TiD 2017
质量竞争力大会

基于埋点的接口自动化框架

去哪儿网 范留杰

下一代软件研发
SOFTWARE DEVELOPMENT

TiD2017

- 整体方案
- 录制原理
- 回放原理
- 智选系列

TiD2017

背景

这次重构，需要手动回归全部功能

这个业务需要补充N个自动化case

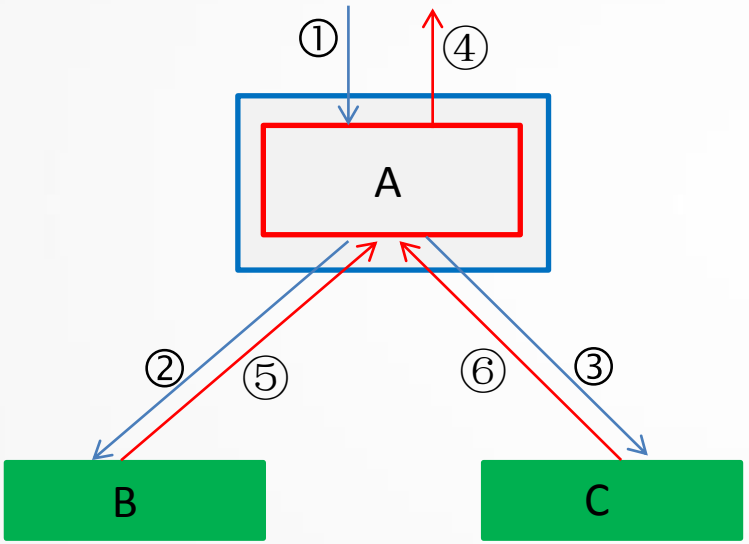
无case自动化

这次大改版，大部分自动化case需要修改

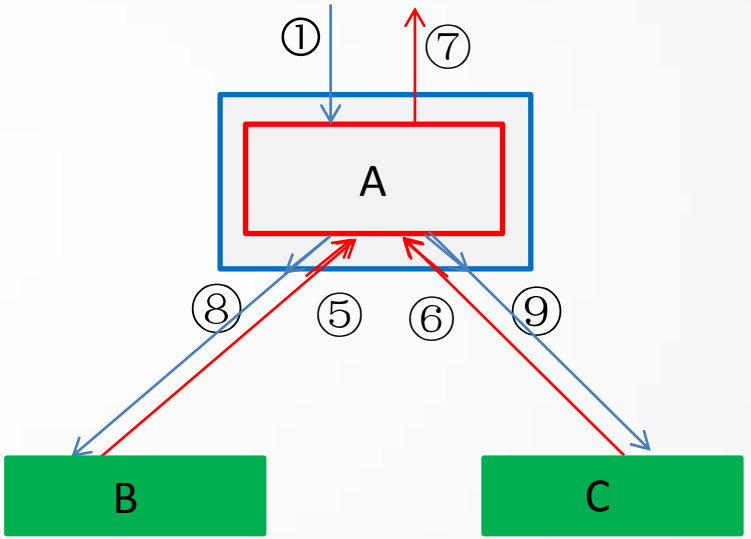
分析问题

- 请求准备+测试数据
  - ✓ 录制+mock
- 发送请求
  - ✓ 发送请求工具
- 结果验证
  - ✓ dump+对比

录制回放原理图



线上环境

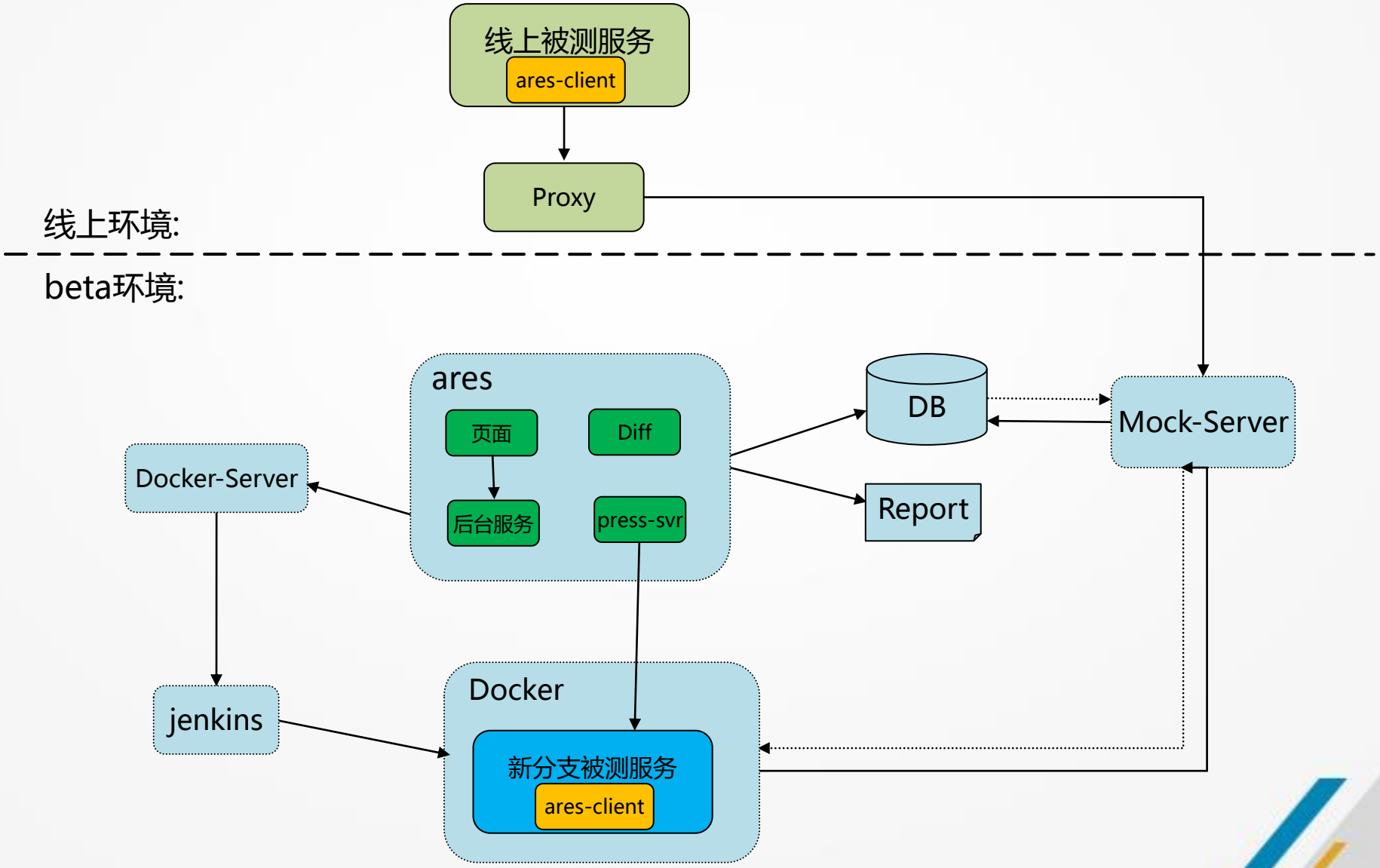


测试环境

① 上游请求，用于回放使用
④ 返回结果，作为对比基线
②③ 下发下游请求，作为对比基线
⑤⑥ 下游返回结果，用于mock

① 上游请求，用于回放使用
⑦ 返回新结果，与基线对比
⑧⑨ 下发下游新请求，与基线对比
⑤⑥ 下游返回结果，用于mock

