Google
Developer
Day2009

设计高效可扩展的OpenSocial应用

王超 (Jacky Wang)
2009年6月5日

Google
Developer
Day2009

“Improve latency is really, really important.”

+ 0.5秒的延迟可令Google损失20%的搜索流量

Marissa Mayer, Google

“Even very small delays would result in substantial and costly drops in revenue.”

+ 0.1秒的延迟可令亚马逊损失1%的销量

Greg Linden, Amazon

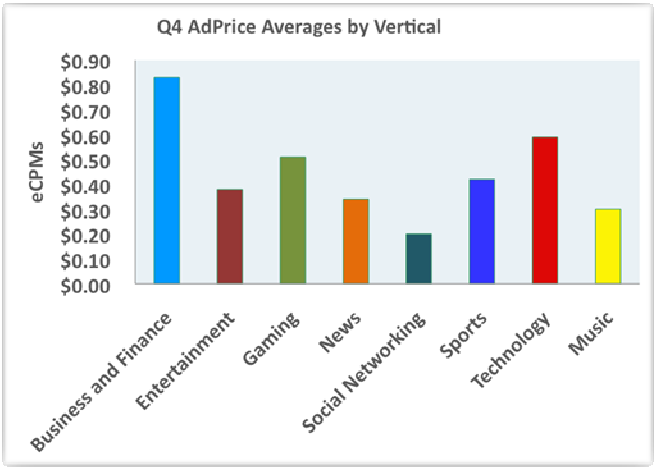
“If you make a product faster, you get that back in terms of increased usage.”

- 减少30K的页面大小令谷歌地图在三周内增长30%

Marissa Mayer, Google

什么是一个成功的社交应用？

- **利**润率单薄
  - 对广告而言，社交类应用的**eCPM要低于一般网站的平均水平**



以上图片来自于 Pubmatic AdPrice Index 2008: http://pubmatic.com/adpriceindex/index.html

- **海量的用**户
  - **直接与社交平台上已注册的上千万用**户交互
  - **通**过病毒式营销，让应用的**使用量得到快速增**长

ID	应用名称	总用户数	活跃用户数	活跃用户%	昨日新增用户
1	开心农场	15,484,840	3,657,336	24%	145,165
2	开心农民	3,477,809	1,219,674	35%	39,801
3	阳光牧场	3,896,120	1,187,551	30%	45,831
4	测试	4,810,747	339,487	7%	31,909
5	踢屁屁	1,545,408	320,470	21%	64,191
6	我的餐厅	859,740	314,569	37%	33,805
7	荣光医院	707,482	302,090	43%	34,723
8	小游戏中心	5,704,370	216,869	4%	3,417
9	篮球巨星	841,457	176,706	21%	103,928
10	德克萨斯扑克	2,410,225	168,716	7%	26,542
11	抢床位	2,491,216	155,915	6%	1,649
12	欢乐卡片	1,204,501	96,853	8%	7,706
13	新同居时代	2,515,382	91,298	4%	0
14	偷心贼	1,283,622	70,466	5%	18,068
15	开心鱼塘	162,931	64,884	40%	63,668
16	全民斗地主	3,041,126	61,076	2%	430

以上图片来自于 校内应用研究: http://www.xnapps.com/apps/active

低利润率 × 大量的用户 = 小提高，大收获！

Google Developer Day2009

讲座大纲

- 衡量大规模应用中局部优化带来的好处
- OpenSocial应用示例：Quartermile
  - 检验不同性能优化方法的成果
  - 真实的性能数据
- 目标
  - 使用户体验得到速度上的提升
  - 降低多人大规模社交类应用的运行成本
  - 展示OpenSocial新规范中的重要功能

实例介绍: Quartermile

Google
Developer
Day2009

Quartermile架构

- OpenSocial应用
- 后端架构
  - 使用Google App Engine搭建
  - 开放一组自定义的JSON RPC API
  - 处理来自多个社交平台上的用户数据
- 前端架构
  - 小应用运行于多个平台上
  - 使用JavaScript库进行RPC远程调用
- 演示: http://www.youtube.com/watch?v=f3oP5xYxeMg
- 源代码: http://bit.ly/quartermile-src
- XML Spec: http://bit.ly/quartermile-app

测试Quartermile

- 不同类型的请求
  - Quartermile API
  - OpenSocial API
  - 应用元素：图片、JavaScript，CSS
- 衡量标准
  - 每次访问的带宽占用（KB）
  - 每次访问的请求（请求次数）
  - 每次访问的延时（毫秒/ms）
  - CPU时间（megacycles）
- 每一种请求都很重要，应该被单独测试
- 对这些请求的控制程度各有不同

Quartermile数据统计

粗略实现

	Bandwidth / PV (kb)	Requests / PV	延时（毫秒）	CPU时间
Quartmile API 请求	4.21	4	6490	2078
OpenSocial API 请求	1.45	2	804	0
应用元素	135.6	22	3152	0

Quartermile粗略实现的运行成本

速度	粗略实现
延时（应用）	2730 MS
延时（页面）	3536 MS
请求数	26
YSlow! SCORE	72

成本	数量	单价	总计
带宽	5582 GB	$0.12	$669.84
CPU时间	1859 HR	$0.10	$185.90
以11QPS计 每月成本			$855.74

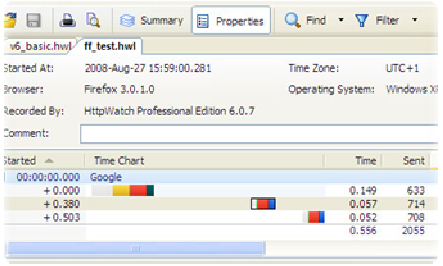
Web开发最佳实践

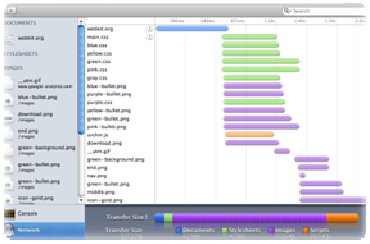
小应用也是页面的一种！

Google
Developer
Day2009

Web开发最佳实践

- 我们主要关注以下几个方面：
  - 应用大小
  - 发送请求的数量
  - 初始的载入延迟
- 衡量手段
  - Firefox: Firebug & YSlow!
  - IE: HttpWatch
  - Safari: Web Inspector





