Alibaba Group 阿里巴巴集团

UC浏览器快开之路

阿里巴巴UC移动事业部-刘成

2017.09.02

Alibaba Group 阿里巴巴集团

目录 CONTENTS

启动意味着什么？

用户口碑

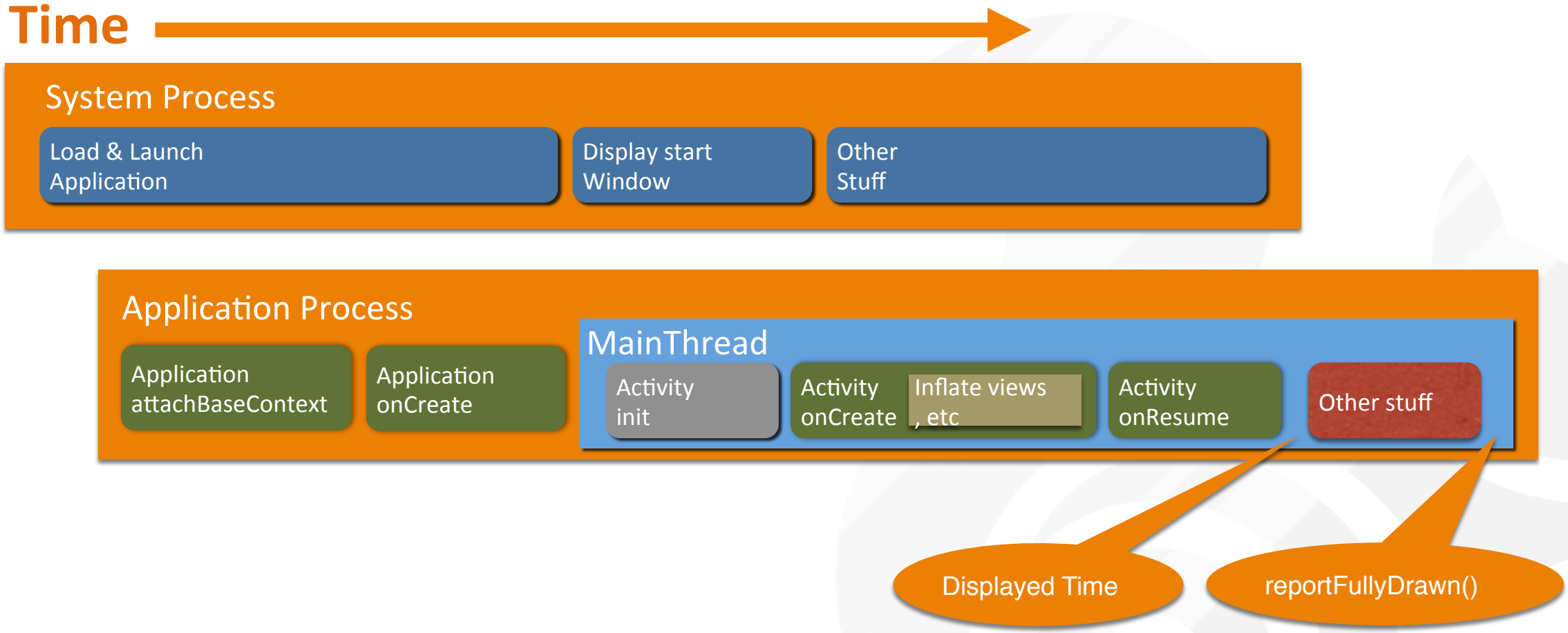
用户对APP的第一认知

三方合作

启动速度是进行三方合作的最基本筹码

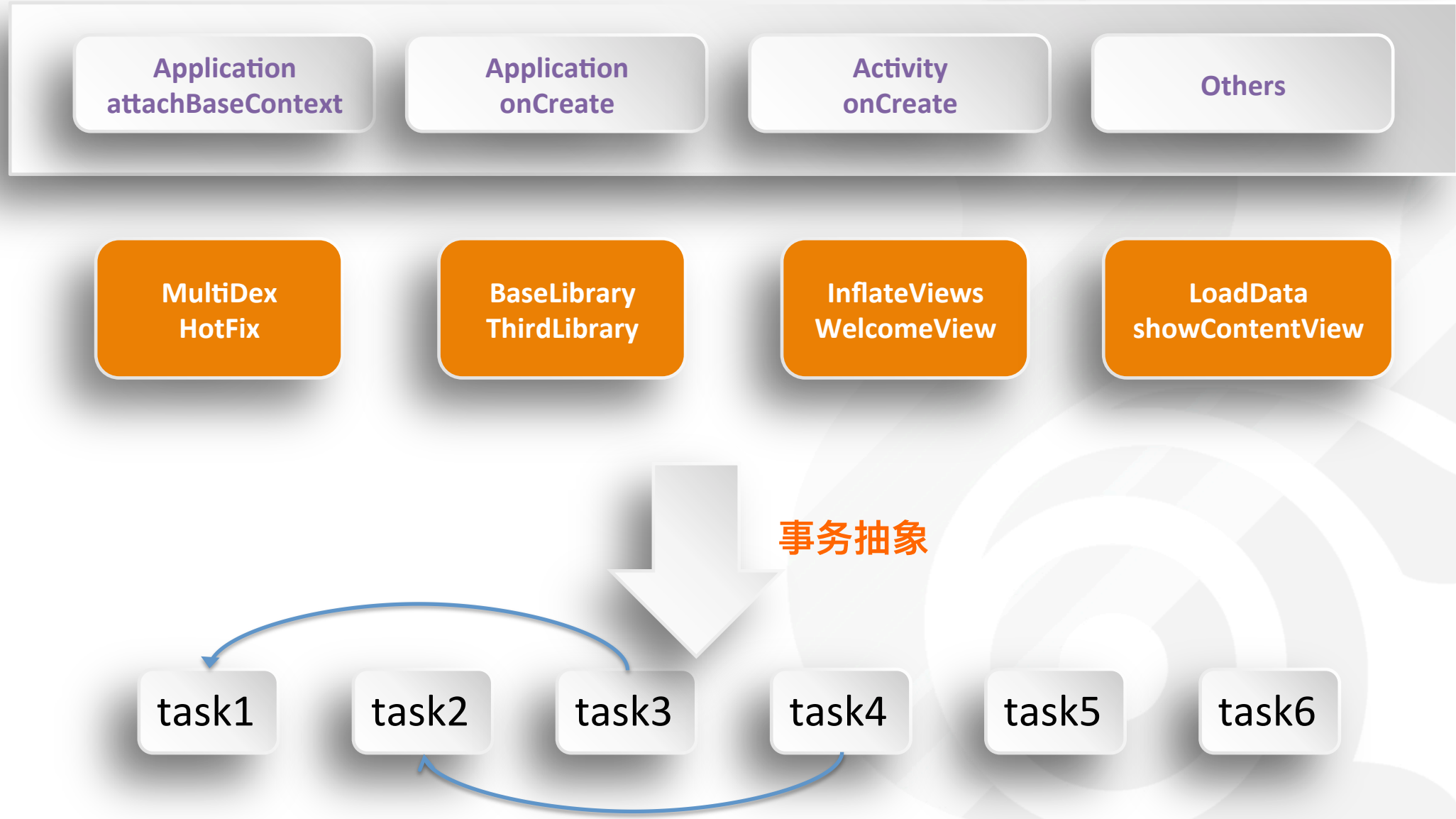
日活留存