架构示意图

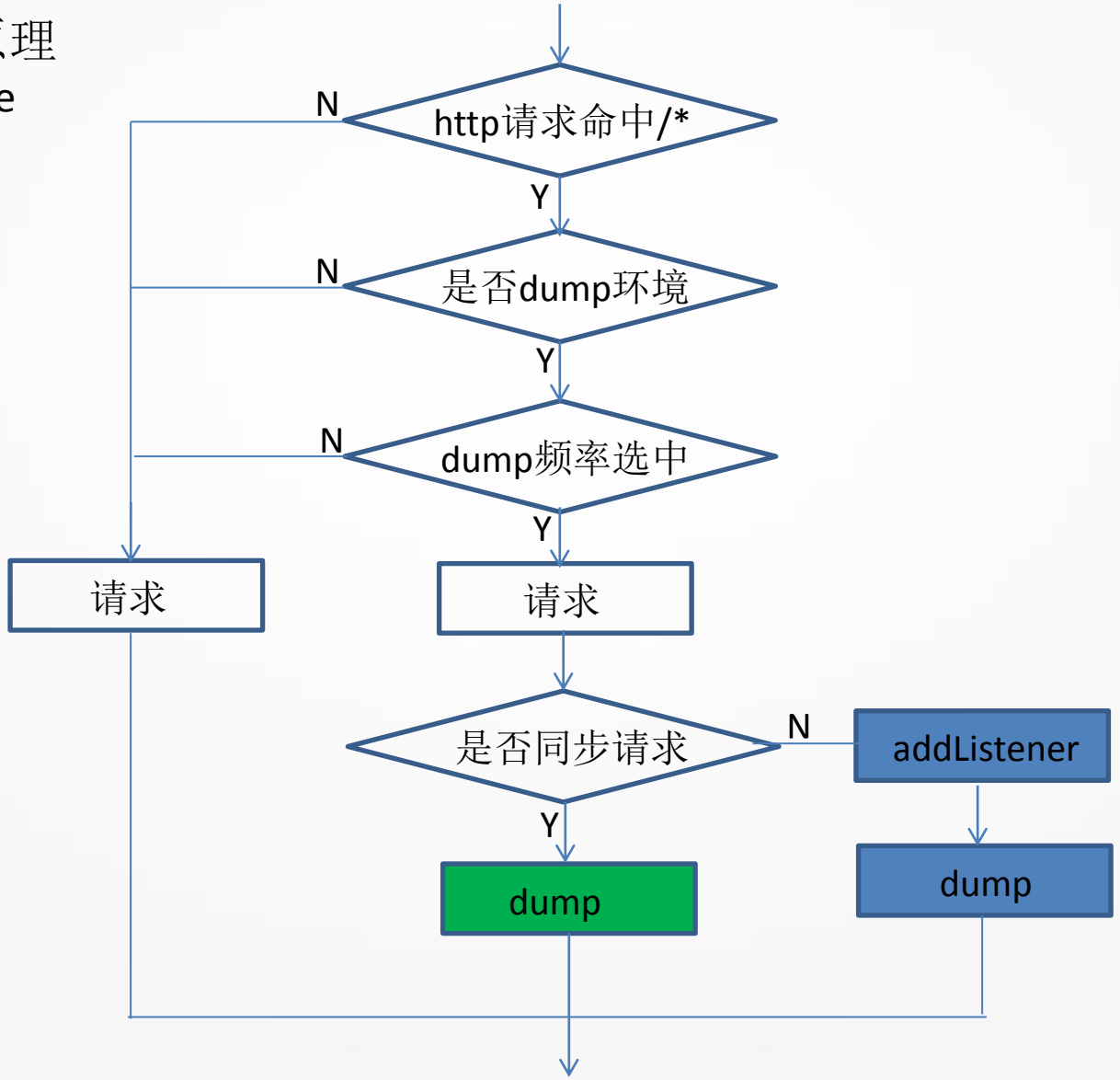

线上被测服务
ares-client
Proxy
线上环境:
beta环境:
ares
页面
Diff
后台服务
press-svr
DB
Mock-Server
Report
Docker-Server
jenkins
Docker
新分支被测服务
ares-client

TiD2017

埋点

➢ 埋点原理

http service



埋点

➢ 目前提供的埋点

1.http request

同步：需要在每个HttpClient后添加

```
private static final QunarClient httpClient = QunarClient.createDefaultClient(500, 1000, 50, 50);
httpClient.addRequestInterceptor(AresSyncHttpInterceptor.INSTANCE);
httpClient.addResponseInterceptor(AresSyncHttpInterceptor.INSTANCE);
```

异步：spring-bean.xml

```
<bean class="com...ares.client.util.AresQunarAsyncClientProcessor"/>
```

2.http service

使用Servlet的Filter机制扩展,支持同步、异步Servlet；需要添加servlet

```
<filter-mapping>
  <filter-name>httpAresFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

埋点

3.dubbo provider&consumer

使用dubbo Filter的扩展，支持同步、异步dubbo；添加dobbo的Filter

```
aresComsumer=com.qunar.hotel.qta.ares.client.point.DubboAresFilter$Consumer
aresProvider=com.qunar.hotel.qta.ares.client.point.DubboAresFilter$Provider
```

4.db

以mybatis插件（Interceptor）形式扩展

在mybits_config.xml中

