DeepFlow

eBPF 零侵扰分布式追踪进展和探索

向阳 @ 云杉网络

2024-03-10

DeepFlow

零侵扰分布式追踪进展和探索

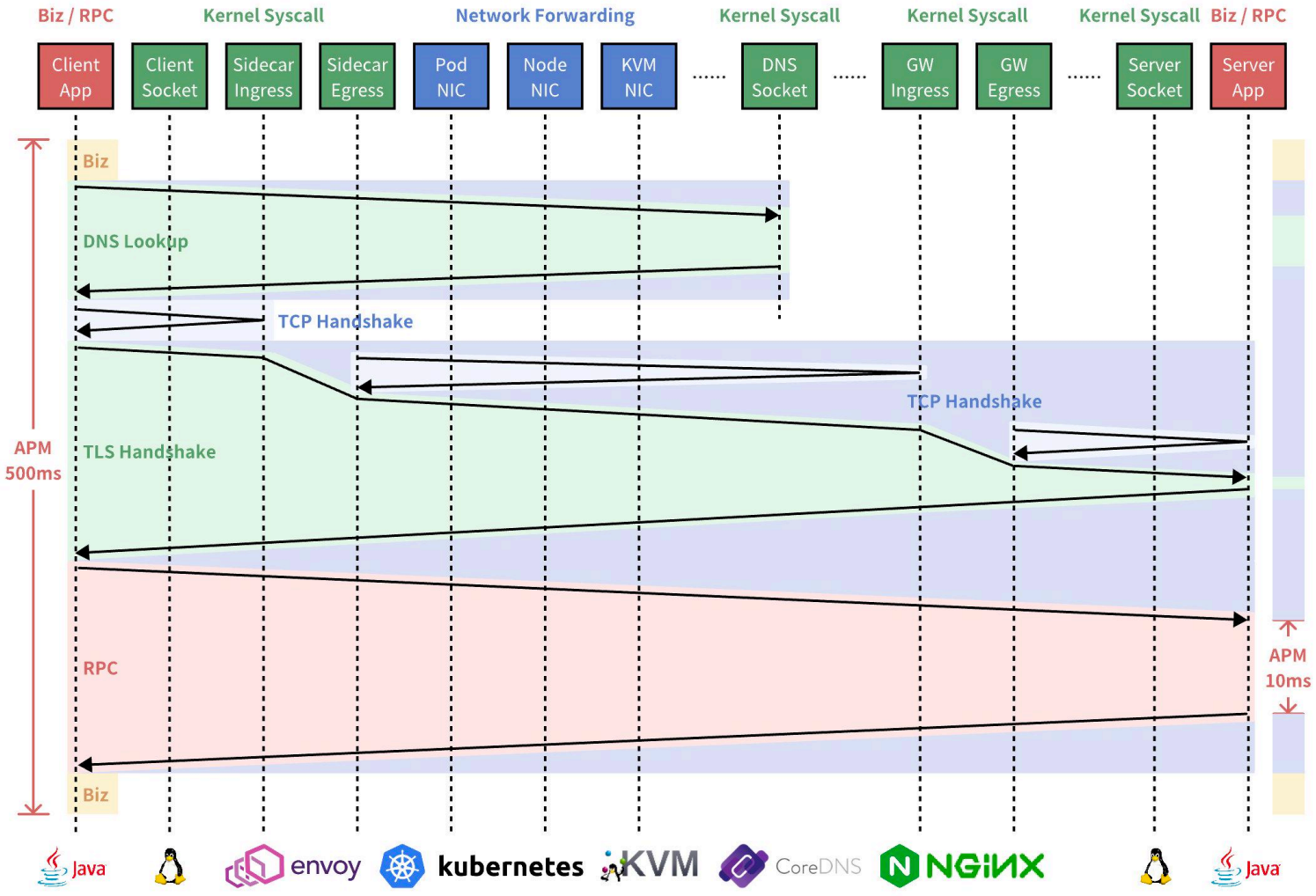
背景

进展

案例

展望

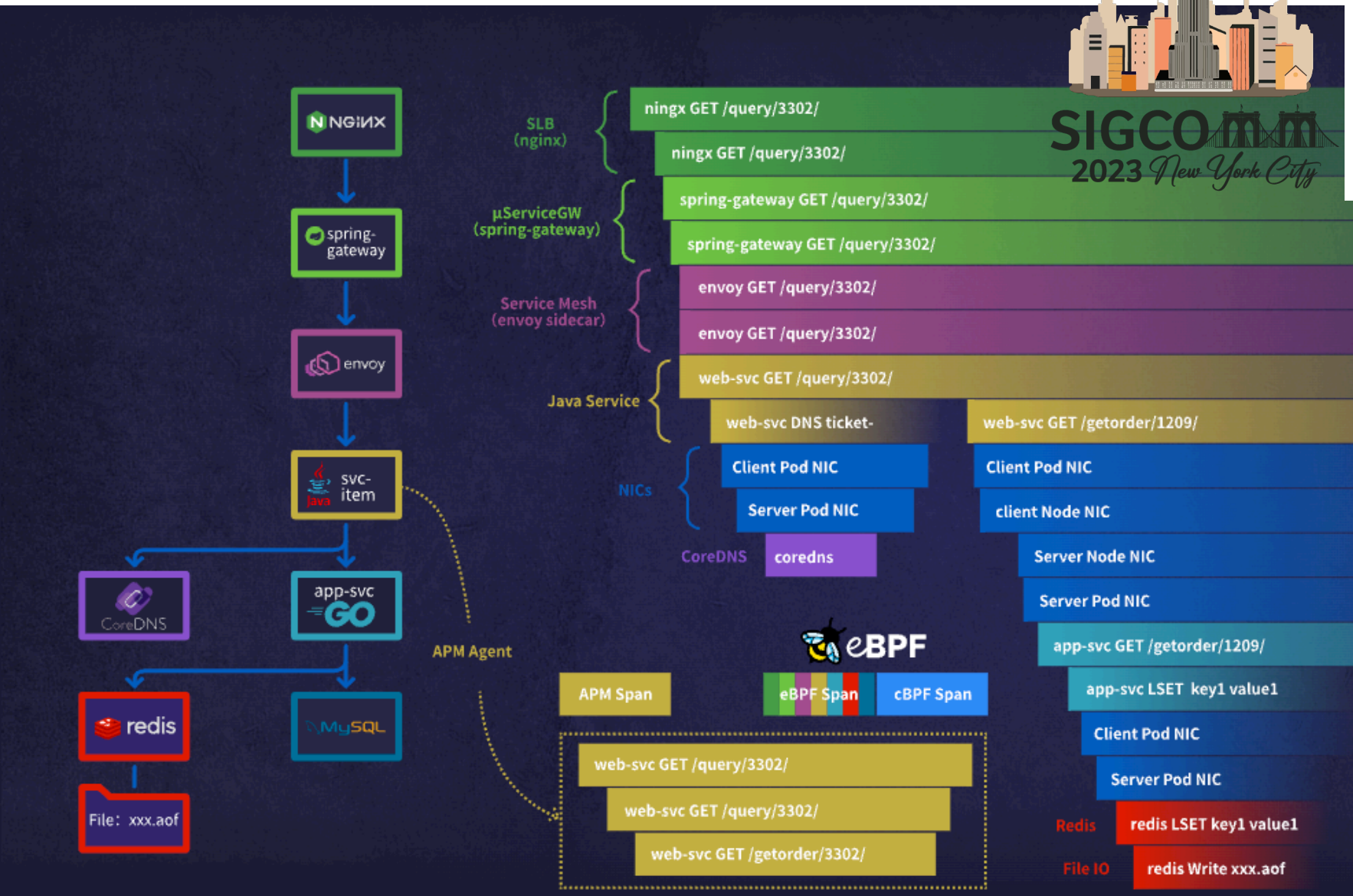
分布式追踪在云原生时代：插桩越来越难、定界越来越慢



GO Rust node.js python

DALL.E

DeepFlow 中的零侵扰分布式追踪



SIGCOMM 2023 New York City

Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code

Junxian Shen[1,2]°, Han Zhang[1,4]•, Yang Xiang[2]⋆, Xingang Shi[1,4], Xinrui Li[3], Yunxi Shen[3], Zijian Zhang[1,2], Yongxiang Wu[3], Xia Yin[3,4], Jilong Wang[1,4], Mingwei Xu[1,4], Yahui Li[1], Jiping Yin[2], Jianchang Song[2], Zhuofeng Li[2], Runjie Nie[2]

