2013 阿里技术沙龙

享技术·聚朋友

鹰眼下的淘宝

分布式调用跟踪系统介绍

淘宝网 **司徒放**（姬风）

jifeng@taobao.com

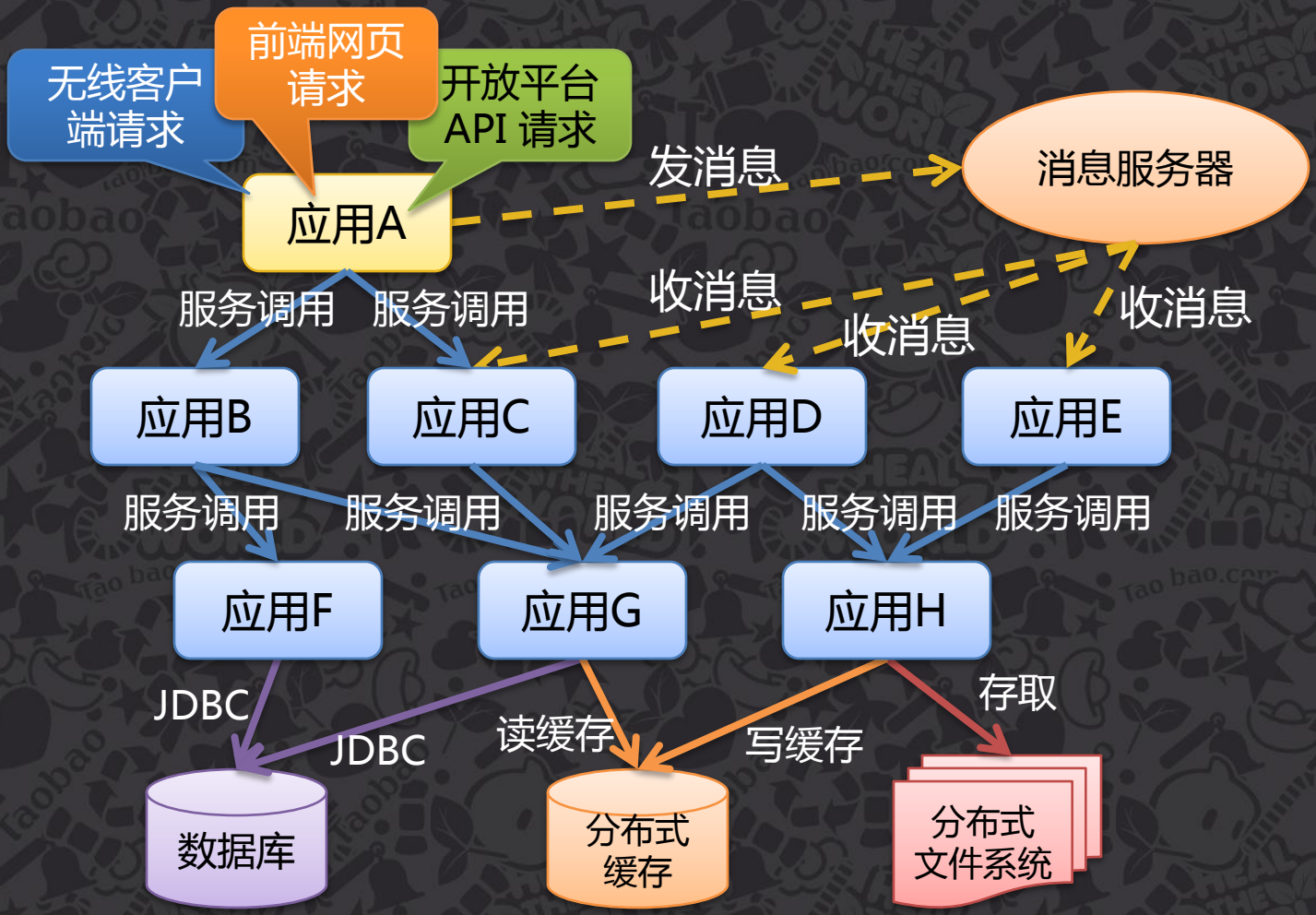
大纲

- 鹰眼是什么
- 鹰眼的使用场景
- 鹰眼的实现

现状

- 日趋复杂的分布式系统
  - 远程服务调用
  - 消息通讯
  - 数据库分库分表
  - 分布式缓存
  - 分布式文件系统
  - ……

现状


无线客户端请求
前端网页请求
开放平台API 请求
应用A
发消息
消息服务器
服务调用
服务调用
收消息
收消息
收消息
应用B
应用C
应用D
应用E
服务调用
服务调用
服务调用
服务调用
服务调用
应用F
应用G
应用H
JDBC
JDBC
读缓存
写缓存
存取
数据库
分布式缓存
分布式文件系统

如何理清这些后端调用关系？

举个例子

- 整个分布式系统
  - 高速公路网
- 前端请求
  - 高速上行驶的车辆
- 处理请求的应用
  - 高速上的收费站

* 图片来源：http://www.etcsd.com/main/business/outlets.jsp

举个例子

- 高速收费站将车辆通行信息记录成日志

[2013-05-01 12:23:34] 鲁A123BC,平度2,S16,济南,￥0

[2013-05-01 12:23:40] 鲁A987DE,平度2,S16,淄博,￥10

[2013-05-01 12:43:15] 鲁A123BC,潍坊1,S20,济南,￥18

[2013-05-01 13:38:29] 鲁A123BC,青州西1,G20,济南,￥10

[2013-05-01 13:38:30] 鲁A567AB,青州西2,G20,潍坊,￥10

[2013-05-01 14:39:27] 鲁A123BC,淄博3,G20,济南,￥15

[2013-05-01 16:42:58] 鲁A123BC,济南3,G20,济南,￥25

……

* 上述内容仅作举例示意说明用，纯属虚构

举个例子

- 可以分析车辆 鲁A123BC 的行驶路线
  - [05-01 12:23:34] 平度2，旅途开始
  - ↓
  - [05-01 13:38:29] 青州西1，耗时 75 分钟，路费 10 元
  - ↓
  - [05-01 14:39:27] 淄博3，耗时 61 分钟，路费 5 元
  - ↓
  - [05-01 16:42:58] 济南3，耗时 123 分钟，路费 10 元

* 上述内容仅作举例示意说明用，纯属虚构

简介

- 鹰眼（ EagleEye ）
  - 基于日志的分布式调用跟踪系统
  - 脱胎于 Google Dapper 论文

  - 核心：**调用链**。每次请求都生成一个全局唯一的ID（ TraceId ），通过它将不同系统的“孤立的”日志串在一起，重组还原出更多有价值的信息

简介

- 目前状况
  - 每日调用链上 1 千亿，来自 500 多个前端，500 多个后端应用，还有数百个数据库、缓存、存储，分析的日志超过 3 千亿行
- 覆盖了淘宝主要的有网络通信的中间件
  - ✓ 前端请求接入：Tengine(nginx)
  - ✓ 分布式 Session：tbsession
  - ✓ 远程服务调用框架 (RPC)：HSF
  - ✓ 异步消息通讯 (MQ)：Notify
  - ✓ 分库分表访问数据库 (JDBC)：TDDL
  - ✓ 分布式缓存 (memcache)：Tair
  - ✓ 分布式文件系统 (HDFS)：TFS
  - ✓ 特定功能的客户端，如搜索、支付等
  - ✓ 其他中间件，如：HttpClient……