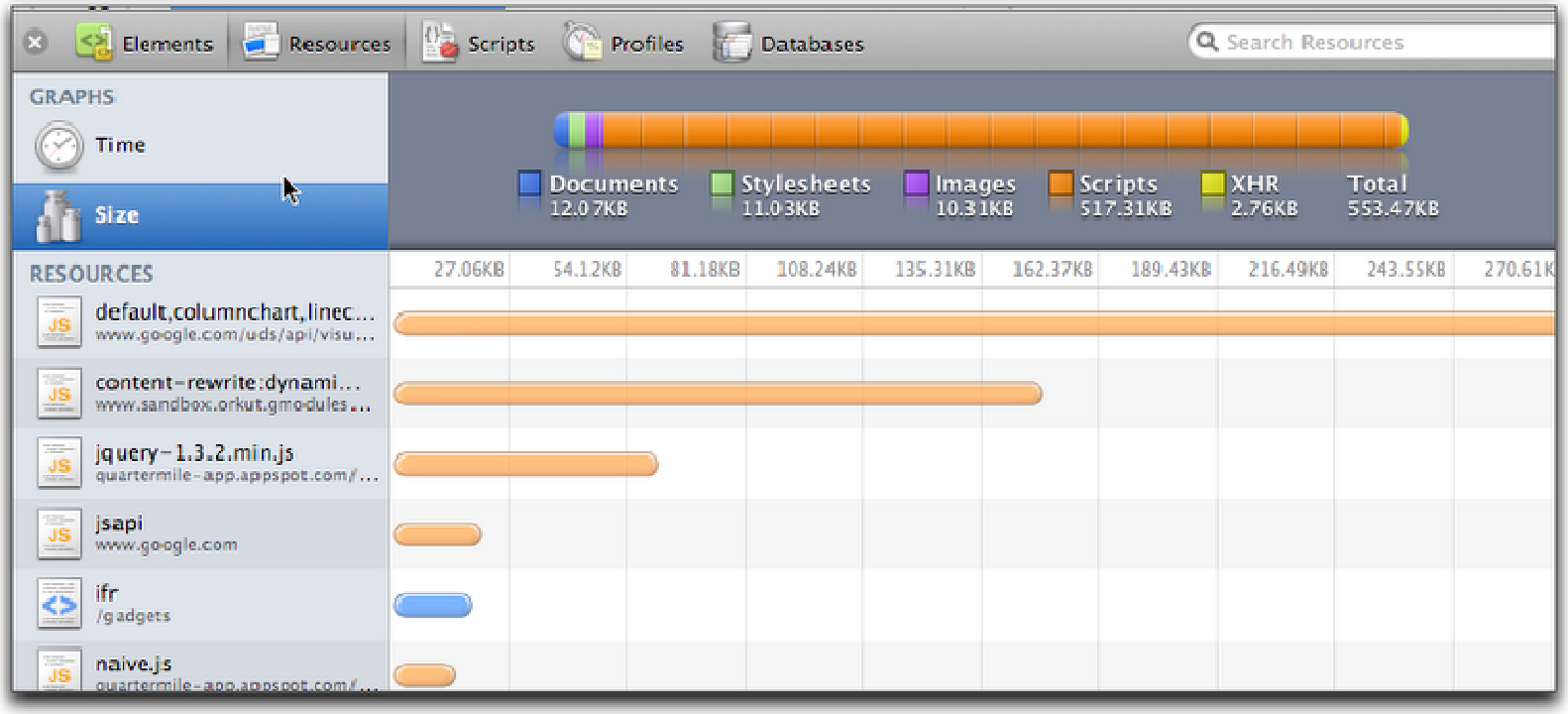
减小JavaScript和CSS文件的大小

- 将JS文件和CSS文件分别串接起来
  - 减少HTTP请求的数目
  - 降低响应延迟
- 混淆、压缩JS文件和CSS文件 (YUI Compressor / JSMin)
  - 减小下载数据量

```
var foo = ['1', '2', '3', '4'];
for (var i = 0 ; i < 4 ; i++) {
  alert(foo[i]);
}
```

```
var f=['1','2','3','4'];for(var i=0;i<4;i++){alert(f[i]);}
```

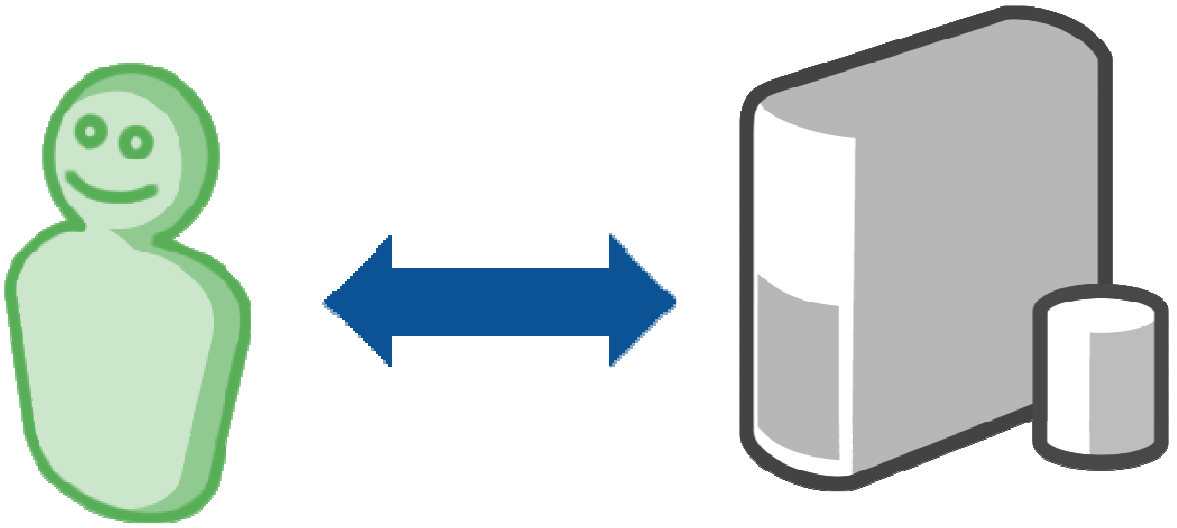
确定应用的大小


Elements
Resources
Scripts
Profiles
Databases
Search Resources
GRAPHS
Time
Size
Documents
12.07KB
Stylesheets
11.03KB
Images
10.31KB
Scripts
517.31KB
XHR
2.76KB
Total
553.47KB
RESOURCES
27.06KB
54.12KB
81.18KB
108.24KB
135.31KB
162.37KB
189.43KB
216.49KB
243.55KB
270.61K
default,columnchart,linec...
www.google.com/uds/api/visu...
content-rewrite:dynami...
www.sandbox.orkut.gmodules...
jquery-1.3.2.min.js
quartermile-app.appspot.com/...
jsapi
www.google.com
ifr
/gadgets
naive.js
quartermile-app.appspot.com/...

