



百度前端基础数据平台介绍

dp : 数据点评

Shared By 张军

前端有什么数据



我是站长

- 网站的PV、UV
- 广告的点击量
- 网站的主要入口
- 要和哪家网络运营商搞好关系

我是产品经理

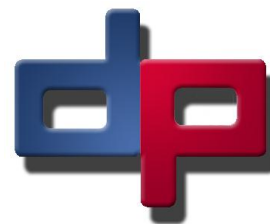
- 某功能使用量
- 针对登陆用户做定制化？
- 区分2G、3G、WIFI用户？
- 用户什么时段访问网页？

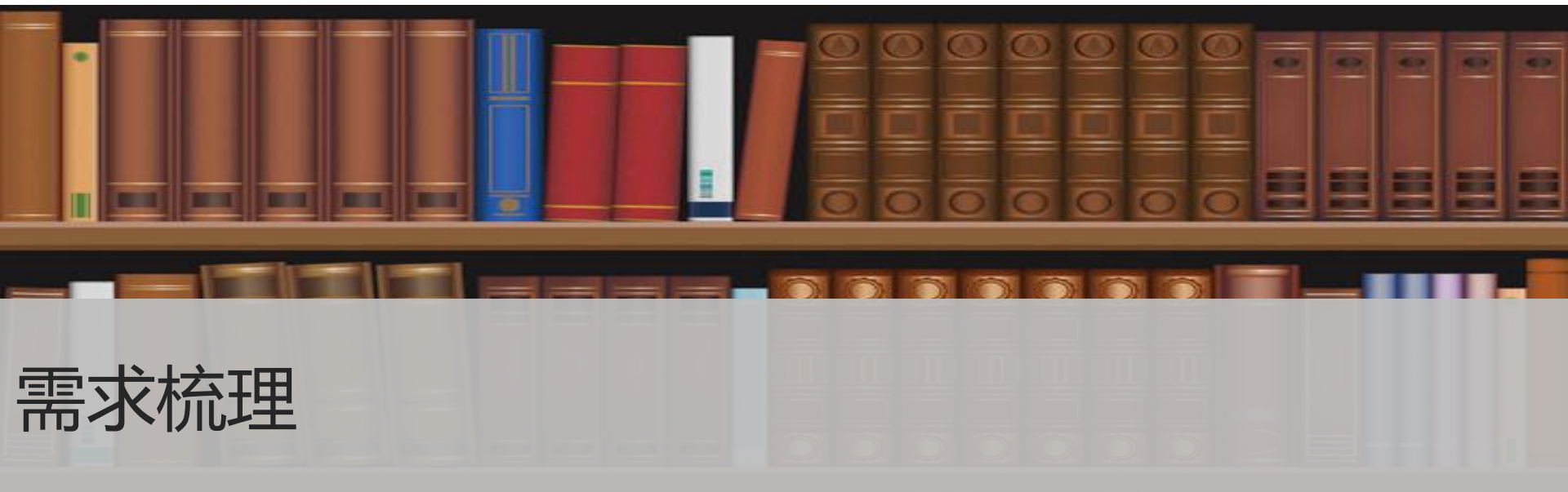
我是前端工程师

- IE6占比
- 分辨率
- 页面性能
- js异常



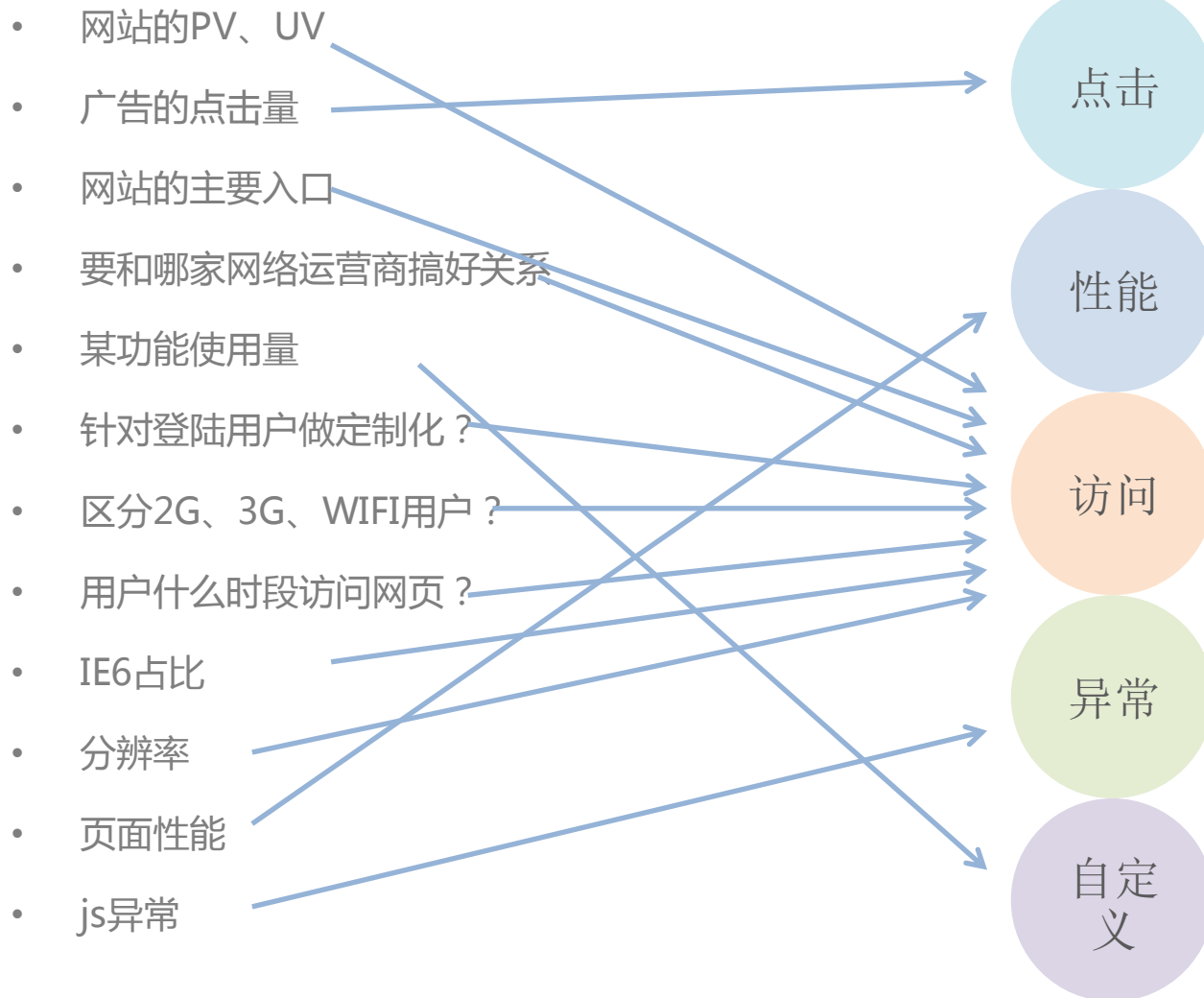
WE NEED YOU





需求梳理

需求划分



指标确定---性能

- 白屏时间
- 首屏时间
- 用户可操作
- 页面总下载

指标确定---访问

- PV、UV
- Refer来源（网站主要入口）
- 运营商（要和哪家网络运营商搞好关系）
- 登陆情况（针对登陆用户做定制化）
- 网络类型（区分2G、3G、WIFI用户）
- 访问时段
- 浏览器分布（IE6占比）
- 分辨率
- 操作系统
- 地域分布