[1]Institute for Network Sciences and Cyberspace, Tsinghua University
[2]Yunshan Networks
[3]Department of Computer Science and Technology, Tsinghua University
[4]Zhongguancun Laboratory

°shenjx22@mails.tsinghua.edu.cn •zhhan@tsinghua.edu.cn ⋆xiangyang@yunshan.net

ABSTRACT

Microservices are becoming more complicated, posing new challenges for traditional performance monitoring solutions. On the one hand, the rapid evolution of microservices places a significant

Mingwei Xu[1,4], Yahui Li[1], Jiping Yin[2], Jianchang Song[2], Zhuofeng Li[2], Runjie Nie[2]. 2023. Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3603269.3604823

DeepFlow

零侵扰分布式追踪进展和探索

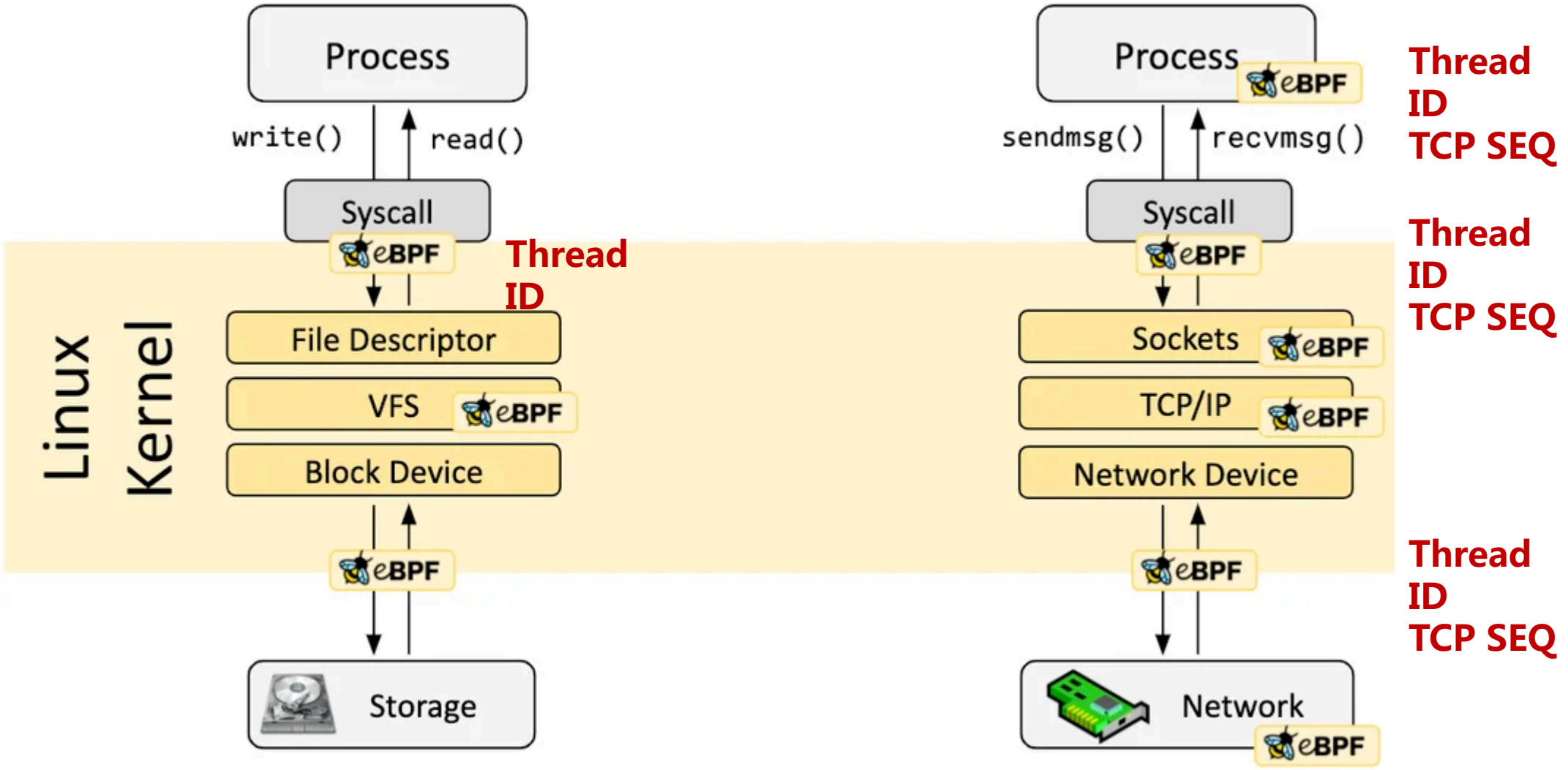
背景

进展

案例

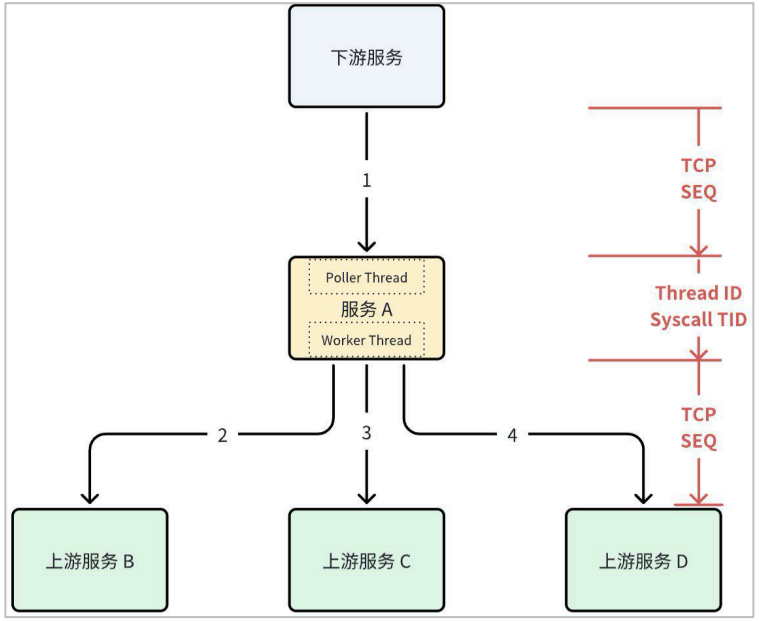
展望

Initial Insight：eBPF 感知 Thread ID 和 TCP SEQ



https://ebpf.io/what-is-ebpf/

挑战一：非阻塞 IO（NIO）：eBPF syscall_trace_id



Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code

Junxian Shen[1 2 °], Han Zhang[1 4 •], Yang Xiang[2 ⋆], Xingang Shi[1 4], Xinrui Li[3], Yunxi Shen[3], Zijian Zhang[1 2], Yongxiang Wu[3], Xia Yin[3 4], Jilong Wang[1 4], Mingwei Xu[1 4], Yahui Li[1], Jiping Yin[2], Jianchang Song[2], Zhuofeng Li[2], Runjie Nie[2]

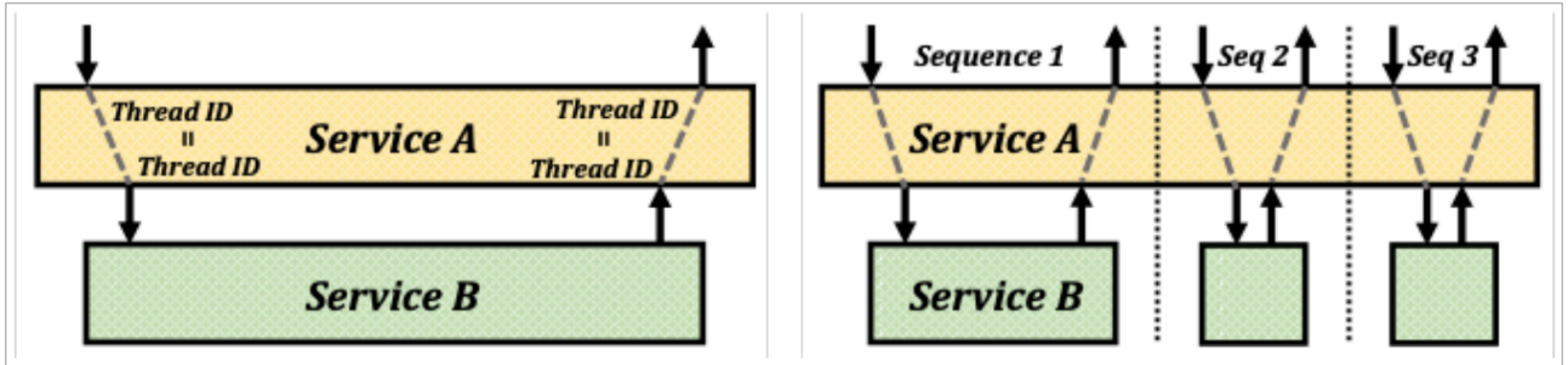
[1]Institute for Network Sciences and Cyberspace, Tsinghua University
[2]Yunshan Networks
[3]Department of Computer Science and Technology, Tsinghua University
[4]Zhongguancun Laboratory

