基于代码覆盖率、程序调用链分析的精准测试探索

360Qtest 顾利萍

目录

CONTENTS

1 背景介绍

2 解决方案

3 应用现状

4 未来规划

背景介绍

测试之痛

迭代一个紧接一个

人手不够啊

RD：改的地方不少，保险起见，把相关功能都回归一遍吧！

改了这么多代码，该回归哪些功能？

测了半天，到底还有哪些代码没覆盖到？

全量自动化回归下吧，用例通过率不高，各种排查。。。

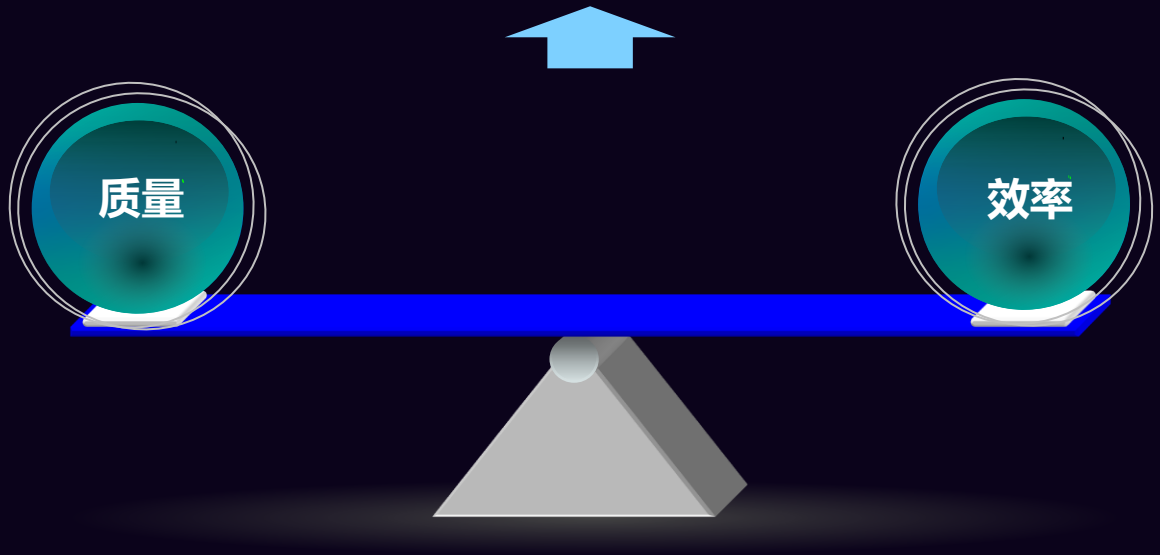
RD：测完了吗？
QA：没有，我需要把用例都回归下；
RD：只改几行代码，需要测这么多吗？

