

UBC Android duration!=(endtime - starttime)原因分析

目录

- [数据查询](#)
- [问题归因](#)
 - [3EE889B5E4E9034891EDB8560FEB4DA1|V27QSLCHY](#)
 - [全部用户日志分析结果](#)
- [场景模拟](#)
 - [模拟多线程调用UBC流式打点](#)
 - [模拟跨进程调用UBC流式打点](#)
 - [手百复现](#)
- [结论](#)
 - [基本流程](#)
 - [解决方案](#)
 - [方案1: 将in标记修改为inout](#)
 - [方案2: flowEnd数据库持久化endtime不修改](#)
 - [方案对比](#)
 - [其他问题](#)
 - [验证脚本](#)

数据查询



18点位duration数据一致性结果统计_DECIMAL.sql

1.3KB



0407_18点位duration数据一致性结果统计_双端.xlsx
10.0KB

20240407 v13.40+ 18点位pv, duration != endtime - starttime的pv, diff差值pv

20240407 18点位Android端总PV为954,214,081，其中有29,599,747条异常数据(endtime = 0，共8214条)，异常数据占比 3.10%。

- **Android**端18点位总PV为954,214,081； duration != endtime - starttime PV29,599,747，占比 3.10%
- 其中，只用 58.47% 的diff值在1ms，见表格：
 - diff <= 1ms 占比58.47%
 - diff <= 10ms 占95.76%
 - diff <= 1s 占比99.94%
 - diff <= 10s 占比99.95%

| | | | | | | | |
|---|----|-------------|------------|-------------|--------------|------------|-------------|
| 1 | | 18总 | 18异常 | diff <= 1ms | diff <= 10ms | diff <= 1s | diff <= 10s |
| 2 | PV | 954,214,081 | 29,599,747 | 17,306,114 | 28,345,682 | 29,581,395 | 29,585,672 |
| 3 | 占比 | - | - | 58.47% | 95.76% | 99.94% | 99.95% |

问题归因

Android数据特征和iOS特征不一致， diff 值的比较分散，数据表的统计数据及明细不太能够帮助归因。

考虑回捞日志辅助排查用户行为——任务 90701

3EE889B5E4E9034891EDB8560FEB4DA1|V27QSLCHY

| event_day | bhv_id | os_name | app_ver | cuid | duration | dur | diff | starttime | endtime | starttime_format | endtime_format |
|-----------|--------|---------|------------|--|----------|--------|------|---------------|---------------|-------------------------|-------------------------|
| 20240407 | 18 | android | 13.52.0.10 | 3EE889B5E4E9034891EDB8560FEB4DA1 V27QSLCHY | 133899 | 133996 | 97 | 1712460092635 | 1712460226631 | 2024/04/07 11:21:32.635 | 2024/04/07 11:23:46.631 |

Yalog 记录 endflow 调用的时间戳是 11:23:46.630，有可能在后续的步骤中，endtime 判定为0而取用了新的时间戳。


| | | pid | csf-tid | cf-tid |
|---|---|-------|---------|--------|
| 1 | | | | |
| 2 | 18_85888_178e5138f29e4da9987c17c6f38cdfa | 27575 | 27598 | 31251 |
| 3 | 18_85889_d1eecdad17e84776868960258ccd4bbf | 27575 | 27575 | 27575 |

从 Yalog 运行时日志发现问题点位(178e5138f29e4da9987c17c6f38cdfa)附近有一个正常 18 点位(d1eecdad17e84776868960258ccd4bbf)，二者在 csf 和 cf 的 tid 特征不一致

- 正常点位：csf、cf 的 tid 一致，说明业务方调用 UBC 方法是在单线程调用的(主线程)
- 异常点位：csf、cf 的 tid 不一致，说明业务方调用 UBC 方法是多线程调用的

因为 setValueWithDuration 方法没有添加 Yalog 日志，所以没办法参与分析

全部用户日志分析结果

 result_export.xlsx
319.7KB

0405~至今。20个用户的回捞日志中:

- 18pv共820条(剔除4条无效数据)
- duration != entime - starttime 共83条，占pv总量10.2%，全部都是非主线程调用的endflow导致的
- duration = endtime - starttime 共737条，其中非主线程调用 endflow 共172条，占 17.2%