大纲

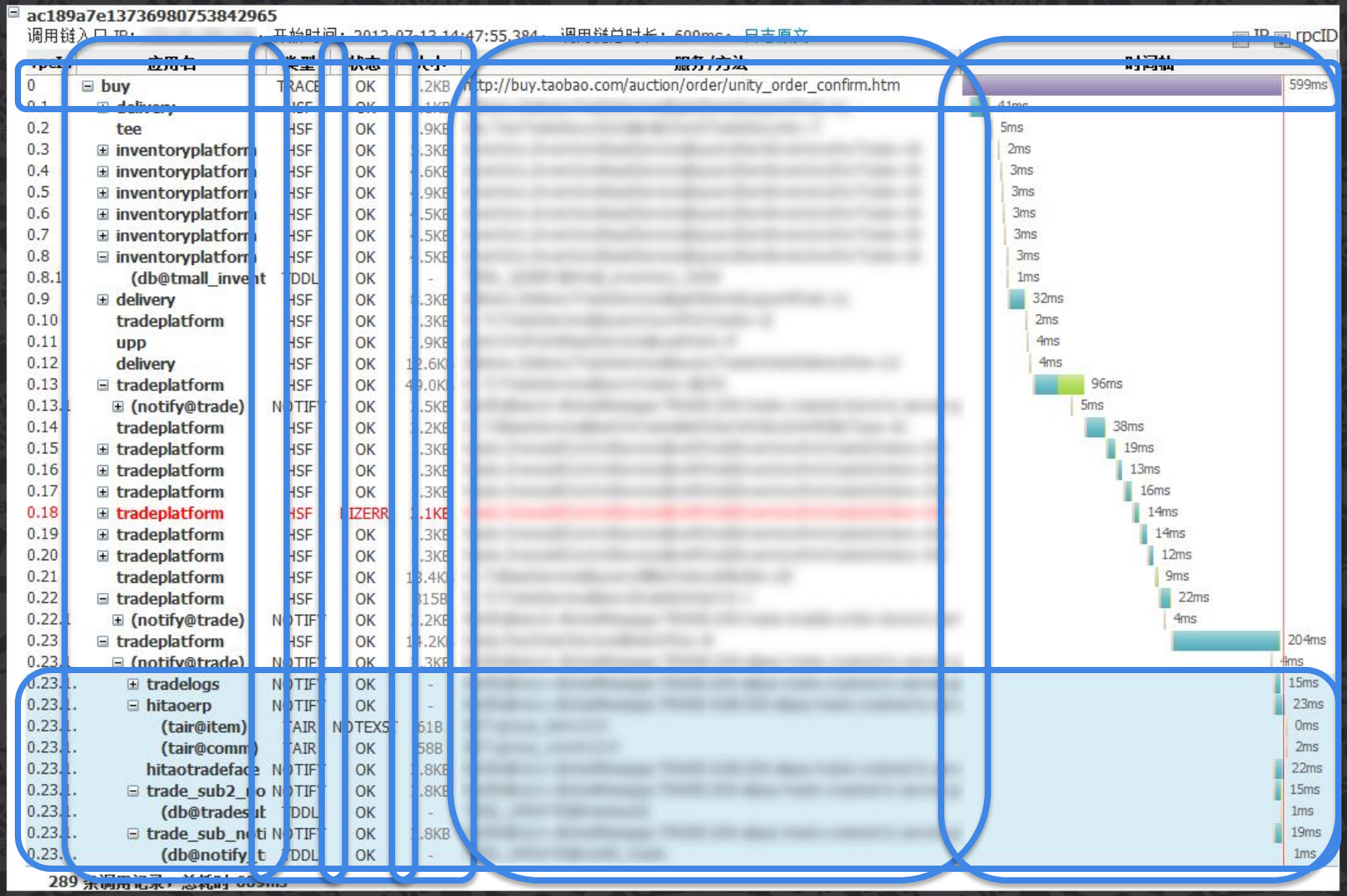
- 鹰眼是什么
- 鹰眼的使用场景
- 鹰眼的实现

1. 调用链跟踪

单条调用链的展现

调用链跟踪


Taobao


ac189a7e13736980753842965
rpcID
0 buy TRACE OK .2KB http://buy.taobao.com/auction/order/unity_order_confirm.htm 599ms
0.2 tee HSF OK .9KB 5ms
0.3 inventoryplatform HSF OK .3KB 2ms
0.4 inventoryplatform HSF OK .6KB 3ms
0.5 inventoryplatform HSF OK .9KB 3ms
0.6 inventoryplatform HSF OK .5KB 3ms
0.7 inventoryplatform HSF OK .5KB 3ms
0.8 inventoryplatform HSF OK .5KB 3ms
0.8.1 (db@tmall_invent TDDL OK - 1ms
0.9 delivery HSF OK .3KB 32ms
0.10 tradeplatform HSF OK .3KB 2ms
0.11 upp HSF OK .9KB 4ms
0.12 delivery HSF OK 12.6K 4ms
0.13 tradeplatform HSF OK 49.0K 96ms
0.13.1 (notify@trade) NOTIF OK .5KB 5ms
0.14 tradeplatform HSF OK .2KB 38ms
0.15 tradeplatform HSF OK .3KB 19ms
0.16 tradeplatform HSF OK .3KB 13ms
0.17 tradeplatform HSF OK .3KB 16ms
0.18 tradeplatform HSF SIZERR .1KB 14ms
0.19 tradeplatform HSF OK .3KB 14ms
0.20 tradeplatform HSF OK .3KB 12ms
0.21 tradeplatform HSF OK 13.4K 9ms
0.22 tradeplatform HSF OK 315B 22ms
0.22.1 (notify@trade) NOTIF OK .2KB 4ms
0.23 tradeplatform HSF OK 14.2K 204ms
0.23.1 (notify@trade) NOTIF OK .3KB 4ms
0.23.1.1 tradelogs NOTIF OK - 15ms
0.23.1.2 hitaoerp NOTIF OK - 23ms
0.23.1.2.1 (tair@item) TAIR NOTEXS 61B 0ms
0.23.1.2.2 (tair@comm) TAIR OK 58B 2ms
0.23.1.2.3 hitaotradefac NOTIF OK .8KB 22ms
0.23.1.2.4 trade_sub2_ NOTIF OK .8KB 15ms
0.23.1.2.4.1 (db@tradesub TDDL OK - 1ms
0.23.1.2.5 trade_sub_noti NOTIF OK .8KB 19ms
0.23.1.2.5.1 (db@notify_t TDDL OK - 1ms
289

调用链跟踪

- 与异常监控集成
  - 排查问题需要查看 n 台机器的监控、日志？

    - 应用报异常或处理超时，在日志打印当前调用链 TraceId
    - 用 TraceId 查询调用链，定位问题
    - 把链路上下文抛出的异常堆栈关联起来
    - 把链路上的服务器 CPU Load、JVM、IO、网络状况关联起来

调用链跟踪

2013-07-20 05:25:28,179 ERROR taobao.hsf -
基于 RPC 协议调用服务
[com.taobao.wireless.trade.api.tmall.hsf.TmallBagInterface:1.0.0]的
[bulidConfirmOrder]方法时出现错误：

所调用的服务目标地址为：[...]
参数信息为：[...] TraceId=ac1828791374269125174692 3
错误原因为超时，请查看服务器端的执行日志是否也超时，执行时间为：3000 毫秒。

HSFTimeOutException
at com.taobao.hsf.....HSFResponseFuture.getResponse(HSFResponseFuture.java:52)
at com.taobao.hsf.....SyncInvokeComponent.invoke(SyncInvokeComponent.java:51)
at ...

