百度前端性能优化基础设施介绍

牛尧 2013.08.21

About me

http://weibo.com/paleswd

不讲

- Best Practices：Yahoo, Google, etc.
- 全流程性能优化
- 针对一个产品的具体优化

从**通用技术**的角度分享百度的**实践**

目录

- 基础服务概览
- 性能数据服务
- 性能分析工具
- 优化实践
- 思考与展望

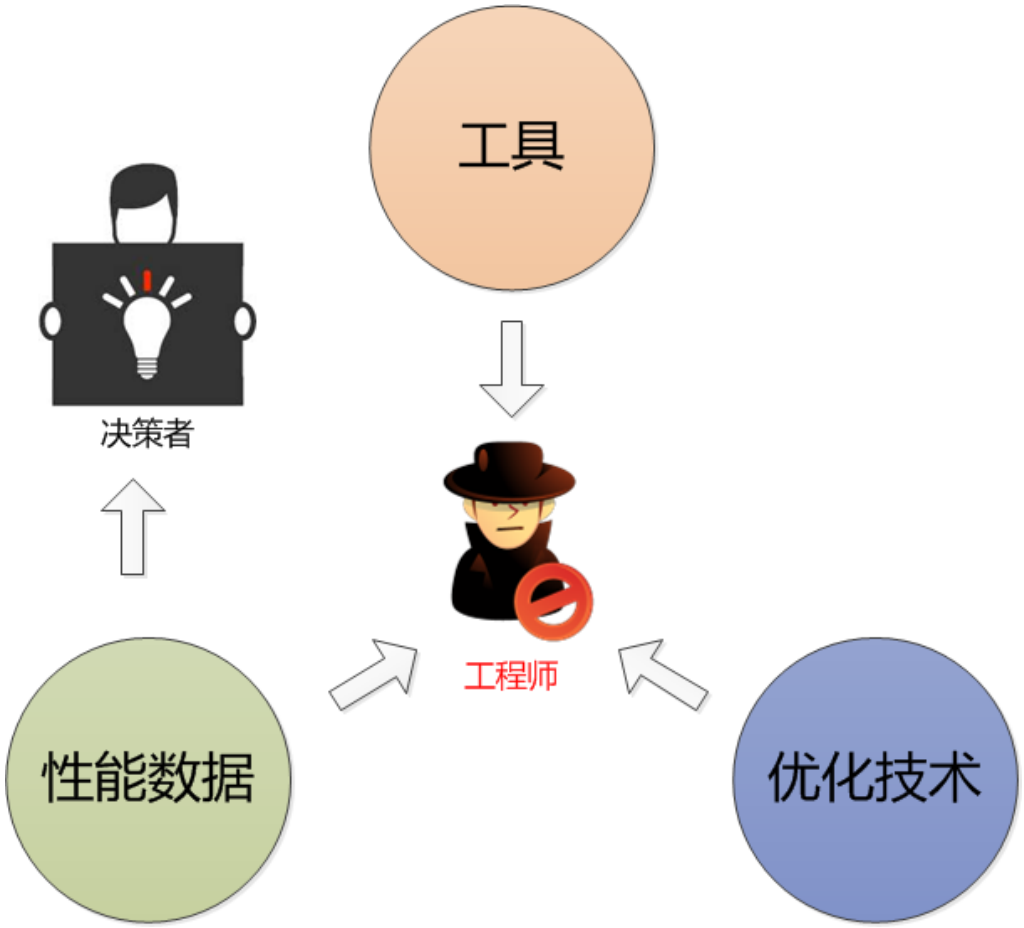
序：基础服务概览

定位

对应用层性能问题提供全方位支持

涵盖发现、分析、解决、效果评估四个阶段
支持Web、Native展示层性能优化

服务介绍



一 性能数据服务

If you can’t measure it, you can't manage it.

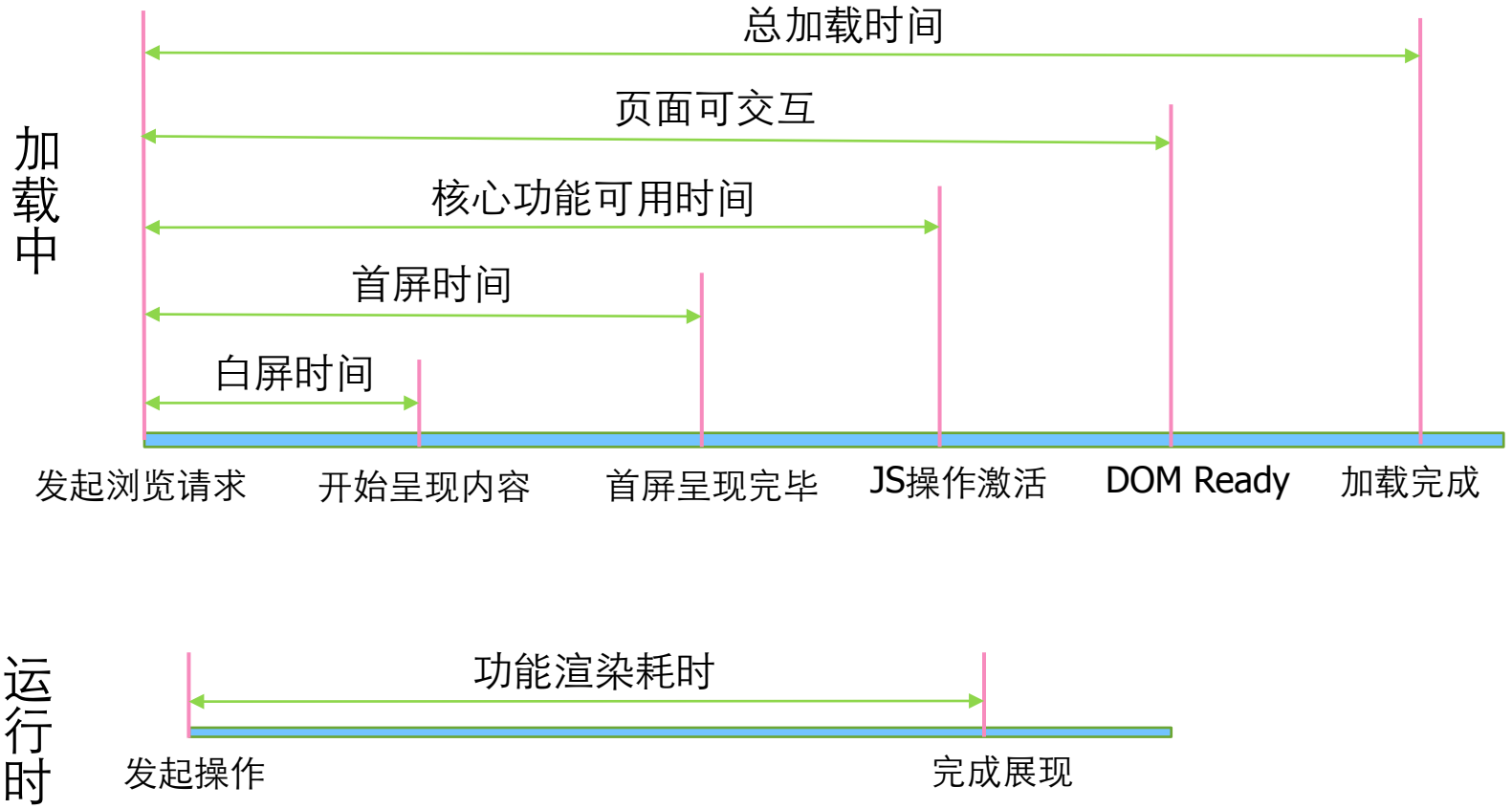
- Peter Drucker

但是，我们需要衡量什么?