```
<plugins>
    <plugin interceptor="com...ares.client.point.AresMyBatisExecutorPlugin"></plugin>
    <plugin interceptor="com...base.monitor.SQLTimeInterceptor" />
</plugins>
```

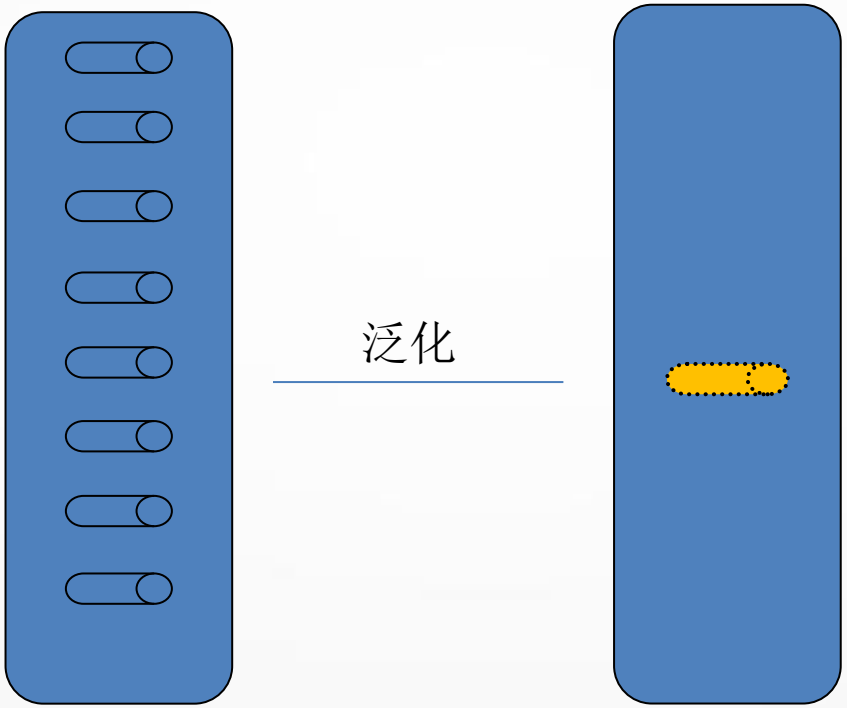
5.time

使用字节码替换，替换系统Date类

TiD2017

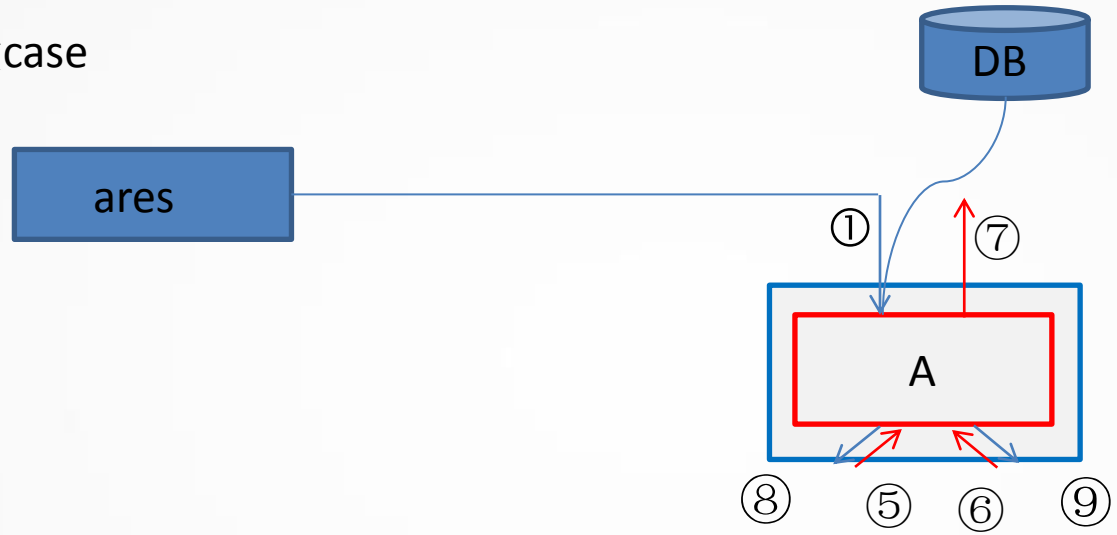
- 整体方案
- 录制原理
- **回放原理**
- 智选系列

回放方案

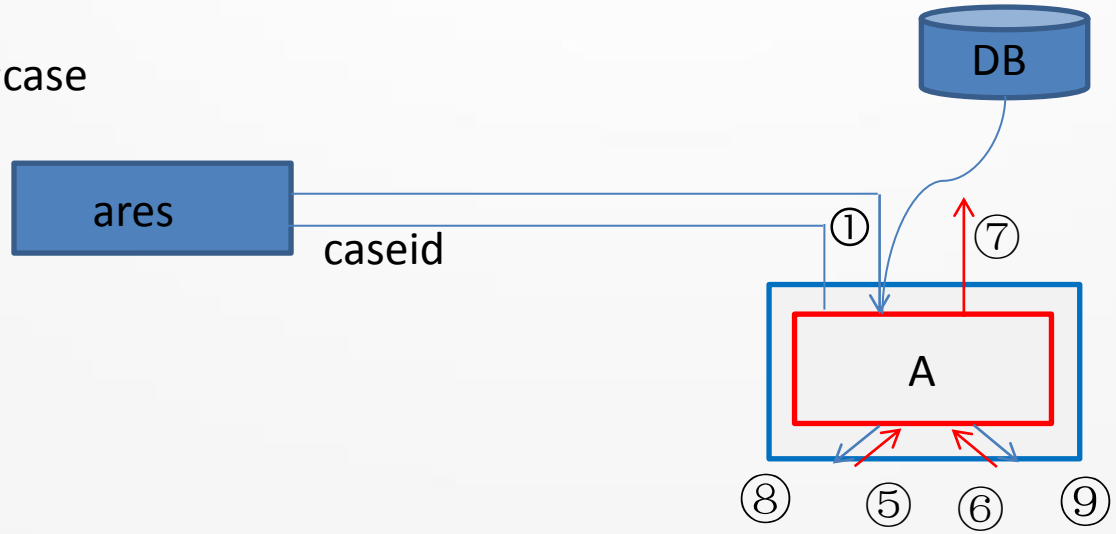


UI自动化组成

执行全量case



执行单个case



回放例子

- http