调用链跟踪

监控系统从日志匹配异常堆栈和错误信息中的 TraceId

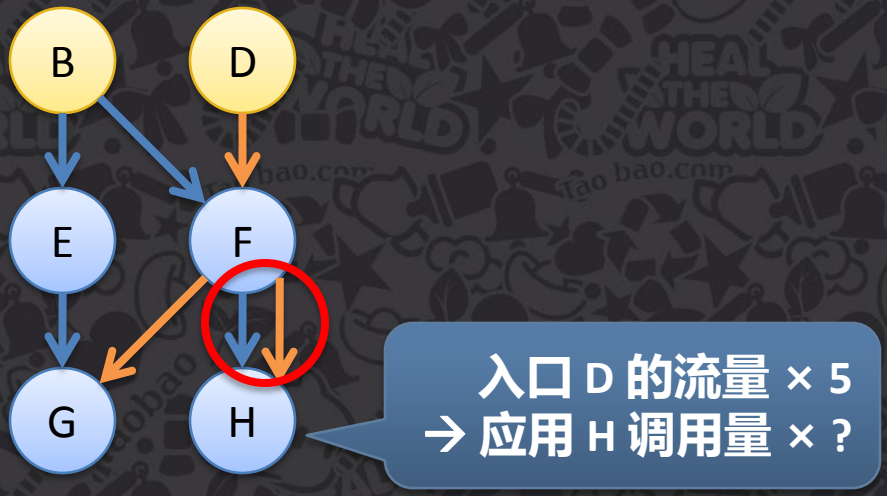
TraceId=ac1828791374269125174692 3

2. 链路分析

对多条调用链做统计和分析

链路分析

- 容量规划
  - 一般系统只统计对直接依赖的调用量
  - 调用链可以得到对间接依赖、异步依赖的调用量

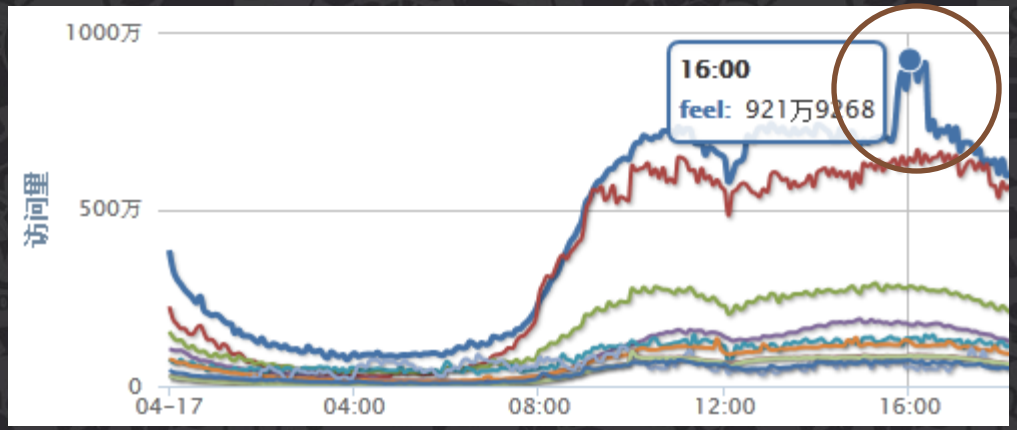


链路分析

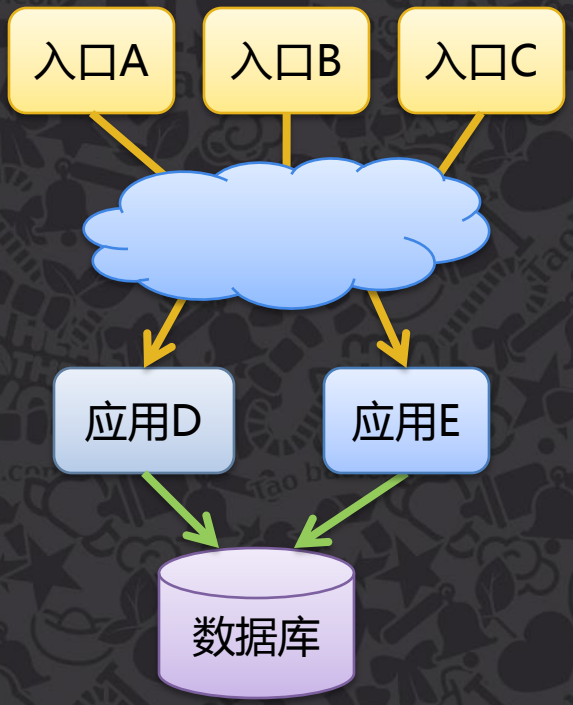
层次	名称	应用	QPS	峰值QPS	调用次数	平均耗时	本地耗时	依赖度	耗时比例	标记	
根	[redacted]	[redacted]	118.13	269.64	1.0	435ms	51ms	100.0%	11.88%		
1	[redacted]	[redacted]	226.02	483.12	1.94	8ms	8ms	98.51%	3.64%		
2	[redacted]	[redacted]	282.74	533.96	4.06	0ms	0ms	58.94%	0.02%		
2	[redacted]	[redacted]	74.41	170.57	1.77	0ms	0ms	35.56%	0.08%		
1	[redacted]	[redacted]	183.31	357.95	1.71	3ms	2ms	90.58%	1.0%		
2	[redacted]	[redacted]	182.59	356.96	1.71	0ms	0ms	90.28%	0.32%		
1	[redacted]	[redacted]	130.91	244.85	1.71	19ms	7ms	64.79%	1.98%		
2	[redacted]	[redacted]	219.58	412.81	2.88	3ms	3ms	64.52%	1.37%		
2	[redacted]	[redacted]	219.55	412.73	2.88	1ms	1ms	64.51%	0.45%		
2	[redacted]	[redacted]	131.28	243.79	1.72	0ms	0ms	64.51%	0.12%		
2	[redacted]	[redacted]	125.25	231.11	1.7	0ms	0ms	62.2%	0.12%		
1	[redacted]	[redacted]	124.27	230.45	1.1	25ms	19ms	96.07%	4.66%	强依赖	
2	[redacted]	[redacted]	120.47	223.52	1.07	5ms	5ms	95.48%	1.3%		
1	[redacted]	[redacted]	116.05	247.93	1.0	4ms	4ms	98.25%	1.05%		
1	[redacted]	[redacted]	114.76	216.99	1.08	43ms	37ms	90.14%	8.28%	强依赖	
1	[redacted]	[redacted]	113.49	214.98	1.0	5ms	5ms	96.08%	1.16%	强依赖	
1	[redacted]	[redacted]	111.23	236.02	1.03	3ms	3ms	91.82%	0.68%	强依赖	
1	[redacted]	[redacted]	111.2	208.29	1.05	13ms	13ms	89.46%	2.88%		
1	[redacted]	[redacted]	110.24	213.12	1.0	214ms	207ms	93.32%	44.5%	瓶颈	
2	[redacted]	[redacted]	116.92	220.86	1.07	5ms	5ms	92.59%	1.28%		
1	[redacted]	[redacted]	106.5	205.27	1.0	3ms	3ms	90.16%	0.81%		

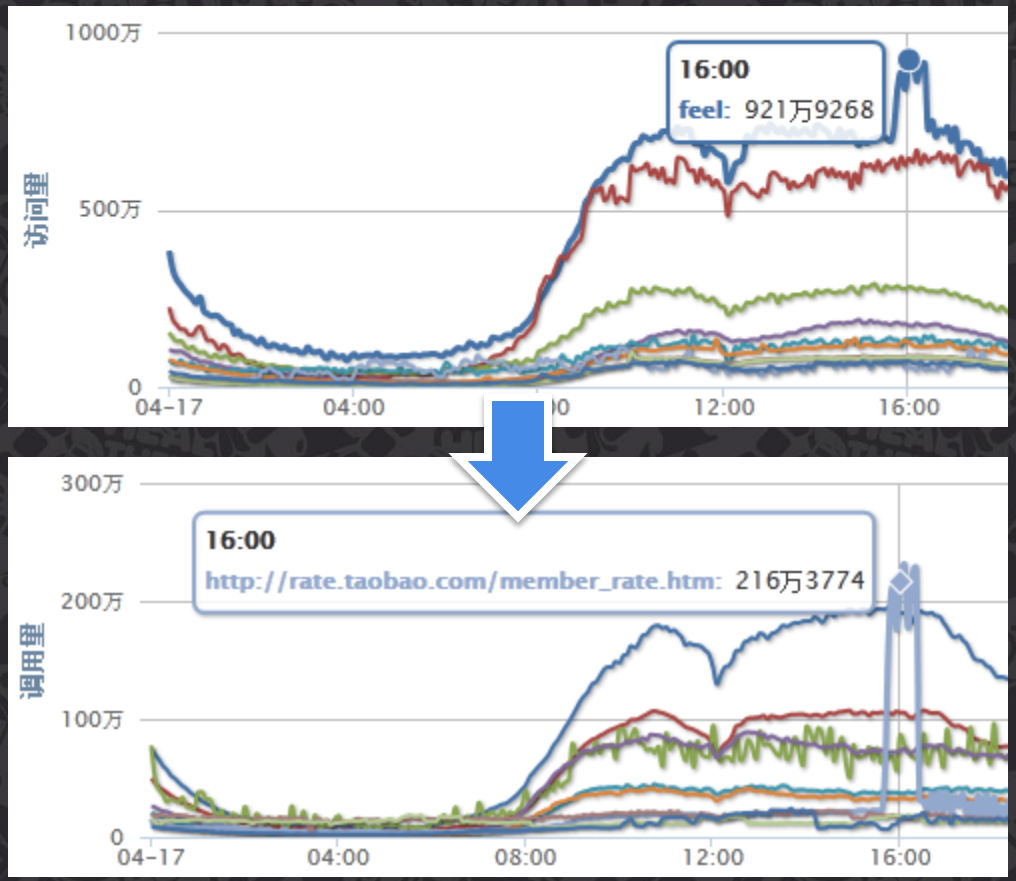
链路分析

- 调用来源
  - 后端数据库请求量突然上涨，需要排查请求来源



链路分析


入口A
入口B
入口C
应用D
应用E
数据库


1000万
500万
0
访问量
16:00
feel: 921万9268
04-17
04:00
12:00
16:00
300万
200万
100万
0
调用量
16:00
http://rate.taobao.com/member_rate.htm: 216万3774
04-17
04:00
08:00
12:00
16:00