Onload?
首屏?
Dom Ready?
Timing?
...

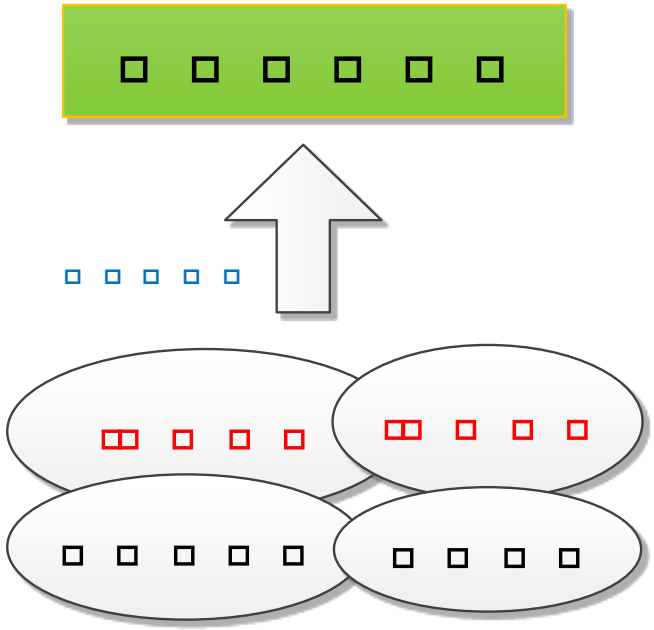
核心指标

- 衡量用户真实体验



核心指标

- 不同指标对用户体验的影响权重不同
  - 高：白屏、首屏
  - 中：核心功能可用
  - 低：Dom Ready、onload



还需要什么数据?

- 性能分析元数据
  - 终端环境
    - 浏览器
    - 网络
    - 操作系统
    - 机型
  - Navigation Timing 接口
    - 网络耗时分布
    - 移动设备支持很差
  - 页面复杂度
    - Dom、img、js、css、html
  - 竞品数据

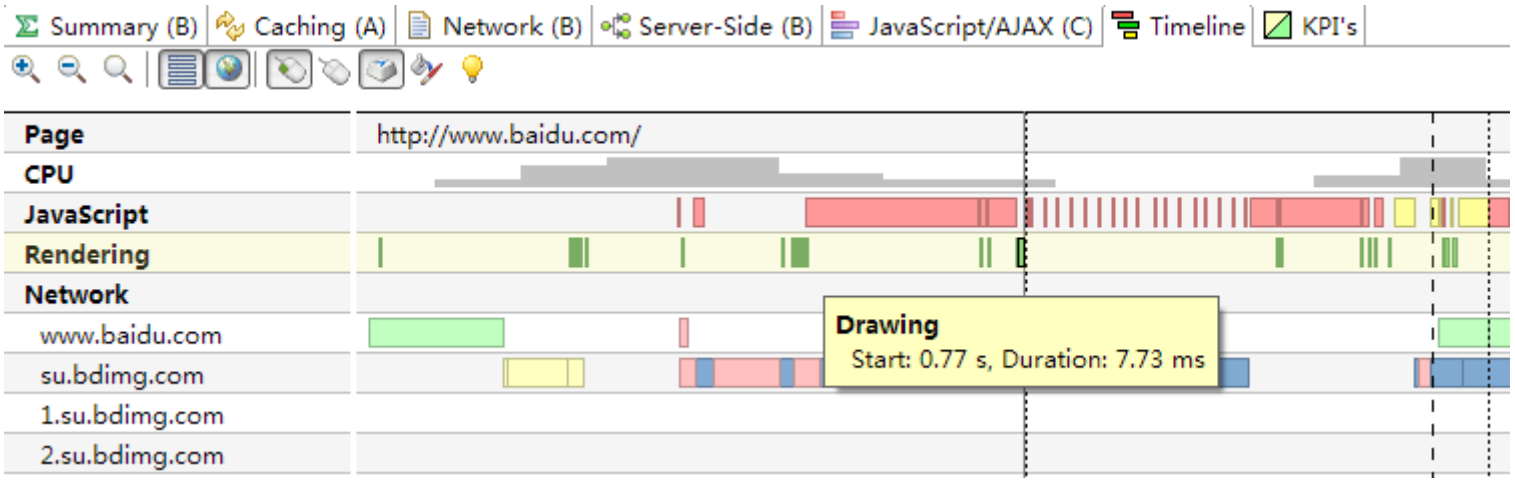
数据采集

- 原则
  - 相关时间很难做到绝对精确
  - 浏览器会让未来更美好
- 全量 or 采样?
  - 建议采样
  - 保证20万以上样本
- 采集脚本是否影响性能?
  - 页面注入的JS量：1.5K~2K
  - 动态加载上报数据的代码
  - 基本无影响

数据采集——难点问题

- 时间原点
  - 点击时用cookie记录
  - window.performance.timing.navigationStart
- 白屏时间
  - Body区域开始打点模拟
- 首屏
  - 假定图片在 onload 之后很快就完成渲染
  - 预估首屏节点位置进行标记
  - 以标记之前图片全部加载完毕的时间模拟

数据采集——IE下的问题



即便JS后置，IE8下仍会阻塞渲染

数据分析——加工成果

- 均值
  - 去掉长尾
- 分布区间
  - 精确到每50ms的用户比例
- 多维细分数据
  - 用户/浏览器/网络状况/机型
- 慢速用户的构成

数据分析——计算平台

- 离线汇总数据
  - 分析平台
    - 单一网站千万以下样本：mysql + 分析脚本
    - 海量数据：Hadoop + Hive
- 在线实时监控
  - 产出内容
    - 10S粒度的实时性能数据
    - Server端渲染耗时预警
  - 流式计算：Storm

数据呈现

如何评价响应时间的优劣?

There are no industry standards !

我们的经验：

类别	快	较快	慢	很慢	指标示例
时间敏感	<1s	1-1.5s	1.5-2.5s	>2.5s	首屏、白屏
时间不敏感	<2s	2-4s	4-8s	>8s	onload

最佳：白屏<1s，首屏<1.5s，onload<3s

Response Time 参考资料:

1. Best Practices on Web Site Performance Optimization
2. Response Times: The 3 Important Limits
3. Website Response Times
4. Acceptable Response Times
5. How Fast Does a Website Need To Be?
6. System Response Time and User Satisfaction