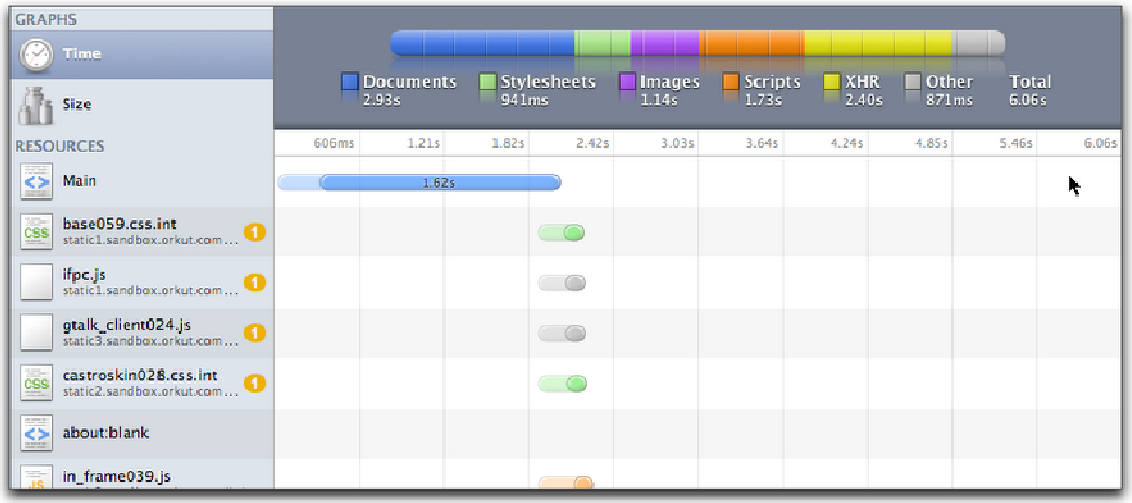
Quartermile的改进

	粗略实现	精简后	节省了
每次请求数据量	33 KB	15 KB	-54%
每月数据量	930 GB	420 GB	
$	$111.60	$50.40	-$61.20