```
if (httpMethod == HttpMethod.GET) {
    httpClient.get(url, option).get();
}else if (httpMethod == HttpMethod.POST) {
    if (!Strings.isNullOrEmpty(postData)) {
        option.setPostBodyData(postData);
    }
    httpClient.post(url, option).get();
}

ExecutableHttpApi("http://" + host + ":" + httpPort,
                apiPO.getApiUrl(), httpMethod, apiPO.getContentType(), "", "", httpClient,
                executeCount, entryIds,traceIds);
```

- dobbo

```
private GenericService genericService;
genericService.$invoke(methodName, paramTypes, params);

ExecutableDubboApi(apiPO.getServiceName(), apiPO.getApiUrl(),
                paramTypes, new Object[paramTypes.length], genericService,
                executeCount, entryIds, traceIds);
```

TiD2017

- 整体方案
- 录制原理
- 回放原理
- **智选系列**

智选1.0

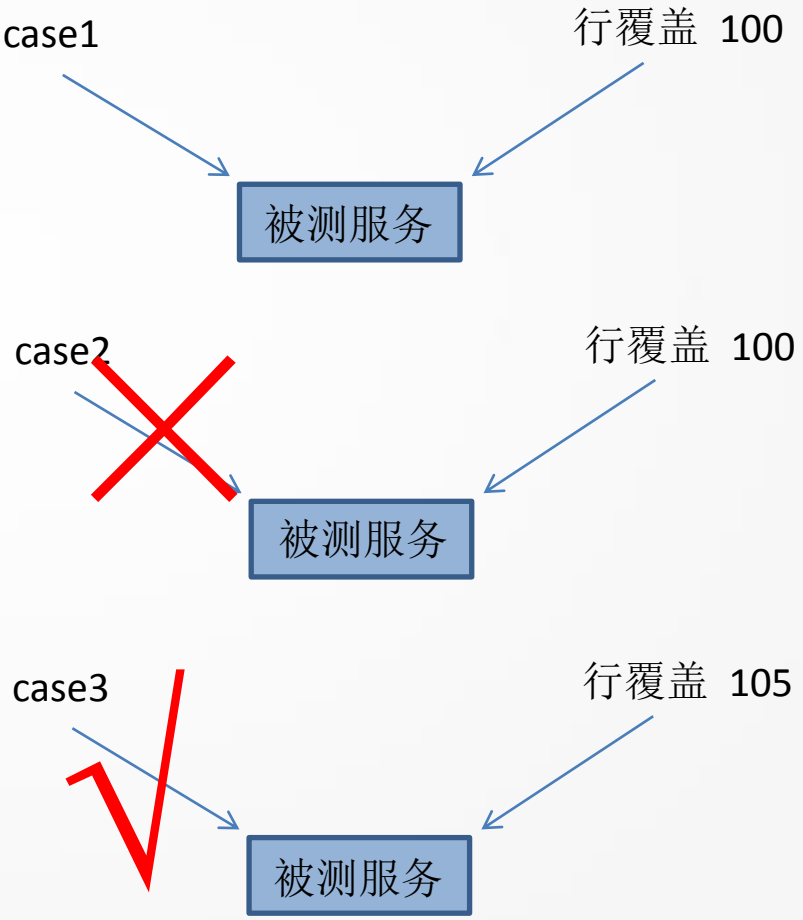
- 覆盖率
  - ✓ 在被测服务中插桩，执行到对应代码行会+1
  - ✓ cobertura、emma、jacoco
  - ✓ 行覆盖、分支覆盖、函数覆盖