链路分析

- 依赖度量
  - 强依赖 & 弱依赖

应用名	类型	状态	大小	服务/方法	时间轴
buy	TRACE	OK	-	http://buy.taobao.com/auction/order/unity_order_confirm.htm	137ms
delivery	HSF	OK	6.2KB		8ms
(tair@uic)	TAIR	OK	57B		0ms
(tair@uic)	TAIR	OK	57B		0ms
tee	HSF	OK	1.9KB		5ms
inventoryplatform	HSF	OK	6.0KB		4ms
(db@tmall_inven	TDDL	OK	-		0ms
inventoryplatform	HSF	OK	6.0KB		4ms
(db@tmall_inven	TDDL	OK	-		1ms
delivery	HSF	OK	6.3KB		5ms
tradeplatform	HSF	OK	1.3KB		4ms
ump	HSF	OK	12.8KB		4ms
upp	HSF	OK	6.6KB		4ms
delivery	HSF	OK	11.5KB		4ms
tradeplatform	HSF	ERR	42.9KB		59ms
(notify@trade)	NOTIFY	OK	2.4KB		6ms

16 条调用记录，总耗时 137ms

强依赖

ERROR

链路分析

- 依赖度量
  - 强依赖 & 弱依赖

应用名	类型	状态	大小	服务/方法	时间轴
buy	TRACE	OK	-	http://buy.taobao.com/auction/order/unity_order_confirm.htm	618ms
tee	HSF	TIMEOUT	1.9KB		9ms
inventoryplatform	HSF	OK	4.8KB		57ms
inventoryplatform	HSF	OK	4.8KB		4ms
inventoryplatform	HSF	OK	4.8KB		3ms
inventoryplatform	HSF	OK	4.8KB		3ms
delivery	HSF	OK	8.1KB		3ms
tradeplatform	HSF	OK	1.3KB		12ms
ump	HSF	OK	13.2KB		2ms
upp	HSF	OK	6.2KB		10ms
delivery	HSF	OK	13.9KB		3ms
tradeplatform	HSF	OK	40.8KB		6ms
tradeplatform	HSF	OK	2.6KB		46ms
itemcenter	HSF	OK	2.0KB		16ms
itemcenter	HSF	OK	2.0KB		10ms
itemcenter	HSF	OK	2.0KB		14ms
itemcenter	HSF	OK	2.0KB		12ms
ump	HSF	OK	8.7KB		12ms
tradeplatform	HSF	OK	10.0KB		53ms
tradeplatform	HSF	OK	814B		5ms
tradeplatform	HSF	OK	10.8KB		23ms
					277ms

261 条调用记录，总耗时 1s230ms

弱依赖

ERROR

链路分析

- 依赖度量
  - 强依赖
    - 调用失败会直接中断主流程
  - 高度依赖
    - 一次链路中调用某个依赖的几率高
  - 频繁依赖
    - 一次链路调用同一个依赖的次数多

链路分析

层次	名称	应用	QPS	峰值QPS	调用次数	平均耗时	本地耗时	依赖度	耗时	标记	
根					1.0	435ms	51ms	100.0%			
1					1.94	8ms	8ms	98.51%	3.64%		
2			282.74	533.96	4.06	0ms	0ms	58.94%	0.02%		
2			74.41	170.57	1.77	0ms	0ms	35.56%	0.08%		
1			183.31	357.95	1.71	3ms	2ms	90.58%	1.0%		
2			182.59	356.96	1.71	0ms	0ms	90.28%	0.32%		
1			130.91	244.85	1.71	19ms	7ms	64.79%	1.98%		
2			219.58	412.81	2.88	3ms	3ms	64.52%	1.37%		
2			219.55	412.73	2.88	1ms	1ms	64.51%	0.45%		
2			131.28	243.79	1.72	0ms	0ms	64.51%	0.12%		
2			125.25	231.11	1.7	0ms	0ms	62.2%	0.12%		
1			124.27	230.45	1.1	25ms	19ms	96.07%	4.66%	强依赖	
2			120.47	223.52	1.07	5ms	5ms	95.48%			
1			116.05	247.93	1.0	4ms	4ms	98.25%			
1			114.76	216.99	1.08	43ms	37ms	90.14%	8.28%	强依赖	
1			113.49	214.98	1.0	5ms	5ms	96.08%	1.16%	强依赖	
1			111.23	236.02	1.03	3ms	3ms	91.82%	0.68%	强依赖	
1			111.2	208.29	1.05	13ms	13ms	89.46%	2.88%		
1			110.24	213.12	1.0	214ms	207ms	93.32%	44.5%	瓶颈	
2			116.92	220.8			s	92.59%	1.28%		
1			106.5	205.2			s	90.16%	0.81%		

高度依赖

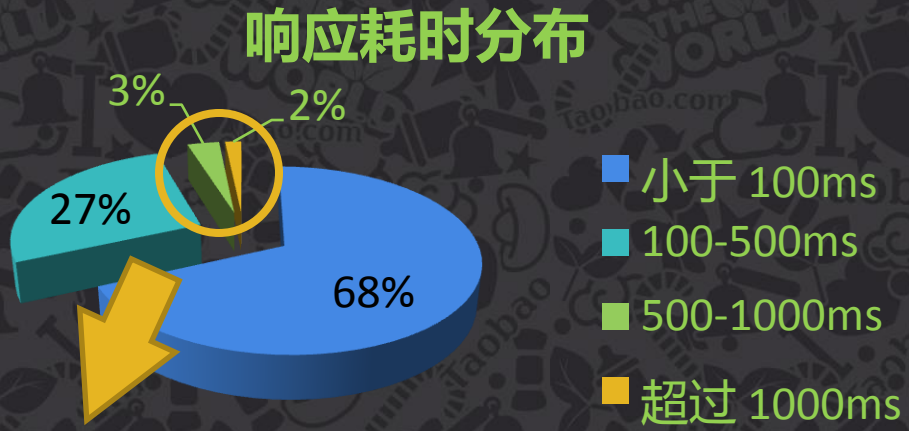
频繁依赖

强依赖

耗时瓶颈

链路分析

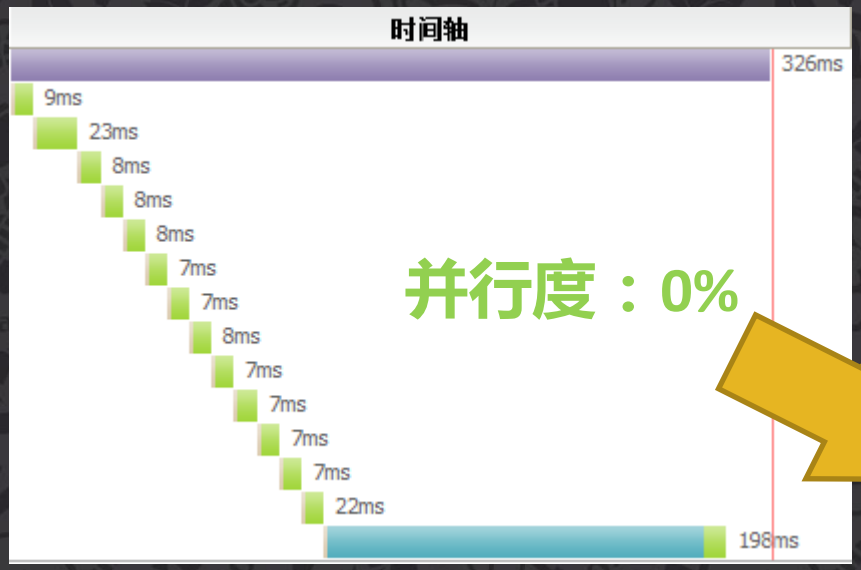
- 调用耗时
  - 瓶颈点
  - 非正常的瓶颈
    - 弱依赖异常导致主流程耗时过长

