2016 Velocity 有感

- Present By Cloughzhang
Dec. 12th, 2016

“如果您对开发运维和Web性能是认真的，Velocity 是必须参加的会议。“

09:30 主题发言 我们足够好了吗? 重新思考服务质量 Theo Schlossnagle (OmniTI/Circonus)		
10:00 主题发言 扩展交互式数据可视化工作 Jeff Catania (Accenture)		
10:30 上午茶歇		
10:50 主题发言 通过性能管理挖掘产品生命周期潜藏的商业价值 唐文 (高升控股)		
10:55 大规模系统平衡性能最佳实践和弹性工程 Betty Tso (Amazon)		
11:30 主题发言 #UseThePlatform——Web组件介绍 Mikhail Sychev (Google/YouTube)		
12:00 自助午餐		
13:30 搜索引擎性能优化的未来——搜索极速浏览框架 陶清乾 (百度)	**13:30** 打造SRE(运维)和开发团队的健康关系 李虓 (LinkedIn)	**13:30** 零点之战——阿里双11技术架构演进之路 丁宇 (阿里巴巴集团)
14:30 应用性能数据可视化 朱建锋 (mmTrix)	**14:30** 数据库可靠性工程 Laine Campbell (OrderWithMe)	**14:30** OceanBase：蚂蚁双十一背后的关系数据库 杨传辉 (蚂蚁金服集团)
15:30 下午茶歇		
15:50 滴滴弹性在线存储平台 周充 (滴滴)	**15:50** Hulu的React/Redux架构实践 程墨 (Hulu)	**15:50** 菜鸟物流云混合云基础架构 黄浩 (菜鸟网络科技有限公司)
16:50 58到家微服务架构实践 沈剑 (58到家)	**16:50** 数亿级用户规模下的React native工程实践 雷志兴 (百度)	**16:50** 双11苏宁易购天猫店的技术应对及其演进 周毅 (苏宁)
17:50 Walle——企业级应用开发模式的探索与创新 沙彦魁 (菜鸟网络科技有限公司)	**17:50** DT时代的业务实时监控之道 杨奕 (阿里巴巴集团)	**17:50** 双十一Weex会场极致性能优化 冯成晓 (阿里巴巴集团), 周婷婷 (阿里巴巴集团)

08:00 来宾登记

大宴会厅1	大宴会厅2	大宴会厅3
09:30 主题发言 构建下一代移动网页应用 谷盛 (Google)		
10:00 主题发言 阿里应用运维体系演变 林昊（毕玄） (阿里巴巴)		
10:30 上午茶歇		
10:50 主题发言 测量服务的可运维性(Measure the operability of your service) 李虓 (LinkedIn)		
11:30 主题发言 有状态服务的数据完整性 Laine Campbell (OrderWithMe)		
12:00 自助午餐		
13:30 高性能MySQL 叶金荣 (知数堂培训)	**13:30** Polymer在YouTube的应用 Mikhail Sychev (Google/YouTube)	**13:30** Swarm优化：从单实例管理1000 nodes到30000 nodes 吴小伟 (阿里巴巴集团)
14:30 HBase在滴滴出行各业务场景应用 朱怀宇 (滴滴)	**14:30** React可视化开发框架 Jeff Catania (Accenture)	**14:30** OWL分布式开源监控最佳实践 吴迎松 (TalkingData)
15:30 下午茶歇		
15:50 天猫双11互动直播间性能优化 刘雄昌 (阿里巴巴集团)	**15:50** 今天谈构建可扩展系统的意义 Theo Schlossnagle (OmniTI/Circonus)	**15:50** 网易蜂巢基于kubernetes的公有云运维实践 刘超 (网易蜂巢)
16:50 大型分布式系统的devops实战 何学奇 (京东)	**16:50** QQ空间亿级服务Web架构 刁维康 (腾讯科技)	**16:50** 阿里巴巴Aliware十年微服务架构演进历程中的挑战与实践 倪超 (阿里巴巴集团)
17:50 海量日志驱动的智能运维 饶琛琳 (日志易)		

与前端相关

- 趋势
  - PWA(G), Fire TV(A)
- 框架
  - react+redux(Hulu), polymer(G), React+D3[RVDF](埃森哲)
- 性能优化
  - 百度搜索全流程[web](B)，手空[Hybrid](T)，手百[RN](B),
- 数据可视化
  - Rethinking QoS(OmniTI/Circonus)，mmTrix

主观感受

- 有干货、有广告、有浑水摸鱼
- 国外趋势和框架、国内解决方案
- 框架不好讲、优化踩坑能引起共鸣

趋势

构建下一代移动网页应用(PWA)

- **Reliable** - Load instantly and never show the downasaur, even in uncertain network conditions.
- **Fast** - Respond quickly to user interactions with silky smooth animations and no janky scrolling.
- **Engaging** - Feel like a natural app on the device, with an immersive user experience.

- service worker(离线访问)
- home screen - add home site
- immersive full screen
- push notifications (GCM)
- Hardware API
- APP Shell

https://www.youtube.com/watch?v=MxTaDhwJDLg&feature=youtu.be

- https://75team.com/post/cds.html
- http://www.leiphone.com/news/201606/UEiart497WUzS62u.html
- https://aliexpress.com
- https://housing.com

个人见解

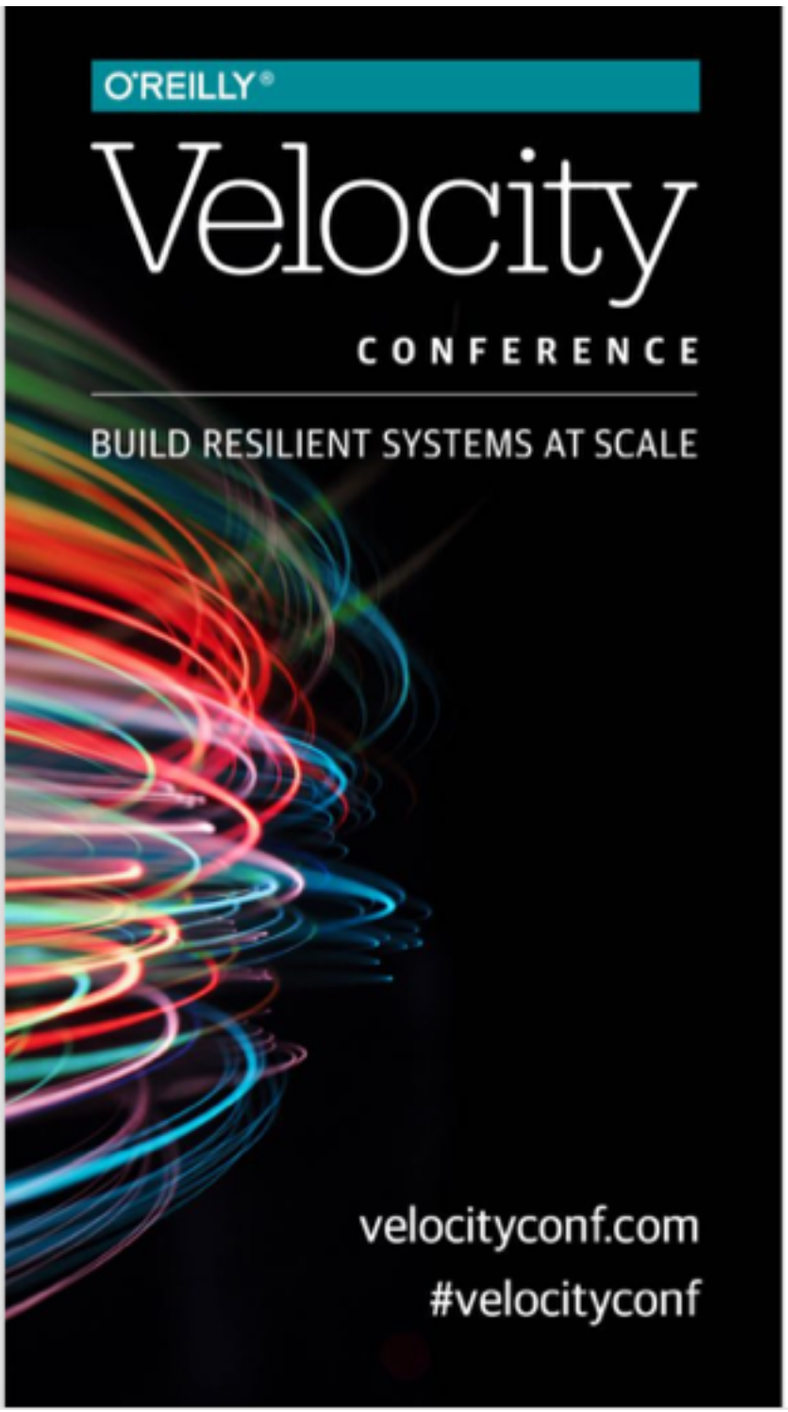
- 浏览器兼容性
- 用户习惯 (之于微信小程序如何？)
- 学习门槛

大规模系统平衡性能最佳实践和弹性工程 (Fire TV)

- 一套代码多端使用（accesibility）

- delaunay triangulation

框架系列



React & Redux in Hulu

程墨 (Morgan Cheng)

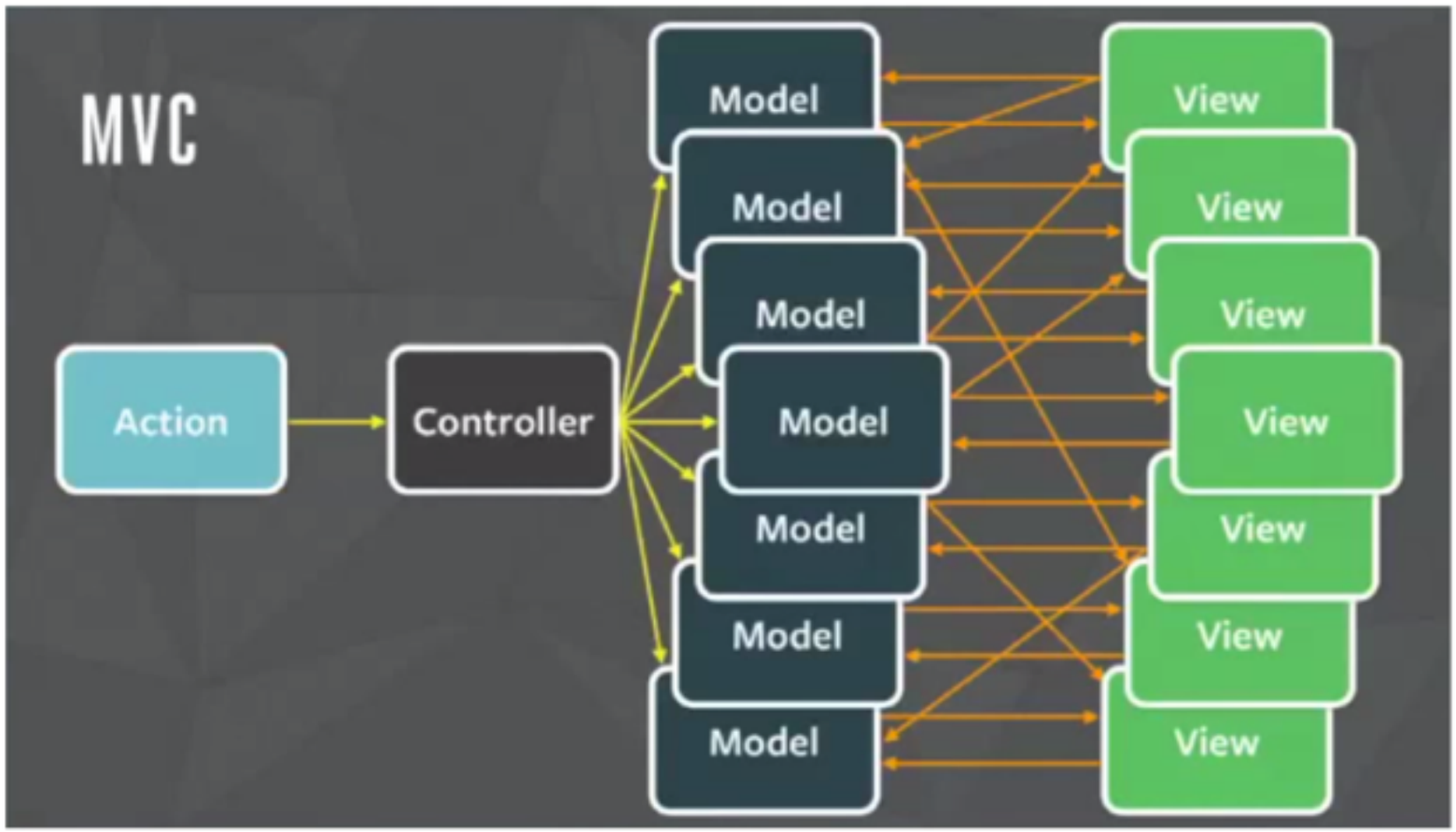
Code in jQuery

```
     */
    elements: {
        // 获奖详情
        $awardDetailTitle: '.js_awardDetailTitle',
        $awardDetail: '.js_awardDetail',
        $awardArrow: '.js_awardDetailArrow',

        $lotteryRuleDetail: '.js_lotteryRuleDetail',
        $lotteryRuleDetailArrow: '.js_lotteryRuleDetailArrow',

        $showButton: '.js_showButton',
        $lotteryButton: '.js_lotteryButton',
        $address: '.js_address',
        $emptyAddress: '.js_emptyAddress',
        $overlay: '.js_overlay',

        $awardIntro: '.js_awardIntro',
        $congrtImg: '.js_congrtImg',

        $award: '.js_award',
        // 奖品列表 3个圈
        $awardList: '.js_awardList',
        $awardListFront: '.js_awardListFront',
        $awardBack: '.js_awardBack',
        // 抽奖后 展示中奖信息详情
        $awardInfo: '.js_awardInfo',

        // 提交框中的地址信息
        $subRecvName: '.js_address .js_subRecvName',
        $subRecvPhone: '.js_address .js_subRecvPhone',
        $subRecvAddress: '.js_address .js_subRecvAddress',

        // 中奖信息显示框
        $awardName: '.js_awardName',
        $awardRecvName: '.js_awardRecvName',
        $awardRecvPhone: '.js_awardRecvPhone',
        $awardRecvAddress: '.js_awardRecvAddress',
        $awardRecvShippingCorpName: '.js_awardRecvShippingCorpName',
        $awardRecvShippingNo: '.js_awardRecvShippingNo',
        $awardRecvComment: '.js_awardRecvComment',

        // 礼品展示框
        $awardInfoTitle: '.js_awardInfoTitle',
        $awardInfoName: '.js_awardInfoName',
        $awardInfoUrl: '.js_awardInfoUrl',
        $awardInfoBack: '.js_awardInfoBack',

        // 礼品展示框
        $awardShowTitle: '.js_awardShowTitle',
        $awardShowUrl: '.js_awardShowUrl',

        // 弹出框
        $confirmPanel: '.js_confirmPanel'
    },
```

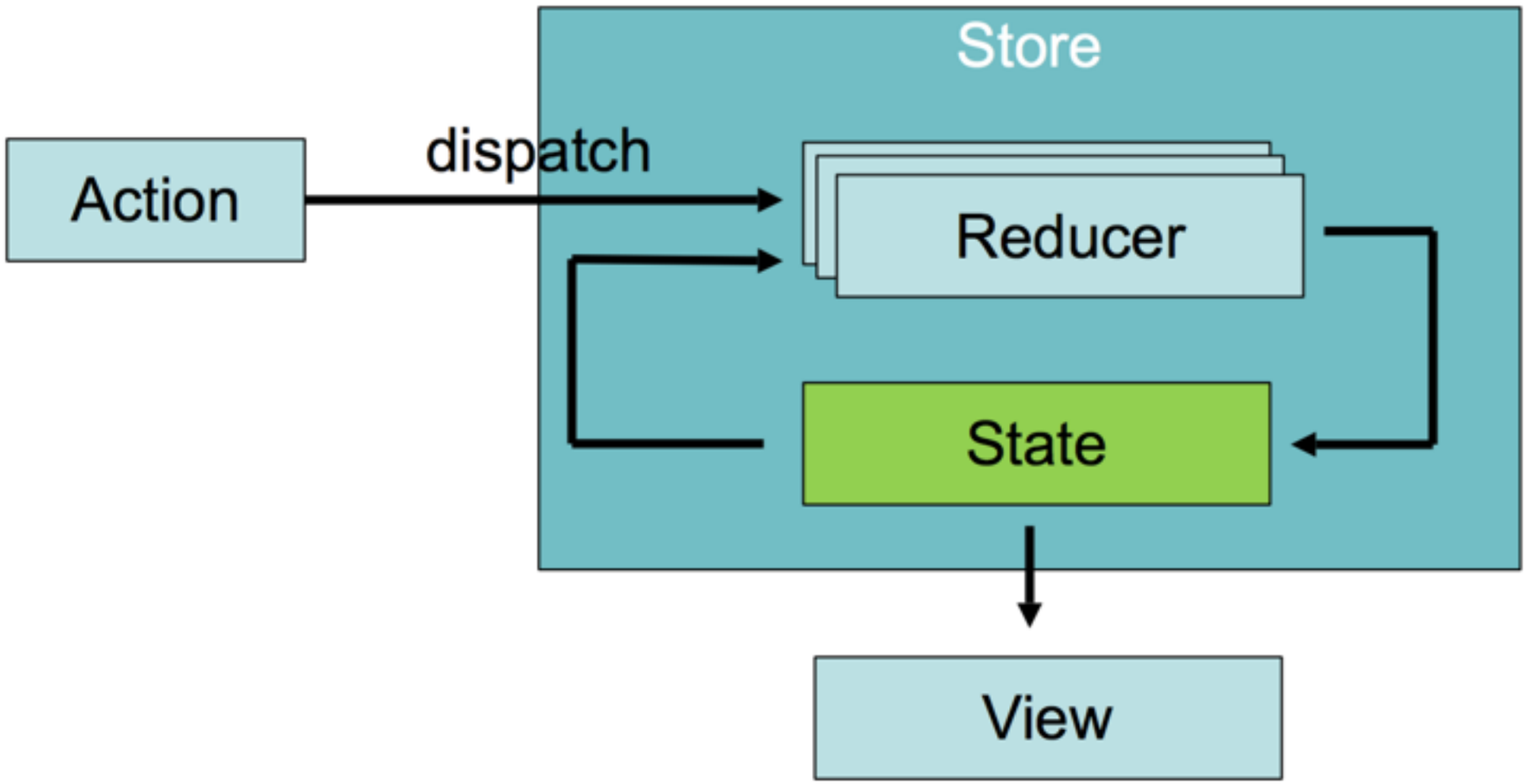
UI = f(state)

React Components are bricks, not building

Problem of MVC



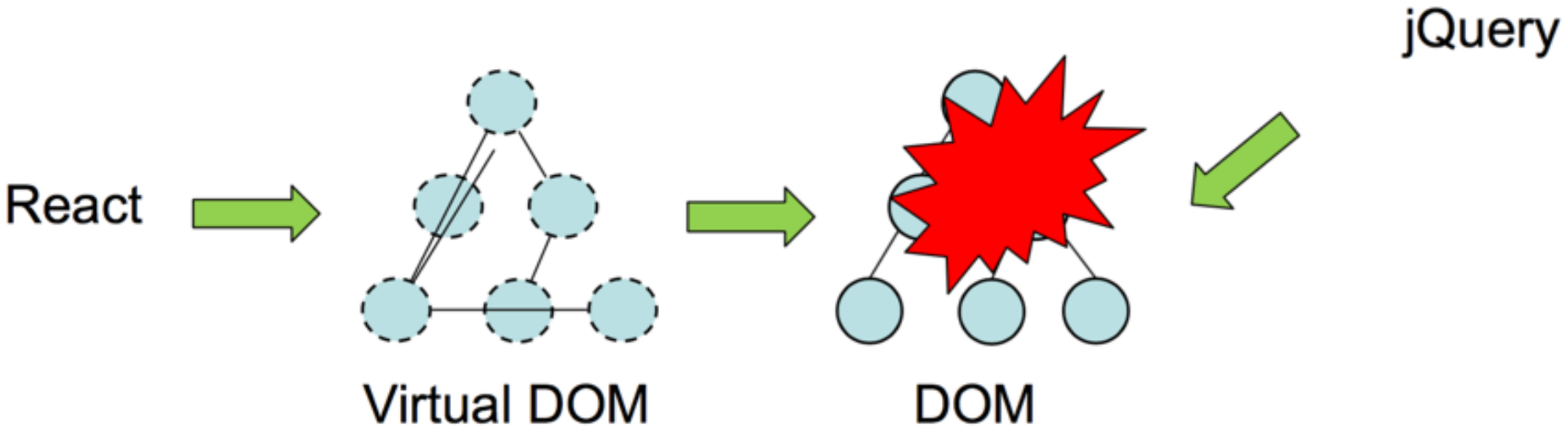
Redux



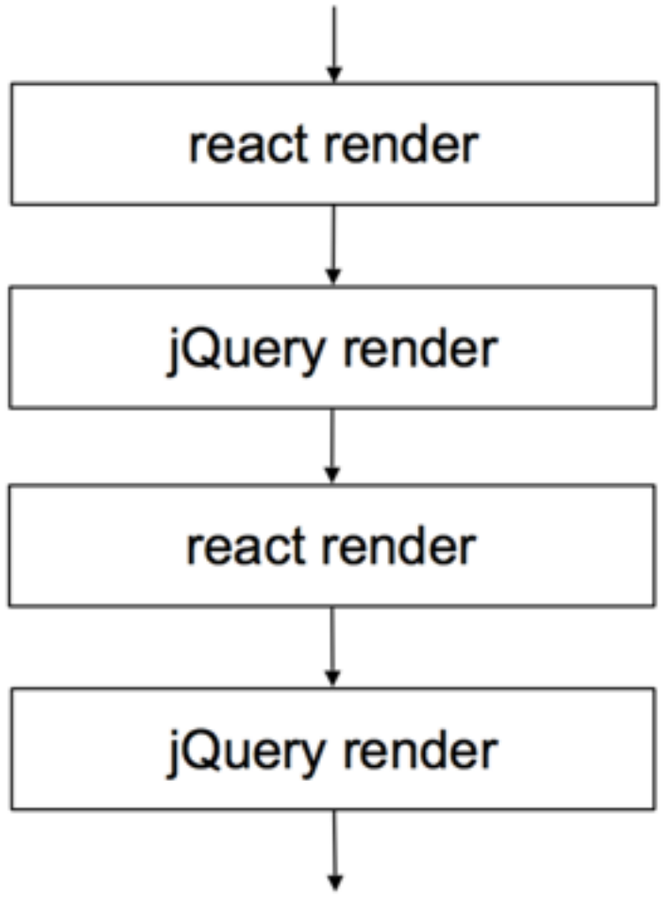
Constraints over Conventions

- jQuery Dies Hard
  - Legacy Code
  - Third Party JavaScript Code
    - e.g. Optimizely