指标确定---点击

- 总点击量
- 人均点击量
- 流出url分布
- 点击时间分布
- 首次点击时间

指标确定---异常

- 异常pv比例
- 异常的浏览器
- js异常提示信息
- 异常的js文件、行数
- 某异常的数量趋势

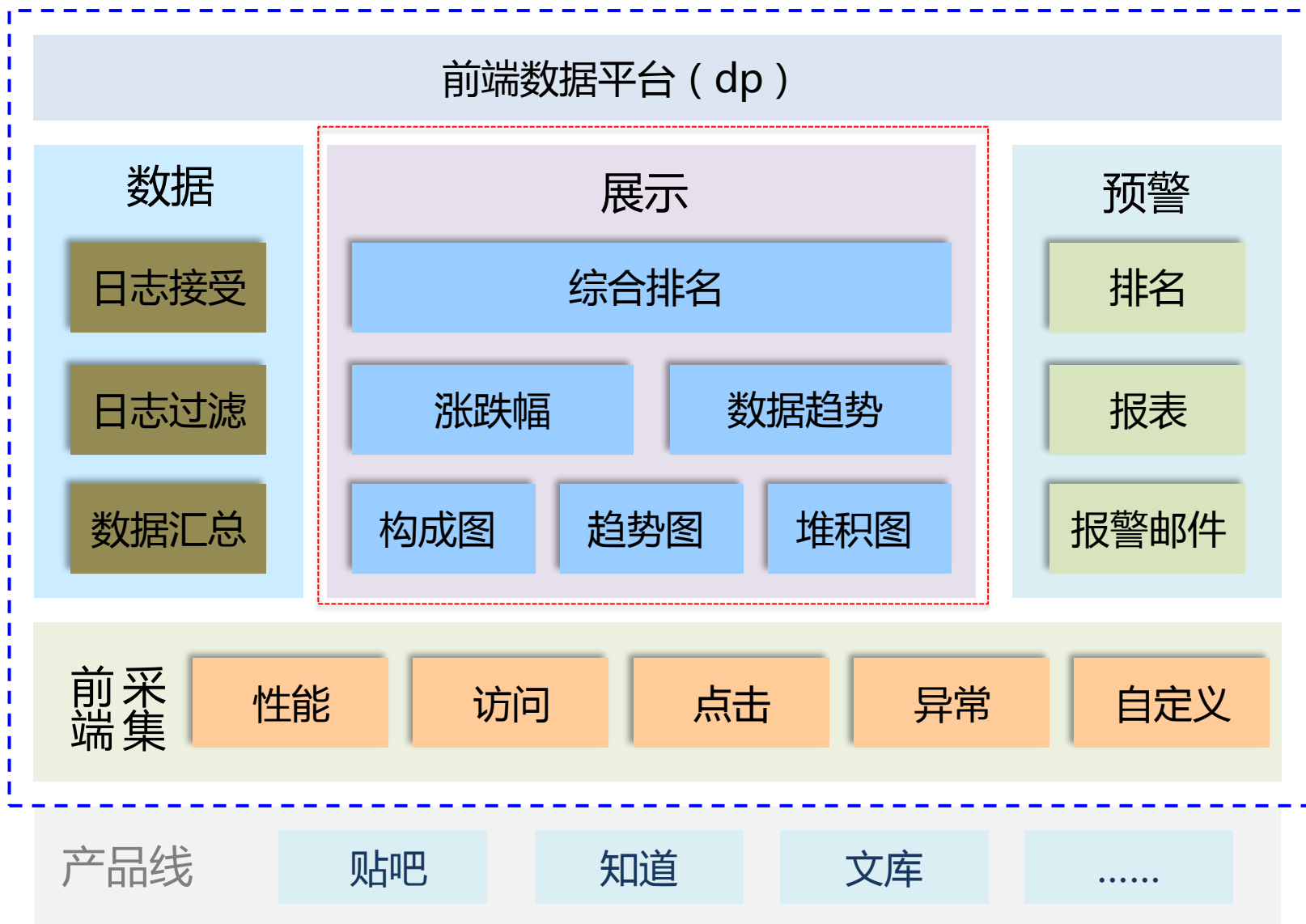
指标确定---自定义

- 数量趋势
- 维度信息



系统设计

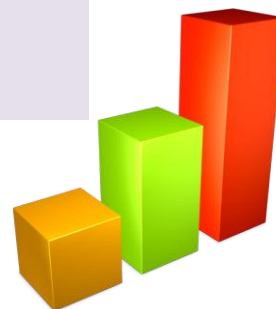
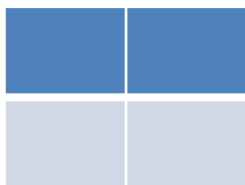
系统构架





展示

数据可视化

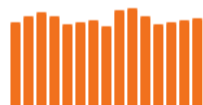


访问数据

PV 750,159,367

同比变化 +9.37% (2014-03-13)

PV波动

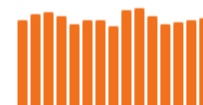


(2014-03-06~2014-03-20)

UV 643,459,874

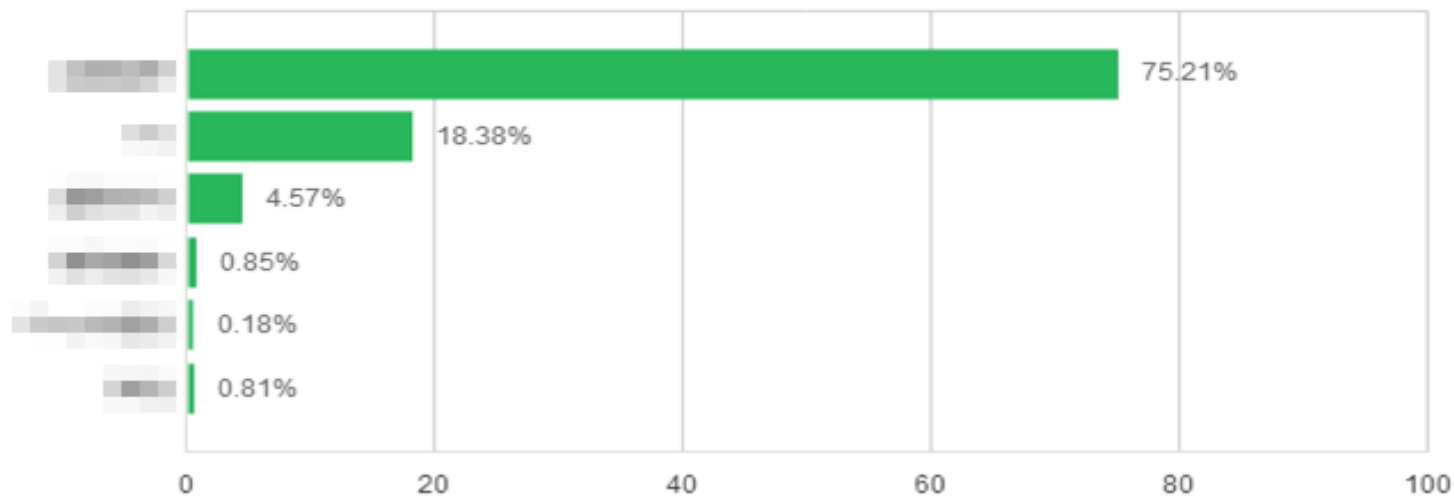
同比变化 +8.10% (2014-03-13)

UV波动

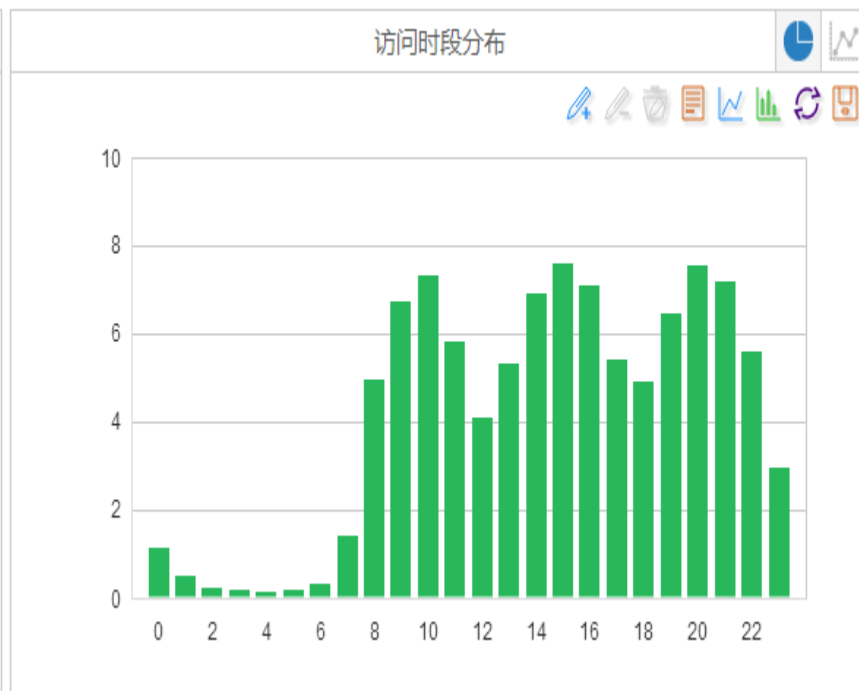
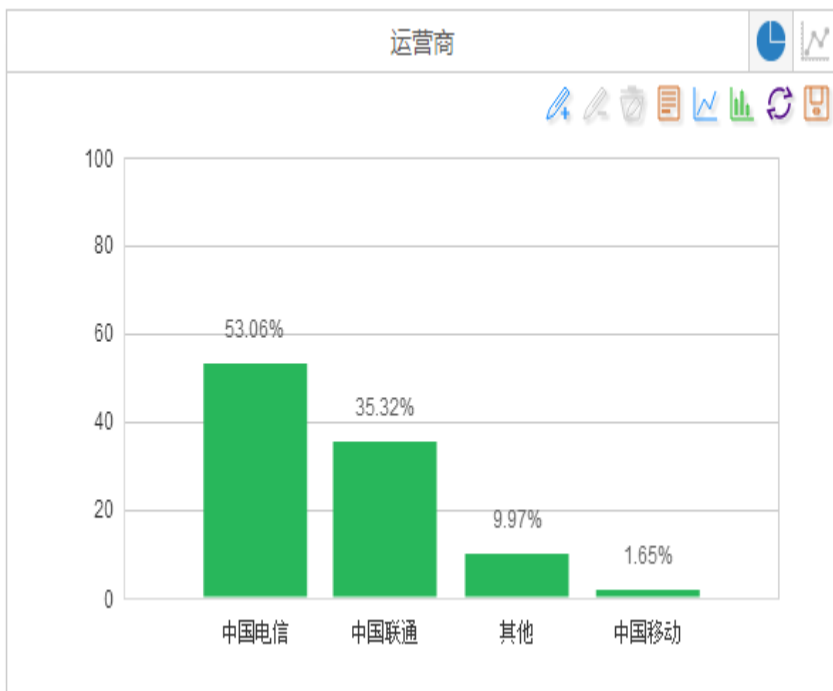


