

多进程场景TODO

目录

- 竞品多进程场景的处理逻辑
 - 1.1 神策
- 方案细化
 - in修改为inout的数据和影响
 - 2.1.1 性能损耗
 - 2.1.1 回写会不会引入新问题
 - 2.2 确定两个方法对应的SQL写法
 - 2.2.1 flowSetValueWithDuration
 - 2.2.2 flowEnd
 - 2.3 Check的IO控制在一次
- 附录
 - 神策时长打点核心流程代码

1. 竞品多进程场景的处理逻辑

导致UBC SDK多进程场景时长不自洽的原因：

- Flow 打点流程上，starttime, endtime, duration 分阶段入库
 - 方案二：duration 和 endtime 同时入库；endflow 阶段若 endtime 已经持久化则不再写入 endtime
- Flow 接口设计上，业务方持有 Flow 实例并在后续的流程中作为参数传入，在多进程场景中对于 Flow 实例的修改不能同步
 - 方案一：保证主进程对 Flow 实例的修改同步到子进程 Flow 实例

1.1 神策

神策文档

Github 神策Android

针对这次跨场景时长不自洽的问题，对**神策**时长打点业务的代码逻辑做了调研，梳理了以下几点特征：

时长点位API：

</>

Java

```
1 SensorsDataAPI#trackTimer(final String eventName, final TimeUnit timeUnit)
2 SensorsDataAPI#trackTimerStart(String eventName) // 开启时长点位，记录starttime，返回一个key值
3 SensorsDataAPI#trackTimerEnd(final String eventName, final JSONObject properties)// 结束时长点位，记录endtime，计算duration并写库
4 SensorsDataAPI#trackTimerPause(final String eventName) // 暂停计时
5 SensorsDataAPI#trackTimerResume(final String eventName) // 恢复计时
```

时长打点的业务流程：

- `trackTimerStart` 方法开启一个时长打点（数据缓存在内存中，不入库）
 - SDK 内部使用 `Map` 对 `eventName` 和 `EventTimer` 做映射(`eventName` 使用 `UUID` 拼出唯一标识字符串作为 `key`)，调用方可以获取到 `key` ；
 - `EventTimer` 由 SDK 内部持有，并通过 `key` 映射访问，只处理计时逻辑
- `trackTimerEnd` 结束时长打点，计算时长并将点位数据入库
- `starttime` , `endtime` 不入库，只用于计算 `duration`
 - `starttime` , `endtime` 取 `SystemClock` API的时间，计算的 `duration` 相当于 UBC SDK 的 `cduration` ，不受系统时间跳变影响；

数据库持久化方案：

SQLite

- 时长打点的业务逻辑不直接操作数据库，而是通过 `ContentProvider` 组件对 `SQLite` 读写

处理多进程场景的方式：

- 在多进程场景中，`ContentProvider` 可以做到在多进程场景下仍是单实例，避免了进程间通信逻辑可能带来的并发问题

结论：

针对UBC SDK多进程场景下时长不自洽的问题，分析了神策SDK的时长打点代码逻辑和多进程处理方式：

结论1：神策SDK采用 `ContentProvider` 组件间接操作SQLite数据库。`ContentProvider` 是 `Android` 中的一个组件，可以作为跨进程通信的一种方式，默认在多进程中保持单实例，通过这种方式处理多进程场景下并发读写数据库的问题

【UBC SDK】采用 `AIDL` 的 `IPC` 方式，业务逻辑也切换到主进程执行

结论2：`starttime` , `endtime` 仅用于计算 `duration` ，不会追加到日志中；所以对于时长是否自洽是不做解释的

【UBC SDK】`starttime` , `endtime` 是具有可读性的时间戳，用于计算 `duration` 日志中也需要保证时长自洽

结论3：结束时长打点时才会通过 `ContentProvider` 组件写库，会有一次 `IPC` 跨进程通信。此前的其他阶段数据都在内存缓存中，没有其他 `IPC` 过程

【UBC SDK】`starttime` , `duration` , `endtime` 会在三个阶段分别写库；若在子进程调用也会有三次 `IPC` ，`Flow` 实例作为方法参数在进程间传递(进程间内存不共享，实例对象需要序列化)

