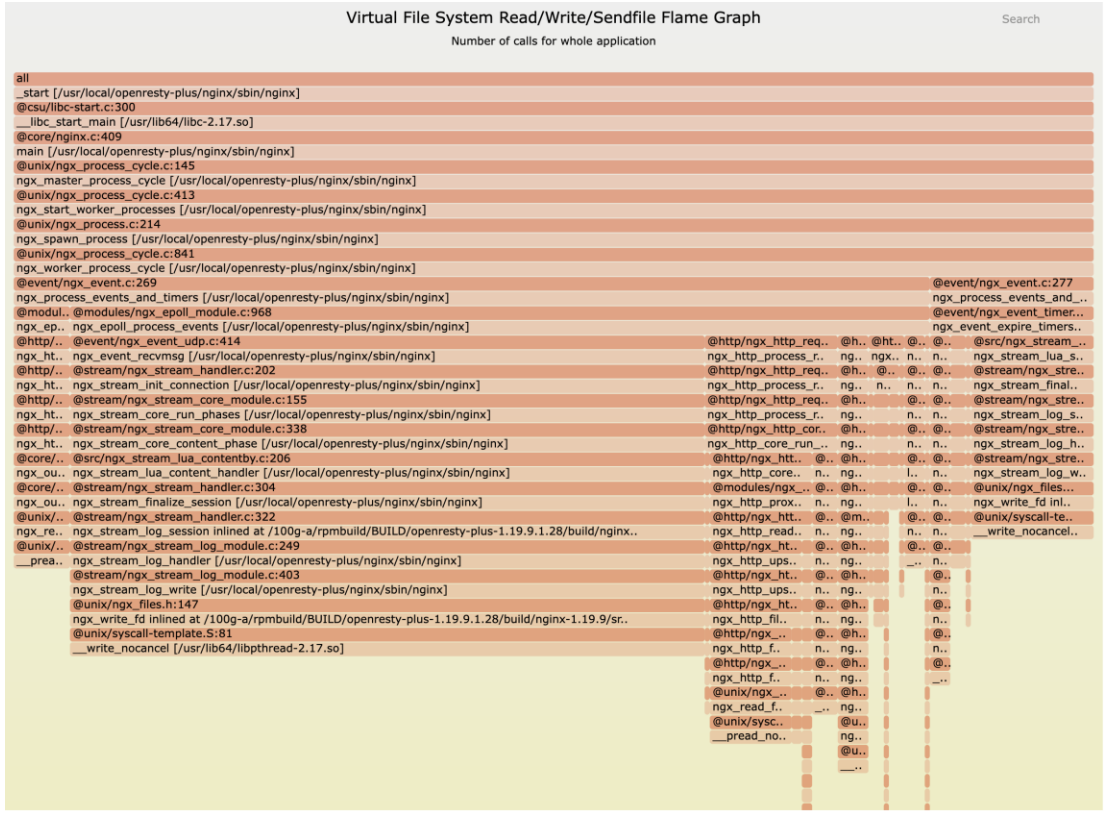


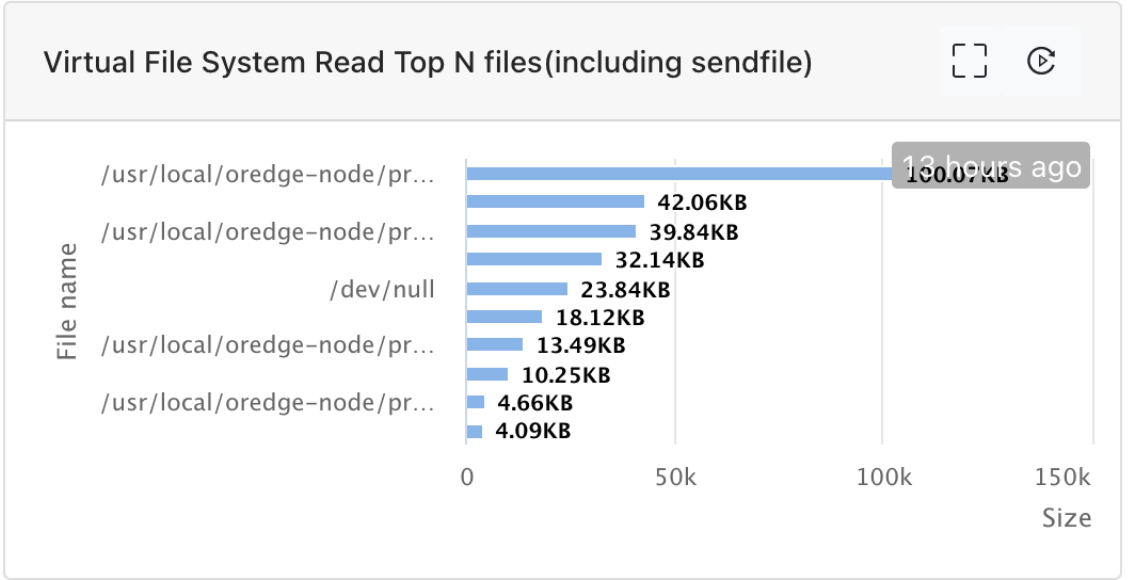
Nginx memory pool allocatons via the mmap syscall