React conflicts with jQuery



```
class Foo extends React.Component{

  componentDidMount() {
    // do DOM manipulation with jQuery
  }

  componentDidUpdate() {
    // do DOM manipulation with jQuery
  }

  render() {
    // render JSX for react
  }
}
```

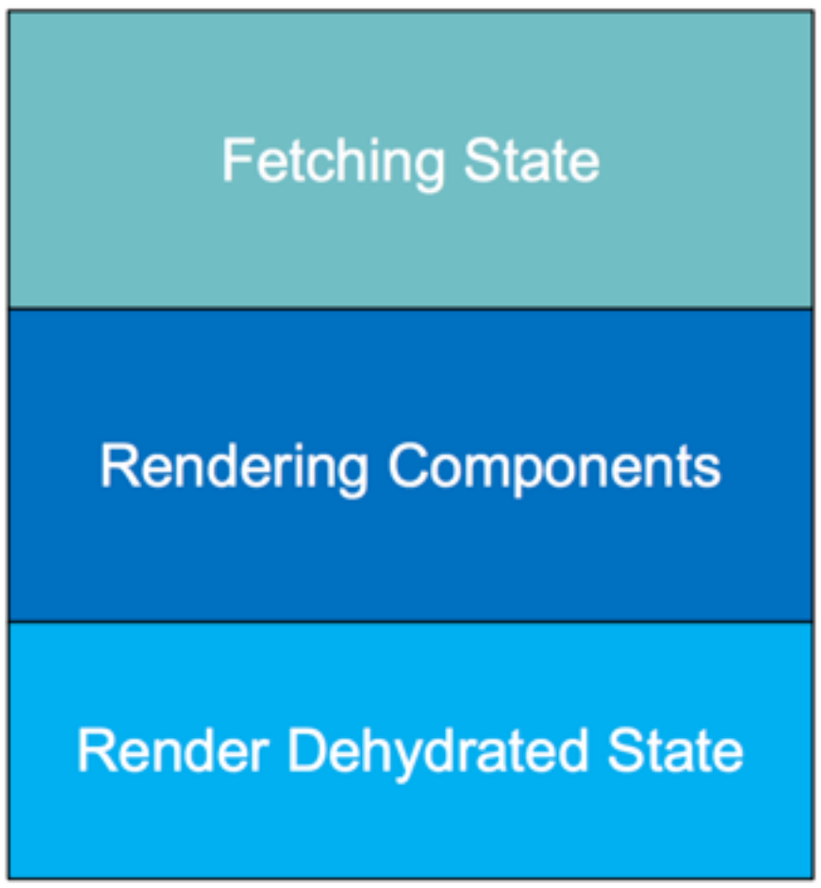


- Set Production Mode
  - **NODE_ENV=production** node app.js

- ~~Use Minified React.js~~
  - require('react/dist/react.min.js');
  - After v0.14, DO_NOT_USE_OR_YOU_WILL_BE_FIRED

- Babel-react-optimize Preset
  - https://github.com/thejameskyle/babel-react-optimize

Server Rendering Process

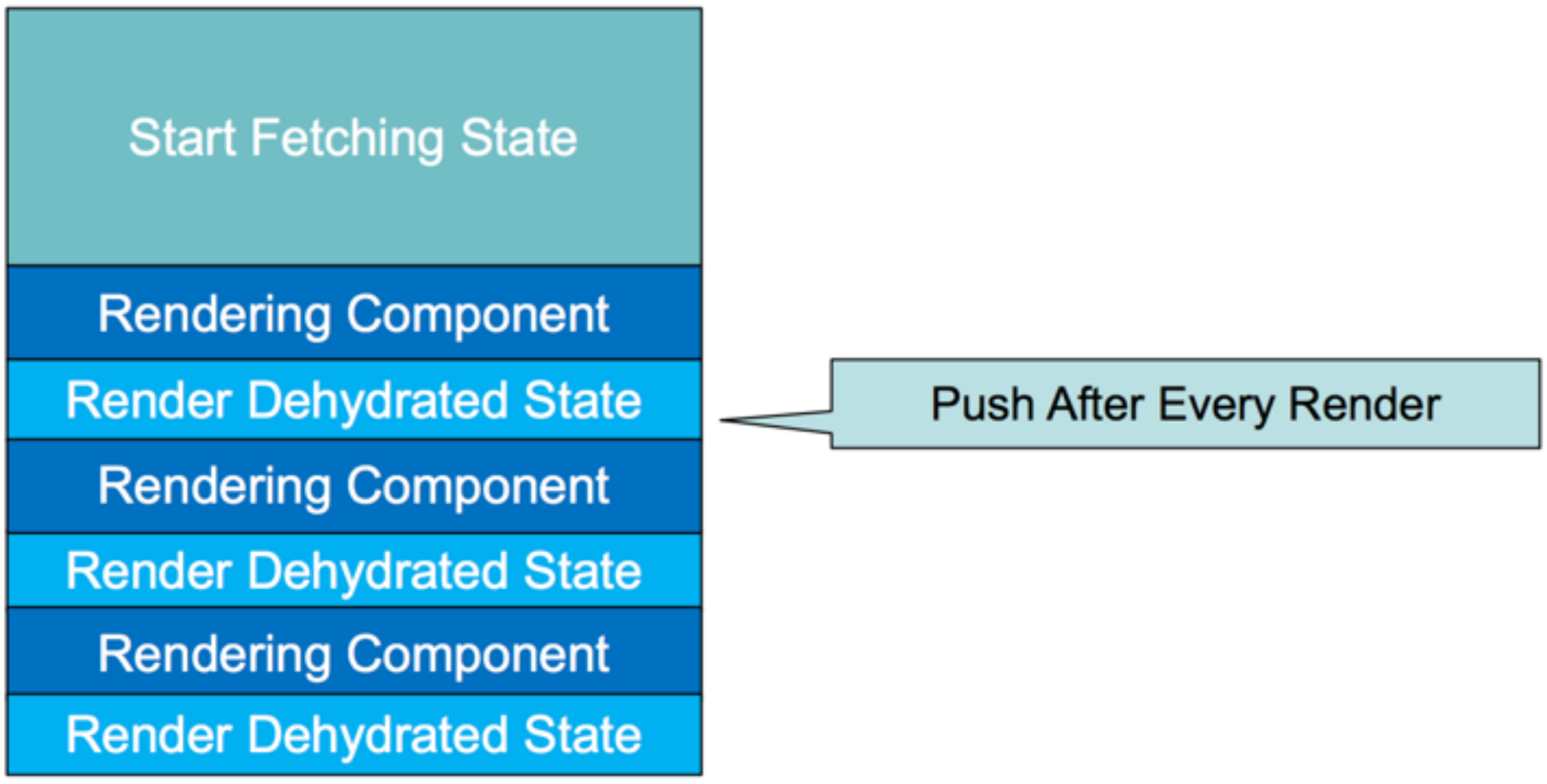



```
fetchState().then(render)
```

```
ReactDOMServer.renderToString(...)
```

```
<script>
   window.initState = ...
</script>
```

Server Rendering Process Improved



Polymer

- HTML Template
- Custom Element
- HTML Import
- shadow DOM
- Web Components

数据可视化

O'REILLY®

Velocity

CONFERENCE

BUILD RESILIENT SYSTEMS AT SCALE

Rethinking Quality of Service

Percentages are not People

大胡子 @postwait

CIRCONUS

velocityconf.com

#velocityconf

When you have a lot of data, what question should you ask?

- Assume 10,000 measurements over a minute...
  Should you consider:
  - The average?
  - The variance?
  - The median?
  - Minimum? Maximum?
  - 95th Percentile? 99th? 99.9th? 75th? 25th? 99.5th? …

- Stop… why?

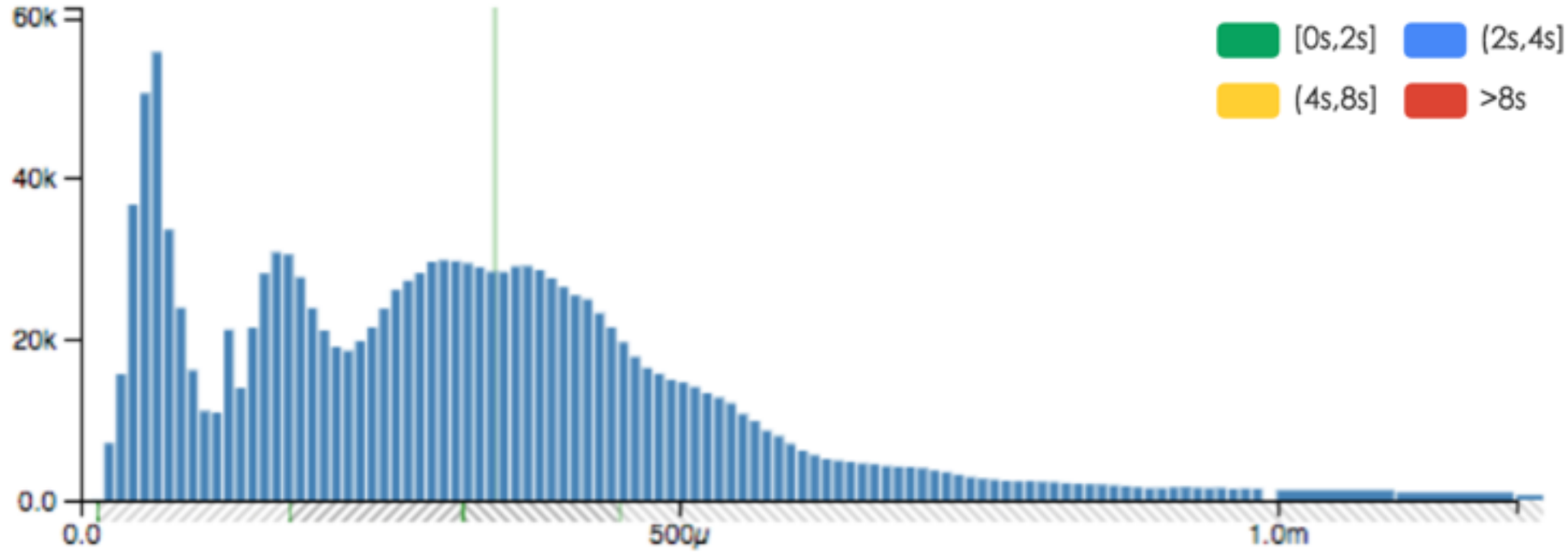
Why do we measure?

- We measure to understand improvement (and degradation)
  - Did we release bad code?
  - Did we fix a latency issue?
  - Are things slower today than yesterday?

- We measure to discern success
  - Are we fast enough?
  - Are our users happy?

What does observed latency actually look like?

Latency of get `latency

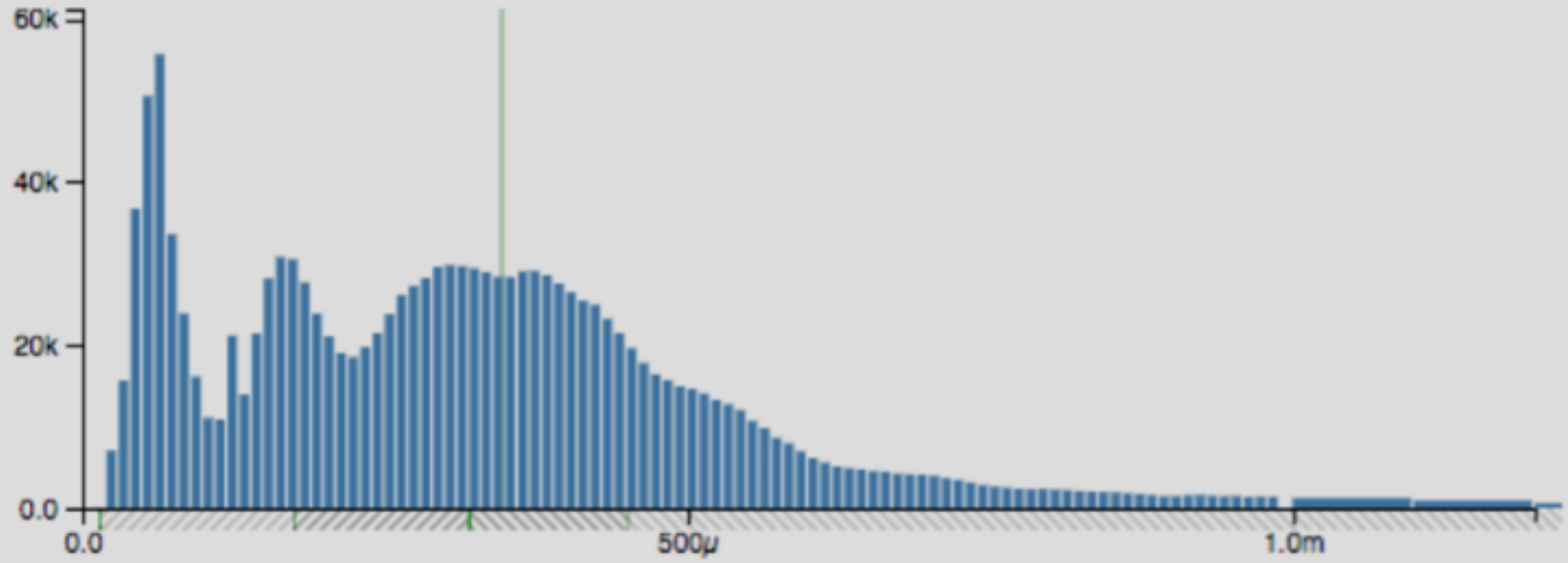


Introducing an inverse quantile

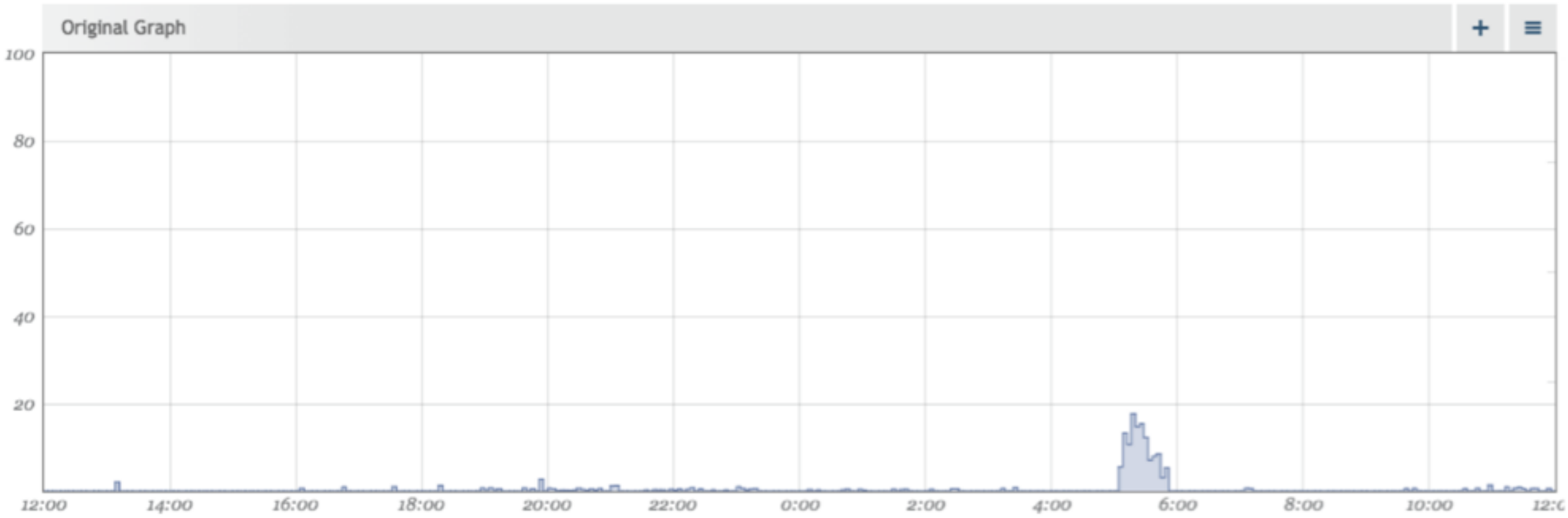
$$q(N, v) = r$$

$$q^{-1}(N, r) = v$$

Latency of get `latency



Percentage of violations: (1 - q^{-1}(1500ms))*100



mmtrix

- 几何分布或中位数代替算数平均值

性能优化

O'REILLY®

Velocity

CONFERENCE

BUILD RESILIENT SYSTEMS AT SCALE

搜索极速浏览框架探索与实践

陶清乾

velocityconf.com

#velocityconf

目录

速度之于搜索

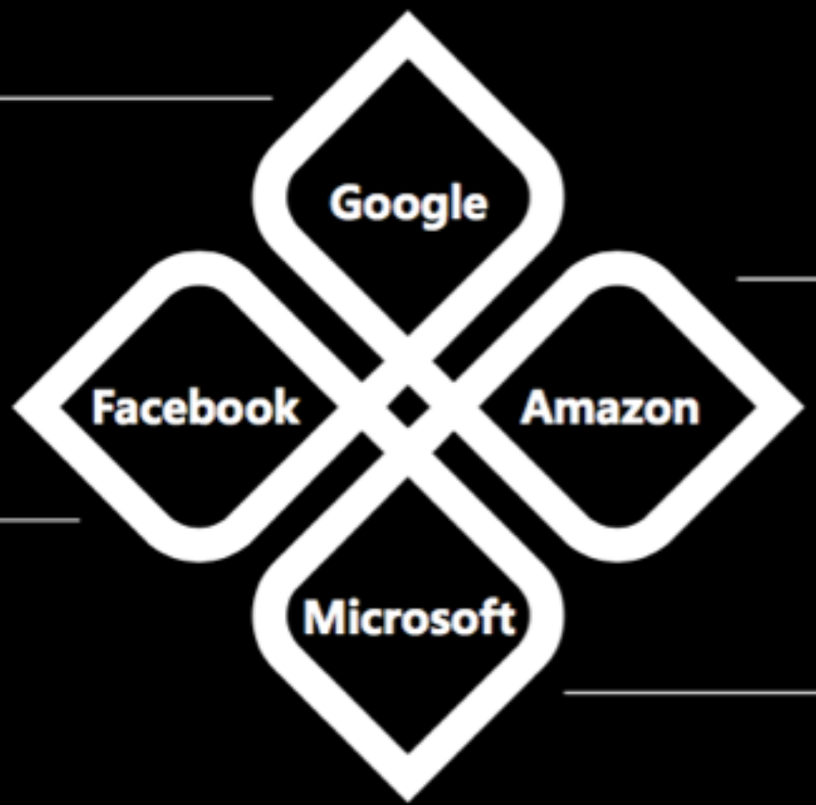
搜索全流程
速度现状

搜索极速浏览框架
探索与实现

应用现状与收益

业界知名公司对速度的认知

每增加500ms，搜索量减少25%



每增加100ms，成单量减少1%

每增加500ms，用户浏览减少3%

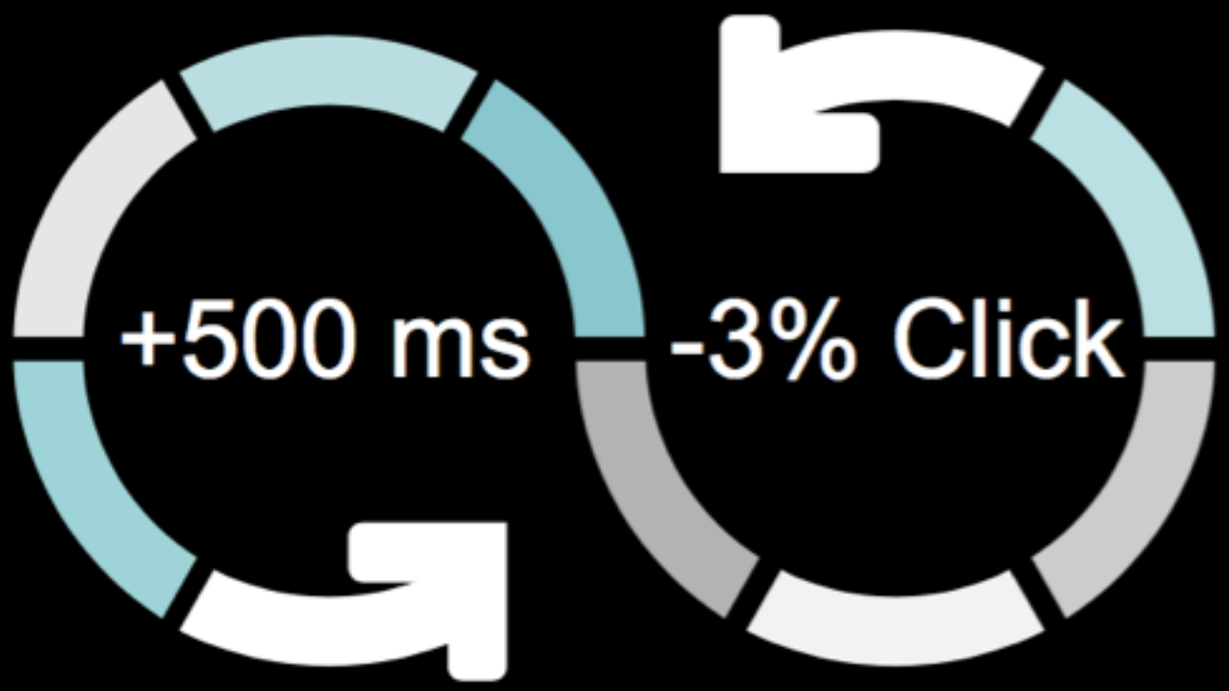
Bing搜索耗时增加1s，收入减少2.8%

其他一些数据

Still not convinced? Check out the following research:

facebook	Facebook pages that are 500ms slower result in a 3% drop-off in traffic, 1000ms is 6% drop-off
amazon.com	If Amazon increased page load time by +100ms they lose 1% of sales
Google	If Google increased page load time +500ms they get 25% fewer searches.
shopzilla	Reduced page load times from 7 seconds to 2 seconds and saw a 7% - 12% increase in conversion rate. Increased page views by 25% by decreasing their load time by 5 seconds.
YAHOO!	If Yahoo increased page load times by +400ms they see a 5 – 9% drop in full-page traffic
mozilla Firefox	If Firefox reduced load times by 2.2 seconds they would see an increase in download conversions by 15.4%
bing	A 1 second delay in Bing results in a 2.8% drop in revenue, 2 seconds is 4.3%
NETFLIX	Netflix saw a 43% drop in outbound traffic after enabling compression
Windows Live Hotmail	Hotmail discovered that a 6 sec delay in time to load caused a 40 Million drop in ad impressions per month, which equates to a $6 Million drop in ad revenues per year

速度对百度搜索的意义



速度与用户体验息息相关！

搜索全流程

百度搜索速度现状

80%用户搜索首屏时间

用户发起搜索

展现搜索结果页

用户点击某条结果

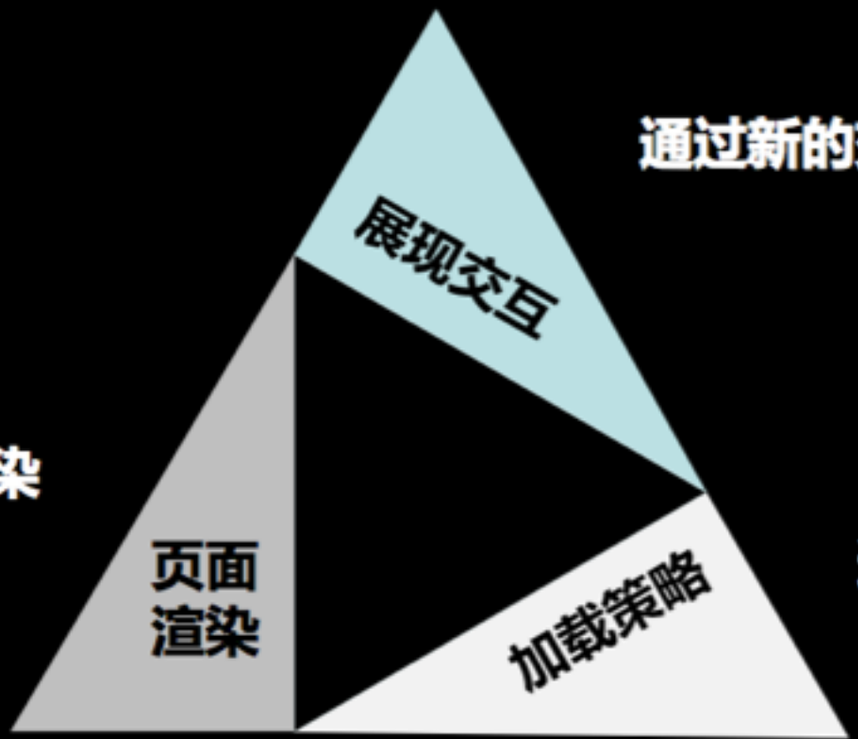
展现搜索结果站点

后续...