(2014-03-06~2014-03-20)

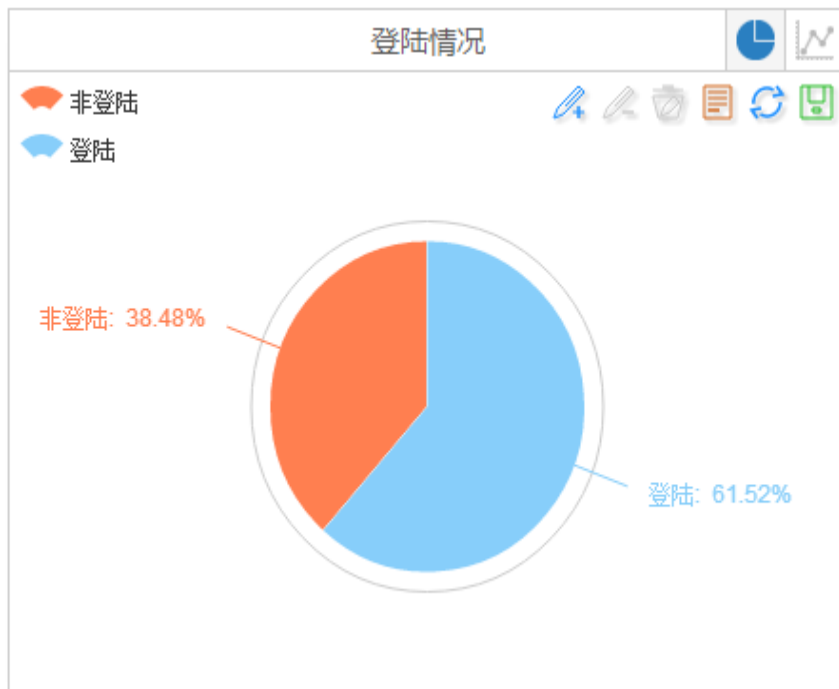
来源分布



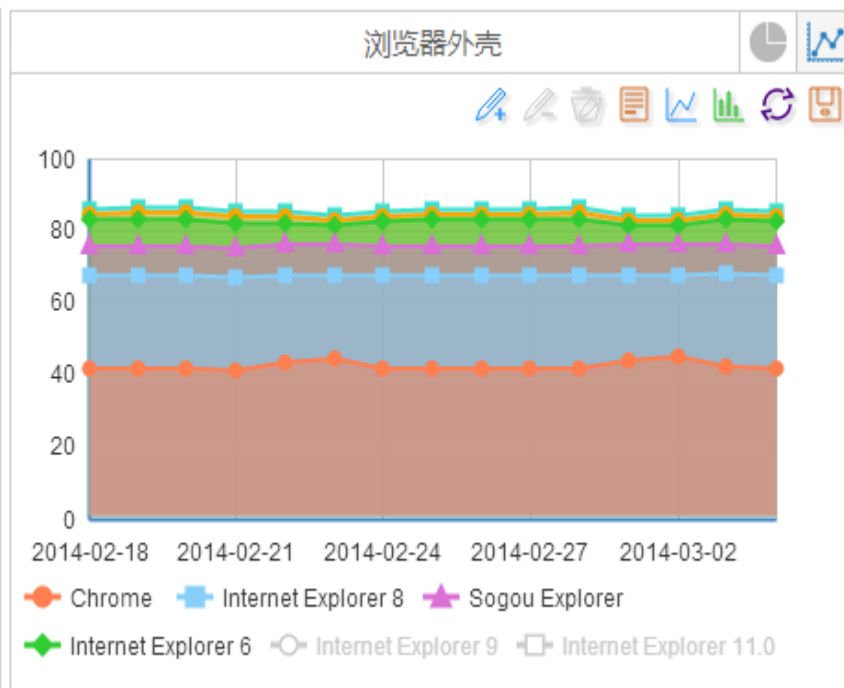
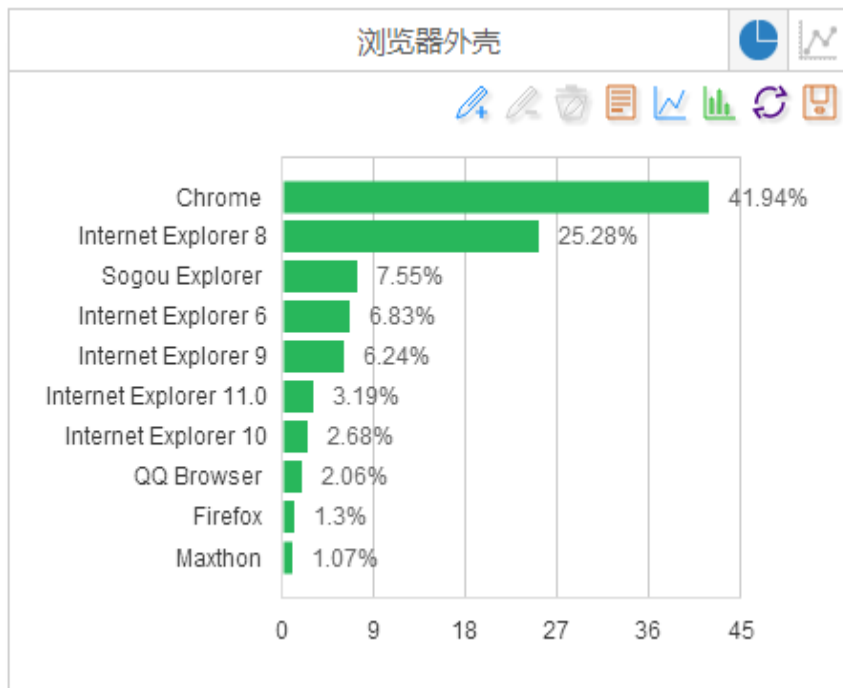
访问数据



访问数据



访问数据



点击数据

人均点击 **10.93**

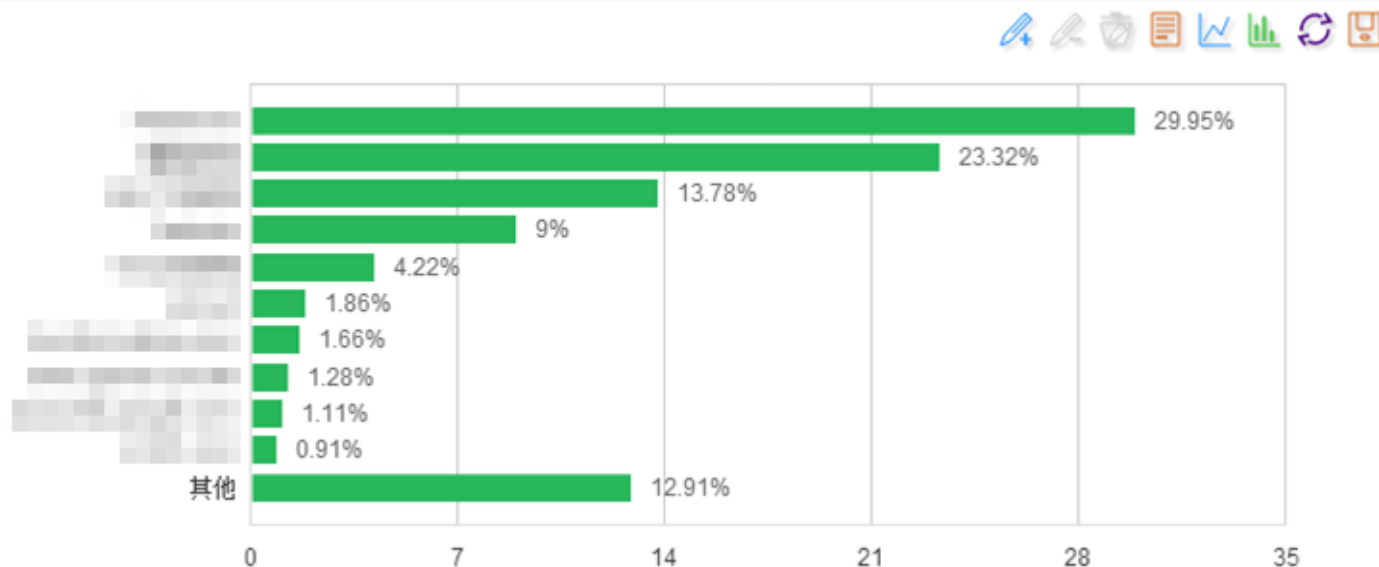
同比变化 **+7.82%** (2014-03-13)

