[Common] 启动优化

文档历史

版本	修订日期	修订者	修订内容
1.0.0	2020.12.28	honggang.xiong	初版

1、需求概况

1.1 需求背景

目前 SPX Driver App(v4.5.0) 冷启动时有较长的黑白屏，用户体验不佳，需分析原因，优化体验 。

1.2 需求目的

(1) 查看应用的启动时间，分析影响启动速度的因素，目标是启动速度能提升20%以上；

(2) 冷启动视觉优化，消除启动时的黑白屏。

2、技术方案设计

2.1 整体方案设计

可包含方案选型、整体架构、数据流、模块间关系等，如果是部分模块变动，需突出变动及影响部分，需对核心模块及关系进行文字描述，需具备良好的架构特性如通用性、扩展性等

本次优化分为代码逻辑优化及视觉优化。在代码方面，通过优化部分初始化项，达到减少启动时间的目的；在视觉方面，通过优化启动窗口StartingWindow，达到秒启动的视觉效果。

2.2 详细方案设计

可包含核心模块功能和交互实现方案详细设计，如：路由、布局、组件、交互、数据流、状态管理、兼容性、性能优化、异常处理、安全防护、扩展性及注意事项等等

2.2.1 代码逻辑优化

Android App 有三种启动状态，每种状态都会影响应用向用户显示所需的时间：冷启动、温启动或热启动。在冷启动中，应用从头开始启动。在另外两种状态中，系统需要将后台运行的应用带入前台。在假定冷启动的基础上进行优化，也可以提升温启动和热启动的性能。详细启动流程可以参考 官方文档。

冷启动耗时通过以下命令统计（-S 表示重启当前应用）：

```
adb shell am start -W -S com.shopee.fms/.ui.activity.LoginActivity
```

优化前的耗时如下：

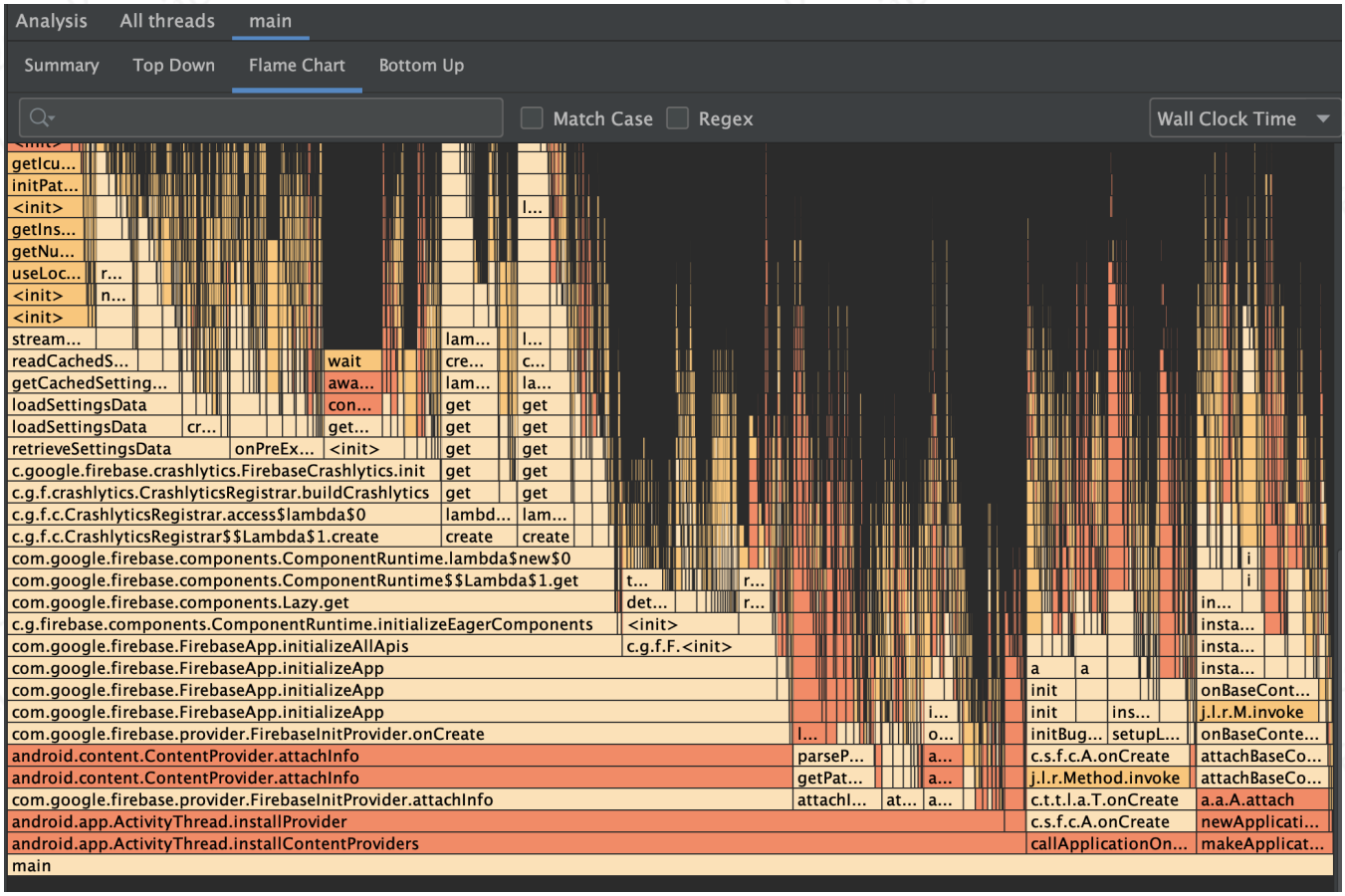
```
MacBookPro:nyear honggangxiong$ adb shell am start -W -S com.shopee.fms/.ui.activity.LoginActivity
Stopping: com.shopee.fms
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.shopee.fms/.ui.
activity.LoginActivity }
Status: ok
Activity: com.shopee.fms/.ui.activity.LoginActivity
ThisTime: 665
TotalTime: 665
WaitTime: 689
Complete
```