°shenjx22@mails.tsinghua.edu.cn •zhhan@tsinghua.edu.cn ⋆xiangyang@yunshan.net

ABSTRACT

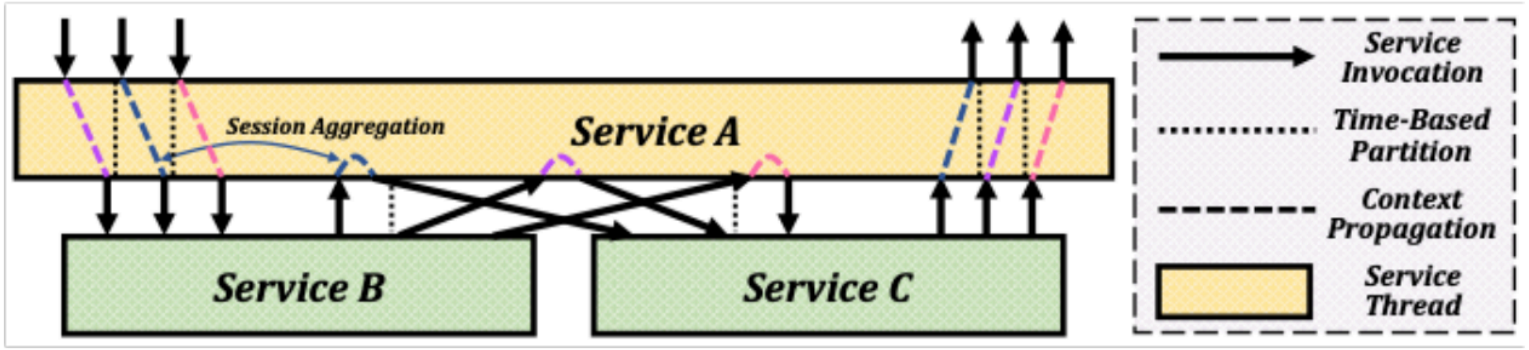
Microservices are becoming more complicated, posing new challenges for traditional performance monitoring solutions. On the one hand, the rapid evolution of microservices places a significant

Mingwei Xu[1 4], Yahui Li[1], Jiping Yin[2], Jianchang Song[2], Zhuofeng Li[2], Runjie Nie[2]. 2023. Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3603269.3604823



(a) Implicit context propagation via thread IDs.

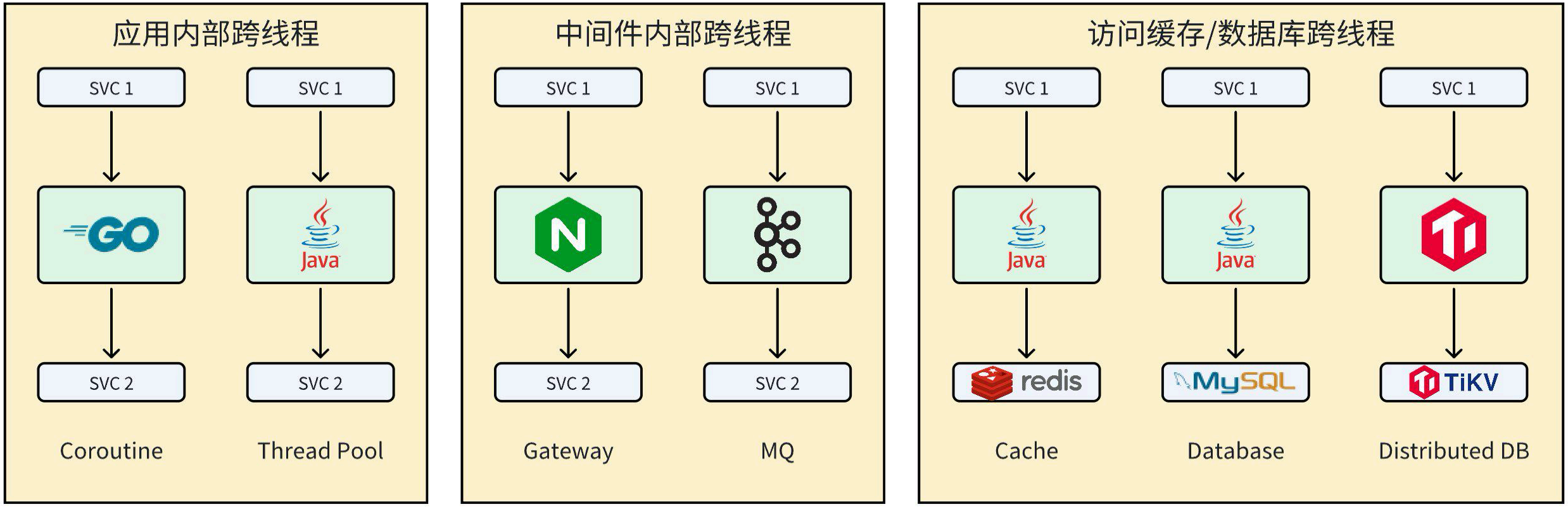
(b) Using time sequences to partition thread-reusing spans.



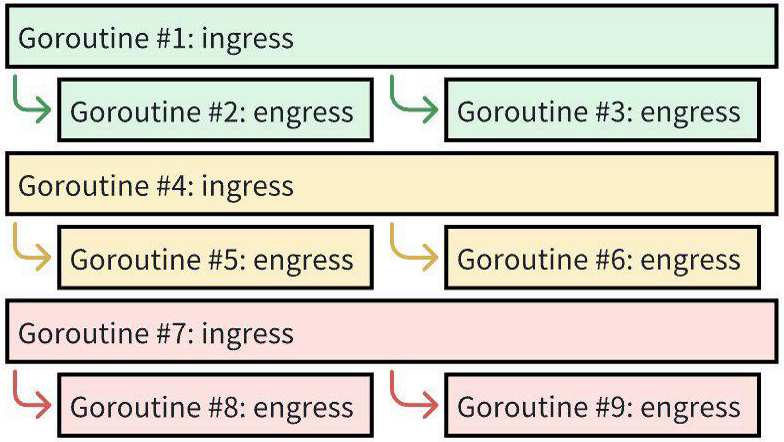
(c) Association for multiple requests or responses.

Figure 7: Intra-component causal association.