人均点击波动



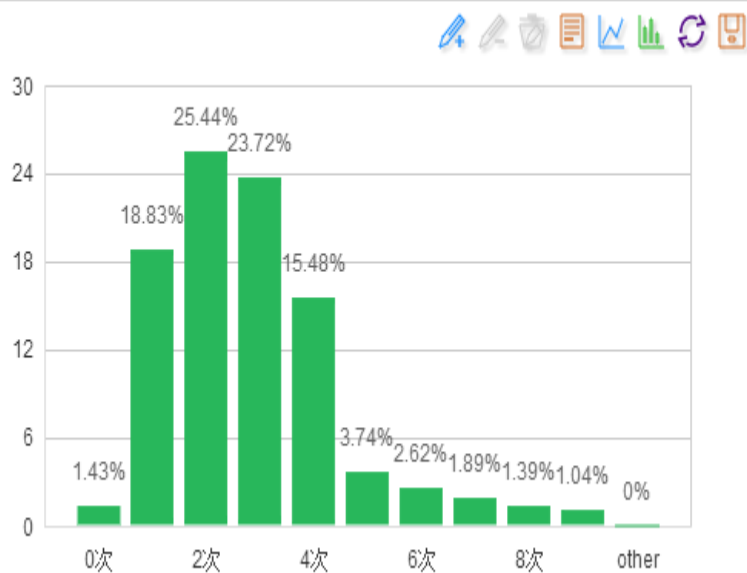
(2014-03-06~2014-03-20)

流出url分布

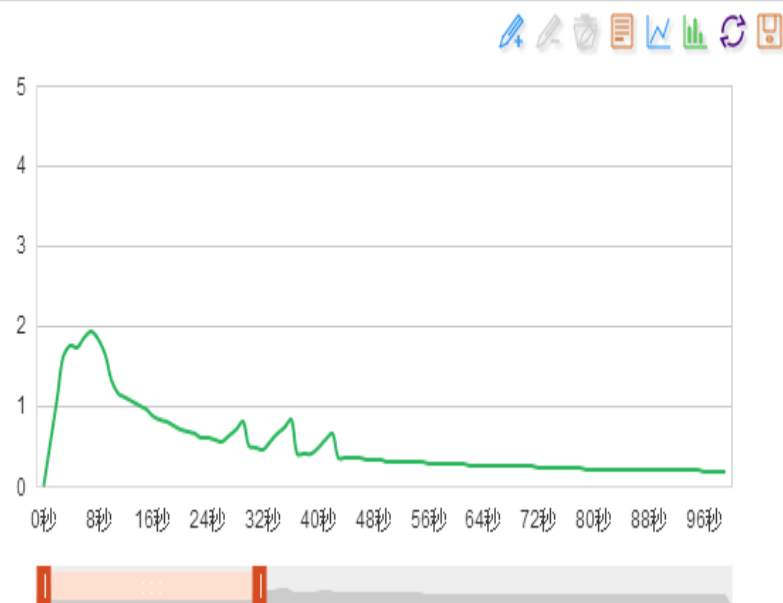


点击数据

点击次数分布



点击时间分布



异常数据—概况

PV异常率 **0.12%**(557,876)

同比变化 **+0.19%** (2014-02-24)

PV异常率波动



(2014-02-17~2014-03-03)

异常PV浏览器



Safari Chrome IE8 QQ Browser IE6 Sogou Explorer IE7 IE9
Other IE11 Firefox IE10 Opera

异常数据—详细

error test

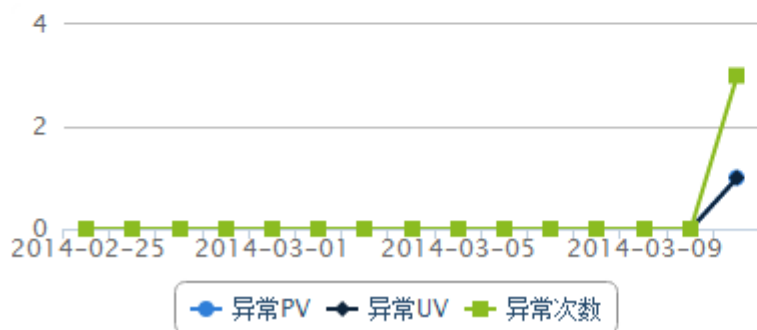
PV异常率 : 0.00%(1) UV异常率 : 0.00%(1) 异常发生次数 : 3

异常发生的位置列表 :

(1) `common/export.js` 异常method : `call` 所在行 : 7 PV异常率 : 0.00%(1) UV异常率 : 0.00%(1) 异常发生次数 : 3

异常发生的浏览器 : `Chrome`(100%)

最近半月趋势 :



自定义数据



维度三(帖子id) - 2014-03-18

1982980514	7015 (-7.8%)
1831116258	3494 (-10.2%)
3743852096	3248 (-24.6%)
3994882921	2778 (-21.5%)
214534818	2703 (-3.30%)
2724281890	2312 (+37.3%)
2182199799	2238 (-19.1%)
274728855	2207 (+17.2%)
3961103225	2197 (-54.5%)
1658387265	1971 (-15.5%)



华丽的背后

FZLOL.COM

前端数据平台 (dp)

数据

日志接受

日志过滤

数据汇总

展示

综合排名

涨跌幅

数据趋势

构成图

趋势图

堆积图

预警

排名

报表

报警邮件

前端采集

性能

访问

点击

异常

自定义

产品线

贴吧

知道

文库

.....

数据流程

采集

- 页面注入js脚本
- 监听事件（click、onerror、domReady等）、暴露接口

发送

- 访问特定的url地址
- 数据作为url的参数，如：<http://www.baidu.com/dp.gif?type=pv>

接收

- server端接收参数数据，并且包括userAgent数据
- 将数据按行存储为文本log文件

格式化

- 读取log文件，过滤脏数据（格式错误、超过阈值等）
- 结构化处理（一个字符串 → 多个字段）

计算

- count、sum、avg、top、group、sort等

入库

- mysql

1、js异常监控

初期： `window.onerror`

优点

- 可以监控到几乎所有js异常
- 产品线的js代码无需任何修改

缺点

- 线上js是混淆压缩的，无行号
- 跨域的js，错误信息是“Script error”
- 跨域的js，获取不到js文件名

产品线js的CDN基本都跨域



Script error

PV异常率：0.46%(648,147) UV异常率：0.88%(544,099) 异常发生次数：5,650,925

异常发生的文件列表：

(1) 未知文件 PV异常率：0.43%(636,688) UV异常率：0.17%(533,301) 异常发生次数：437,837

1、js异常监控

Then : try/catch

优点

- 不怕跨域
- 无惧js压缩和混淆，捕获压缩前的行号
- 精确定位到模块、方法名

```
a.js
1  var a = {
2      init: function(){
3          try{
4              hello();
5          }catch(e){
6              alog('exception.send',{
7                  msg: e.message,
8                  line: 2
9                  module: 'a.js',
10                 method: 'init'
11             });
12         }
13     }
14 }
```

难道要手动给所有js加try/catch？

Of course not ! js压缩打包时生成代码

FIS : <https://github.com/fex-team/fis> 

```
a.js
1  var a = {
2      init: function(){
3          //addTry begin
4          hello();
5          //addTry end
6      }
7  }
```

2、大数据

数据达到 PB 级别

Hadoop

- Nodejs + map/reduce
- <http://hadoop.apache.org>

Hive

- 类SQL语句来执行hadoop任务
- <http://hive.apache.org>



总结

前端基础数据平台

- 确定数据指标
- 数据采集
- 大数据处理
- 可视化展示



**Did I Miss
Something?**

说好的性能呢？



百度前端基础数据平台介绍

性能监控篇

Shared By 张涛

主要议题

1. 为什么要监控性能
2. 常用监控方式
3. 开始搭建性能监控系统
4. 利用监控解决问题

Part 1 为什么要监控性能

1 利益

Google 延迟 400ms	搜索量下降 0.59%
Bing 延迟 2s	收入下降 4.3%
Yahoo 延迟 400ms	流量下降 5-9%
Mozilla 页面打开减少 2.2s	下载量提升 15.4%
Netflix 开启 Gzip	性能提升 13.25% 带宽减少 50%