App 在冷启动过程中会顺序调用以下回调 Application.attachBaseContext(Context) → Provider.onCreate() → Application.onCreate() → Activity.onCreate (Bundle)，这些回调中的所有方法调用都将影响启动时间。

在 SPX Driver App(v4.5.0) 的冷启动回调中，进行的初始化操作及耗时如下：

- install MultiDex、tinker (15ms)
- init firebase components (100ms)
- setupLeakCanary (9ms，release 模式耗时可忽略)
- init env (58ms，release 模式耗时可忽略)
- init bugly (81ms)
- init sound (4ms)
- init toast (0ms)
- setup UIMatchingHelper (1ms)
- init tracker (1ms)
- init logan (9ms)
- LoginActivity.onCreate(Bundle) (80ms)

Method Tracing 如下 (通过 Debug Api 记录及导出 trace)：



其中耗时较长的操作有：

- init firebase，通过 FirebaseInitProvider 初始化 firebase 所有组件，涉及模块较多，暂不做调整；

- init bugly，目前 firebase 已接入 crash report，bugly 移至延迟初始化，后续 firebase 稳定后可移除；
- LoginActivity.onCreate(Bundle)，目前缺少闪屏页，冷启动都通过 LoginActivity，在已有登录记录时 LoginActivity 的布局初始化实际是多余的，通过添加闪屏页 SplashActivity，可减少本项耗时。

整体代码调整为延迟初始化 bugly、添加闪屏页 SplashActivity，优化后的启动耗时如下，提升了 22% 左右：

```
MacBookPro:nyear honggangxiong$ adb shell am start -W -S com.shopee.fms/.ui.activity.SplashActivity
Stopping: com.shopee.fms
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.shopee.fms/.ui.
activity.SplashActivity }
Status: ok
Activity: com.shopee.fms/.ui.activity.SplashActivity
ThisTime: 517
TotalTime: 517
WaitTime: 536
Complete
```

后续初始化项更多更复杂时，可以考虑引入初始化库，eg: Jetpack Startup。

2.2.2 视觉优化

StartingWindow 的处理方式：

1. 使用系统默认的 StartingWindow：用户点了应用图标启动应用，马上弹出系统默认的 StartingWindow（就是做动画的那个 Window），等应用加载好第一帧之后，StartingWindow 消失，显示应用第一帧，无缝衔接，体验还不错，这也是通常大部分 Android 应用的场景；比如大部分 Android 系统的自带应用，即刻、汽车之家等；
2. 自己定制简单的 StartingWindow：用户点了应用图标启动应用，弹出应用自己定制的StartingWindow，等应用加载好第一帧之后，定制的 StartingWindow 消失，显示应用主界面，由于 StartingWindow 是自己定制的，启动的时候 Decode Bitmap 或者 Inflate 自定义 Layout 会有一定的耗时，但是总的来说与系统默认的差别不大，用户体验优；这样的应用包括淘宝、京东、微博、今日头条、美团等；
3. 把 StartingWindow 禁掉或者设置透明：用户点了应用图标启动应用，由于 StartingWindow 被禁掉或者被设置透明，所以会出现点击图标后，除了图标黑一下之外没有任何响应，过个 1-N 秒（取决于应用第一帧的加载速度），直接显示应用主界面。这样的**毒瘤**应用包括：微信、微信读书、UC 浏览器、支付宝、工商银行、米家等。

-------摘自《知乎 救救你的 StartingWindow》

针对 SPX Driver App，采用第二种方式，设置白色背景+spx logo 为主题的 windowBackground，即可优化冷启动视觉效果。

背景 xml 如下:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape>
            <solid android:color="@color/white" />
        </shape>
    </item>
    <item>
        <bitmap
            android:gravity="center"
            android:src="@drawable/logo_signature" />
    </item>
</layer-list>
```

2.3 接口设计

对外提供的接口相关信息，如接口名称、协议、参数，及注意事项等（如未涉及则不需填写）

2.4 数据存储设计

数据库、表、文件等存储方案设计（如未涉及则不需填写）

2.5 监控方案设计

可包含业务监控如PV、点击、曝光、用户行为等，及质量监控如页面性能、错误、API质量等部分

2.6 部署方案设计

可包括部署相关准备工作如机器、资源申请等，部署顺序、目标节点，灰度和回滚方案等

3、风险评估

列出可能存在的风险及应对策略，如外部依赖、技术成熟度、关联影响、不可控因素等

其他

以上未包含的其他补充项说明