百度搜索结果站点速度现状

平均加载时间

全流程优化思路

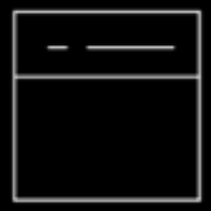


通过新的交互方式改善用户交互感知

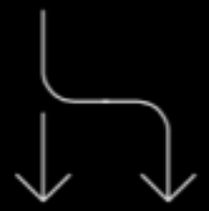
前置基础框架，优化渲染

通过预加载/预渲染再次提速

框架解决这些问题

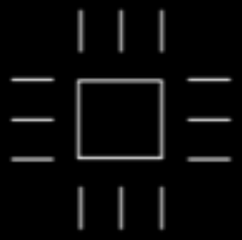


创建容器视图
提升展现交互体验



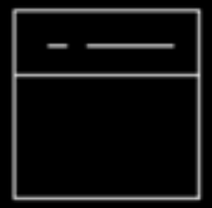
异步加载渲染
提升页面渲染速度

提前预取实例
优化网络与服务开销



提供组件机制
优化细粒度模块速度

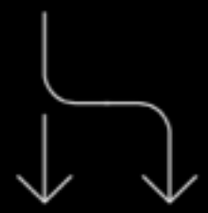
百度搜索：海底捞



创建容器视图
提升展现交互体验

#velocityconf

O'REILLY
Velocity



异步加载渲染
提升页面渲染速度

同步请求

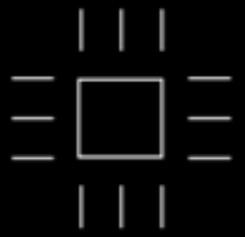
sf?openapi=1&dspName=iphone&f... /from=844b	GET	200 OK	document	Other	80.2 KB 251 KB	323 ms 302 ms		

异步请求

sf?openapi=1&dspName=iphone&f... /from=844b	GET	200 OK	xhr	VM2657:1 Script	42.5 KB 142 KB	366 ms 343 ms		

提前预取实例
优化网络与服务开销

提前预取页面内容



提供组件机制
优化细粒度模块速度

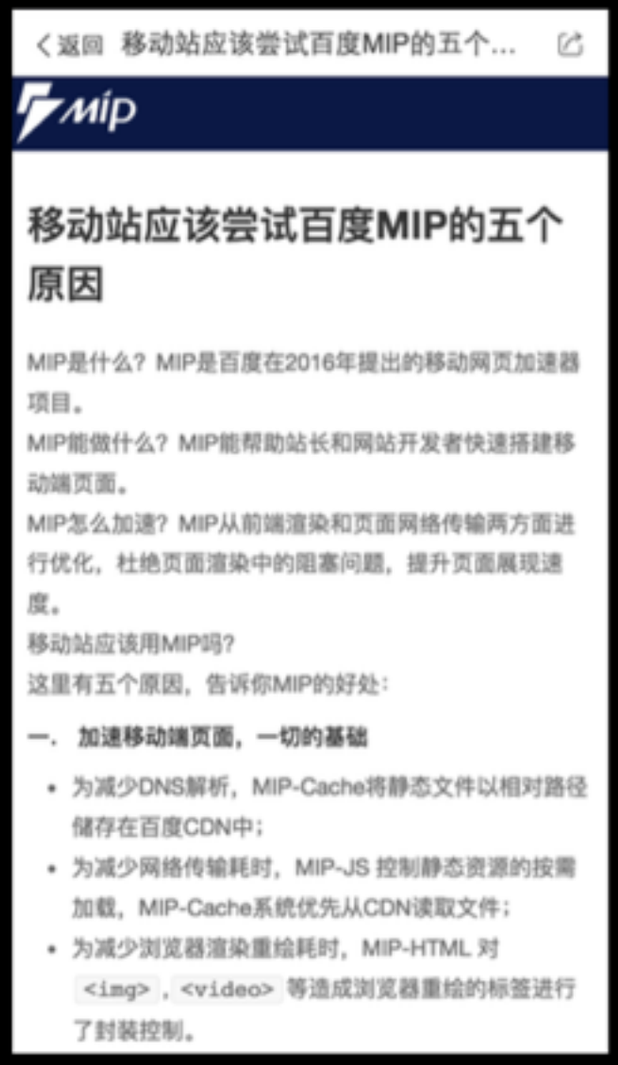
加速原理


传统页面
点击
网络连接
服务端响应
基础视图渲染
页面内容渲染
组件加载
加速页面
点击
展现交互响应
网络连接
基础视图渲染
服务端响应
页面内容渲染
组件加载

站点如何达到类似体验

Frame

MIP Bridge



MIP Page

Frame：极速浏览框架

MIP Bridge：MIP页面在框架中的运行环境

MIP Page：MIP页面

应用现状

到达提升
5%~40%

覆盖
10亿+页面

速度提升
30%~80%

核心价值：搜索体验的提升

对于我们的意义

QQ空间亿级服务Web架构

刁维康（腾讯）

QQ空间亿级服务Web架构

为什么选择留言板?

日访问量一亿+

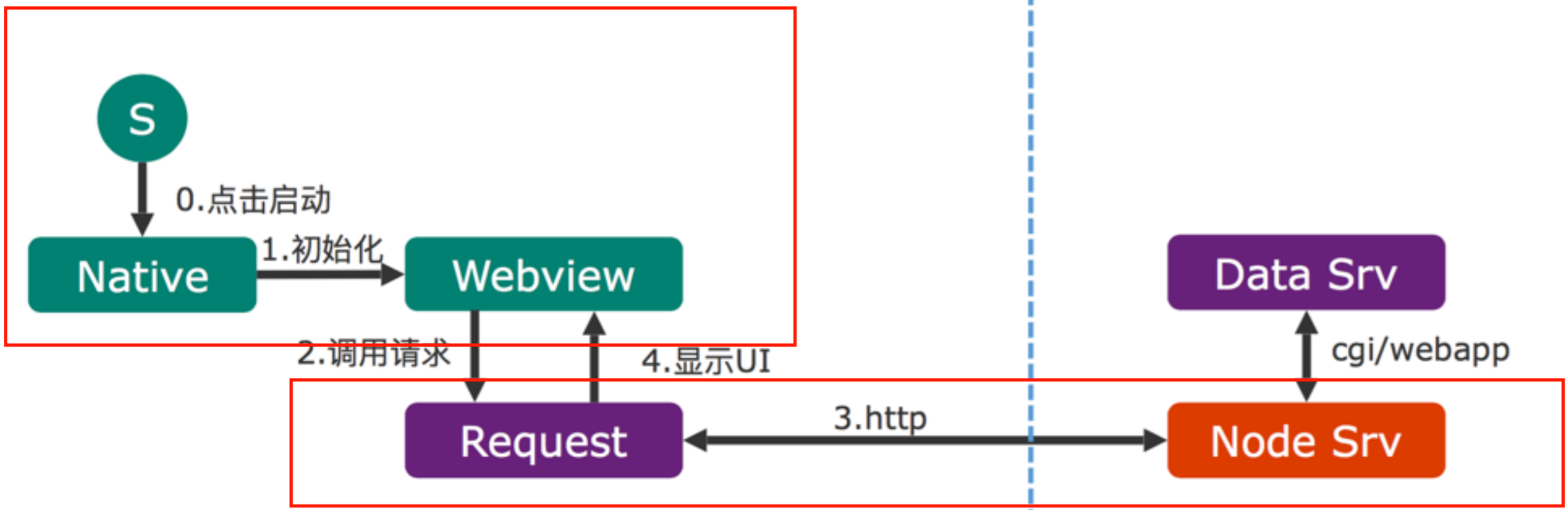
第一个从Native → Hybrid

留言板空白好久才显示出来

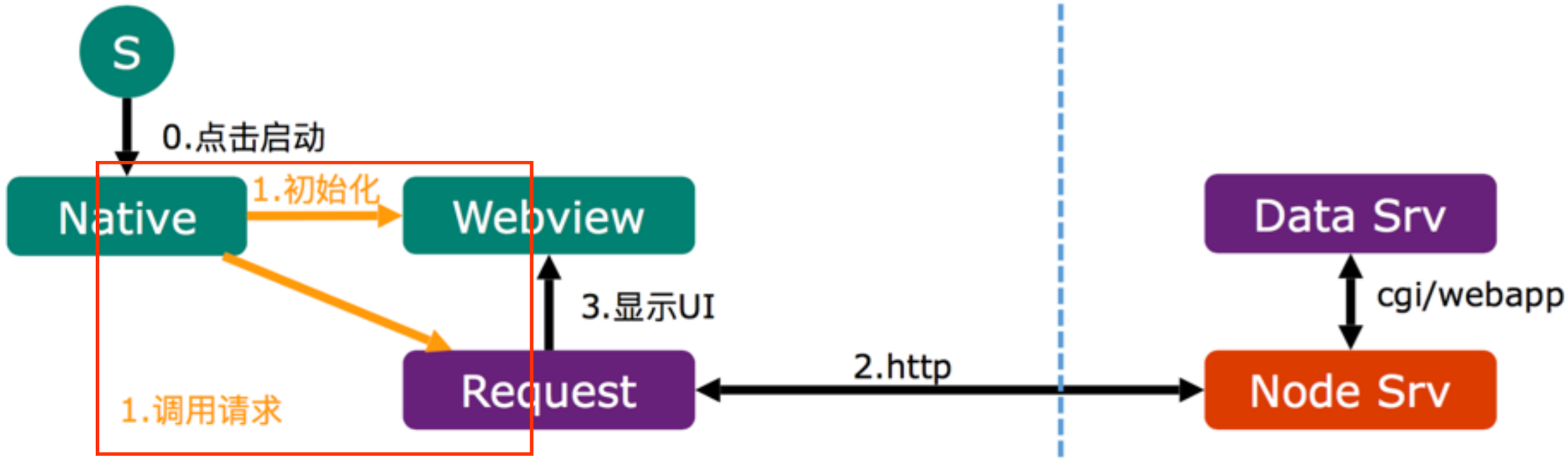
更新后打开留言板慢死了

1.

首屏加载慢



总耗时 = 初始化WV + 网络请求



总耗时 = Max(初始化WV，网络请求)

优化前后对比（ms）

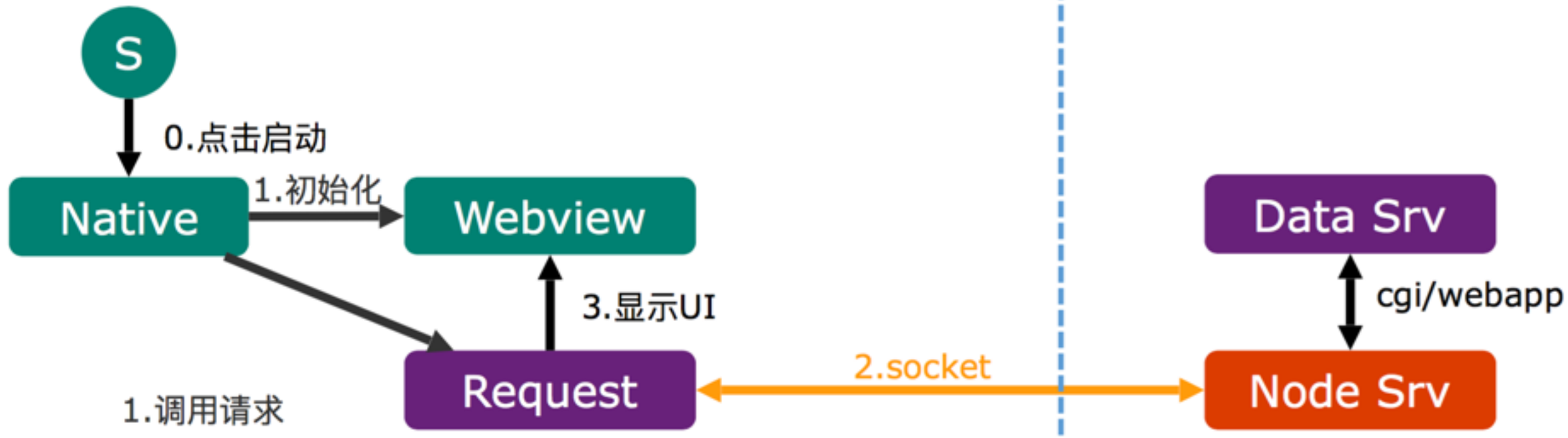
	iOS	Andorid
WV启动	400	800
加载数据	800	1200
优化前总	1200	2000
优化后总	800	1200

QQ消息能发送，留言板打不开

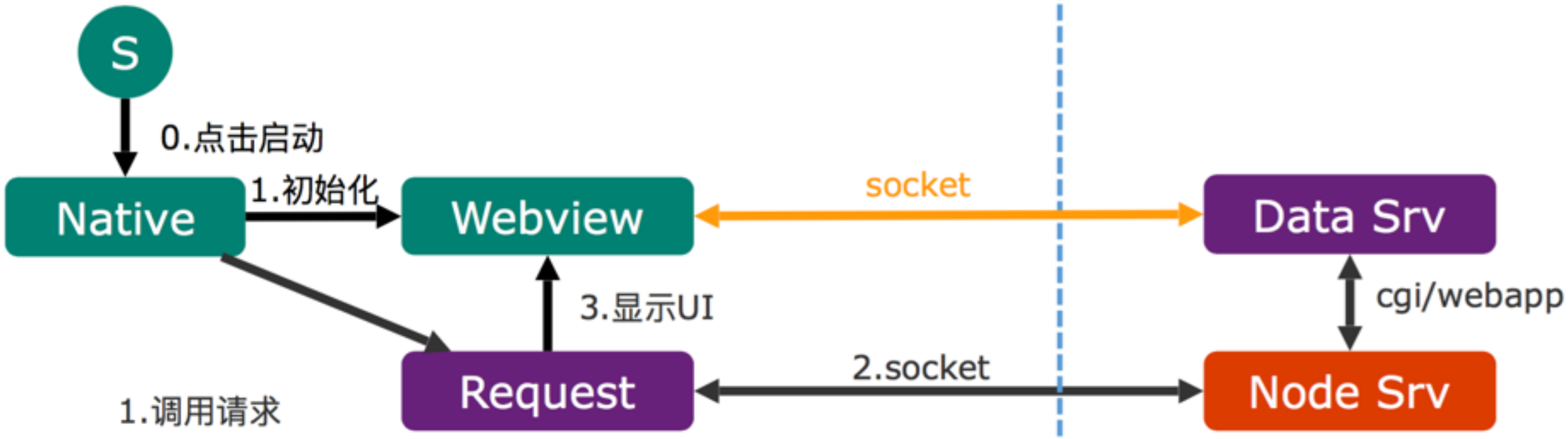
打开留言板居然有运营商广告

2.

稳定性和私密性差


S
0.点击启动
Native
1.初始化
Webview
3.显示UI
1.调用请求
Request
2.socket
Data Srv
cgi/webapp
Node Srv

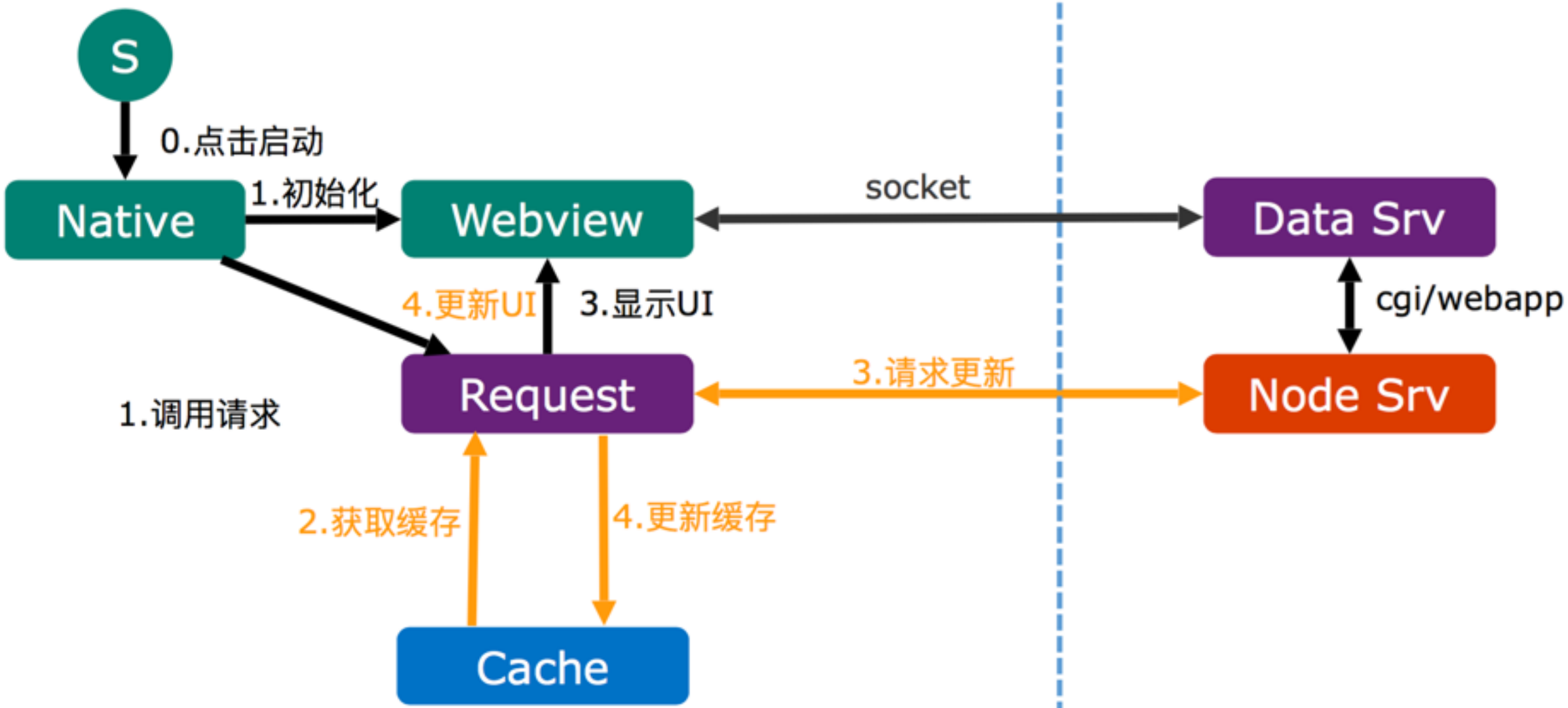
能聊天就能打开页面


S
0.点击启动
Native
1.初始化
Webview
socket
Data Srv
3.显示UI
cgi/webapp
1.调用请求
Request
2.socket
Node Srv

异步请求的复用方案

3.

二次打开还不够快


S
0.点击启动
Native
1.初始化
Webview
socket
Data Srv
4.更新UI
3.显示UI
cgi/webapp
1.调用请求
Request
3.请求更新
Node Srv
2.获取缓存
4.更新缓存
Cache

二次请求使用缓存提速

校验本地缓存的有效性?