数据最佳呈现形式

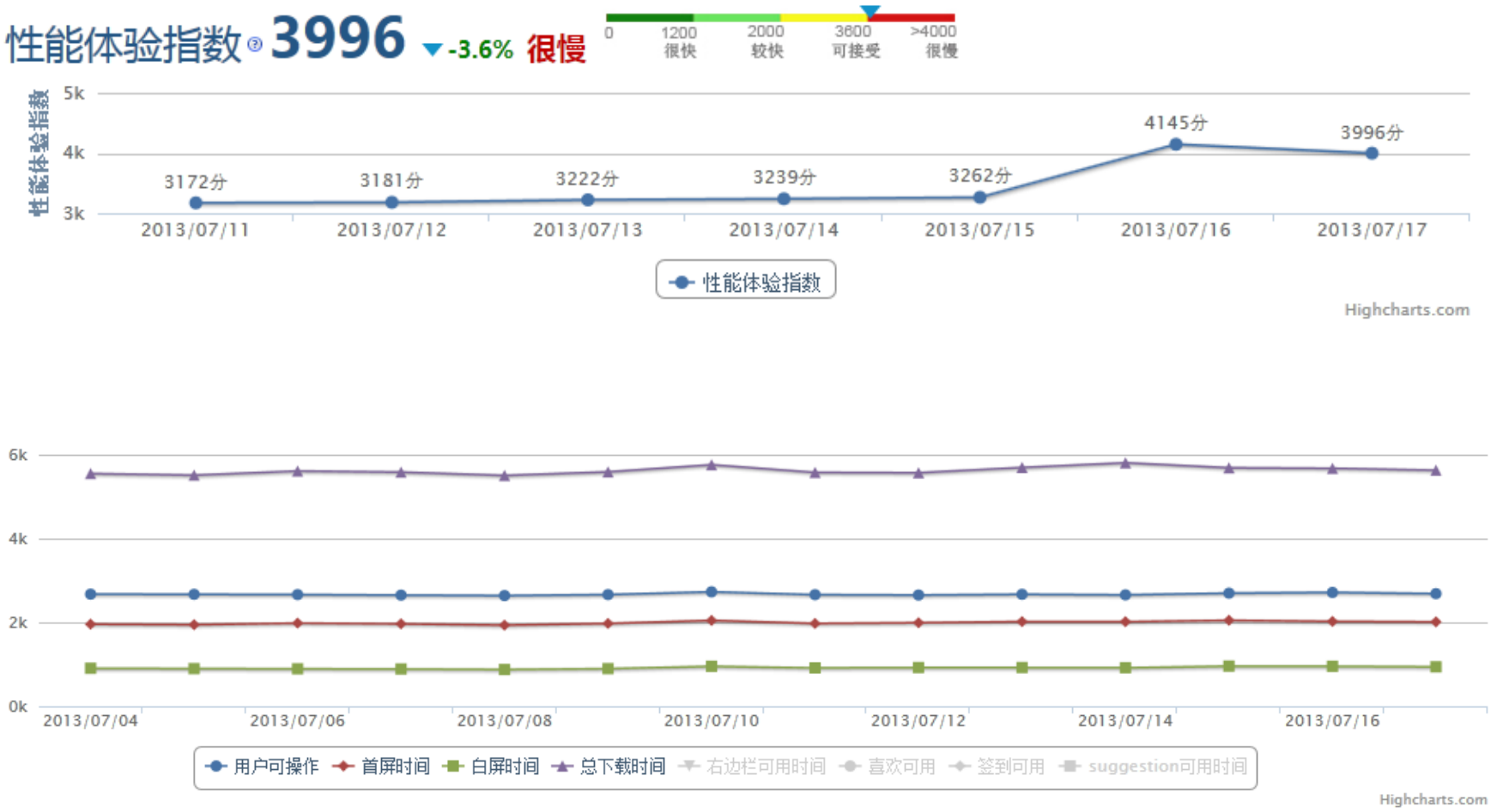
- 汇总Table
- 趋势图
- 分布区间图

性能汇总表

性能概况 昨天 | 昨天或上周

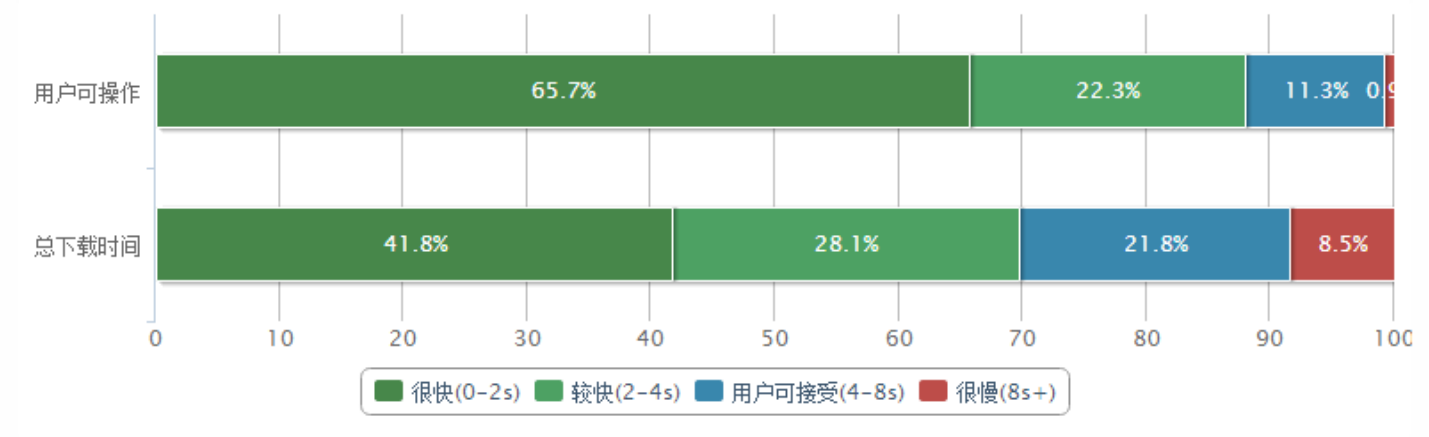
指标项	昨日	前日	相对前日	上周同期	相对上周同期
白屏时间	945ms	932ms	1.39%	1025ms	-7.80%
首屏时间	1985ms	1980ms	0.25%	2093ms	-5.16%
用户可操作	2665ms	2659ms	0.23%	2754ms	-3.23%
总下载时间	5780ms	5830ms	-0.86%	5834ms	-0.93%
喜欢可用	1978ms	1970ms	0.41%	2060ms	-3.98%
签到可用	1953ms	1944ms	0.46%	2033ms	-3.94%
右边栏可用时间	2852ms	2845ms	0.25%	2921ms	-2.36%
suggestion可用时间	2841ms	2833ms	0.28%	2922ms	-2.77%