场景模拟

模拟多线程调用UBC流式打点

编写Python脚本抓取Logcat分析日志特征：日志特征和线上不一致，很难复现由于多线程调用导致的线程安全问题；且模拟场景会有很多丢失 content 的日志，和线上数据特征不一致。


模拟跨进程调用UBC流式打点

复现场景：打开多进程的Activity并退出

| A | B | C | D | E | F | G | H | I | J |
|-------------|---------------|------------|---------------|------------|----------|-------|-------|----------------------|---|
| time | starttime | starttime | endtime | endtime | duration | dur | diff | raw_log | |
| 04-15 15:16 | 1713165135054 | 2024-04-15 | 1713165135581 | 2024-04-15 | 0.524 | 0.527 | 0.003 | b'04-15 15:12:15.642 | |
| 04-15 15:16 | 1713165140849 | 2024-04-15 | 1713165142127 | 2024-04-15 | 1.270 | 1.278 | 0.008 | b'04-15 15:12:22.197 | |
| 04-15 15:16 | 1713165222878 | 2024-04-15 | 1713165224082 | 2024-04-15 | 1.203 | 1.204 | 0.001 | b'04-15 15:13:44.143 | |
| 04-15 15:16 | 1713165381766 | 2024-04-15 | 1713165382020 | 2024-04-15 | 0.242 | 0.254 | 0.012 | b'04-15 15:16:22.112 | |
| 04-15 15:16 | 1713165394142 | 2024-04-15 | 1713165396201 | 2024-04-15 | 2.057 | 2.059 | 0.002 | b'04-15 15:16:36.244 | |
| 04-15 15:16 | 1713165401141 | 2024-04-15 | 1713165404087 | 2024-04-15 | 2.944 | 2.946 | 0.002 | b'04-15 15:16:44.120 | |
| 04-15 15:16 | 1713165406734 | 2024-04-15 | 1713165409588 | 2024-04-15 | 2.849 | 2.854 | 0.005 | b'04-15 15:16:49.615 | |
| 04-15 15:16 | 1713165414195 | 2024-04-15 | 1713165416546 | 2024-04-15 | 2.348 | 2.351 | 0.003 | b'04-15 15:16:56.570 | |
| 04-15 15:16 | 1713165467891 | 2024-04-15 | 1713165472608 | 2024-04-15 | 4.714 | 4.717 | 0.003 | b'04-15 15:17:52.647 | |
| 04-15 15:16 | 1713165476454 | 2024-04-15 | 1713165479749 | 2024-04-15 | 3.294 | 3.295 | 0.001 | b'04-15 15:17:59.782 | |