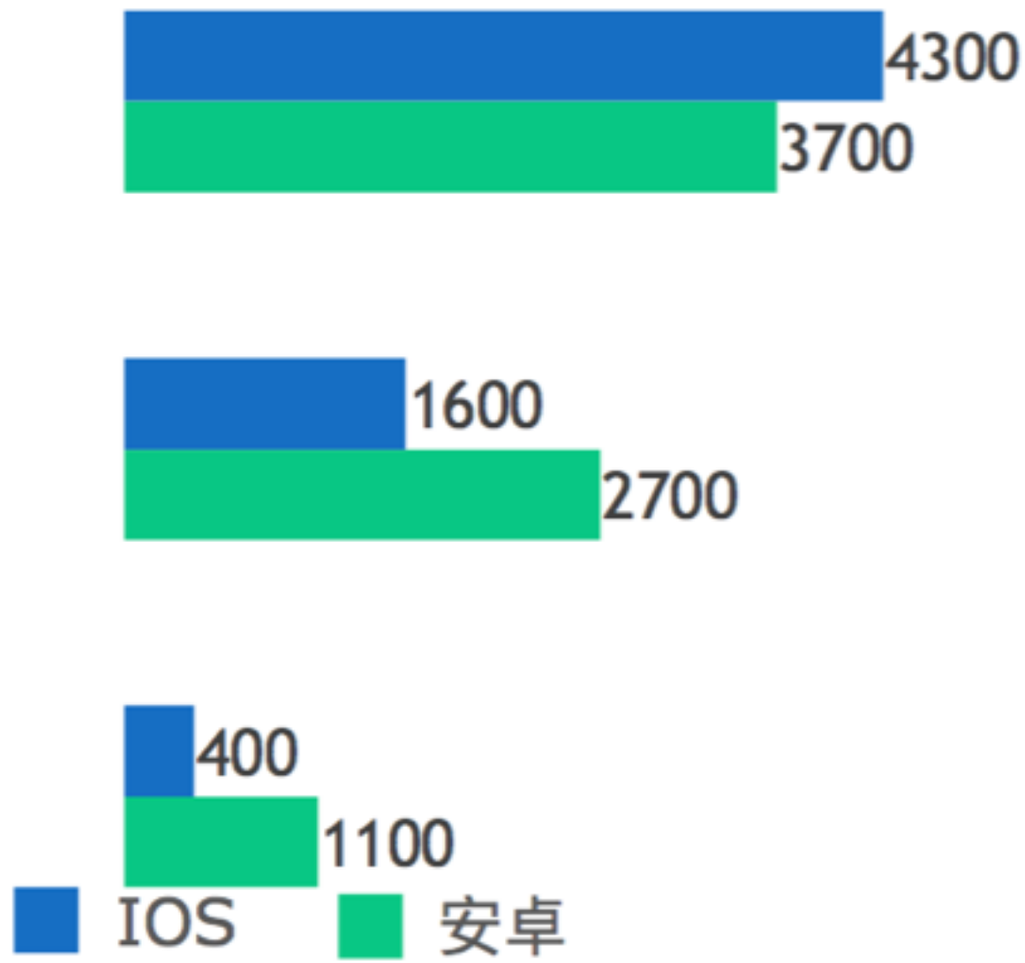
“ETAG”

TTP

打开

缓存

首屏耗时对比（单位ms）



数据来源：来自手机QQ留言板功能


上午1:48
63%
返回
留言板
留言
留个言吧...
vitoxu
回复
11-08 15:24 空间描述
网络无法连接
确定
vitoxu
回复
11-08 15:24 空间描述
我来也~
IM COMING
TWO WORLDS. ONE HOME.
diaodiao
删除

离线也能打开

首屏离线后的问题

首屏html在本地

弱网CSS加载慢

无网CSS加载失败

首屏会白屏

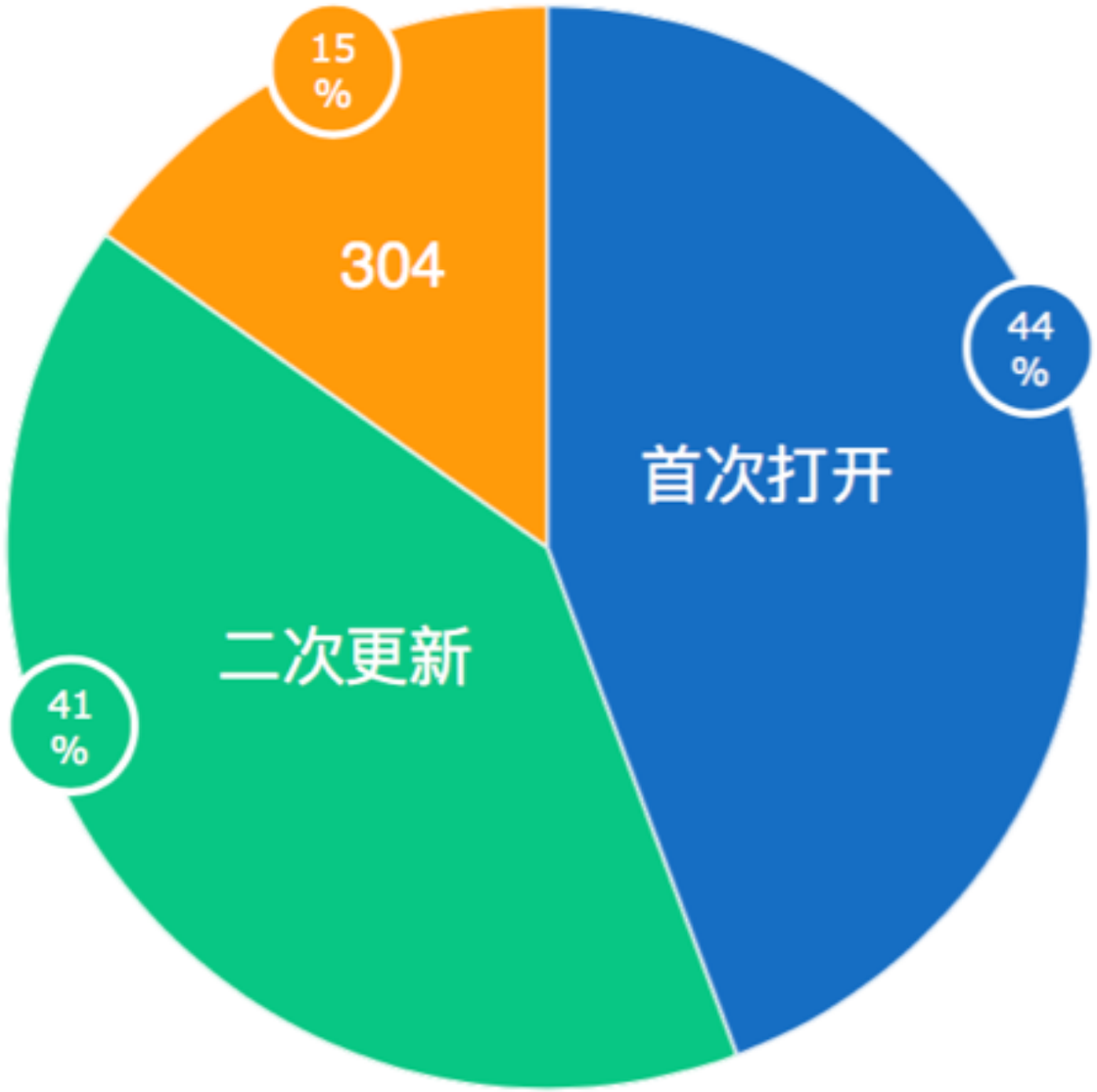
首屏无样式

解决方案：CSS文件内联

4.

首屏流量变大

首屏请求情况



数据来源：来自手机QQ留言板功能

动静分离

```
<!DOCTYPE html>
<html>
  <head>
    <title>
    xx的留言板
    </title>
    <style>
    ...内联样式...
    </style>
    </head>
    <body>
    <!-- dynamic-start -->
    <div>...用户内容...</div>
    <!-- dynamic-end -->
    <script src="./msg.js">
    </script>
    </body>
</html>
```

HTML

=

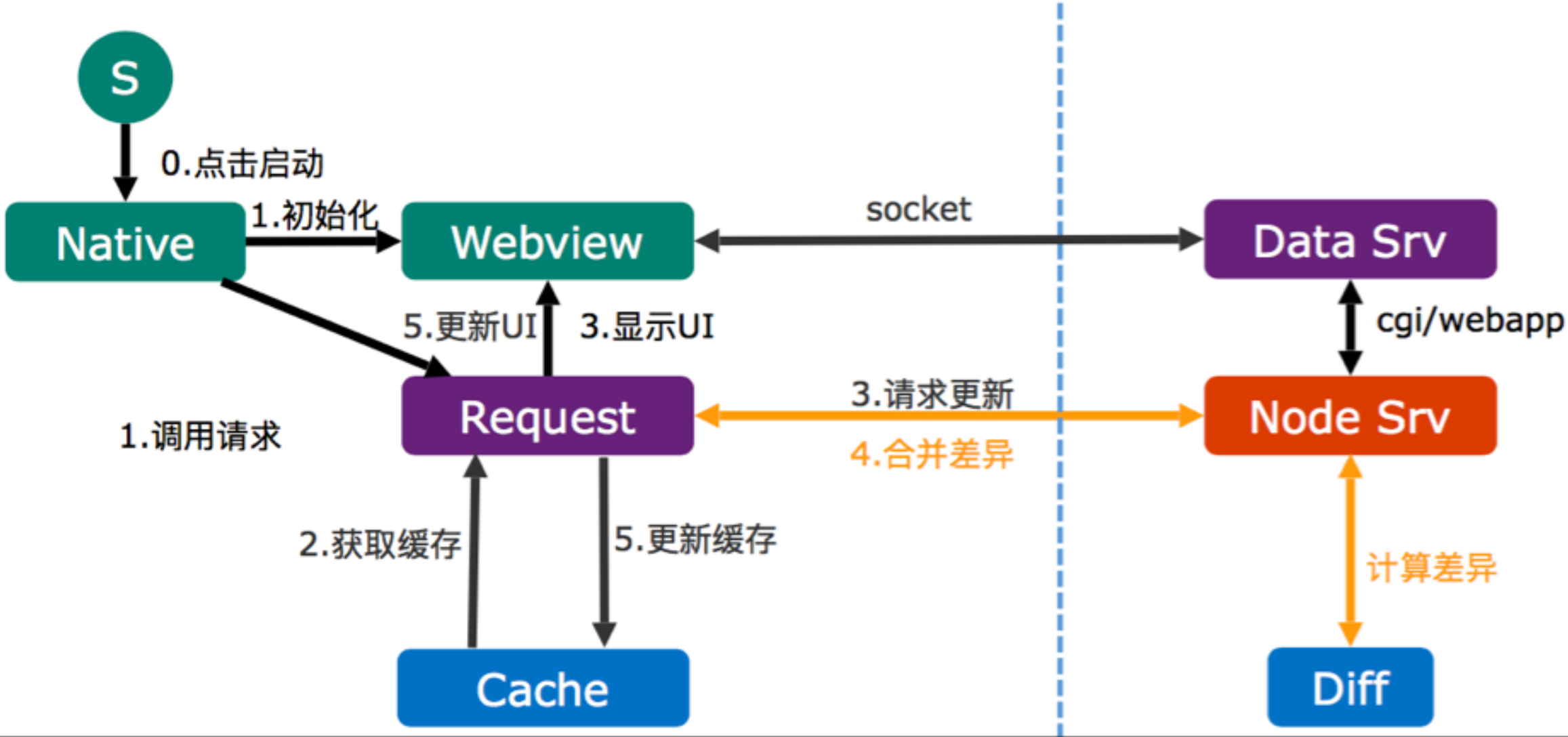
```
<!DOCTYPE html>
<html>
    <head>

    {title}

    <style>
    ...内联样式...
    </style>
    </head>
    <body>

      {dynamic}

    <script src="./msg.js">
    </script>
    </body>
</html>
```

Templat
e

+

```
{
    title: `<title>
            xx的留言板
    </title>`,

    dynamic: `<div>...
    用户内容...</div>`
}
```

Data


S
0.点击启动
Native
1.初始化
Webview
socket
Data Srv
cgi/webapp
5.更新UI
3.显示UI
1.调用请求
Request
3.请求更新
Node Srv
4.合并差异
2.获取缓存
5.更新缓存
计算差异
Cache
Diff

差异化更新省流量

流量减少 40kb → 6kb 85%

本地有缓存 41%

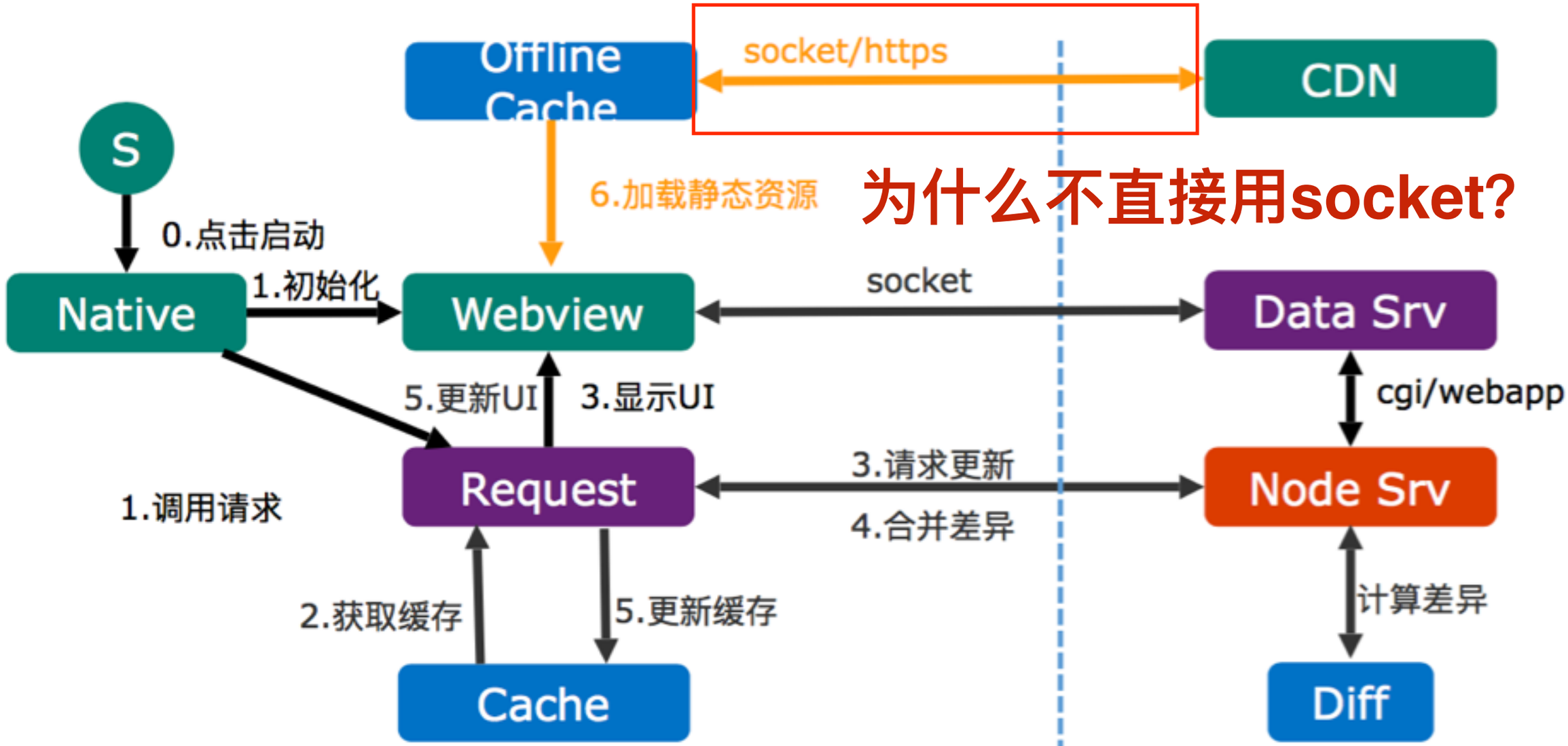
X 命中diff 43%

总流量降低 15%

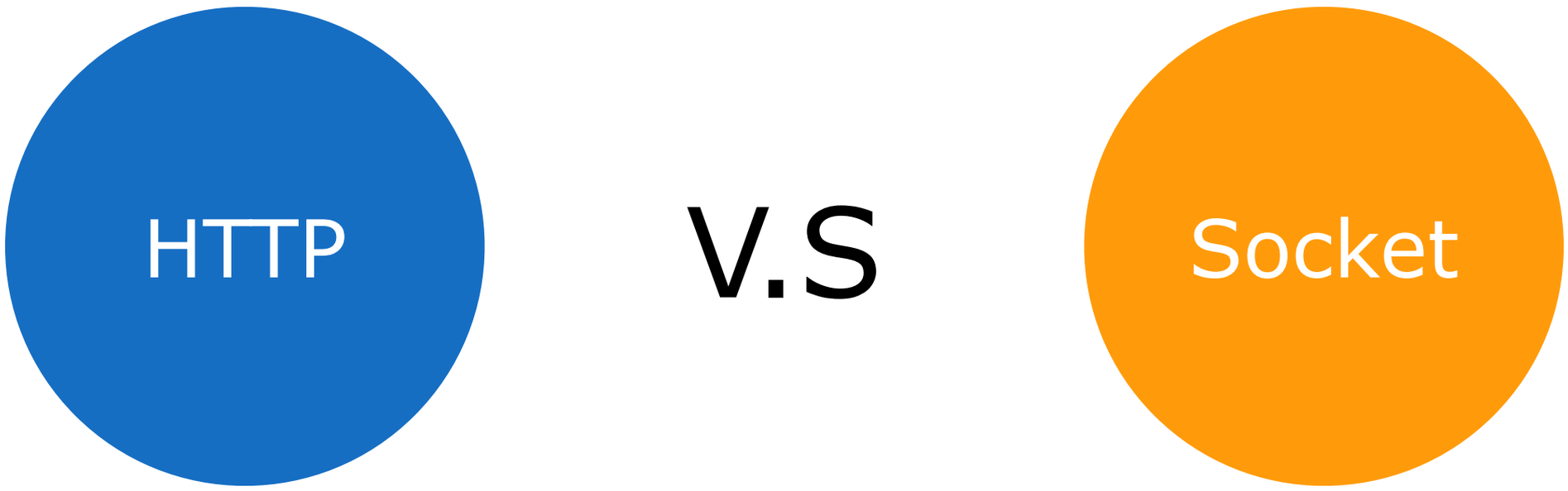
数据来源：来自手机QQ留言板功能


并行
通道
缓存
DIFF

静态资源怎么办?


Offline Cache
socket/https
CDN
S
6.加载静态资源
为什么不直接用socket?
0.点击启动
Native
1.初始化
Webview
socket
Data Srv
5.更新UI
3.显示UI
cgi/webapp
1.调用请求
Request
3.请求更新
4.合并差异
Node Srv
计算差异
2.获取缓存
5.更新缓存
Cache
Diff

静态资源离线缓存



用Socket比HTTP慢30%

数亿级用户规模下的
React native工程实践

berg

OUTLINE

- 为什么选择React native
- 与现有业务和迭代融合
- 性能优化实践
- React Native的工程价值

手机百度

- 迭代周期：4周
- 主要Feature：数十个
- 开发工时：数千人时

- 跨部门，沟通成本高
- 业务种类多，集成成本高

过去的解耦方案

迭代速度慢

开发、集成成本高

准入流程长

APK插件/SDK

Hybrid

体验、效果差

本地能力依赖发版

React Native@手机百度

- RN版本：**v0.35**
- RN用户数：**2亿+**
- RN启动次数：**15亿+**
- 总崩溃率：**0.07%**
- 24h收敛率：**98%**

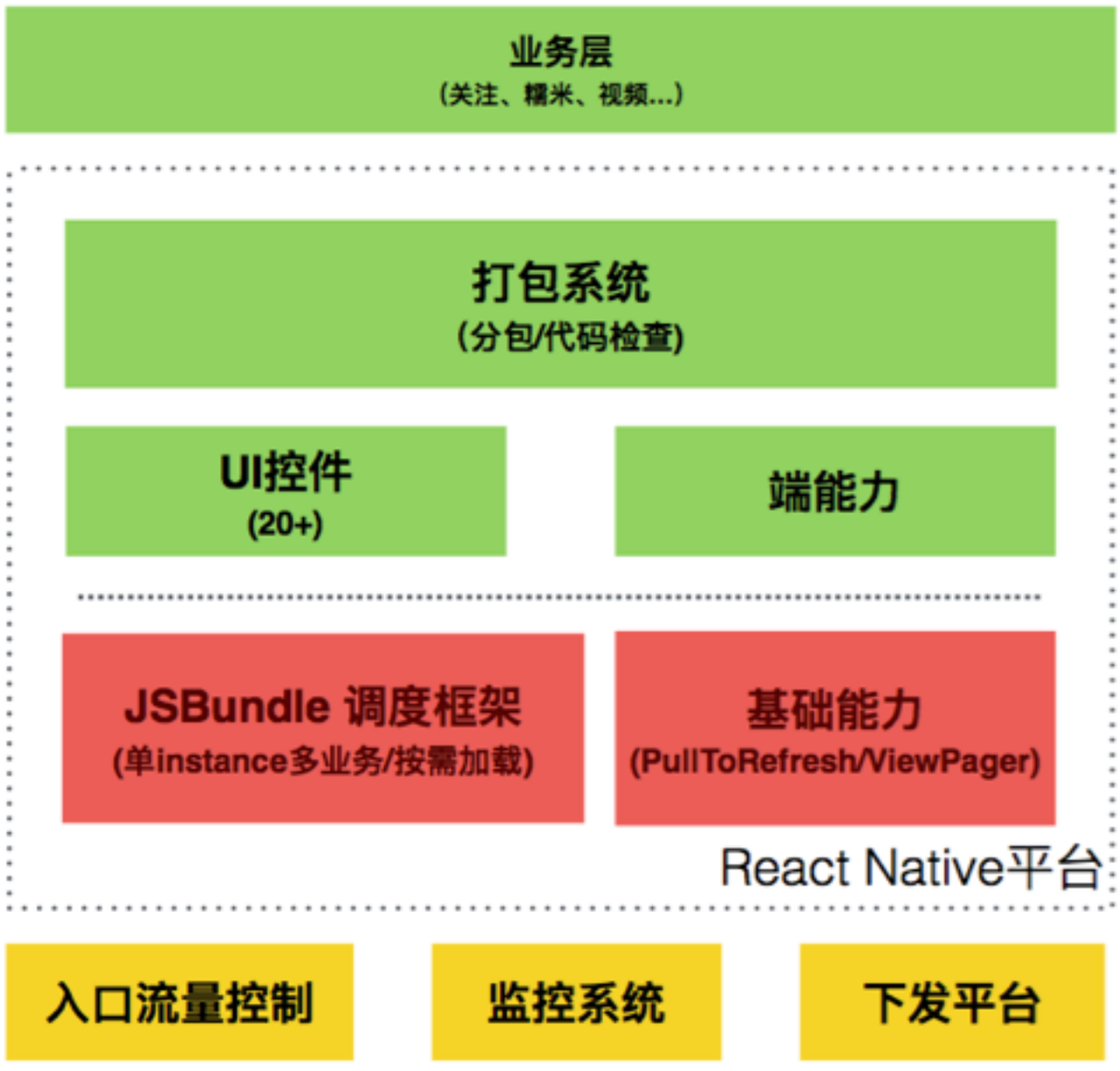
5-8倍的迭代速度

React Native 的工程实践

- 与现有业务融合
  选型、人力投入、回退方案…
- 迭代流程、质量控制
  准入、灰度、发布、回滚、监控…
- 性能优化
  加载时间、Listview、动画…