启动速度一定程度上影响APP的日活跃和留存情况

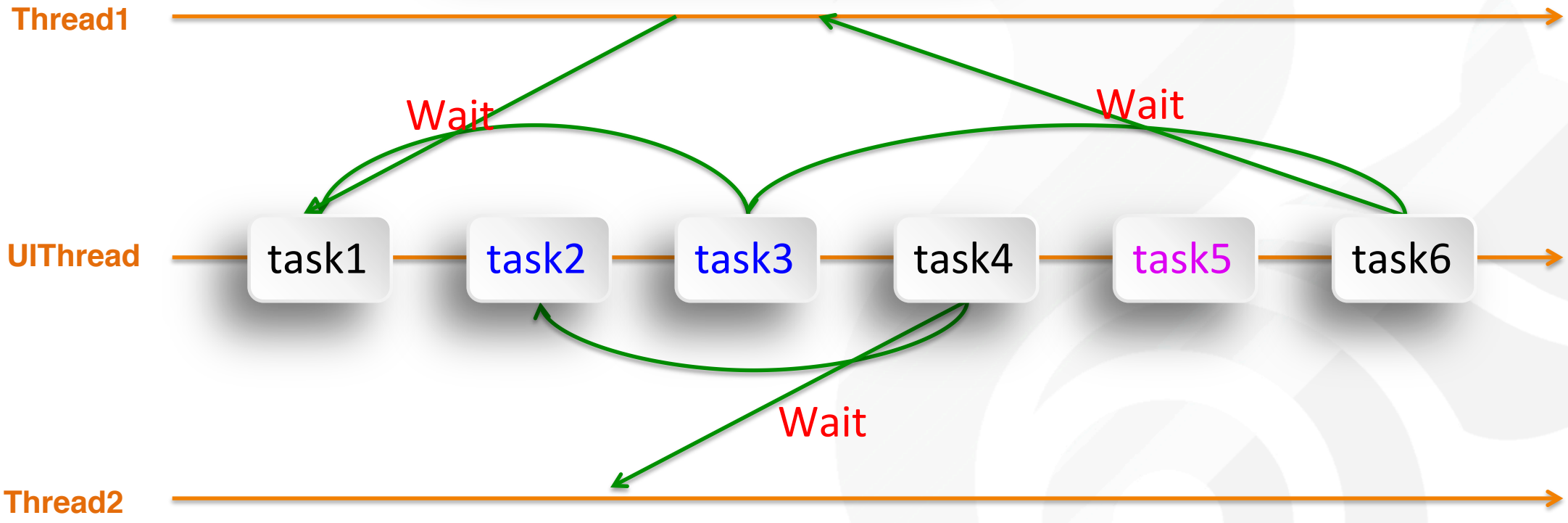


ActivityManager: Displayed {package}/.StartupTiming: +1s111ms

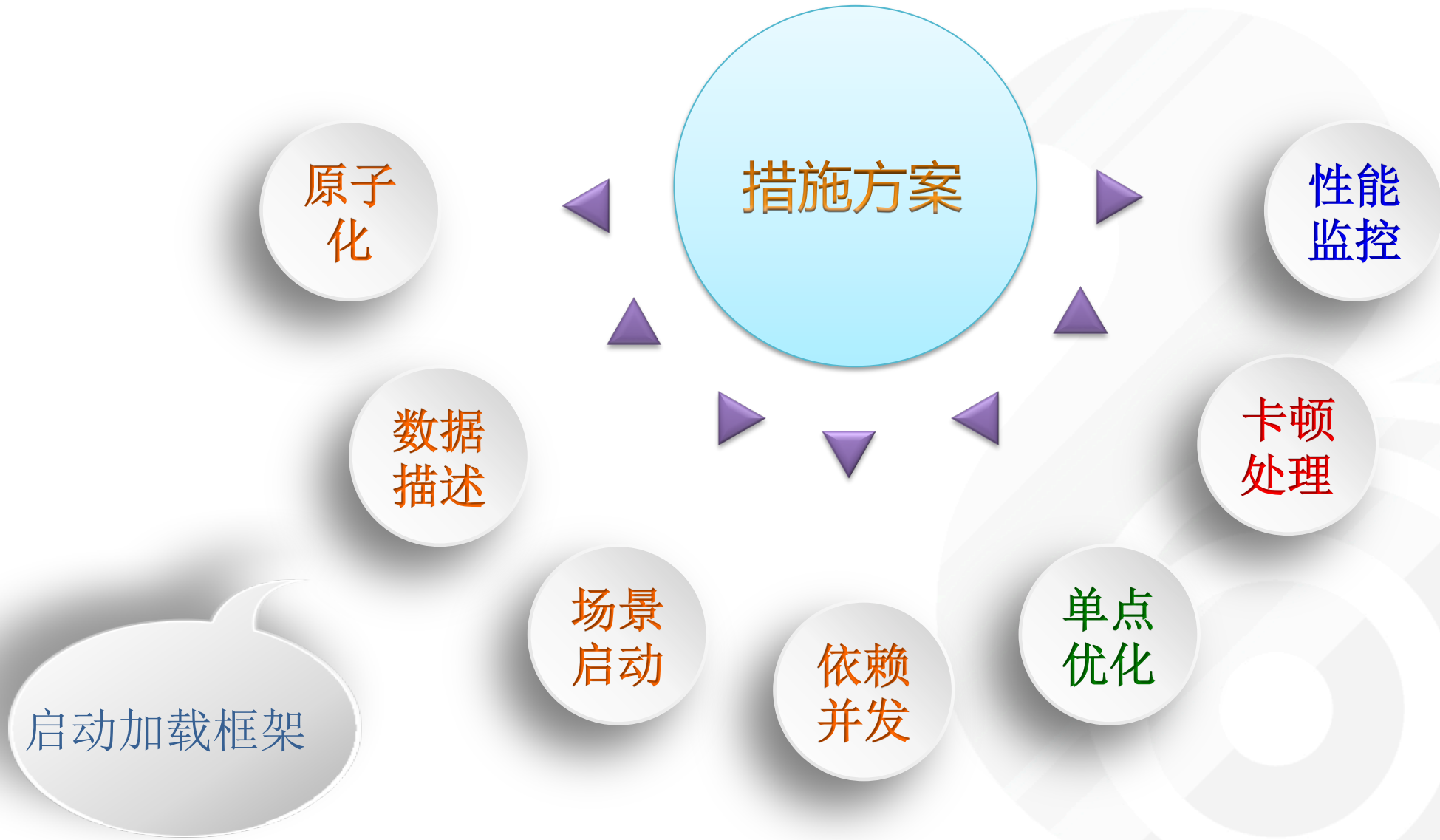
system_process I/ActivityManager: Fully drawn {package}/.MainActivity: +1s540ms