上线当日，心里忐忑+忐忑... bless

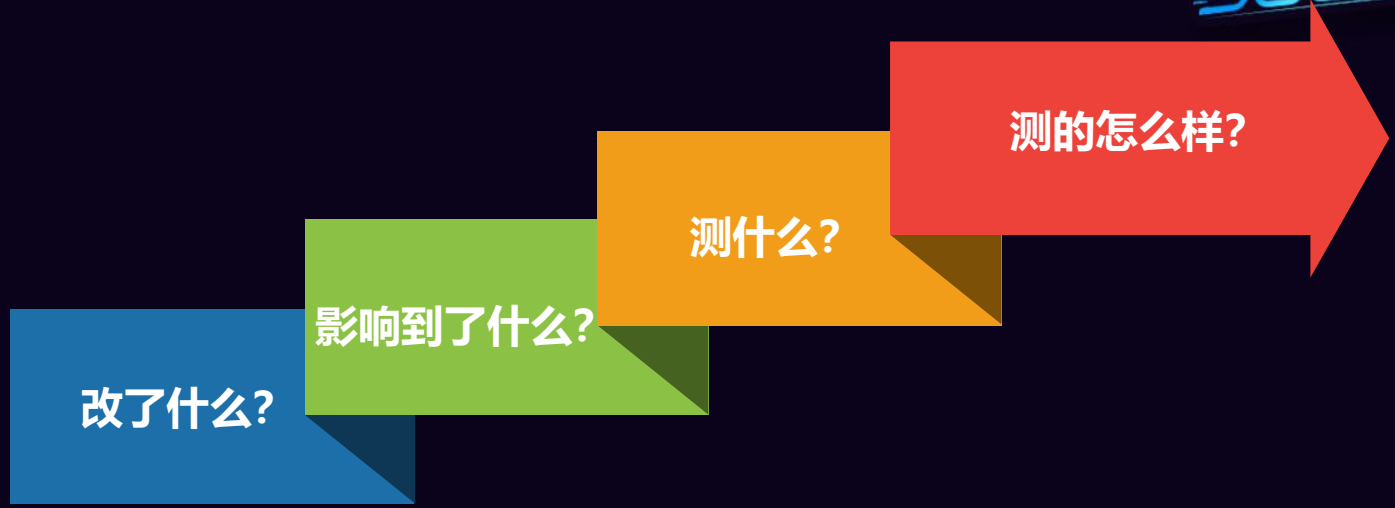
作为测试，我太难了！！！

找出问题

测多少？谁说了算？



最终要解决的问题



模糊的
基于能力的
基于经验的
主观判断为主的
缺乏数据评估的

明确的
基于数据的
基于分析的
有客观依据的
可量化评估的

精准测试

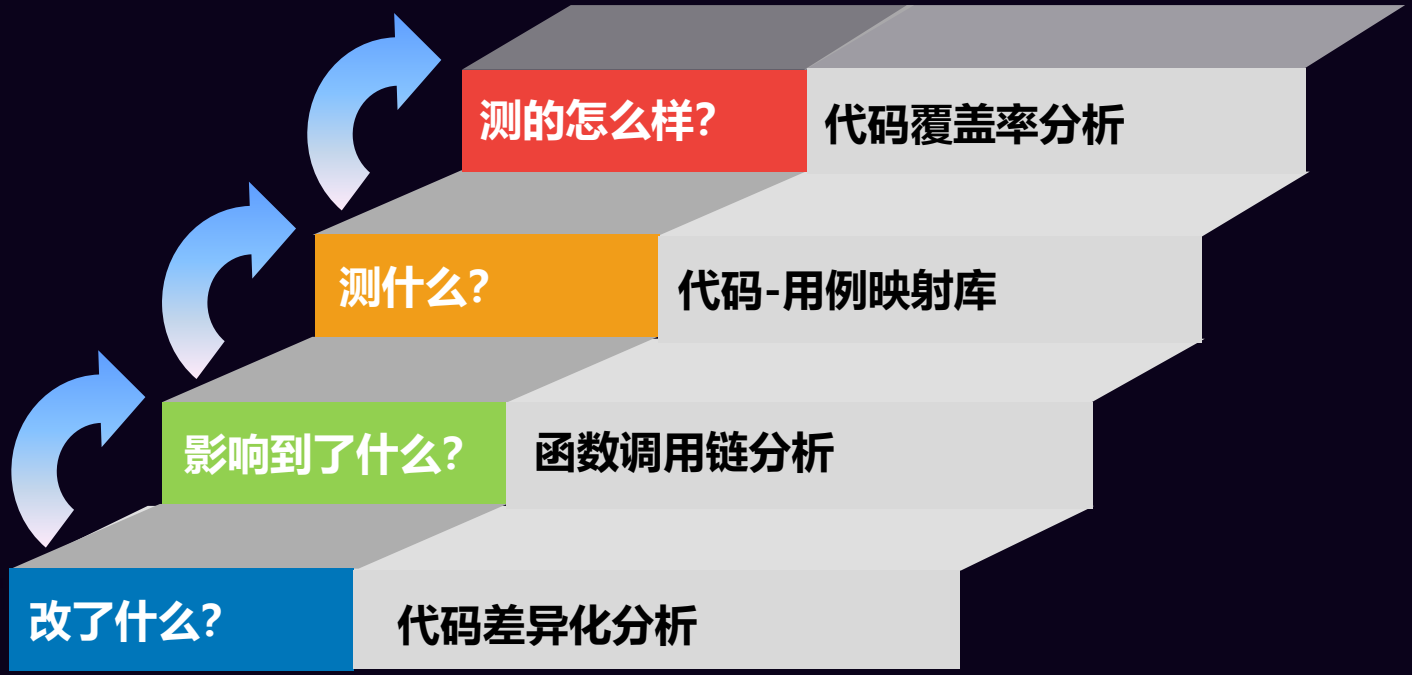
精准测试的概念

精准测试是一种可追溯的软件测试技术。即借助于一定的技术手段，通过算法的辅助对传统软件测试过程进行可视化、分析及优化的过程。

- 正向追溯：可以追溯case执行路径，为排查、定位问题快速提供依据
- 反向追溯：通过变更代码快速圈定相关的case集，降低回归的盲目性和工作量

解决方案

解决思路



整体方案设计


生成端
展现端
查询端
数据可视化平台
自动插桩服务
1
带有调用信息收集代码的可执行程序
2
自动写入
调用链服务
收集
分析
加工
生成
绘图
存储
3
写入
映射库
查询
9
Case执行
自动化case平台
补充case
11
10
覆盖率服务
评估端
提交代码
4
代码差异化分析
6
5
变更函数列表
8
调度执行
7
关联case列表

改了什么?