[数据来源1](#) [数据来源2](#)

2 体验



40%

用户会离开如果页面没在**3s**内响应



80%

不会再次访问一个性能体验糟糕的
移动网站

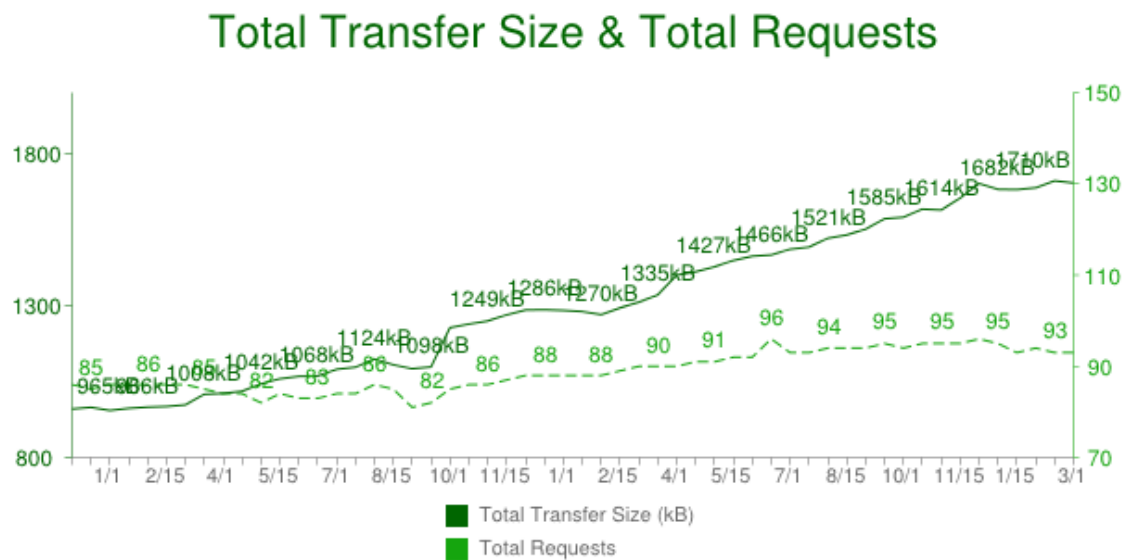


50%

会告诉朋友不要访问这类网站

Source: Colt McAnlis (Google)/Guy Podjarny (Akamai)

③ 移动时代的挑战

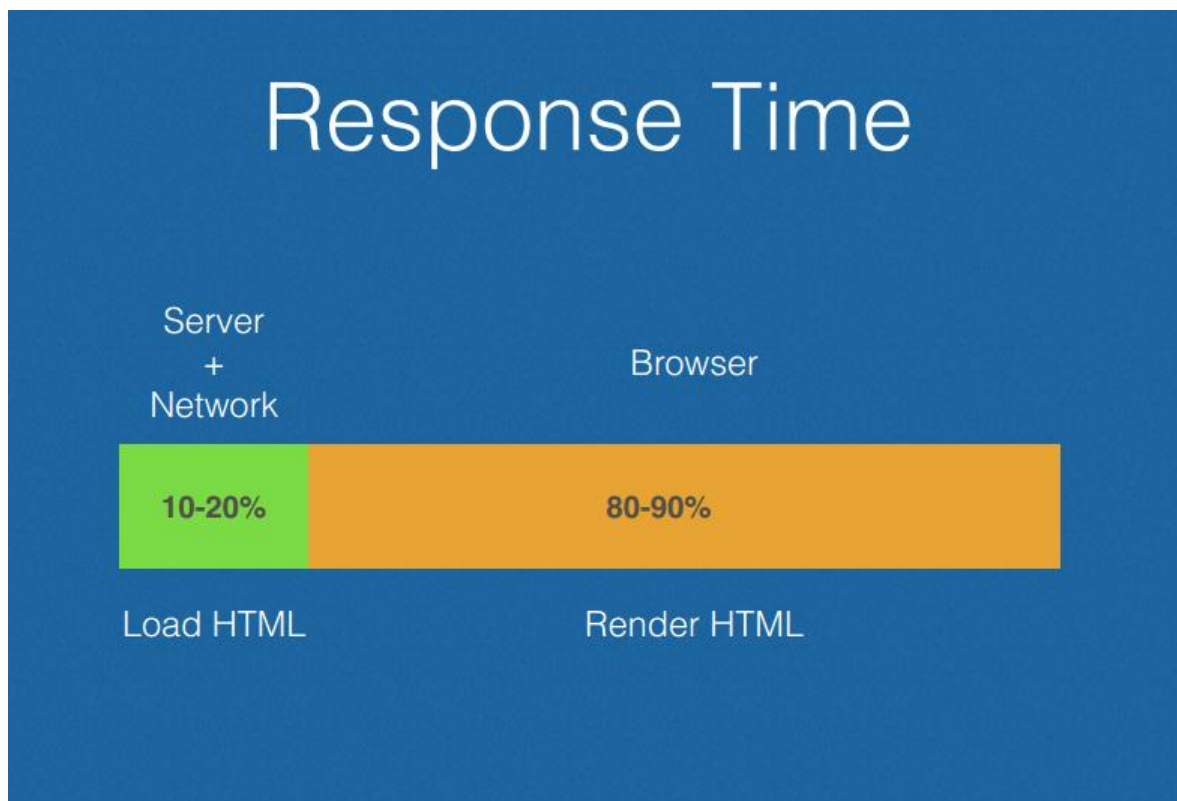


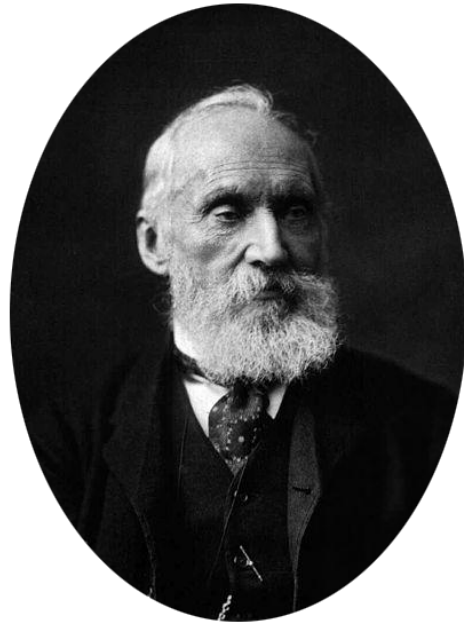
i



数据: [Http Archive](#)

4 前端渲染瓶颈





“If you cannot measure it, you
cannot improve it”

——William Thomson

Part 2 常用监控方式

WebPageTest

输入url获取各地测试数据

The screenshot displays the WebPageTest interface for a performance test of qq.com. The top navigation bar includes links for HOME, TEST RESULT (highlighted), TEST HISTORY, FORUMS, DOCUMENTATION, and ABOUT. A 'Need help improving?' link is also present. The main content area shows the test results for qq.com, including the test location (Dulles, VA - IE 11 - Cable) and the test time (2014年3月15日 下午5:22:25). A summary of performance metrics is shown in a row of colored boxes: F (First Byte Time), A (Keep-alive Enabled), A (Compress Transfer), F (Compress Images), F (Progressive JPEGs), F (Cache static content), and a green checkmark for Effective use of CDN. Below this, a table provides detailed performance metrics for the test. The table is divided into two main sections: 'Document Complete' and 'Fully Loaded'. The 'Document Complete' section shows a Load Time of 102.079s, First Byte of 5.660s, Start Render of 11.998s, and Speed Index of 30075. The 'Fully Loaded' section shows a Time of 0.000s, Requests of 0, and Bytes In of 125 KB. Below the table, there are three tabs: Waterfall, Screen Shot, and Video. The Waterfall tab is selected, showing a detailed waterfall chart of the test results. The Screen Shot tab shows a screenshot of the qq.com homepage. The Video tab shows a video of the test results. A 'Re-run the test' button is located on the left side of the page. The bottom of the page features a download bar with a file named 'phantomjs-logo.png' and a link to '显示所有下载内容...'. The URL 'http://www.webpagetest.org/' is displayed at the bottom of the image.