常规启动优化方案

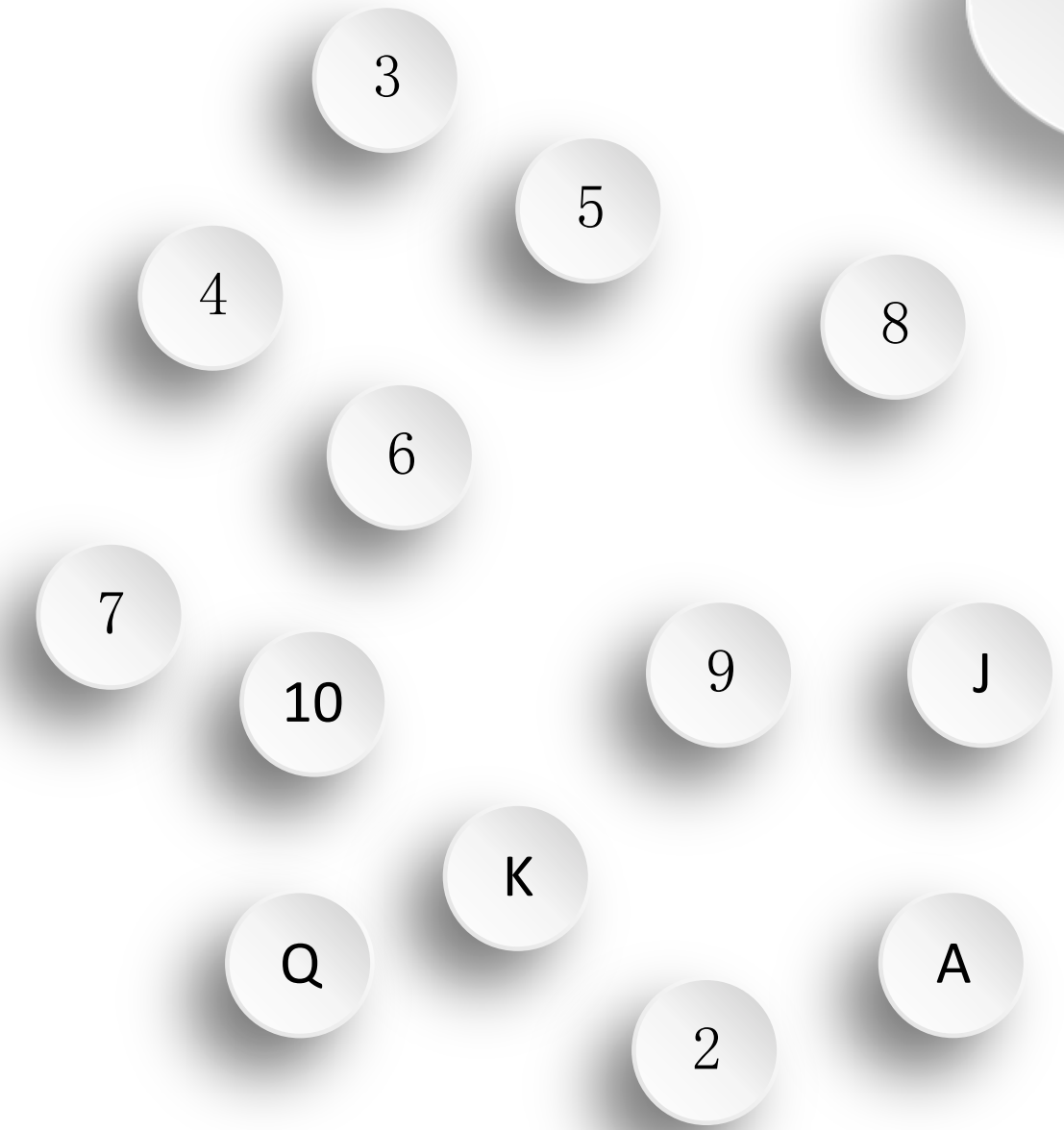


- 并发导致的代码复杂度如何处理？
- 并发依赖如何解决？
- 不当并发导致的性能瓶颈和性能下滑如何处理？
- 延时任务导致启动后卡顿如何处理？
- 启动速度增长问题能否提前发现，而不是事后优化？
- 优化成果能否坚守下来，能不再做周而复始的优化工作吗？