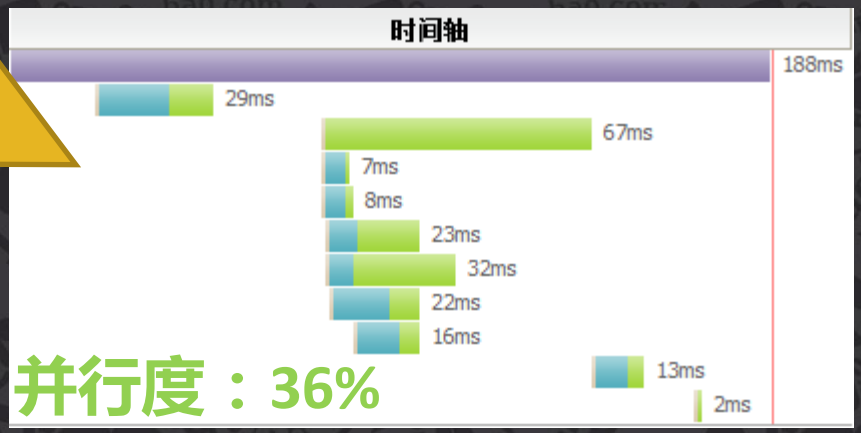


应用名	类型	状态	大小	服务/方法	时间轴
⊟ tmallbuy	TRACE	OK	-	http://buy.tmall.com/order/confirm_order.htm	3s130ms
tradeplatform	HSF	OK	1.3KB		3ms
delivery	HSF	TIMEOUT	845B		3s8ms
⊞ itemcenter	HSF	OK	10.8KB		9ms
⊞ itemcenter	HSF	OK	3.5KB		4ms
⊞ pointcenter	HSF	OK	681B		2ms
⊞ memberplatform	HSF	OK	1.3KB		3ms
⊞ inventoryplatforr	HSF	OK	4.7KB		3ms
⊞ delivery	HSF	OK	7.0KB		7ms
⊞ maybach	HSF	OK	12.5KB		42ms
tradeplatform	HSF	OK	732B		1ms
⊞ delivery	HSF	OK	12.3KB		7ms
⊞ tradeplatform	HSF	OK	3.3KB		18ms
tradeplatform	HSF	OK	816B		1ms

链路分析

- 调用并行度
  - 为链路并行、异步优化提供参考

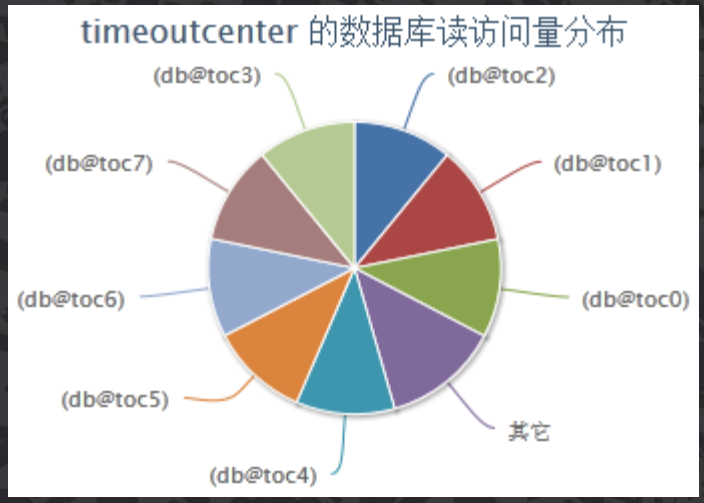




$$并行度 = 1 - \frac{实际执行时间}{本地执行时间 + \sum 直接子依赖执行时间}$$

链路分析

- 调用路由情况
  - 调用分布均衡性
    - 是否存在热点？
  - 检验网络路由
    - 路由是否正常？
    - 网络是否封闭？



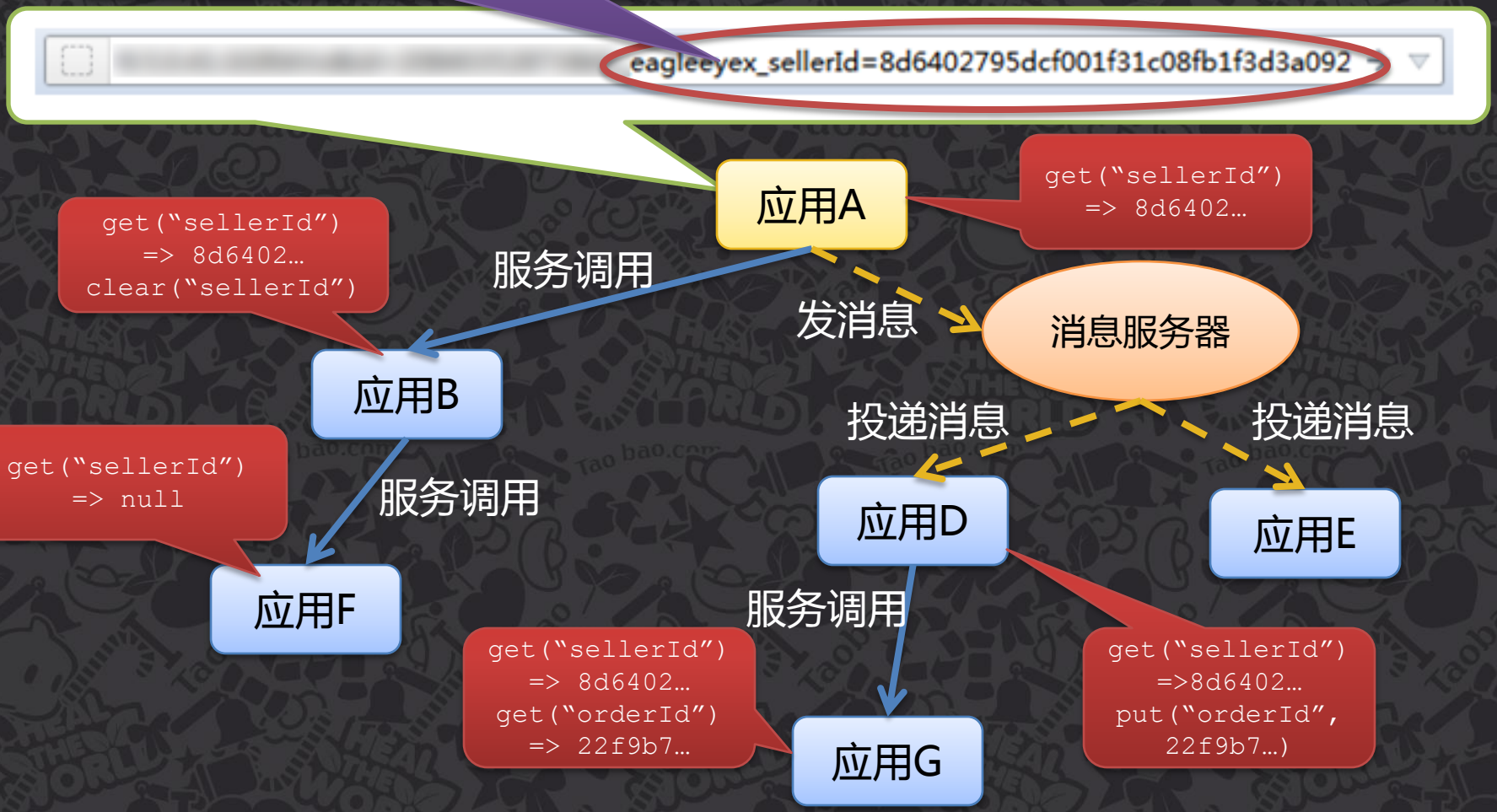
253B	单元内	GET:group_market1:3	0ms
634B	单元内	GET:group_market1:6	2ms
3.6KB	单元内	delivery.DeliveryTradeServ	4ms
257B	单元内	GET:group_uic:76	0ms
759B	单元内	trade.BuyOrderStoreServi	2ms
248B	单元内	GET:group_uic:83	1ms
6.2KB	单元内	delivery.DeliveryTradeServ	5ms
257B	单元内	GET:group_uic:76	1ms
2.0KB	单元内	point.ProPointReadService	7ms
93B	跨单元	GET:group_1:136	1ms
3.1KB	单元内	GET:group_market1:221	2ms
3.9KB	单元内	trade.ITradeServiceCente	48ms
3.9KB	跨单元	trade.ITradeServiceProvid	44ms
1.0KB	忽略	matrix.CoinTradeService:2	30ms
3.1KB	单元内	GET:group_market1:221	3ms
721B	单元内	tc.TcTradeService:1.0.0.c	4ms