Web Page Performance Test for
qq.com

From: Dulles, VA - IE 11 - Cable
2014年3月15日 下午5:22:25

Summary Details Performance Review Content Breakdown Domains Screen Shot

Re-run the test

Raw page data - Raw object data
Export HTTP Archive (.har)
See in ShowSlow
View Test Log

	Load Time	First Byte	Start Render	Speed Index	Document Complete			Fully Loaded		
	Time	Requests	Bytes In	Time	Requests	Bytes In	Time	Requests	Bytes In	
First View	102.079s	5.660s	11.998s	30075	0.000s	0	125 KB	102.079s	16	125 KB

Waterfall

Screen Shot

Video

First View (Error: Timed Out)

phantomjs-logo.png


显示所有下载内容...

http://www.webpagetest.org/

PhantomJS

命令行获取页面加载状况

Fork me on GitHub

 **PhantomJS**

SOURCE CODEDOCUMENTATIONAPIEXAMPLESFAQ

Full web stack No browser required


PhantomJS is a headless WebKit scriptable with a JavaScript API. It has **fast** and **native** support for various web standards: DOM handling, CSS selector, JSON, Canvas, and SVG.


[Download v1.9](#) [Get started](#)


Simple Javascript example

```
console.log('Loading a web page');
var page = require('webpage').create();
var url = 'http://www.phantomjs.org/';
page.open(url, function (status) {
    //Page is loaded!
    phantom.exit();
});
```

Community:

 Read the release notes

 Join the mailing list

 Report bugs

PhantomJS is an optimal solution for

phantomjs-logo.png

显示所有下载内容...

JS监测

部署脚本到页面中采集



监控方式对比

类型	优点	缺点	示例
非侵入式	<ul style="list-style-type: none">指标齐全客户端主动监测竞品监控	<ul style="list-style-type: none">无法知道性能影响用户数采样少容易失真无法监控复杂应用与细分功能	Yslow , Pagespeed , PhantomJS , UAQ
侵入式	<ul style="list-style-type: none">真实海量用户数据能监控复杂应用与业务功能用户点击与区域渲染	<ul style="list-style-type: none">需插入脚本统计网络指标没有全部统计到无法监控竞品	DP , Google统计

需要一种可持续、基于用户访问真实情况、能监控业务功能的监控

综合利用发挥最大价值

我们的做法

- 使用JS监测线上用户真实访问性能 **为主**
- 使用phantomJS等工具线下分析页面静态资源 **为辅**

Part 3 开始搭建性能监控系统

1 监控什么指标

工程师视角

DPS查询

TCP连接

发送请求

等待响应

html传输

静态资源下载

解析文档

执行JS/CSS规则

计算布局

渲染完成



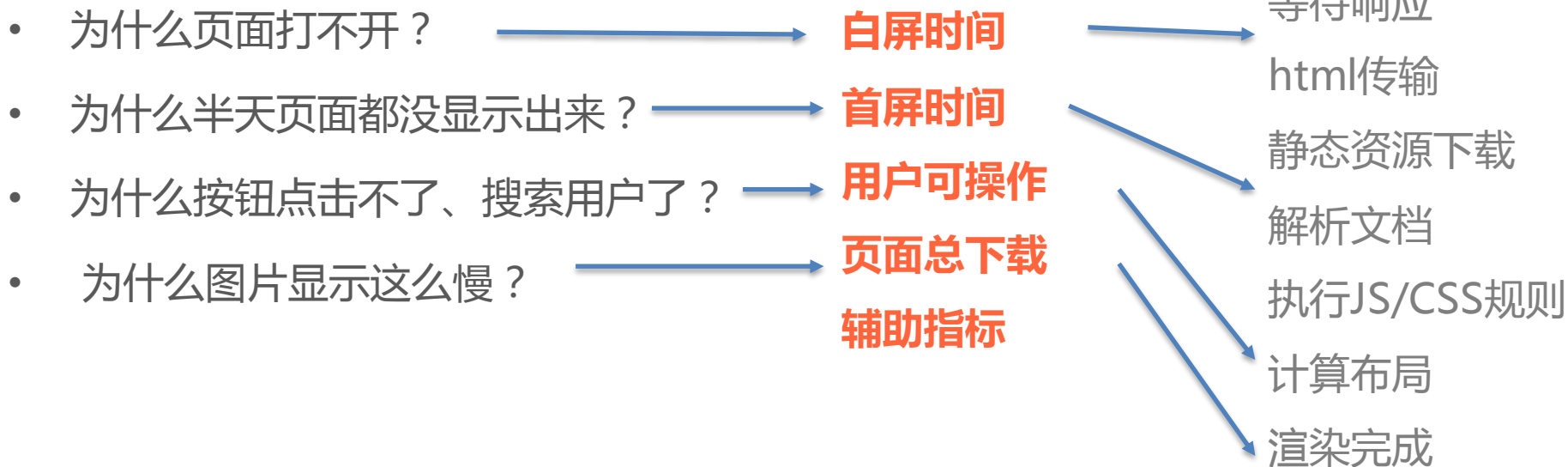
联系

用户视角

- 为什么页面打不开？
- 为什么半天页面都没显示出来？
- 为什么按钮点击不了、搜索用户了？
- 为什么图片显示这么慢？

1 监控什么指标

基于用户角度的关键指标选取



② 数据采集 - 统计起点

用户点击一个链接或者输入url确认开始统计

方式一

使用cookie/hash记录用户点击超链接的时间戳

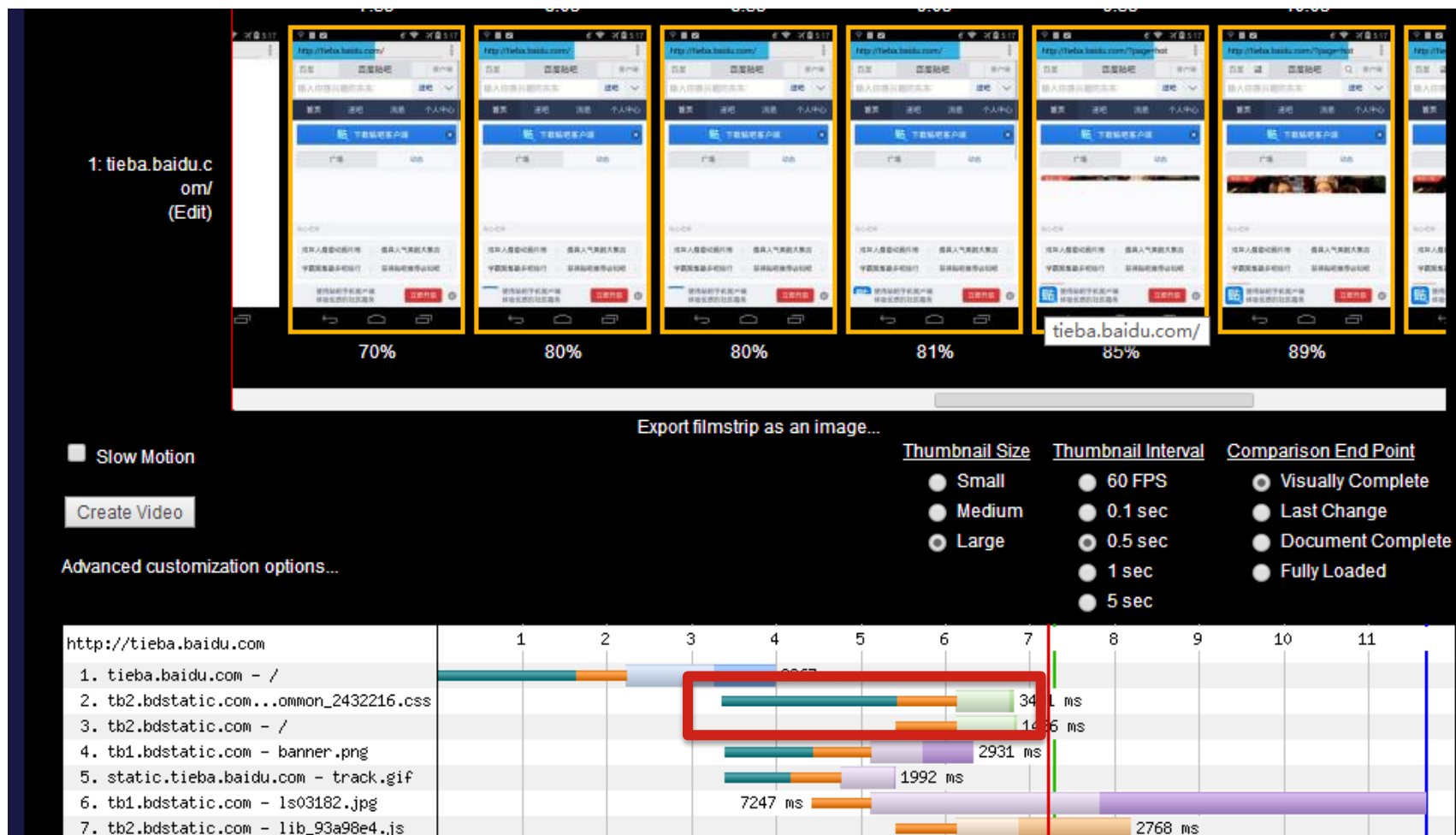
方式二

使用 [Navigation Timing](#)接口

② 数据采集 - 白屏时间

页面白屏结束出现在头部资源下载完附近

首次出现内容



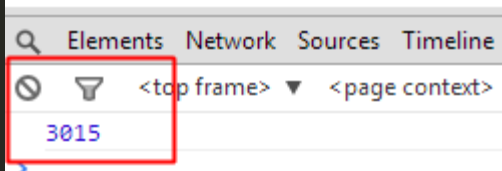
http://www.webpagetest.org/video/compare.php?tests=140318_OV_BA1-r:1-c:0