Alibaba Group
阿里巴巴集团

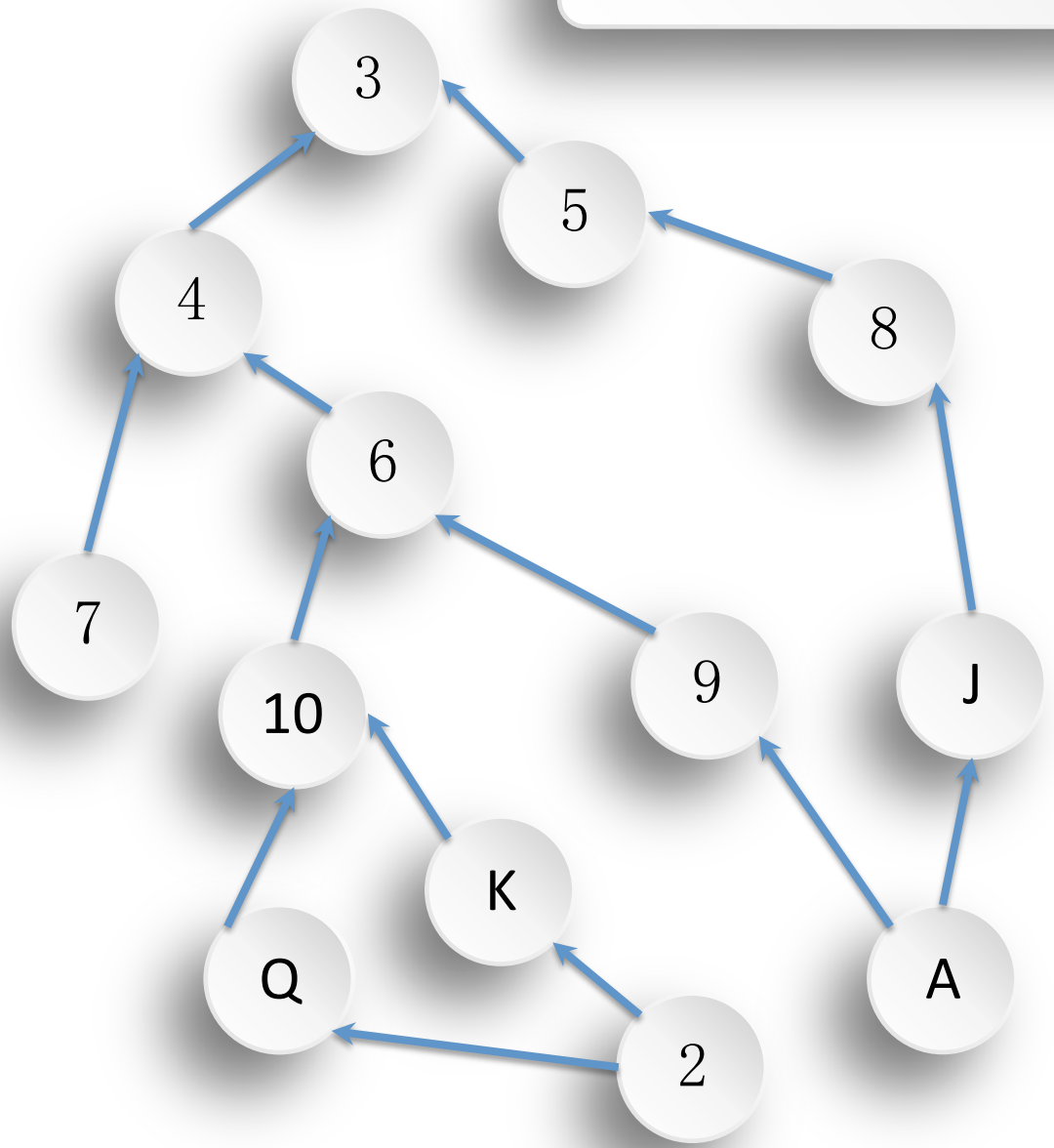
启动加载框架



加载什么？

怎么加载？

启动任务原子化



目的

解决环形依赖

任务依赖最小化

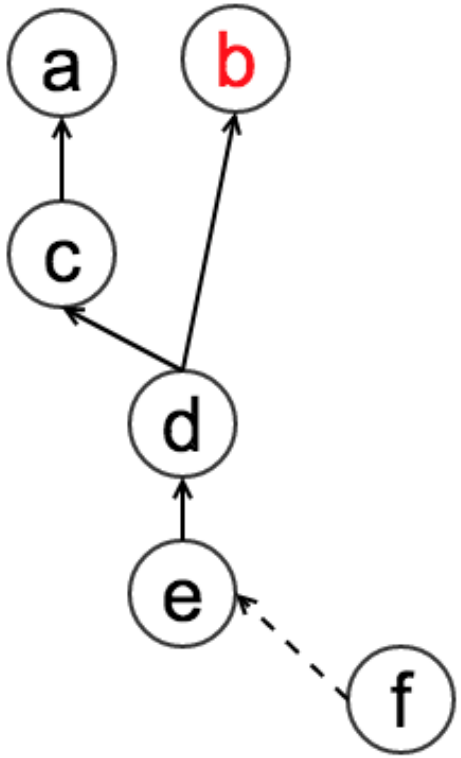
原则

业务内聚

颗粒度<200ms

Alibaba Group 阿里巴巴集团

可描述



DAG

```
//step1 构建需要执行的任务，原子化任务
Task a = new TaskA();
Task b = new TaskB();
Task c = new TaskC();
Task d = new TaskD();
Task e = new TaskE();
Task f = new TaskF();
```

启动任务

```
//step2 进行一些工程配置，不是非必须
Config.setXXX();
Config.setXXX();
Config.setXXX();
```

启动工程配置

```
//step3 构建工程依赖关系图
Project.Builder builder = Project.createBuilder();
builder
    .add(a) // 添加任务a，这个任务默认跑在非UI线程
    .add(b, true) // 添加任务b，第二个参数表示b任务运行在UI线程
    .add(c).depend(a) //添加任务c，任务c必须依赖任务a完成才能执行
    .add(d).depend(b, c) // 添加任务d，任务d必须依赖b和c都完成才能执行
    .add(e).depend(d).barrier() //添加任务e，该任务是一个屏障,拦住后续任务
    .add(f);
Project project = builder.finish(); //构建完整个启动依赖关系图

//step4 开始执行工程
project.start(); //开始执行整个工程
```