3. 透明数据传输

将业务数据与调用链集成

透明数据传输

eagleeyex_**sellerId**



透明数据传输

- 鹰眼自身需要传递 TraceId 等上下文信息
- 在调用链上透明传输业务数据
  - 调用路由控制
    - 传递特定环境的标识，用于调用路由判断
  - 调试指令
    - 在 URL 上设置调试指令，操纵后端服务
  - 前端网关特有的数据
    - 把前端应用特有的数据传到若干层后端的某个服务中

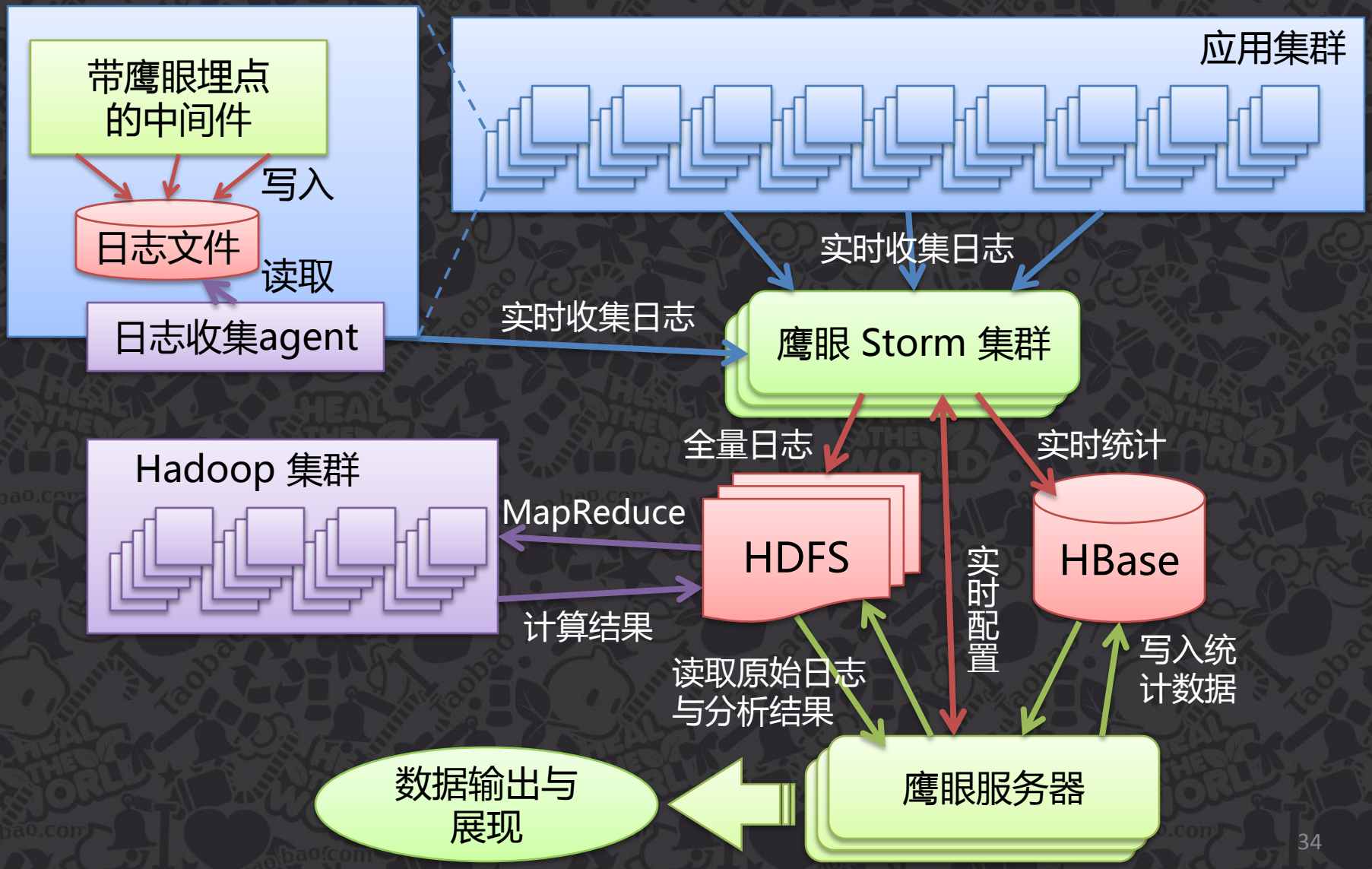
小结

- 调用链跟踪
- 链路分析
- 透明数据传输

大纲

- 鹰眼是什么
- 鹰眼的使用场景
- 鹰眼的实现

整体架构


带鹰眼埋点
的中间件
写入
日志文件
读取
日志收集agent
应用集群
实时收集日志
实时收集日志
鹰眼 Storm 集群
全量日志
实时统计
Hadoop 集群
MapReduce
HDFS
HBase
计算结果
实时配置
读取原始日志
与分析结果
写入统
计数据
数据输出与
展现
鹰眼服务器