衡量延迟 – 感觉 v.s. 现实

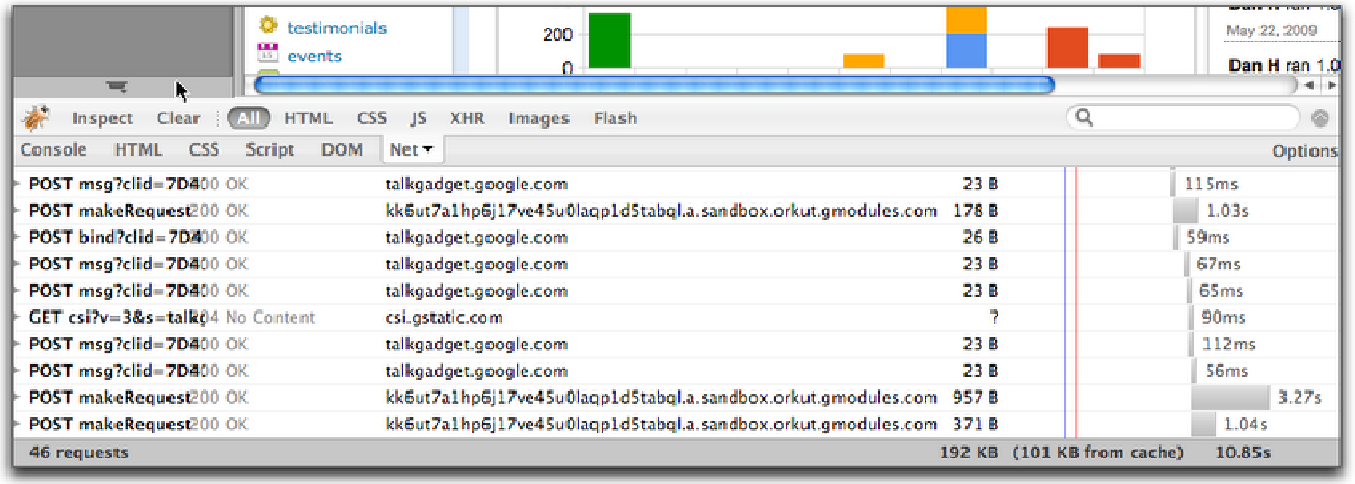


衡量延迟

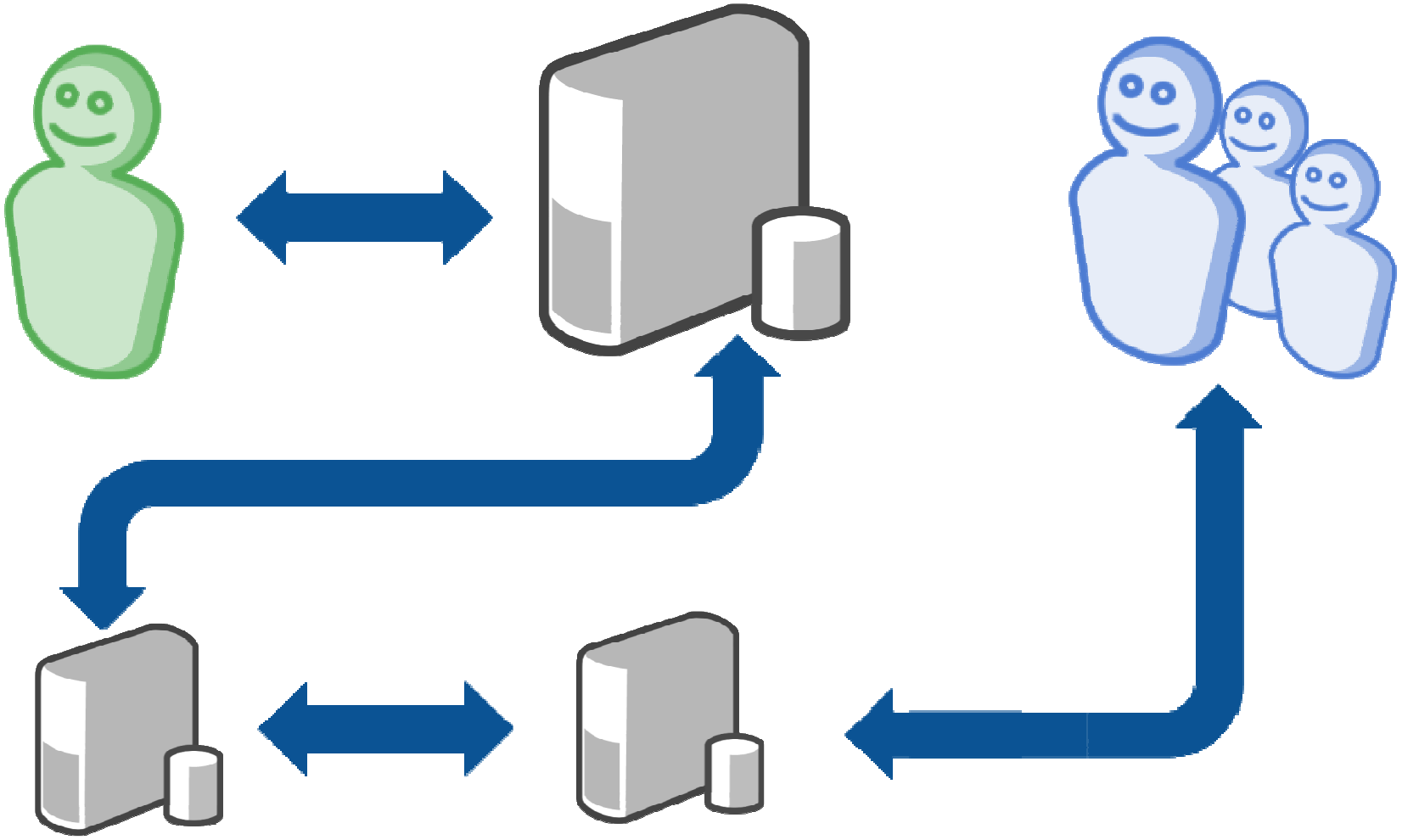


Web Inspector

Firebug

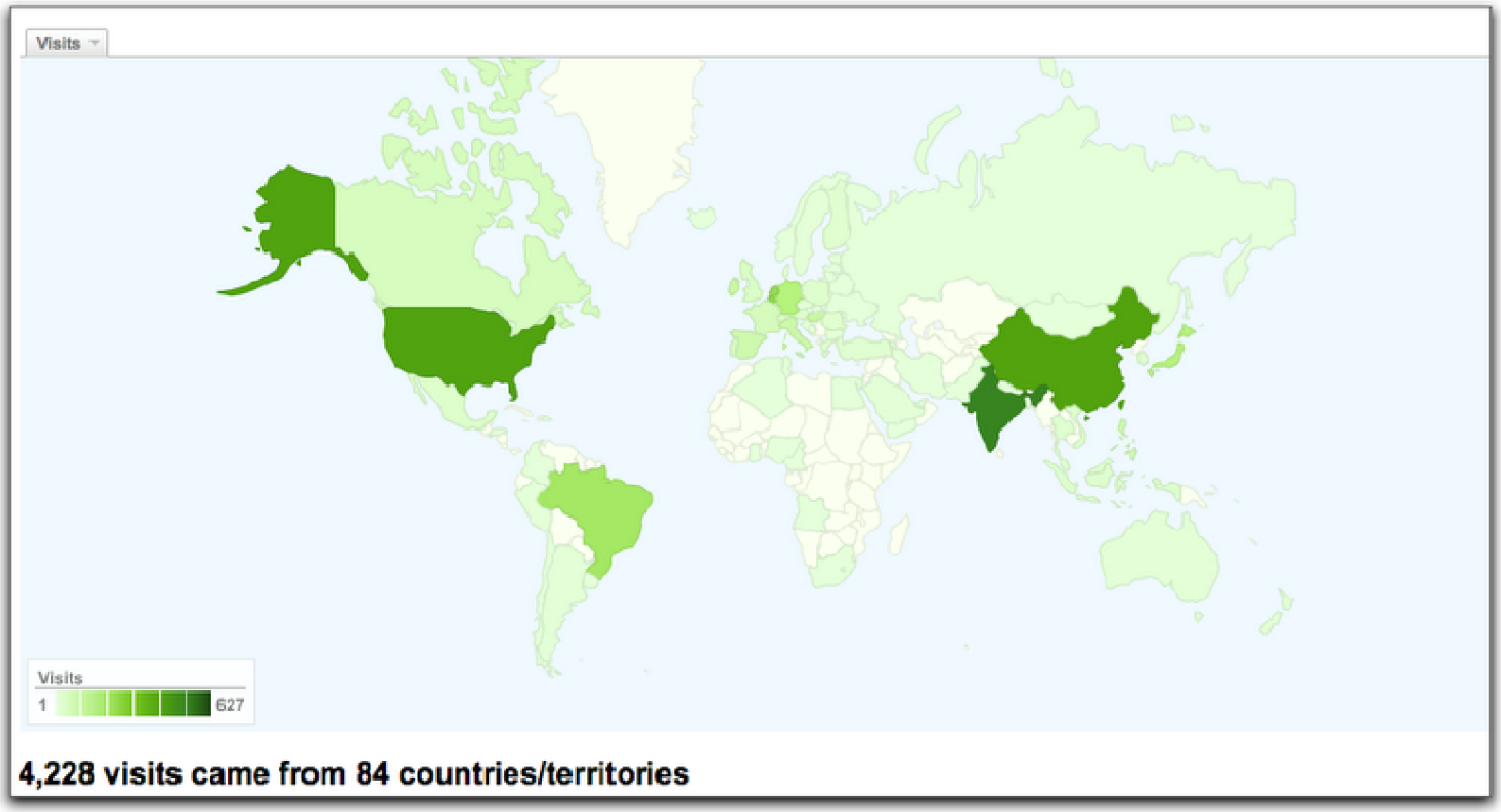


延迟不是恒定不变的数值，需要多次测量



减少延迟可以提高用户满意度，并提高参与性

- 不同的地点：至少是应用用户最多的几个国家、地区



自动收集延迟信息

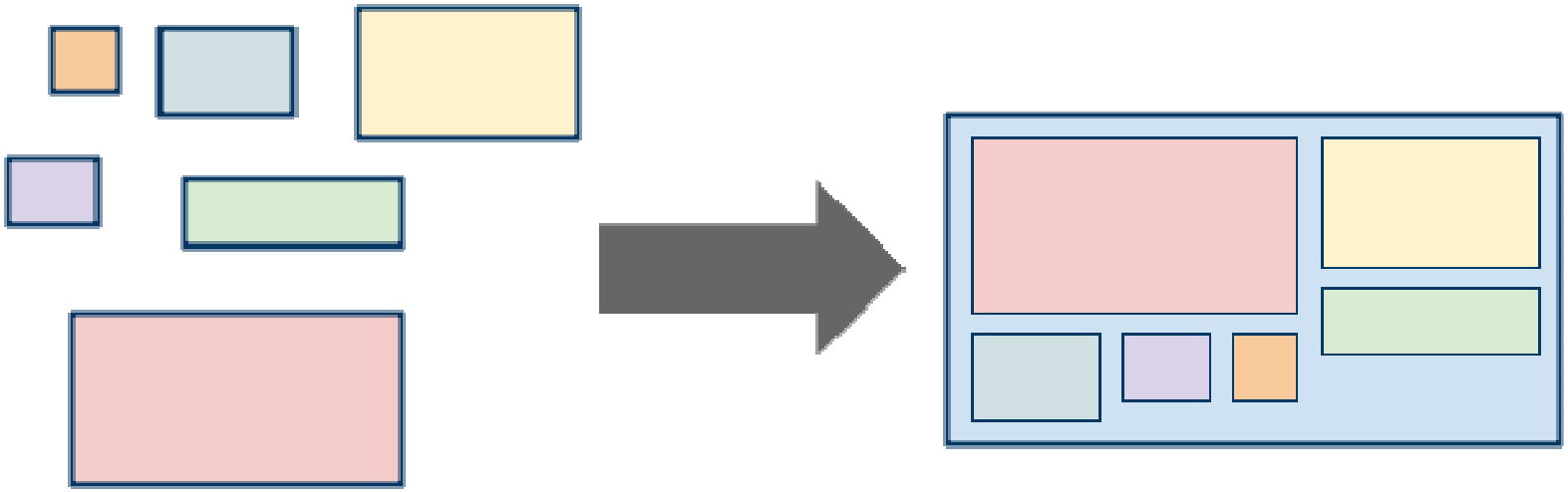
- 比如，用以下JavaScript来统计：

```
var startTime = new Date();
var imgElement = new Image()
imgElement.onload = function() {
  var endTime = new Date();
  var latency = endTime. getTime() - startTime.getTime();