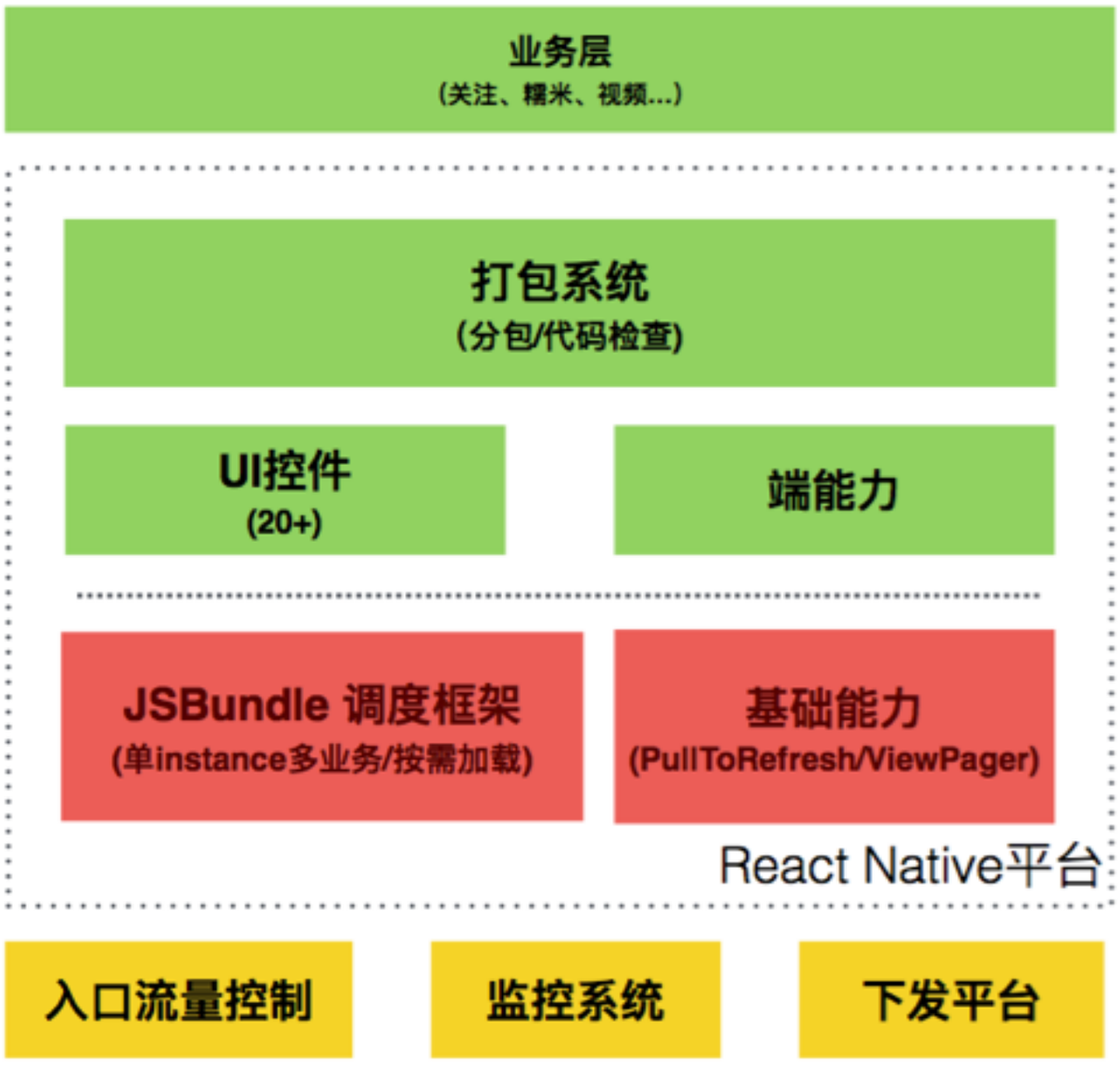
需要系统性的考虑

整体架构 – 兼容性



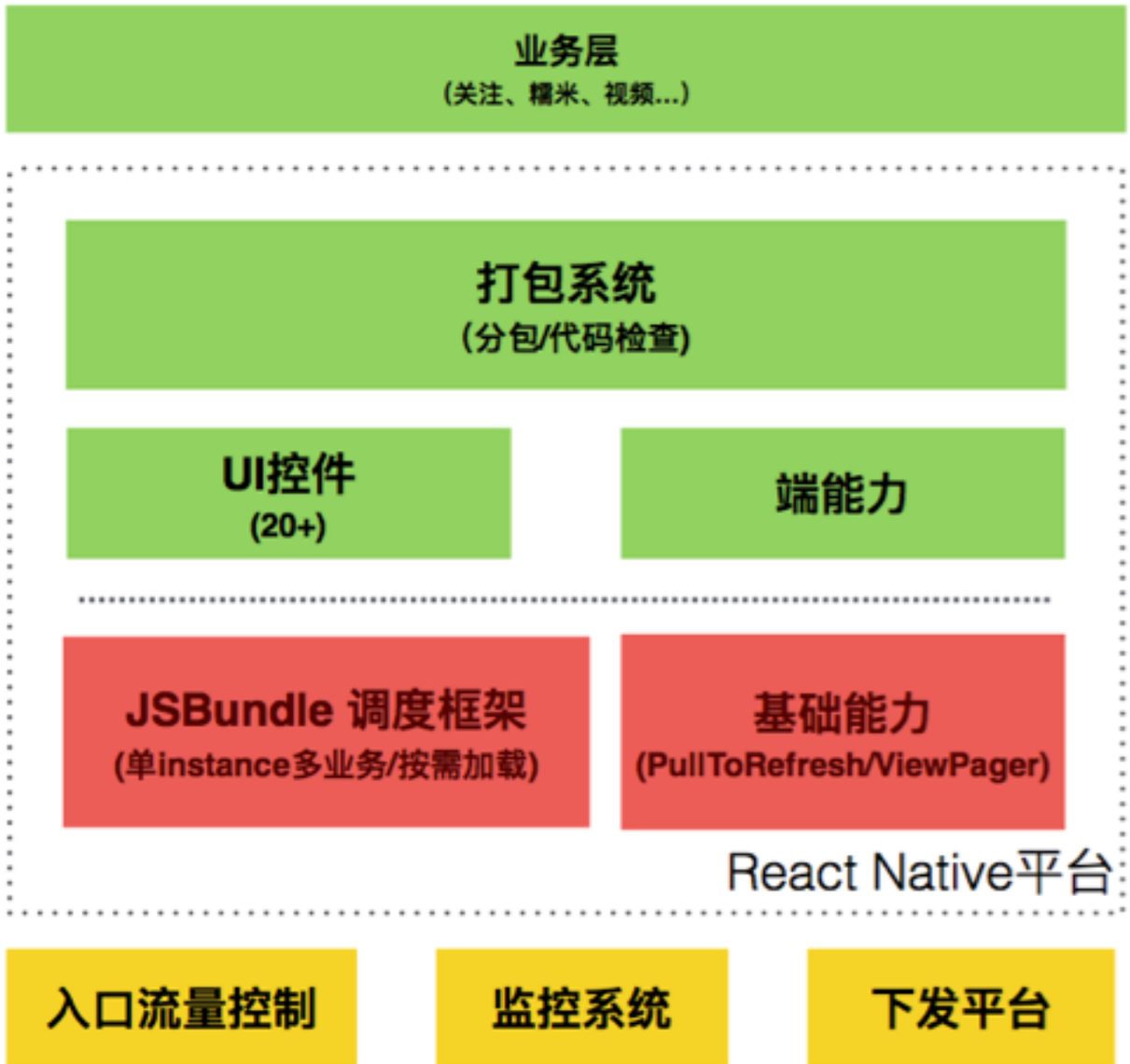
- Android：
  - >=4.1 (99%)
  - V29 >=4.4 (88%)
    listview滑动内存释放缓慢，或与硬件加速有关
- iOS： >=8.0 (95%)
- 与宿主工程的兼容性
  soLoader、Fresco、OKHttp

整体架构 – 兼容性



- 降级和兼容方案
  - 随时回撤
  - AB Test

整体架构 – 业务层



- HTML -> JSX
- CSS -> CSS-layout
  无float/px/css继承/css sprites
- ES 5 -> ES 6
- DOM -> React native View
- 推荐Redux
  统一管理state，减少事件滥用

业务层

- HTML -> JSX
- CSS -> CSS-layout
  无float/px/css继承/css sprites
- ES 5 -> ES 6
- DOM -> React native View

- 性能
  - 空间换时间、便利性换性能
  - 减少重绘、减少通讯

统一交互协议

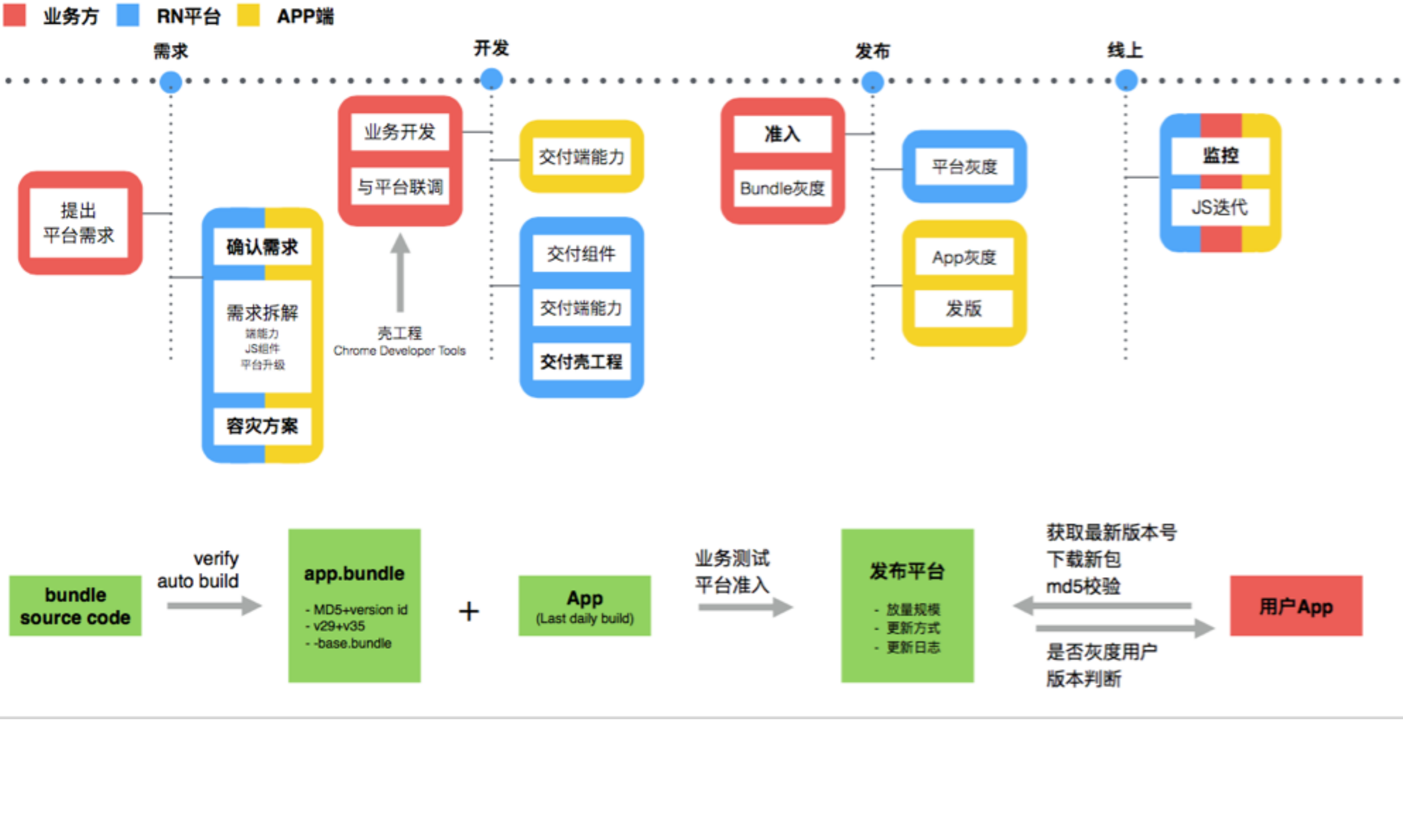

```
import {invoke} from 'baidu-invoke';
invoke('ClassName.methodName', {
    key: value
})
  .then(data => console.log(data.result))
  .catch(error => console.log(error.code, error.message));
```


```
import {eventListener} from 'baidu-invoke';
let event = eventListener.addListener($type, evt => {});
event.remove();
eventListener.removeAllListeners($type);
```

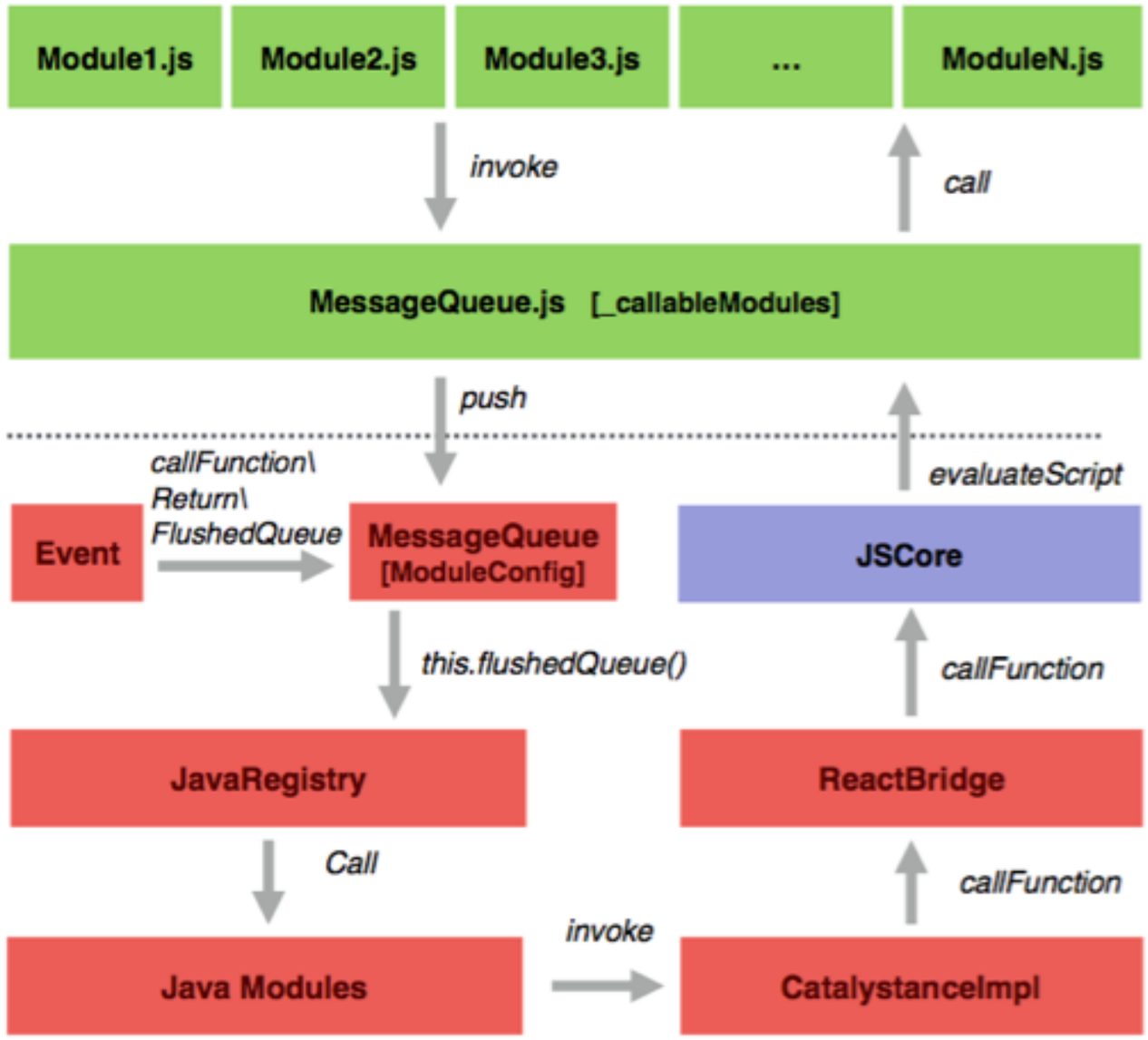

```
{
    "rn_bundle_id": "rnplugin.myattention",
    "rn_component_name": "MyAttention"
}
```

通用增量接口

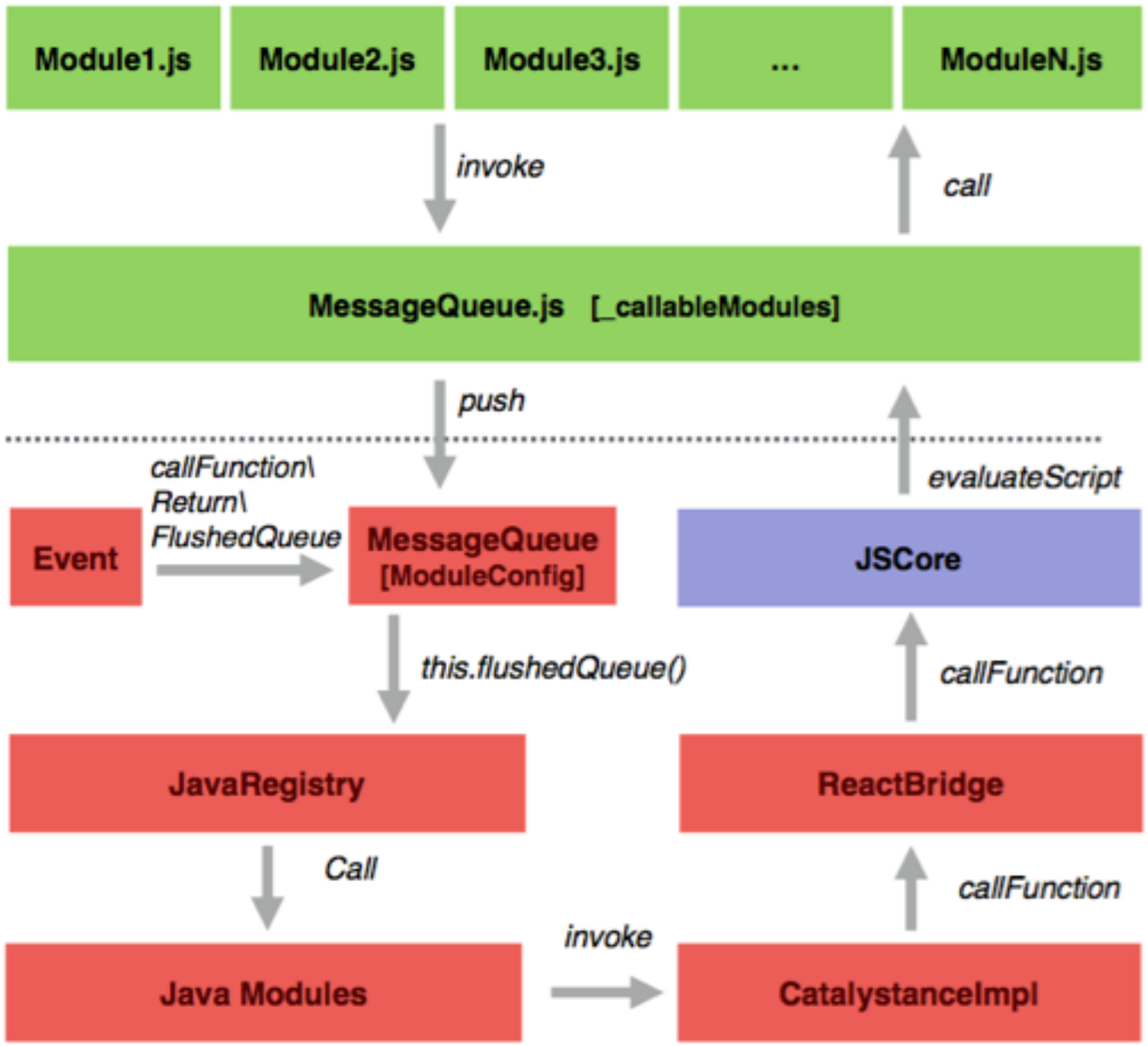
融入迭代流程


业务方
RN平台
APP端
需求
开发
发布
线上
提出
平台需求
确认需求
需求拆解
端能力
JS组件
平台升级
容灾方案
业务开发
与平台联调
壳工程
Chrome Developer Tools
交付端能力
交付组件
交付端能力
交付壳工程
准入
Bundle灰度
平台灰度
App灰度
发版
监控
JS迭代
bundle
source code
verify
auto build
app.bundle
- MD5+version id
- v29+v35
- -base.bundle
+
App
(Last daily build)
业务测试
平台准入
发布平台
- 放量规模
- 更新方式
- 更新日志
获取最新版本号
下载新包
md5校验
是否灰度用户
版本判断
用户App

