监控埋点，以终为始

吴其敏

平安银行 零售首席架构师

QCon 上海 2019

自我介绍

- 20多年程序员，互联网老兵，坚持写代码
- 追求简单，喜欢造轮子
- 曾在易趣、eBay、大众点评、携程等互联网公司工作
- 现在平安银行，平台架构部负责人
- 开源分布式实时监控CAT 作者
  - 每天万亿级消息处理能力（美团点评、携程网、平安银行等等）
  - 2012年开源至今，已有数百家公司在使用

CAT开源

dianping / cat

Unwatch 1,086 | Unstar 11,530 | Fork 3,862

Code | Issues 44 | Pull requests 3 | Projects 0 | Wiki | Security | Insights | Settings

CAT 作为服务端项目基础组件，提供了 Java, C/C++, Node.js, Python, Go 等多语言客户端，已经在美团点评的基础架构中间件框架（MVC框架，RPC框架，数据库框架，缓存框架等，消息队列，配置系统等）深度集成，为美团点评各业务线提供系统丰富的性能指标、健康状况、实时告警等。

Edit

apm java distributed realtime monitoring tracing metrics Manage topics

7,261 commits | 7 branches | 7 releases | 67 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

qmwu2000 Update Dockerfile-cn — Latest commit eb4c14e 3 minutes ago

.github/ISSUE_TEMPLATE	Update issue templates	6 minutes ago
cat-alarm	【bug fix】- 服务端配置了采样率必须要更改路由配置才能生效	7 days ago
cat-client	Add unit tests for com.site.lookup.util.StringUtils	3 months ago
cat-consumer	Fix typos found in the code base to make sure the most accurate Engli...	5 months ago
cat-core	Fix typos found in the code base to make sure the most accurate Engli...	5 months ago
cat-hadoop	从Cat.getCatHome()获取路径而不是写死！	11 months ago
cat-home	【Transaction告警、Event告警、心跳告警、异常告警】- 告警开关	14 days ago

http://github.com/dianping/cat

目录

关于故障

故障影响 = 故障影响面 X 故障发生的概率 X 故障持续时长

✤降低故障影响面：

- 高内聚低耦合、熔断限流、异步化设计、服务分层分级治理、同城双活、异地多活、…
- 低谷发布、功能开关、灰度能力、降级能力、…
- 系统预警、隔离部署、防止雪崩、…
- …

✤降低故障发生的概率：

- API设计、组件化设计、领域建模、…
- 代码审查、代码检测、…
- 环境标准化、单元测试、自动化测试、性能测试、破坏性测试、Chaos Monkey、…
- 灰度发布、统一变更、故障复盘、减少人工操作、…
- …

