

# 手腾JS资源版本增量更新方案

2019

# about me

卢勇福 (waynelu)

微博:

<http://t.qq.com/waynelu>

<http://www.weibo.com/u/1849616271>

github:

<https://github.com/luyongfugx>



# 提纲

- 背景和问题
- 传统cdn静态资源方式存储js
- html5离线存储方式存储js
- js增量更新算法设计
- 增量更新接入方案
- 实战效果

## 背景和问题

2014.3.1-2014.3.31期间手腾的几个webapp

共发布:12次 ,

发布间隔:2 ~ 3天 ,

每次修改js:3至4个

js请求:2亿多次

js修改:基本小于5% , 不超过20% 的。

## 背景和问题

在快速迭代的敏捷开发过程中，我们通过快速更新版本及时响应了用户需求和bug修复，但是我们同时也浪费了用户极为看重的流量资源！！！



# 传统cdn静态资源方式存储js cdn+浏览器cache

优点：

- 1.简单，容易维护
- 2.304 cache

缺点：

- 1.缓存会失效，用户强制刷新时可能会有http请求
- 2.快速迭代版本过程中少量修改，全量更新

# html5离线存储方式存储js

## html5 appcache (离线存储 )

优点：

真正的离线，只有版本更新才会有请求

缺点：

1.新版本启用刷新体验问题

2.难于维护，灰度等策略比较难实施

3.快速迭代版本过程中少量修改，全量更新

# 本地存储模拟离线存储

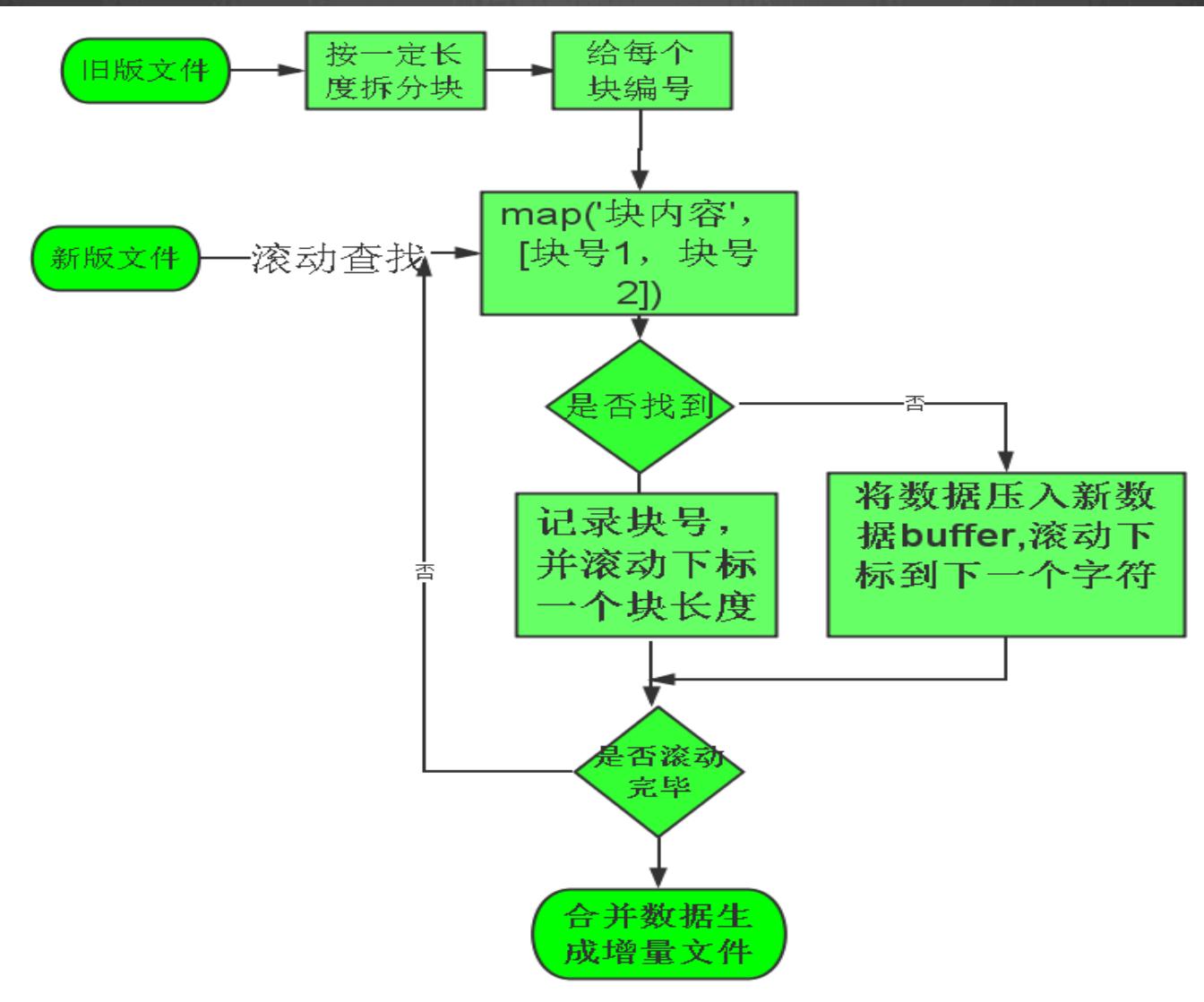
## 可行性：

1. 大小5m,一般js小于1m
2. 跨域问题：Access-Control-Allow-Origin:\*