For total 35.00 MB in 35 memory mappings with 49,464 chunks



OpenResty XRay 文件 IO 性能分析



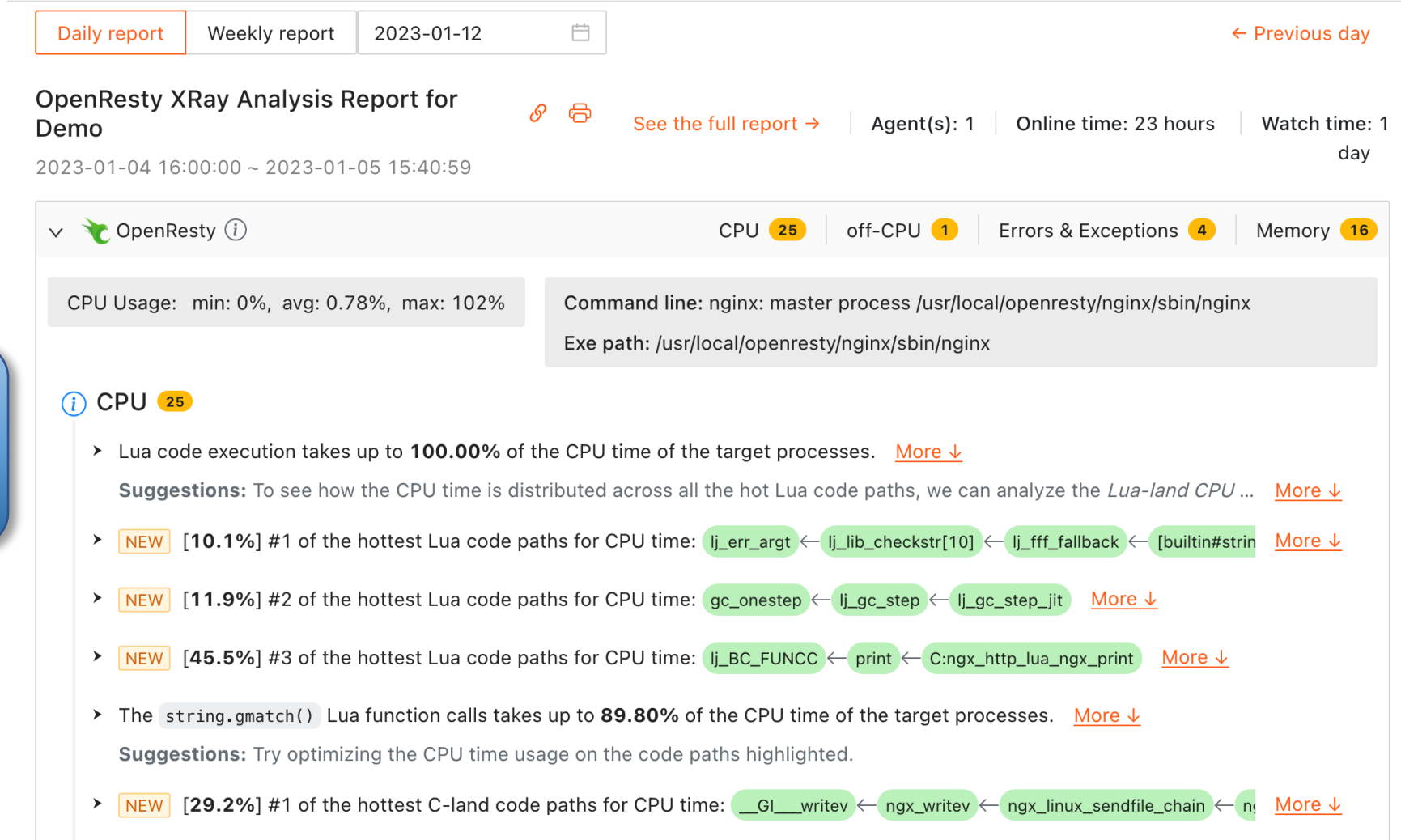


在线智能网络抓包

只抓真正有异常的网络连接上的包

OpenResty XRay 自动分析诊断报告

阅读自动分析和诊断报告的博客文章



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 ↓

自动延时分析与诊断

Latency 5

↑ 22.2% [**100%**] #1 of the hottest Lua code paths for Newly Created CoSocket: C:ngx_http_lua_socket_tcp_connect ← connect ← C:ngx_http_lua_socket_tcp_connect ← check_peer ← spawn_checker ← check_peers ← pcall ← [builtin#pcall]

See Job 4418885005 for more details.

Collapse ↑

[**100%**] #2 of the hottest Lua code paths for Newly Created CoSocket: C:ngx_http_lua_socket_tcp_connect ← connect ← C:ngx_http_lua_socket_tcp_connect ← _request ← request_admin ← pcall ← [builtin#pcall] More ↓