变化趋势图

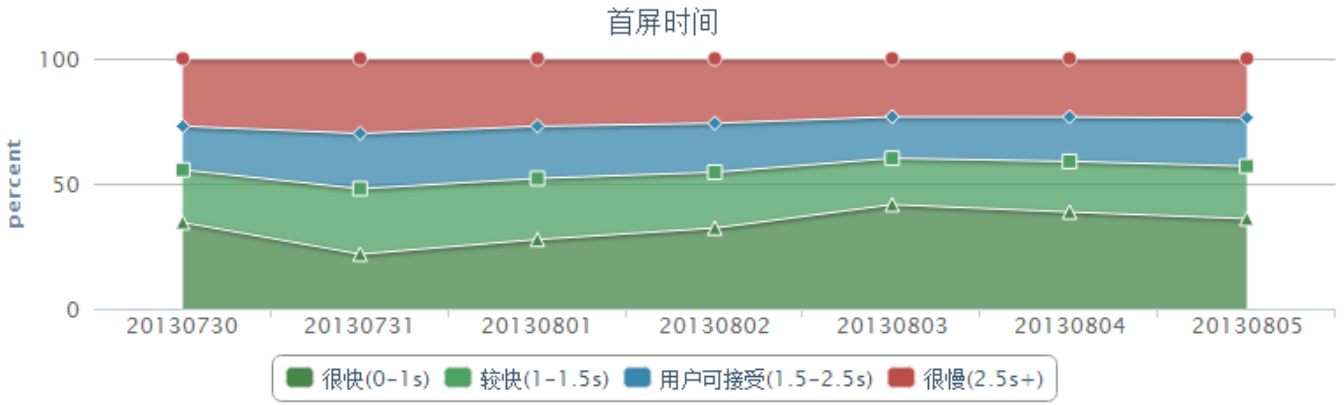

性能体验指数 3996 -3.6% 很慢
0 1200 很快 2000 较快 3600 可接受 >4000 很慢
性能体验指数
5k
4k
3k
3172分
3181分
3222分
3239分
3262分
4145分
3996分
2013/07/11
2013/07/12
2013/07/13
2013/07/14
2013/07/15
2013/07/16
2013/07/17
性能体验指数
Highcharts.com
Time(ms)
6k
4k
2k
0k
2013/07/04
2013/07/06
2013/07/08
2013/07/10
2013/07/12
2013/07/14
2013/07/16
用户可操作
首屏时间
白屏时间
总下载时间
右边栏可用时间
喜欢可用
签到可用
suggestion可用时间
Highcharts.com

分布区间

分布概况



分布趋势



对外数据服务

- 基础性能数据
- 周报&预警
- 竞品对比
- 页面性能排行榜
- 收益评估
- 慢速用户分析

参考资料:

1. Steve Sounders：Moving beyond window.onload()
2. WebPagetest Metrics
3. Performance Monitoring： Synthetic VS RUM
4. Benchmarking the New Front End

二 性能分析工具

工具需要支持哪些浏览器?

国内主流浏览器内核

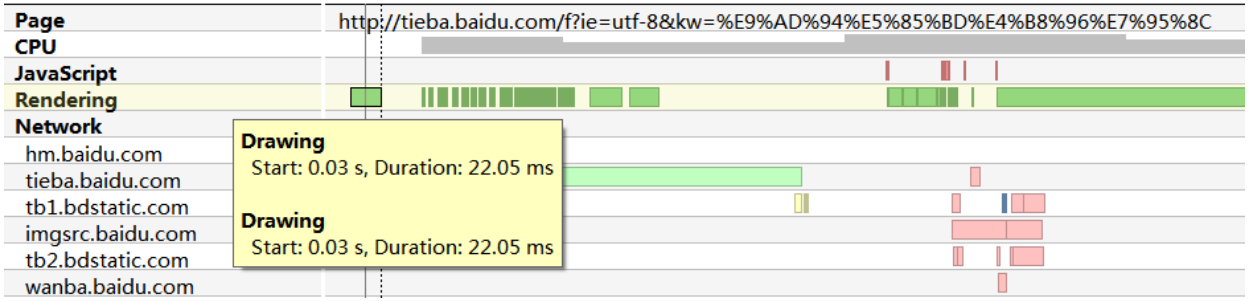
1. IE8
2. Chrome
3. IOS Safari

数据来自：百度统计、CNZZ

强大的 dynatrace

- IE下性能分析的利器
- First Impression Time 不一定准

URL	Rank	First Impression Tim...	OnLoad ...	Total Load ...
http://tieba.baidu.com/2012	E(54)	4563	4165	4250
http://tieba.baidu.com/f?kw=2012	F(42)	35	3009	3034
http://tieba.baidu.com/f?ie=utf-8&kw...	D(69)	53	2717	2740
http://tieba.baidu.com/f?ie=utf-8&kw...	D(65)	42	3406	3406



浏览器内置分析工具

- Yslow & PageSpeed

- Safari Developer Tools
- Chrome Developer Tools

- Google Web Tracing Framework
  - http://google.github.io/tracing-framework/
  - 支持移动、PC，性能分析的神器

现有工具的不足

1. 集成困难
2. 使用及数据解读成本高
3. 可扩展性差

我们遇到的问题

1. 1000+个阿拉丁如何快速评估性能
2. 如何提供性能数据给其它平台使用
3. 渲染瓶颈到底在哪里

所以，我们需要

云端性能评测服务

它可以：

1. 对页面性能进行基本评价
2. 给出浏览器渲染细节及瓶颈

Pagecheck

- 基于规则的站点进行自动评分服务
  - 评分及优化建议
  - 瀑布流
  - 代码覆盖率
- 规则设计
  - 综合PageSpeed 和 Yslow的规则
  - 打分规则调整
- 实现
  - PhantomJS + netsniff.js
  - 代码覆盖率基于 Esprima 实现

Pagecheck

文档 | 检测历史 | 问题反馈 | 登录

Bai du 性能检测

请输入要 Chrome Mozilla/5.0 (X11; Linux i 分析

查看瀑布流图形

没有通过的规则(17)

- 指定正确的Vary头信...
- 合并外部资源
- 检查CSS文件时候包...
- 检查文件中是否包含...
- 启用资源缓存
- 检查JS文件是否被压...
- HTML压缩检测
- 检查页面中img的wid...
- 优化图片
- 减少DNS查询
- 减少DOM大小
- 减少同域名的请求
- 头部只放一个JS文件
- 减少全局变量数
- 减少URL的长度
- 检查CSS文件中是否...
- 减少重复代码的大小

已经通过的规则(15)

检测结果 总分:66 (错误信息)

☹ 指定正确的Vary头信息 (70/100) 检测规则

以下请求的var设置不正确，应该为Accept-Encoding、User-Agent的组合

- http://tieba.baidu.com/tb/static-common/component/commonLogic/com... vary的值应该是Accept-Encoding或User-Agent或两者的组合