2 数据采集 - 白屏时间

头部资源下载完成 ≈ 白屏时间

```
<meta charset="UTF-8"/>
<script>
    var start_time = +new Date; //测试时间起点，实际统计起点为DNS查询
</script>
<!-- 3s后这个js才会返回 -->
<script src="script.php"></script>
<script>
    var end_time = +new Date; //时间终点
    var headtime = end_time - start_time; //头部资源加载时间
    console.log(headtime);
</script>
</head>
<body>
    <p>在头部资源加载完之前页面将是白屏</p>
    <p>script.php被模拟设置3s后返回，head底部内嵌JS等待前面js返回后才执行</p>
    <p>script.php替换成一个执行长时间循环的js效果也一样</p>
</body>
```

script.php替换成一个执行长时间循



头部底部内嵌JS来统计头部资源加载 -> 白屏时间

② 数据采集 - 首屏时间



图片是制约**首屏**的主要因素

获取首屏内图片的加载耗时即可获取首屏时间

2 数据采集 - 首屏时间

首屏统计流程



3 页面onload之后找到最慢一张图片加载时间

1 首屏大概位置执行统计JS

② 数据采集 - 首屏时间

一些陷阱：

1. 图片加载完成、出错,gif图片重复触发加载事件的处理
2. iframe的处理：同图片
3. **异步渲染**的处理：异步数据插入后再计算首屏
4. **css背景图片**的处理：首屏重要css背景通过js发起图片请求判断是否已加载
5. **没有图片**则以文字出现时间为准，可认为此统计js执行时刻

② 数据采集 - 可操作时间

Domready or 核心JS加载完毕(模块化异步加载情况)

② 数据采集 - 总下载时间

onload

同步为主

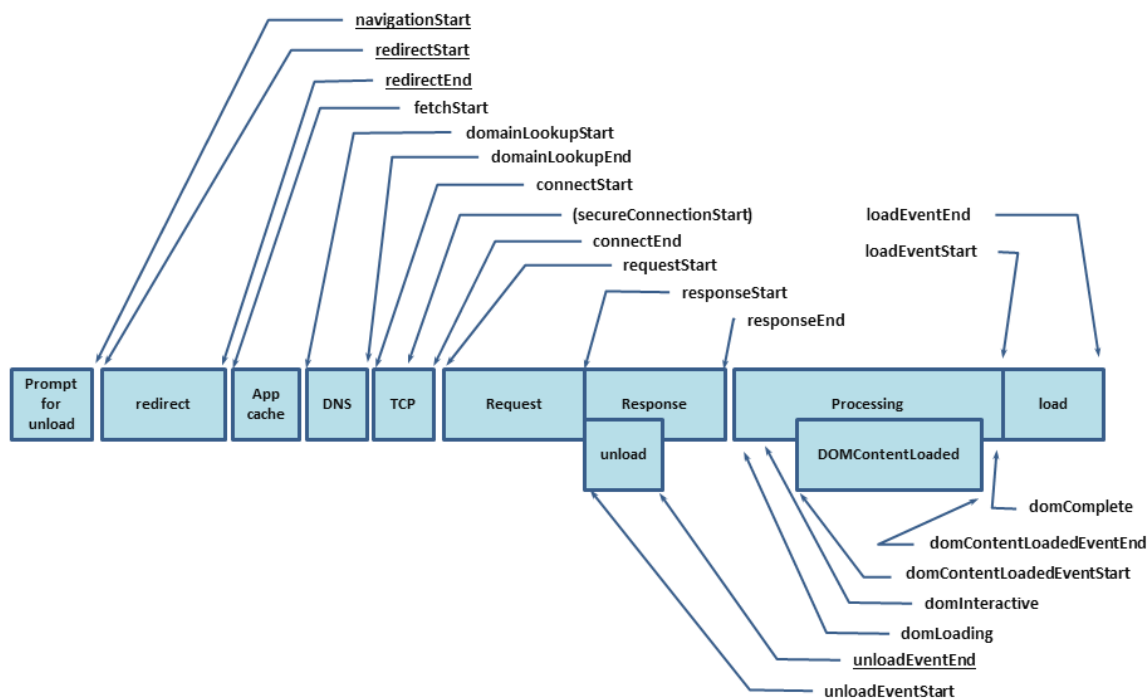
or

异步渲染完成

异步为主

2 数据采集 - 网络指标

Performance Timing



```
> window.performance.timing
▼ PerformanceTiming {LoadEventEnd: 138840335031388403350128...}
  connectEnd: 1388403349694
  connectStart: 1388403349667
  domComplete: 1388403350296
  domContentLoadedEventEnd: 1388403350139
  domContentLoadedEventStart: 1388403350128
  domInteractive: 1388403350128
  domLoading: 1388403349759
  domainLookupEnd: 1388403349667
  domainLookupStart: 1388403349663
  fetchStart: 1388403349658
  loadEventEnd: 1388403350311
  loadEventStart: 1388403350310
  navigationStart: 1388403349617
  redirectEnd: 0
  redirectStart: 0
  requestStart: 1388403349694
  responseEnd: 1388403349765
  responseStart: 1388403349744
  secureConnectionStart: 0
  unloadEventEnd: 0
  unloadEventStart: 0
  proto : PerformanceTiming
```

http://www.html5rocks.com/en/tutorials/webperformance/basics/?redirect_from_locale=zh

② 数据采集 - 用户网络

移动端网络一直是瓶颈，如何统计用户使用的网络类型？

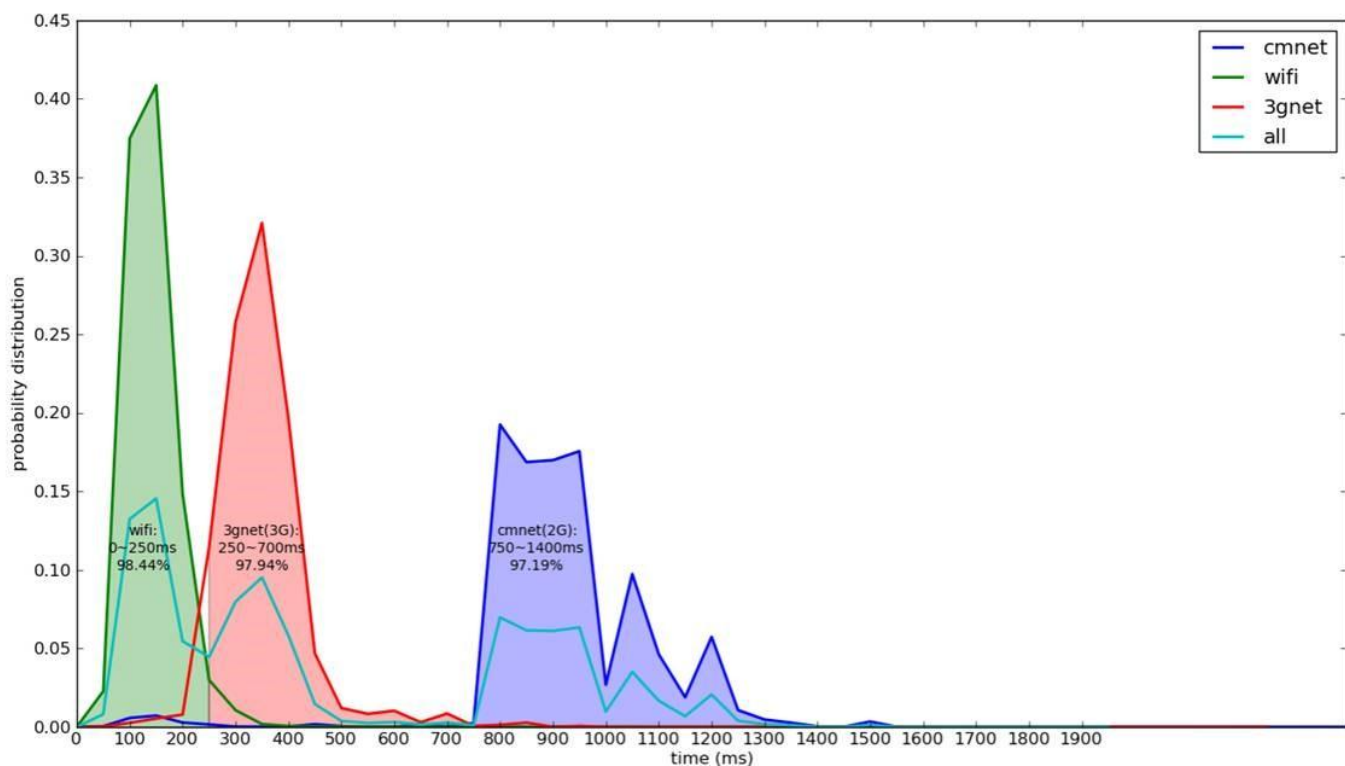
1. Html5接口 navigator.connection
2. 使用测速根据下载速度分布确定IP段对应的网络