3. key-value形式本地数据，用eval执行

## 优点：

1. 减少不必要的http请求,有更新才有请求，省流量
2. 避免离线存储带来的子刷新体验问题
3. 增量更新!

# js增量更新算法设计



# js增量更新算法设计

旧文件分成n块

chunk0	chunk1	chunk2	chunk3	chunk4	chunk5
--------	--------	--------	--------	--------	--------

通过滚动查找得到增量更新文件

chunk0	data1	chunk1	chunk2	data2	chunk3	chunk4	chunk5
--------	-------	--------	--------	-------	--------	--------	--------

最终增量文件表示如下:

1, data1, 2, 3, data2, 4, 5, 6

进一步合并顺序块得到:

[1, 1], data1, [2, 2], data2, [4, 3]

新文件内容:

chunk0+data1+chunk1+chunk2+data2+chunk3+chunk4+chunk5

# js增量更新算法设计

以  $s = '1345678abcdefghijklmnpq'$  修改为

$a = '13456f78abcd2efghijklmnpq'$  例

设块长度为4则，源文件分成(第一行块号，第二行数据):

1	2	3	4	5	6	7
$s = '1$	3456	78ab	cdef	ghij	klmn	opq'

通过滚动查找比对，得到新的文件构成如下

新数据	2	新数据	3	新数据	5	6	7
$a = '1$	3456	f	78ab	cd2ef	ghij	klmn	opq'

