

携程React Native应用的工程化实践

无线平台研发部

赵辛贵

Agenda

- React Native的使用现状
- CRN框架介绍
- 性能优化探索
- 发布运维
- 经验与实践

选择React Native的原因

性能体验

用户体验佳
接近Native开发水平

动态更新

随时发布
支持产品快速迭代

跨平台

三端运行
降低开发维护成本

包大小

占用App Size小
单页面200/17.2KB

社区资源

社区活跃
资源丰富

RN在携程内部使用现状

13
Apps

- 集团内13个App集成RN
- 核心App全部接入
- 8个App纯RN开发

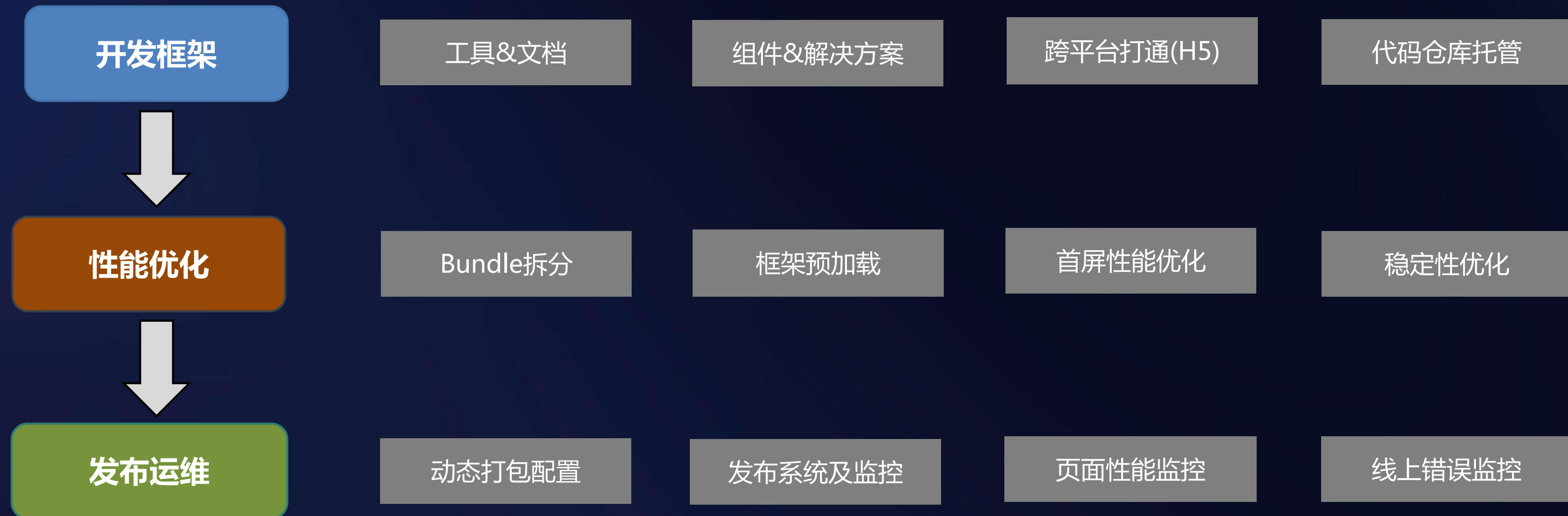
104
Bundles

- 83个在携程旅行App中使用
- 单bundle超过100个Pages
- 核心业务场景使用(机票)

300%
PV增长

- 2016-2018PV年化300%复合增长
- PV在2018超过H5，达到其2倍

CRN框架介绍



CRN是基于React Native定制，适合携程业务的跨平台开发框架，提供从开发、发布、运维全生命周期的跨平台开发解决方案。


```
msi-pc:~ jim$ crn
```

API

类

1.三大组件—App

2.三大组件—Page

3.三大组件—ViewPort

4.全局对象

5.页面间通信

6.导航栏选择

7.跳转Hybird/Native的URL规则

五、CRN的几个设计

1.三大组件—App

2.三大组件—Page

3.三大组件—ViewPort

4.全局对象

5.页面间通信

6.导航栏选择

7.跳转Hybird/Native的URL规则

crn.site.ctripcorp.com

crn.site.ctripcorp.com

页面路由与跳转

1. CRN入门和版本说明文档

2. CRN新增组建和API的设计文档

3. CRN开发常见问题和解决方案文档

有三个界面page1,page2,page3。默认初始化界面为page1

1、url访问 `/rn_helloworld/_crn_config?CRNType=1&CRNModuleName=HelloWorld` 打开page1界面

2、url访问 `/rn_helloworld/_crn_config?CRNType=1&CRNModuleName=HelloWorld&initialPage=page2` 打开page2界面

组件和解决方案

包含

- 100+业务组件和公共组件支持
- iOS/Android跨平台统一组件
- CRN Web代码转换
- 业务开发技术支持

View组件

- AdView
- Button
- Carousel
- CRNListView
- CRNListViewDataSource
- DatePicker
- DatePickerWidget
- HeaderView
- HtmlText
- LinearGradient
- LoadControl
- LoadingFailedView
- LoadingNoDataView
- LoadingView
- MapView
- Page
- RefreshControl
- ScrollView
- SegmentedControl
- Slider
- SpriteImage
- SwipView
- IconfontView
-

API组件

- App
- ABTesting
- AddressBook
- Application
- Calendar
- Channel
- Device
- Encrypt
- Env
- Event
- Fetch
- ImagePicker
- Location
- Log
- Pay
- PhotoBrowser
- QRCode
- ScreenShot
- Share
- Storage
- Toast
- URL
- User
- Zip
-

CRN性能优化

- 页面加载流程
- Bundle拆分和框架预加载
- 业务代码按需加载
- 业务代码预加载
- 渐进式渲染页面

Demo

原始RN



CRN优化



RN打包文件分析

```
/* 1. 头部--全局定义部分 */
(function(global) {
  global.__DEV__=true;
  global.__BUNDLE_START_TIME__=Date.now();
})(typeof global !== 'undefined' ? global : typeof self !== 'undefined' ? self : this);

/* 2. 中间--各模块定义部分 */
__d(0 /* RNMessage/index.android.js */, function(global, require, module, exports) {
  /*...code...*/
  module.exports=require(12 /* ./src/index */);
}, "RNMessage/index.android.js");
__d(188 /* InitializeJavaScriptAppEngine */, function(global, require, module, exports) {
  /*...code...*/
  require(80 /* RCTDeviceEventEmitter */ );
}, "InitializeJavaScriptAppEngine");
__d(473 /* BorderRadius */, function(global, require, module, exports) {
  /*...code...*/
  module.exports = BorderRadius;
}, "BorderBox");
__d(474 /* resolveBoxStyle */, function(global, require, module, exports) {
  /*...code...*/
  module.exports = resolveBoxStyle;
}, "resolveBoxStyle");

/* 3. 尾部--引擎初始化+执行入口模块 */
;require(188);//InitializeJavaScriptAppEngine
;require(0);//入口模块
```