代码差异化分析

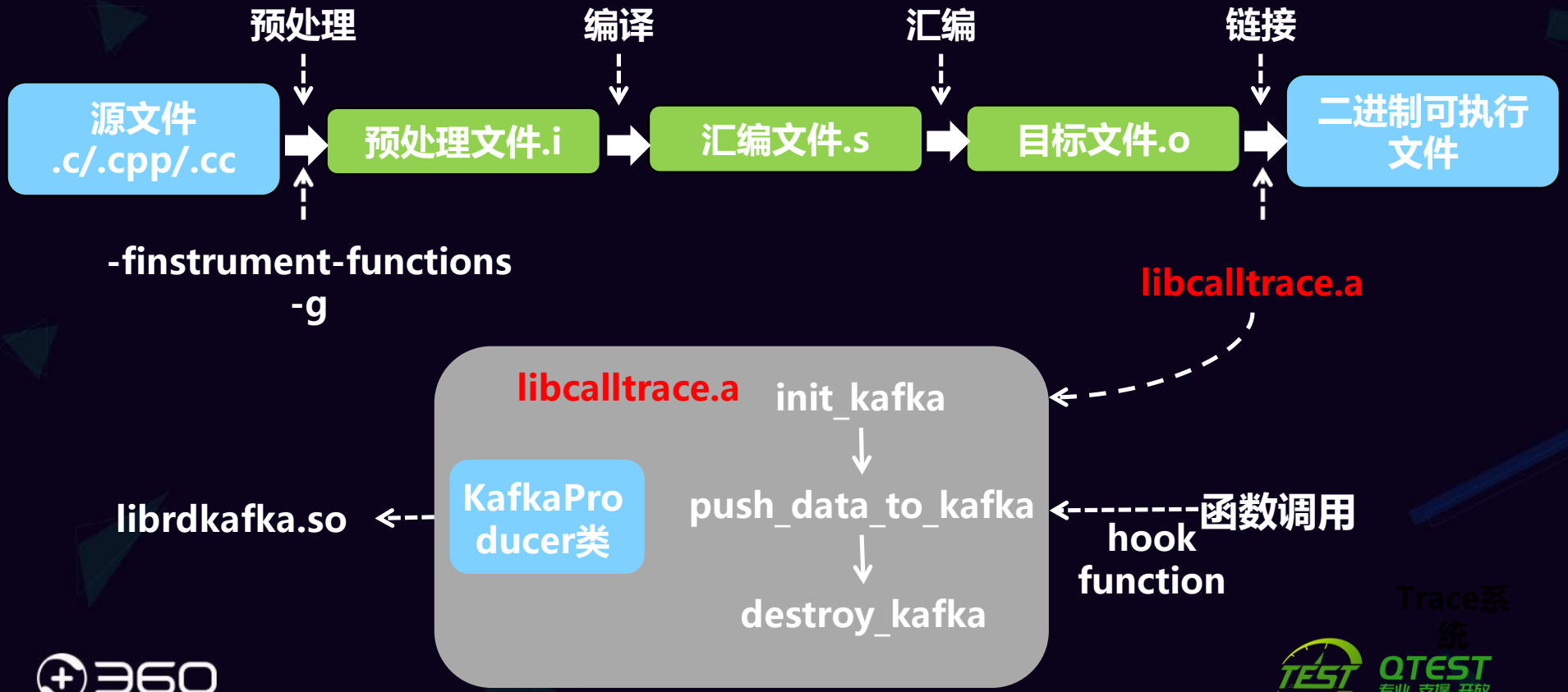
影响到了什么？测什么？

技术方案选取

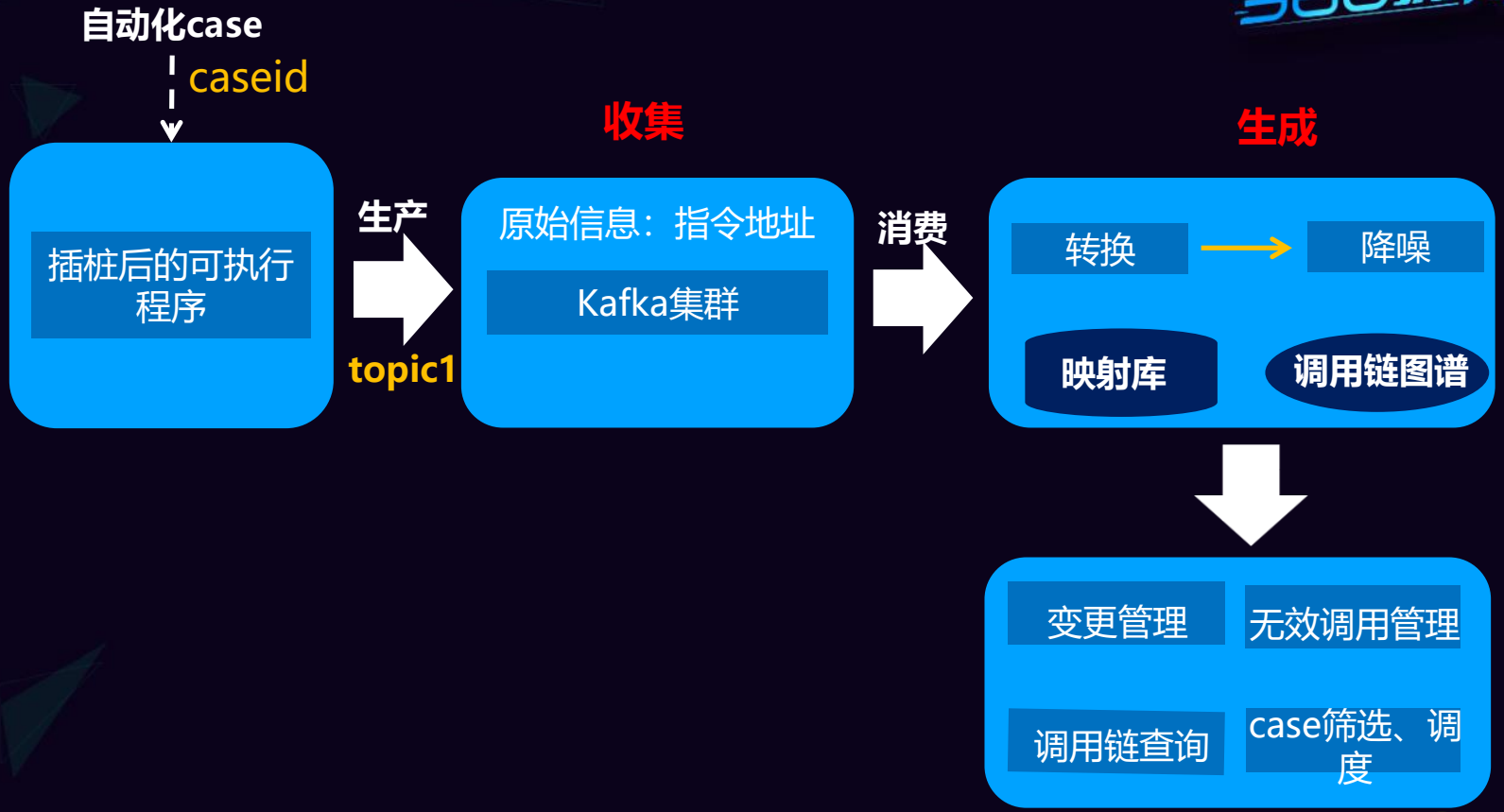
如何建立case和代码（函数列表、调用链）的映射?