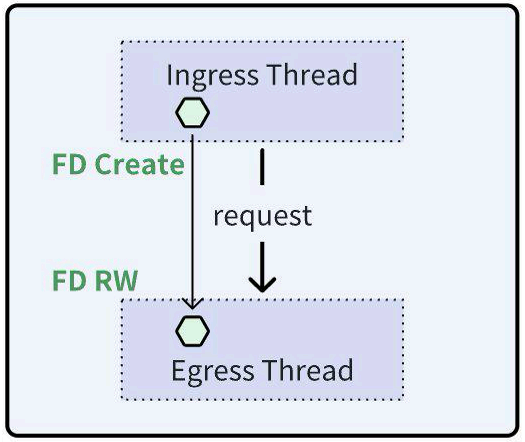
挑战二：跨线程追踪的挑战和进展



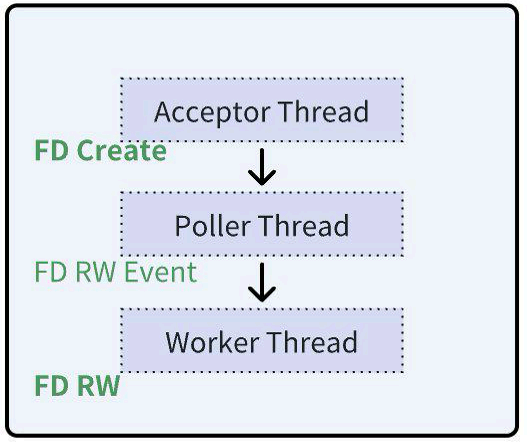
利用「生命周期」追踪：Coroutine、Socket



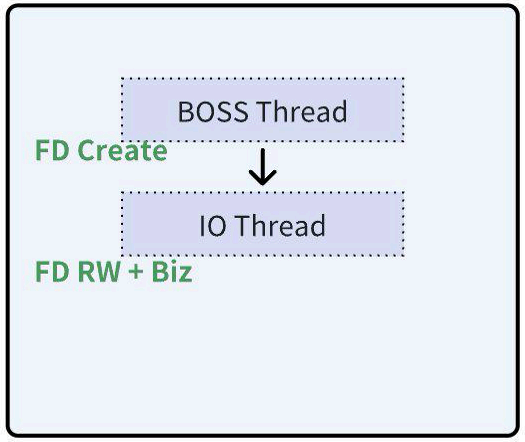
Golang Goroutine Create



Nginx Upstream Socket Create



Tomcat Threading Model



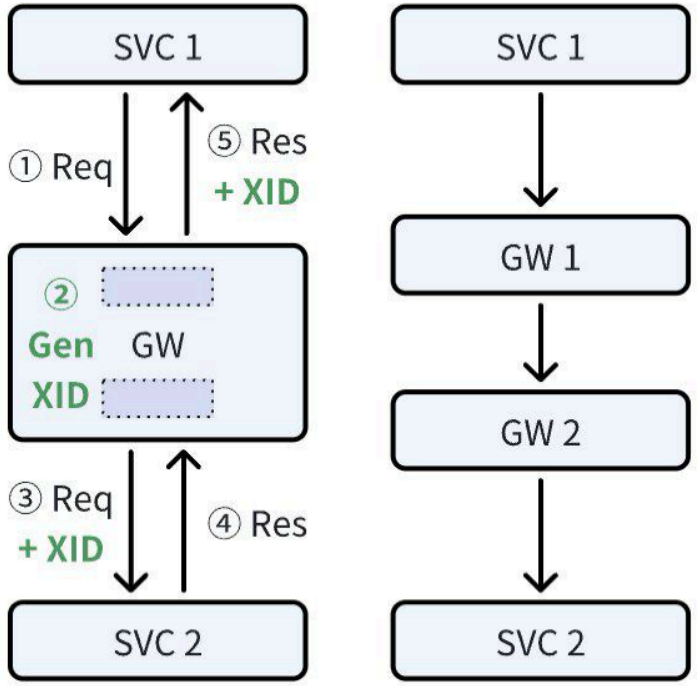
Dubbo DirectDispatcher

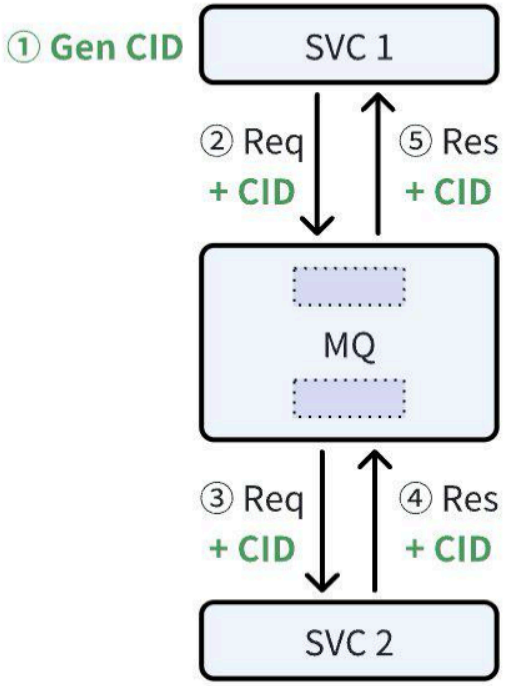
利用「事实标准」追踪：X-Request-ID、Correlation ID

Nginx、HAProxy、Envoy、...

Kafka、ActiveMQ
Request-Response Model

[Envoy: X-Request-ID]
[Nginx: X-Request-ID]
[HAProxy: X-Request-ID]
[BFE: X-Bfe-Log-Id]
[腾讯云 CLB: Stgw-request-id]
[百度云 BLB: X-BLB-Request-Id]
.....





利用「业务规范」追踪：金融交易唯一 ID（Wasm Plugin）

ICS 03.060
CCS A 11

JR

中 华 人 民 共 和 国 金 融 行 业 标 准

JR/T 0248—2022

证券期货业机构内部接口　证券交易

Internal interface of securities and futures institutions Stock exchange

2022-11-14 发布　　2022-11-14 实施

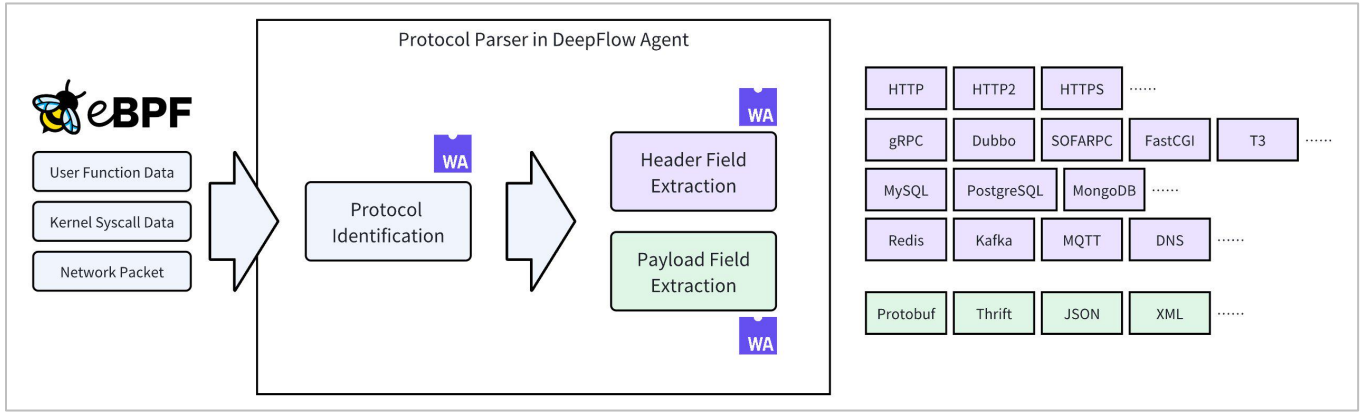
中国证券监督管理委员会　发布

6.4.1.9　组件<Request>

组件<Request>表示查询请求及相关属性。组件<Request>结构定义如表11所示。

表 11　组件<Request>

标签	字段名称	是否必须	说明
组件<Request>		Y	
710	PosReqID	Y	请求唯一标识符
724	PosReqType	Y	请求类别 0=查询股份 9=查询资金 103=查询资金流水 104=查信用资产负债 105=信用合约查询 106=担保证券查询 107=标的证券查询 108=基金交易查询 109=基金份额查询 205=可融券数量查询



结合轻量级插桩追踪：RequestID、SQL Comment

OpenTelemetry

Skywalking

Agent 太重维护不便？

轻量级 Agent
仅注入+传播 RequsetID

[3] 当应用在 SQL 语句的注释中注入 TraceID（或复合的 TraceID + SpanID）时，DeepFlow 支持提取并用于跨线程的分布式追踪。DeepFlow 支持提取几乎任意位置的 SQL 注释（但必须出现在 AF_PACKET 获取到的首包中，或者 eBPF 获取到的第一个 Socket Data 中）：

```
/* your_trace_key: 648840f6-7f92-468b-b298-d38f05c541d4 */ SELECT col FROM tbl
SELECT /* your_trace_key: 648840f6-7f92-468b-b298-d38f05c541d4 */ col FROM tbl
SELECT col FROM tbl # your_trace_key: 648840f6-7f92-468b-b298-d38f05c541d4
SELECT col FROM tbl -- your_trace_key: 648840f6-7f92-468b-b298-d38f05c541d4
```