标准打包bundle文件结构



RN性能优化核心概念

/*源码*/

```
import page1 from './src/Page1.js';
```

```
import page2 from './src/Page2.js';
```

/*编译后*/

```
var _Page = require(662); // 662 = ./src/Page1.js
```

```
var _Page2 = _interopRequireDefault(_Page);
```

```
var _Page3 = require(663); // 663 = ./src/Page2.js
```

```
var _Page4 = _interopRequireDefault(_Page3);
```

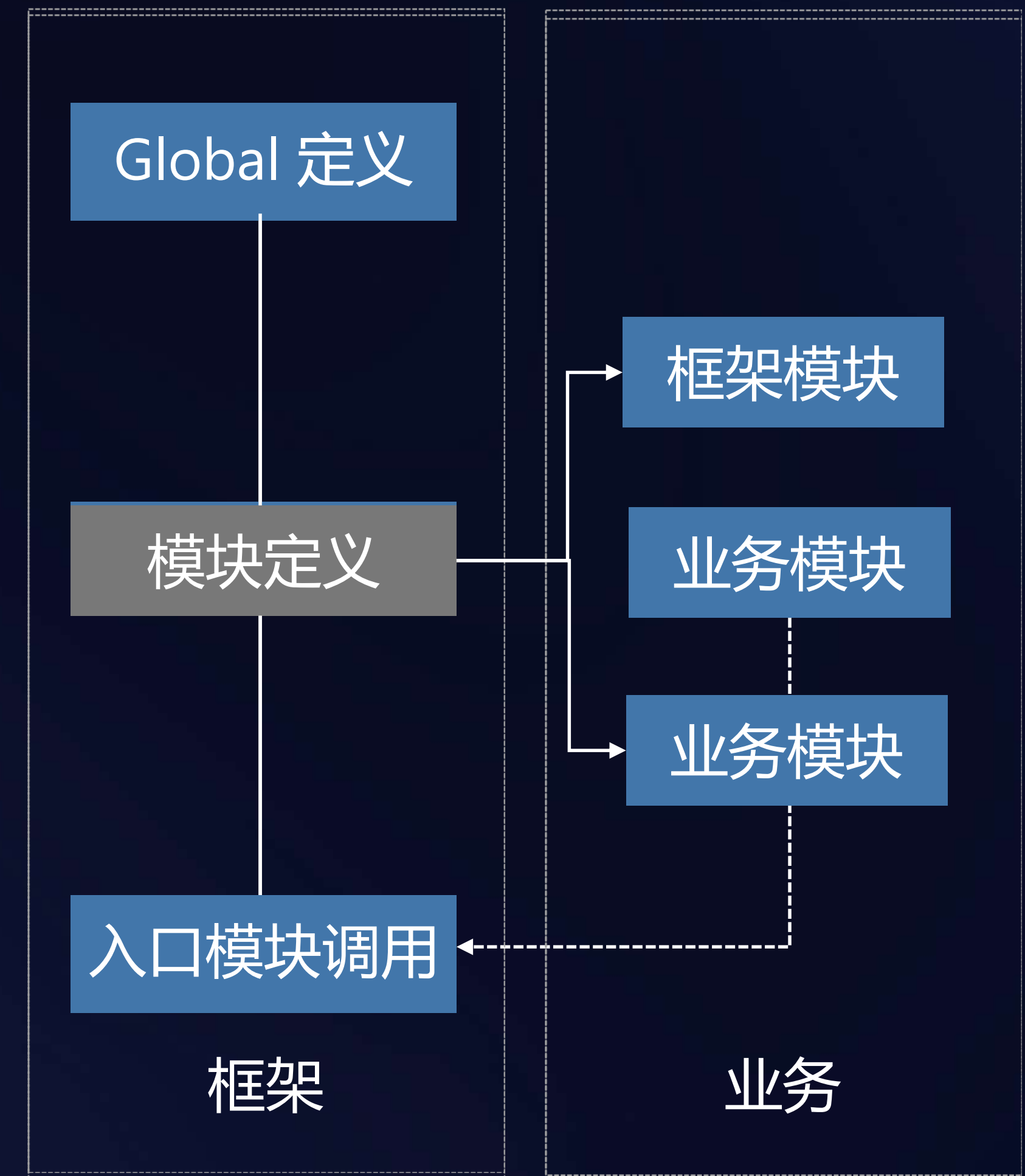
CRN页面加载全流程



- 灰色部分为可选部分，黄色部分为可优化点
- CRN框架加载 – 框架代码拆分和预加载
- 业务代码加载 – 业务代码懒加载
- 业务页面渲染 – 渐进式渲染

CRN框架代码拆分

- RN自带框架模块550+
- 框架入口模块可设计成空白页面