最终增量文件表示如下数组:

```
[ "a= '1",2,"f",3,"cd2ef",5,6,7]
```

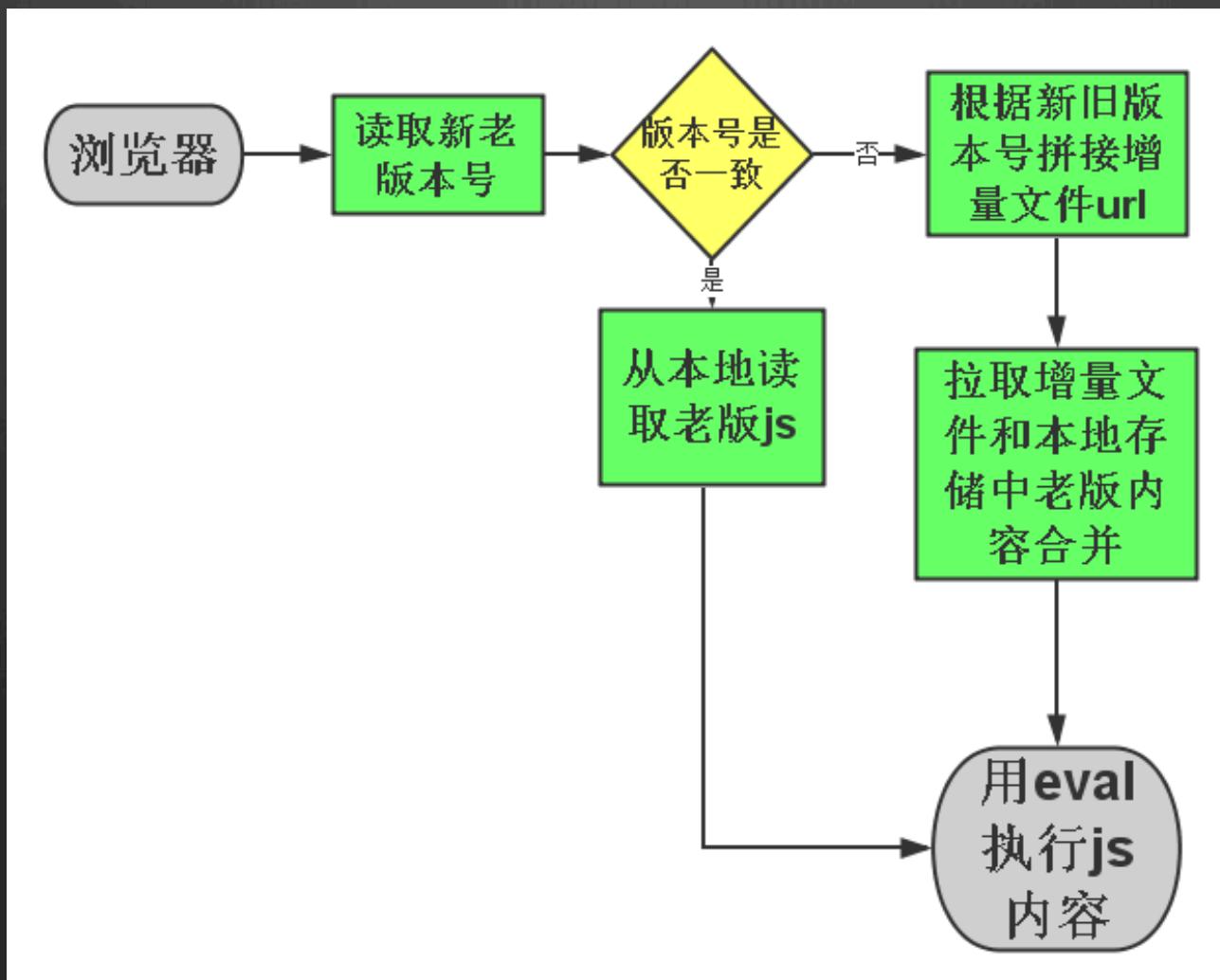
进一步合并顺序块，可用一个js数组表示为

```
[ "a= '1" ,[2,1], "f" ,[3,1], "cd2ef" ,[5,3]
```