  // 将延迟数据回报给服务器
}
imgElement.src = "http://your.server.com/ping.gif";
```

- 或者，使用Google Analytics

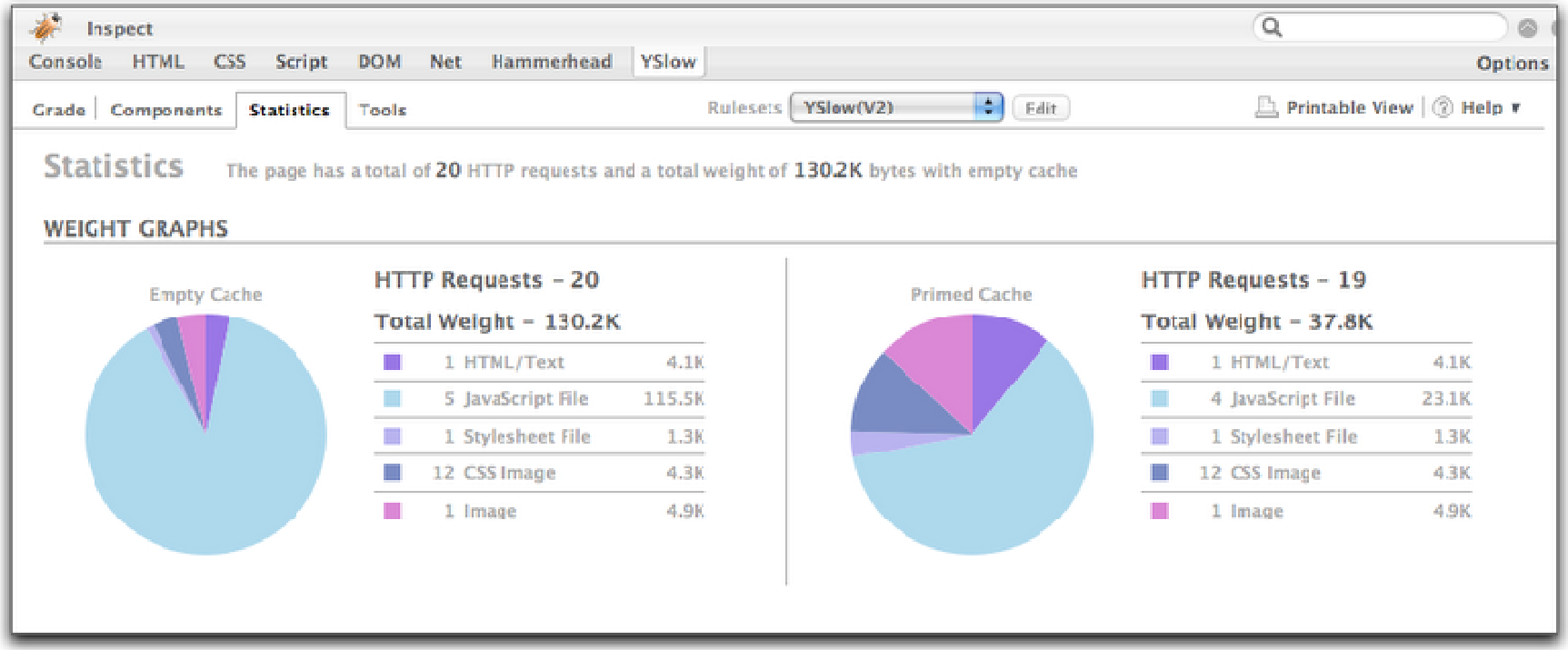
图片拼接 (CSS Sprites)

- 将很多小图片拼接进一个大图片(Sprite)当中去，然后利用CSS来选择性地展示其中的某一部分
  - 减少请求数
    - 浏览器可能存在限制：每个host只有2个并发连接
  - 减少延迟



如何知道请求的个数？

- 用YSlow!统计



Quartermile的改进

	粗略实现	CSS Sprites 精简后	Palette Fix 精简后	节省了
延时 （应用）	592 MS	378 MS	325 MS	-45%
数据大小	9.59 KB	11.15 KB	5.82 KB	-39%
请求数	15	1	1	-93%
$	$27.30	$38.28	$16.80	-$10.5