方案	优点	问题	
基于代码覆盖率	团队已有现成的实现	缺少执行时的调用链路信息	×
基于opentracing的全链路跟踪	调用链路精准、无噪声	C++手工埋点代价太高	×
基于静态分析的链路调用	能比较简单地获取到较全的调用信息	只能作为实际调用的近似表达；（运行时绑定） 多模块系统无法获取全局；	×
通过自动插桩，收集case执行时的调用链路信息	无需手工埋点； 运行时调用链路； 模块间调用链路；	调用噪声； 新增代码；	√

1-自动插桩服务


预处理
编译
汇编
链接
源文件
.c/.cpp/.cc
预处理文件.i
汇编文件.s
目标文件.o
二进制可执行
文件
-finstrument-functions
-g
libcalltrace.a
libcalltrace.a
init_kafka
KafkaPro
ducer类
push_data_to_kafka
destroy_kafka
librdkafka.so
函数调用
hook
function

2-调用信息收集服务


自动化case
caseid
插桩后的可执行程序
生产
topic1
收集
原始信息：指令地址
Kafka集群
消费
生成
转换
降噪
映射库
调用链图谱
变更管理
无效调用管理
调用链查询
case筛选、调度
展现
Trace系统

自动插桩带来的优劣势

对源代码无侵入；

只需实现插桩库，一次实现，时时链接时时用；

对于运行时绑定的调用关系是精准的；

太粗暴；所到之处，无一幸免；

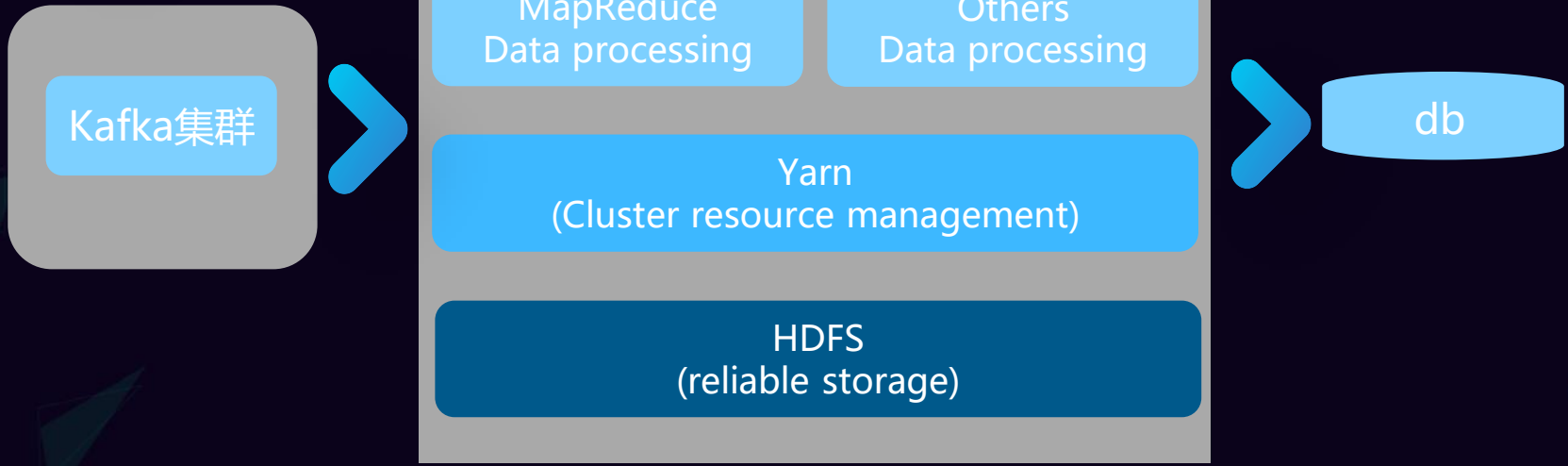
常驻线程产生对case有效调用产生的噪声；

Trace系统

3 离线分析—降噪

无效调用收集

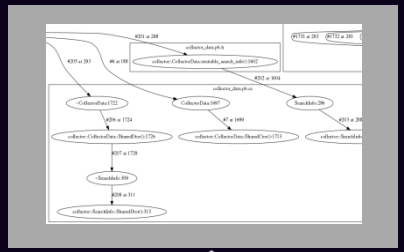
Kafka topic2



自动分析 + 人工干预

Trace系统

4-双向映射库提供正反追溯



提交代码

代码差异化分析

无效变更过滤

变更函数列表

相关的case集

case列表

映射库

调用图谱

function列表

case管理平台

用例管理

用例执行

测试报告

case调度执行

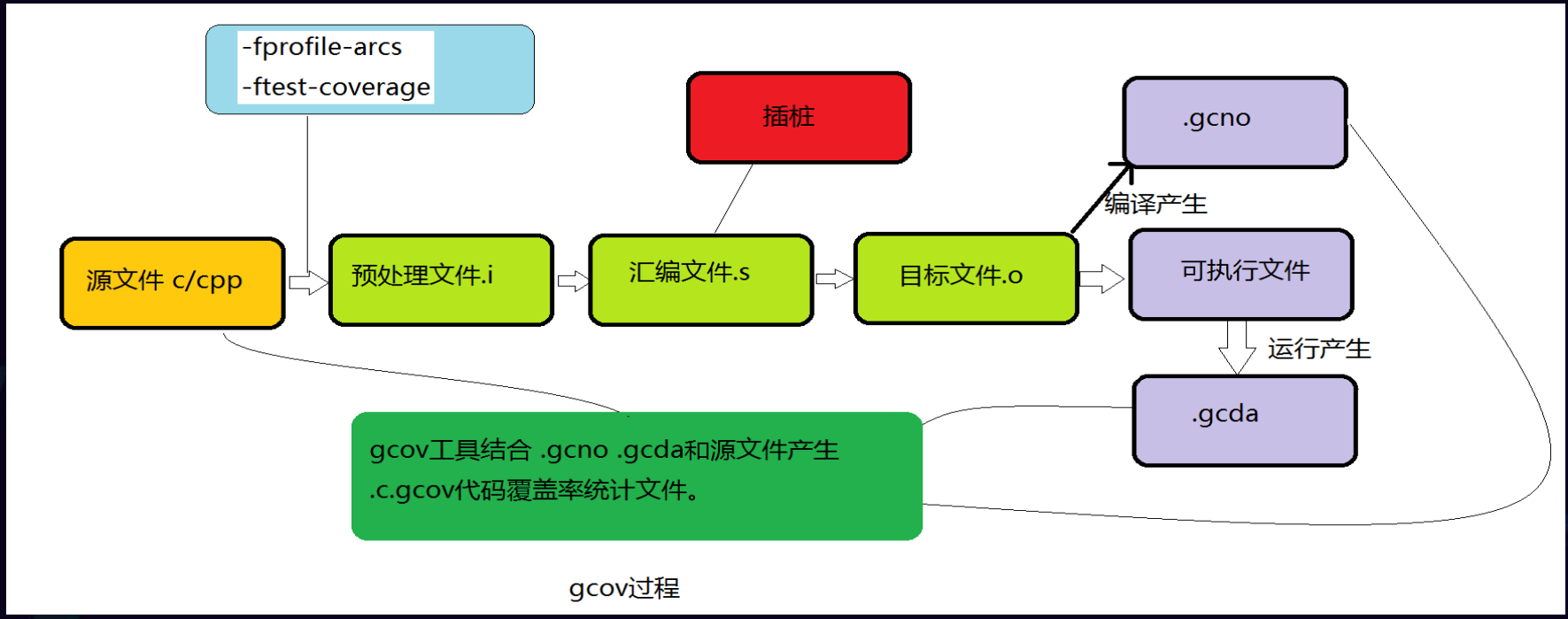
测的怎么样?