↑ 105.67 ms [**106.97 ms**] #1 of the hottest Lua code paths for Request Yield Latency: lua_yield ← lj_BC_FUNCC ← ngx_sleep ← C:ngx_http_lua_ngx_sleep ← limit_req_rate ← helper_1 ← xpcall ← [builtin#xpcall] ← run_rewrite_phase ← access ← access_by_lua(nginx.conf:586) More ↓

↑ 4.00 ms [**16.00 ms**] #2 of the hottest Lua code paths for Request Yield Latency: lua_yield ← ngx_stream_lua_socket_tcp_receive ← lj_BC_FUNCC ← receive ← C:ngx_stream_lua_socket_tcp_receive ← go ← content_by_lua(nginx.conf:140) More ↓

~~[**1.58 ms**] #2 of the hottest Lua code paths for Request Yield Latency: lua_yield ← lj_BC_FUNCC ← ngx_sleep ← C:ngx_stream_lua_ngx_sleep ← limit_req ← process_req ← go ← content_by_lua(nginx.conf:132)~~ More ↓

off-CPU 自动诊断报告

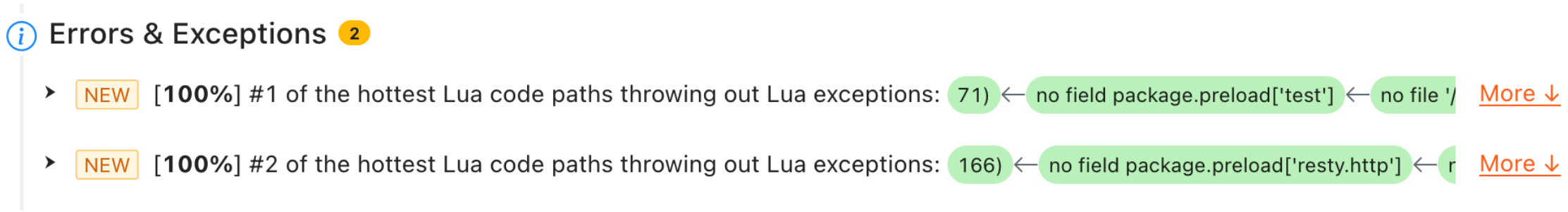
off-CPU 8

- NEW Lua code execution takes up to **99.99%** of the off-CPU time of the target processes. More ↓

  Suggestions: Try optimizing the off-CPU time usage on the code paths highlighted.