React Native 通讯流程



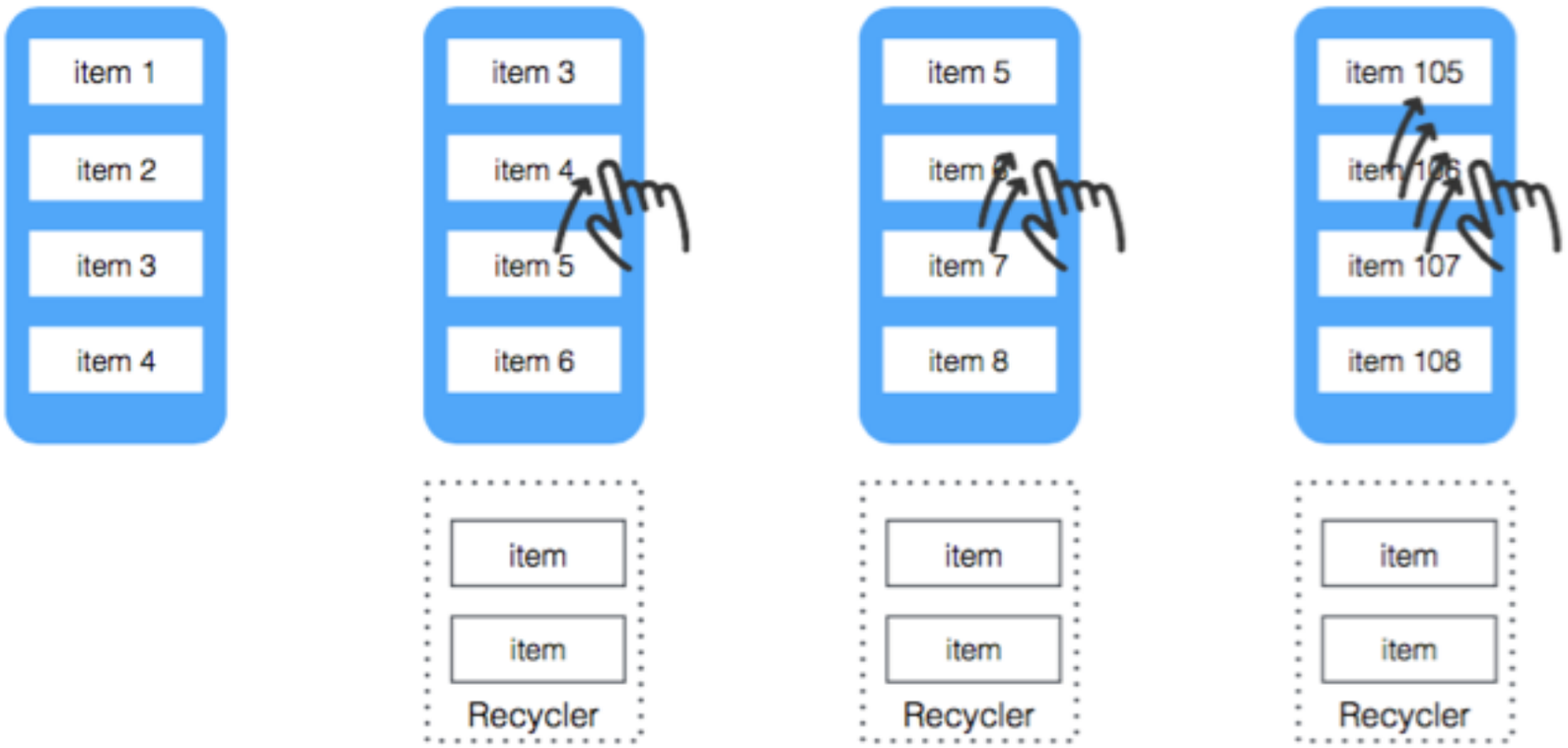
React Native 通讯流程

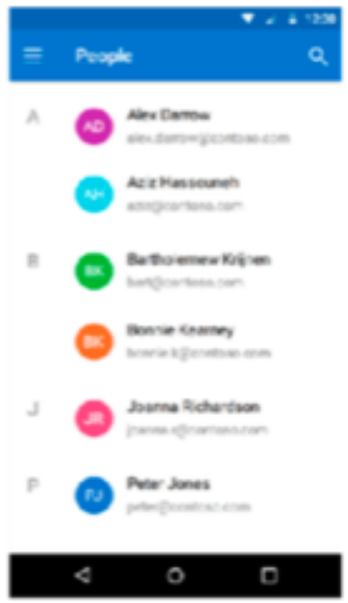


- 通讯频繁 by design
  JS拥有足够的灵活性
- 易导致卡顿
  - **事件频率高**
  - **JS运行时间过长**
  - **UI频繁重绘**
- 初始化成本较高
  - **生成模块配置表**

Listview @ React Native

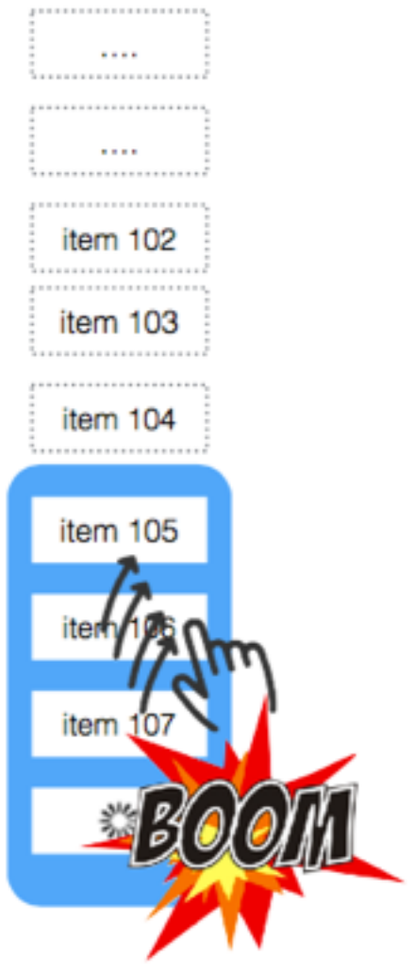
Listview @ Native





Anchor

Listview @ React Native

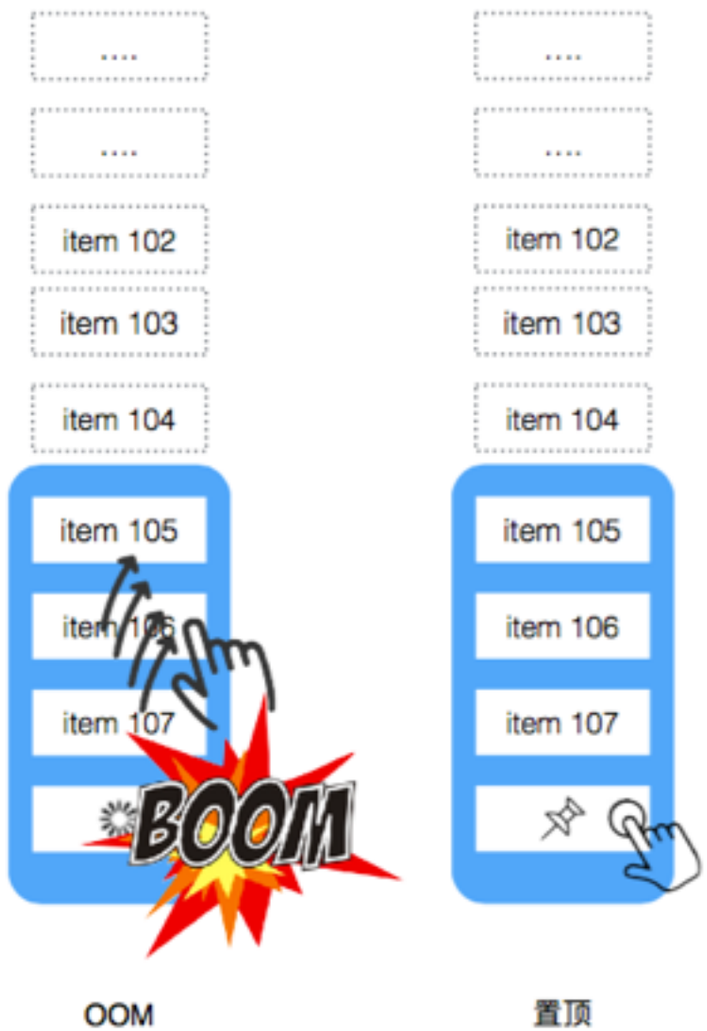


OOM

置顶

ttp://facebook.github.io/react-native/releases/0.31/docs/performance.html#listview-initial-rendering-is-too-slow-or-scroll-performance-is-bad-for-large-lists

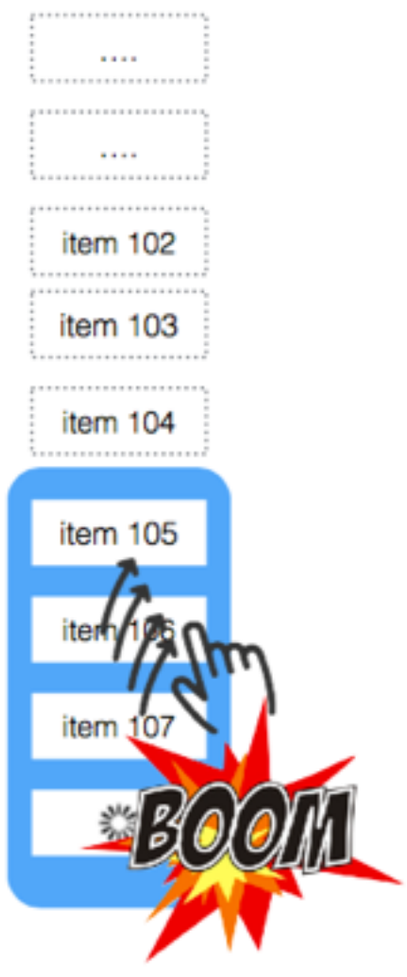
Listview 优化方案



OOM　　置顶

- 设计决定了View无法复用
  - **内存紧张**
- DOM diff雪上加霜
  - **大量重新渲染**
- 业务级优化
  - **预留空单元格**
  - **需大批量刷新时，从头渲染**

Listview 优化方案



OOM

置顶

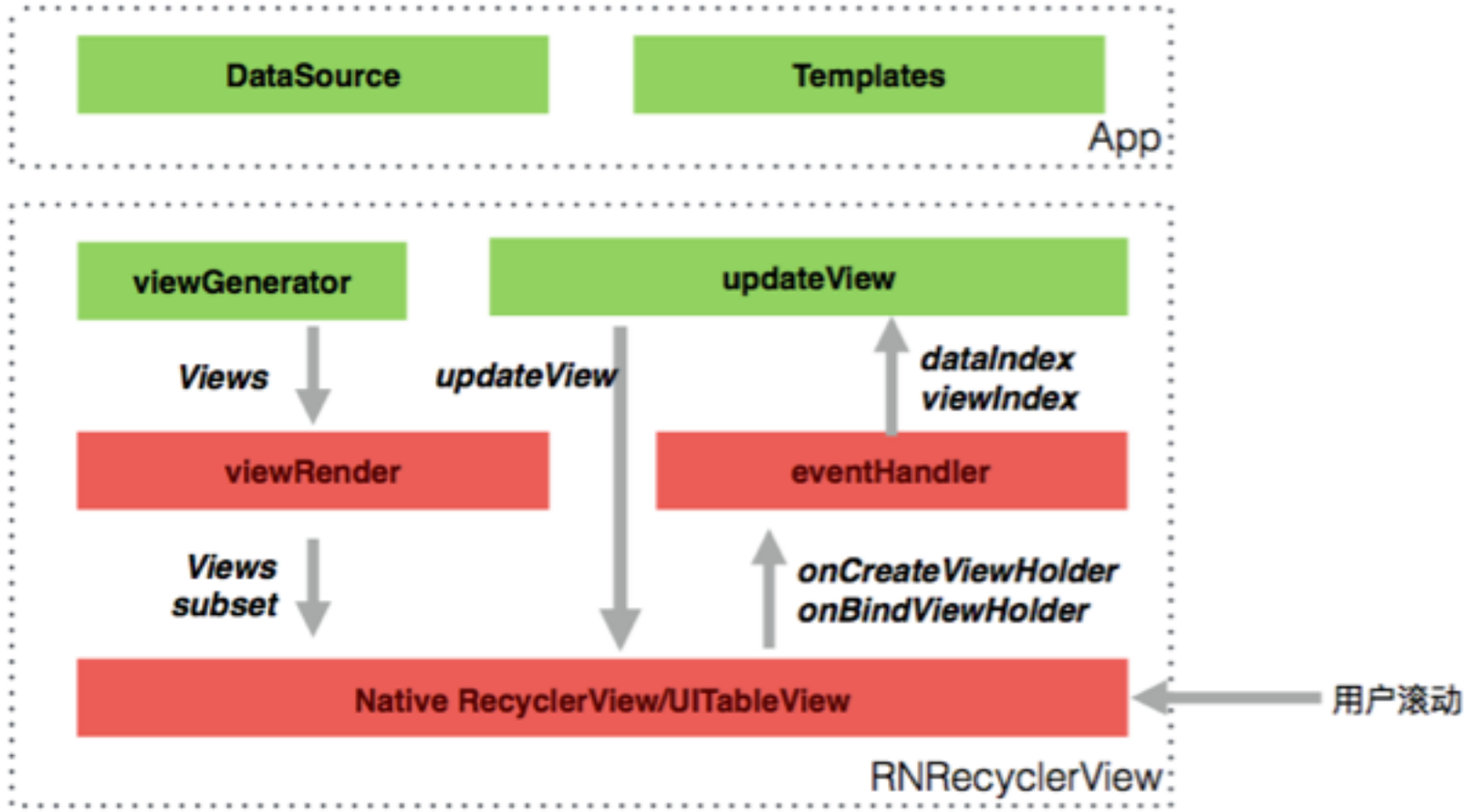
删除非可视区view


```
render() {
    return createElement(
        emptyView, {
            style: {
                width: this.viewProperties.width,
                height: this.viewProperties.height
            }
        });
}
```

- 实现十分简单
- 适合渲染强度低的列表
- 滚动过快会闪烁

Listview 优化方案

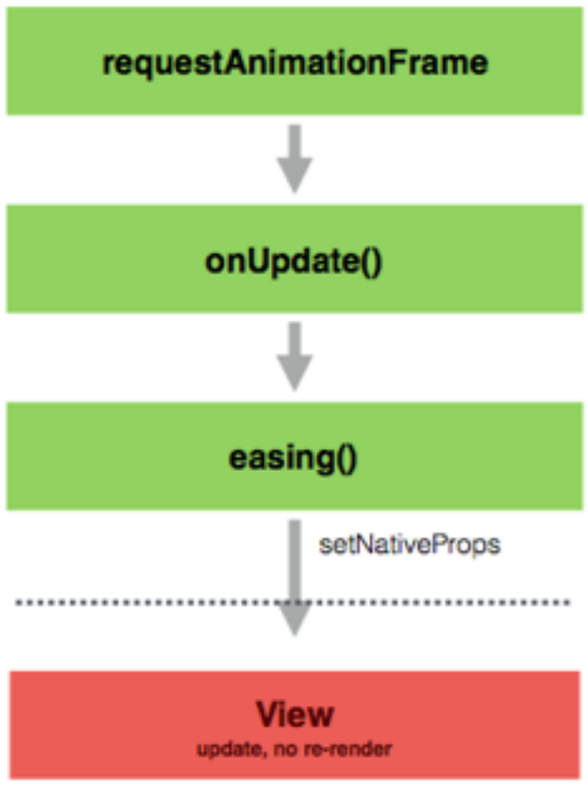
模板复用



- 可彻底解决OOM

- 一定程度上违背了RN的设计原则
- item在滚动时可能会跳动、闪烁

Animations @ React Native

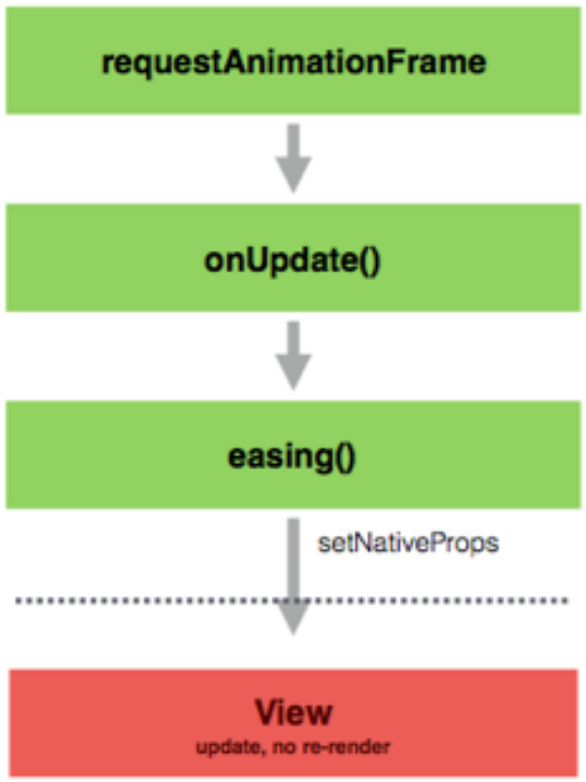


RN默认实现

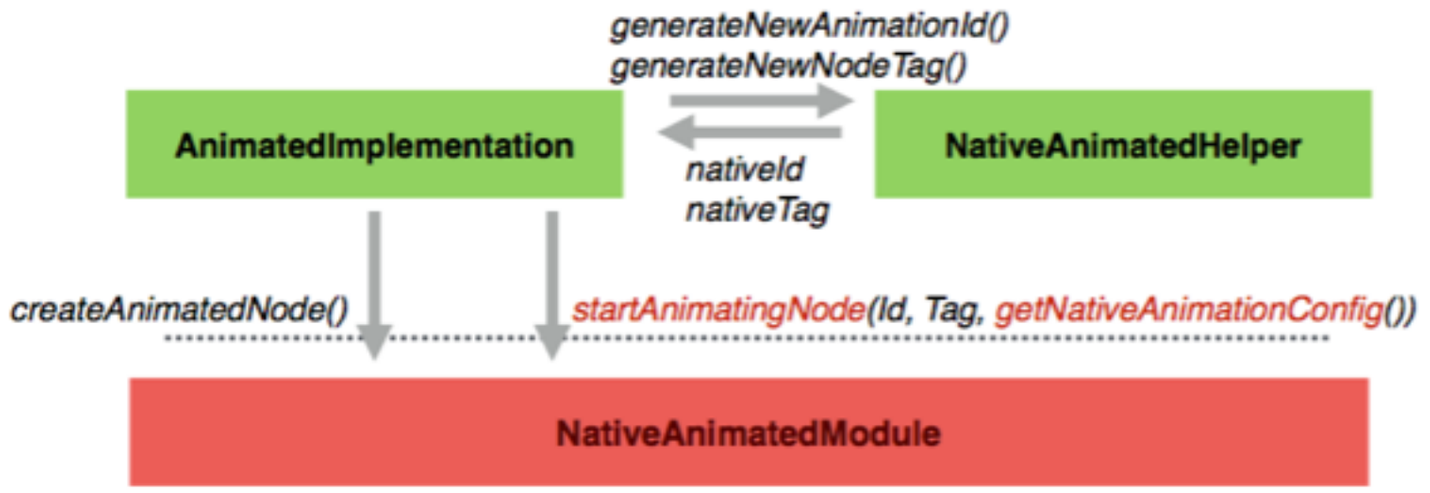
- 16ms
  - Javascript计算Props
  - 通讯
  - View Update
- 易导致丢帧
- systrace.py

https://facebook.github.io/react-native/docs/android-ui-performance.html
https://facebook.github.io/react-native/docs/animations.html
https://facebook.github.io/react-native/docs/direct-manipulation.html

Animations @ React Native



RN默认实现



useNativeDriver : true

- Native 直接完成动画绘制
  - **Javascript计算frames**
  - **通讯一次**
  - **View Update**
- 实验阶段，不够可靠

Animations 优化

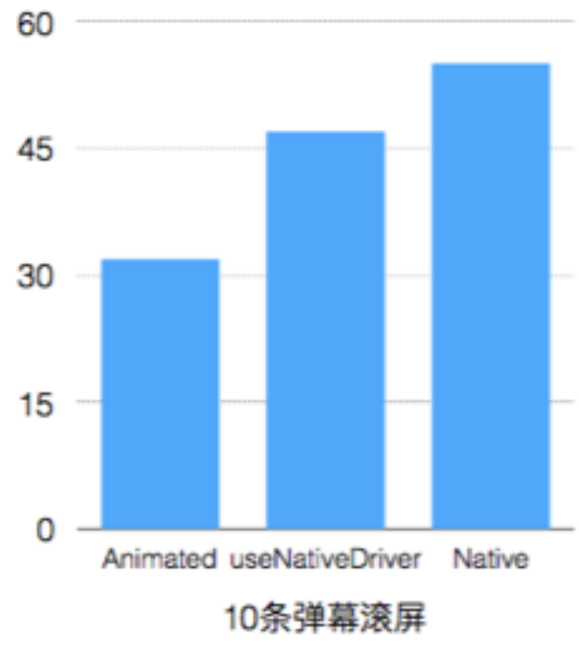
<AnimationView />

```
this.refs.view.
setTranslate({from, to}).
setOpacity(from, to).
setEasing({type: “bounce”}).
setDuration(1000).
onStart(() => {}).
start()
```