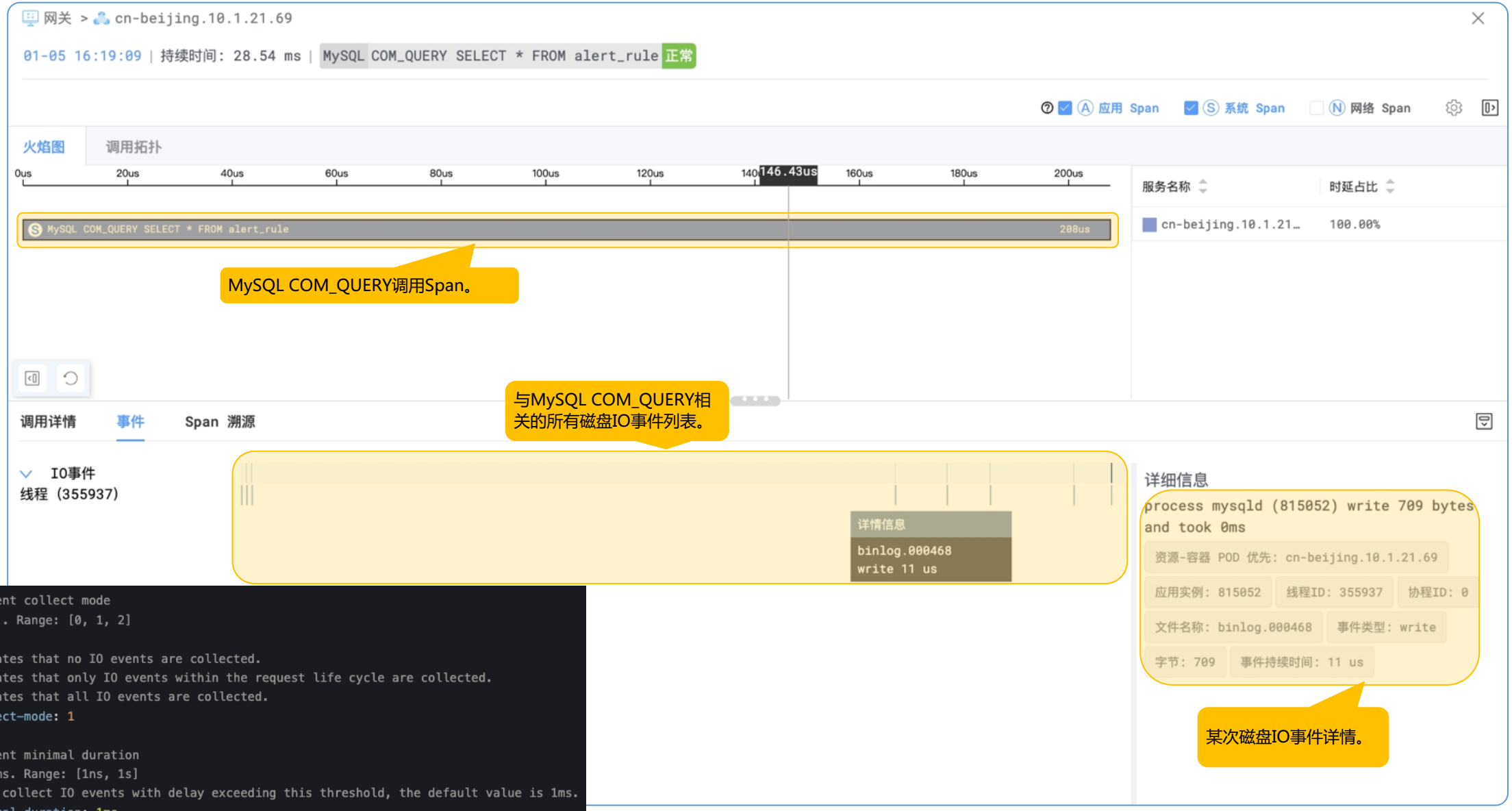
虽然如此，我们**强烈建议您在 SQL 语句头部添加注释**，以降低 SQL 解析的性能开销。上面的示例中，`your_trace_key` 取决于 Agent 配置项中 `http_log_trace_id` 的值（但请注意如果使用 traceparent / sw8 / uber-trace-id，请遵循 OpenTelemetry / SkyWalking / Jaeger 的协议规范）。例如当 `http_log_trace_id = traceparent, sw8` 时，DeepFlow 能够从 `00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01` 中提取符合 OpenTelemetry 规范的 TraceID 和 SpanID：

```
/* traceparent: 00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01 */ SELECT col FROM tbl
```

https://deepflow.io/docs/zh/features/universal-map/l7-protocols/#mysql

DeepFlow 支持解析追踪 ID 的协议：HTTP(1/2)、Dubbo、gRPC、SOFARPC、FastCGI、MySQL、Kafka、...（支持 Wasm 插件）
DeepFlow 支持解析的 APM Tracing 协议：OpenTelemetry、SkyWalking、Jaeger、...（支持自定义）

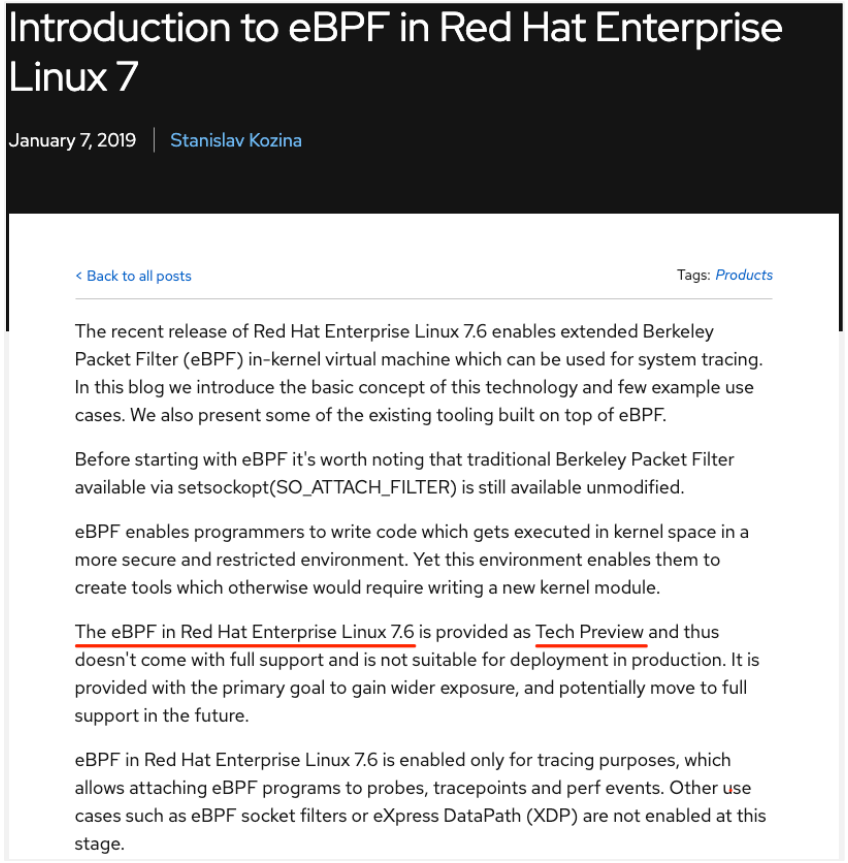
关联 IO：将 Span 与磁盘读写事件自动关联




```
## eBPF IO event collect mode
## Default: 1 . Range: [0, 1, 2]
## Note:
##   0: Indicates that no IO events are collected.
##   1: Indicates that only IO events within the request life cycle are collected.
##   2: Indicates that all IO events are collected.
io-event-collect-mode: 1

## eBPF IO event minimal duration
## Default: 1ms. Range: [1ns, 1s]
## Note: Only collect IO events with delay exceeding this threshold, the default value is 1ms.
io-event-minimal-duration: 1ms
```

内核依赖：低版本内核的挑战



Introduction to eBPF in Red Hat Enterprise Linux 7

January 7, 2019 | Stanislav Kozina

< Back to all posts Tags: *Products*

The recent release of Red Hat Enterprise Linux 7.6 enables extended Berkeley Packet Filter (eBPF) in-kernel virtual machine which can be used for system tracing. In this blog we introduce the basic concept of this technology and few example use cases. We also present some of the existing tooling built on top of eBPF.

Before starting with eBPF it's worth noting that traditional Berkeley Packet Filter available via setsockopt(SO_ATTACH_FILTER) is still available unmodified.

eBPF enables programmers to write code which gets executed in kernel space in a more secure and restricted environment. Yet this environment enables them to create tools which otherwise would require writing a new kernel module.