- 进入业务时，框架空白页面加载业务代码
- JS执行环境(Instance)和UI分离

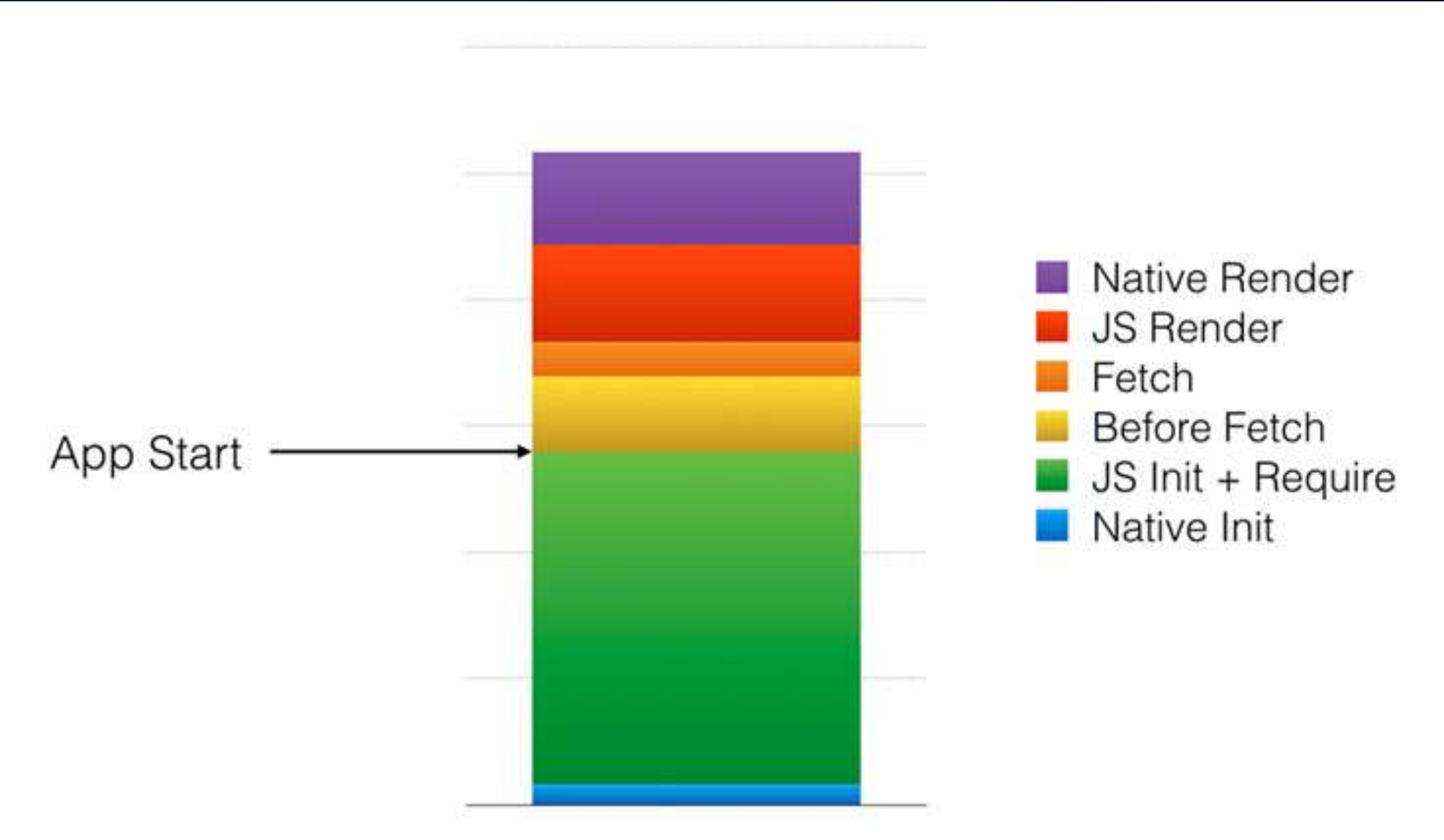


CRN框架预加载和缓存策略

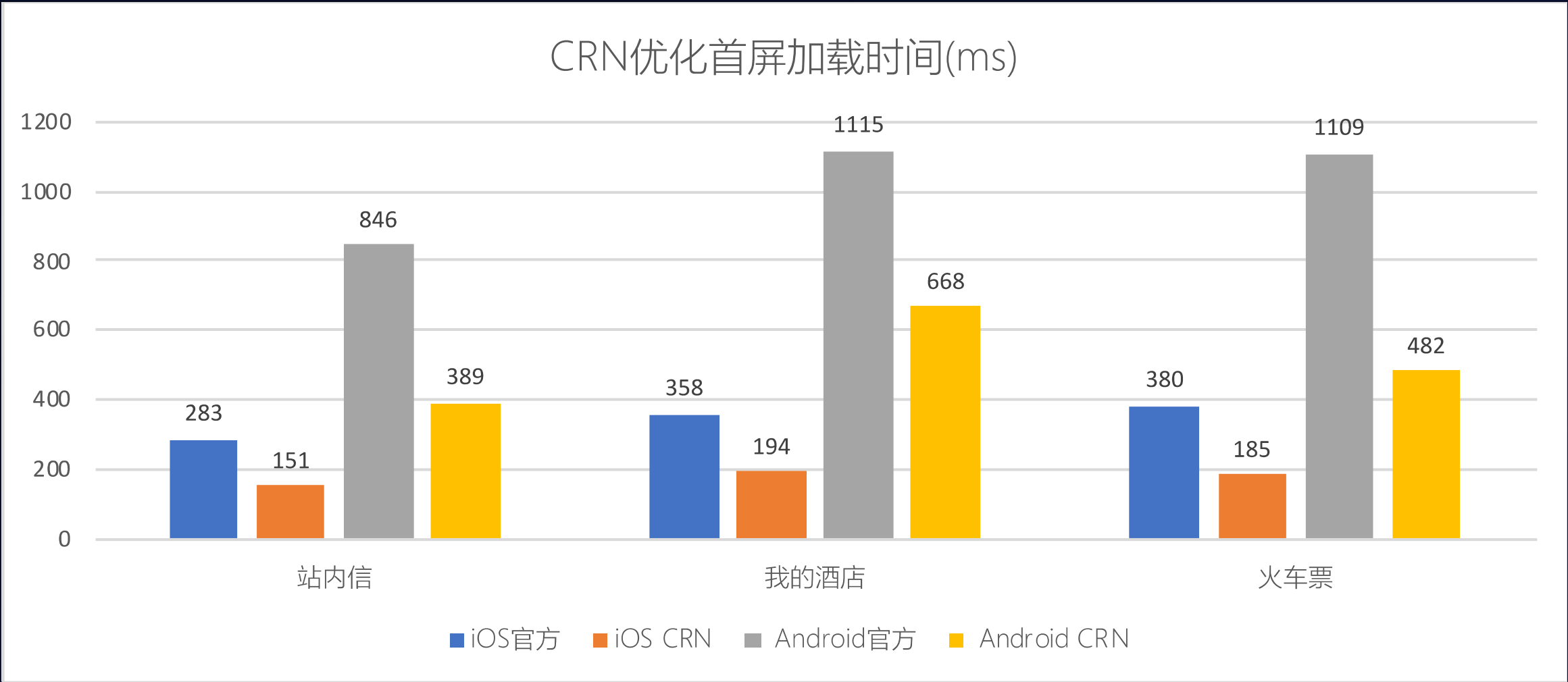


- 后台预创建好框架代码(Ready)的实例
- 缓存加载过的业务模块(Dirty)实例，加速再次加载
- 线上数据：进入业务时，Ready 80%, Dirty 15%

CRN框架预加载效果



RN业务加载的耗时分布



CRN预加后载业务加载的耗时

业务复杂度增加导致的首屏加载慢

CRN 业务页面路由配置示例

```
const PageA = lazyRequire("pages/PageA")
const PageB = lazyRequire("pages/PageB")
const PageC = lazyRequire("pages/PageC")
const PageD = lazyRequire("pages/PageD")
//设置页面路由表
let pageList = [PageA, PageB, PageC, PageD];
App.startApp(pageList);
```