```
function A() {
    if （ i == 10 ）{
        print “is 10”
    } else {
        print “is not 10”
    }
}
```

```
function B() {
    if （ i == 0 ）{
        print “is 10”
    }
}
```

智选1.0

- 功能重复的case去重
  - ✓ 根据覆盖逻辑去重
  - ✓ 根据覆盖率去重
- 原理
  - ✓ 行覆盖率贡献少的丢掉
- 实施
  - ✓ 插桩，记录覆盖率
  - ✓ 累加，行覆盖是否增加



智选2.0

- 和修改代码相关的case
  - ✓ 借助代码分析
  - ✓ 找出每个case相关的代码
- 原理
  - ✓ 源码分析得到变化的函数
  - ✓ 得到函数有关的case
- 实施
  - ✓ 源码分析+git分析
  - ✓ 覆盖率分析

```
1 function A() {
2     if （ i == 10 ）{
3         print “is 10”
4     }
5 }
6 function B() {
7     if （ i == 0 ）{
8         print “is 10”
9     }
10}
```

源码分析：

{“function”:”A”,”begin”:1,”end”:5}
{“function”:”B”,”begin”:6,”end”:10}

git diff分析：

{“diff_line”:”3”}
diff function:A

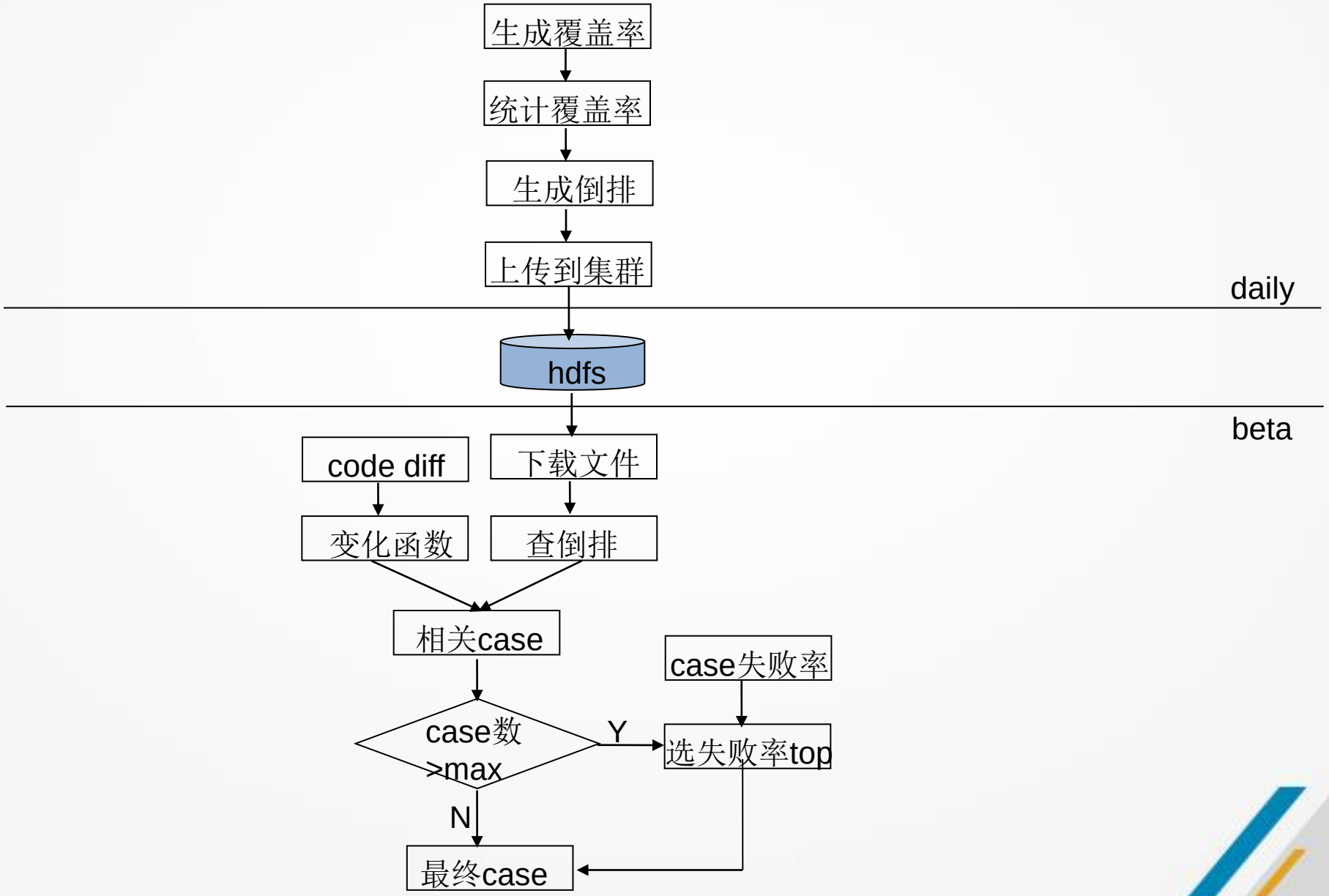
覆盖率分析：

case1:A,C
case2:B,C
正排

→

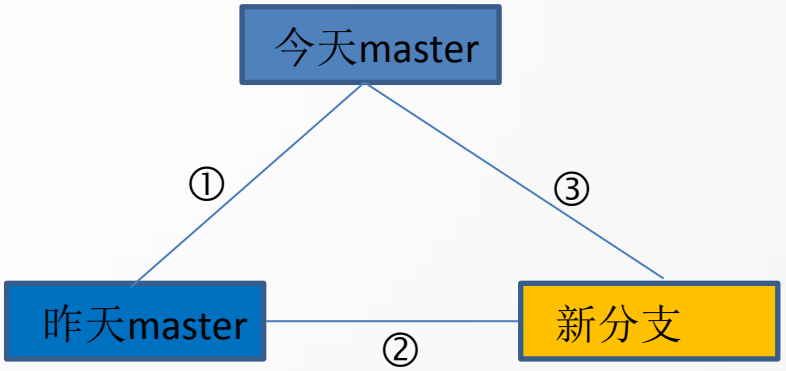
A->case1 √
B->case2
C->case1,case2
倒排

智选2.0

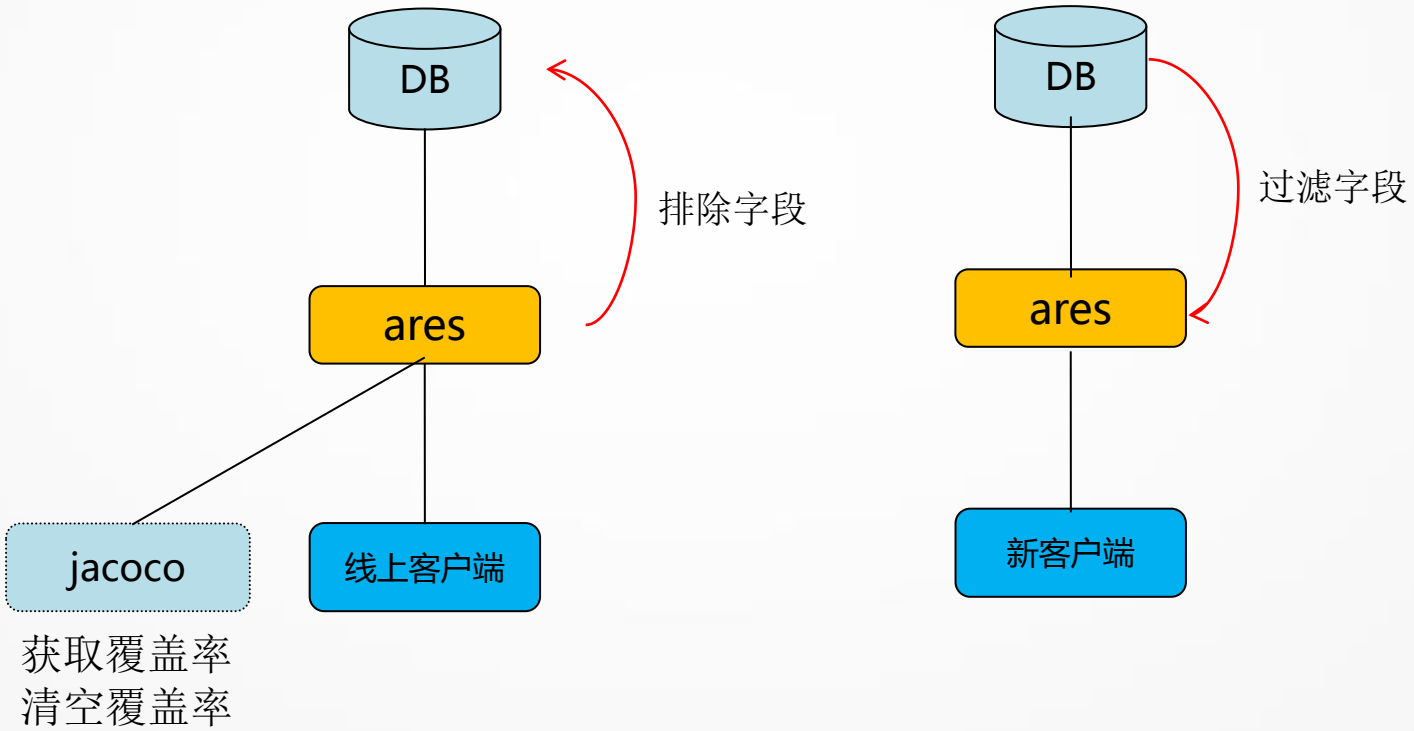

生成覆盖率
统计覆盖率
生成倒排
上传到集群
daily
hdfs
beta