The eBPF in Red Hat Enterprise Linux 7.6 is provided as Tech Preview and thus doesn't come with full support and is not suitable for deployment in production. It is provided with the primary goal to gain wider exposure, and potentially move to full support in the future.

eBPF in Red Hat Enterprise Linux 7.6 is enabled only for tracing purposes, which allows attaching eBPF programs to probes, tracepoints and perf events. Other use cases such as eBPF socket filters or eXpress DataPath (XDP) are not enabled at this stage.

体系架构	发行版	内核版本	kprobe [4]	Golang uprobe	OpenSSL uprobe	perf
X86	CentOS 7.9	3.10.0 [1]	Y	Y [2]	Y [2]	Y
	RedHat 7.6	3.10.0 [1]	Y	Y [2]	Y [2]	Y
	*	4.9–4.13				Y
	*	4.14 [3]	Y	Y [2]		Y
	*	4.15	Y	Y [2]		Y
	*	4.16	Y	Y		Y
	*	4.17+	Y	Y	Y	Y
ARM	CentOS 8	4.18	Y	Y	Y	Y
	EulerOS	5.10+	Y	Y	Y	Y
	麒麟 KylinOS V10 SP3+	4.19.90–52.25+	Y	Y	Y	Y
	其他发行版	5.8+	Y	Y	Y	Y