启动数据描述

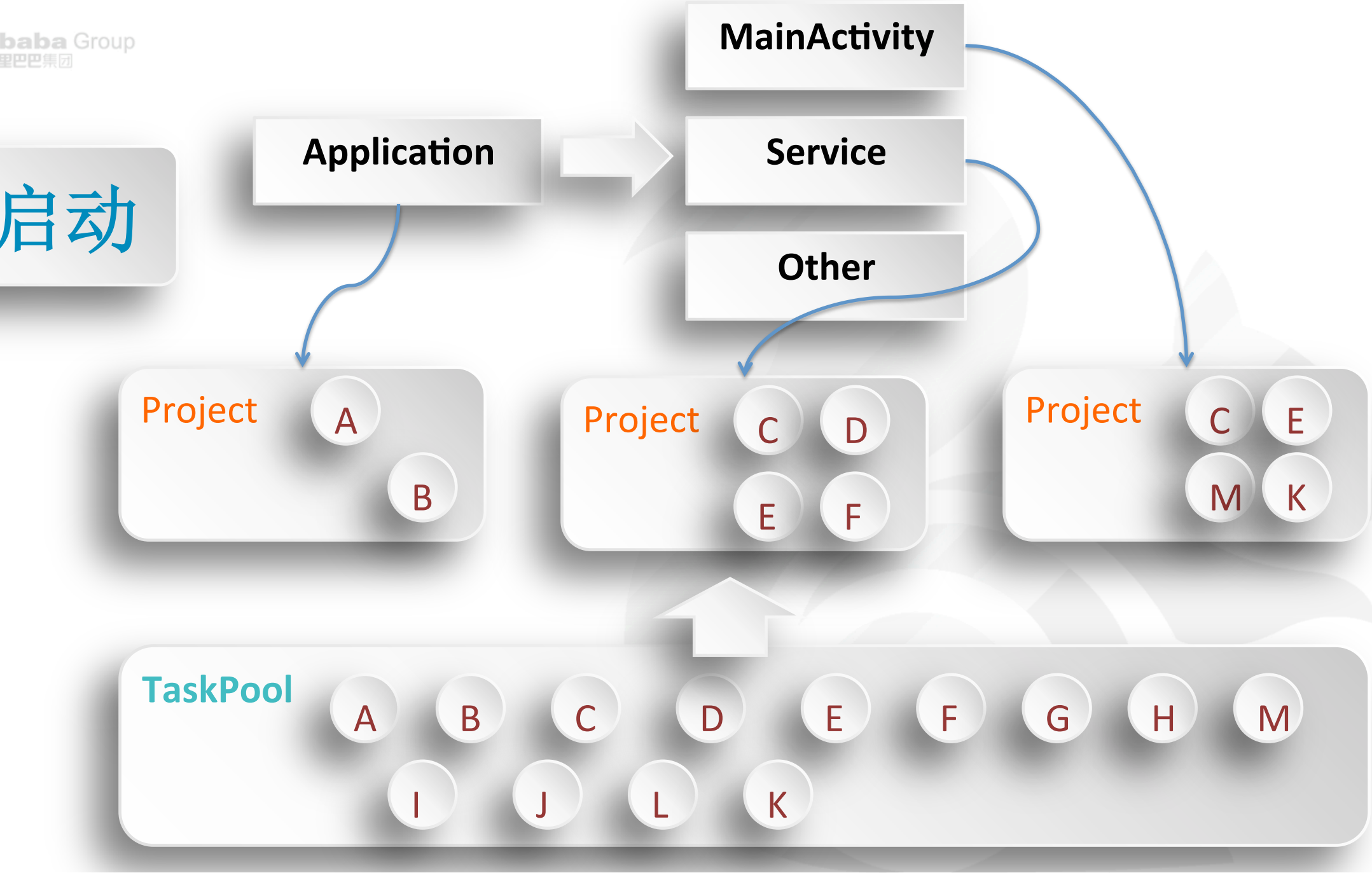
adb shell am start –n com.UCMobile/com.UCMobile.main.UCMobile –seq “id1,id2,id3,…”

启动要把所有任务都加载完吗？

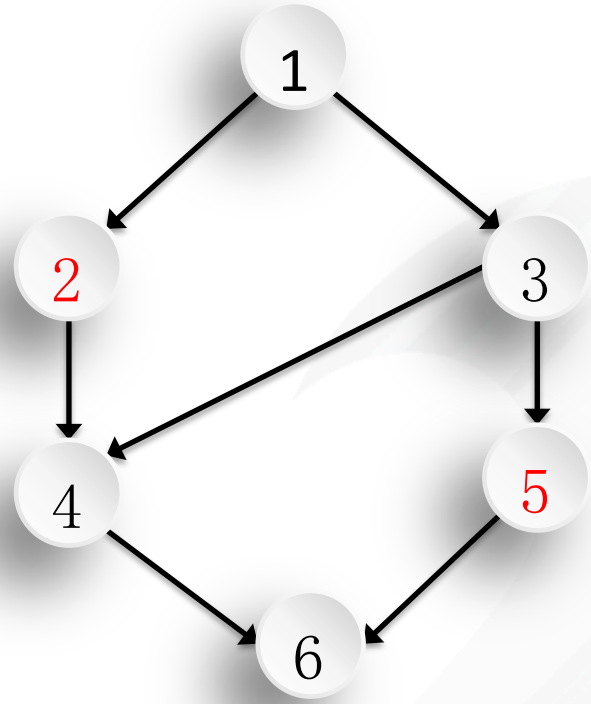
浪费不必要的内存

启动时间过长

场景启动



依赖并发



从入度为0的点开始

UIThread

Thread1

Thread2

并发引入新的难题

启动任务
近30项

启动序列
7W+

如何保障
质量？

并发加载框架上线风险规避方案

保证本地以及实验室的机型正常（自动化+截图）

1 本地保障

灰度验证

2 收集启动序列

Top300的启动序列pv占比98%
本地自动化脚本验证Top300启动序列
2%通过线上数据监控做优化

3 序列校验

线上切换

4 保持原串行启动方案
线上可动态切换回
原串行启动逻辑

Alibaba Group 阿里巴巴集团

单点优化

实现优化

Tips

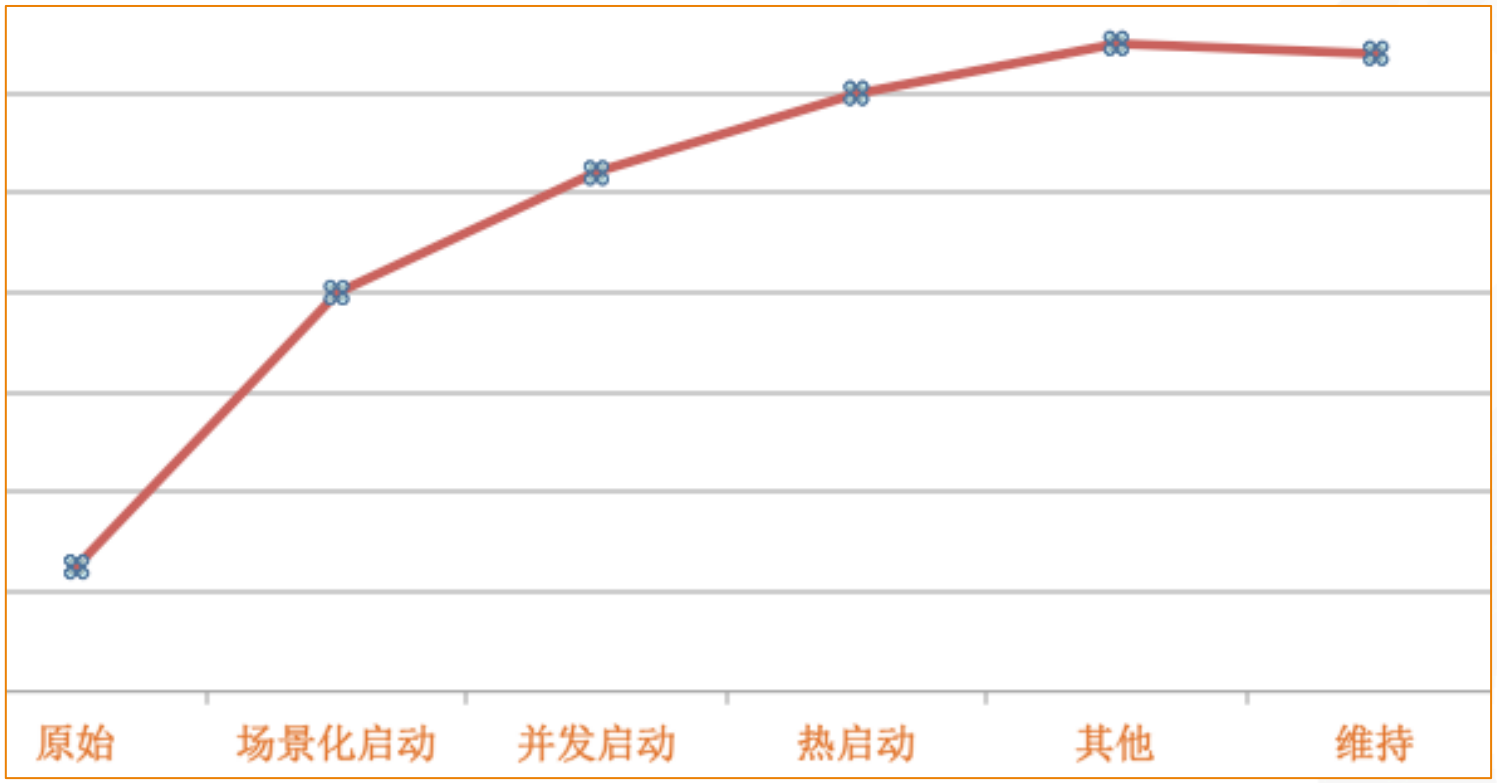
- 避免嵌套UI/频繁反射
- 超级类做预加载
- 预启动
- IO任务集中到一个后台线程
- 启动过程内存数据选择落地
- 启动关联类集中到主dex