调整缓存的头描述

- 对于静态数据，使用浏览器缓存
  - **减少**总体带宽和请求数
  - **降低响**应延迟
- Apache **配置**实例：

```
<FilesMatch "\.(css|js|gif|jpe?g|png)$">
  Header set Cache-Control "max-age=290304000, public"
</FilesMatch>
```

- **通**过制定请求参数（常见为版本号）来强迫刷新缓存

```
http://example.org/css/style.css?v=3
```

服务器辅助优化

让应用容器来帮你的忙！

Google
Developer
Day2009

服务器辅助下的自动优化

- 社交类应用：小提高，大收获
- 社交网络(SNS)承载了大量的应用：小改进，大不同！

- 服务器辅助优化的优势：
  - 作为一个大规模部署的网站，在网络架构上常常更具延展性
  - 由Gadget Spec渲染出的HTML是服务器可控的

静态内容代理

- 社交网络可以帮助应用分担一部分数据流量
  - 使用内容分发网络(CDNs)
  - 广泛部署在网络边界(Edge)，稀释流量

- 对于距离服务器非常远的客户端尤其有益

```
var div = $("#flashcontainer");
var url = gadgets.io.getProxyUrl("http://me.com/flash.swf");
gadgets.flash.embedFlash(url, div, 10);
```

内容重写

- SNS可以控制输出的HTML代码
  - CSS优先，JavaScript置后
  - 连接并精简静态文件内容
  - 重写静态内容的URL，以方便前述代理
- 在Gadget Spec中可以实现的缓存控制

```
<Module>
  <ModulePrefs>
    <Optional feature="content-rewrite">
      <Param name="include-urls"></Param>
      <Param name="exclude-urls">.*</Param>
      <Param name="include-tags"></Param>
    </Optional>
...
```

OpenSocial最佳实践

量身定制之0.9版发布 – 社交应用提速！

Google
Developer
Day2009

数据请求批量化 (Batching)

- 一次API调用 = 一个HTTP请求

```
osapi.people.getViewer().execute(onVwr);
osapi.people.getOwner().execute(onOwnr);
osapi.people.getViewerFriends().execute(onFrnd);
osapi.people.getOwnerFriends().execute(onOFrnd);
```

- 争取在一个HTTP请求中得到尽量多的数据

```
var batch = osapi.newBatch()
  .add("vwr", osapi.people.getViewer())
  .add("vfd", osapi.people.getViewerFriends())
  .add("owr", osapi.people.getOwner())
  .add("ofd", osapi.people.getOwnerFriends())
  .execute(onData);
```

- 对缓存的影响 – RESTful v.s. RPC

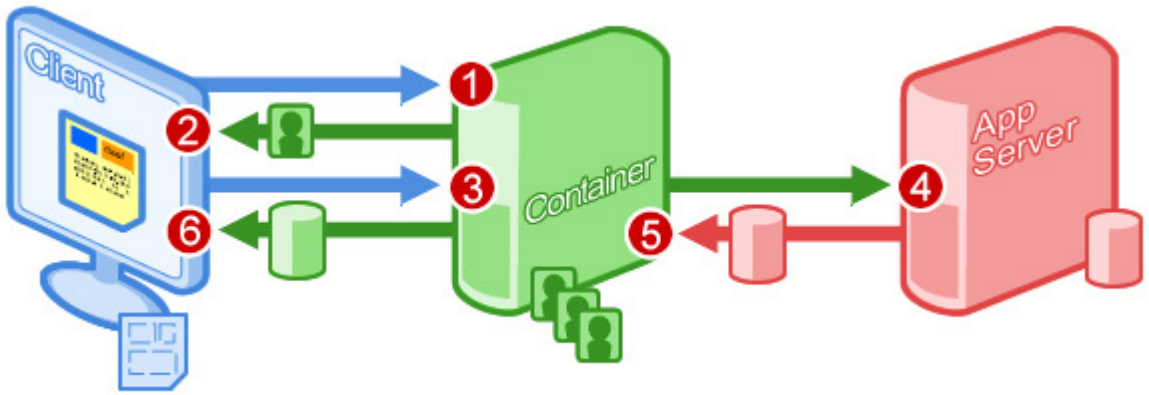
Quartermile的改进

速度	粗略实现	批量化	差值
延时（应用）	2730 MS	2743 MS	+13 MS
延时（页面）	3536 MS	3504 MS	-32 MS
请求数	26	24	-2
YSlow! SCORE	72	74	+2

成本	数量	单价	总计	变化
带宽	5576 GB	$0.12	$669.12	
CPU时间	1499 HR	$0.10	$149.90	
以11QPS计 每月成本			$819.02	-$36.72

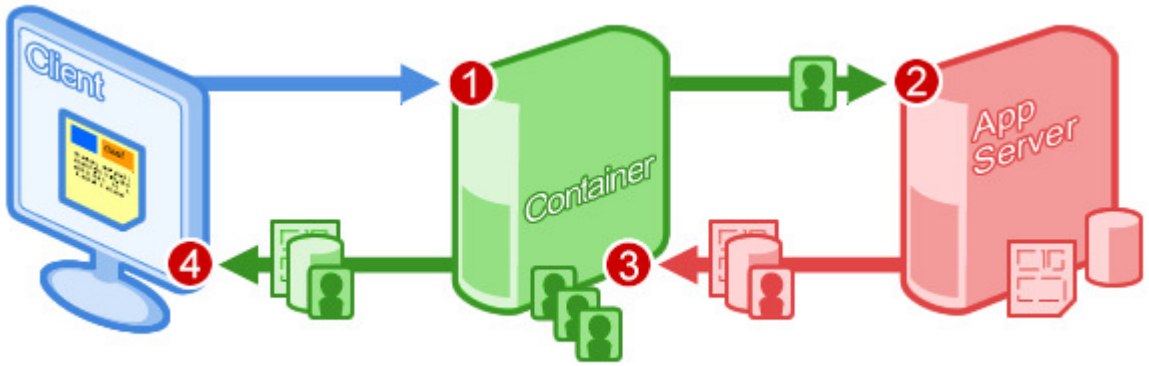
数据流水线 + 内容代理

- 粗略实现中，有24个HTTP请求！
- 能不能减少请求的数量？
- 数据流水线(Data Pipelining) + 内容代理(Proxied Content)