✤缩短故障持续时长：

- 完善监控告警体系
- 24小时值班计划
- 应急响应机制
- 依赖降级手段
- 紧急切换机制
- …

关于监控

业务监控：从客户角度评估对业务状态

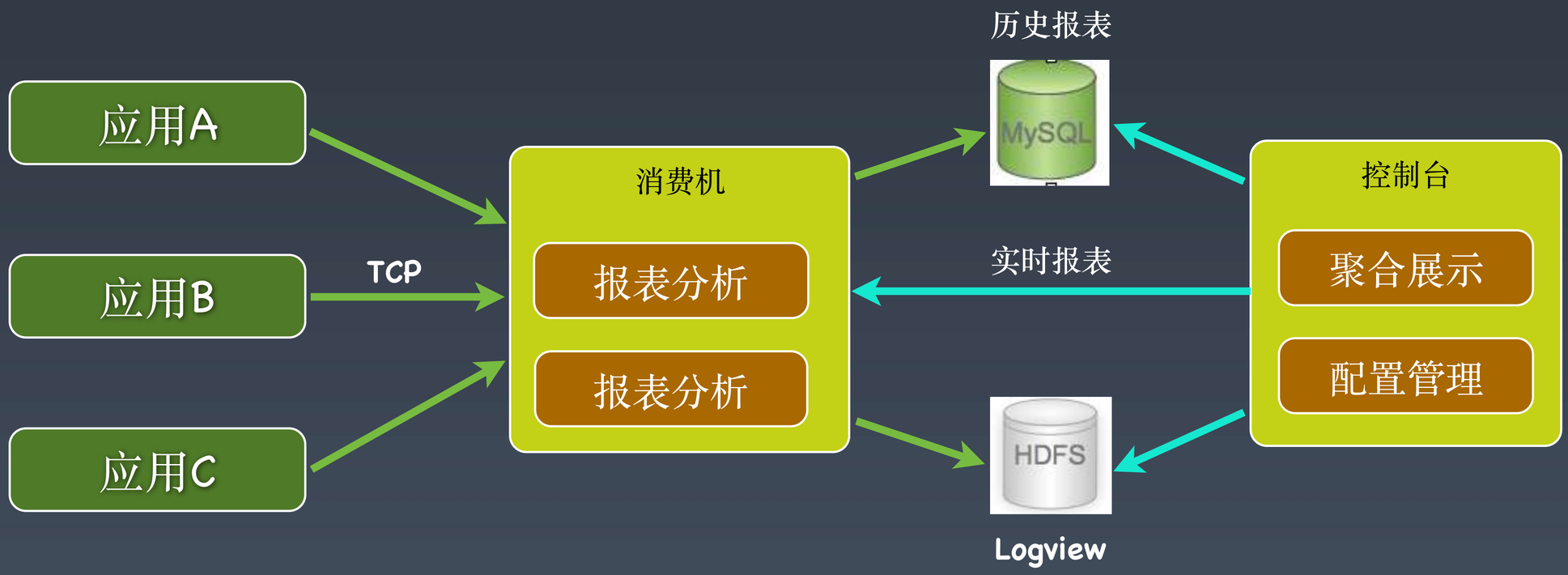
应用监控：从应用层面反映系统的状态

基础监控：从系统和资源层面反映系统的状态

对于监控，埋点是基础，决定监控的成败！

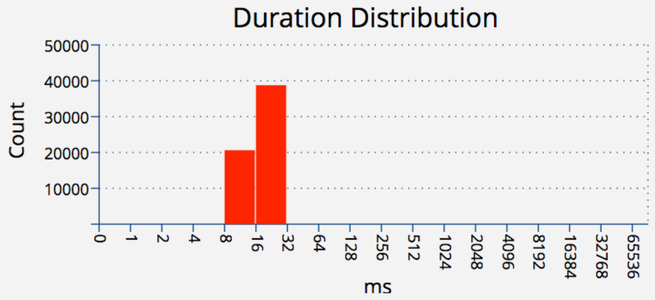
目录

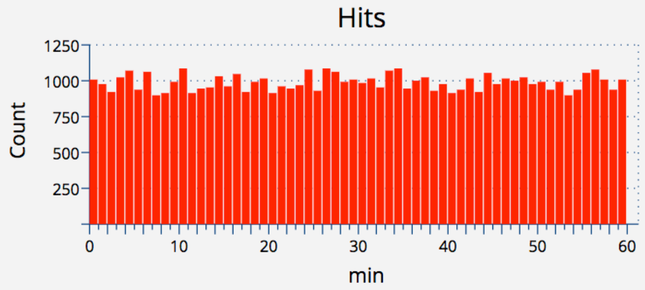
CAT架构—实时、海量、高效

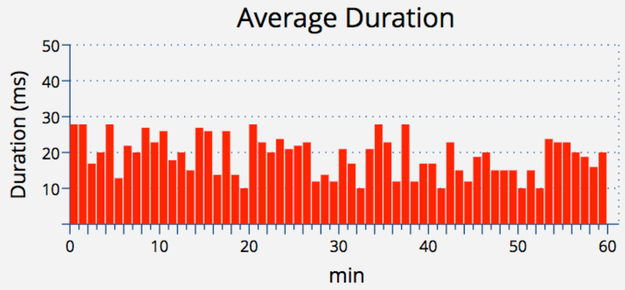


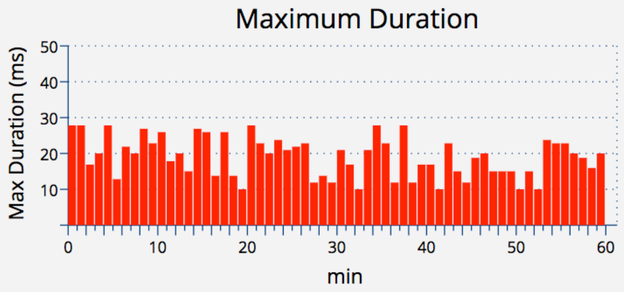
报表

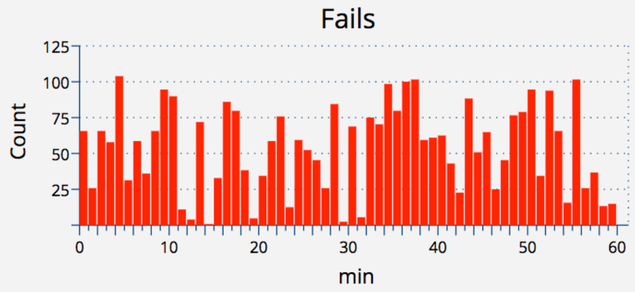
Type	Avg (ms)	Self (ms)	Total (ms)	Count	Fails	Fail %	Min (ms)	Max (ms)	QPS
[:: show ::] Service	20.7	20.7	1,232,625.0	59,621	3,127	5.2448%	10	29	993.7
[:: show ::] URL.Call	20.6	20.6	1,236,076.0	59,977	2,918	4.8652%	10	30	999.6
[:: show ::] SQL	19.9	19.9	1,185,752.0	59,722	2,889	4.8374%	10	30	995.4
[:: show ::] Service.Call	19.4	19.4	1,162,735.0	59,915	2,741	4.5748%	10	30	998.6
[:: hide ::] URL	19.2	19.2	1,143,319.0	59,508	3,269	5.4934%	10	28	991.8

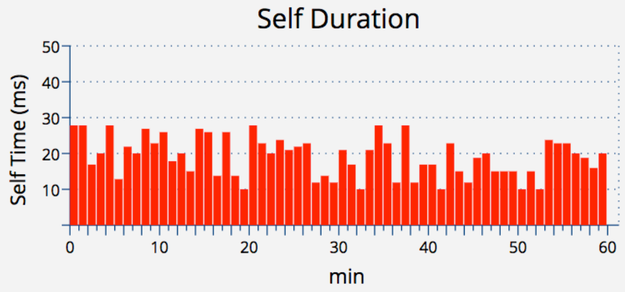






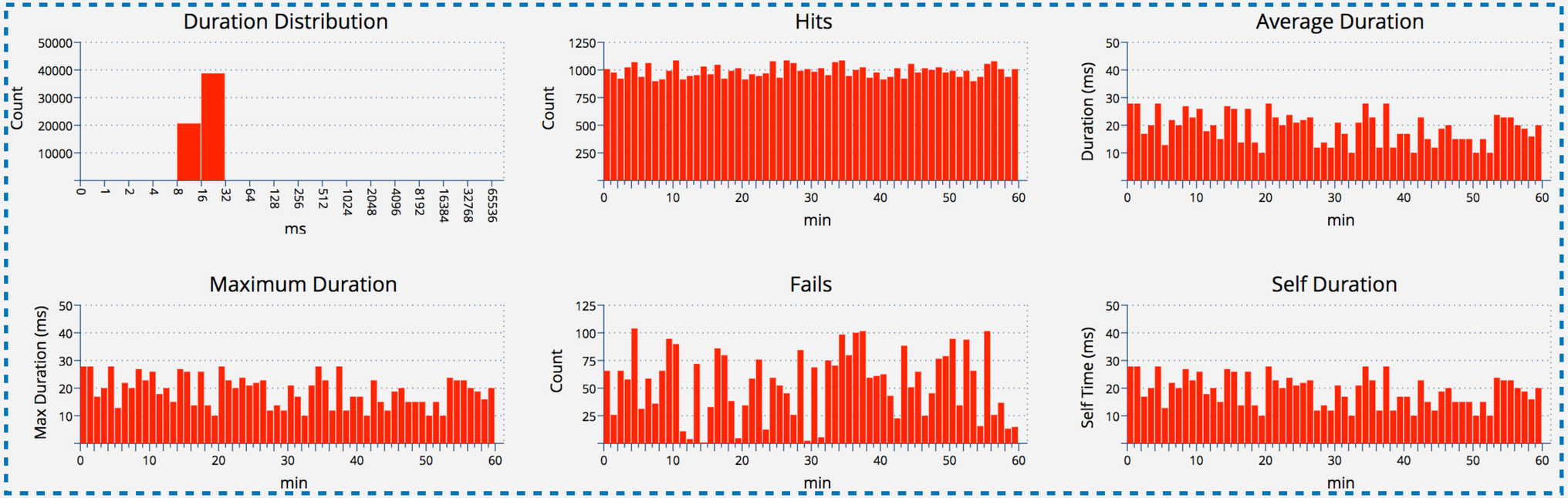






关注点

Type	Avg (ms)	Self (ms)	Total (ms)	Count	Fails	Fail %	Min (ms)	Max (ms)	QPS
[:: show ::] Service	20.7	20.7	1,232,625.0	59,621	3,127	5.2448%	10	29	993.7
[:: show ::] URL.Call	20.6	20.6	1,236,076.0	59,977	2,918	4.8652%	10	30	999.6
[:: show ::] SQL	19.9	19.9	1,185,752.0	59,722	2,889	4.8374%	10	30	995.4
[:: show ::] Service.Call	19.4	19.4	1,162,735.0	59,915	2,741	4.5748%	10	30	998.6
[:: hide ::] URL	19.2	19.2	1,143,319.0	59,508	3,269	5.4934%	10	28	991.8