建立分析闭环：代码覆盖率统计

高的覆盖率 ≠ 高的测试质量

覆盖率的作用更多是用来证明测试是不充分的！

代码覆盖率统计：gcov原理



gcov过程

gcov原理

代码覆盖率统计：实际业务需求

我们的需求

- 数据合并：多人测试；多次测试；
- 数据存储分析：历史数据存储；多次build对比；
- 支持增量代码覆盖率统计
- 通用性：其他业务能以较低成本接入

代码覆盖率统计：实现

Part 3

应用现状

问题解决

迭代一个紧接一个

人手不够啊

RD：改的地方不少，保险起见，把相关功能都回归一遍吧！

改了这么多代码，该回归哪些功能？

测了半天，到底还有哪些代码没覆盖到？

全量自动化回归下吧，用例通过率不高，各种排查。。。

RD：测完了吗？
QA：没有，我需要把用例都回归下；
RD：只改几行代码，需要测这么多吗？

上线当日，心里忐忑+忐忑... bless

作为测试，我太难了！！！

应用场景

1 评估影响范围

明确改了什么，影响到了哪些功能。

2 根据变更代码圈定case

解决测试冗余，漏测的问题。

3 程序执行透明化

提供case执行路径，帮助熟悉代码。

4 评估精准测试效果

精准分析评估反馈建立闭环。

5 发现测试死角

找到应该覆盖，但未覆盖的代码。

6 标记冗余代码

为RD删代码提供数据参考。

7 代码准入门槛

增量覆盖率指标衡量RD自测程度。

8 问题自动排查

提供case执行轨迹跟踪。

应用场景：变更管理&case筛选

场景：上线后一段时间出bug了，某QA想看下当时的代码变更情况，并重跑下当时的case。

变更管理　查询

commit_id：　精准匹配

commit_id	module	file	func	操作
348930c	budgetsvr	req_handler.cpp	bgtsrv::ReqHandler::PackRes()	查看关联case

func_name：bgtsrv::ReqHandler::PackRes()

批量执行

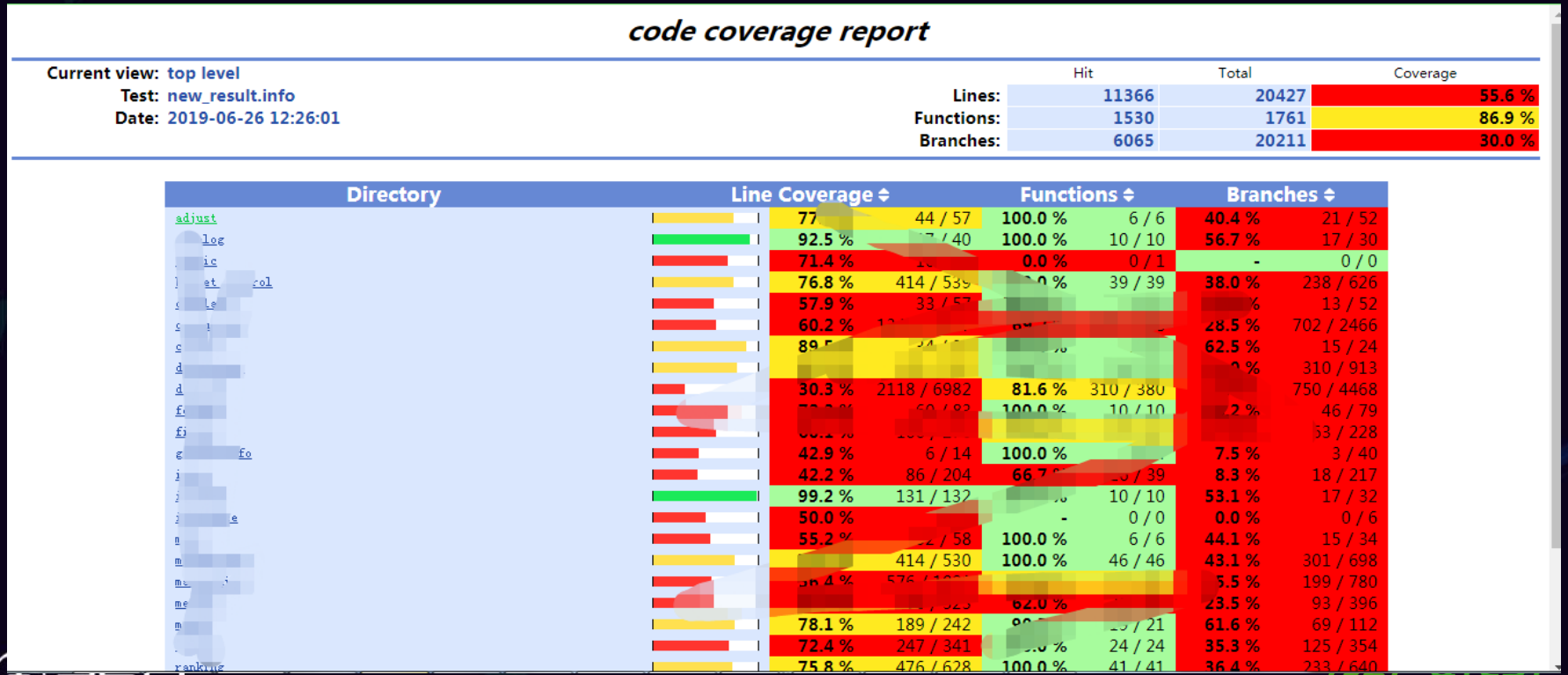
module_name	case_id	case_desc
[illegible]	test_lm_contr_1	channel34[illegible]设预算:默认预算50000
b[illegible]vr	test_lm_contr_6_d50c5650	sp[illegible]l_ac[illegible]nt_chnl_ratio[illegible]annel12
bu[illegible]svr	test_lm_contr_6_d50c7ebe	34[illegible]:ui[illegible]
buc[illegible]svr	test_lm_contr_6_d50c8d82	sp[illegible]l_ac[illegible]nt_ch[illegible].atio测试_channe[illegible]
bud[illegible]svr	test_lm_contr_6_d50c9c82	34[illegible]:ui[illegible]sp[illegible]al与default词表
bud[illegible]svr	test_lm_contr_6_d50cc766	specia[illegible]hnl_ratio测[illegible]hannel8
bud[illegible]svr	test_lm_contr_6_d5100340	plan:special_ac[illegible].atio测试_channel12
budgetsvr	test_lm_contr_6_d5114caa	账户[illegible]渠道控制