☹ 合并外部资源 (0/100) 检测规则

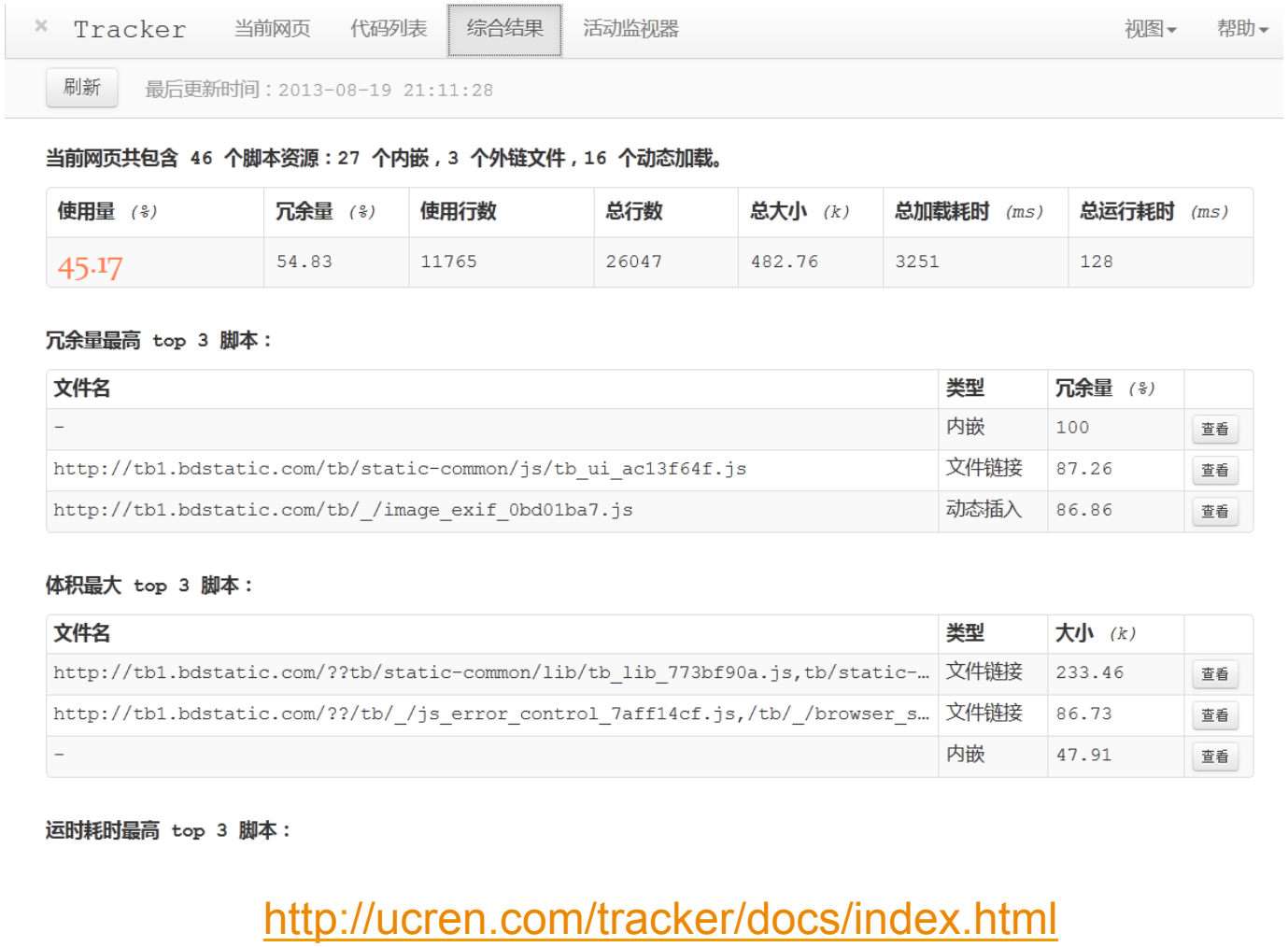
考虑对以下资源进行合并

- http://tb1.bdstatic.com/ 可以合并JS类资源，css图片可使用css sprite合并，css、js可根据需要合并
- http://tb2.bdstatic.com/ 可以合并IMAGE类资源，css图片可使用css sprite合并，css、js可根据需要合并
- http://imgsrc.baidu.com/ 可以合并IMAGE类资源，css图片可使用css sprite合并，css、js可根据需要合并
- http://tieba.baidu.com/ 可以合并IMAGE类资源，css图片可使用css sprite合并，css、js可根据需要合并