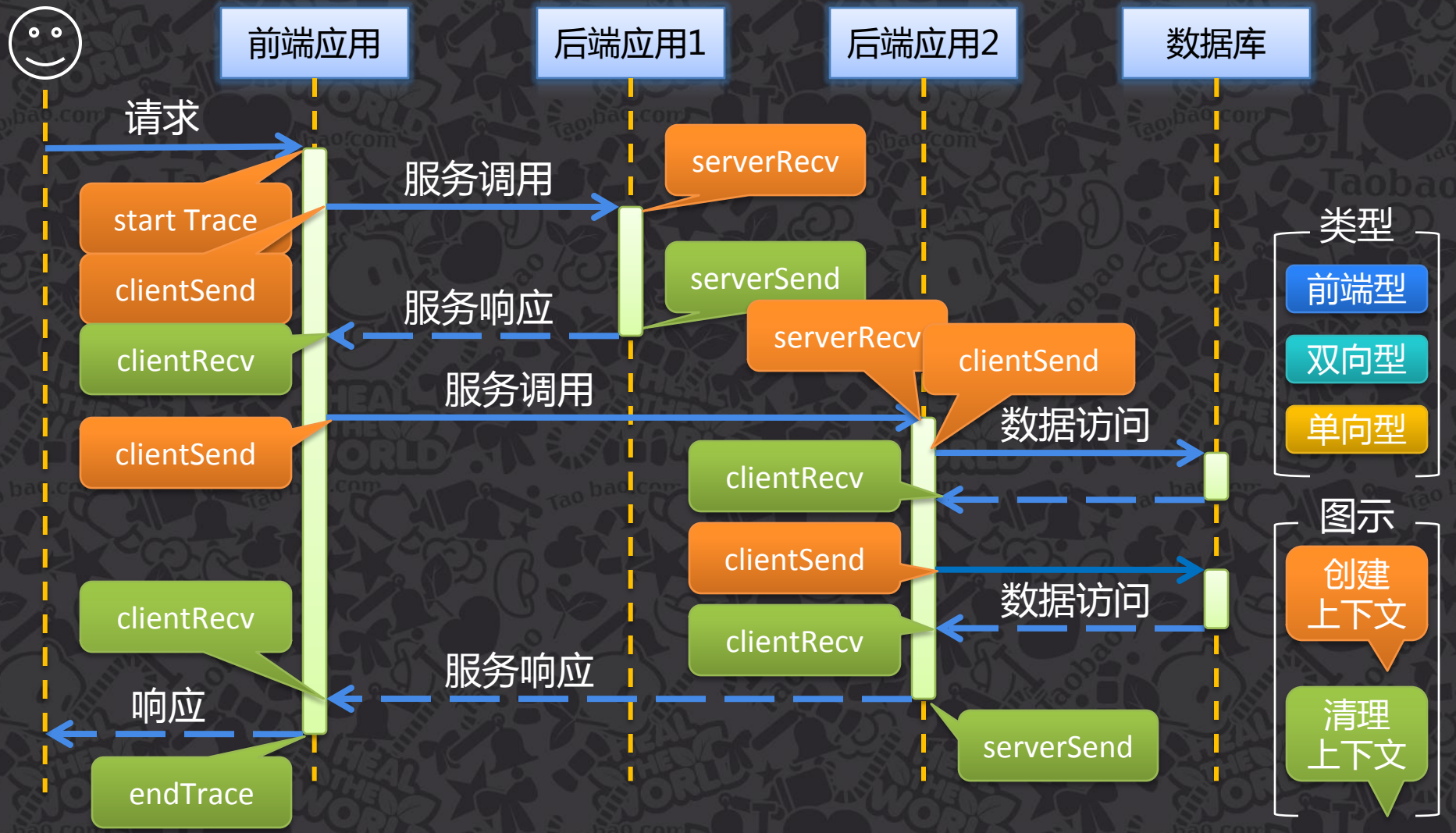
整体实现介绍

1. 埋点和输出日志
2. 收集和存储日志
3. 分析调用链

埋点和输出日志

- 如何埋点实现透传
  - 在中间件创建调用上下文，生成日志埋点
  - 调用上下文放在本地 ThreadLocal，对业务透明
  - 调用上下文在中间件的网络请求中传递

埋点和输出日志


前端应用
后端应用1
后端应用2
数据库
请求
start Trace
clientSend
clientRecv
clientSend
clientRecv
响应
endTrace
服务调用
服务响应
服务调用
服务响应
serverRecv
serverSend
serverRecv
clientSend
clientRecv
clientSend
clientRecv
serverSend
数据访问
数据访问
类型
前端型
双向型
单向型
图示
创建上下文
清理上下文

埋点和输出日志

- 做了哪些埋点
  - TraceId、RpcId、开始时间、调用类型、对端 IP
  - 处理耗时
  - 处理结果（ResultCode）
  - 传输量：请求大小/响应大小
  - 与中间件相关的数据

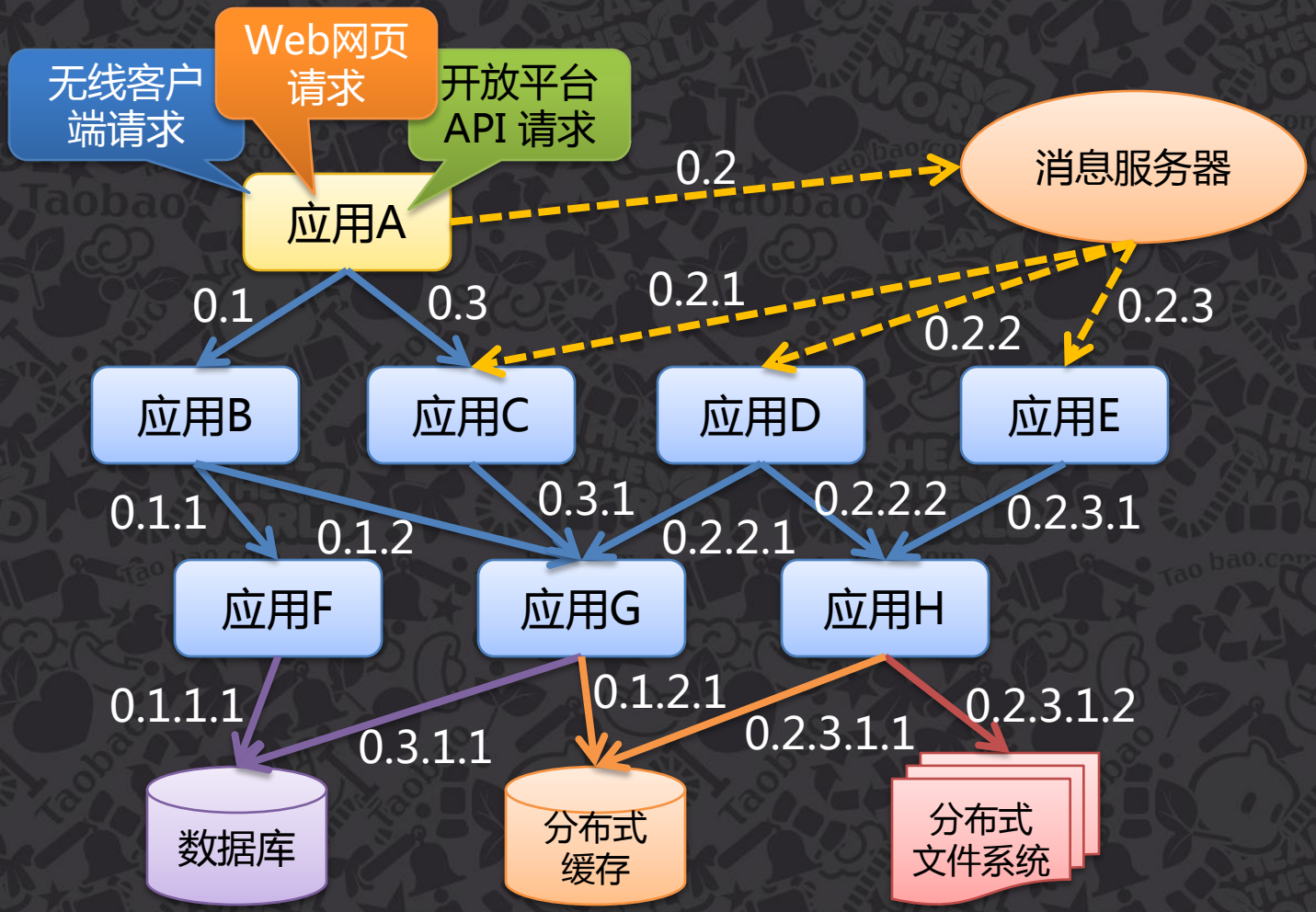
埋点和输出日志

- 调用上下文：TraceId
  - 关联一次请求相关的日志，全局唯一，在各个系统间传递
  - 是否需要业务语义？
    - IP 地址：在淘宝环境可直接映射到前端应用
    - 创建时间：在存储时用于分区
    - 顺序数：用于链路采样
    - 进程号：可选，单机多进程的应用使用
    - 标志位：可选，用于调试和标记

埋点和输出日志

- 调用上下文：RpcId
  - 标识日志埋点顺序和嵌套关系，也在各个系统间传递
- 调用关系
  - 同步 / 异步 / 一对多调用
- 用什么方式实现 RpcId 适合表示上述关系？
  - 顺序编号：1、2、3…
  - 多级编号：0、0.1、0.2、0.2.1…

埋点和输出日志


无线客户端请求
Web网页请求
开放平台API 请求
应用A
0.2
消息服务器
0.1
0.3
0.2.1
0.2.2
0.2.3
应用B
应用C
应用D
应用E
0.1.1
0.1.2
0.3.1
0.2.2.1
0.2.2.2
0.2.3.1
应用F
应用G
应用H
0.1.1.1
0.3.1.1
0.1.2.1
0.2.3.1.1
0.2.3.1.2
数据库
分布式缓存
分布式文件系统

埋点和输出日志

- 输出日志时面临的挑战
  - 减少对业务线程的影响，降低资源消耗
  - 每个网络请求至少 1 行日志，QPS 越高日志产生越快

埋点和输出日志

- 解决方案：自己实现日志输出
  - 异步线程写日志
  - 对调用链做采样
  - 开关控制
  - 对服务等长字符串做编码
  - 日志输出缓存，限制 IO 次数，每秒刷新
  - 日志文件按大小滚动，自动清理
  - 统一字符编码，统一时区