随着业务复杂度增加，按需加载变慢

CRN按需加载方案

- 开发阶段
 - LazyRequire等价于require
 - 页面lazyRequire替换import
- 打包阶段
 - Babel插件转换路径为模块ID
- 运行阶段
 - Page切换时，自动load()执行

```
//lazyRequire初始化
LazyModule lazyRequire(modulePath);

//lazyRequire定义
LazyModule = {
  load(); //执行真正的模块代码，返回执行结果
};
```

Require耗时分析工具

CRN Profile Tool	
iPhone 8	刷新 清空日志
模块路径	耗时(ms)
- instanceld : 20181120202546444-3 moduleName : rn_ttd_act date : 20181120202546	448
+ rn_common 框架部分耗时	335
- rn_ttd_act 业务部分耗时	113
+ /index.ios.js	4
- /src/views/overseasindex/index.js	106
+ /src/views/overseasindex/redux/store.js	11
+ /src/views/overseasindex/containers/Home.js	94
/node_modules/@ctrip/crn/lib/MessageBox.js	1
/node_modules/regenerator-runtime/runtime.js	1
/node_modules/@ctrip/crn/lib/LinearGradient/index.ios.js	1
/node_modules/react-native/Libraries/Image/ImageBackground.js	0
/node_modules/react-native/Libraries/Animated/src/bezier.js	0

辅助发现加载存在性能瓶颈的模块

其他按需加载方案

- Getter API导出模块
- inlineRequire

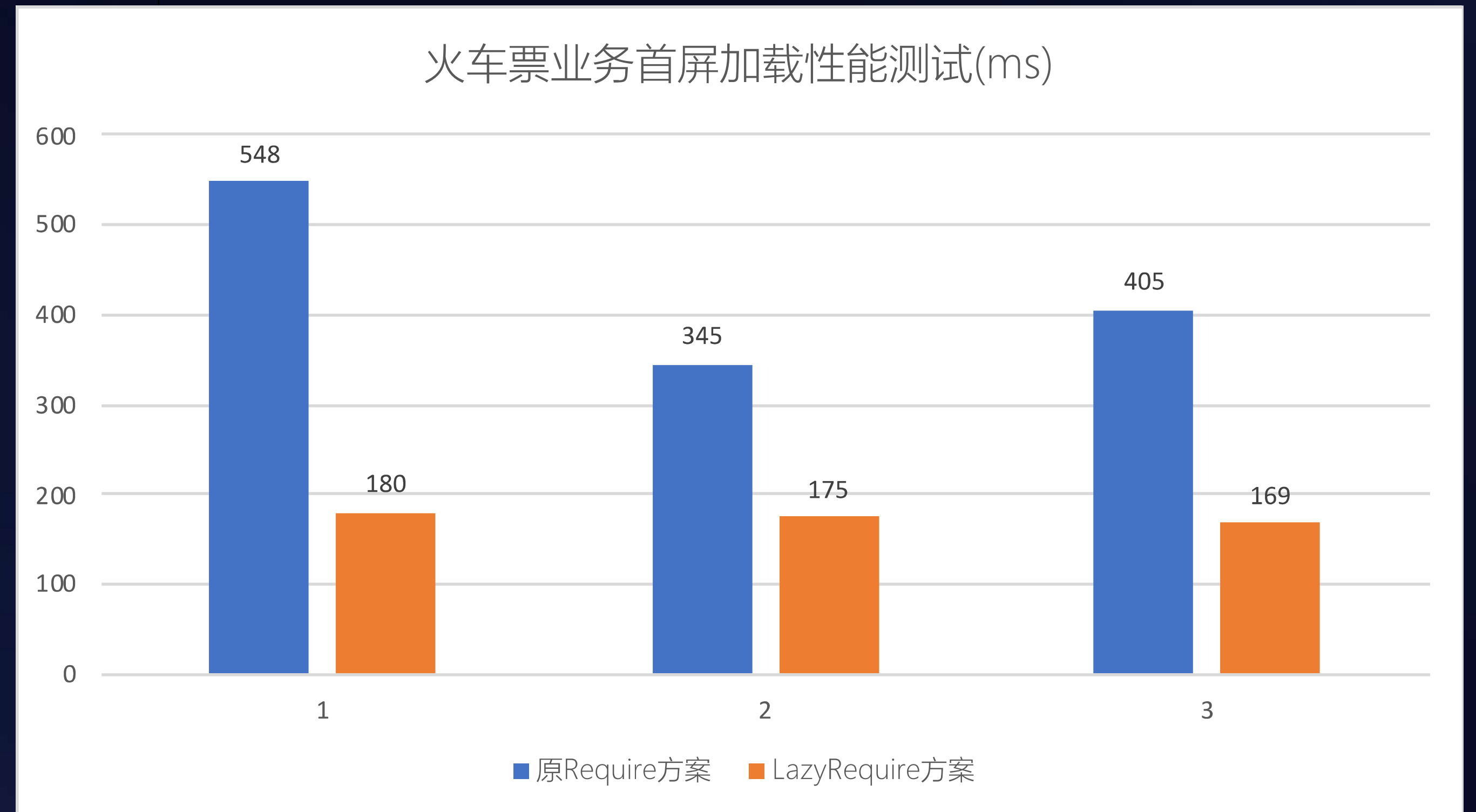
```
//import VeryExpensiveModule from './VeryExpensive';
import静态加载模块 ; },
let VeryExpensiveModule = null; ; },

export default class Optimized extends Component {
  someEvent = () => {
    if (VeryExpensiveModule == null) {
      VeryExpensiveModule = require('./VeryExpensive').default; id'); },
    }
    }; require('path').default, 动态执行模块代码 ]模块
  }
}
```

Lazy Require性能数据

数据

- 业务首屏加载性能提升明显
- 支持了复杂业务采用CRN开发



数据基于RN0.41 , iPhone Simulator多次测试 (2017年9月)

