

# 滴滴客户端构建

作者：郑涛



**SFDC**

SegmentFault  
Developer Conference

# 目录

1

开发规范及构建流程

2

构建挑战

3

插件化支持

4

未来规划



# 开发规范及构建流程



**SFDC**

SegmentFault  
Developer Conference

# 不同团队

快车

出租车

顺风车

平台SDK



SFDC

SegmentFault  
Developer Conference



# 三个阶段

8+业务线，60+开发，分布北京、上海、杭州

Eclipse时代

AS初期

Now

单项目



多项目基于  
源码集成



多项目基于  
**aar**集成

牵一发  
动全身

要求高  
难维护

较理想



SFDC

SegmentFault  
Developer Conference

# 开发规范

平台SDK : 基础模块, 公共功能

业务线 : 业务迭代

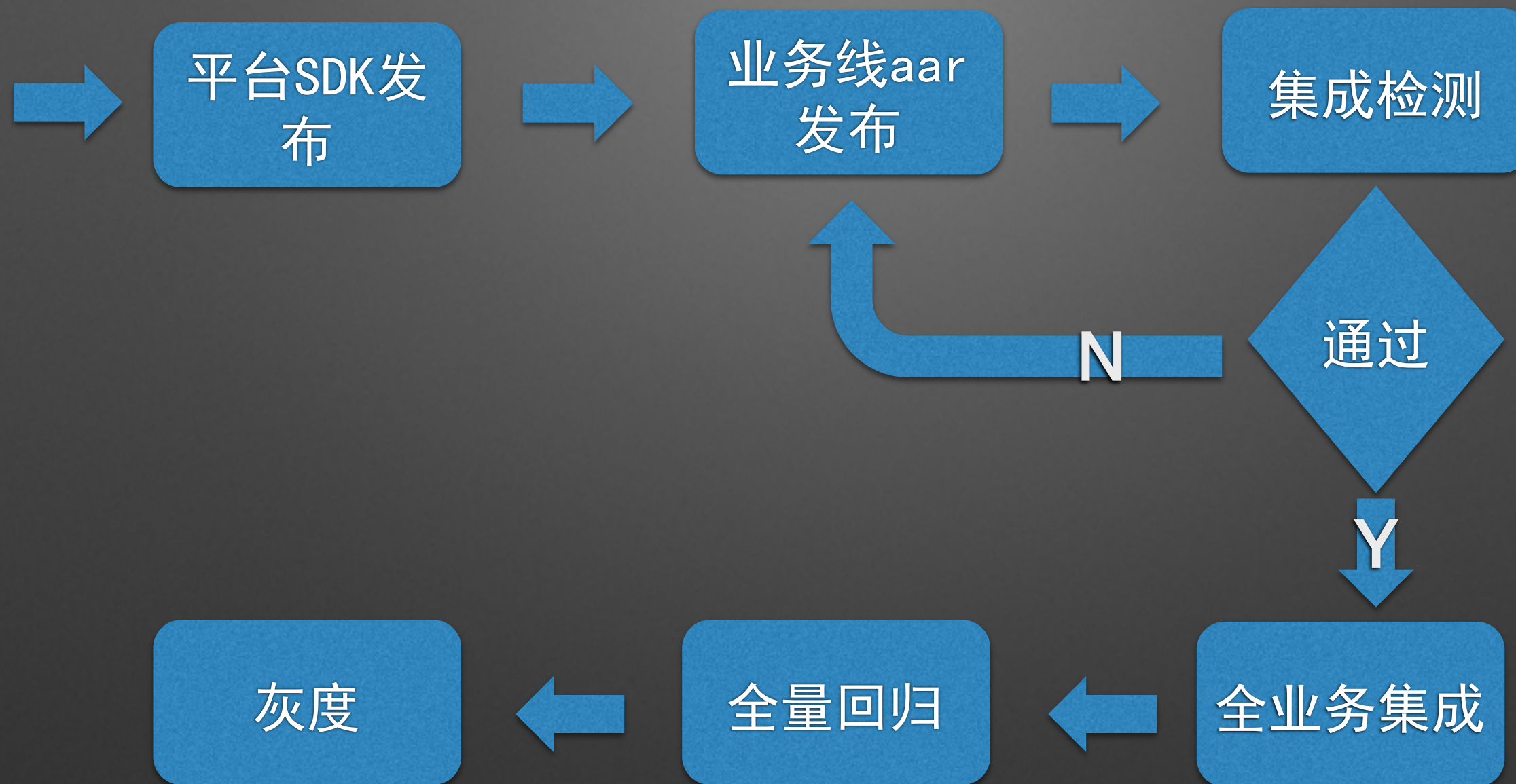
业务组件入口 : SPI 机制, 解耦

资源命名 : 业务前缀, 避免资源冲突

协作: 平台发版周期内提前输出SDK



# 构建流程





# AAR发布流程



需要如下参数用于构建项目:

BRANCH

master

请填写需要构建的分支名, commitid, tag等

CONFIG

Release

构建类型: Release或者Debug

UPLOAD\_AAR

☐

是否需要上传该aar版本到maven。如需要, 请勾选

INTEGRATION

☐

是否需要集成到主工程。如需要, 请勾选。勾选后会同时上传aar到Maven。  
注意: 集成之前会先进行编译检查, 检查通过才会集成该aar版本。  
编译检查Job: **one\_android\_integration\_compile**

开始构建



SFDC

SegmentFault  
Developer Conference



# 集成检测

是什么？

集成工程Git分支+Jenkins定制Job

啥作用？

减少构建失败概率，定位原因

保护发布环境，使其稳定



# 构建挑战



**SFDC**

SegmentFault  
Developer Conference

# AAR和Project依赖切换



# AAR集成在开发阶段带来的不便:

SDK开发阶段  
调试?

基于aar和project构建互相切换

跨业务线  
联调?

支持依赖多project构建

业务线依  
赖可选?

支持屏蔽业务线依赖





# 依赖project原生方案

settings.gradle

```
include 'facebook-sdk'  
project(':facebook-sdk').projectDir = new File('../FacebookSDK/facebook')
```

build.gradle

```
dependencies {  
    //compile 'com.facebook.mobile:sdk:1.0.0'  
    compile project(':facebook-sdk')  
}
```



# 屏蔽aar依赖原生方案

build.gradle

```
dependencies {  
  
    // 业务线 aar  
    //compile 'com.didi.zzzz:zzzz:1.0.0'           //租车  
    //compile 'com.didi.bbbb:bbbb:1.0.0'         //巴士  
    //compile 'com.didi.ssss:ssss:1.0.0'         //试驾  
    //compile 'com.didi.mmmm:mmmm:1.0.0'         //顺风车  
    compile 'com.didi.kkkk:kkkk:1.0.0'           //快车  
    compile 'com.didi.tttt:taxi:1.0.0'           //出租车  
  
    // SDK aar  
    compile 'com.didi.sdk:framework:1.0.0'  
}
```



# 原生方案问题

修改多个  
文件

操作繁琐，效率低

git提交前  
回滚

失误会影响团队  
心理负担重



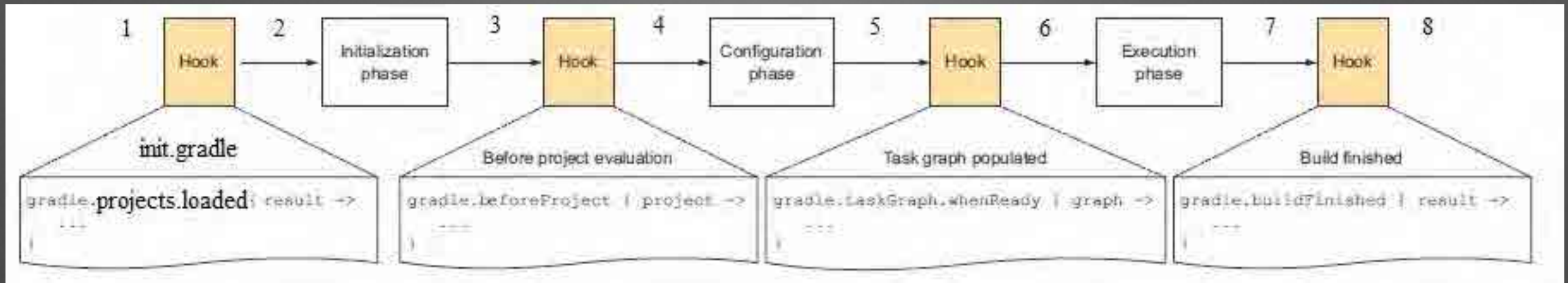


# 改进方案

```
{
  "_comment" : {
    "type": "3种类型: aar(默认, 基于aar编译)
              source(基于源码编译, 需要配置project_dir)
              block(屏蔽该业务线)",
    "project_dir": "源码本地路径, 只有type为source时生效"
  },
  "components": [
    {
      "description": "平台SDK",
      "type": "aar",
      "project_dir": "../PassengerFramework",
      "group_id": "com.didid.sdk",
      "artifact_id": "framework"
    },
    {
      "description": "出租车",
      "type": "source",
      "project_dir": "../Taxi",
      "group_id": "com.didid.tttt",
      "artifact_id": "tttt"
    }
  ]
}
```



# 改进方案



Init

settings.gradle动态include

Config

build.gradle动态compile

# main dex 定制



# 问题演进

Too many method references: xxxx; max is 65536.