☹ 检查CSS文件时候包含未使用的规则 (95/100) 检测规则

CSS文件包含30%以上的规则在页面初始化中没有被使用到

Pagecheck

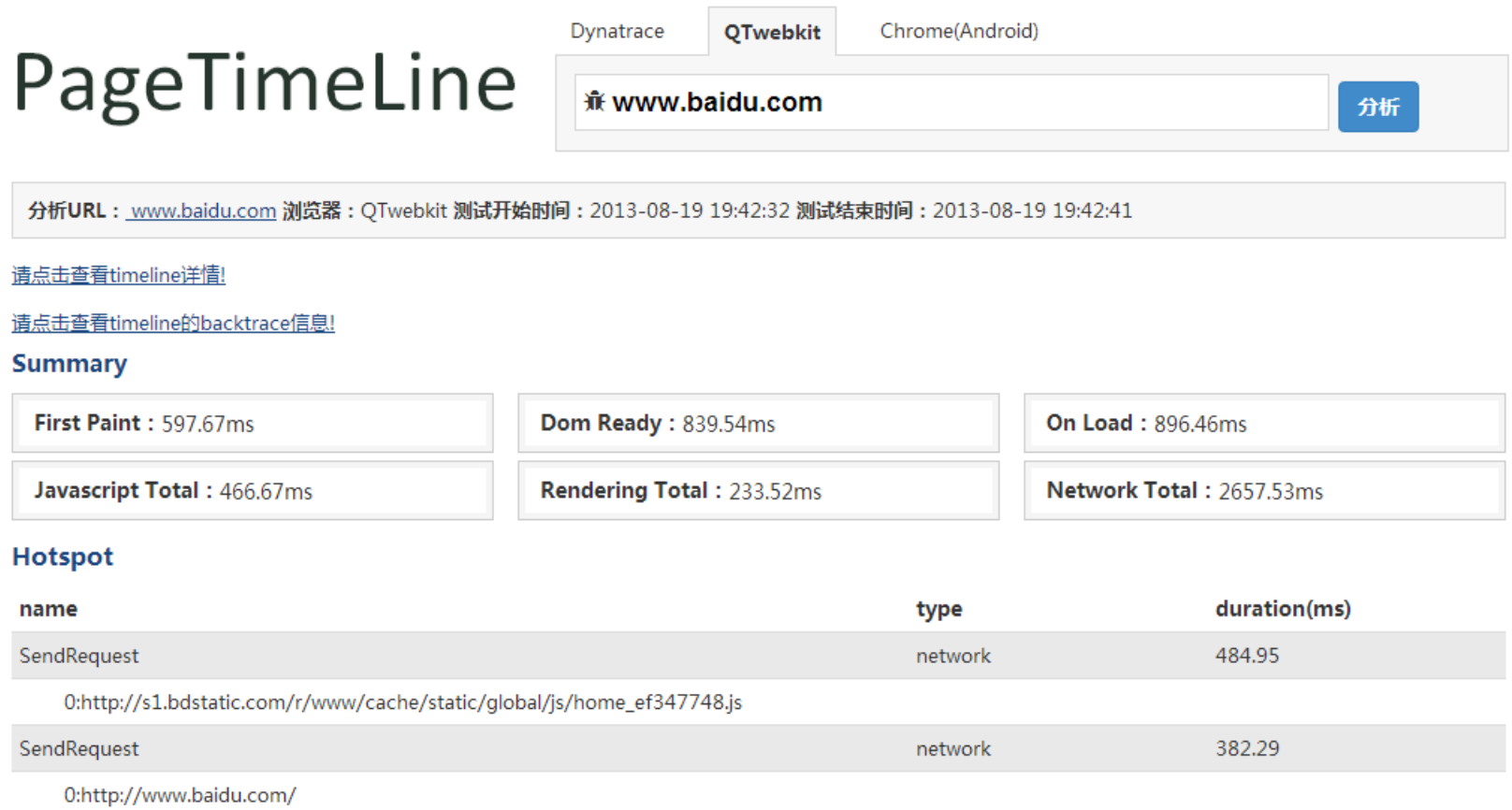


http://ucren.com/tracker/docs/index.html

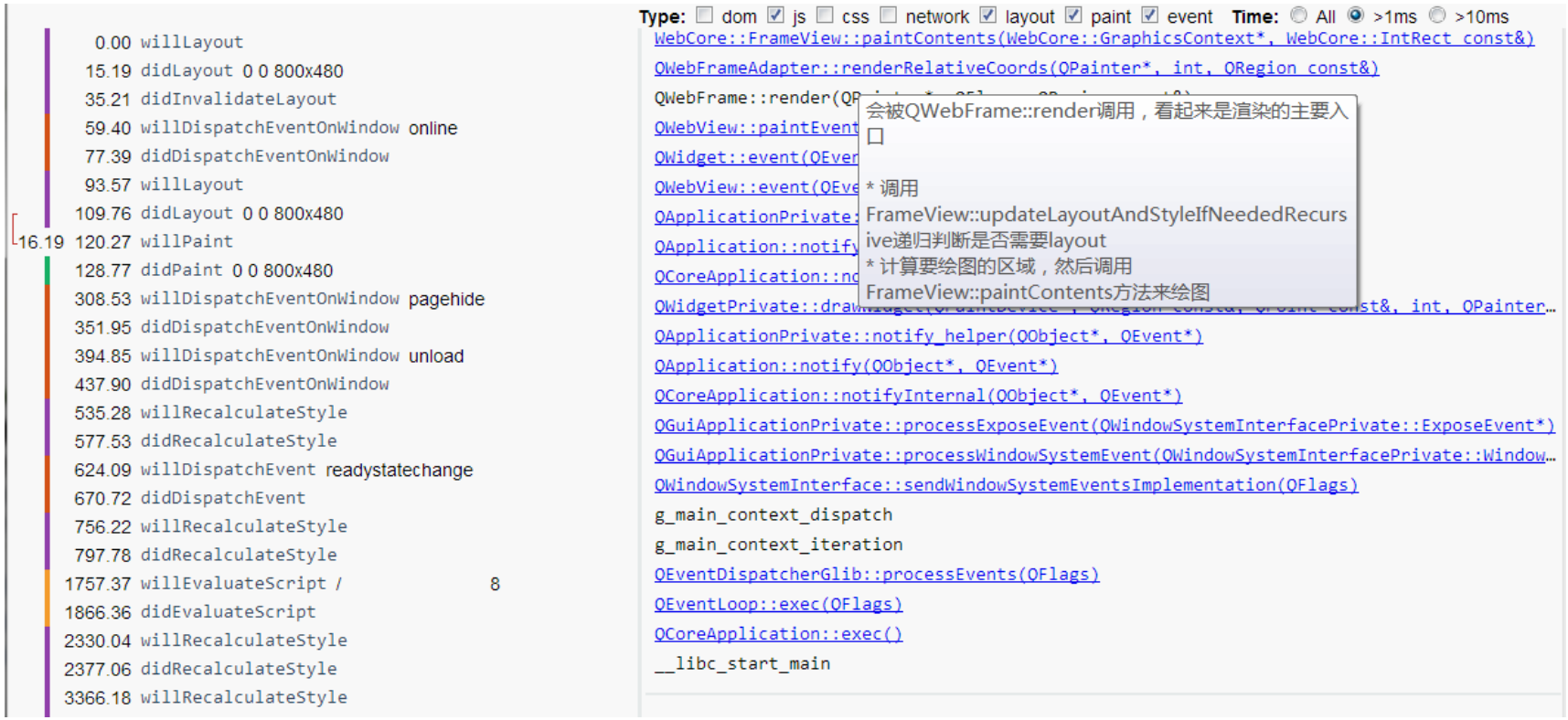
Page Timeline

- 基于webkit内核的渲染分析工具
  - 修改部分代码重新编译
    - QTWebKit
    - Chrome For Android
    - Webview
  - 浏览器渲染动作的详细介绍
    - 近600个webkit函数的功能介绍
- 产出报告
  - Hotspot
  - Timeline

Page Timeline——Hotspot



Page Timeline



让浏览器不再是黑盒

三　优化实践

为什么做性能优化?

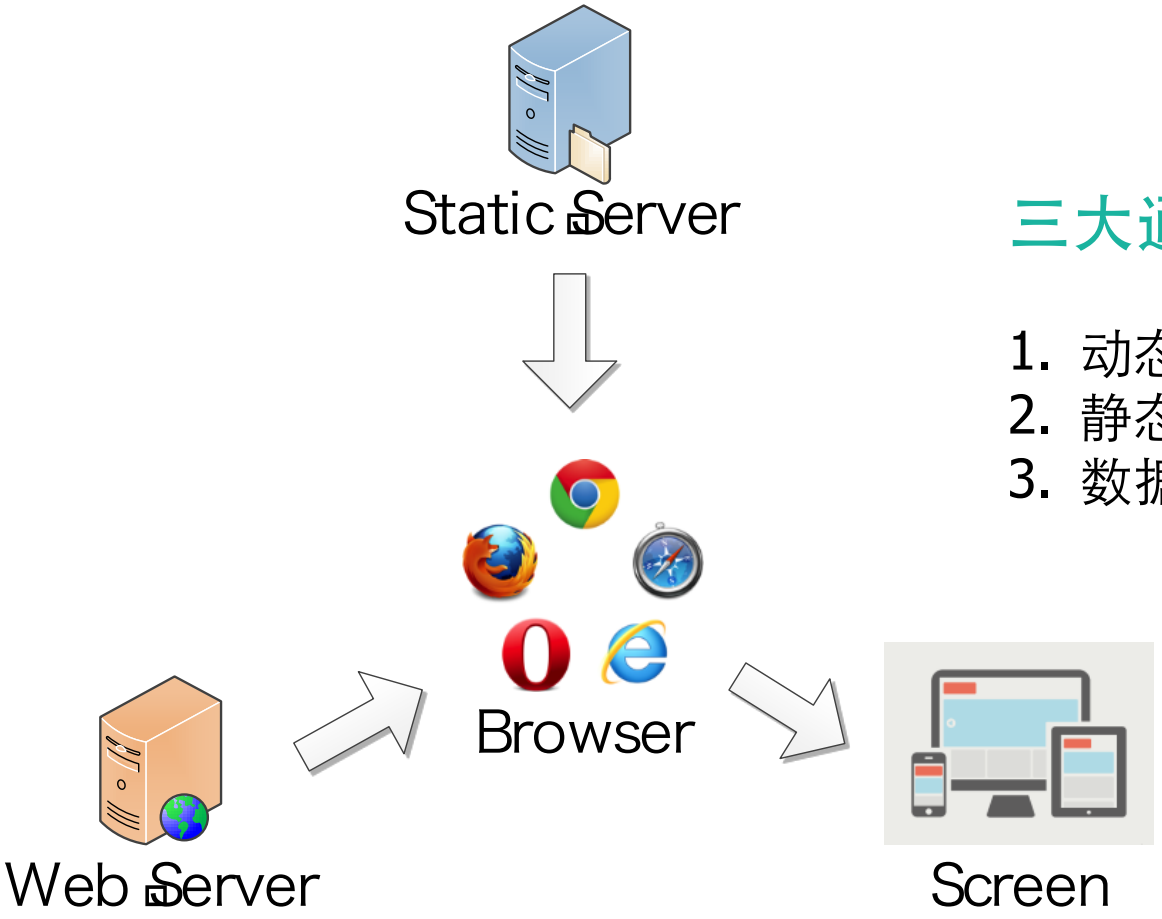
性能引发的后果

- 产品体验
  - 页面不流畅
  - 浏览器假死或闪退
  - 内容可见时间过长
- 资源消耗
  - 网络带宽
  - 终端上的CPU+内存
  - 服务器