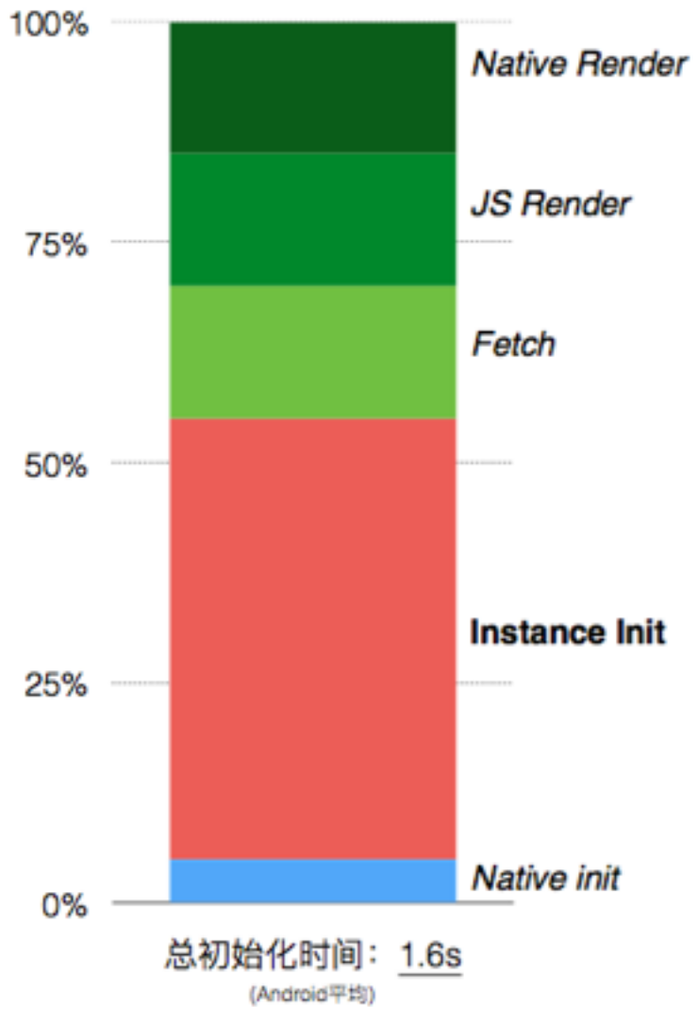
Pre-defined Animations

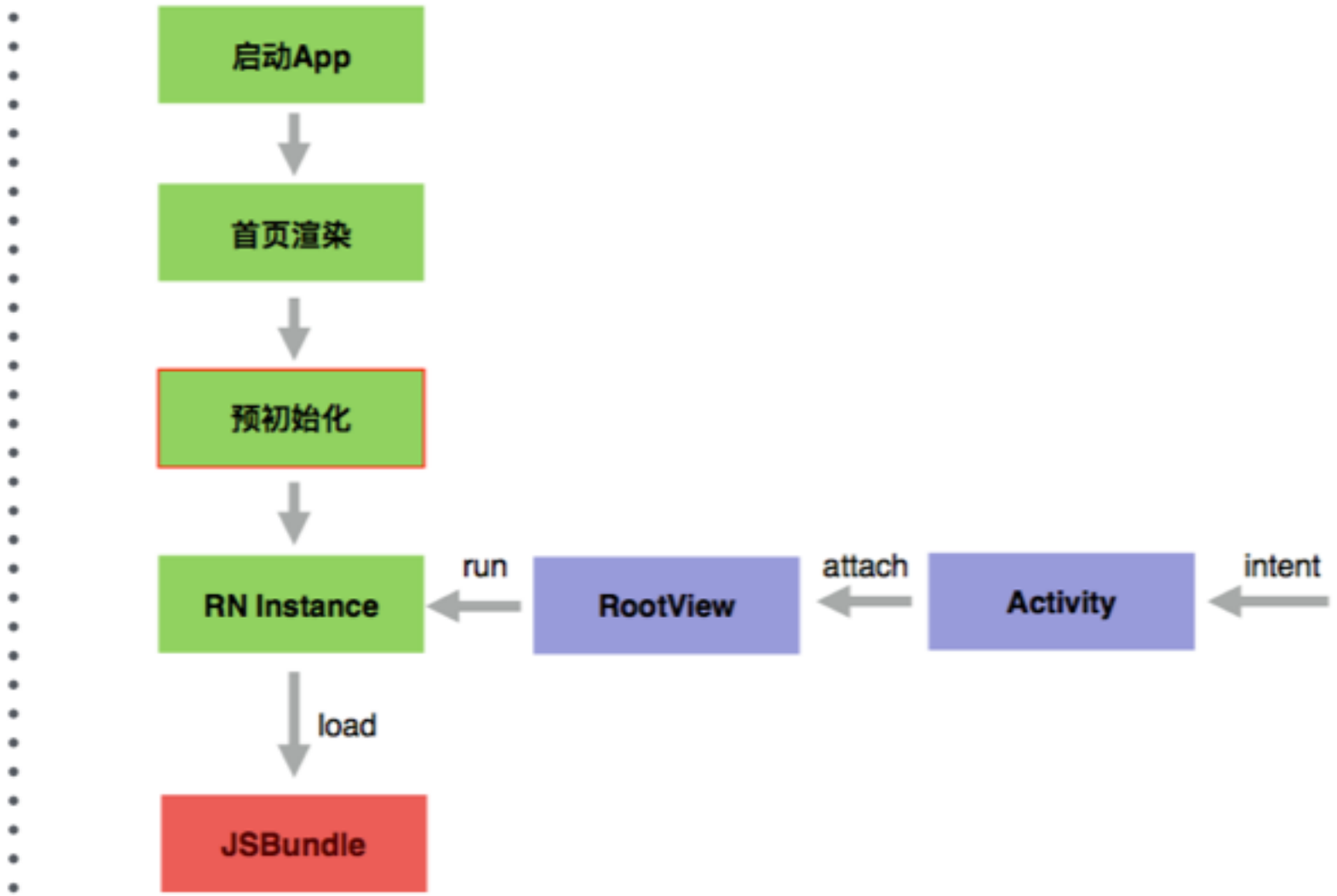
提供最常用的动画和easing实现

- **性能完全等价Native实现**
- **在长动画下优势明显**

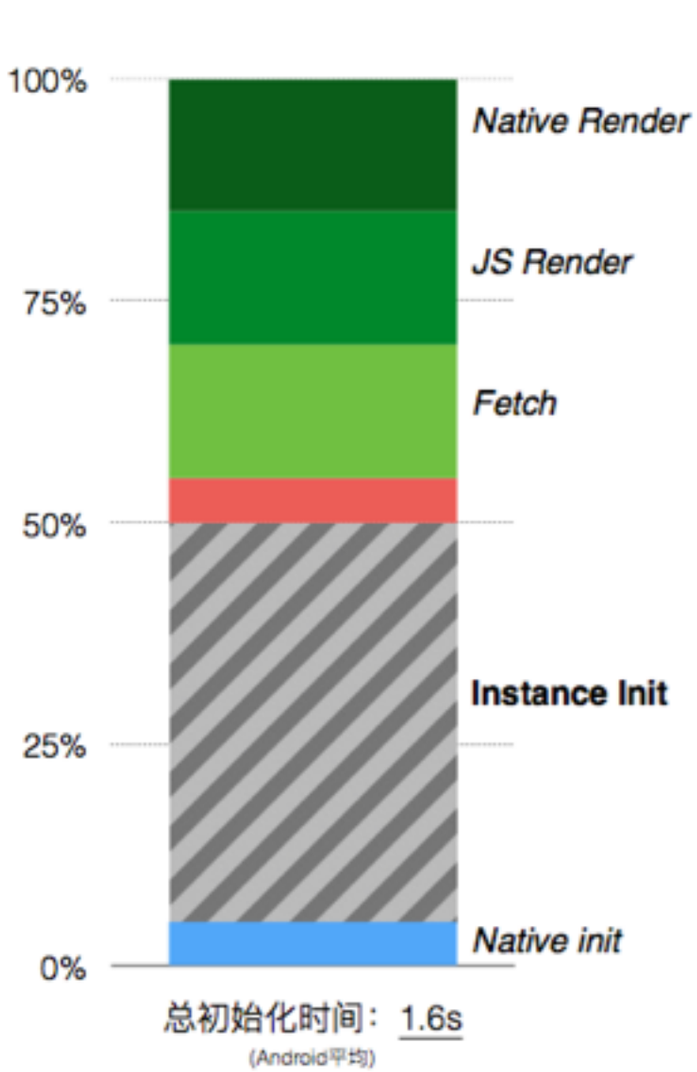


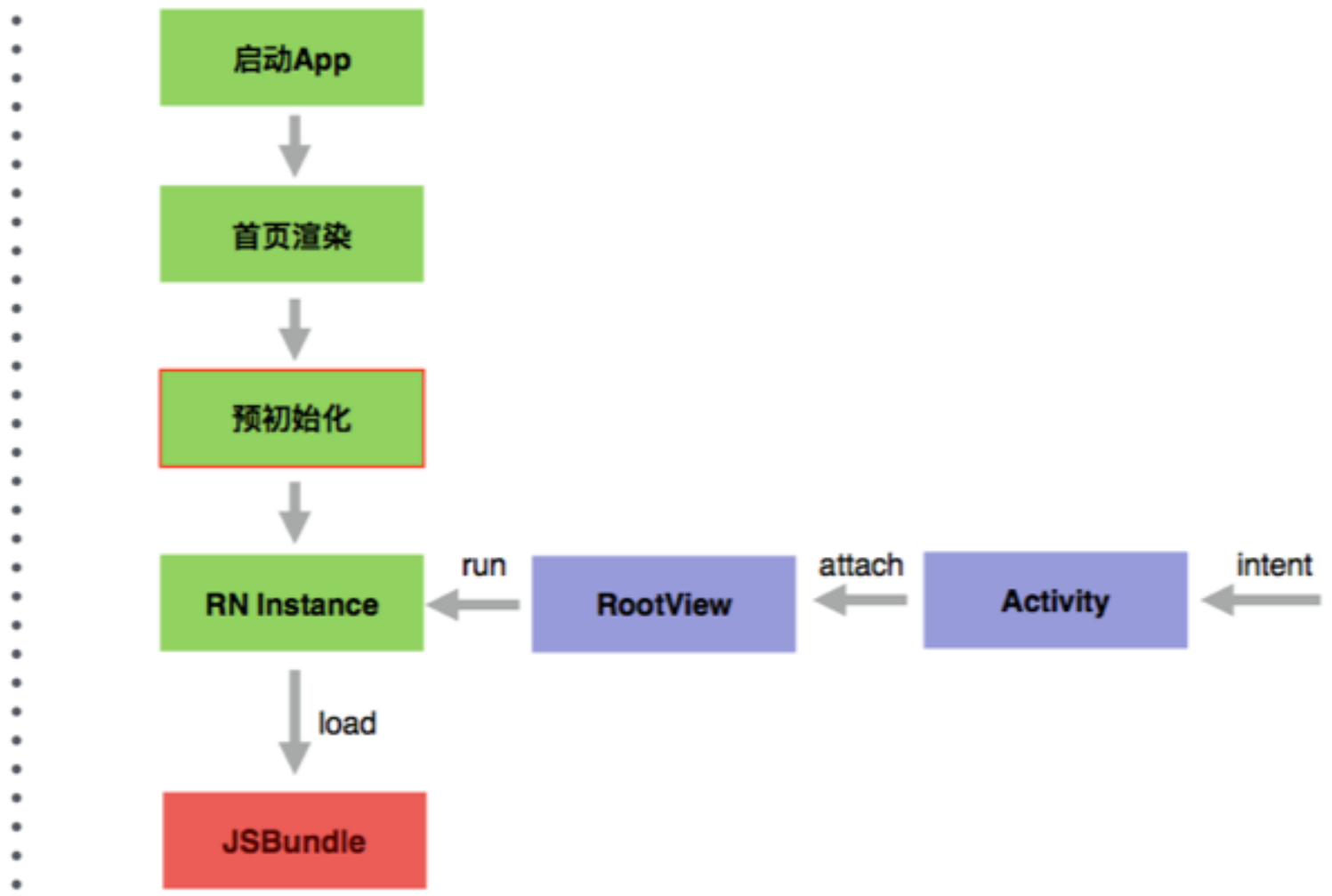
启动速度优化


100%
75%
50%
25%
0%
Native Render
JS Render
Fetch
Instance Init
Native init
总初始化时间：1.6s
(Android平均)

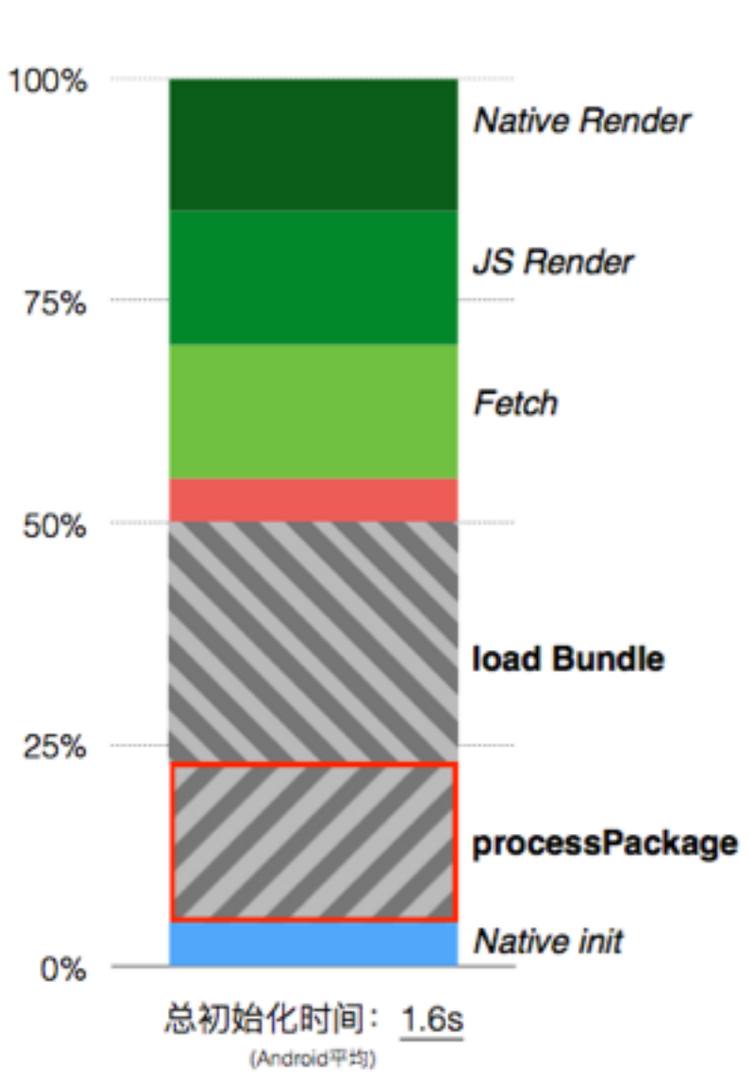

启动App
首页渲染
预初始化
RN Instance
run
RootView
attach
Activity
intent
load
JSBundle

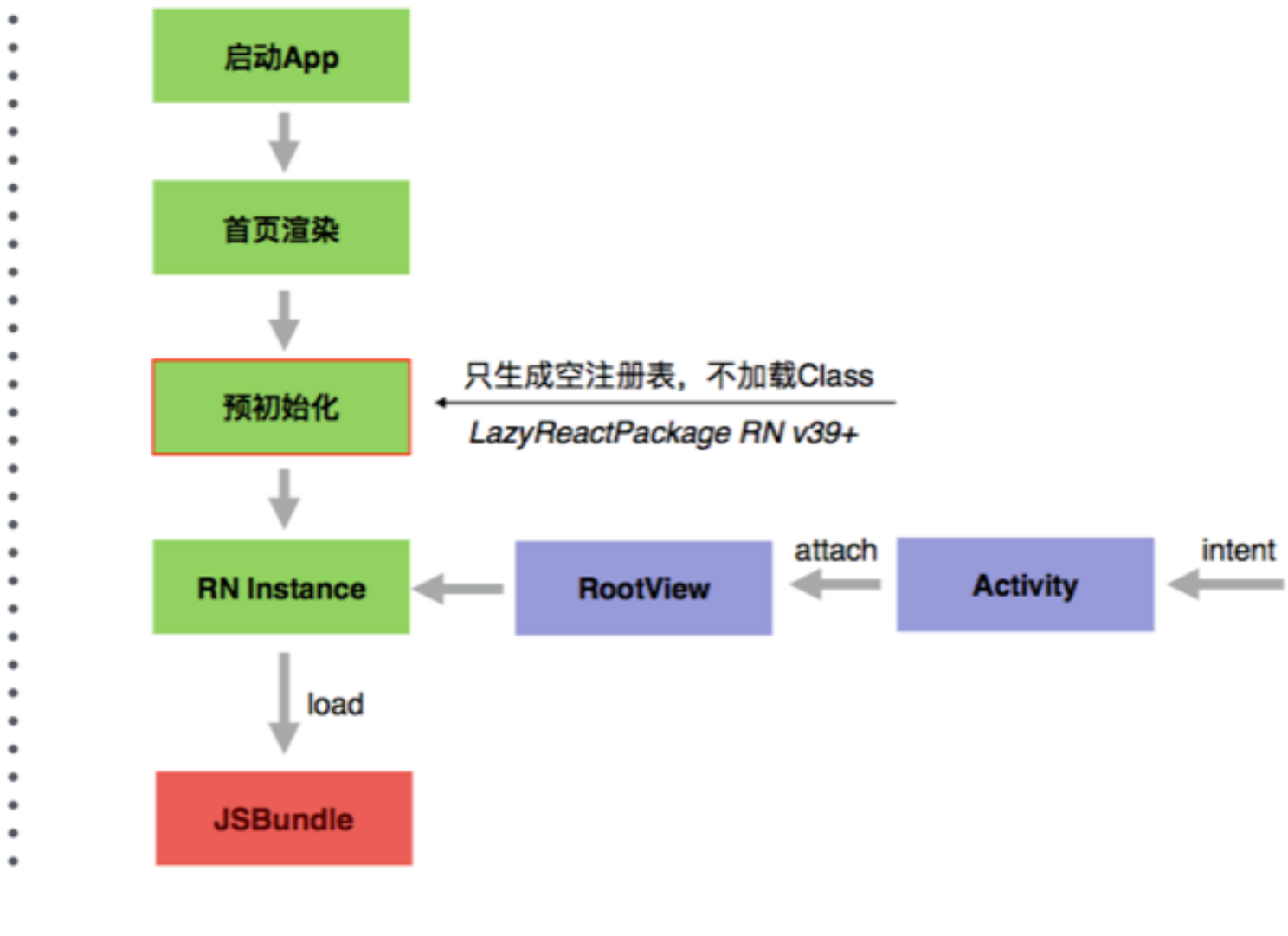
启动速度优化 – 预初始化


100%
75%
50%
25%
0%
Native Render
JS Render
Fetch
Instance Init
Native init
总初始化时间： 1.6s
(Android平均)


启动App
首页渲染
预初始化
RN Instance
run
RootView
attach
Activity
intent
load
JSBundle

启动速度优化 – 延迟生成模块配置表





启动速度优化


23:03
90%
Bai du 百度
关注
视频
娱乐
贴吧
新闻
小说
我现在就叫它神作，是不是有种钦定的感觉?
跑男5杨洋替代鹿晗，取代杨颖居然瘦成竹竿的她
东方头条
谷歌地球VR震撼来袭 坐在家里环游世界成现实
按住说话 百度一下


Instance A
Bundle A
Instance B
Bundle B
Instance C
Bundle C

启动速度优化 – 单Instance多业务


```
Import App1 from ‘./path/to/app1’;
Import App2 from ‘./path/to/app2’;
Import App3 from ‘./path/to/app3’;

AppRegistry.registerComponent("App1", () => App1)
AppRegistry.registerComponent("App2", () => App2)
AppRegistry.registerComponent("App3", () => App3)
```