数据流水线 + 内容代理

- 利用新加入OpenSocial 0.9标准的“数据流水线”功能，我们可以定义哪些社会化数据将被推送到应用服务器上
- 应用服务器处理这些数据，并返回渲染好的HTML以供显示
- 现在已经在iGoogle和Orkut的安全沙盒中部署，即将在更多的OpenSocial容器（社交类网站）上推广该功能



数据流水线 + 内容代理

```
<Module>
  <ModulePrefs ... etc .../>
  <Content type="html" view="profile" authz="signed"
           href="http://yoursite.com/proxied.php">
    <os:ViewerRequest key="vwrData" fields="id, displayName"/>
    <os:OwnerRequest  key="ownData"/>
    <os:PeopleRequest key="ownFriends"
                      userId="@owner"
                      groupId="@self"/>
  </Content>
</Module>
```

```
<?php
$postData = json_decode(file_get_contents("php://input"));

echo "<h1>你好，{$postData['vwrData']['name']}</h1>";
echo "以下列出了{$postData['ownData']['name']}的朋友们:";
echo "<br/>";
foreach ($postData['ownFriends'] as $friend) {
   echo "{$friend['name']}<br/>";
}
```

Quartermile的改进

速度	粗略实现	流水线 + 代理	差值
延时（应用）	2730 MS	1094 MS	-1636 MS
延时（页面）	3536 MS	2861 MS	-675 MS
请求数	26	20	-6
YSlow! SCORE	72	76	+4

成本	数量	单价	总计	变化
带宽	3705 GB	$0.12	$444.60	
CPU时间	902 HR	$0.10	$90.20	
以11QPS计 每月成本			$534.80	-$320.94

数据失效范型 (Invalidation Pattern)

- 于OpenSocial 0.9版中引入的新开发方式
- 当应用服务器的内部状态发生变化的时候，可以通过RESTful / RPC方式发送请求，来使过去的数据失效
  - 可以基于用户ID或URL
  - 应用程序的ID是通过失效请求中的2-Legged OAuth取得

```
POST /api/rest/cache/invalidate
HOST opensocial.example.org
Content-Type: application/json
{
  invalidationKeys: [
    "http://www.myapp.com/gadgetspec.xml",
    "http://yoursite.com/proxied.php"
    "user:123"]
}
```

优化数据存储结构

结构、位置、性能

Google
Developer
Day2009

优化数据存储结构 – 通用优化

- 考虑对朋友列表的数据库做JOIN操作？NO!
- 如果不使用Google App Engine，需要仔细计划：
  - 主 / 从数据库分布
  - 表分割
- 使用Memcache来缓存数据
  - 只缓存常被用到的，提高命中率
  - 只缓存需要经过复杂计算的结果，提高效率
- 考虑将最常用的数据打包成JSON格式，而不是传统的关系数据库条目

优化数据存储结构 – 后台处理

- 考虑用后台处理来优化性能
  - 不是所有的信息都需要最新的
  - 即用即取，不影响用户交互
  - 对于类似“你所有朋友的新鲜事”的请求特别有效
- 可以采用开源的队列系统实现
- 或采用App Engine，有cron.yaml文件：

```
cron:
- description: process activities entries
  url: /tasks/processActivities
  schedule: every 1 minutes
```

以Quartermile为例，设计数据存储结构

- **核心原**则
  - 应用的核心功能设计应脚踏实地，很可能受限于存储结构
  - **事先**给用户的硬性限制，胜过糟糕的用户体验

- 现状：类似“**你朋友的…”操作**，开销是非常大的！
  - Orkut.com**上**每人可以有1,000**个好友**
  - MySpace.com**上**每人可以有100,000+**个好友**…

- **解决方案**：**区分**优先级，分而治之
  - **不是**100,000**个好友都需要看到你的运**动量
  - **分成更**细的小组：“**更好的好友**” – **加**强了凝聚力！

组员状态 – 实时更新

- 组有多大？
  - 目标：一个数据库操作，可以取回所有组员在某周内的运动数据
    - App Engine 每次最多返回1000个条目
    - 一个条目对应一次运动
    - 每天2~3次运动 = 每周15~20次运动
  - 每组1000 / 16.6 = 60个用户

- 实时发送请求，获得最新的数据

泛好友状态 – 后台处理

- 运行很慢！
  - 获取好友列表
  - 对每个好友，考察最近的新鲜事
  - 按时间排序
- 后台处理
  - 应用服务器通过2-Legged OAuth获取好友列表及新鲜事
  - 计算并排序
  - 将计算结果保存起来以备后用（可能会是过期数据）
- 保存在哪儿？