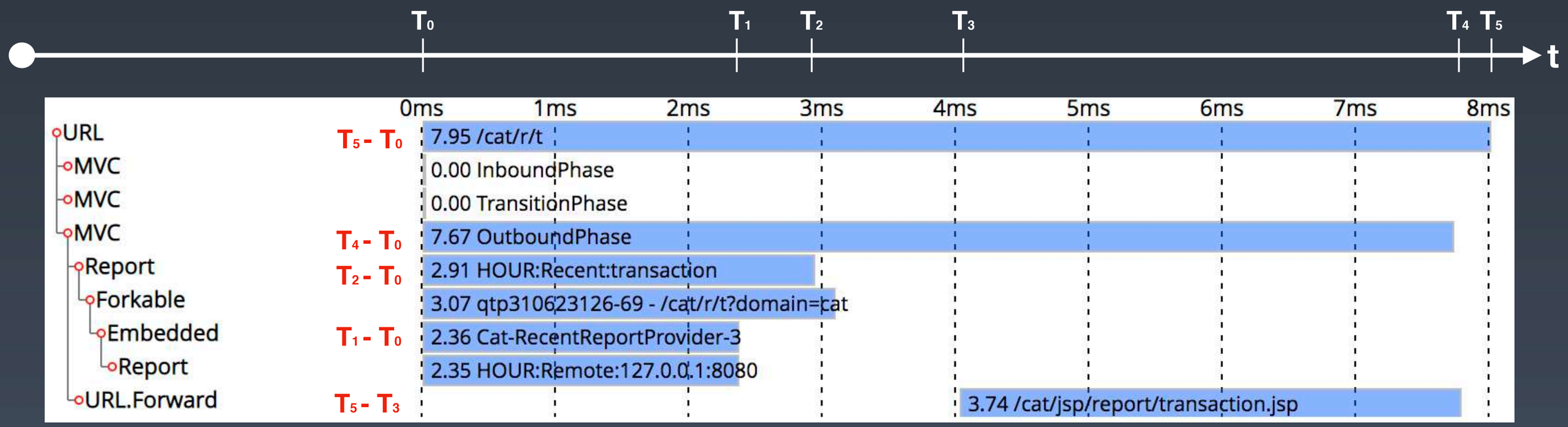


消息树

Timestamp	Type	Name	Status	Duration	Key value pairs
t19:35:10.208	URL	/cat/r/t			
E19:35:10.208	URL.Server	localhost			IPS=0:0:0:0:0:0:0:1&VirtualIP=0:0:0:0:0:0:0:1&Server=localhost&Referer=http://loca
E19:35:10.208	URL.Method	GET			HTTP/GET /cat/r/t?domain=cat
A19:35:10.208	MVC	InboundPhase			0.00ms
A19:35:10.208	MVC	TransitionPhase			0.00ms
t19:35:10.208	MVC	OutboundPhase			
t19:35:10.208	Report	HOUR:Recent:transaction			
t19:35:10.208	Forkable	qtp310623126-69 - /cat/r/t?domain=cat			
t19:35:10.208	Embedded	Cat-RecentReportProvider-3			
A19:35:10.208	Report	HOUR:Remote:127.0.0.1:8080			2.35ms
T19:35:10.210	Embedded	Cat-RecentReportProvider-3			2.36ms
T19:35:10.211	Forkable	qtp310623126-69 - /cat/r/t?domain=cat			3.07ms
T19:35:10.210	Report	HOUR:Recent:transaction			2.91ms /cat/r/service/transaction/cat/hour/2019100519/type?ip=All&count=1
t19:35:10.212	URL.Forward	/cat/jsp/report/transaction.jsp			
E19:35:10.212	URL.Forward.Method	GET			HTTP/GET /cat/jsp/report/transaction.jsp?domain=cat
T19:35:10.215	URL.Forward	/cat/jsp/report/transaction.jsp			3.74ms
T19:35:10.215	MVC	OutboundPhase			7.67ms
T19:35:10.215	URL	/cat/r/t			7.95ms module=r&in=t&out=t

原始数据（raw data）和关系（structure）

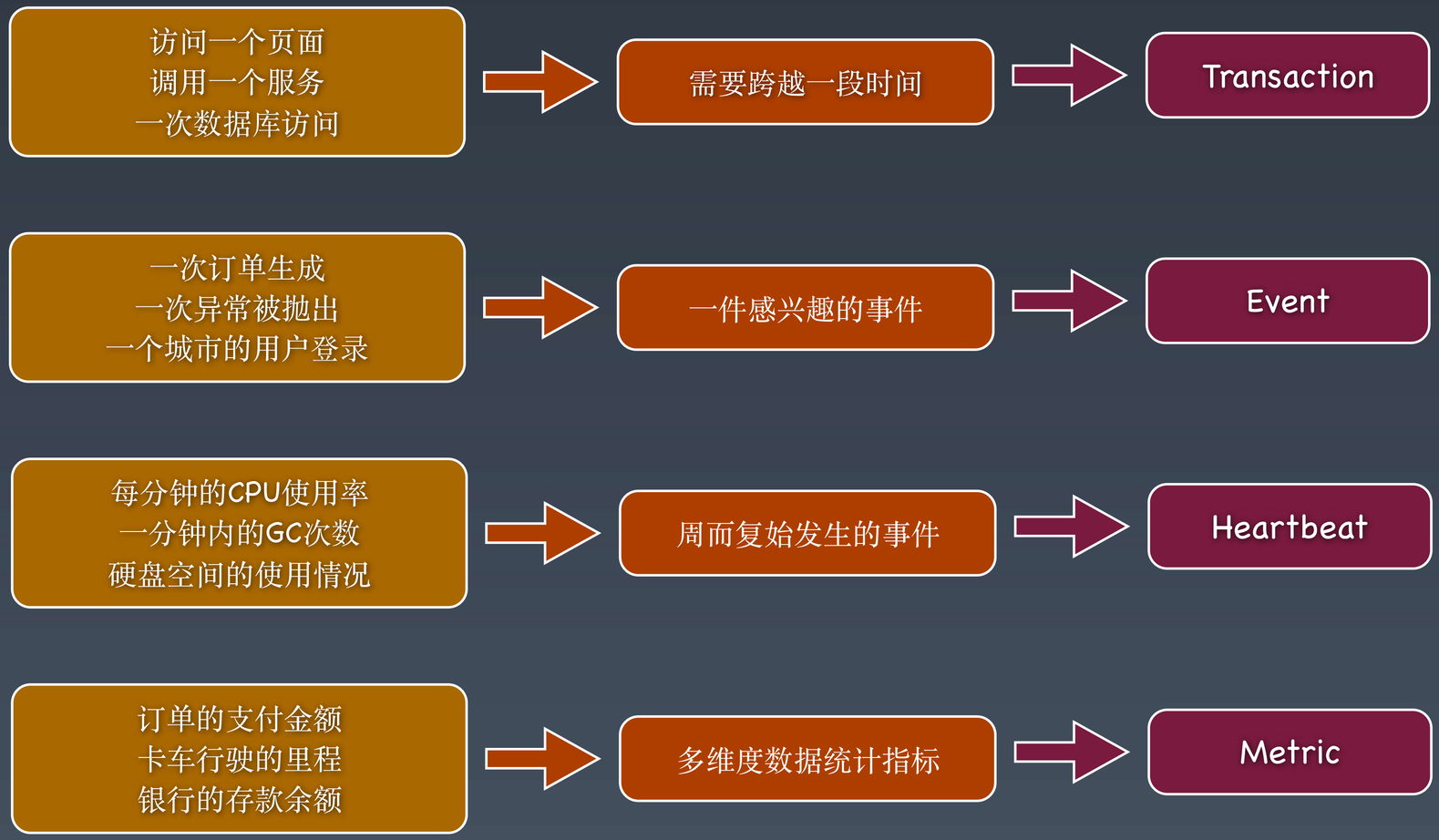
时间轴



通过对时间的不断细化，还原客观事实

目录

模型


访问一个页面
调用一个服务
一次数据库访问
需要跨越一段时间
Transaction
一次订单生成
一次异常被抛出
一个城市的用户登录
一件感兴趣的事件
Event
每分钟的CPU使用率
一分钟内的GC次数
硬盘空间的使用情况
周而复始发生的事件
Heartbeat
订单的支付金额
卡车行驶的里程
银行的存款余额
多维度数据统计指标
Metric

埋点API

```
Transaction t = Cat.newTransaction("URL", "/cat/r/t");

t.addData("module", "r");

Cat.logEvent("URL.Server", "localhost", "...");
Cat.logEvent("URL.Method", "GET", "...");

try {
    doInboundPhase();
    doTransitionPhase();
    doOutboundPhase();

    t.success();
} catch (Exception e) {
    t.setStatus(e);
} finally {
    t.complete();
}
```

// 记录：Timestamp，Type，Name

// 补充：Key value pairs（可多次）

// 内嵌：两个事件

// 进一步处理业务逻辑以及埋点

// 标记：Status - 成功

// 标记：Status - 失败及原因

// 计算：Duration

数据点

要素：

- 时间（Timestamp）
- 类别（Type & Name）
- 状态（Status）
- 耗时（Duration）
- 嵌套关系

上下文：

- 应用（Domain）
- 机器（IP）
- 进程（Process）
- 线程（Thread）

埋点四步法

1）目标设定

- 监控哪些指标?
- 解释哪些问题?