业务代码预加载

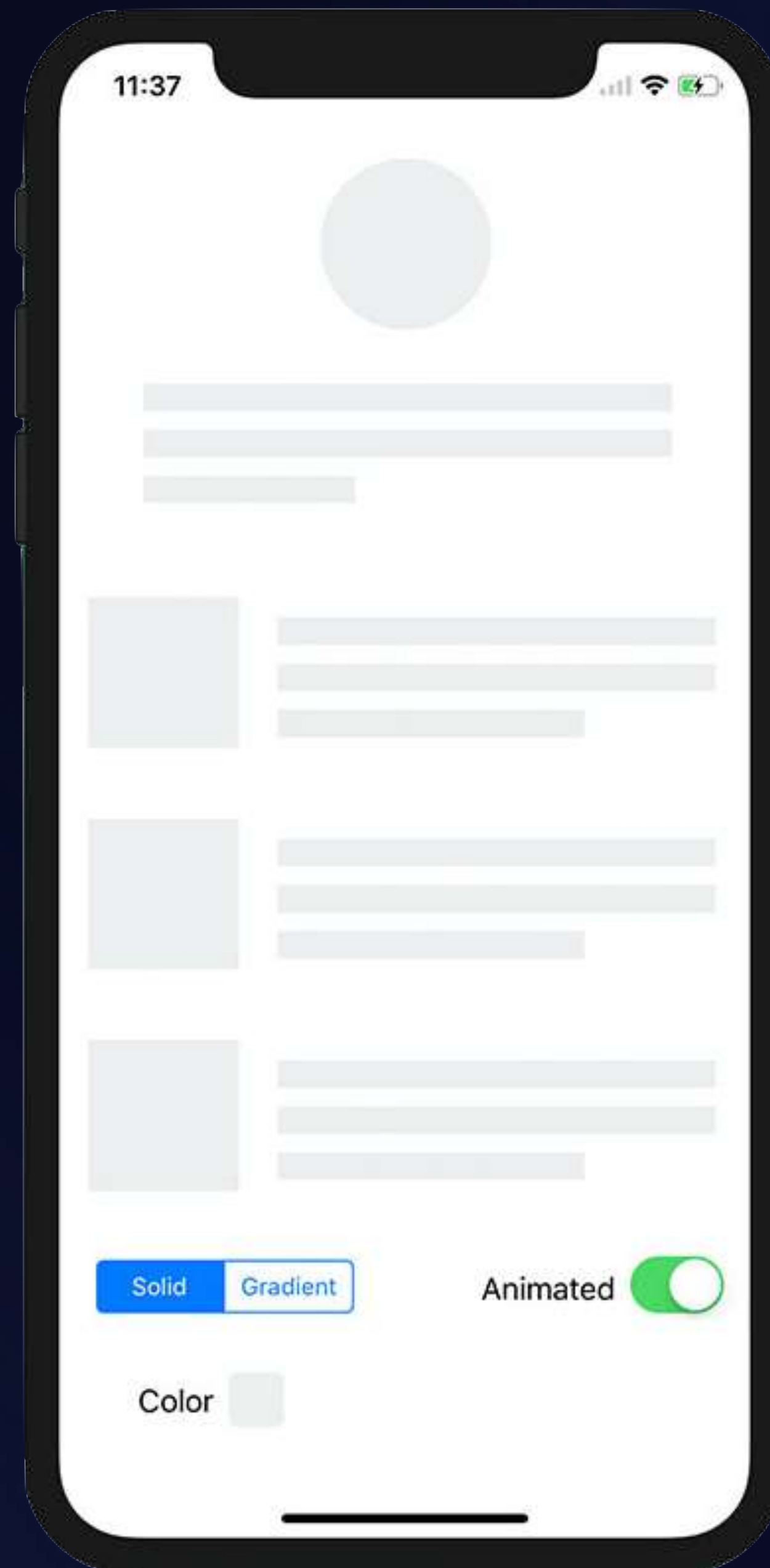
策略

- 尽可能多的预先执行业务代码
- 预加载模式下，lazy Require强制加载
- 缓存按照业务名而非URL区分，提高缓存利用率
- 内存占用问题(每个业务Android上2MB左右内存增加)