https://deepflow.io/docs/zh/ce-install/overview/#运行权限及内核要求

DeepFlow

零侵扰分布式追踪进展和探索

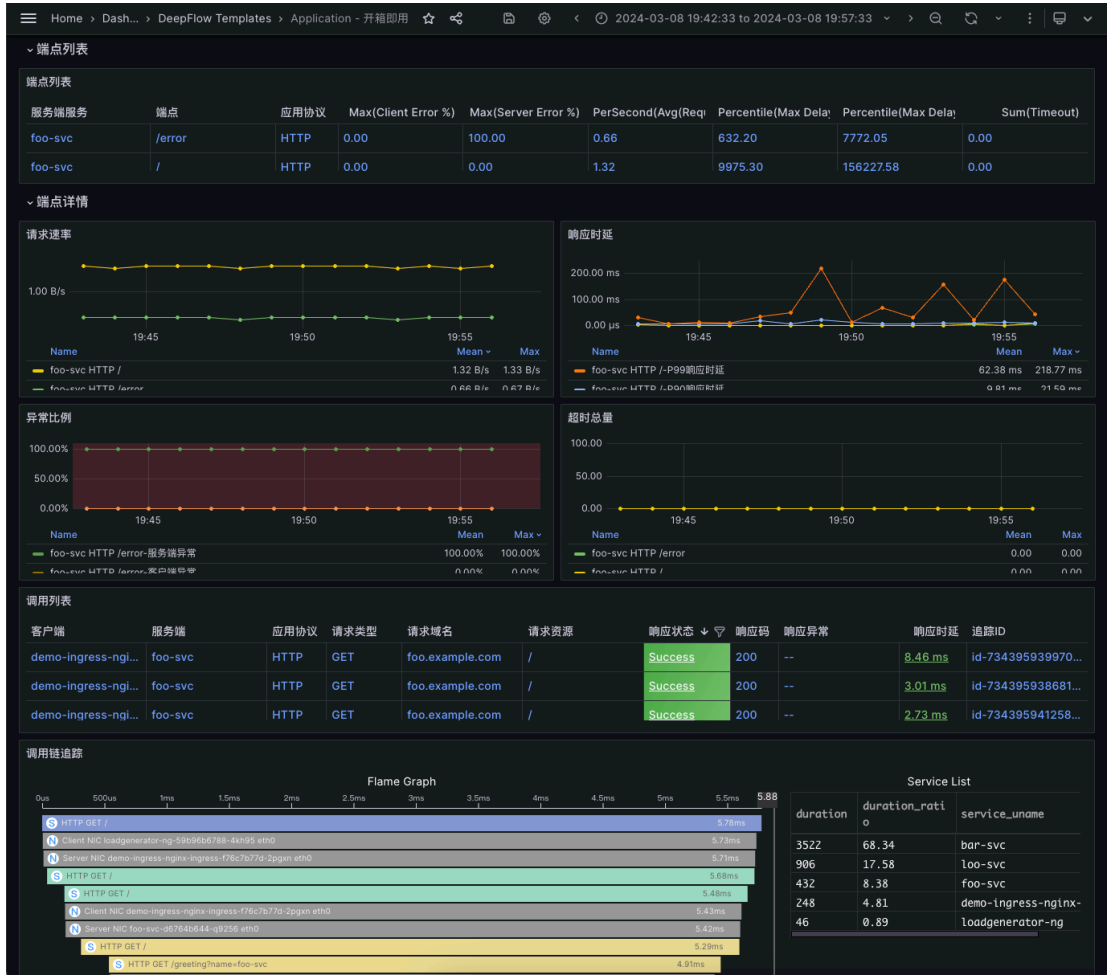
背景

进展

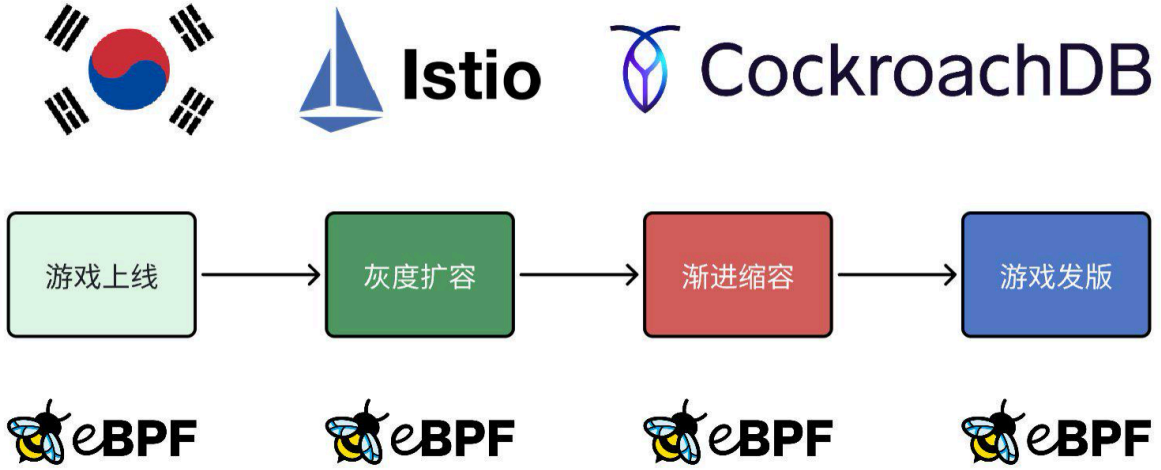
案例

展望

DeepFlow 社区用户（腾讯 IEG）在游戏应用中的实践



从加速排障，到加速版本迭代、线上变更



零侵扰可观测性对 APM 覆盖完善
SLA 99.99+% 有什么价值？
研发效能、技术信任

DeepFlow

零侵扰分布式追踪进展和探索

背景

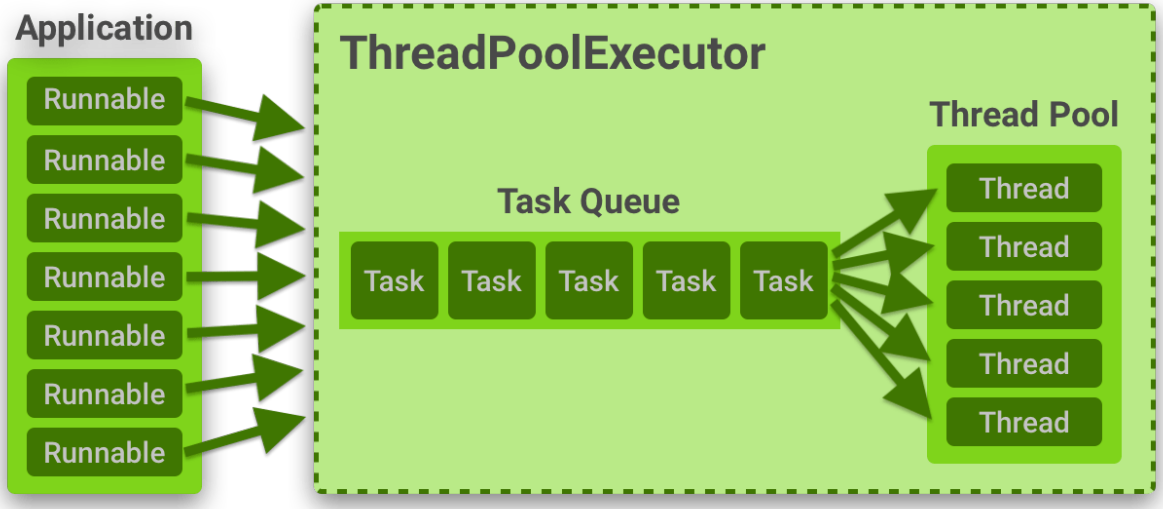
进展

案例

展望

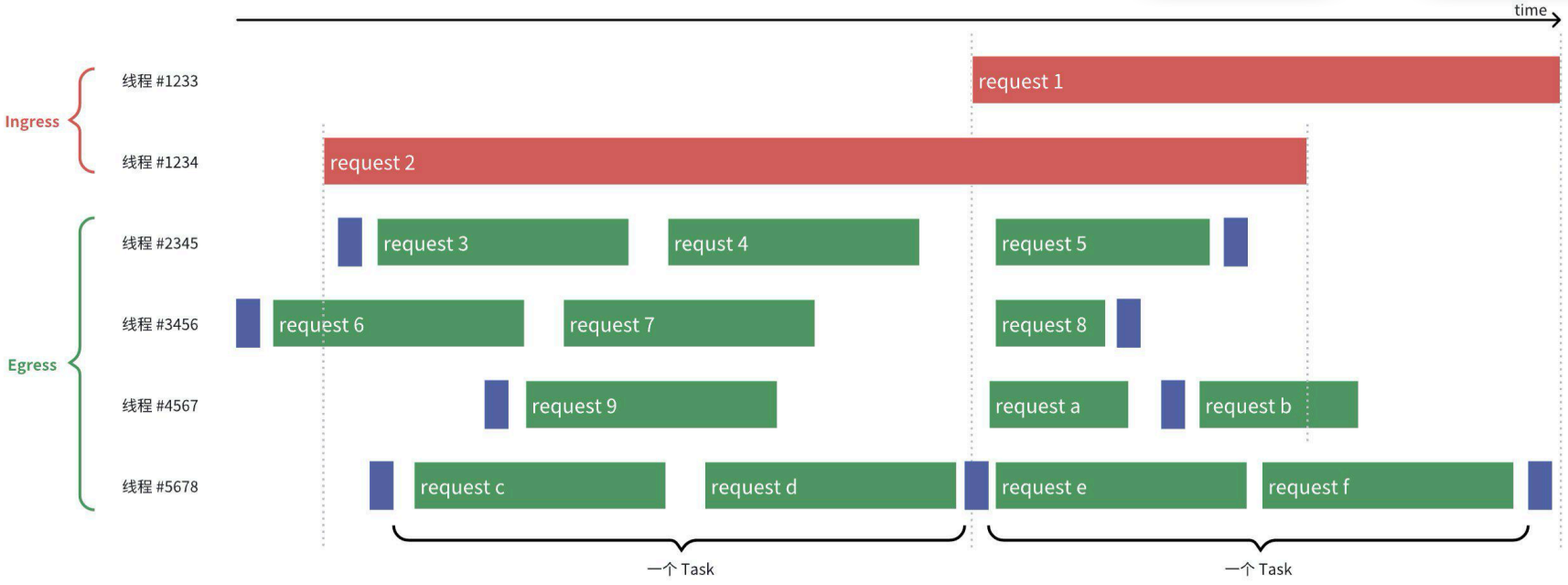
跨线程（Java 线程池）的挑战：eBPF uprobe + 推理

能否零侵扰推理 Task 边界？
能否零侵扰推理 Ingress/Egress 调用关联？



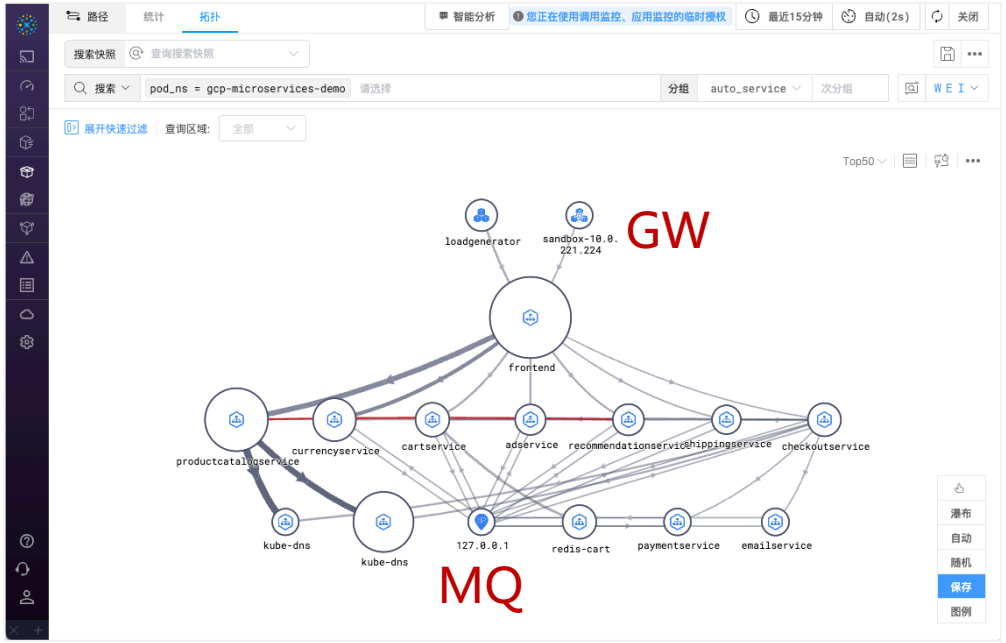
进程 svc-user 共运行了 6 个线程：

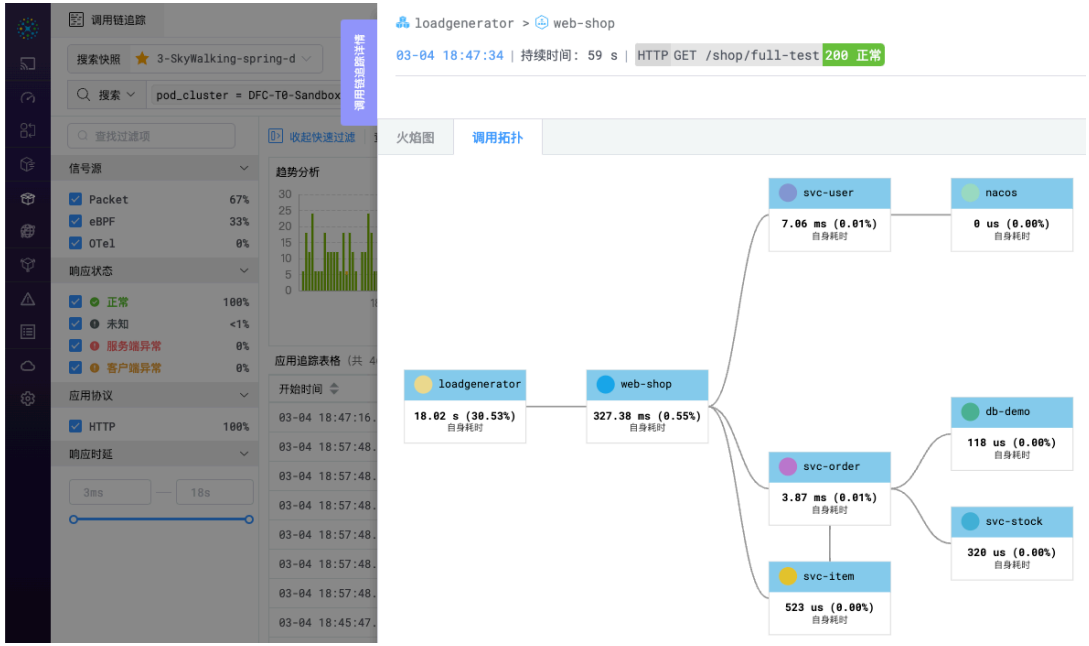
- 2 个线程负责处理 Ingress 调用（IO 线程）
- 4 个线程负责处理 Egress 调用（业务线程）