2）方案设计

- 做一个切合实际场景的埋点方案
- 采集预设的监控指标

3）实施落地

- 找到合适的代码修改点
- 按照设计方案完成数据埋点

4）验证反馈

- 验证报表中的结果是否符合预期?
- 回到第一步，优化迭代

埋点示例 - 用户登录

目标：

监控多种用户登录方式（次数、成功率和时长）

设计：

消息类型	Type	Name	Status
Transaction	UserLogin	<登录类型>	成功与否

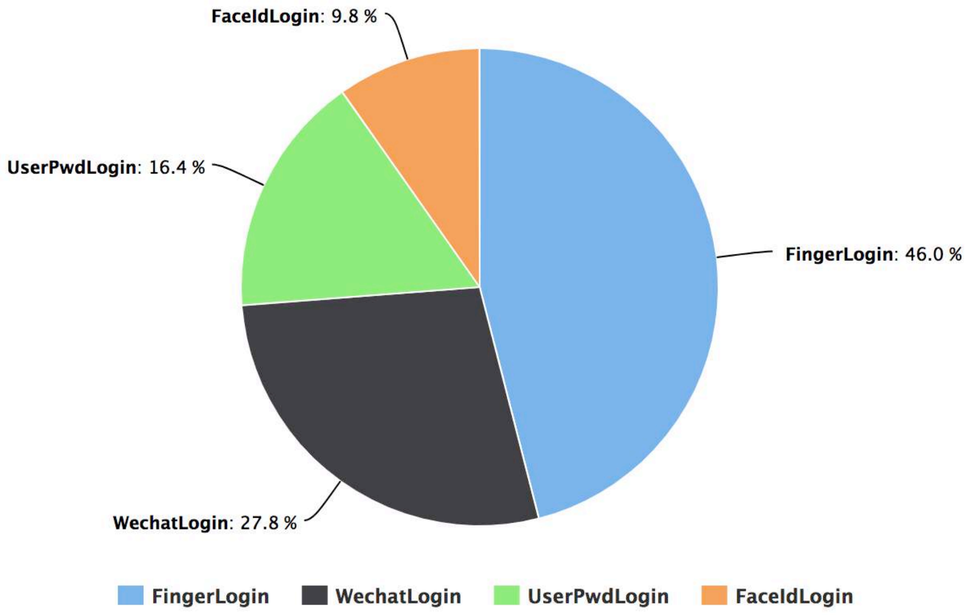
代码：

```
Transaction t = Cat.newTransaction("UserLogin", "UserPwdLogin");

try {
   userPasswordLogin(username, password); // perform login logic

   t.success();
} catch (LoginException e) {
   t.setStatus(e); // login failed due to user password mismatched
   throw e;
} catch (RuntimeException e) {
   t.setStatus(e); // login failed due to unexpected exception
   throw e;
} finally {
   t.complete();
}
```

Name	Avg (ms)	Self (ms)	Total (ms)	Count	Fails	Fail %	Min (ms)	Max (ms)	QPS	Percentage
[:: show ::] Type: UserLogin	95.8	95.8	35,187,957	367,493	4,364	1.1875%	41	178	6,124.9	100.00%
[:: show ::] WechatLogin	120.8	120.8	12,325,595	102,057	3,029	2.9679%	66	178	1,701.0	27.77%
[:: show ::] UserPwdLogin	97.9	97.9	5,919,427	60,435	1,136	1.8797%	50	145	1,007.2	16.45%
[:: show ::] FaceIdLogin	92.7	92.7	3,323,123	35,861	65	0.1813%	45	134	597.7	9.76%
[:: show ::] FingerLogin	80.5	80.5	13,619,812	169,140	134	0.0792%	41	116	2,819.0	46.03%



埋点示例 - 用户登录

目标：

监控多个用户登录级别和出错码（次数、成功率）

设计：

消息类型	Type	Name	Status
Event	UserLogin.Level	<登录级别>	成功与否
Event	UserLogin.ErrorCode	<出错码>	-

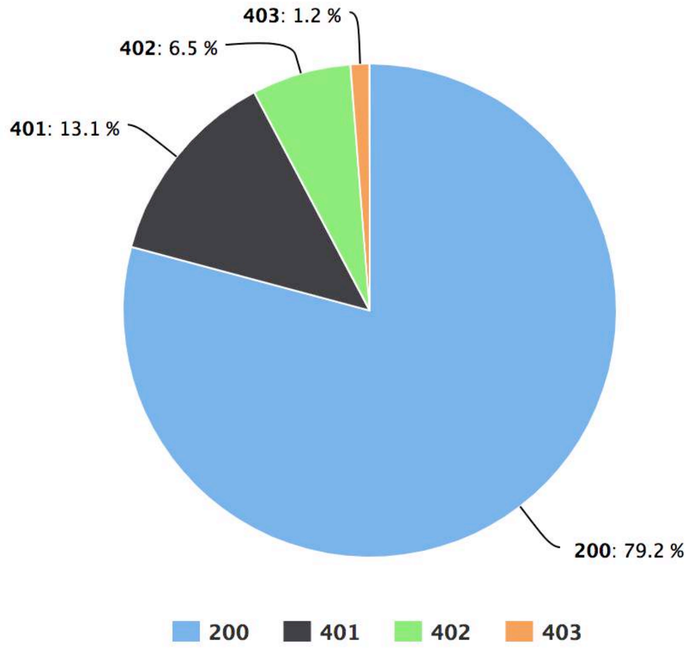
代码：

```
AuthToken authToken = getAuthenticationToken();

try {
   doUserLogin(authToken); // do real user login

   Cat.logEvent("UserLogin.Level", authToken.getLevel());
   Cat.logEvent("UserLogin.ErrorCode", "200");
} catch (LoginException e) {
   Cat.logEvent("UserLogin.Level", authToken.getLevel(),
         e.getClass().getSimpleName(), e.getMessage());
   Cat.logEvent("UserLogin.ErrorCode", e.getErrorCode());
}
```

Name		Count	Fails	Fail %	QPS	Percentage
[:: show ::]	Type: UserLogin.ErrorCode	45,793	0	0%	761.0	100.00%
[:: show ::]	200	36,249	0	0%	604.0	79.16%
[:: show ::]	401	5,998	0	0%	99.0	13.10%
[:: show ::]	402	2,978	0	0%	49.0	6.50%
[:: show ::]	403	568	0	0%	9.0	1.24%



埋点AOP

基于Annotation（arg0表示第一个参数，arg1表示第二个参数，以此类推）

```
@CatTransaction(type = "UserLogin", name = "UserPwdLogin", keys = "username", values = "${arg0}")
public void userPasswordLogin(String username, String password) throws LoginException {
    // do real user login
}
```