手百复现

复现路径：搜索 百度健康 -> 打开 百度健康 小程序-> 应用切后台

 logcat_export_baiduapp.xlsx
14.1KB

| A | B | C | D | E | F | G | H | I | J | K | L |
|------------|-----------|-----------|-----------|-----------|----------|--------|-------|-----------|------------|------------------------|---|
| time | starttime | timefor | endtime | timefor | duration | dur | diff | cduration | uuid | raw_log | |
| 04-16 11:3 | 171323832 | 2024-04-1 | 171323833 | 2024-04-1 | 6.974 | 6.974 | 0 | - | 141ccd401b | '04-16 11:32:29.528 1 | |
| 04-16 11:3 | 171323834 | 2024-04-1 | 171323834 | 2024-04-1 | 2.623 | 2.623 | 0 | - | 3a472075c | b'04-16 11:32:36.828 1 | |
| 04-16 11:3 | 171323835 | 2024-04-1 | 171323835 | 2024-04-1 | 5.183 | 5.183 | 0 | - | 12b1ea157b | '04-16 11:32:42.338 1 | |
| 04-16 11:3 | 171323836 | 2024-04-1 | 171323837 | 2024-04-1 | 10.869 | 10.869 | 0 | 12.208 | 6ecf54f2ec | b'04-16 11:32:56.311 1 | |
| 04-16 11:3 | 171323837 | 2024-04-1 | 171323838 | 2024-04-1 | 9.296 | 9.3 | 0.004 | - | 37b5ebad9 | b'04-16 11:33:06.849 1 | |
| 04-16 11:3 | 171323839 | 2024-04-1 | 171323839 | 2024-04-1 | 2.680 | 2.68 | 0 | - | dc71e4569b | '04-16 11:33:21.495 1 | |
| 04-16 11:3 | 171323839 | 2024-04-1 | 171323840 | 2024-04-1 | 7.519 | 7.519 | 0 | - | 19c95abac | b'04-16 11:33:28.739 1 | |
| 04-16 11:3 | 171323840 | 2024-04-1 | 171323841 | 2024-04-1 | 5.796 | 5.802 | 0.006 | - | bf19296fb | b'04-16 11:33:33.619 1 | |
| 04-16 11:3 | 171323841 | 2024-04-1 | 171323841 | 2024-04-1 | 2.399 | 2.399 | 0 | - | da6381227b | '04-16 11:33:41.093 1 | |
| 04-16 11:3 | 171323841 | 2024-04-1 | 171323847 | 2024-04-1 | 56.506 | 56.506 | 0 | - | 246459c30b | '04-16 11:34:37.218 1 | |
| 04-16 11:3 | 171323847 | 2024-04-1 | 171323848 | 2024-04-1 | 11.272 | 11.272 | 0 | - | 72150f4e2b | '04-16 11:34:47.905 1 | |
| 04-16 11:3 | 171323848 | 2024-04-1 | 171323850 | 2024-04-1 | 15.403 | 15.403 | 0 | - | 9a0a61031b | '04-16 11:35:03.972 1 | |
| 04-16 11:3 | 171323850 | 2024-04-1 | 171323850 | 2024-04-1 | 3.847 | 3.847 | 0 | - | ee8e3ffe4e | b'04-16 11:35:07.056 1 | |
| 04-16 11:3 | 171323850 | 2024-04-1 | 171323850 | 2024-04-1 | 3.656 | 3.656 | 0 | - | c3966628fb | '04-16 11:35:11.259 1 | |
| 04-16 11:3 | 171323850 | 2024-04-1 | 171323851 | 2024-04-1 | 2.611 | 2.617 | 0.006 | - | 0d0e226eb | b'04-16 11:35:13.272 1 | |
| 04-16 11:3 | 171323851 | 2024-04-1 | 171323852 | 2024-04-1 | 2.440 | 2.44 | 0 | - | 784cd4417b | '04-16 11:35:21.955 1 | |
| 04-16 11:3 | 171323852 | 2024-04-1 | 171323852 | 2024-04-1 | 7.793 | 7.793 | 0 | - | 8de8df1c2b | '04-16 11:35:29.083 1 | |
| 04-16 11:3 | 171323854 | 2024-04-1 | 171323855 | 2024-04-1 | 10.766 | 10.766 | 0 | - | 14aab238f | b'04-16 11:35:55.450 1 | |
| 04-16 11:3 | 171323855 | 2024-04-1 | 171323855 | 2024-04-1 | 2.716 | 2.721 | 0.005 | - | 643227bc8b | '04-16 11:35:56.554 1 | |