- NEW [**21.5%**] #1 of the hottest Lua code paths for off-CPU time: __read_nocancel ← _IO_file_underflow@@GLIBC_2.2.5[2] ← _IO_default_xsgetn[2] ← lj_BC_FUNCC ← read ← [builtin# More ↓
- NEW [**31.7%**] #2 of the hottest Lua code paths for off-CPU time: __read_nocancel ← fread[2] ← lj_BC_FUNCC ← read ← [builtin#io.method.read] ← _getAccessLog ← _getIfNumber More ↓
- NEW [**32.6%**] #3 of the hottest Lua code paths for off-CPU time: __read_nocancel ← lj_BC_FUNCC ← read ← [builtin#io.method.read] ← _getProxyIgnoreHeaderInfo ← _getIfNumber More ↓

异常错误自动诊断报告


Errors & Exceptions 2
NEW [100%] #1 of the hottest Lua code paths throwing out Lua exceptions: 71) ← no field package.preload['test'] ← no file '/ More ↓
NEW [100%] #2 of the hottest Lua code paths throwing out Lua exceptions: 166) ← no field package.preload['resty.http'] ← r More ↓

安全问题自动分析

- 自动检查和报告未开启 TLS 加密的连接
- 动态扫描未开启证书来源校验的 TLS 连接
- 检查使用了非安全的 SSL 协议版本
- 扫描远程 shell 命令执行的事件与代码上下文

Core Dump 进程遗骸分析（进程崩溃）

All | Core Dump Analysis

Application Type * OpenResty

Analyzer * openresty-core-dump-analysis (OpenResty Core Dump Analysis)

Core File * /tmp/core.3859164

Executable File Path * /usr/local/openresty/nginx/sbin/nginx

▶ Advanced Settings

Core File Meta Data

File Name: /tmp/core.3859164

Executable File Path: /usr/local/openresty/nginx/sbin/nginx

Size: 14.72MB

Analyze

从 Core Dump 文件提取深层信息

Analysis 7173425

(gdb) lbt
C:ngx_md5_body
trace#1:access.lua:4
check_token
@/usr/local/openresty/lualib/access.lua:3
auth
@/usr/local/openresty/lualib/access.lua:21
@access_by_lua(nginx.conf:51):2

(gdb) full_lbt
C:ngx_md5_body
trace#1:access.lua:4
check_token
@/usr/local/openresty/lualib/access.lua:3
auth
@/usr/local/openresty/lualib/access.lua:21
headers = "access"
token = "hello"
@access_by_lua(nginx.conf:51):2

Compiler Output | Analyzer Output | Graphs

(gdb) ngx_process_info
parent: 3859163
process: worker 0

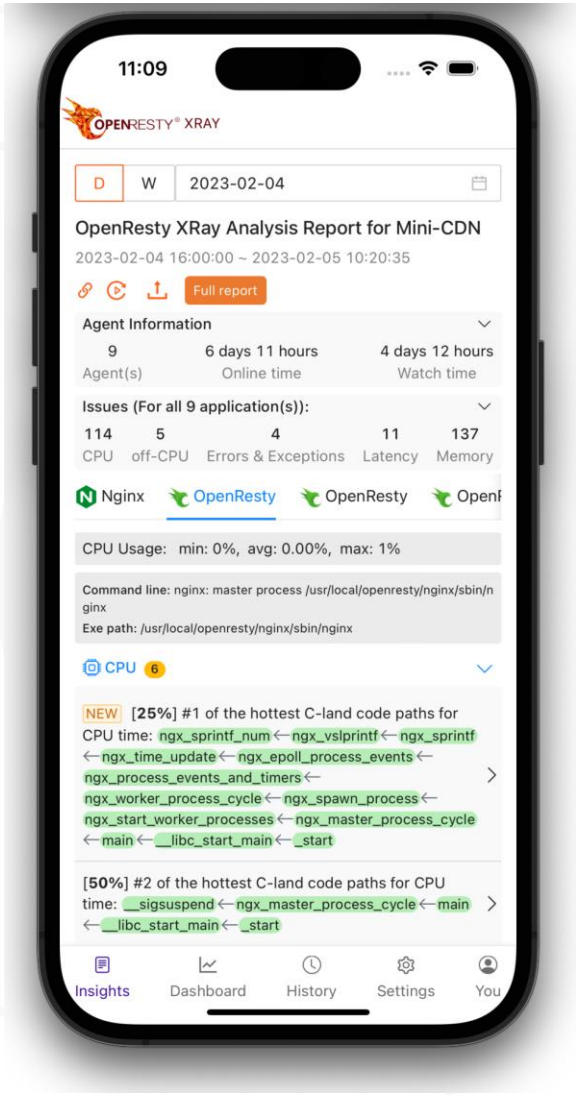
(gdb) cur_http_req
current phase: access
schema: http, req_size: 52, resp_size: 0GET / HTTP/1.1
Host: localhost:80
TOKEN: hello