生成 API 拓扑的挑战：Trace API v.s. eBPF sock_ops

从 Service Map 到 API Map



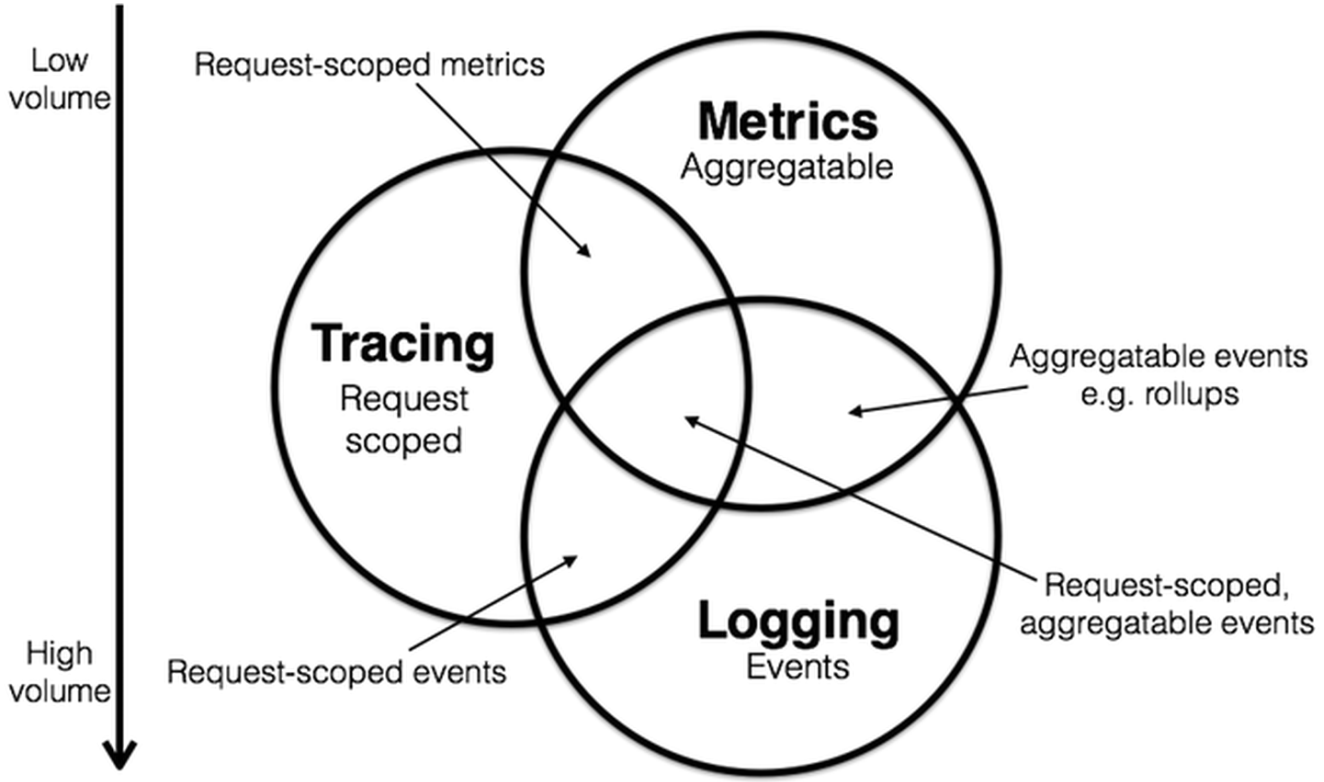


eBPF Span 没有 TraceID 如何聚合为 API 拓扑？
eBPF Span 能否传播 Trace Context？

TCP segment header

Offsets		0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0 0				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if *data offset* > 5. Padded at the end with "0" bits if necessary.)																															
⋮	⋮																																
56	448																																

eBPF + Logging

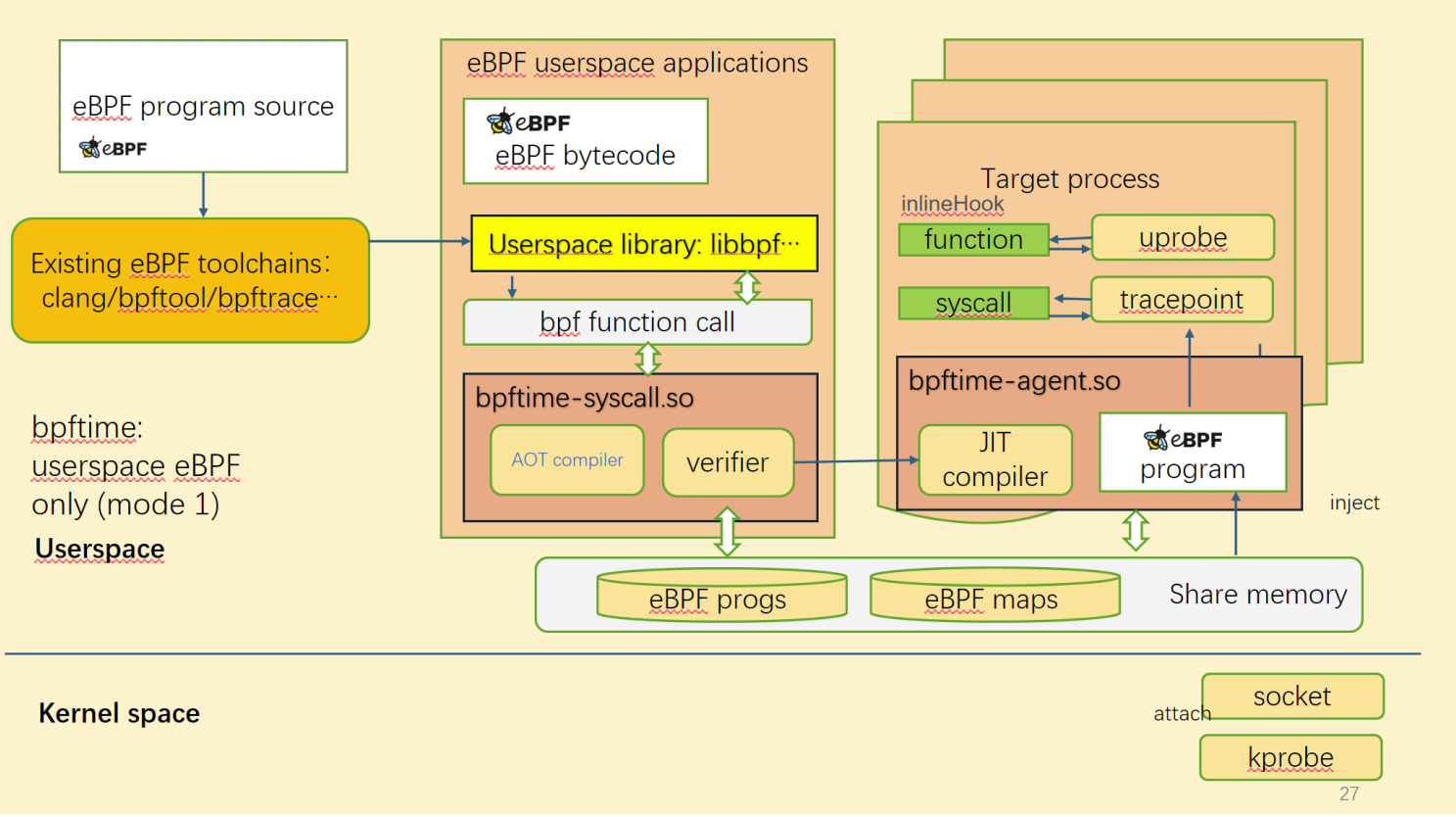


eBPF

✅ 日志天然无需落盘

✅ 日志天然和 Trace 关联

bpftime: Userspace eBPF runtime for fast Uprobe & Syscall Hook & Extensions



Performance Benchmarks

How is the performance of `userspace uprobe` compared to `kernel uprobes`?

Probe/Tracepoint Types	Kernel (ns)	Userspace (ns)
Uprobe	3224.172760	314.569110
Uretprobe	3996.799580	381.270270
Syscall Tracepoint	151.82801	232.57691
Manually Instrument	Not avaliable	110.008430

通向零侵扰可观测性之路

DeepFlow：零侵扰实现云原生应用的全栈可观测性

Prometheus

Open Telemetry

SKywalking

Pyroscope

AS SINK

DeepFlow

AS SOURCE

eBPF

Wasm

Grafana

Prometheus

Open Telemetry

SKywalking

RCA / AIOps / SEC / ...