2. 方案细化

2.1. in修改为inout的数据和影响

2.1.1 性能损耗

在Demo中调试，修改前后方法处理时间上几乎没有区别。都是在2~5ms

2.1.1 回写会不会引入新问题

这里的回写有作用域，`flowSetValueWithDuration` 方法区间内的变化回被回写。当前的逻辑是 `Flow` 实例对于 `UBC SDK` 来说都是由方法参数传入的，并不会被长期持有并修改。

经过排查，在 `flowSetValueWithDuration` 方法中主进程对于 `Flow` 实例仅对 `endtime` 做了修改。

2.2 确定两个方法对应的SQL写法

2.2.1 flowSetValueWithDuration

</> UBCDatabaseHelper#updateFlowValue

Java

```
1 void updateFlowValue(String flowId, int flowHandle, String value) {
2     // ...
3     final ContentValues cv = new ContentValues();
4     cv.put(COLUMN_CONTENT, value);
5     StringBuilder sb = new StringBuilder();
6     sb.append(COLUMN_FLOW_ID)
7         .append("=\\"")
8         .append(flowId)
9         .append("\\")
```

```
10         .append(" AND ")
11         .append(COLUMN_FLOW_HANDLE_ID)
12         .append(" = ")
13         .append(flowHandle);
14     final String where = sb.toString();
15     int count = db.update(TABLE_FLOW, cv, where, null);
16     // ...
17 }
```

</> SQL

SQL

```
1 UPDATE TABLE_FLOW
2 SET COLUMN_CONTENT = value
3 WHERE COLUMN_FLOW_ID = flowID AND COLUMN_FLOW_HANDLE_ID = flowHandle
```

2.2.2 flowEnd

</>

Java

```
1 void endFlow(String flowId, int flowHandle, long endTime, JSONArray slotArray, String logId) {
2     final ContentValues cv = new ContentValues();
3     cv.put(COLUMN_STATE, Constants.FLOW_STATE_END);
4     cv.put(COLUMN_END_TIME, endTime);
5     if (!TextUtils.isEmpty(logId)) {
6         cv.put(COLUMN_LOGID, logId);
7     }
8     if (slotArray != null && slotArray.length() > 0) {
9         cv.put(COLUMN_SLOT, slotArray.toString());
10    }
11    StringBuilder sb = new StringBuilder();
```

```
12 sb.append(COLUMN_FLOW_ID)
13     .append("=\")
14     .append(flowId)
15     .append("\")
16     .append(" AND ")
17     .append(COLUMN_FLOW_HANDLE_ID)
18     .append(" = ")
19     .append(flowHandle);
20 final String where = sb.toString();
21 count = db.update(TABLE_FLOW, cv, where, null);
22 }
```

</>

SQL

```
1 UPDATE TABLE_FLOW
2 SET
3     COLUMN_STATE = FLOW_STATE_END, # Flow状态为END
4     COLUMN_END_TIME = endtime, # endtime
5     COLUMN_LOGID = logID, # logId
6     COLUMN_SLOT = slotArray.toString() # slot
7 WHERE COLUMN_FLOW_ID = flowID AND COLUMN_FLOW_HANDLE_ID = flowHandle
```

2.3 Check的IO控制在一次

<https://stackoverflow.com/questions/17079697/update-command-with-case-in-sqlite>

<https://stackoverflow.com/questions/4968841/case-statement-in-sqlite-query>

https://www.sqlite.org/lang_expr.html

</>

Plain Text

```
1 UPDATE tableA
2 SET B ='abcd',
3     C =CASEWHEN C ='abc'THEN'abcd'ELSE C
4     ENDWHEREcolumn=1;
```

</>

Java

```
1 void endFlow(String flowId, int flowHandle, long endTime, JSONArray slotArray, String logId) {
2     final ContentValues cv = new ContentValues();
3     cv.put(COLUMN_STATE, Constants.FLOW_STATE_END);
4     cv.put(COLUMN_END_TIME, endTime);
5     if (!TextUtils.isEmpty(logId)) {
6         cv.put(COLUMN_LOGID, logId);
7     }
8     if (slotArray != null && slotArray.length() > 0) {
9         cv.put(COLUMN_SLOT, slotArray.toString());
10    }
11    StringBuilder sb = new StringBuilder();
12    sb.append(COLUMN_FLOW_ID)
13        .append("=\")
14        .append(flowId)
15        .append("\")
16        .append(" AND ")
17        .append(COLUMN_FLOW_HANDLE_ID)
18        .append(" = ")
19        .append(flowHandle);
20    final String where = sb.toString();
21    count = db.update(TABLE_FLOW, cv, where, null);
22 }
```