(gdb) ubt
0x42e958 ngx_md5_body [/usr/src/debug/openresty-1.21.4.1/build/nginx-1.21.4/src/core/ngx_md5.c:199]
0x42f1be ngx_md5_final [/usr/src/debug/openresty-1.21.4.1/build/nginx-1.21.4/src/core/ngx_md5.c:91]
0x4ea997 ngx_http_lua_ffi_md5 [/usr/src/debug/openresty-1.21.4.1/build/nginx-1.21.4/../ngx_lua-0.10.21/s
7f763447ffd3: []
0x4f86a1 ngx_http_lua_run_thread [/usr/src/debug/openresty-1.21.4.1/build/nginx-1.21.4/../ngx_lua-0.10.2

OpenResty XRay 移动端 App

任何时候从任何地方察看您的在线应用

- Android (Google Play)
- iOS (苹果商店)



OpenResty XRay
并非只针对 OpenResty 应用

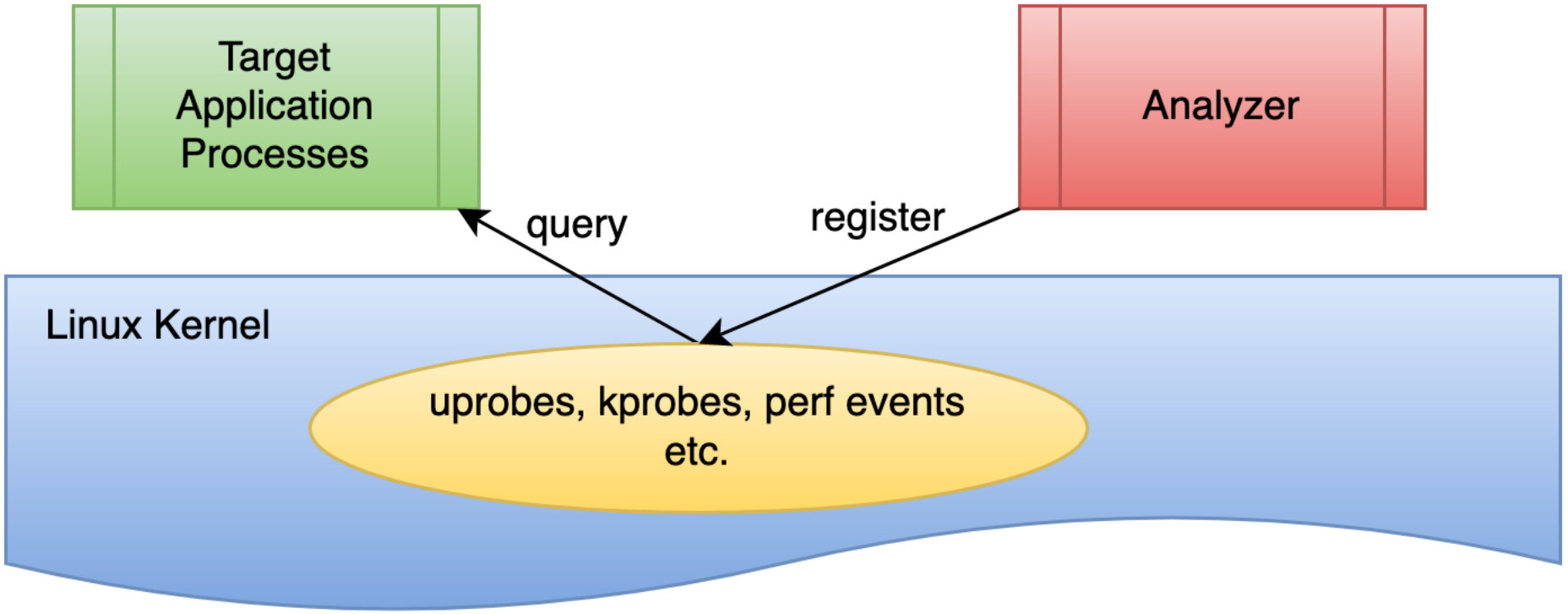
- Nginx，LuaJIT，OpenResty，Python，PHP，Go，Erlang，Perl，Envoy，Ruby，Redis，Rust，Kong
- 初步支持：
  PostgreSQL
- 即将发布：
  NodeJS, Java