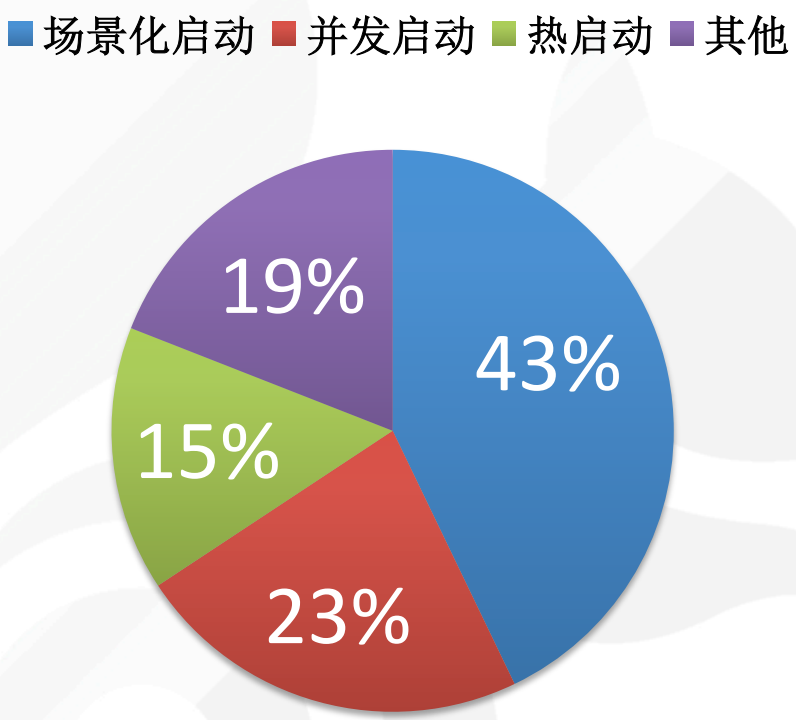
交互优化

Tips

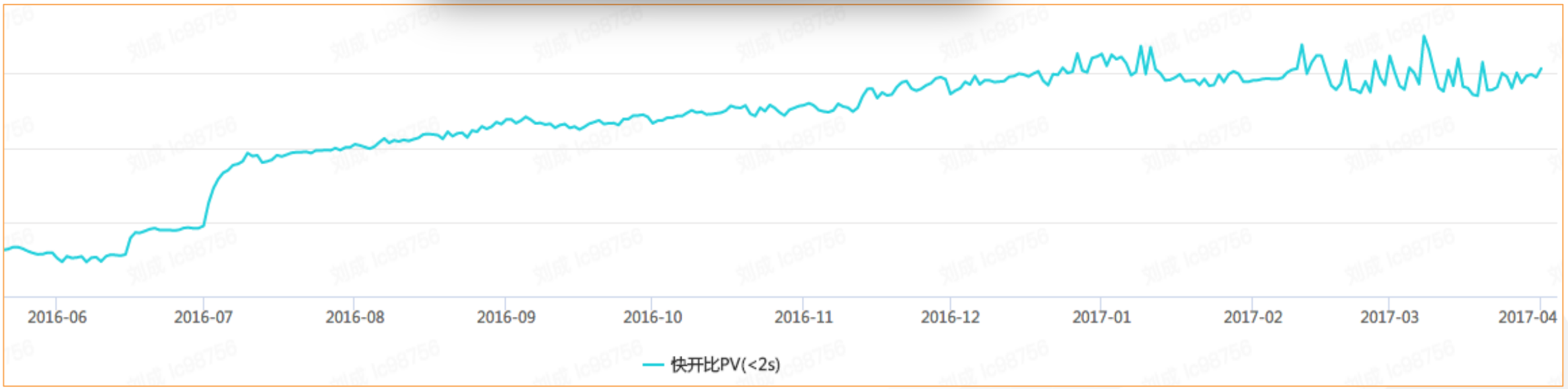
- 截图启动
- Loading界面启动
- 主题设置

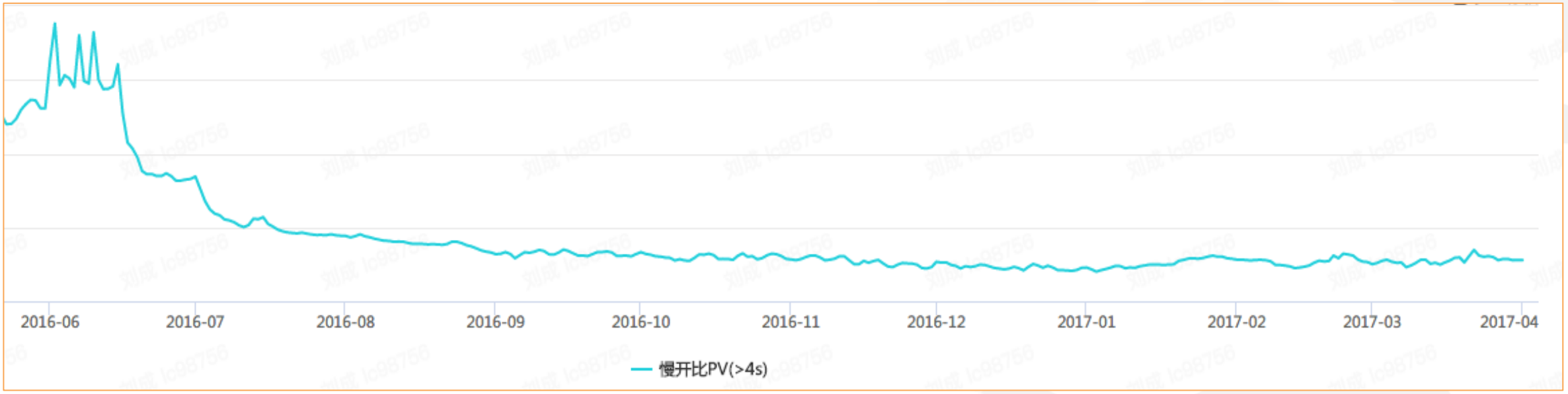
优化效果说明





大盘数据

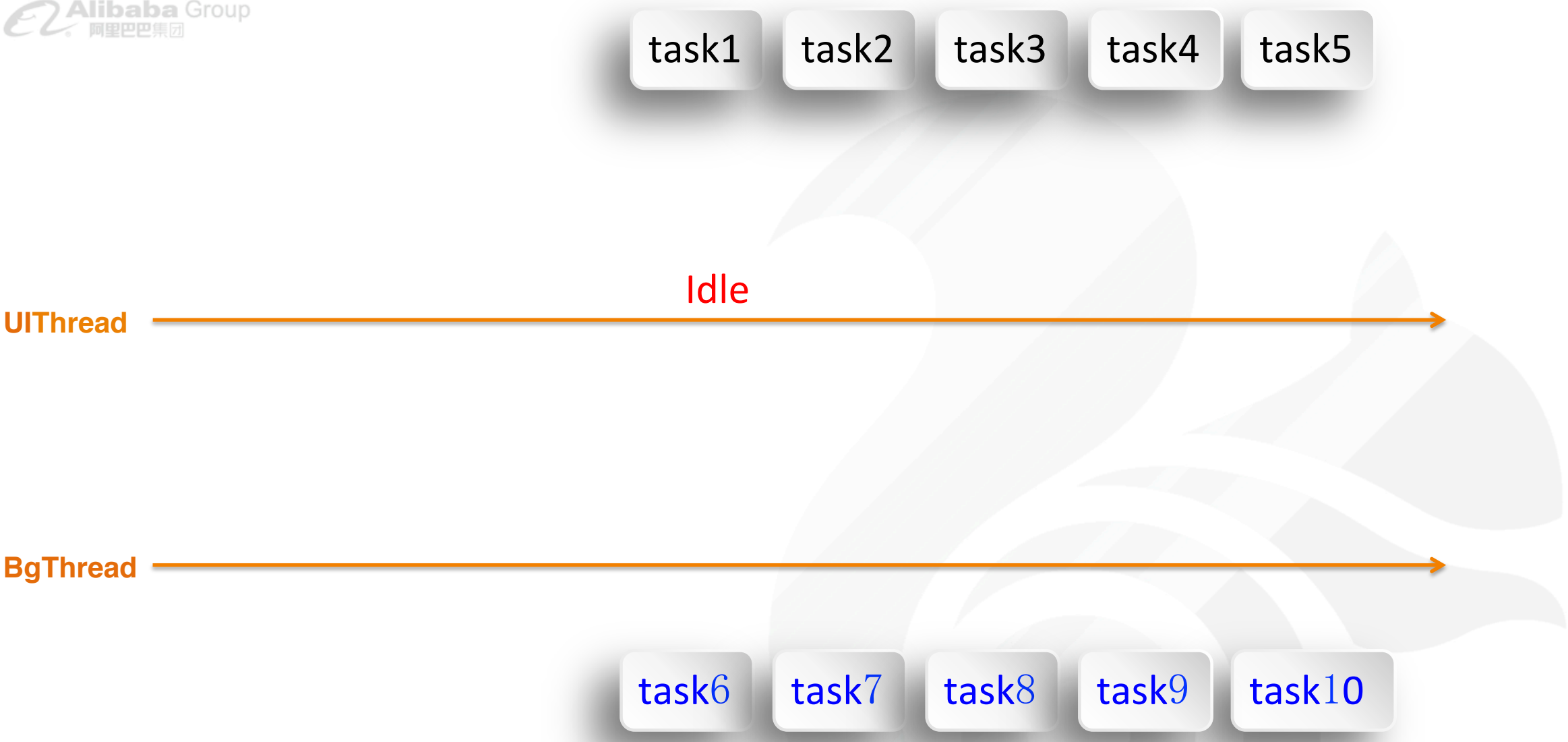

2016-06
2016-07
2016-08
2016-09
2016-10
2016-11
2016-12
2017-01
2017-02
2017-03
2017-04
快开比PV(<2s)


2016-06
2016-07
2016-08
2016-09
2016-10
2016-11
2016-12
2017-01
2017-02
2017-03
2017-04
慢开比PV(>4s)

Alibaba Group
阿里巴巴集团

启动后卡顿处理

启动后事务太多



Alibaba Group 阿里巴巴集团

性能监控-启动速度

怎么更早的发现问题？

启动速度变慢怎么
快速定位问题？

启动监控代码对原本启动
流程的入侵怎么解决？

代码提交

实验室

线上控量

Alibaba Group 阿里巴巴集团

文件名	问题描述	责任人	新增时间	类型	优先级	状态	操作
[redacted]xml：60	UselessParent	[redacted]	2017-08-22 18:43:32	lint	中	未解决	忽略 指派 认领 变更
[redacted].xml：6	UselessParent		2017-05-03 13:49:50	lint	中	未解决	忽略 指派 认领 变更