测试在前台退出百度健康小程序时几乎都能保证diff = 0。这个结果也和归因数据契合：

duration = endtime - starttime 共737条，其中非主线程调用 endflow 共172条，占 17.2%

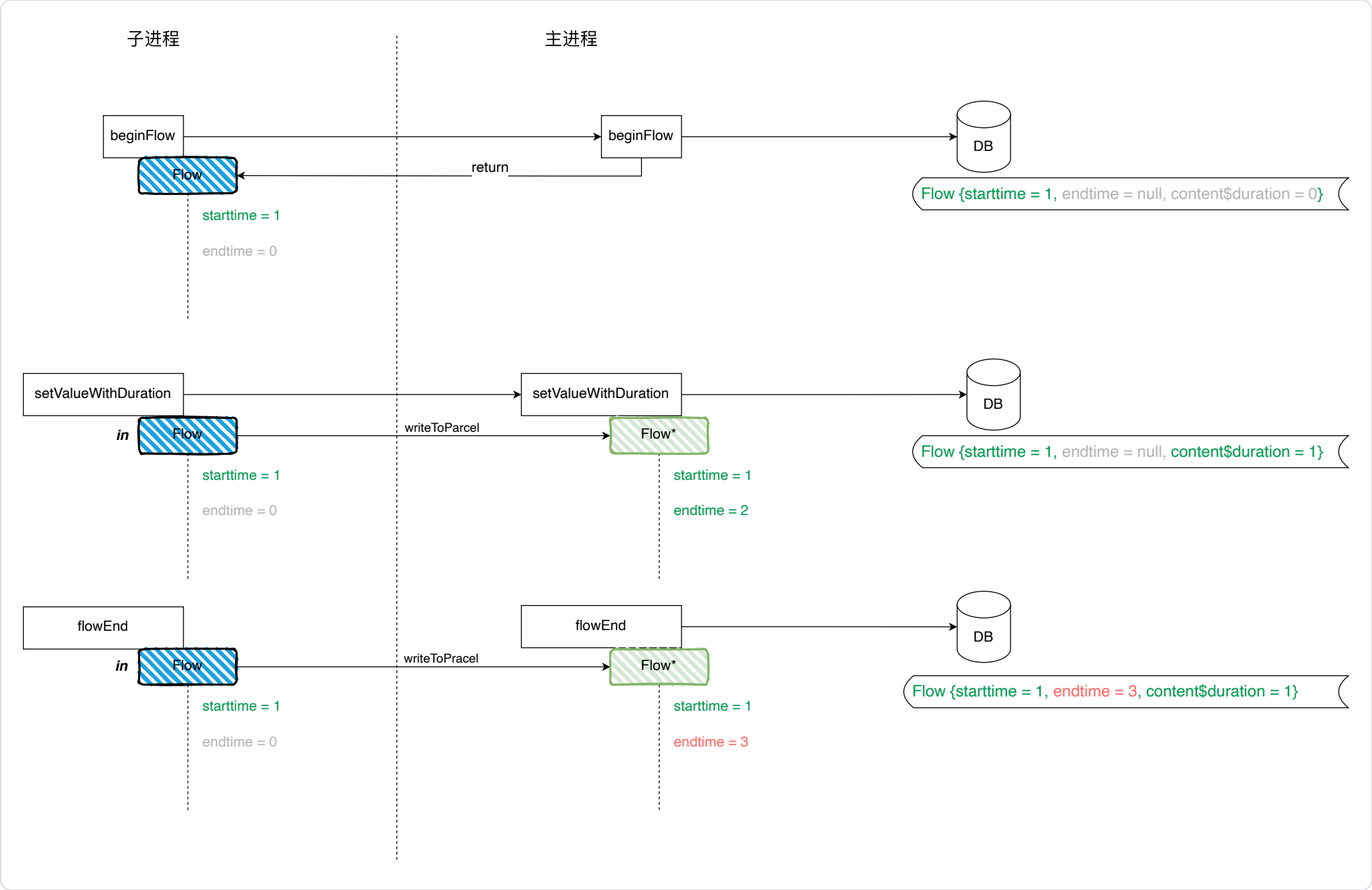
结论

跨进程场景导致 endtime 在 flowSetValueWithDuration() 和 flowEnd() 方法中被赋值两次

基本流程

涉及到的三个UBC方法

- UBCServiceManager#beginFlow(): Flow
- UBCServiceManager#flowSetValueWithDuration()
- UBCServiceManager#flowEnd()