支持绝大多数主流 Linux 发行版和容器布署方式

Ubuntu

Debian

OpenResty XRay 基于先进的动态追踪技术



动态追踪的优点

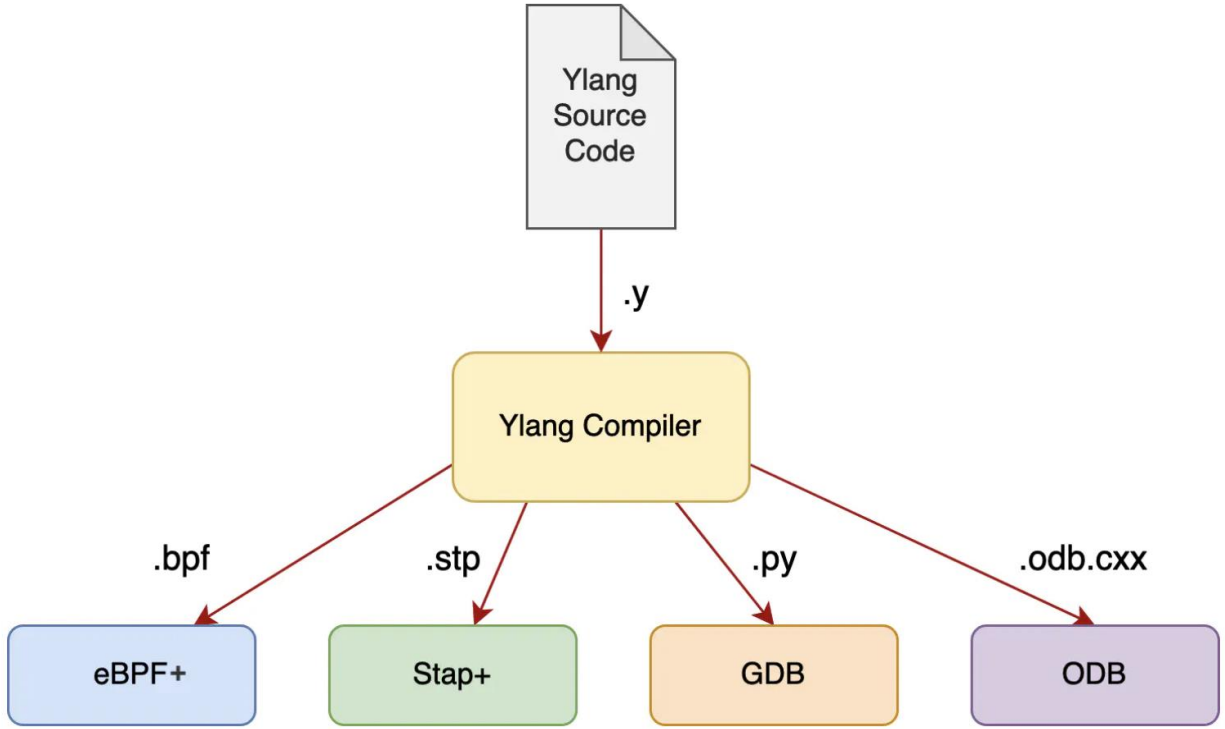
- 非侵入式，无需修改应用
- 热插拔，通常无需应用配合（很多开源动态追踪工具有时还是要求应用配合）
- 开销通常较低，可以在数据源进行聚合汇总
- 事发后的在线实时调试能力
- 全技术栈分析无死角
- 按需采样
- 不采样时严格 0 损耗