渐进式渲染方案

策略

- 适用于复杂页面场景
- 先渲染骨架图/头部部分
- 列表页面先渲染可视部分，滑动时按需渲染



发布与运维

- 发布系统
- 性能报表系统
- 异常上报系统

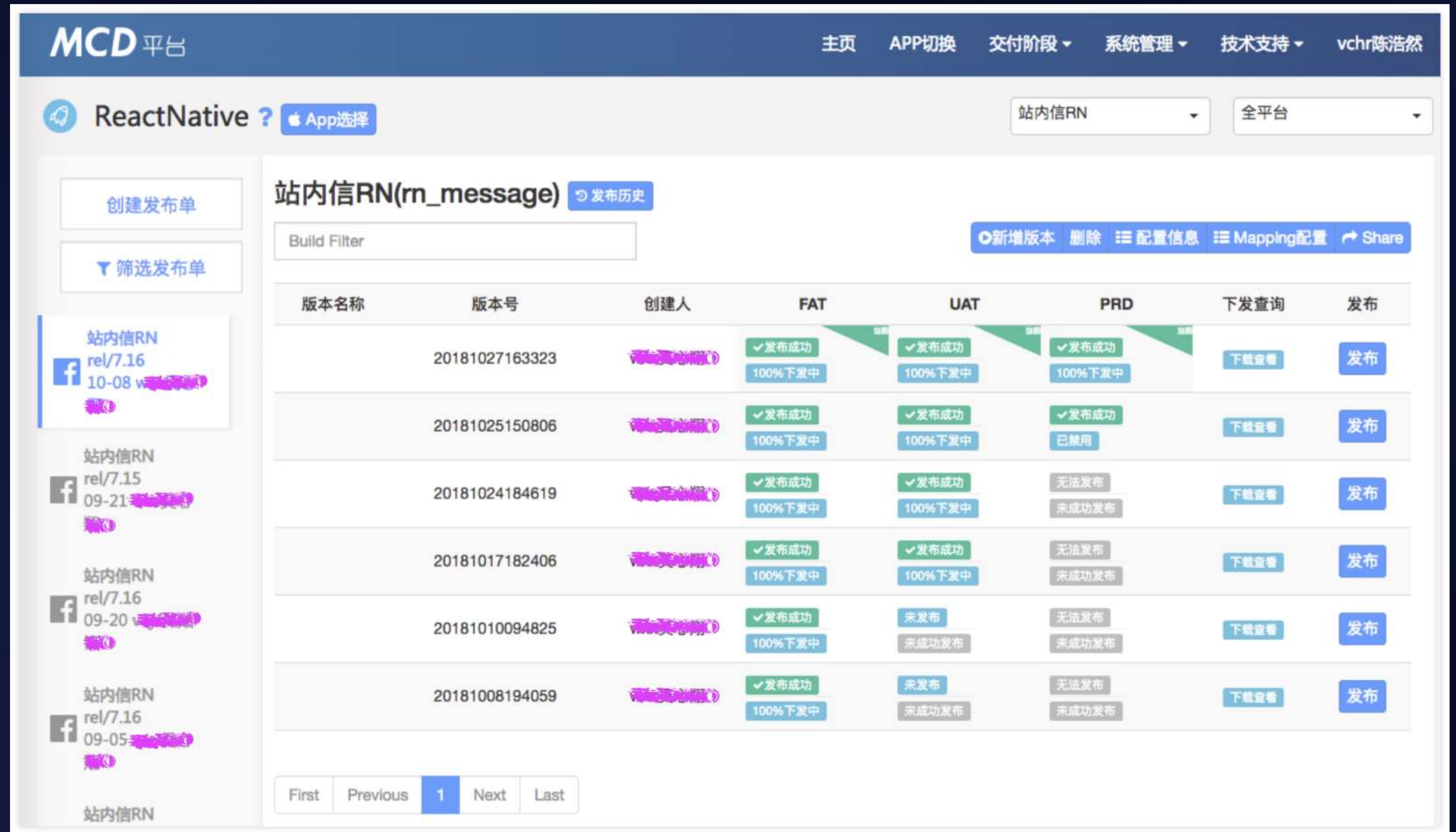
发布平台

功能

- 按版本/平台发布
- 灰度、回滚支持
- 发布结果查看
- 实时达到率

数据

- 平均每包大小15KB
- 实时到达率：首屏入口85%，二级及以上页面入口97%



The screenshot displays the MCD platform interface for managing ReactNative releases. The main section is titled '站内信RN(rn_message)' and includes a 'Build Filter' input and a '发布历史' (Release History) button. Below this is a table listing release records with columns for version name, version number, creator, and deployment status across different environments (FAT, UAT, PRD). Each record also has a '下发查询' (Download Query) button and a '发布' (Release) button.

版本名称	版本号	创建人	FAT	UAT	PRD	下发查询	发布
站内信RN rel/7.16 10-08 v[redacted]	20181027163323	[redacted]	✓发布成功 100%下发中	✓发布成功 100%下发中	✓发布成功 100%下发中	下载查看	发布
站内信RN rel/7.15 09-21 v[redacted]	20181025150806	[redacted]	✓发布成功 100%下发中	✓发布成功 100%下发中	✓发布成功 已禁用	下载查看	发布
站内信RN rel/7.16 09-20 v[redacted]	20181024184619	[redacted]	✓发布成功 100%下发中	✓发布成功 100%下发中	无法发布 未成功发布	下载查看	发布
站内信RN rel/7.16 09-05 v[redacted]	20181017182406	[redacted]	✓发布成功 100%下发中	✓发布成功 100%下发中	无法发布 未成功发布	下载查看	发布
站内信RN rel/7.16 09-05 v[redacted]	20181010094825	[redacted]	✓发布成功 100%下发中	未发布 未成功发布	无法发布 未成功发布	下载查看	发布
站内信RN rel/7.16 09-05 v[redacted]	20181008194059	[redacted]	✓发布成功 100%下发中	未发布 未成功发布	无法发布 未成功发布	下载查看	发布

Navigation controls at the bottom include 'First', 'Previous', '1' (selected), 'Next', and 'Last'.