相当于

```
Transaction t = Cat.newTransaction("UserLogin", "UserPwdLogin");

t.addData("username", username);

try {
    userPasswordLogin(username, password);
} catch (LoginException e) {
    t.setStatus(e);
} finally {
    t.complete();
}
```

优点：

- 简洁、可读性好
- 代码实现
- 更新需发布生效

缺点：

- 功能受限
- 场景受限

埋点ASM

基于配置（arg0表示第一个参数，arg1表示第二个参数，以此类推）

```
<class name="org.unidal.cat.demo.instrument.UserLogin">
   <method name="userPasswordLogin" desc="(Ljava/lang/String;Ljava/lang/String;)V">
      <transaction type="UserLogin" name="UserPwdLogin">
         <key>username</key>
         <value>${arg0}</value>
      </transaction>
   </method>
</class>
```

相当于

```
@CatTransaction(type = "UserLogin", name = "UserPwdLogin", keys = "username", values = "${arg0}")
public void userPasswordLogin(String username, String password) throws LoginException {
   // do real user login
}
```

优点：

- 无代码侵入
- 对源码无要求
- 可多次热加载

缺点：

- 调试不友好

埋点ASM

基于Java（可独立打成jar包）

```
@MixinMeta(targetClass = UserLogin.class)
public class UserLoginOverride {
   private void userPasswordLogin(String username, String password) throws LoginException {
      Transaction t = Cat.newTransaction("UserLogin", "UserPwdLogin");

      t.addData("username", username);

      try {
         $_userPasswordLogin(username, password);
      } catch (LoginException e) {
         t.setStatus(e);
         throw e;
      } catch (RuntimeException e) {
         t.setStatus(e);
         throw e;
      } finally {
         t.complete();
      }
   }

   private void $_userPasswordLogin(String username, String password) throws LoginException {
      // do real user login
   }
}
```

相当于

```
@CatTransaction(type = "UserLogin", name = "UserPwdLogin", keys = "username", values = "${arg0}")
public void userPasswordLogin(String username, String password) throws LoginException {
   // do real user login
}
```

优点：

- 功能强大
- 适用范围广
- 无代码侵入
- 对源码无要求

缺点：

- 只能加载一次
- 调试不友好

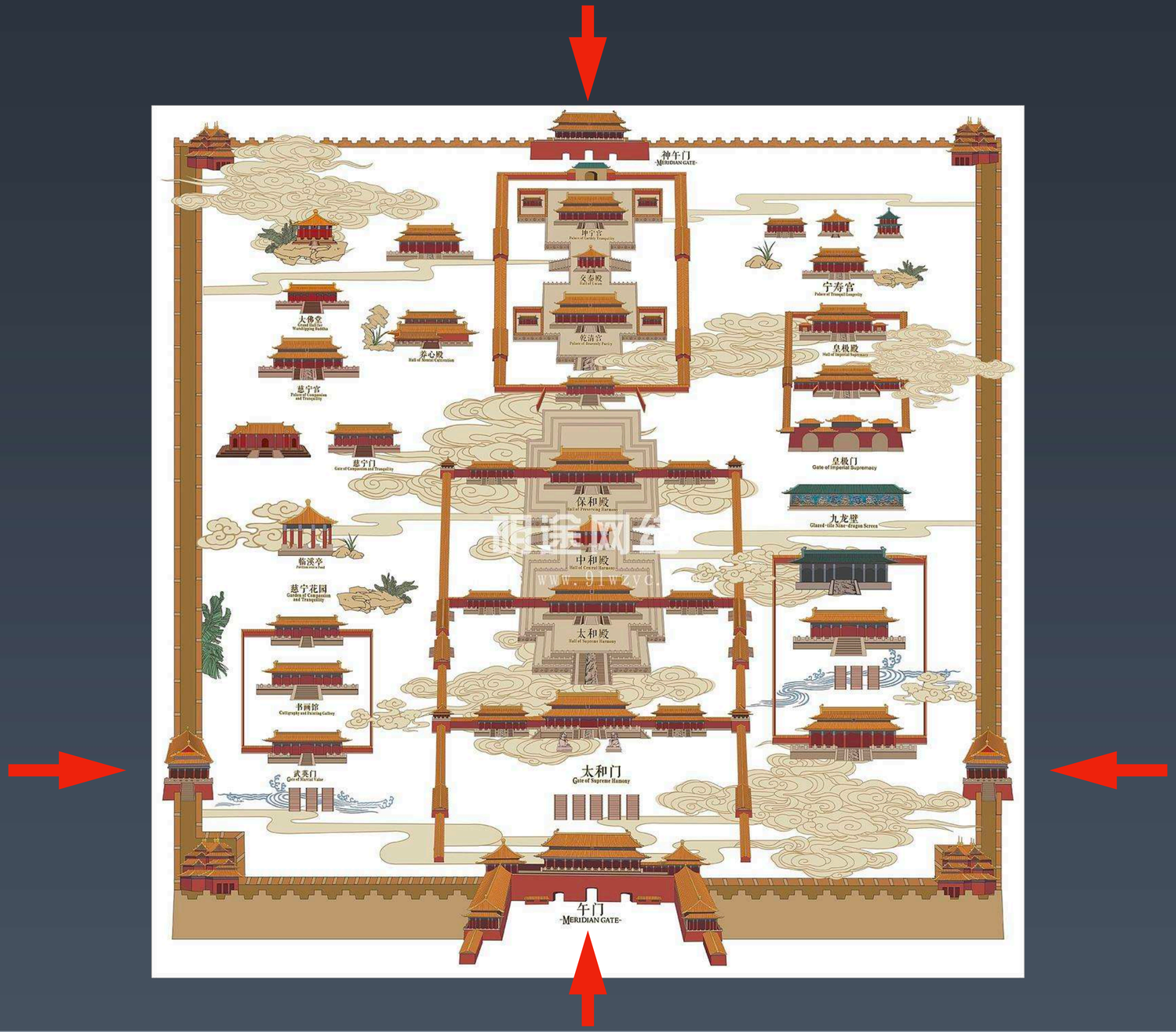
埋点方法 - 小结

方法	优点	缺点
API	❖ 高性能 ❖ 可调试 ❖ 代码灵活 ❖ 功能强大 ❖ 适用于复杂场景，如框架、中间件、老系统	❖ 依赖源代码 ❖ 侵入性太强 ❖ 有开发成本 ❖ 变更需发布
AOP	❖ 可读性好 ❖ 弱代码侵入	❖ 依赖源代码 ❖ 变更需发布 ❖ 对代码结构有要求 ❖ 适用于简单场景
ASM 配置	❖ 无源代码要求 ❖ 无代码侵入 ❖ 动态挂载/卸载 ❖ 适用于第三方库	❖ 对代码结构有要求 ❖ 适用于简单场景
ASM 代码	❖ 功能强大 ❖ 适用范围广 ❖ 无代码侵入 ❖ 对源码无要求	❖ 只能加载一次 ❖ 调试不友好

埋点原则

“不确定的” 要埋点

“不符合预期的” 是问题



常见埋点

- 入口（Inbound）
  - Web（Page Controller）
  - API（Rest Controller）
  - RPC（RPC Provider）
  - MQ（Message Consumer)
  - Job（Job Agent）
- 出口（Outbound）
  - HTTP Call（i.e. Rest API）
  - RPC Call（i.e. Dubbo）
  - DAL（Data Access Layer）
  - Cache（Redis & Memcache）
  - MQ（Message Producer）
- 网关（Gateway）