应用场景：记录case执行链路

QA：这个case执行失败了，经过我查看调用图，发现当时的执行逻辑是这样的。。。

应用场景：指出测试死角

QA：测了这么多，还有哪些代码行，哪些分支没覆盖到？



code coverage report

Current view: top level
Test: new_result.info
Date: 2019-06-26 12:26:01

	Hit	Total	Coverage
Lines:	11366	20427	55.6 %
Functions:	1530	1761	86.9 %
Branches:	6065	20211	30.0 %

Directory	Line Coverage		Functions		Branches	
adjust	77	44 / 57	100.0 %	6 / 6	40.4 %	21 / 52
log	92.5 %	/ 40	100.0 %	10 / 10	56.7 %	17 / 30
ic	71.4 %		0.0 %	0 / 1	-	0 / 0
et col	76.8 %	414 / 539	%	39 / 39	38.0 %	238 / 626
le	57.9 %	33 / 57			%	13 / 52
	60.2 %				28.5 %	702 / 2466
	89.5				62.5 %	15 / 24
					%	310 / 913
	30.3 %	2118 / 6982	81.6 %	310 / 380		750 / 4468
			100.0 %	10 / 10	2 %	46 / 79
						53 / 228
fo	42.9 %	6 / 14	100.0 %		7.5 %	3 / 40
	42.2 %	86 / 204	66.7	/ 39	8.3 %	18 / 217
	99.2 %	131 / 132	%	10 / 10	53.1 %	17 / 32
e	50.0 %		-	0 / 0	0.0 %	0 / 6
	55.2 %	/ 58	100.0 %	6 / 6	44.1 %	15 / 34
		414 / 530	100.0 %	46 / 46	43.1 %	301 / 698
	54.4 %				5.5 %	199 / 780
			62.0 %		23.5 %	93 / 396
	78.1 %	189 / 242		/ 21	61.6 %	69 / 112
	72.4 %	247 / 341	%	24 / 24	35.3 %	125 / 354
rankfug	75.8 %	476 / 628	100.0 %	41 / 41	36.4 %	233 / 640

应用场景：RD自测评估

QA：某某RD偷懒，自测case增量覆盖率才10%，发邮件通报下。

```
          :        2 :   return league_src_whitelist;
          :          : }
          :        2 [illegible] e) {
[ + - ]:        2 :     DspS[illegible]meI[illegible]dsp =[illegible]DspSmoothConsum[illegible],
[ - + ]:        2 :     if ([illegible]){
          :        0 :         C[illegible]OG("new Dsp Smooth Consume lis[illegible]);
          :        0 :         return NULL;
          :          :     }
          :          :
          :        2 :     FILE* fo[illegible]name[illegible]tr(), "r");
[ - + ]:        2 :     if (NULL == fp){
```