性能报表系统

功能

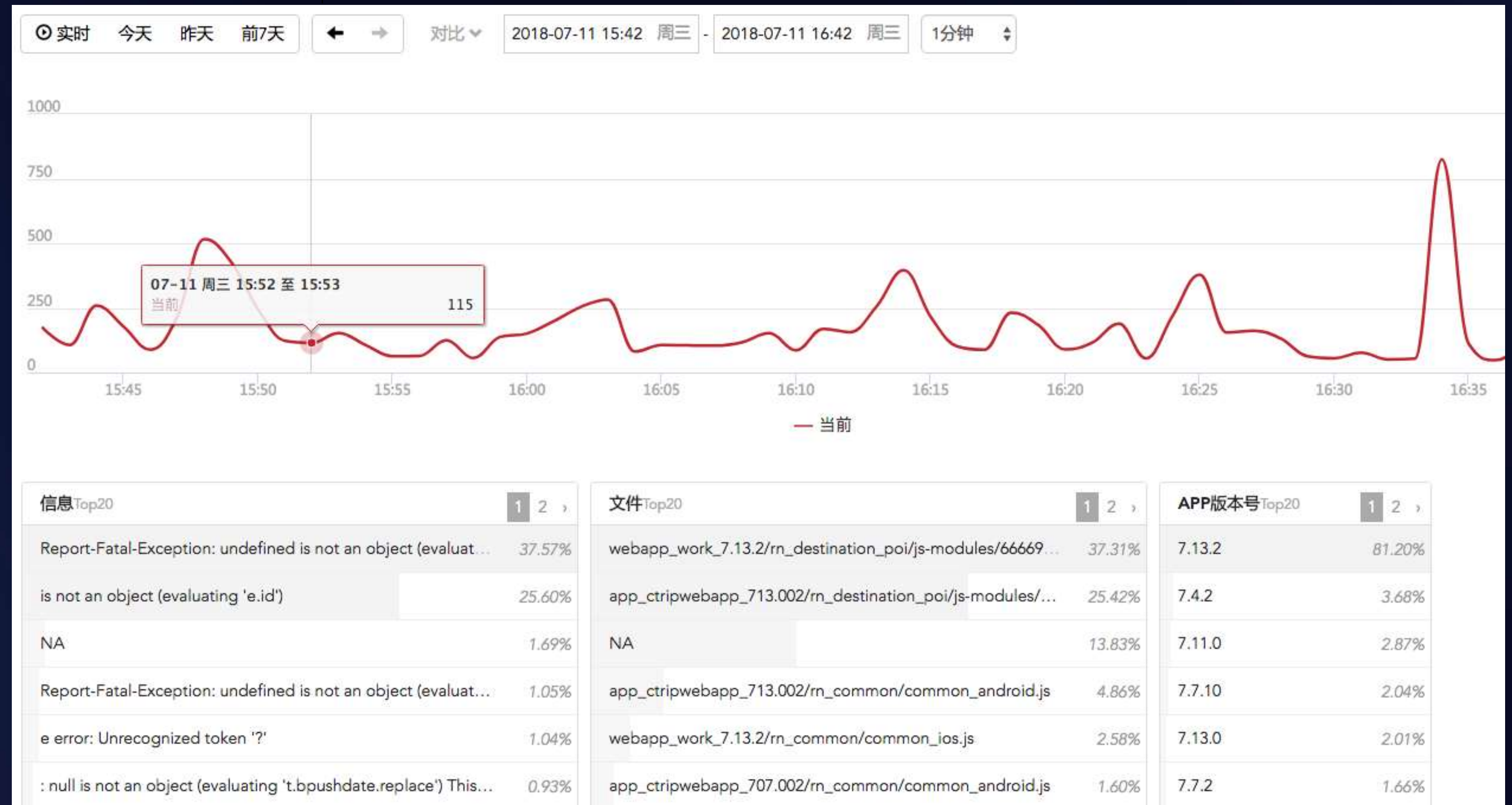
- 线上框架首屏渲染时间
- 多维度过滤筛选
- 耗时分布



异常上报系统

功能

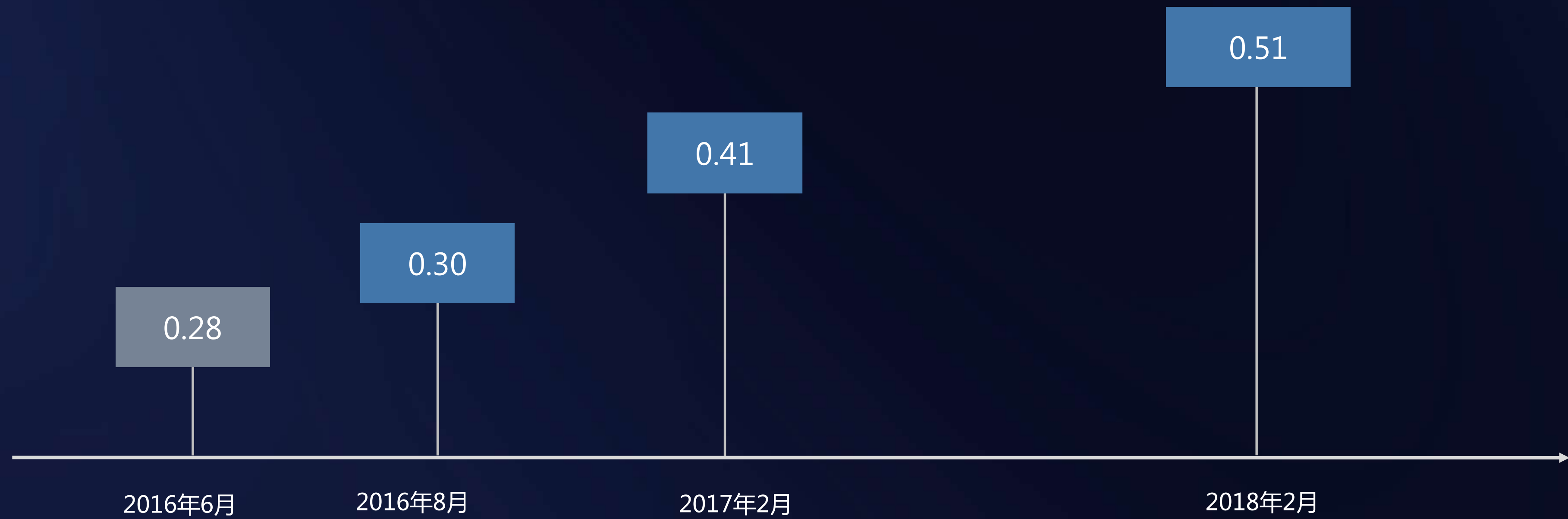
- 实时上报Native/JS异常
- 多维度过滤筛选
- 支持报警配置
- 发布必备辅助系统



实践经验

- 版本升级
- 版本和依赖管理
- 分平台打包
- Android稳定性优化

CRN版本升级



CRN版本升级历程

CRN版本升级

- 升级流程



- 升级成本

- 框架团队成本2-3周
- 业务升级一周完成，主要在发布和回归测试

- 其它

- 升级方案尽可能对业务简单
- 避免业务包做跨RN版本的发布
- 升级频率8-12个月/次

版本和依赖管理

- 依赖规则

- 业务包只依赖框架包
- 业务包之间平行，不允许依赖

- 版本管理

- 固定版本，避免使用^、*
- 构建过程检查依赖的版本

```
"dependencies": {  
  "@ctrip/crn": "git+http://[redacted]/crn#rel/7.7",  
  "react": "15.4.2",  
  "react-native": "0.41.0",  
  "react-native-swiper": "^1.5.4"  
},
```