目录

埋点范例 - HTTP服务端

Message	Type	Name	Key Value Pairs	Comments
t	HTTP	<path>		页面被访问的次数、成功率和耗时 <path>: /cat/r/t
E	HTTP.Server	<domain>	ClientIP=<client ip>&Referer=<referer> &Agent=<agent>	被访问域名的分布 <client-ip>: 10.0.0.1 <domain>: www.example.com <agent>: Mozilla/5.0
E	HTTP.Method	<method>	<url>	HTTP方法的次数和比例 <method>: GET <url>: http://www.example.com/cat/r/t?domain=cat
E	HTTP.Status	<status>		结果的状态码分布 <status>: 200
T	HTTP	<uri-path>		

埋点范例 - HTTP客户端

Message	Type	Name	Key Value Pairs	Comments
t	HTTP.Client	<uri>	<query-string>	目标页面的次数、成功率和耗时 <uri>: http://localhost:8080/cat/r/t <query-string>: query string after '?'
E	HTTP.Client.Method	<method>	reqSize=<req-size>&agent=<agent>	HTTP方法的次数和比例 <req-size>: 123 <agent>: Apache-HttpClient-4.0.0
E	HTTP.Client.Status	<status>		结果的状态码分布 <status>: 200
E	HTTP.Client.ResSize	<size-range>	size=<size>	结果的大小分布 <size-range>: [4K, 8K) <size>: 5378
T	HTTP.Client	<uri>		

埋点范例 - RPC服务端

Message	Type	Name	Key Value Pairs	Comments
t	Service	<service-name>		服务被访问的次数、成功率和耗时 <service-name>: OrderService.placeOrder
E	Service.ReqSize	<size-range>	size=<size>	请求体的大小分布 <size-range>: [4K, 8K) <size>: 5378
E	Service.ResSize	<size-range>	size=<size>	响应体的大小分布 <size-range>: [8K, 16K) <size>: 10687
E	Service.Status	<status>		服务状态码分布 <status>: 200
T	Service	<service-name>		

埋点范例 - RPC客户端

Message	Type	Name	Key Value Pairs	Comments
t	Service.Client	<service-name>		目标服务的调用次数、成功率和耗时 <service-name>: OrderService.placeOrder
E	Service.Client.ReqSize	<size-range>	size=<size>	请求体的大小分布 <size-range>: [2K, 4K) <size>: 3670
E	Service.Client.Status	<status>		服务调用的状态码分布 <status>: 200
T	Service.Client	<service-name>		

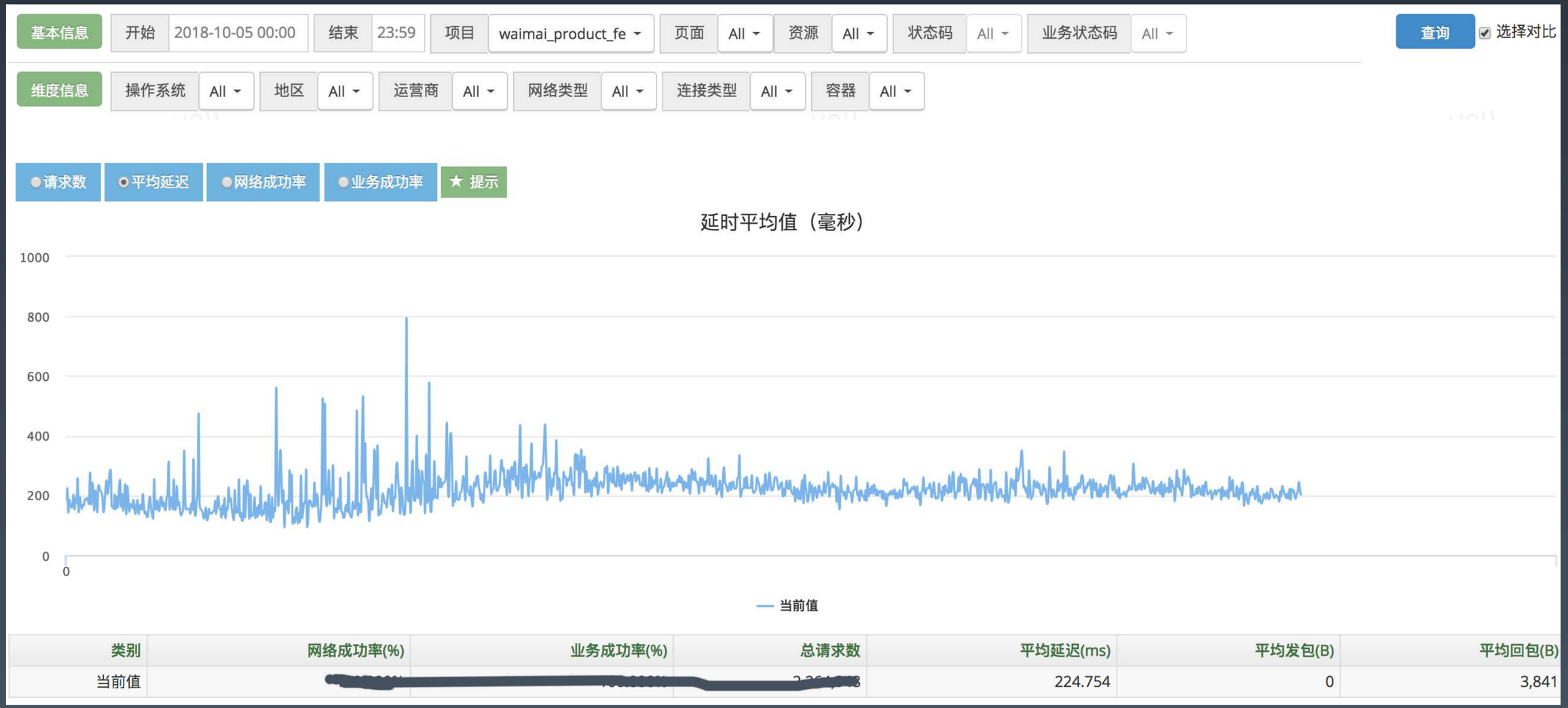
埋点范例 - DAL客户端

Message	Type	Name	Key Value Pairs	Comments
t	SQL	<query-name>	<params>	数据库查询的次数、成功率和耗时 <query-name>: OrderDAO.findByPK <params>: 1234
E	SQL.DataSource	<data-source>	<connection-properties>	数据源分布及连接串信息 <data-source>: jdbc:mysql://10.0.0.2:3306/cat <connection-properties>: useSSL=true&autoReconnect=true
E	SQL.Method	<method>		查询方法分布 <method>: SELECT|UPDATE|INSERT|DELETE|SP
E	SQL.Rows	<size-range>	size=<size>	结果集大小分布 <size-range>: [8, 16) <size>: 10
T	SQL	<query-name>		