<http://www.w3.org/TR/netinfo-api/>



② 数据采集 - 用户网络

通过IP测速来获取全用户IP段下载速率，将不同速率分布分位三个区间分别对应2g 3g WIFI



③ 数据输出

让数据会说话

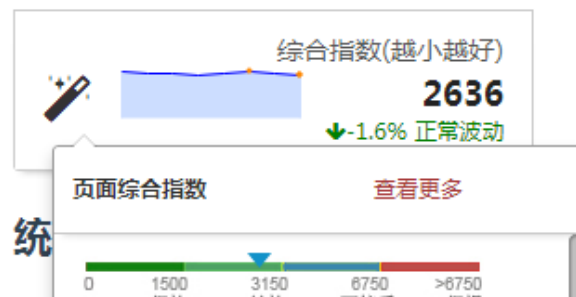
3 数据输出

-	产品线	页面 图 自 心	与昨日相比PV异常与昨日相比点击量异常	12.34万	3264	20	↓ 2名	<div><div></div><div></div><div></div><div></div></div>	<div></div>										
IE6浏览器占比		首次点击时间(秒)		无点击用户占比		登录用户占总用户的		最大占比的分辨率 占		主要流量来源 占 48.41%									
6.16%		均值 46.09 中位数 24		67.52%		61.15%		33.44%		百度									
1366*768																			
<div>综合指数变动趋势</div> <div></div>				<div>UAQ一周竞品监控(点击条形查看UAQ竞品报表)</div> <div>暂无竞品数据</div>				<div>关键指标</div> <div><div>很快</div><div>较快</div><div>用户可接受</div><div>很慢</div></div>											
<div>排名变动趋势</div> <div></div> <div>192020192021181918181818191820</div>				<div>1223ms</div> <div>白屏时间</div>				<div>2106ms</div> <div>用户可操作</div>				<div>3066ms</div> <div>首屏时间</div>				<div>3140ms</div> <div>总下载时间</div>			
+	产品线	页面 图 自 心		100.53万	3314	21	↑ 3名	<div><div></div><div></div><div></div><div></div><div></div></div>	<div></div>										

3 数据输出

综合评价

较快



页面排名

31/55

上升3名



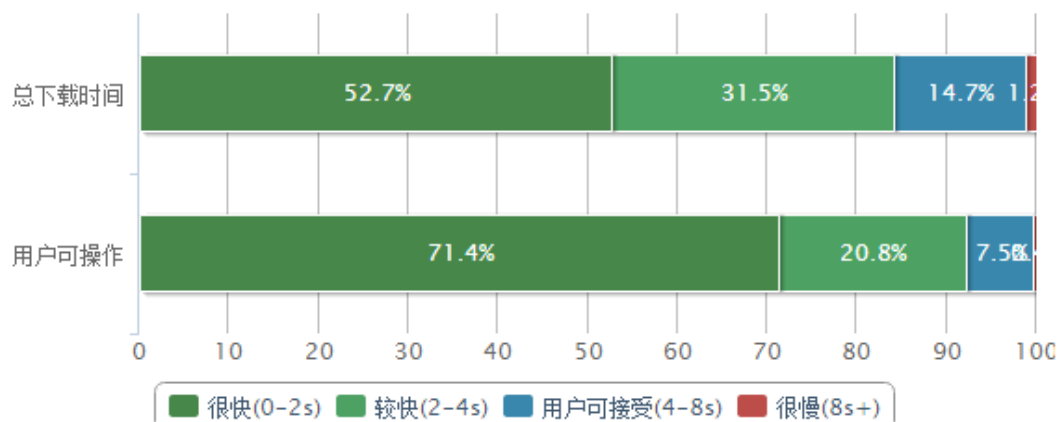
统计评级



昨天采样 84.03万(最佳20万+)

总体比例分析

累积分布



>渲染耗时

6

首屏时间

7

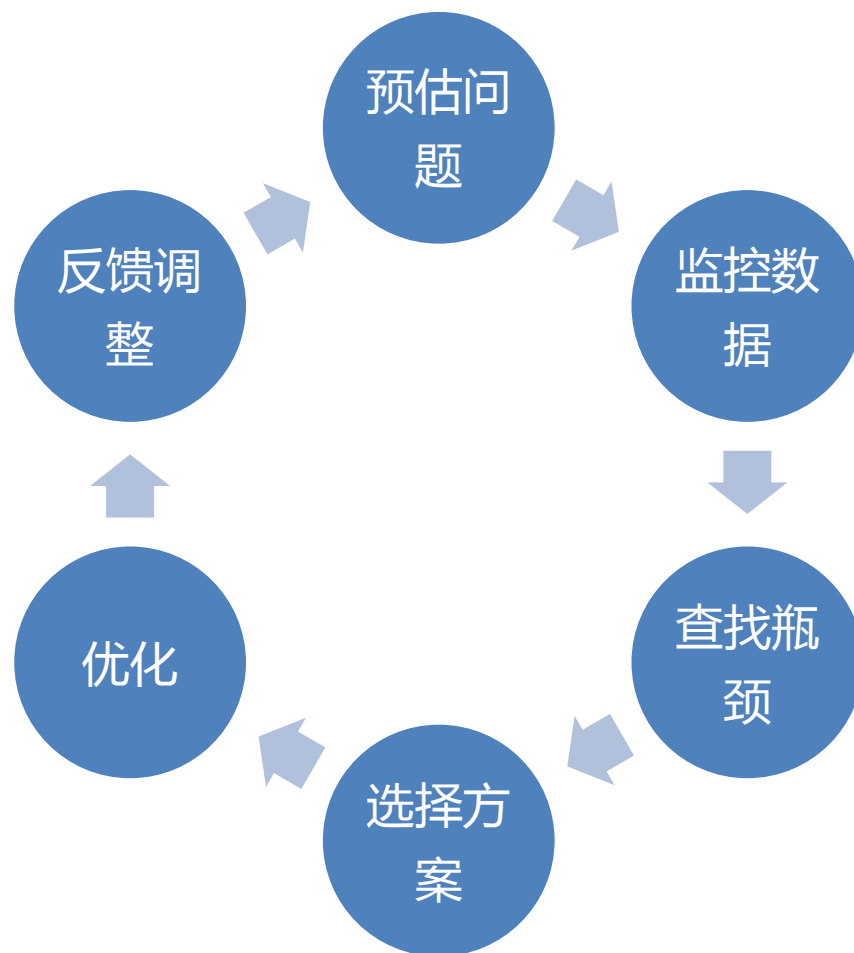
用户可操作

8

自定义核心指

Part 4 利用监控解决问题

解决流程



Case：巴西DNS优化

- 现状分析：巴西等国家网络慢，国外部署机房又少，网页加载很慢
- 预估问题：可能是某静态资源域名下DNS查询很慢
- 监控数据：使用resource timing监控第三方域名DNS等时间
- 查找瓶颈：确认DNS查询确实很长
- 选择方案：服务迁移到本地等
- 优化实施：按照方案实施优化
- 反馈调整：DNS时间大为减少，页面加载有所提升，但仍需优化

回顾总结

1. 综合利用各种监控优势
2. 多从用户角度思考
3. 尝试多种方案，灵活利用Html5等新技术
4. 采集关键数据
5. 监控必须解决问题为基础
6. 没有牛逼的技术，关键在于更好解决需求

Thanks

感谢大家的光临！

- FEX官网 <http://fex.baidu.com>
- Navigation Timing监控性能
- Resource Timing
- Facebook测速方案
- 基于phantomJS的性能分析工具phantomas
- <http://www.webperformancetoday.com/>
- js异常： <http://t.cn/8sLQ9W9>
- Echarts图表库： <https://github.com/ecomfe/echarts>