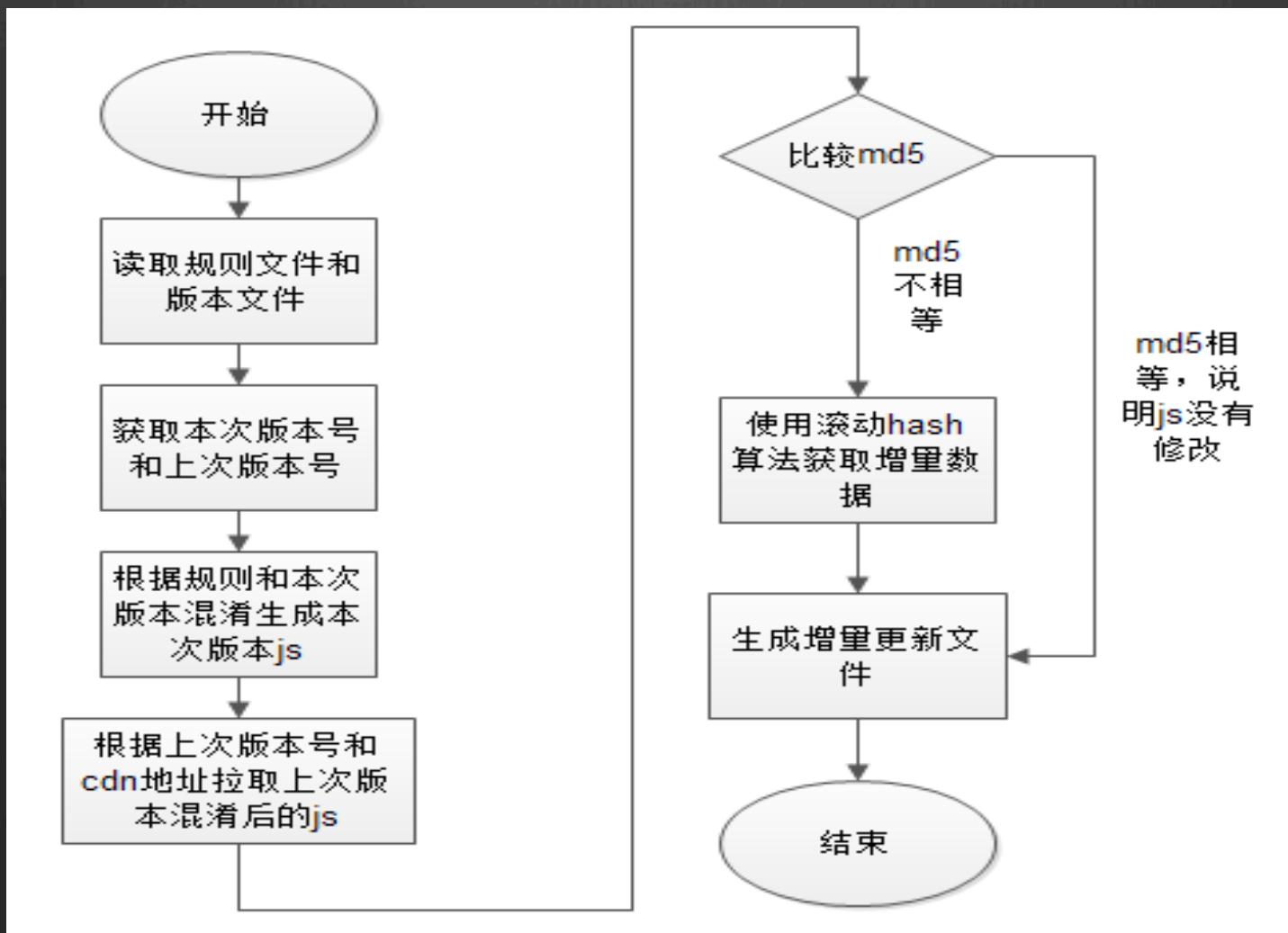
# js增量更新算法设计

```
/* @param source是上一个版本内容 ,  
@param trunkSize是块大小 ,  
@param checkSumCode是两个版本间的增量文件数组  
function merge(source,trunkSize,checkSumCode){  
    var strResult="";  
    for(var i=0;i<checkSumCode.length;i++){  
        var code=checkSumCode[i];  
        if(typeof (code)=='string'){  
            strResult+=code;  
        }  
        else{  
            var start=code[0]*trunkSize;  
            var strSize=code[1]*trunkSize;  
            var oldcode=source.substr(start,strSize);  
            strResult+=oldcode;  
        }  
    }  
    return strResult;  
}
```

# 浏览器端更新流程



# server端实现之打包工具实现



# server端实现之打包工具实现

## 增量文件和全量文件大小对比

名称	修改日期	类型	大小
page	2013/11/27 16:13	文件夹	
auto.min.css	2013/11/27 16:13	层叠样式表文档	59 KB
base-201310300008_2013112700009.js	2013/11/27 16:13	JS 文件	6 KB
base-2013112700009.js	2013/11/27 16:13	JS 文件	53 KB

增量文件内容接图：

EditPlus - [E:\newworkspace\webapp\_infoapp\auto\autolib\touch\release\201311270009\base-201310300008\_201311270009.js]

文件(F) 编辑(E) 视图(V) 搜索(S) 文档(D) 项目(P) 工具(T) 窗口(W) 帮助(H)

1 | {"modify":true,"chunkSize":12,"data":[[0,176],"A++]}function",[177,4],"(N[c(t)]||[],[182,36]," r=c(t),s=N[r]||N[r]=[],[220,26]," ){de