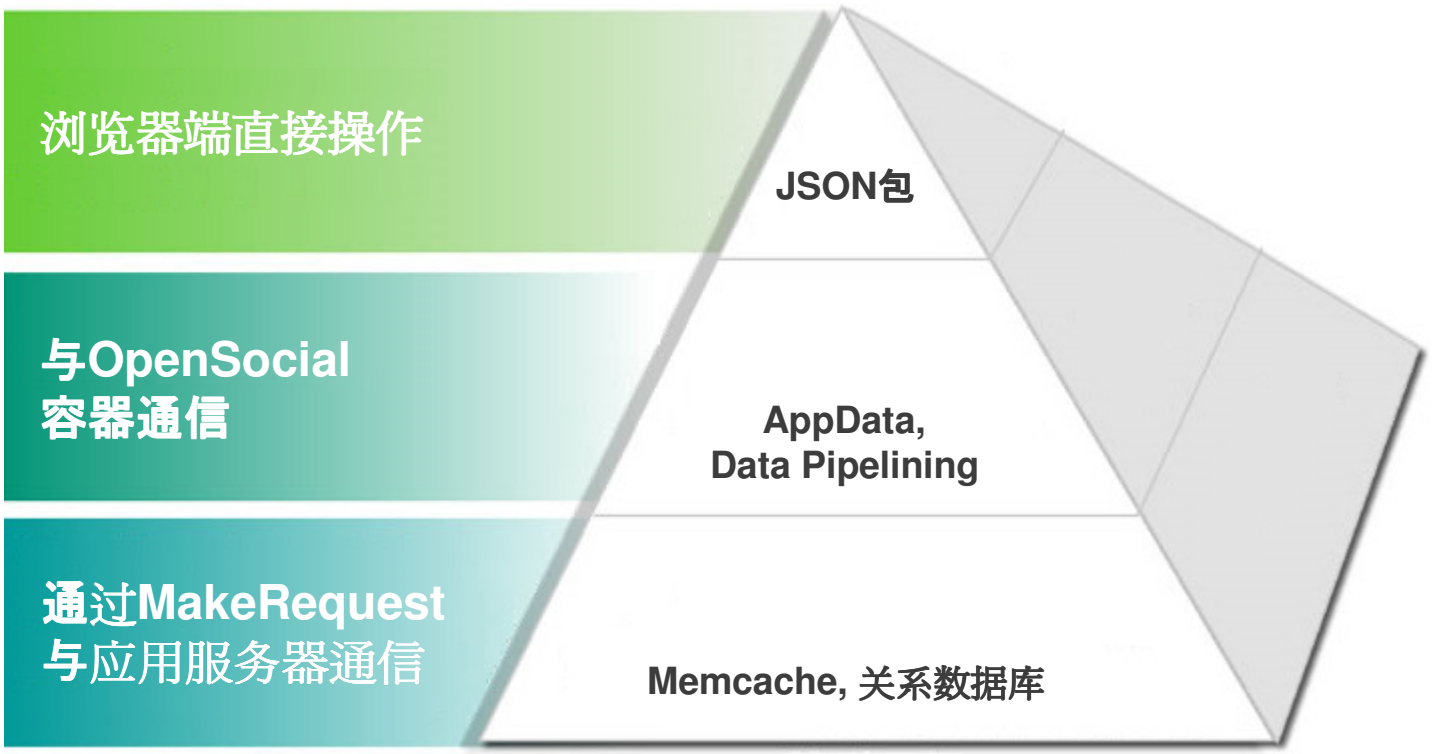
AppData再认识

- AppData可能是OpenSocial规范中被误解误用最多的部分！

- 数据存储：“键 – 值”对
  - 公开的存储，不能放置保密内容
  - 浏览器可以用JavaScript访问和修改，因此不可信
  - 快！应用直接与OpenSocial容器通信，缓存缓慢数据的好地方

- 流程
  - 应用服务器后台进行复杂的数据计算
  - 将计算结果推送入OpenSocial容器的AppData
  - 应用渲染时，直接从AppData获取数据，避免了应用服务器操作

数据存储结构的快捷性分层



容器强制优化

自发诚信与客观要求

Google
Developer
Day2009

容器强制优化

- 应用实现的水平参差不齐
  - **容易引入**错误，坏的情况下可能使整个页面停止响应

- **容器**：
  - 让每个应用更快 = **容器更快**
  - **每个**应用只能在允定的范围内影响用户响应延迟
  - **不能**仅仅依靠应用开发者的诚信，容器需要做出一些限制

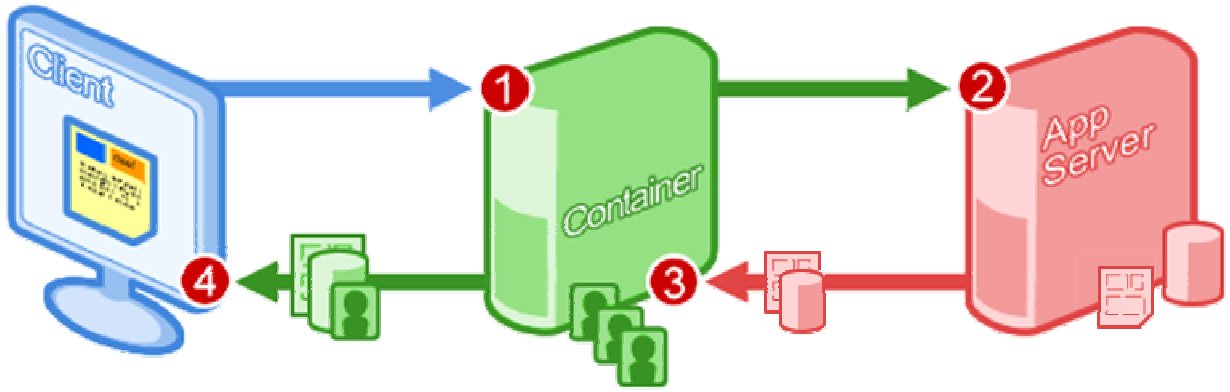
iGoogle的延迟惩罚

- 应用目录（用户在其中搜索并安装有趣的应用）将很快把延迟作为应用排名的参数之一
- 还有其他的隐性惩罚措施

- **演示**：
  - iGoogle上各个应用的载入速度差别：
    http://www.youtube.com/watch?v=TkPnT2gjx0k

Orkut的Profile页面只允许基于模板的应用显示

- Profile页面
  - Orkut上被访问最多的页面，用户也希望在页面上看到应用
- OpenSocial 0.9版引入了模板系统
  - 可以显示社会化信息，如Friends/Activities， 及AppData
  - 不能从应用服务器获取额外的信息
- Profile页面得到了极快的渲染速度
- 是AppData缓存的一个很好应用



Orkut模板范例

```
<Module>
  <ModulePrefs title="模板范例">
    <Require feature="opensocial-data" />
    <Require feature="opensocial-templates">
      <Param name="process-on-server">true</Param>
    </Require>
  </ModulePrefs>
  <Content type="html" view="profile"><![CDATA[
    <script type="text/os-data">
      <os:PeopleRequest key="friends"
                        userId="@viewer"
                        groupId="@friends"/>
    </script>
    <script type="text/os-template">
      <div repeat="${friends}">${Cur.name.givenName}</div>
    </script>
  ]]></Content>
</Module>
```

总结

比较前述所有的优化方法

	应用大小	发送请求的数量	载入延迟
减小JS和CSS文件大小	✓	✓	✓
图片拼接	✓	✓	✓
调整缓存的头描述	✓	✓	✓
静态内容代理			✓
内容重写	✓	✓	✓
数据请求批量化		✓	✓
数据流水线		✓	✓
数据失效泛型		✓	✓
存储结构优化			✓
后台处理			✓
AppData缓存			✓
受限的Profile页面(模板系统)			✓

比较：Quartermile最优实现

速度	粗略实现	最优实现	差值
延时（应用）	2730 MS	833 MS	-1636 MS
延时（页面）	3536 MS	2686 MS	-675 MS
请求数	26	6	-20
YSlow! SCORE	72	89	+17

成本	数量	单价	总计	变化
带宽	3616 GB	$0.12	$433.92	
CPU时间	963 HR	$0.10	$96.30	
以11QPS计 每月成本			$530.22	-$325.52

感谢您参加本场讲座！

提问 / 回答

设计高效可扩展的OpenSocial应用

更多信息，请访问code.google.com

Google
Developer
Day2009

Google
Developer
Day2009