整体实现介绍

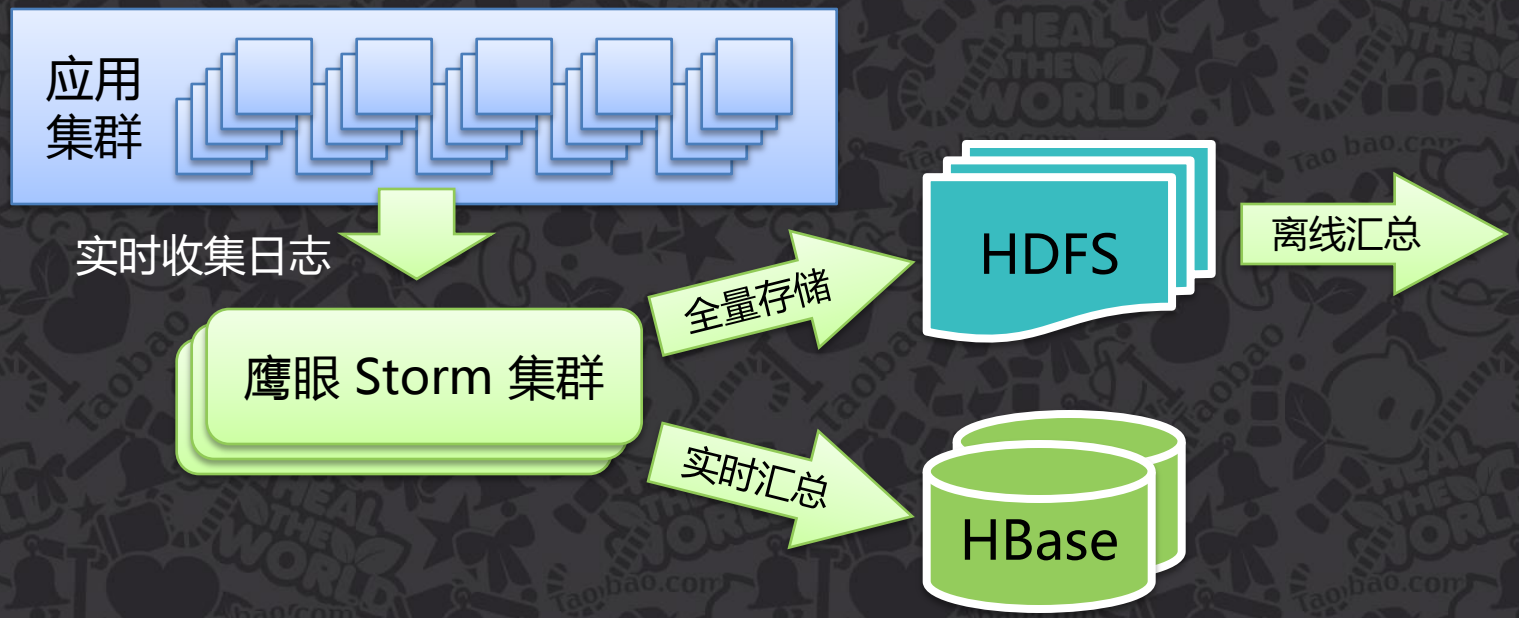
1. 埋点和输出日志
2. 收集和存储日志
3. 分析调用链

收集和存储日志

- 实时收集日志
  - 调用链的日志分散在调用经过的各个服务器上
  - 离线分析需要将同一条调用链的日志**汇总**在一起

收集和存储日志

- 按 TraceId 汇总日志
  - 基于 数据库：用 TraceId 关联查询
  - 基于 HBase：TraceId 做 rowkey，实时性强
  - 基于 HDFS：顺序存储，后续 MapReduce 汇总



整体实现介绍

1. 埋点和输出日志
2. 收集和存储日志
3. 分析调用链

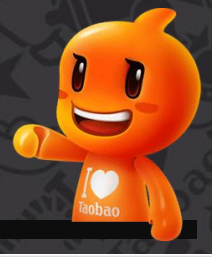
分析调用链

- 基于入口对调用链做链路分析
  - 入口是调用链的源头
  - 同一个入口背后走的是同一套业务逻辑
    - 入口的多条调用链 → 入口链路
  - 如果业务代码没更新，链路的执行逻辑一般不会改变

分析调用链

- 离线分析
  - 多条调用日志 汇总（ by TraceId ）
  - 还原调用关系 → 重组（ by RpcId ）
  - 分析链路
- 实时分析
  - 借助离线分析的产出，如链路形态、依赖情况等
  - 对**单条调用日志**直接分析，不需要汇总、重组
  - 得到链路上的调用情况，如 QPS、 RT、 错误情况

分析调用链



离线分析得到链路形态

实时分析回填调用数据

离线分析产出其他信息

					QPS					依赖度	耗时比例	标记	
根	[redacted]				118.13	269.64	1.0	435ms	51ms	100.0%	11.88%		
1	[redacted]				226.02	483.12	1.94	8ms	8ms	98.51%	3.64%		
2	[redacted]				282.74	533.96	4.06	0ms	0ms	58.94%	0.02%		
2	[redacted]				74.41	170.57	1.77	0ms	0ms	35.56%	0.08%		
1	[redacted]				183.31	357.95	1.71	3ms	2ms	90.58%	1.0%		
2	[redacted]				182.59	356.96	1.71	0ms	0ms	90.28%	0.32%		
1	[redacted]				130.91	244.85	1.71	19ms	7ms	64.79%	1.98%		
2	[redacted]				219.58	412.81	2.88	3ms	3ms	64.52%	1.37%		
2	[redacted]				219.55	412.73	2.88	1ms	1ms	64.51%	0.45%		
2	[redacted]				131.28	243.79	1.72	0ms	0ms	64.51%	0.12%		
2	[redacted]				125.25	231.11	1.7	0ms	0ms	62.2%	0.12%		
1	[redacted]				124.27	230.45	1.1	25ms	19ms	96.07%	4.66%	强依赖	
2	[redacted]				120.47	223.52	1.07	5ms	5ms	95.48%	1.3%		
1	[redacted]				116.05	247.93	1.0	4ms	4ms	98.25%	1.05%		
1	[redacted]				114.76	216.99	1.08	43ms	37ms	90.14%	8.28%	强依赖	
1	[redacted]				113.49	214.98	1.0	5ms	5ms	96.08%	1.16%	强依赖	
1	[redacted]				111.23	236.02	1.03	3ms	3ms	91.82%	0.68%	强依赖	
1	[redacted]				111.2	208.29	1.05	13ms	13ms	89.46%	2.88%		
1	[redacted]				110.24	213.12	1.0	214ms	207ms	93.32%	44.5%	瓶颈	
2	[redacted]				116.92	220.86	1.07	5ms	5ms	92.59%	1.28%		
1	[redacted]				106.5	205.27	1.0	3ms	3ms	90.16%	0.81%		

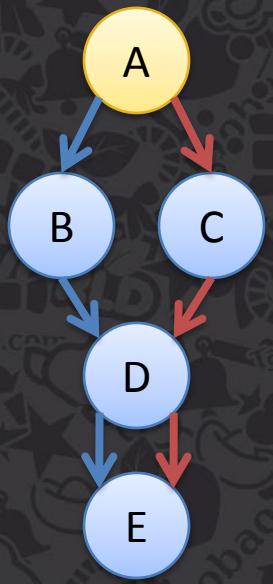
如何根据单条调用日志定位调用所在的链路位置？

分析调用链

- Step1：定位入口链路
  - 入口应用：可以从 TraceId 隐藏的业务数据得到
  - 入口签名：每行日志都带的 4 字节 ID
    - 在创建链路时生成，用于标识入口
- Step2：定位在链路中的具体位置
  - 链路签名：每行日志都带的 4 字节 ID
    - 请求到达服务端时，在原签名上叠加当前签名

分析调用链

- 入口签名与链路签名的计算方法



链路层次	入口签名	链路签名
A	*h(A)*	0
A → B	*h(A)*	*h(B)*
A → C	*h(A)*	*h(C)*
A → B → D	*h(A)*	*h(B)* * 31 + *h(D)*
A → C → D	*h(A)*	*h(C)* * 31 + *h(D)*
A → B → D → E	*h(A)*	(*h(B)* * 31 + *h(D)*) *31 + *h(E)*
A → C → D → E	*h(A)*	(*h(C)* * 31 + *h(D)*) *31 + *h(E)*

鹰眼的实现小结

1. 埋点和输出日志
   - 中间件埋点，基于 ThreadLocal
   - 异步写，采样
2. 收集和存储日志
   - 实时抓日志，按 TraceId 汇总，不同的存储方式
3. 分析调用链
   - 基于入口的链路分析
   - 实时分析：入口和链路签名

谢谢

淘宝网 **司徒放**（姬风）

jifeng@taobao.com