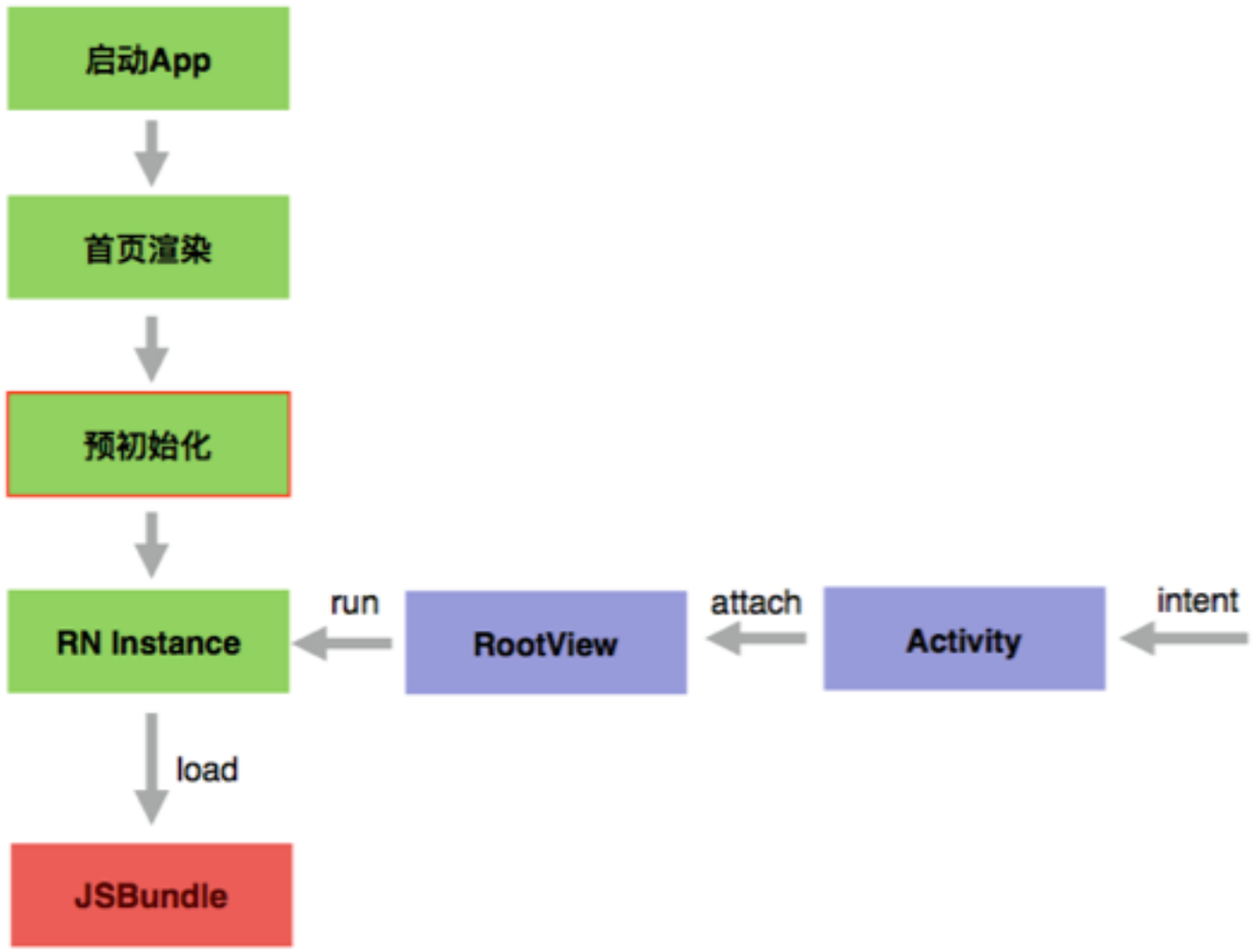
RCTRootView A

RCTRootView B → moduleName → Instance → Index Bundle

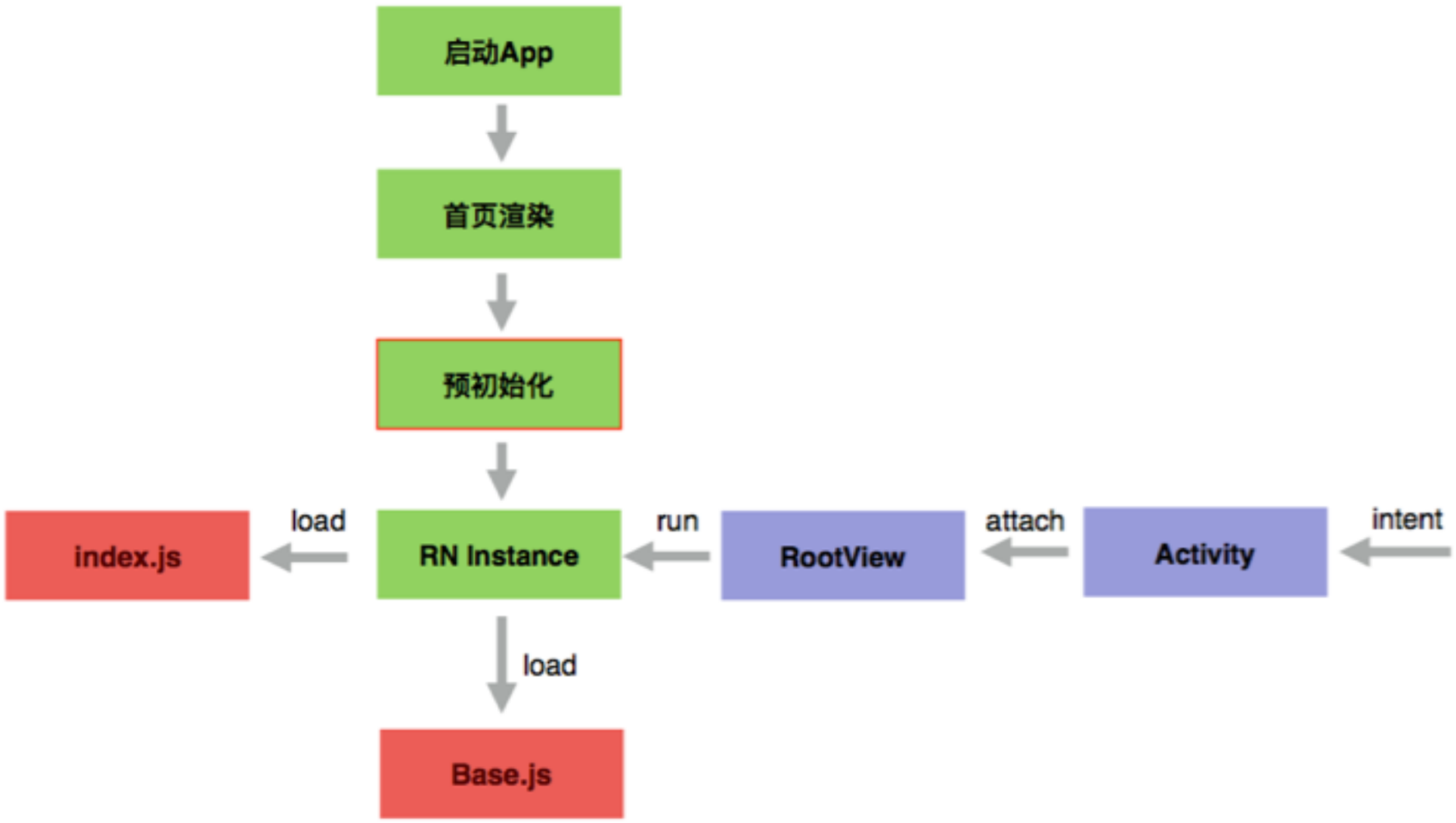
RCTRootView C

- 减轻预加载开销
  - 集中打包
  - component方式加载业务
  - 无法使用global变量

启动速度优化 – 按需加载


启动App
首页渲染
预初始化
RN Instance
run
RootView
attach
Activity
intent
load
JSBundle

启动速度优化 – 按需加载

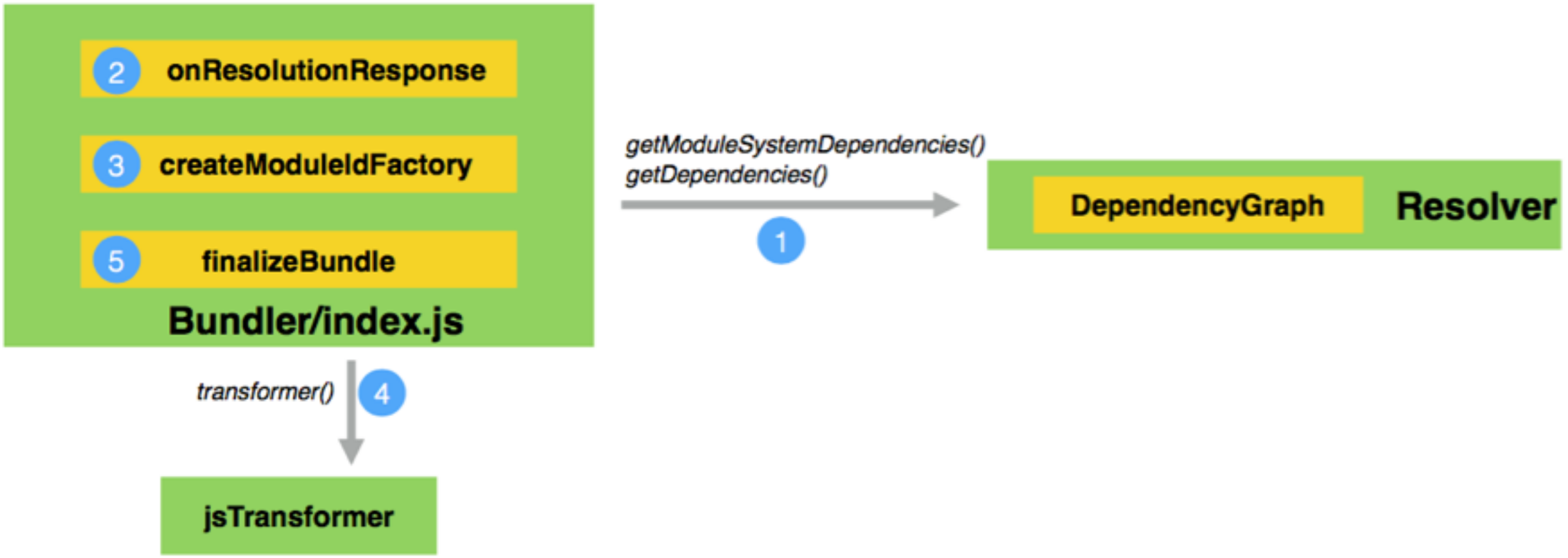

启动App
首页渲染
预初始化
index.js
load
RN Instance
run
RootView
attach
Activity
intent
load
Base.js

启动速度优化 - 按需加载

```
/*系统模块，全局定义，polyfill,babel*/
/*prelude.js*/!function(_){_.__DEV__=!1,_.__BUNDLE_START_TIME__=Date.now()}("undefined"!=typeof global?global:"undefined"
/*require.js*/!function(r){"use strict";function t(r,t){r in u||(u[r]={factory:t,hasError:!1,isInitialized:!1,exports:voi
/*polyfills.js*/!function(n){Object.assign=function(n,e){for(var f=1;f<arguments.length;f++){var l=arguments[f];if(null!=
/*console.js*/!function(n){function e(n){function e(e){return function(){var t;t=1===arguments.length&&"string"==typeof a
/*error-guard.js*/!function(r){function n(){var n=function(r){throw r};r.ErrorUtils.setGlobalHandler(n)}var l={inGuard:0
/*Number.es6.js*/!function(e){void 0===Number.EPSILON&&Object.defineProperty(Number,"EPSILON",{value:Math.pow(2,-52)}),vo
/*String.prototype.es6.js*/!function(t){String.prototype.startsWith||(String.prototype.startsWith=function(t){"use strict
/*Array.prototype.es6.js*/!function(e){function r(e,r){if(null==this)throw new TypeError("Array.prototype.findIndex calle
/*Array.es6.js*/!function(n){Array.from||(Array.from=function(n){if(null==n)throw new TypeError("Object is null or undefi
/*Object.es7.js*/!function(e){!function(){var e=Object.prototype.hasOwnProperty;"function"!=typeof Object.entries&&(Objec
/*babelHelpers.js*/!function(e){var r=e.babelHelpers={};r.createRawReactElement=function(){var e="function"==typeof Symbo
/*入口模块*/
__d(0/*index.js*/,function(e,t,r,n){var i=t(12),u=(babelHelpers.interopRequireDefault(i),t(41)),l=t(377),o=babelHelpers.i
/*react-native基础模块*/
__d(12 /*./node_modules/react/react.js*/,function(t,s,c,e){"use strict";c.exports=s(13)});
__d(13 /*./node_modules/react/lib/React.js*/,function(e,t,r,n){"use strict";var a=t(14),o=t(15),c=t(27),l=t(30),s=t(31),i
__d(14 /*./node_modules/object-assign/index.js*/,function(r,e,t,n){"use strict";function o(r){if(null===r||void 0===r)thr
__d(15 /*./node_modules/react/lib/ReactChildren.js*/,function(t,n,u,e){"use strict";function r(t){return(""+t).replace(k,
__d(16 /*./node_modules/react/lib/PooledClass.js*/,function(n,o,e,t){"use strict";var r=o(17),i=(o(18),function(n){var o=
...
__d(177 /*./node_modules/react-native/Libraries/JavaScriptAppEngine/Initialization/InitializeJavaScriptAppEngine.js*/,fun
...
__d(375,function(r,t,e,n){"use strict";var i=t(17),a=t(15),o=t(19),u=t(22),f=(t(18),t(21),{create:function(r){if("object"
__d(376,function(r,o,n,i){"use strict";function t(r){return Array.isArray(r)?r.concat():r&&"object"==typeof r?e(new r.con
/*业务模块*/
__d(377,function(e,t,r,l){Object.defineProperty(l,"__esModule",{value:!0});var a=t(12),n=babelHelpers.interopRequireDefau
__d(378,function(e,t,d,n){"use strict";function o(e){return e&&e.__esModule?e:{default:e}}n.__esModule=!0,n.connect=n.Pro
__d(379,function(e,t,r,o){"use strict";function n(e){return e&&e.__esModule?e:{default:e}}function i(e,t){if(!(e instance
__d(380,function(e,s,u,i){"use strict";i.__esModule=!0;var p=s(12);i.default=p.PropTypes.shape({subscribe:p.PropTypes.fun
__d(381,function(o,e,n,r){"use strict";function t(o){"undefined"!=typeof console&&"function"==typeof console.error&&conso
__d(382,function(t,e,r,o){"use strict";function s(t){return t&&t.__esModule?t:{default:t}}function n(t,e){if(!(t instance
__d(383,function(e,t,r,n){"use strict";function u(e,t){if(e===t)return!0;var r=Object.keys(e),n=Object.keys(t);if(r.lengt
__d(384,function(n,t,r,u){"use strict";function e(n){return function(t){return(0,i.bindActionCreators)(n,t)}}u.__esModule
__d(385,function(e,t,d,o){"use strict";function r(e){return e&&e.__esModule?e:{default:e}}o.__esModule=!0,o.compose=o.app
__d(386,function(e,t,n,r){"use strict";function o(e){return e&&e.__esModule?e:{default:e}}function i(e,t,n){function r(){
__d(387,function(t,r,n,o){function c(t){if(!a(t)||e(t)!=i)return!1;var r=u(t);if(null===r)return!0;var n=s.call(r,"constr
__d(388,function(n,t,o,i){function e(n){return null==n?void 0===n?f:d:(n=Object(n),g&&g in n?r(n):u(n))}var c=t(389),r=t(
/*执行系统模块和运行系统模块*/
;require(177);/*InitializeJavaScriptAppEngine*/
;require(0);/*入口模块*/
```

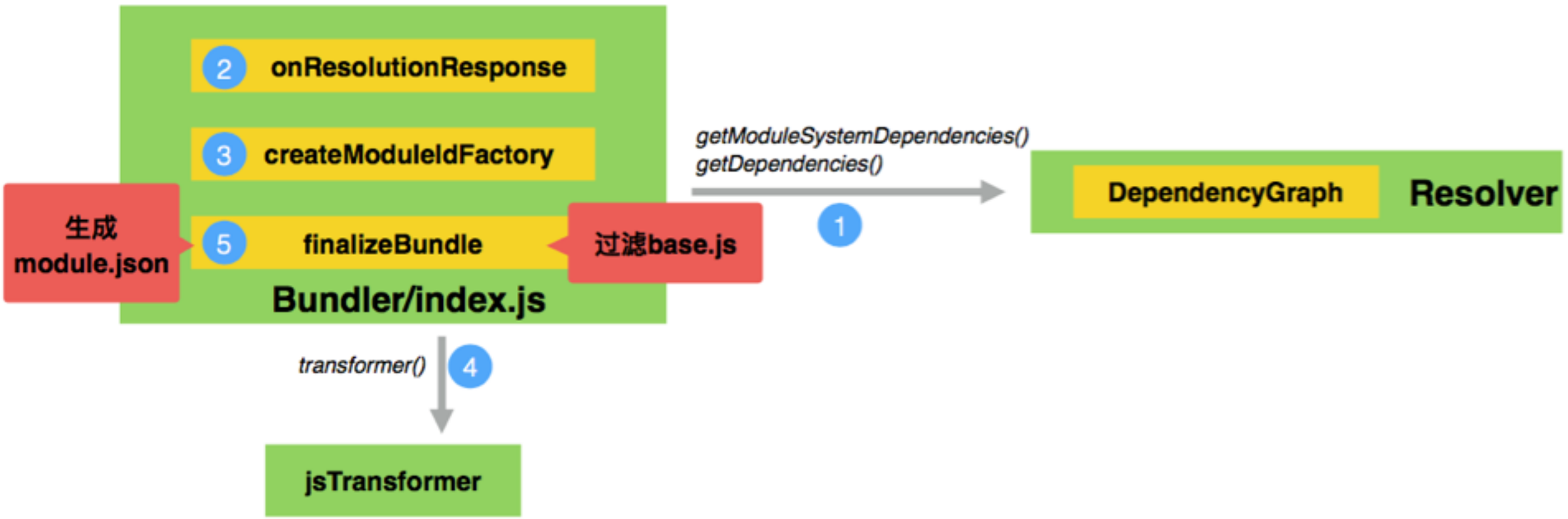
- bundle.js
  - 模块化加载每个module
  - 每行一个module
  - moduleId是自增数字

启动速度优化 – 按需加载



react-packger打包流程

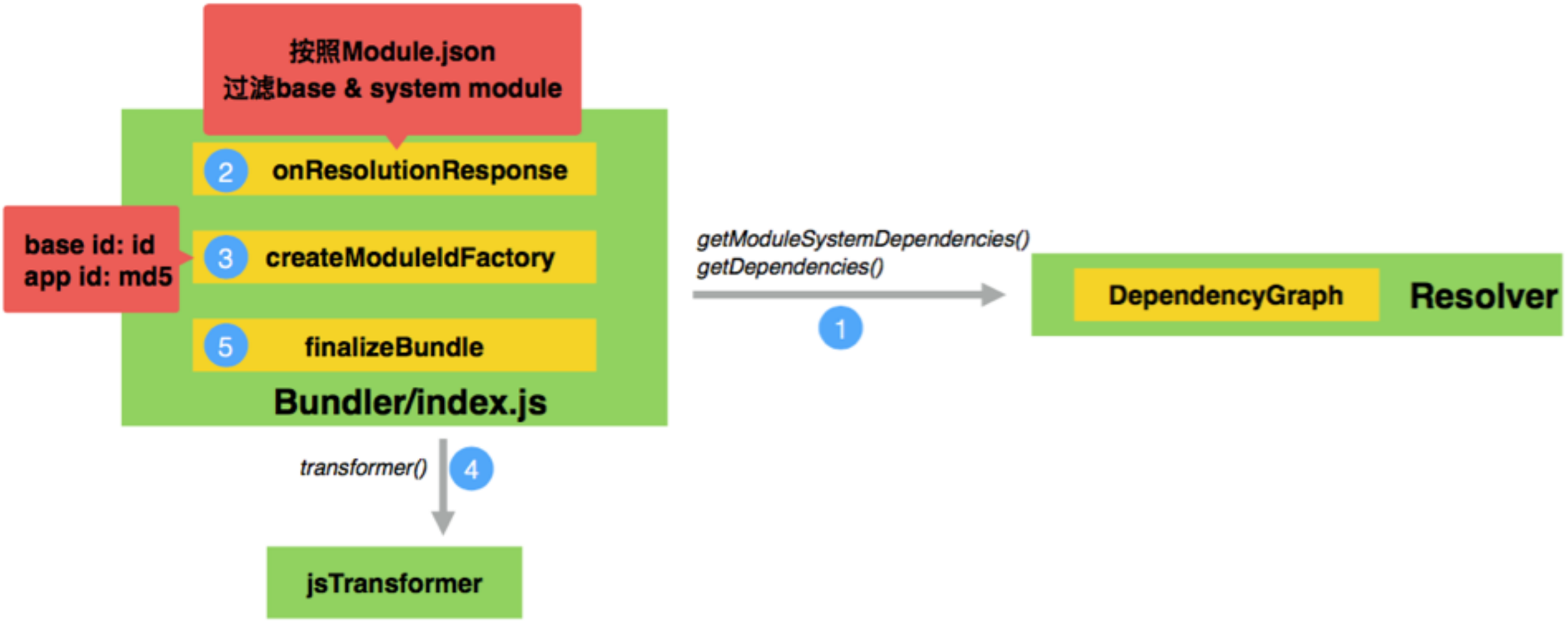
启动速度优化 – 按需加载



打包 base.bundle（修改后）

react-native —entry-file base.js —bundle-output ./base.bundle **—module-ouput ./module.json**

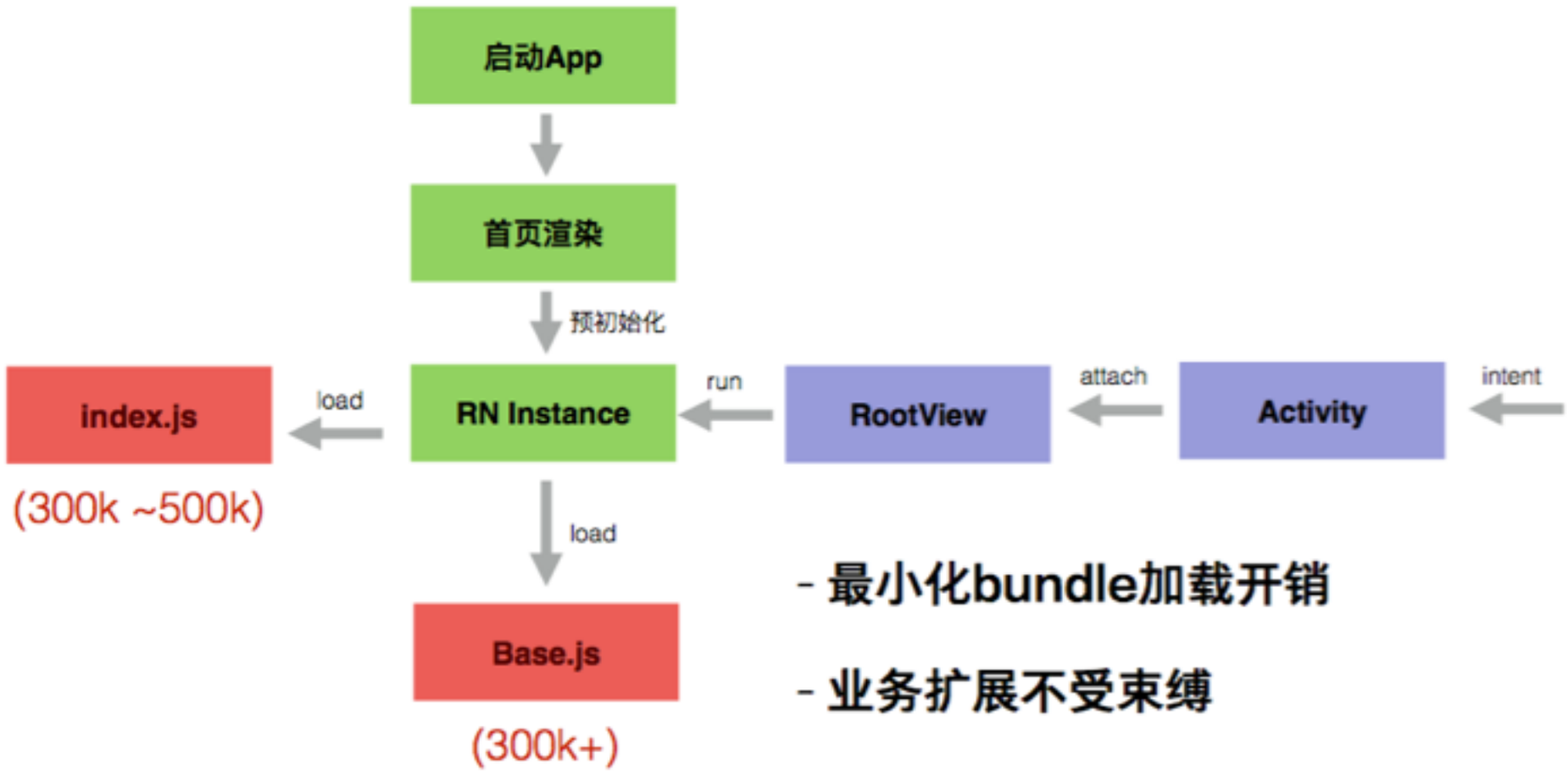
启动速度优化 – 按需加载



打包 app.bundle（修改后）

react-native —entry-file index.js —bundle-output ./index.bundle **—module-input ./module.json**

启动速度优化 – 按需加载



- 最小化bundle加载开销
- 业务扩展不受束缚

跨端方案对比

	优点/特点	缺点	适用场景
APK插件	框架成本低 性能、体验一步到位	开发成本高 iOS依赖发版	对性能和体验要求高 同时**相对稳定**
Hybrid	移植成本极低 通过端能力优化特定交互 业务随时发版	交互相对简陋 端能力依赖发版	**快速迭代 或 短期运营** 需要接入第三方 支持**自定义内容展示**
React Native	流畅度近似原生应用 开发成本相对Native较低 业务随时发版	有学习和开发成本 要求稳定性和平台化程度	**兼顾快速迭代和体验** **平台级app**

SUMMARY

- React Native 已能在超大规模的app中使用
  - **在发版周期、开发成本和使用体验之间找平衡**
- 使用React Native是个系统性工程
  - **做到有机融入App**
  - **确保全流程控制和质量保证**
- 几个作用较大的性能优化点
  - **Listview/动画/启动速度**