</>

SQL

```
1 UPDATE TABLE_FLOW
2 SET
3     COLUMN_STATE = FLOW_STATE_END, # Flow状态为END
4     COLUMN_END_TIME = endtime, # endtime
5     COLUMN_LOGID = logID, # logId
6     COLUMN_SLOT = slotArray.toString() # slot
7 WHERE COLUMN_FLOW_ID = flowID AND COLUMN_FLOW_HANDLE_ID = flowHandle
8
9 UPDATE TABLE_FLOW
10 SET
11     COLUMN_STATE = FLOW_STATE_END, # Flow状态为END
12     COLUMN_END_TIME = CASE WHEN COLUMN_END_TIME IS NULL THEN endTime ELSE COLUMN_END_TIME END, # endtime
13     COLUMN_LOGID = logID, # logId
14     COLUMN_SLOT = slotArray.toString() # slot
15 WHERE COLUMN_FLOW_ID = flowID AND COLUMN_FLOW_HANDLE_ID = flowHandle
```

可以使用 `CASE WHEN` 保证只执行一次 SQL，但是需要替换 API -> `db.execSQL(sql)`，并自己拼接 SQL 语句。需要对原本的方法做较大的改动

</>

SQL

```
1 String updateSql = "UPDATE flow SET endtime= CASE WHEN endtime IS NULL THEN " + endTime + " ELSE endtime END,state=" +
    Constants.FLOW_STATE_END + " WHERE flowid=\"" + flowId + "\"" AND flowhandle = "+flowHandle;
2 db.execSQL(updateSql);
```

`endFlow` 这里原本使用的是 `ContentValues` 并使用 `db.update(TABLE_FLOW, cv, where, null)` 方法，尝试 SQL 注入后发现会导致 `endtime` 写入字符串：

_id	flowid	flowhandle	state	beginTime	endTime	content	option	reserve1	reserve2	slot	€	Na
51	18	260	1	17137676...	null	{"duratio...	0	null	null	null	n	
54	18	273	1	17137677...	null	{"duratio...	0	null	null	null	n	
57	18	286	1	17137679...	null	{"duratio...	0	null	null	null	n	
58	18	297	1	17137681...	null	{"duratio...	0	null	null	null	n	
60	18	309	1	17137681...	null	{"duratio...	0	null	null	null	n	
61	18	320	1	1713768...	null	{"duratio...	0	null	null	null	n	
62	18	331	1	1713769...	CASE W...	{"duratio...	0	null	null	null	n	
63	18	332	1	1713769...	null	{"duratio...	0	null	null	null	n	
64	18	343	1	17137725...	CASE W...	{"duratio...	0	null	null	null	n	
65	18	344	1	17137725...	null	{"duratio...	0	null	null	null	n	

CASE WHEN endTime IS NULL THEN 0 ELSE 1 END

Length: 43

附录

神策时长打点核心流程代码

SensorsDataAPI#trackTimerStart(String eventName)

给 `eventName` 生成 `EventTimer` 实例并缓存到 `Map` 中

</> com.sensorsdata.analytics.android.sdk.SensorsDataAPI#trackTimerStartJava

```
1  @Override
2  public String trackTimerStart(String eventName) {
3      try {
4          final String eventNameRegex = String.format("%s_%s_%s", eventName, UUID.randomUUID().toString().replace("-",
5              "_"), "SATimer");
6          trackTimer(eventNameRegex, TimeUnit.SECONDS);
7          trackTimer(eventName, TimeUnit.SECONDS);
8          return eventNameRegex;
9      } catch (Exception ex) {
10         SLog.printStackTrace(ex);
11     }
12 }
```

```
11     return "";  
12 }
```

</> com.sensorsdata.analytics.android.sdk.SensorsDataAPI#trackTimer

Java

```
1  @Deprecated  
2  @Override  
3  public void trackTimer(final String eventName, final TimeUnit timeUnit) {  
4      final long startTime = SystemClock.elapsedRealtime();  
5      mTrackTaskManager.addTrackEventTask(new Runnable() {  
6          @Override  
7          public void run() {  
8              try {  
9                  SDataHelper.assertEventName(eventName);  
10                 EventTimerManager.getInstance().addEventTimer(eventName, new EventTimer(timeUnit, startTime));  
11             } catch (Exception e) {  
12                 SALog.printStackTrace(e);  
13             }  
14         }  
15     });  
16 }
```