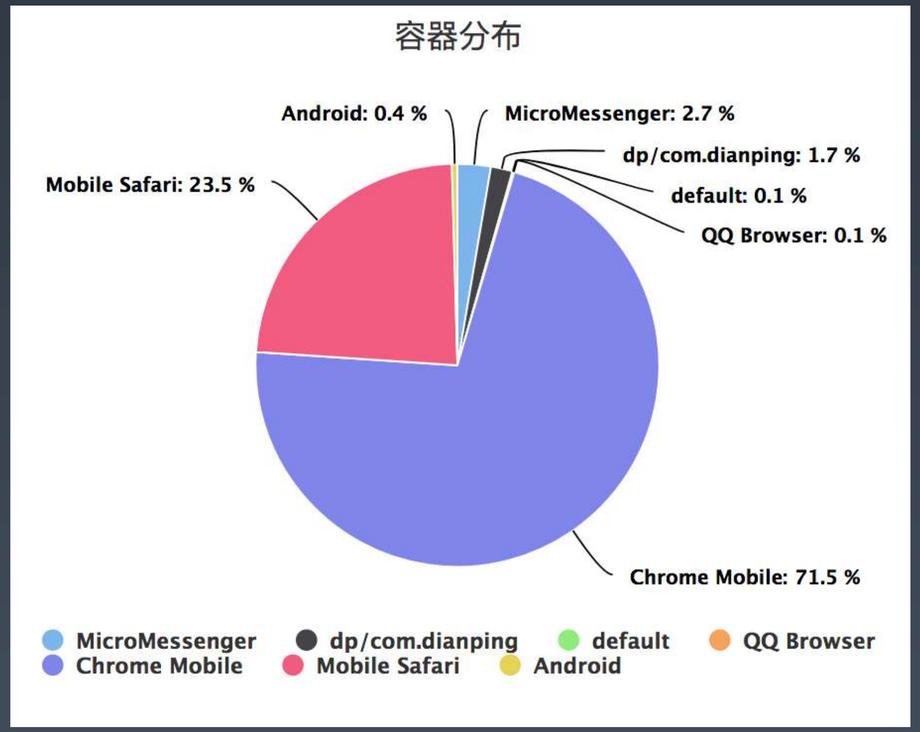
埋点范例 - Redis客户端

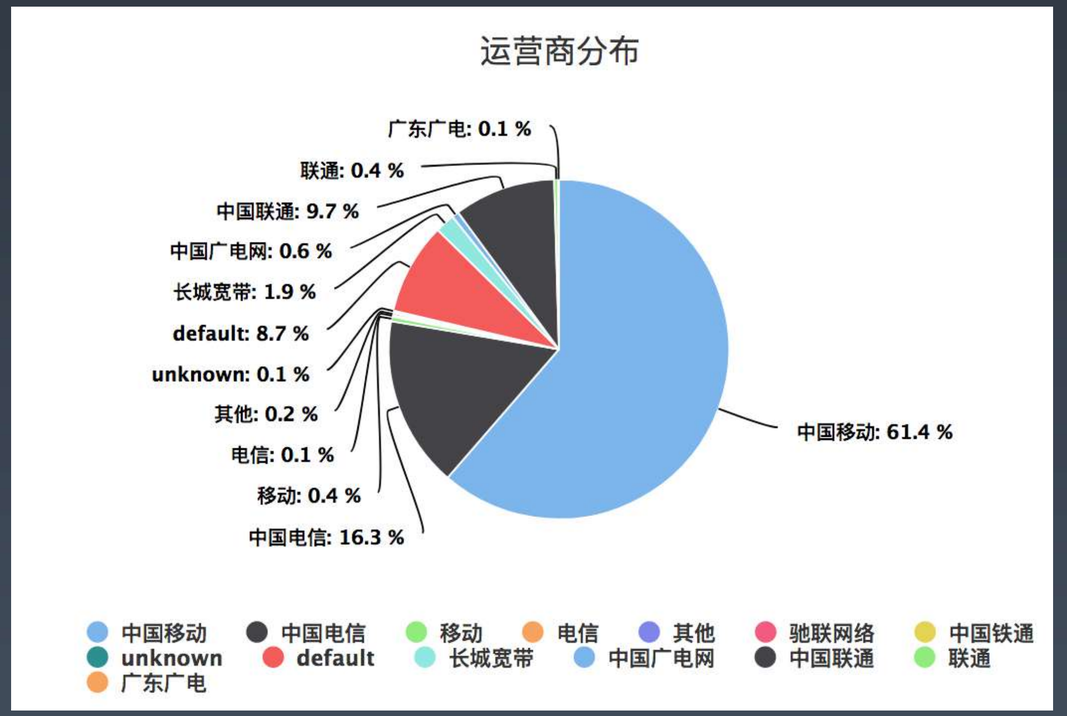
Message	Type	Name	Key Value Pairs	Comments
t	Redis	<query-name>	<params>	缓存查询的次数、成功率和耗时 <query-name>: Order.get <params>: 1234
E	Redis.DataSource	<data-source>	<connection-properties>	数据源分布和连接串信息 <data-source>: order://10.0.0.3:6379/0 <connection-properties>: maxIdle=10
E	Redis.Method	<method>		访问方法分布 <method>: get|set|mget|…
E	Redis.ResSize	<size-range>	size=<size>	结果集大小分布 <size-range>: [8K, 16K) <size>: 12345
T	Redis	<query-name>		