react-native-swiper

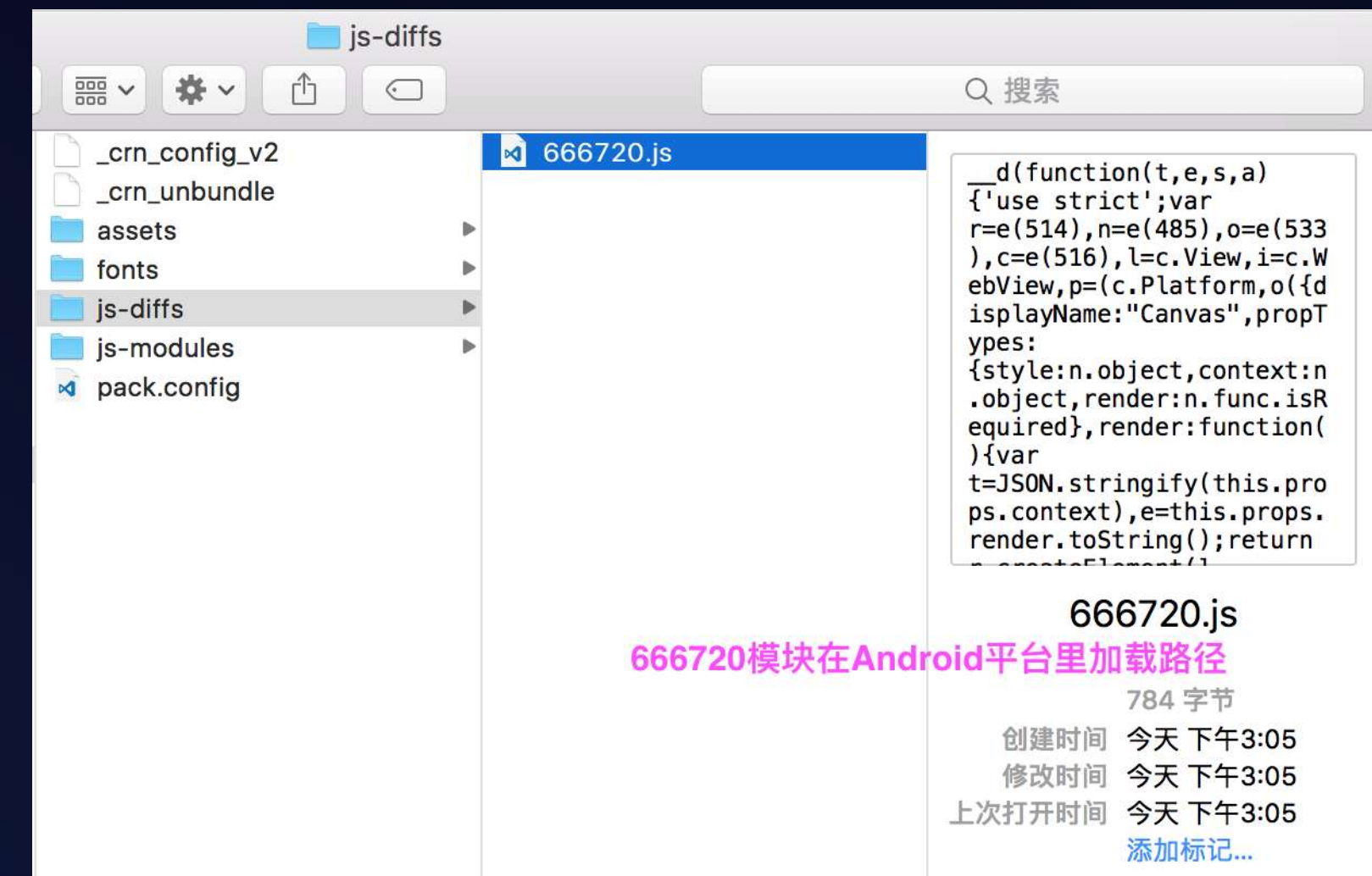
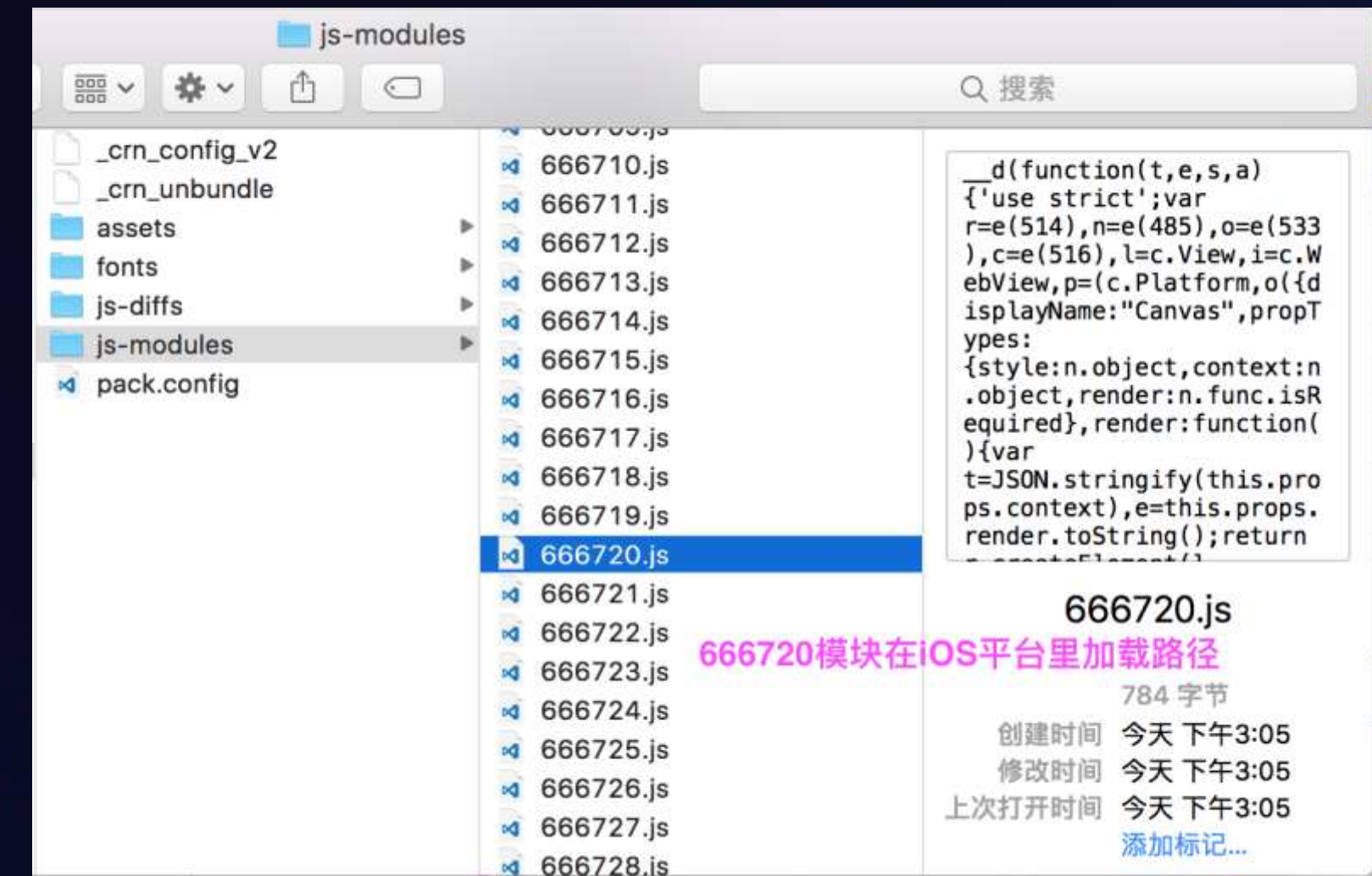
[Readme](#)[1 Dependencies](#)[54 Dependents](#)

Version History

1.5.9	10 months ago
1.5.8	10 months ago
1.5.7	10 months ago
1.5.5	10 months ago
1.5.4	2 years ago
1.5.3	2 years ago
1.5.2	2 years ago

分平台打包

- 原方案
 - CR业务包共用，以iOS平台打包
 - 框架包分开打包
- 解决方案
 - 平台独立组件，打包产物不一致
 - 分2次打包，打包之后merge打包产物
 - Android优先在js-diffs中查找模块



Android稳定性优化

- 常规异常处理
 - 跟进crash堆栈逐一处理
 - so加载失败，try catch重试补偿
- JSC导致的Crash
 - 堆栈只有libjsc.so
 - 升级jsc版本，性能无差异，稳定性提升明显
- HashMap导致的问题
 - 多instance并行创建场景下偶现
 - 非线程安全导致的死锁

总结

- RN是一项适合业务大规模使用的跨平台开发框架
- RN落地需要考虑性能稳定性优化，以及配套系统建设
- 携程核心业务将会全面转向CRN开发

本PPT来自2018携程技术峰会
更多技术干货，请关注“携程技术中心”微信公众