</> com.sensorsdata.analytics.android.sdk.core.business.timer.EventTimerManager#addEventTimer

Java

```
1  public void addEventTimer(String eventName, EventTimer eventTimer) {  
2      synchronized (mTrackTimer) {  
3          // remind: update startTime before runnable queue  
4          mTrackTimer.put(eventName, eventTimer);  
5      }  
6  }
```

SensorsDataAPI#trackTimerEnd(final String eventName, final JSONObject properties)

</> com.sensorsdata.analytics.android.sdk.SensorsDataAPI#trackTimerEnd()

Java

```
1 @Override
2     public void trackTimerEnd(final String eventName, final JSONObject properties) {
3         final long endTime = SystemClock.elapsedRealtime();
4         try {
5             final JSONObject cloneProperties = JSONUtils.cloneJsonObject(properties);
6             mTrackTaskManager.addTrackEventTask(new Runnable() {
7                 @Override
8                 public void run() {
9                     if (eventName != null) {
10                         EventTimerManager.getInstance().updateEndTime(eventName, endTime);
11                     }
12                     try {
13                         JSONObject _properties =
14 SAModuleManager.getInstance().invokeModuleFunction(Modules.Advert MODULE_NAME,
15 Modules.Advert.METHOD_MERGE_CHANNEL_EVENT_PROPERTIES, eventName, cloneProperties);
16                         if (_properties == null) {
17                             _properties = cloneProperties;
18                         }
19                         mSAContextManager.trackEvent(new
20 InputData().setEventName(eventName).setEventType(EventType.TRACK).setProperties(_properties));
21                     } catch (Exception e) {
22                         SLog.printStackTrace(e);
23                     }
24                 }
25             });
26         }
```

```
24     } catch (Exception e) {
25         SALog.printStackTrace(e);
26     }
27 }
```

</> com.sensorsdata.analytics.android.sdk.core.event.EventProcessor#process

Java

```
1  /**
2   * data process
3   *
4   * @param input DataInput
5   */
6  protected synchronized void process(InputData input) {
7      try {
8          // 1. assemble data
9          Event event = assembleData(input);
10         // 2. store data
11         int errorCode = storeData(event);
12         // 3. send data
13         sendData(input, errorCode);
14     } catch (Exception e) {
15         SALog.printStackTrace(e);
16     }
17 }
```

</> com.sensorsdata.analytics.android.sdk.data.adapter.EventDataOperation#insertData()

Java

```
1  @Override
2  int insertData(Uri uri, JSONObject jsonObject) {
3      try {
```

```
4         if (deleteDataLowMemory(uri) != 0) {
5             return DbParams.DB_OUT_OF_MEMORY_ERROR;
6         }
7         int instant_event = InstantEventUtils.isInstantEvent(jsonObject);
8         ContentValues cv = new ContentValues();
9         String eventJson = jsonObject.toString();
10        cv.put(DbParams.KEY_DATA, eventJson + "\t" + eventJson.hashCode());
11        cv.put(DbParams.KEY_CREATED_AT, System.currentTimeMillis());
12        cv.put(DbParams.KEY_IS_INSTANT_EVENT, instant_event);
13        contentResolver.insert(uri, cv);
14    } catch (Throwable e) {
15        SALog.i(TAG, e.getMessage());
16    }
17    return 0;
18 }
```

</> com.sensorsdata.analytics.android.sdk.data.SensorsDataContentProvider#insert

Java

```
1  @Override
2  public Uri insert(Uri uri, ContentValues values) {
3      // 不处理 values = null 或者 values 为空的情况
4      if (values == null || values.size() == 0) {
5          return uri;
6      }
7      try {
8          int code = uriMatcher.match(uri);
9          if (code == SAProviderHelper.URI_CODE.EVENTS) {
10             return mProviderHelper.insertEvent(uri, values);
11         } else if (code == SAProviderHelper.URI_CODE.CHANNEL_PERSISTENT) {
12             return mProviderHelper.insertChannelPersistent(uri, values);
13         }
14     } catch (Exception e) {
15         SALog.e(TAG, "insert error: " + e.getMessage());
16     }
17     return null;
18 }
```

```
13         } else {  
14             mProviderHelper.insertPersistent(code, uri, values);  
15         }  
16         return uri;  
17     } catch (Exception e) {  
18         SLog.printStackTrace(e);  
19     }  
20     return uri;  
21 }
```