- 子进程启动时

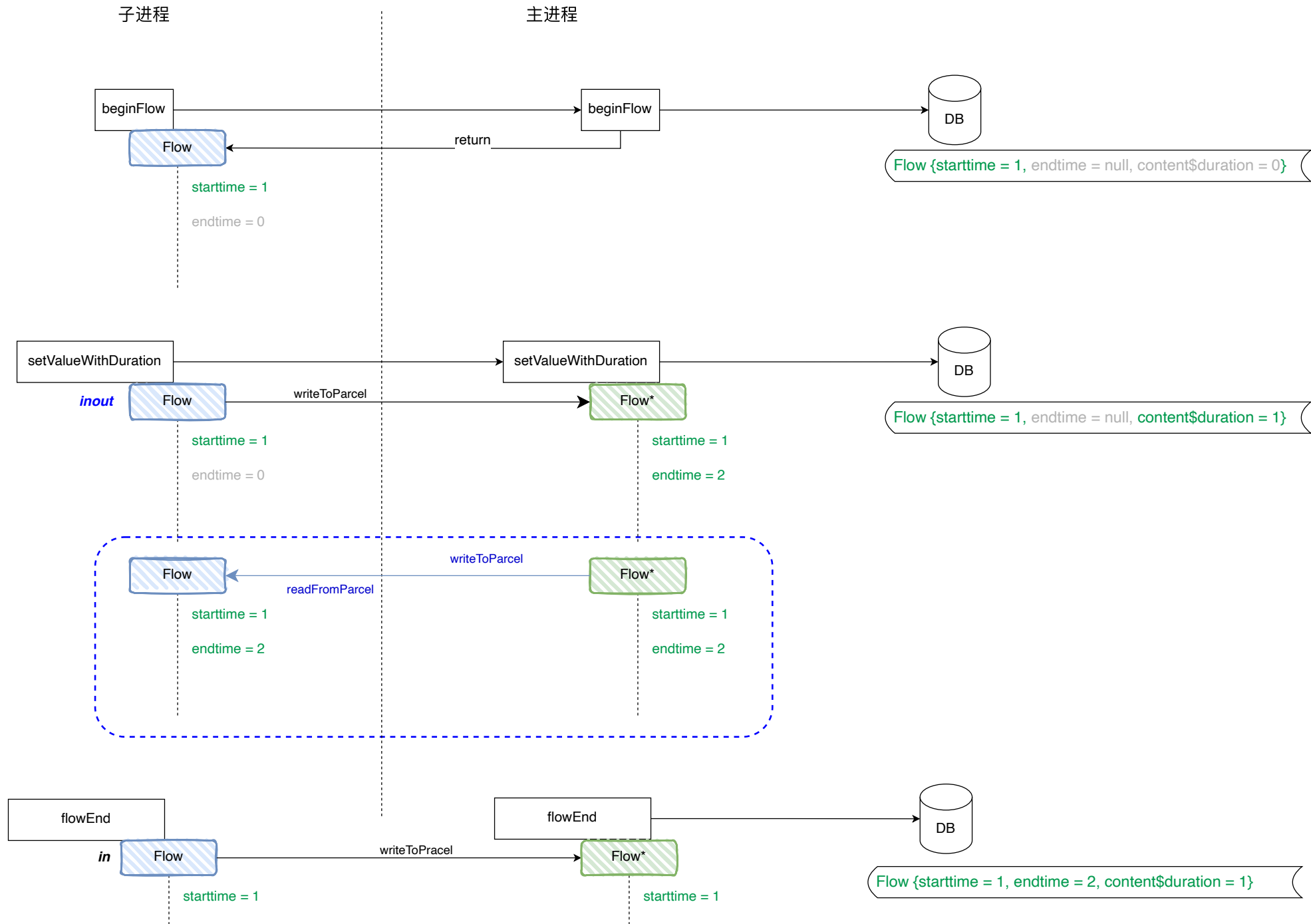
- beginFlow
 - IPC 调用主进程方法，获得一个 Flow 实例，子进程持有该 Flow 实例
- 子进程退出时
 - setValueWithDuration
 - IPC 调用主进程方法，子进程 Flow 实例作为方法的参数，标记为 in，主进程获得一个序列化的 Flow* 实例
 - Flow* 的 endtime 取时间戳，同时计算时长 duration 并持久化到数据库中
 - endFlow 方法，通过 IPC 调用主进程方法
 - IPC 调用主进程方法，子进程 Flow 实例以 in 模式传入方法列表，主进程获得一个序列化的 Flow* 实例
 - Flow* 的 endtime 值仍为 0，为保证 endtime 不为 0，又取了一次时间戳，并把数据持久化到数据库

结论：setValueWithDuration 的 IPC 方法以 in 来标记 Flow 参数，数据流向是 子进程 -> 主进程，主进程对 Flow* 赋值了 endtime 并不能回写到子进程的 Flow 实例。之后调用 flowEnd 时得到的 Flow* 实例 endtime 仍然未赋值，导致了 endtime 二次赋值，持久化到数据库时覆盖了原本的 endtime，而 content\$*** 内容保持不变，导致了 **endtime - starttime != duration**

解决方案

方案1: 将 in 标记修改为 inout

数据双向流动。主进程修改 Flow* 后，主进程多一步 writeToParcel，子进程会多一步 readFromParcel 来同步主进程对 Flow* 实例的修改