[redacted]r.java：77	ApplySharedPref		2017-06-22 15:08:35	lint	中	未解决	忽略 指派 认领 变更
[redacted].java：204	ApplySharedPref		2017-06-22 15:08:35	lint	中	未解决	忽略 指派 认领 变更

Alibaba Group 阿里巴巴集团

UCTC

测试用例	测试设备	性能指标	V11.6.6.951(base,170823202242)	V11.6.6.951(test,170823202029)	对比	详细对比
【核心数据】启动时间_首次20_交叉_关闭截图	Nexus5_Slave5_1	启动时间(screencap)切尾均值(ms)	0	0	0	N/A
		启动时间(benchmark)切尾均值(ms)	1277.83	1262	-15.82	查看
		启动时间(wa log)切尾均值(ms)	1530.33	1518.67	-11.65	查看
		退出时间(benchmark)切尾均值(ms)	1306.92	1303.58	-3.34	查看
		启动时间(intent)切尾均值(ms)	1489.13	1476.5	-12.63	N/A
		有效测试次数	20	20	0	N/A
【核心数据】启动时间_非首次30_交叉_关闭截图	Nexus5_Slave5_4	启动时间(screencap)切尾均值(ms)	0	0	0	N/A
		启动时间(benchmark)切尾均值(ms)	1011.61	988.84	-22.76	查看
		启动时间(wa log)切尾均值(ms)	1284.56	1274.84	-9.72	查看
		退出时间(benchmark)切尾均值(ms)	1223	1221.05	-1.95	查看
		启动时间(intent)切尾均值(ms)	1347.38	1340.69	-6.69	N/A
		有效测试次数	30	29	-1	N/A