OpenResty XRay 的全新一代动态追踪技术

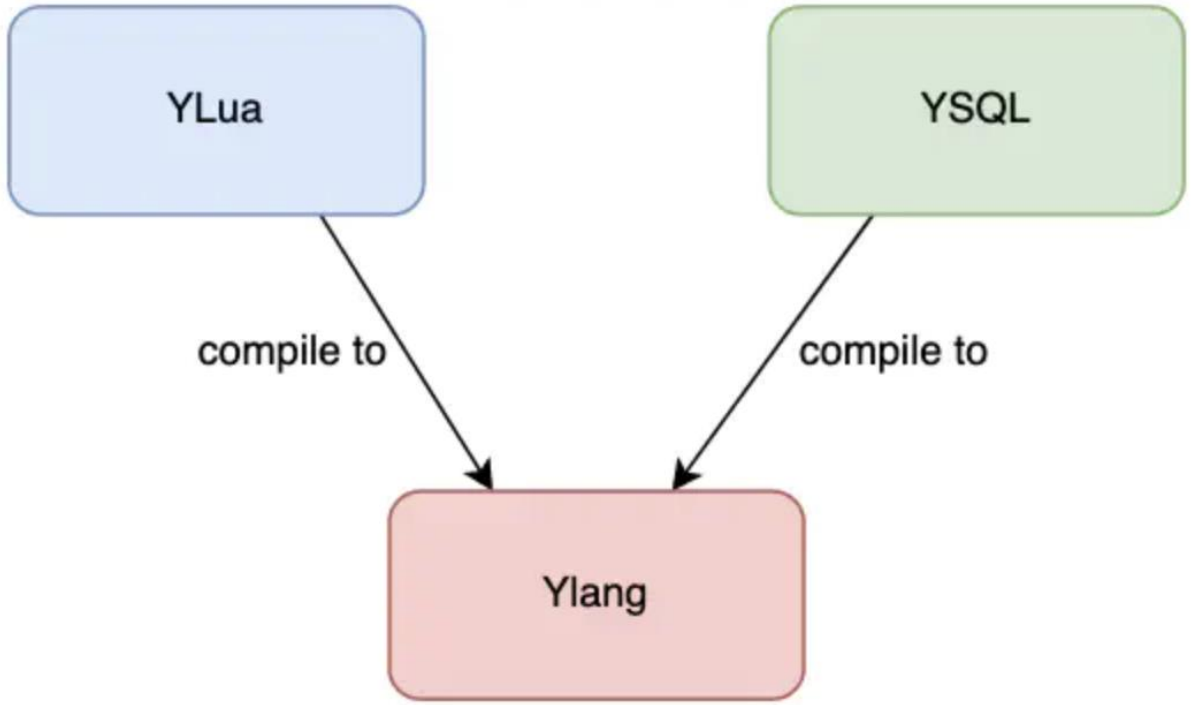
- Y 语言（Ylang）编译器（支持 GNU C 和标准 C 语言的绝大部分语法）
- Ylua 语言
- YSQL 语言
- Stap+ 显著改进了 SystemTap
- eBPF+ 显著改进了 eBPF（同时 LLVM+ 也大大改进了开源 LLVM)
- ODB 是超轻量版的 GDB
- Ylang 编译器也能生成高度优化的 GDB 的 Python 扩展代码
- 针对线上生产环境的严格性能损耗控制

“一次编写，到处运行”