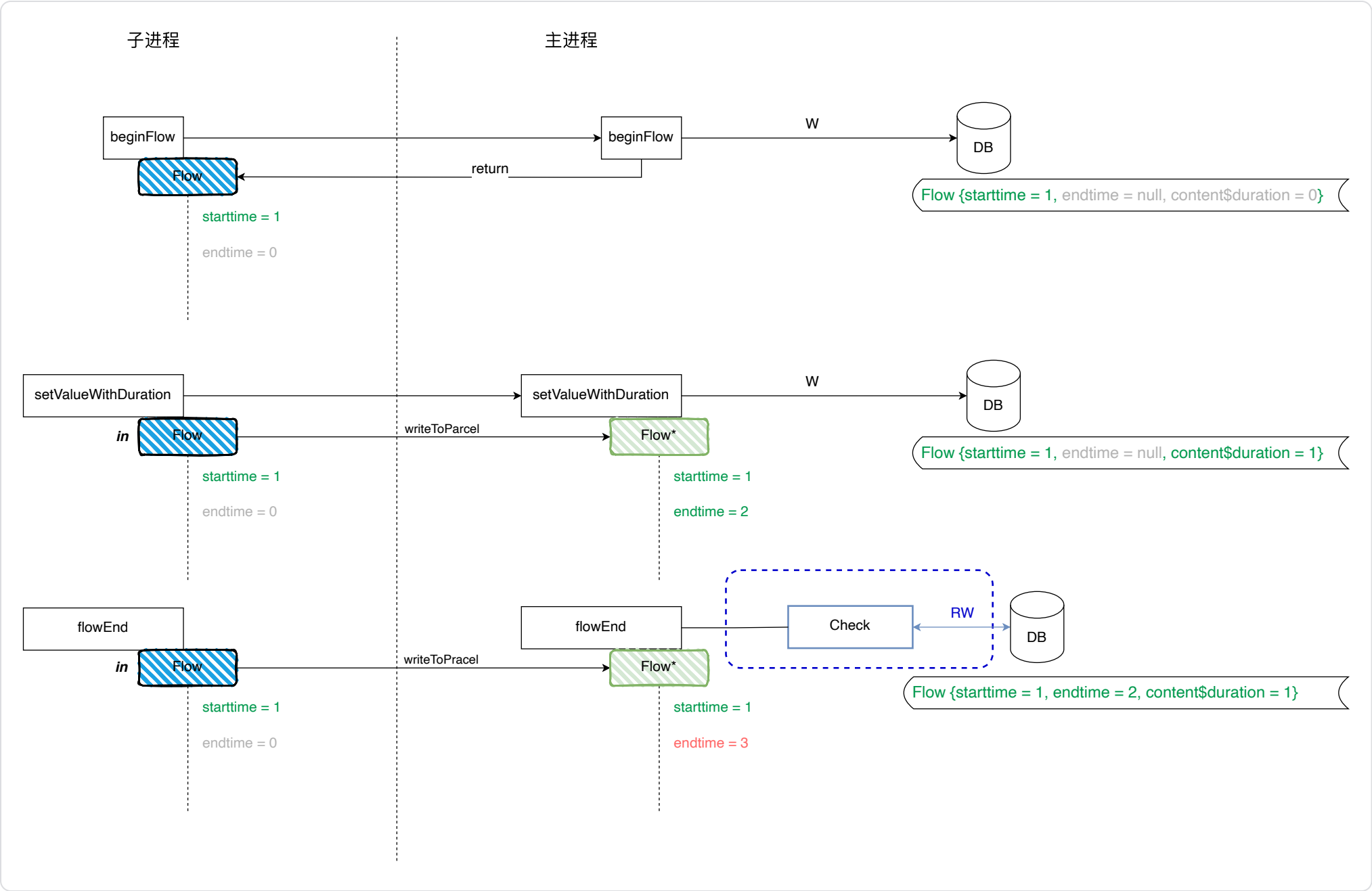


endtime = 2

endtime = 2

方案2: flowEnd数据库持久化endtime不修改

flowEnd 方法数据库持久化过程中检查该条日志的 endtime 是否已经有值，若有值则不修改 endtime



生成代码diff:

```
402 402     case TRANSACTION_flowSetValueWithDuration:
403 403     {
404 404         data.enforceInterface(descriptor);
405 405         com.baidu.ubc.Flow _arg0;
406 406         if ((0!=data.readInt())) {
407 407             _arg0 = com.baidu.ubc.Flow.CREATOR.createFromParcel(data);
408 408         }
409 409         else {
410 410             _arg0 = null;
411 411         }
412 412         java.lang.String _arg1;
413 413         _arg1 = data.readString();
414 414         this.flowSetValueWithDuration(_arg0, _arg1);
415 415         reply.writeNoException();
416 416         if ((_arg0!=null)) {
417 417             reply.writeInt(1);
418 418             _arg0.writeToParcel(reply, android.os.Parcelable.PARCELABLE_WRITE_RETURN_VALUE);
419 419         }
420 420         else {
421 421             reply.writeInt(0);
422 422         }
423 416         return true;
424 417     }
```

```
864 864     @Override public void flowSetValueWithDuration(com.baidu.ubc.Flow flow, java.lang.String value)
865 865     {
866 866         android.os.Parcel _data = android.os.Parcel.obtain();
867 867         android.os.Parcel _reply = android.os.Parcel.obtain();
868 868         try {
869 869             _data.writeInterfaceToken(DESCRIPTOR);
870 869             if ((flow!=null)) {
871 869                 _data.writeInt(1);
872 869                 flow.writeToParcel(_data, 0);
873 869             }
874 869             else {
875 869                 _data.writeInt(0);
876 869             }
877 869             _data.writeString(value);
878 871             boolean _status = mRemote.transact(Stub.TRANSACTION_flowSetValueWithDuration, _data, _reply,
879 872             if (!_status && getDefaultImpl() != null) {