Alibaba Group 阿里巴巴集团

详细对比数据

整体对比	基准版本(503138; 有效次数:30)	监控版本(503139; 有效次数:29)	对比差值	对比百分比
intervalStartTime	1018.73	1001.17	-17.56	-1.72%
子项对比	-	-	-	-
21	69.9	81.58	11.67	16.7%
22	280.06	284.95	4.89	1.74%
11	25.3	28.31	3	11.89%
27	34.79	35.06	0.26	0.74%
8	0.4	0.48	0.07	19.99%
14	0.4	0.44	0.03	9.99%
15	0.1	0.13	0.03	30%
4	0.03	0.06	0.02	80%
19	25.26	25.27	0	0.01%
26	0.1	0.06	-0.04	-40%
5	14.93	14.86	-0.07	-0.49%

UCTC

```
@Aspect
public class StartupTraceAspect {
    ...
                任务函数执行前后
    @Around("execution(* com.[redacted](..)) && enabledTrace()")
    public void startTaskTracing(ProceedingJoinPoint joinPoint) throws Throwable{
        int taskId = ((Task)joinPoint.getTarget()).getTaskId();
        boolean traceThisTask = traceSteps.contains(taskId); // 获取此启动步骤是否需要tracing
        if (traceThisTask) {
            startMethodTracing("" + taskId); //开始trace
        }
        joinPoint.proceed(); //原任务执行
        if (traceThisTask) {
            stopMethodTracing(); //结束trace
        }
    }

    ...
}
```

打包开关

traceSteps中的内容是从本地文件读取

Alibaba Group
阿里巴巴集团

性能监控-启动后卡顿

用户不操作界面不刷新怎么衡量帧率？

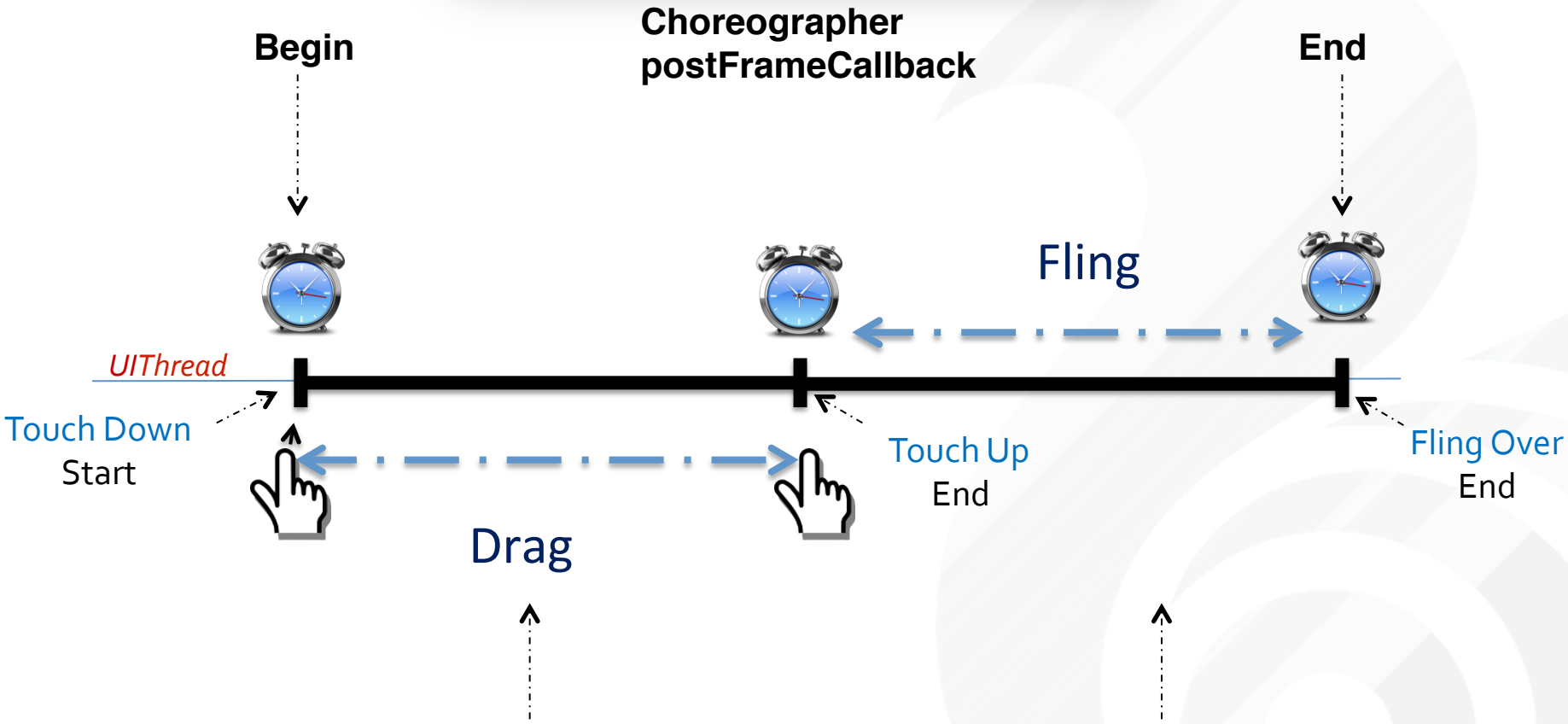
```
Choreographer.FrameCallback mFrameCallback = new Choreographer.FrameCallback() {
    @Override
    public void doFrame(long frameTimeNanos) {
        mFrames++;
        Choreographer.getInstance().postFrameCallback(mFrameCallback);

    }
};
Choreographer.getInstance().postFrameCallback(mFrameCallback);
```

如何减少监控
带来的性能开销？

缩短监控帧率的时间

只在用户操作的短暂时间内开启

如何监控用户操作过程？



卡顿帧率监控日志

```
---
PreV
PreI
PreJ
CurV
CurI
CurJ
Drag:false
TouchPoint:Point(561, 1807)
ScreenPoint:Point(1080, 1920)
SM:28

--- --- --- --- --- --- --- --- --- --- ---
```

Alibaba Group
阿里巴巴集团

Q&A

Alibaba Group
阿里巴巴集团

THANKS!