目标设定

- 资源维度
  - 带宽节省
  - 机器消耗
- 体验维度
  - 白屏、首屏时间
  - 核心产品功能可用时间
  - 移动端需要额外关注 load 时间

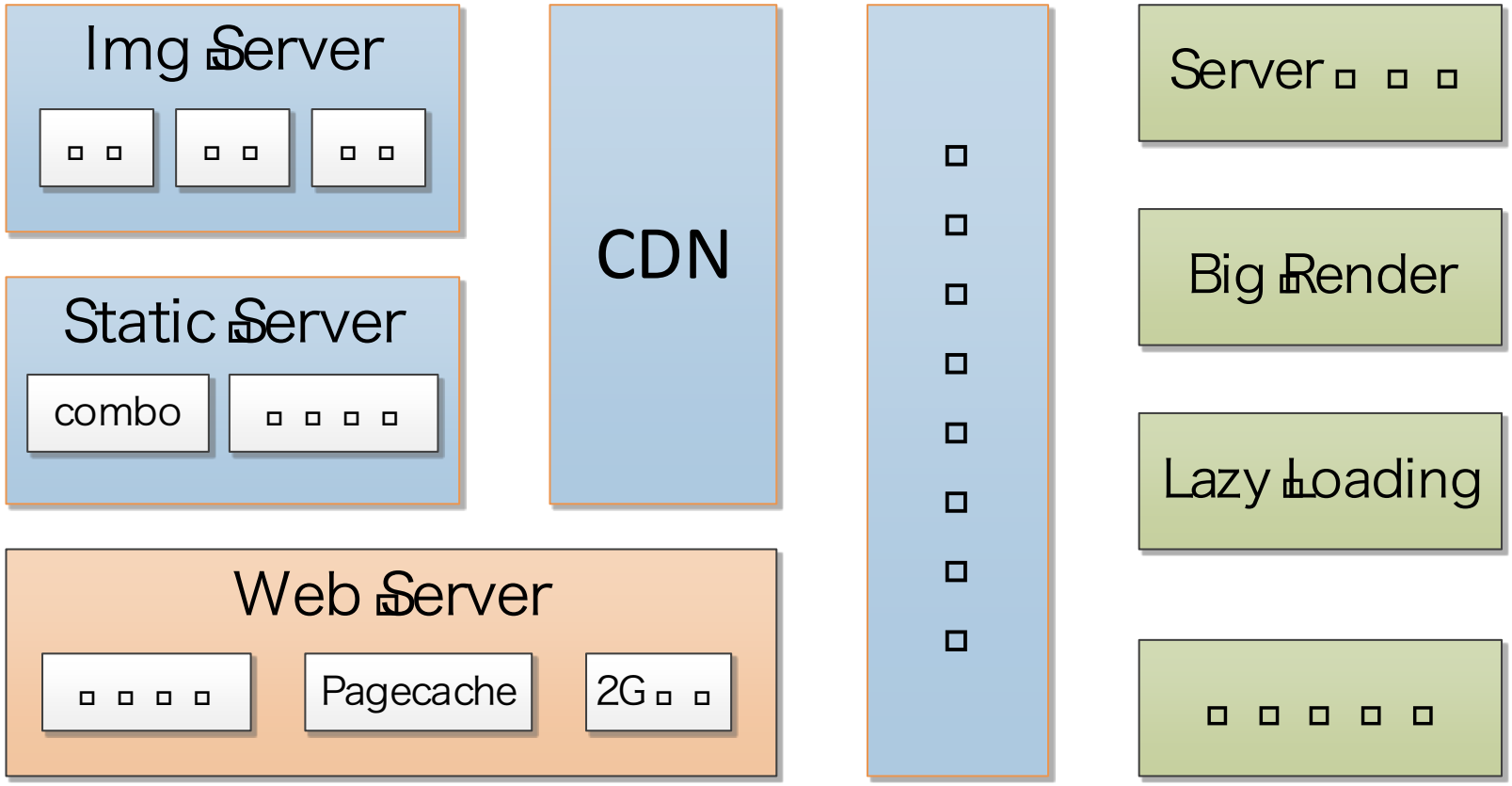
总体思路



三大通道

1. 动态内容 -> 浏览器
2. 静态资源 -> 浏览器
3. 数据或HTML -> 可视窗体

优化基础服务


Img Server
Static Server
combo
CDN
Web Server
Pagecache
2G
Server
Big Render
Lazy Loading

通道一、三

- 业界关注很多
- 有工业化实现

重点分享通道二的解决方案

基于MAP的资源管理方案

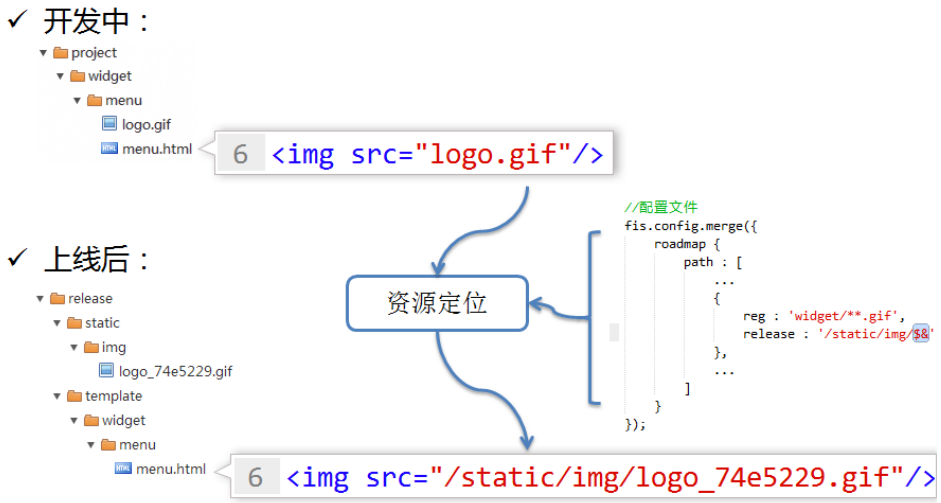
- 静态资源管理
  - Best Practices中有很多条相关法则
  - 是一个苦差事
  - 可以在架构层面解决