code diff
下载文件
变化函数
查倒排
相关case
case失败率
case数
>max
Y
选失败率top
N
最终case

智选3.0

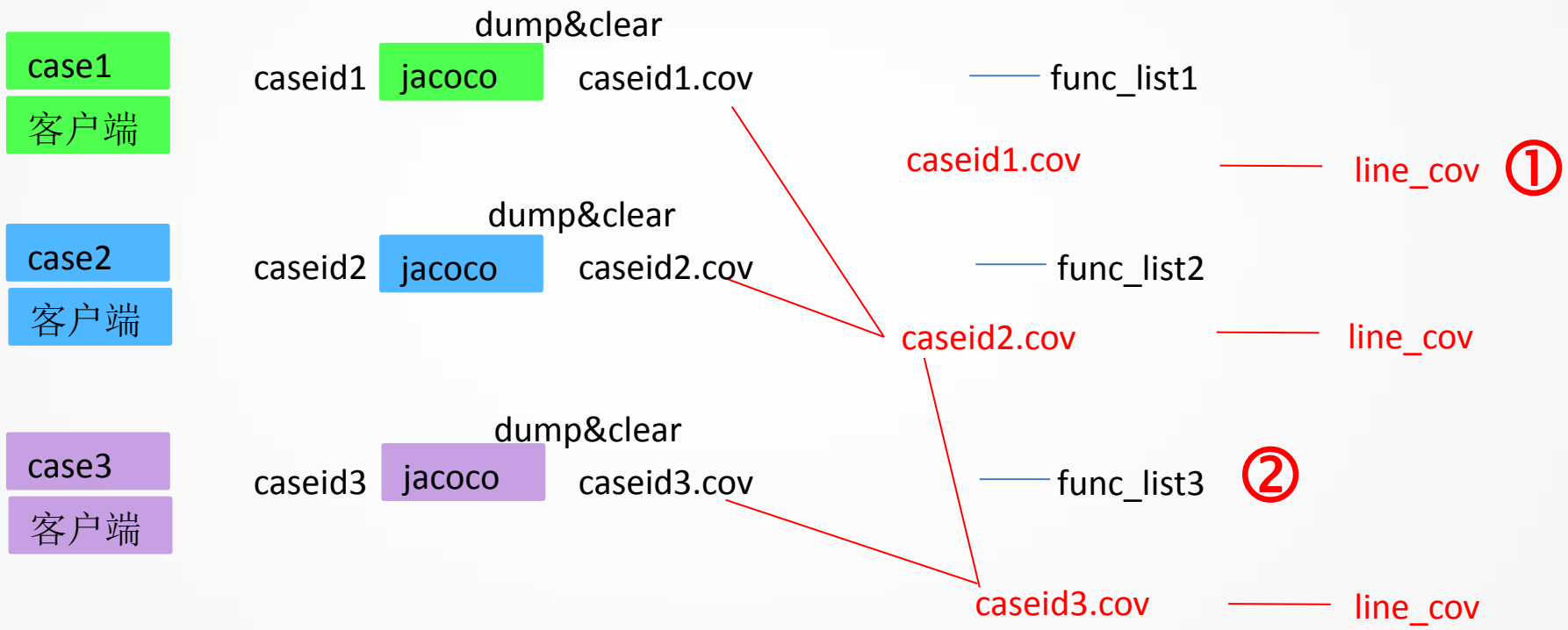
- 失败多
  - ✓ 去掉和这次修改无关的失败
  - ✓ 去随机+去master master diff
- 原理
  - ✓ case中check点是当时master结果
  - ✓ new branch和现在master的diff
  - ✓ 使用现在master和check点对比找出diff忽略
- 实施
  - ✓ 使用master执行case找出diff
  - ✓ 把diff作为排除字段



智选——合为一



智选——合为一



聚合 func->caselist

ignore key1.key2.key3 ③

TiD2017

- 整体方案
- 录制原理
- 回放原理
- 智选系列

TiD 2017
质量竞争力大会

谢谢大家的聆听

下一代软件研发
SOFTWARE DEVELOPMENT