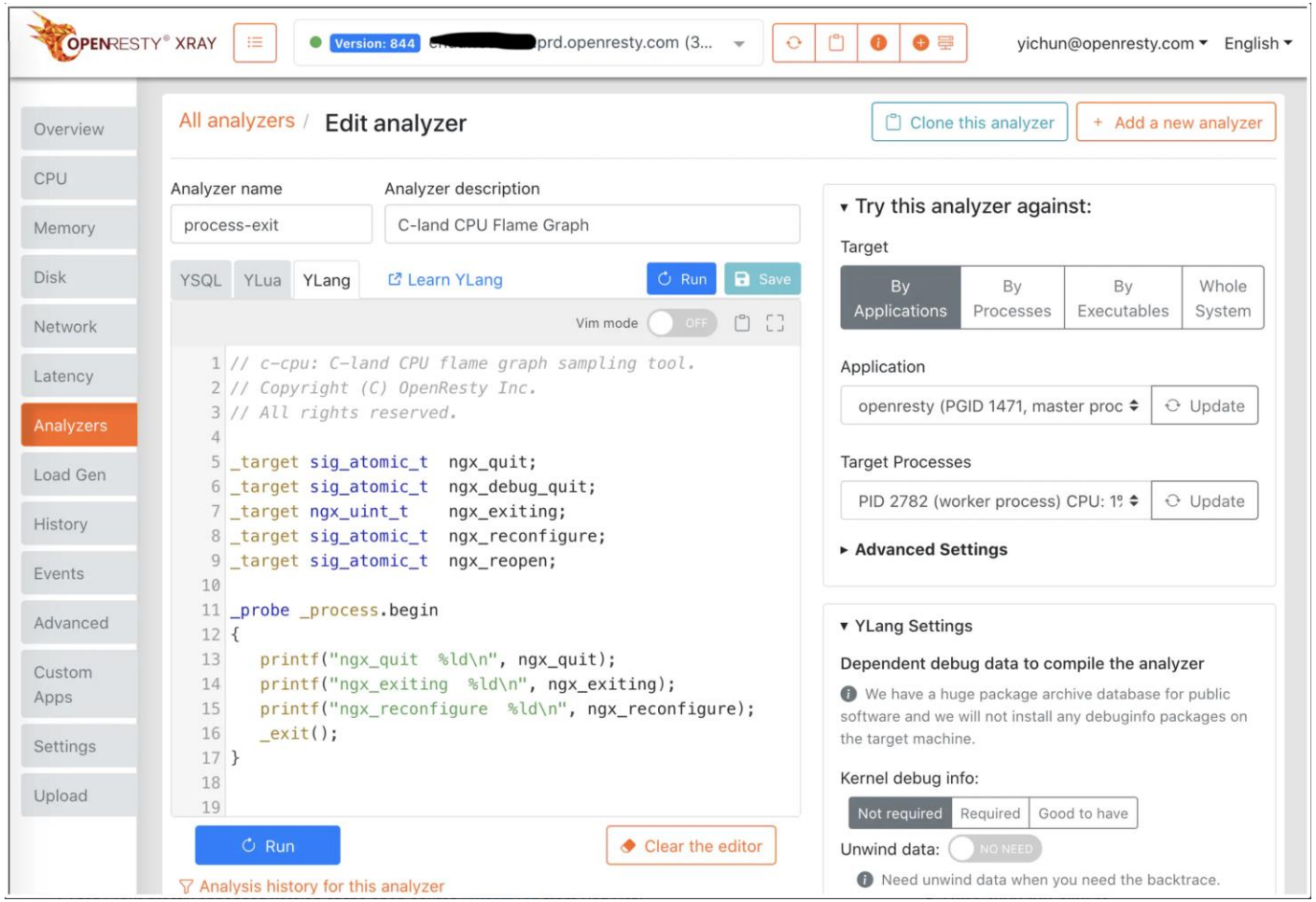
从 Y 语言代码到各种不同运行时的代码



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

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）

为众多企业客户所信任


StackPath


Microsoft


bilibili


aftership


ADP


Qunar.Com


zoom

了解更多

OpenResty XRay 常见问答

访问 OpenResty XRay 官方博客

访问 OpenResty XRay 官方主页

试用 OpenResty XRay 产品