更多的埋点场景

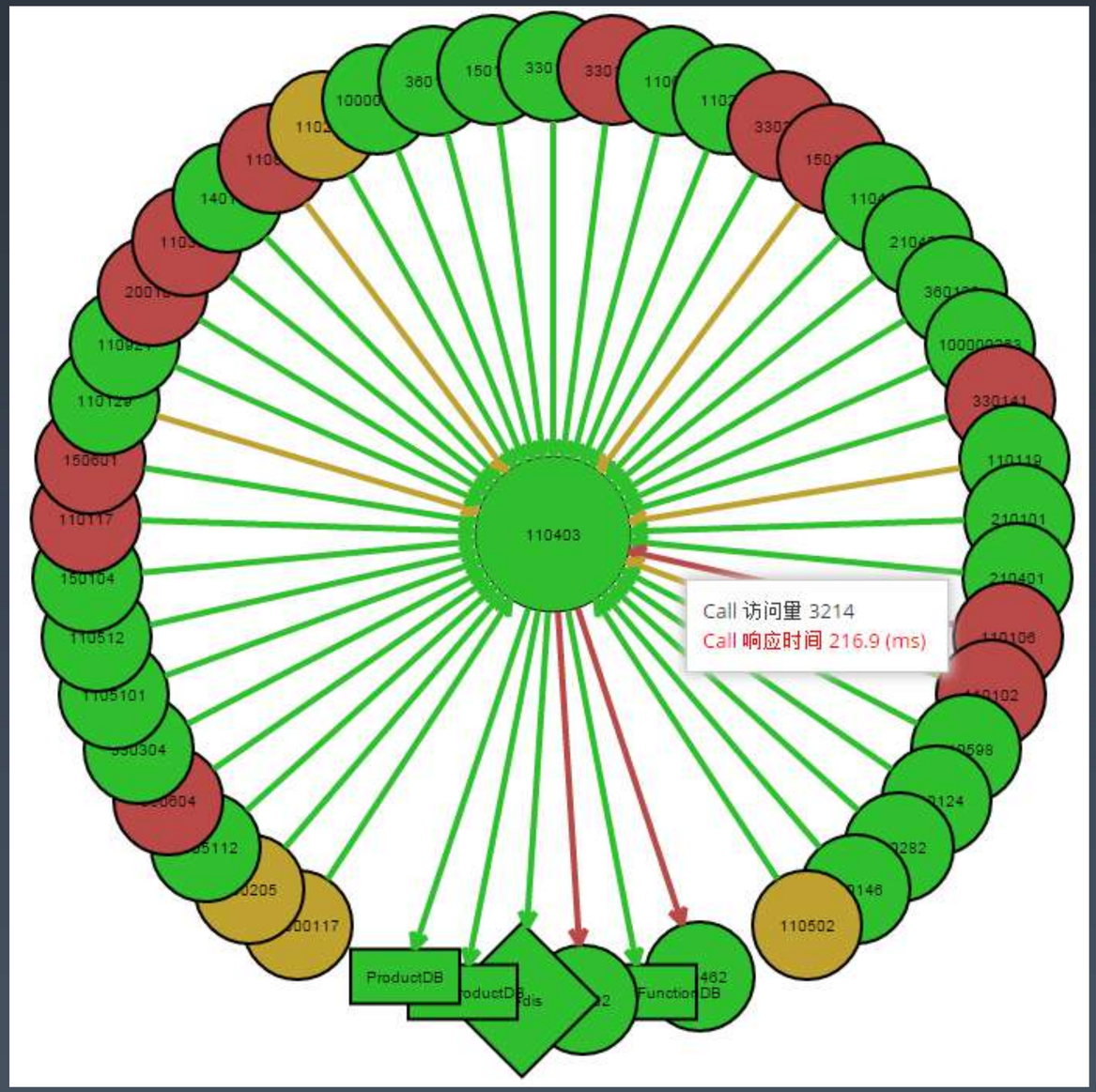

基本信息
开始
2018-10-05 00:00
结束
23:59
项目
waimai_product_fe
页面
All
资源
All
状态码
All
业务状态码
All
查询
选择对比
维度信息
操作系统
All
地区
All
运营商
All
网络类型
All
连接类型
All
容器
All
请求数
平均延迟
网络成功率
业务成功率
★ 提示
延时平均值（毫秒）
1000
800
600
400
200
0
0
当前值
类别
网络成功率(%)
业务成功率(%)
总请求数
平均延迟(ms)
平均发包(B)
平均回包(B)
当前值
224.754
0
3,841

更多的埋点场景

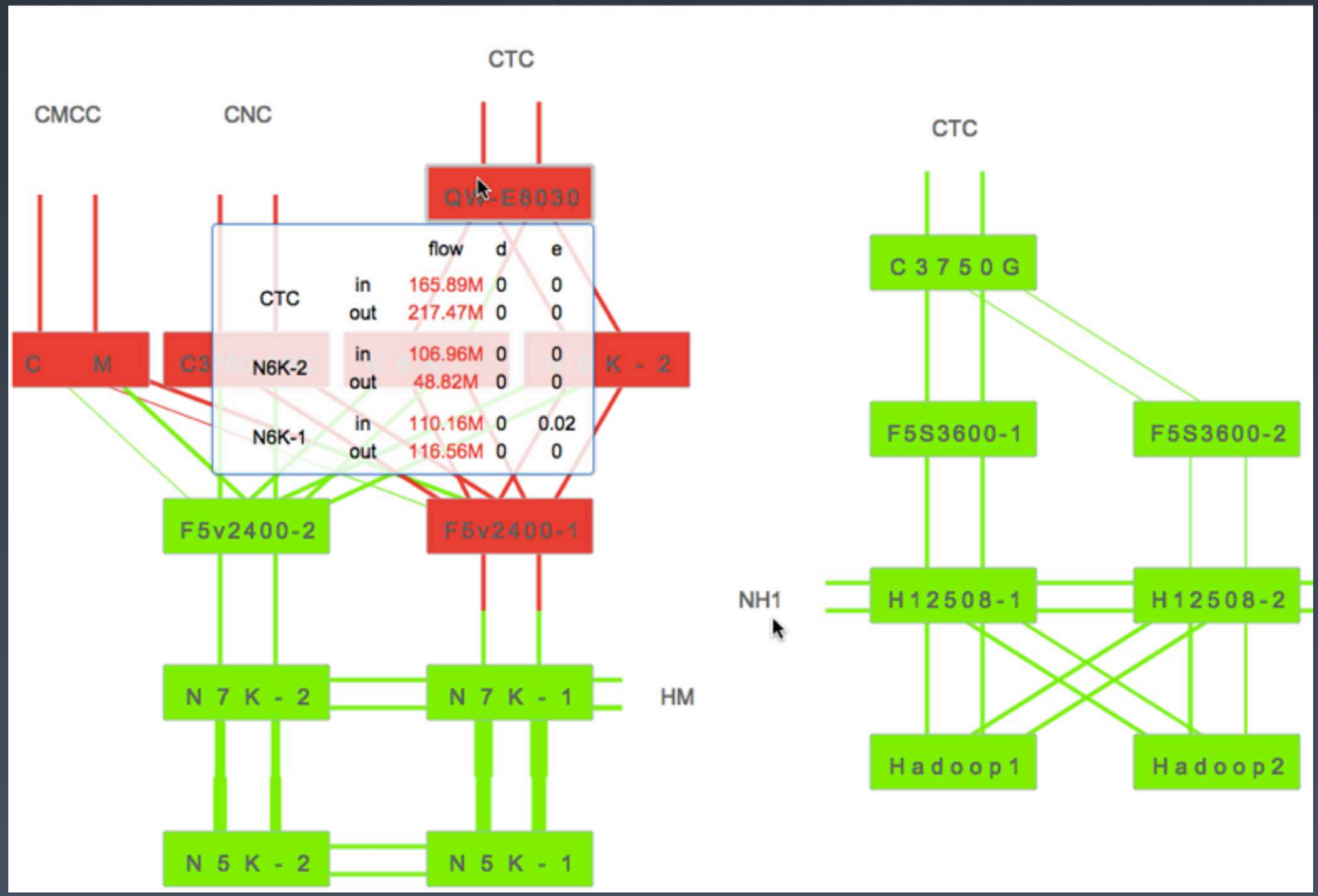

容器分布
Android: 0.4 %
MicroMessenger: 2.7 %
dp/com.dianping: 1.7 %
default: 0.1 %
QQ Browser: 0.1 %
Mobile Safari: 23.5 %
Chrome Mobile: 71.5 %
MicroMessenger
dp/com.dianping
default
QQ Browser
Chrome Mobile
Mobile Safari
Android


运营商分布
广东广电: 0.1 %
联通: 0.4 %
中国联通: 9.7 %
中国广电网: 0.6 %
长城宽带: 1.9 %
default: 8.7 %
unknown: 0.1 %
其他: 0.2 %
电信: 0.1 %
移动: 0.4 %
中国电信: 16.3 %
中国移动: 61.4 %
中国移动
中国电信
移动
电信
其他
驰联网络
中国铁通
unknown
default
长城宽带
中国广电网
中国联通
联通
广东广电

更多的埋点场景



更多的埋点场景


CTC
CMCC
CNC
CTC
QW-E8030
flow d e
CTC
in 165.89M 0 0
out 217.47M 0 0
N6K-2
in 106.96M 0 0
out 48.82M 0 0
N6K-1
in 110.16M 0 0.02
out 116.56M 0 0
C M
K - 2
F5v2400-2
F5v2400-1
N 7 K - 2
N 7 K - 1
HM
N 5 K - 2
N 5 K - 1
C 3 7 5 0 G
F5S3600-1
F5S3600-2
NH1
H12508-1
H12508-2
Hadoop1
Hadoop2

小结

- 故障影响：故障影响面，故障发生的概率，故障持续时长
- 埋点关注：时间序列的数据和关系
- 埋点步骤：目标设定，方案设计，实施落地，验证反馈
- 埋点方式：侵入式（客户端SDK和AOP配置等），非侵入式（字节码）
- 监控：以 “问题” 为中心，不符合 “预期” 的是问题
- 埋点：以 “不确定性” 为中心

THANKS!