应用场景：冗余代码标记

某RD：早就想清理下废弃代码了，但就怕删出啥问题来。

code coverage report

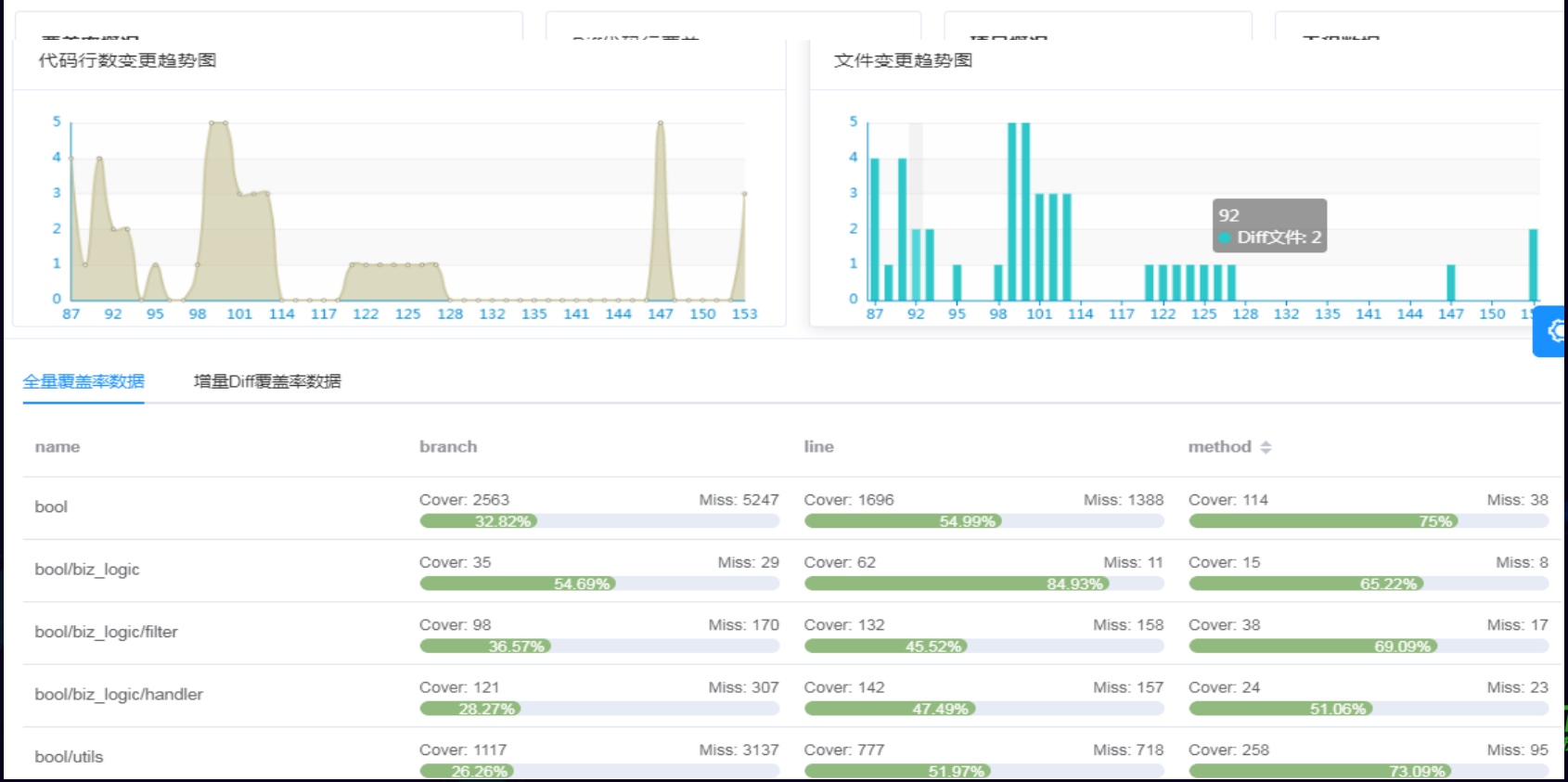
Current view: top level - [illegible] (source / functions)
Test: new_result.info
Date: 2019-06-26 14:49:46

	Hit	Total	Coverage
Lines:	17	29	58.6 %
Functions:	4	6	66.7 %
Branches:	6	14	42.9 %

```
Branch data     Line data    Source code
                          : #inclu[illegible]bject.h"
                          : #include [illegible]
                          : #include "[illegible]
                          :
                          : using names[illegible]
                          :
                        1 : ActiveObjec[illegible] = new ActiveObjectMgr;
                          :
                        0 : int  ActiveObject[illegible]
                          : {
                        0 :           std::set[illegible]ct*>::iterator itr = m_objSet.begin();
   [ # # ]:             0 :           for (;itr[illegible]et.end();++itr)
                          :           {
                        0 :                   [illegible]);
                          :           }
                        0 :           return 0;
                          : }
                          :
                        0 : int  ActiveObjectMgr::[illegible]All
                          : {
                        0 :           std:[illegible]iterator itr = m_objSet.begin();
   [ # # ]:             0 :           for ([illegible]end();++itr)
                          :           {
                        0 :                   [illegible]();
                          :           }
                        0 :           return [illegible]
                          : }
                          :
                        1 : int ActiveObje[illegible]All()
```

应用场景：覆盖率&代码变更趋势

QA：咦，这次新代码提交后，覆盖率忽然降了好多，什么原因呢?



name	branch	line	method
bool	Cover: 2563 Miss: 5247 32.82%	Cover: 1696 Miss: 1388 54.99%	Cover: 114 Miss: 38 75%
bool/biz_logic	Cover: 35 Miss: 29 54.69%	Cover: 62 Miss: 11 84.93%	Cover: 15 Miss: 8 65.22%
bool/biz_logic/filter	Cover: 98 Miss: 170 36.57%	Cover: 132 Miss: 158 45.52%	Cover: 38 Miss: 17 69.09%
bool/biz_logic/handler	Cover: 121 Miss: 307 28.27%	Cover: 142 Miss: 157 47.49%	Cover: 24 Miss: 23 51.06%
bool/utils	Cover: 1117 Miss: 3137 26.26%	Cover: 777 Miss: 718 51.97%	Cover: 258 Miss: 95 73.09%

Part 4

未来规划

未来规划

1 问题自动排查

2 动静态调用链结合

3 双向映射精准性提升

THANKS!