- 在开源项目FIS中完整实现

http://fis.baidu.com/

基于MAP的资源管理方案

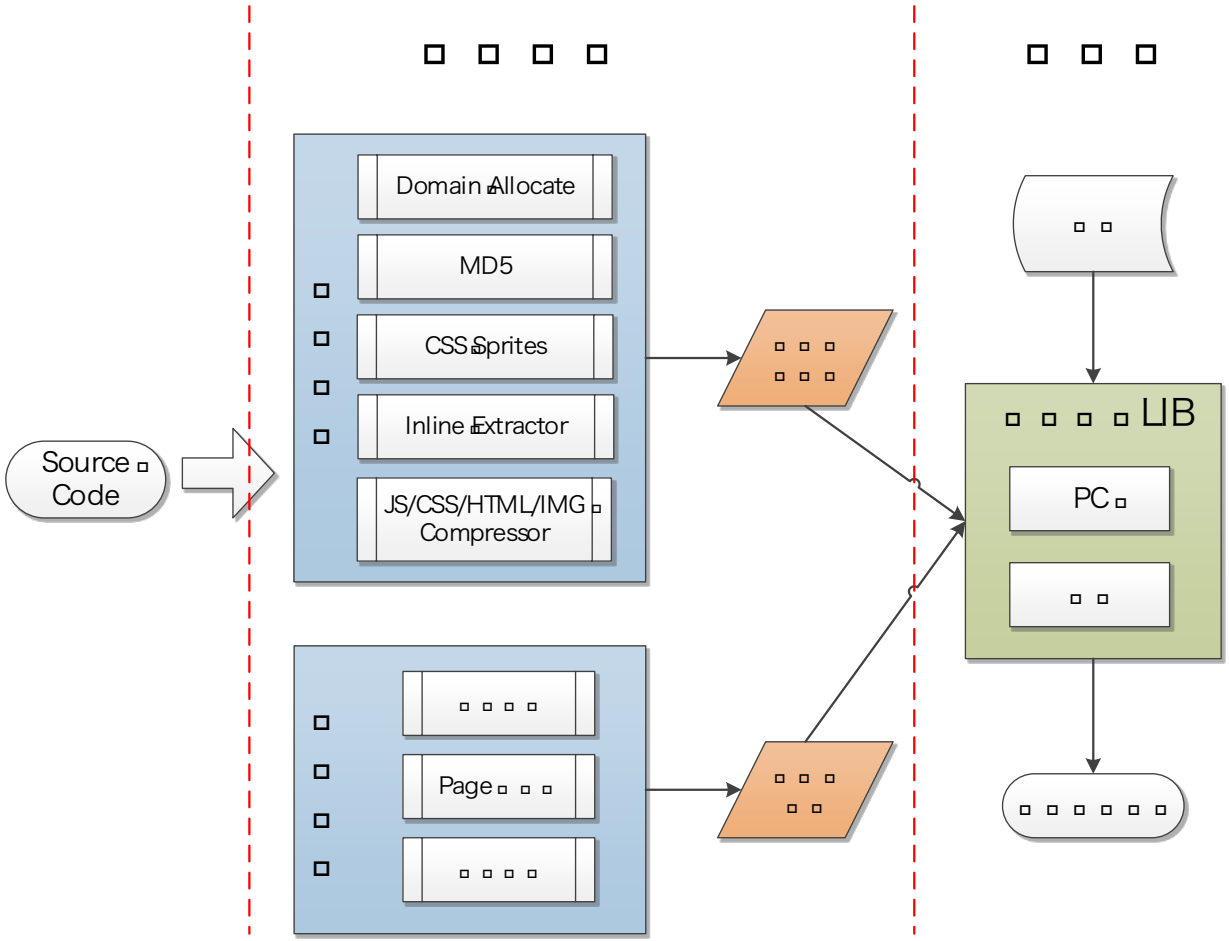
- 扩展三种资源控制能力
  - 内容嵌入
  - 依赖声明
  - 资源定位



资源定位

- 两阶段处理
  - 预编译：产出最优资源个体及资源配置表
  - 运行时：全局最优化资源调度

基于MAP的资源管理方案



详见：基于map.json的前后端架构设计

基于MAP的资源管理方案

工程师只管写码

资源的加载交给类库和工具完成

提供最大的灵活度可扩展性

四　思考与展望

性能和产品指标有何关联?

暂无定论

性能大幅下降，会影响产品指标
但反之并不一定成立

优化，何日到头?

定期性能重构

or

全流程性能准入

准入难点：

1. 产品基准环境
2. 准入模型
3. 环境支持：
   - 持续集成 Jenkins
   - 性能测试环境

1. 性能优化只是体验优化的**一种途径**
2. 很难做到每个用户都很快
3. 产品、浏览器、硬件都在高速迭代

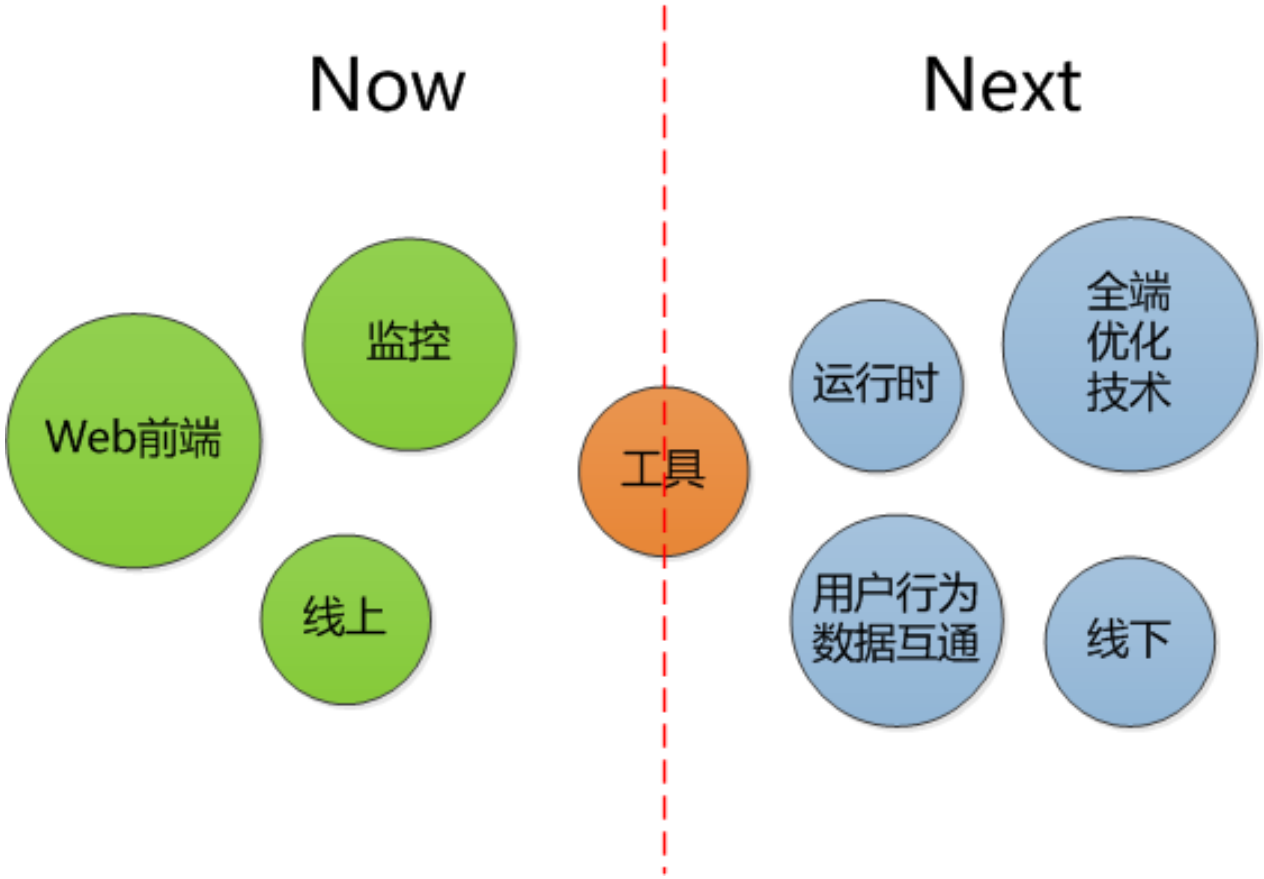
警惕**过早优化**和**过度优化**

移动潮流，如何应对?

现状：

1. 卡顿、闪退是关键词
2. 网络加载、端处理能力是瓶颈
3. 传统Web、Single Page App、Native、WebView 群雄并起

路在何方?



结：要点回顾

总结

- 监控
  - 衡量真实用户体验，关注产品核心功能
  - 趋势图+分布比例
- 工具
  - 云端评测系统
  - 关注浏览器渲染细节
- 优化
  - 资源加载可以在架构层面做到最优
  - 警惕过早优化和过度优化

资源推荐

两个站点

1. Web Performance Today
2. Planet Performance

That’s All.