Google MultiDex



Too many classes in --main-dex-list,  
main dex capacity exceeded



# Android默认拆dex方案

collect{variant}MultiDexComponents

manifest\_keep.txt



shrink{variant}MultiDexComponents

componentClasses.jar



create{variant}MainDexClassList

maindexlist.txt



dx.jar





# maindexlist超过65535?

修改gradle task?

定制manifest\_keep.txt生成逻辑

修改mainDexClasses?

定制maindexlist.txt生成逻辑

手工定义maindexlist.txt + Application委托



SFDC

SegmentFault  
Developer Conference

# maindexlist gradle配置

```
1
2 defaultConfig {
3     multiDexEnabled = false
4 }
5
6 afterEvaluate {
7     tasks.matching {
8         it.name.startsWith('dex')
9     }.each { dx ->
10         if (dx.additionalParameters == null) {
11             dx.additionalParameters = []
12         }
13         dx.additionalParameters += '--set-max-idx-number=50000'
14         dx.additionalParameters += "--main-dex-list=$projectDir/maindexlist.txt"
15     }
16 }
```



# 新问题

Android gradle 1.3.0

`dx.additionalParameters += "--main-dex-list=maindexlist.txt"`

Android gradle 1.5.0+



不支持

偷梁换柱



SFDC

SegmentFault  
Developer Conference

# 插件化支持

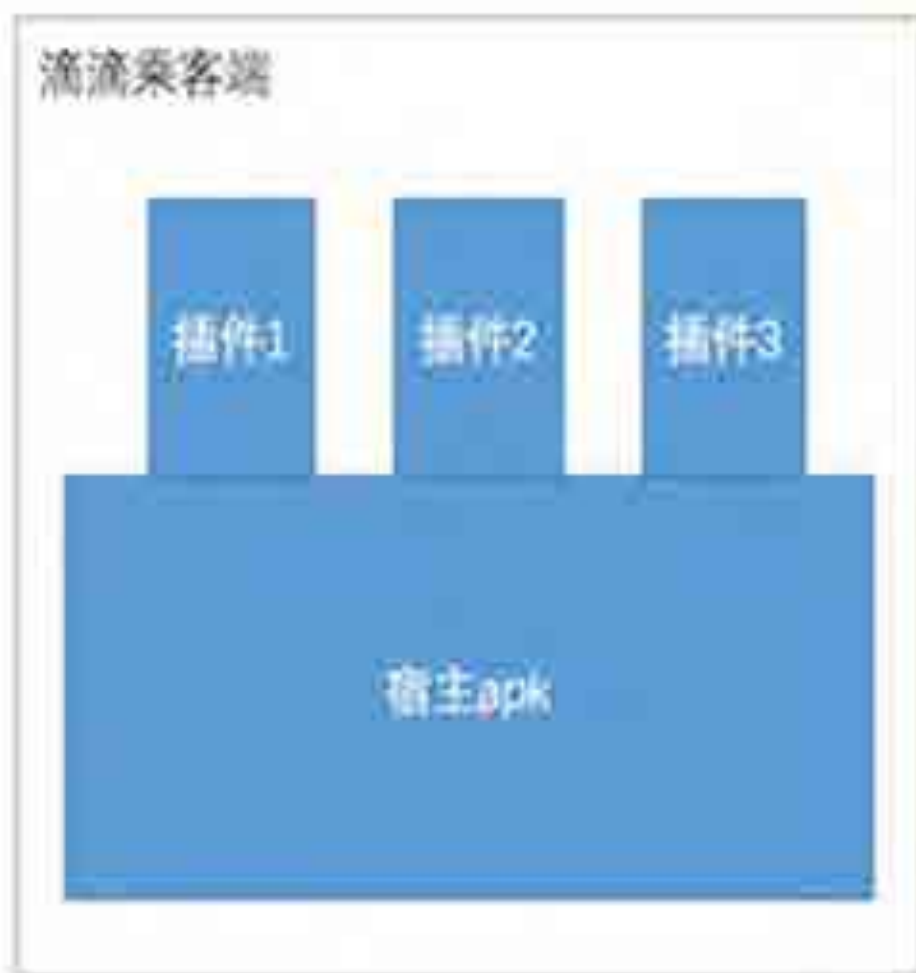


**SFDC**

SegmentFault  
Developer Conference



# 插件化概览



VS



# 宿主端构建

- ◆ 锁定资源ID
- ◆ 保留R符号表、mapping等文件
- ◆ 保留依赖列表



# 插件端构建目标

## 透明、操作简单

- 开发体验：开发插件apk和 aar无区别、无限制
- 构建体验：独立运行apk & 插件apk

## 剔除插件APK中所有宿主已经存在的类和资源

- 增加gradle task: `assemblePlugin`
- 配置宿主R符号表，依赖列表



# 插件端构建思路

**class**

- ① 宿主类(aar, jar) 移除
- ② JavaCompile、Proguard

**resource**

- ① 宿主资源收集、移除
- ② 重建ARSC、XML引用
- ③ Asset、JavaRes





# 未来规划

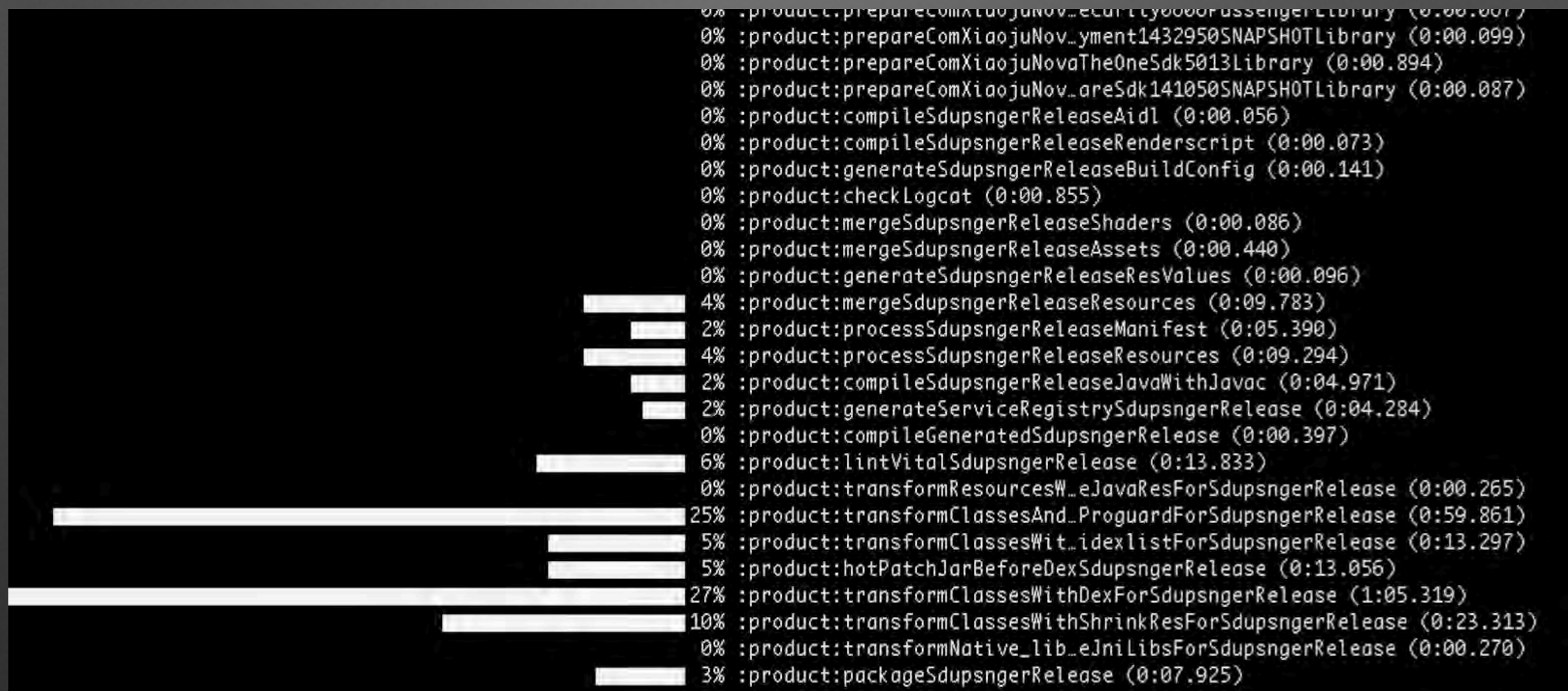


**SFDC**

SegmentFault  
Developer Conference

# 性能

保持现有构建习惯下，提升性能



# 构建期工具链

## 静态检查

扫描检错、组件推广、规则可配置

## 工具整合

资源混淆、app性能分析、AOP





# Q&A



**SFDC**

SegmentFault  
Developer Conference