# 打包工具实现方式的优缺点

优点：

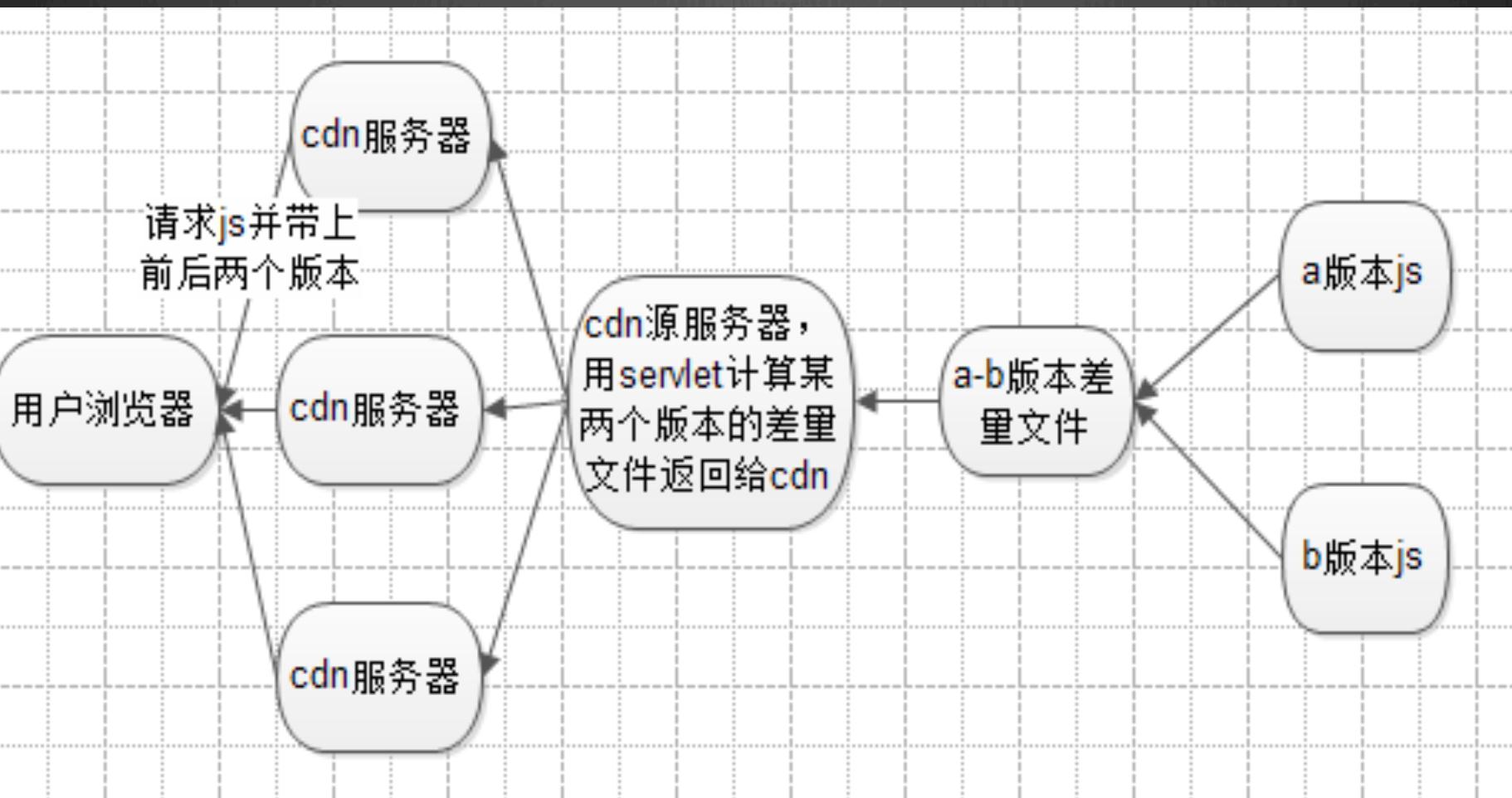
离线完成，然后直接上传cdn，跟传统方式没有区别

缺点：

只能跨一个版本实现增量更新

对回退版本是需要全量更新

# 具体实现改进-servlet代理



# 具体实现-servlet代理

优点：

1. 可跨多个版本增量更新
2. 回退版本也可以增量
3. 无需手动生成增量文件

缺点：

需要自己能控制cdn源服务器

对cdn的push是被动的

# 手机腾讯网增量更新接入方案

## MT—手机腾讯网前端统一框架

特点：

1. 基于AMD的js模块管理加载器
2. 无缝接入本地存储，js增量更新，兼容统一版本和单个js文件单个版本两种版本管理方式
3. mtbuild增量文件生成工具，infocdn手腾js增量更新cdn源

# 第三方模块管理工具插件

**seajs:**

storeinc , seajs增量更新插件,配套spm-storeinc-build来实现打包混淆

**requirejs:**

rstoreinc , requirejs增量更新插件 , 配套有修改后的混淆打包工具r.js

**增量文件生成代理:**

php 版 : storeinc.php

Nodejs版: storeincServer.js

**纯算法版 :**

makeinc.js,merge.js

# 我们正在筹建的一个开源项目

网址：

<http://mt.tencent.com>

Github:

<https://github.com/mtjs/mt>

# 实战效果

目前接入增量更新的webapp:

手腾掌上车典

手腾秀车

手腾nba

手腾爱电影

手腾个人中心

手腾阅读

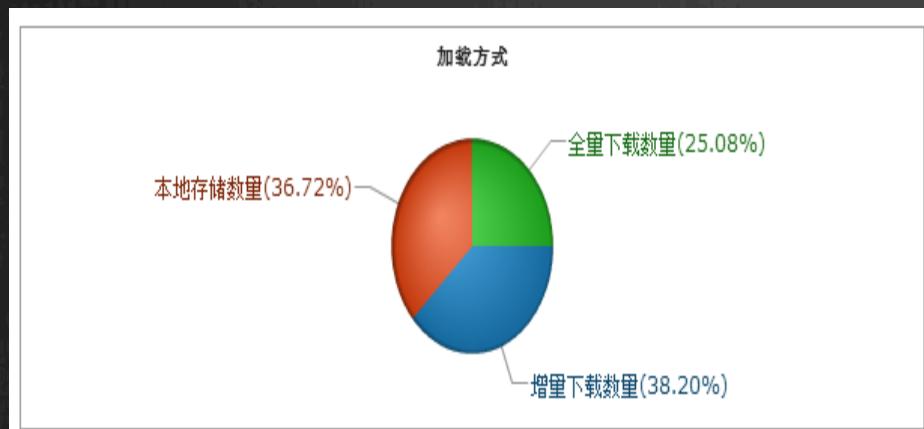
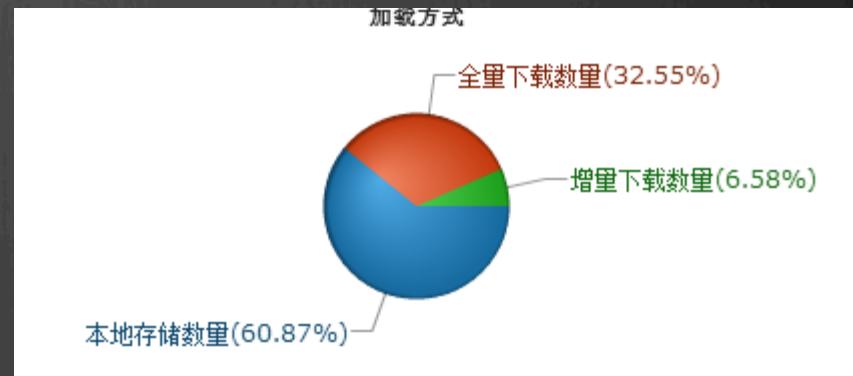
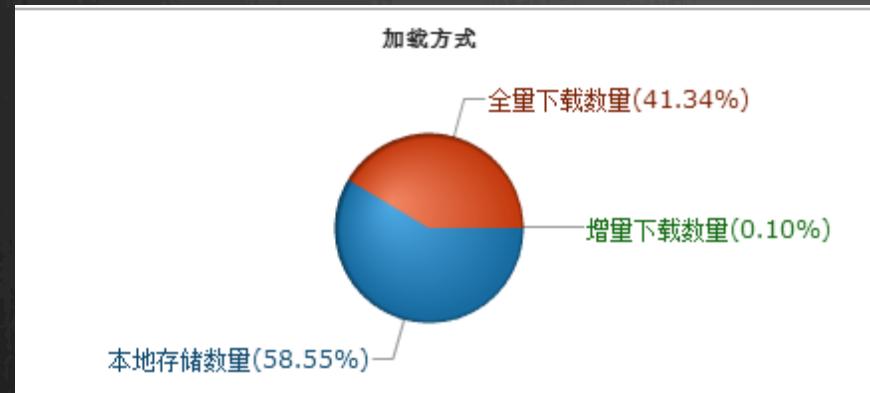
手腾体育猜图。 . . .

详见<http://4g.qq.com>里的轻应用



# 实战效果

秀车，掌上车典，掌上NBA几个webapp的js资源请求方式占比



# 实战效果

2014.3.1-2014.3.31手腾的几个webapp

js请求：2亿多次

本地存储读取：1千5百万

全量请求：4千万次

增量更新请求：2千3百万次

平均每个增量命中比全量节省10k数据，通过增量更新我们大概为用户节省流量：

$$(23000000 * 10) / (1024 * 1024) = 219G$$

END

无更新不下载！！！

Q&A