880 873                 getDefaultImpl().flowSetValueWithDuration(flow, value);
881 874                 return;
882 875             }
883 876             _reply.readException();
884 876             if ((0!=_reply.readInt())) {
885 876                 flow.readFromParcel(_reply);
886 876             }
887 877         }
888 878         finally {
```

方案对比

「方案1」

优点

- 多进程实例：解决 setValueWithDuration 方法 IPC 过程中 Flow 实例数据不同步的问题

缺点：

- 性能方面：IPC 过程多一步 writeToParcel 和 readFromParcel 的过程，有一定的性能损耗

「方案2」

优点

- 保证了当调用 endFlow 方法时，保护数据库中 endtime 有值的情况下不被修改

缺点

- 多进程实例：多进程 Flow 实例数据不同步，和现有逻辑保持一致

最初设计 `aidl` 接口方法时，`Flow` 对象的 `endtime` 并没有在 `setValueWithDuration` 方法赋值。

其他问题

- 多进程场景 `flowEnd` 中 `flow.markEnd()` 失效；
 - 可以使用「方案1」修复。「方案2」可以保证 `endtime` 不被篡改，但是入库检查前的逻辑仍会被执行
- `UBCApiCollector.getInstance().collect(flow.getId(), 0, UBCApiCollector.ApiType.FLOW_END)`；方法会在 `IPC` 过程调用两次(子进程+主进程)

```
public void flowEnd(Flow flow) {
    if (DEBUG) {
        Log.d(TAG, "end flow, mId:" + flow.getId() + " handle" + flow.getHandle() + " mValid:"
            + flow.getValid());
    }

    if (flow == null || !flow.getValid()) {
        return;
    }

    // 如果flow已结束，则忽略后续的end, cancel操作
    if (flow.hasEnd()) {
        if (DEBUG) {
            Log.d(TAG, "flow has end, should not end again!!! ubc id=" + flow.getId()
                + ", flow handle=" + flow.getHandle());
        }
        return;
    }


    UBCApiCollector.getInstance().collect(flow.getId(), option: 0, UBCApiCollector.ApiType.FLOW_END);

    flow.markEnd();

    if (AppProcessManager.isServerProcess()) {
        // 监控漏斗采集flowEnd调用次数
        FunnelHelper.collectLogFunnel(BypassConstants.Funnel.CALL_ON_FLOW, flow.getId(), System.currentTimeMillis());
        // 运行时日志记录业务方调用flowEnd
        YalogHelper.saveRuntimeLogWithFlow(flow.getId(), flow.getHandle(), flow.getUUID(),
            EnumConstants.RunTime.CALL_FLOW_END);
    }
}
```

验证脚本

logcat_duration_check.py



3.8KB