

大流量Web系统性能优化实践

君山

2015.10.15

Geekbang>

极客邦科技

全球领先的技术人学习和交流平台

扫我，码上开启新世界



Geekbang>

InfoQ | EGO NETWORKS | StuQ

InfoQ

专注中高端技术人员
的社区媒体

EGO NETWORKS

EXTRA GEEKS' ORGANIZATION
高端技术人员
学习型社交网络

StuQ

实践驱动的IT职业
学习和服务平台



促进软件开发领域知识与创新的传播



实践第一 案例为主

时间：2015年12月18-19日 / 地点：北京·国际会议中心

欢迎您参加ArchSummit北京2015, 技术因你而不同



ArchSummit北京二维码



[北京站]

2016年04月21日-23日



关注InfoQ官方信息
及时获取QCon演讲视频信息

关于我

- 许令波(君山)
- 在阿里6年，做了4件还不错的事情
- 商品详情、店铺、图片空间TL
- 关注大流量web系统的架构和性能优化工作。

目录

- 这些年的挑战
- 我们走过的路
- 我们的经验

流量爆发增长



系统还比较脆弱



环境造就了技术



搞不好就淹死了



业务爆发增长



只有在这里才能遇到



遇到的挑战

- 流量爆发增长带来机器的成倍增加，系统必须要能水平扩展
- 流量的峰值（秒杀），单商品或者用户维度会出现热点，给cache带来瓶颈
- 大面积的攻击，如何区分正常流量防止误杀
- 复杂的业务逻辑给系统系统的耦合度和数据的分类更加困难

■ 我们走过的路

- 系统代码层面的优化
- 架构优化
- 链路优化

代码级优化



ZProfiler

ASWP

代码优化实践：模板引擎的热点

- Velocity是动态解释性语言,执行效率较差
- 页面复杂,反射调用非常多
- 发现模板渲染占用了60%以上的CPU时间。
- 整个页面输出比较大,平均在100KB左右。

Method	Count	%
com.alibaba.citrus.turbine.pipeline.valve.RenderTemplateValve.renderTemplate(String, Context, TurbineRunDataInternal)	130,468	82 %
com.alibaba.citrus.service.template.impl.TemplateServiceImpl.writeTo(String, TemplateContext, Writer)	130,208	82 %
com.alibaba.citrus.service.velocity.impl.VelocityEngineImpl.writeTo(String, TemplateContext, Writer)	130,196	82 %
com.alibaba.citrus.service.velocity.impl.VelocityEngineImpl.mergeTemplate(String, Context, Writer, String)	130,196	82 %
com.alibaba.citrus.service.velocity.impl.VelocityEngineImpl.mergeTemplate(String, String, Context, Writer)	130,184	82 %
org.apache.velocity.Template.merge(Context, Writer)	130,056	82 %
com.alibaba.citrus.service.velocity.support.RenderableHandler.referenceInsert(String, Object)	58,455	37 %
com.taobao.cms.client.CMSTool.importRgn(String, long)	43,218	27 %
com.taobao.security.util.SecurityUtilLib.richtext(Object)	8,052	5 %
com.alibaba.citrus.service.velocity.impl.CustomizedUberspectImpl.getPropertyGet(Object, String, Info)	3,940	2 %
com.alibaba.citrus.service.velocity.impl.parser.ASTStringLiteralEnhanced.value(InternalContextAdapter)	3,340	2 %
com.alibaba.citrus.service.velocity.impl.TemplateContextAdapter.icacheGet(Object)	2,796	2 %
com.alibaba.citrus.service.velocity.impl.parser.ASTStringLiteralEnhanced.value(InternalContextAdapter)	1,020	1 %
com.alibaba.citrus.service.velocity.impl.CustomizedUberspectImpl.getPropertyGet(Object, String, Info)	796	1 %
com.alibaba.service.urlbroker.util.URIBroker.render()	672	0 %

代码优化实践： sketch模板引擎

- 将Velocity模板直接转成Java类去执行, 将Velocity语法转成Java语法
- 将方法的反射调用转成直接Java原生方法调用
- 减少页面大小, 删除空行等无效字符输出
- 将页面中的字符转成字节输出减少编码转换

代码优化实践：class.forName热点

- Class.forName会导致线程block

Total	Function
0.0 ms 0.32 %	▶ groovy/lang/GroovyClassLoader.parseClass(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/Class;
0.1 ms 36.62 %	▼ org/apache/velocity/app/event/EventHandlerUtil.iterateOverEventHandlers(Ljava/util/Iterator;Lorg/apache/velocity/app/event/EventHar
0.1 ms 36.62 %	▼ com/alibaba/citrus/service/velocity/support/RenderableHandler.referenceInsert(Ljava/lang/String;Ljava/lang/Object;)Ljava/lang/Obj
0.1 ms 36.62 %	▼ com/taobao/tbskip/common/webx/util/TolerantControl.render(Ljava/lang/String;
0.1 ms 36.62 %	▼ com/alibaba/turbine/module/TemplateModule.execute(Lcom/alibaba/turbine/service/rundata/RunData;)V
0.1 ms 36.31 %	▼ com/taobao/tbskip/web/common/module/control/upp/UppProPoint.execute(Lcom/alibaba/turbine/service/rundata/RunData;L
0.1 ms 36.31 %	▼ com/taobao/point/platform/service/client/ProPointReadServiceClient.findItemPointFromTair(Lcom/taobao/point/platform/dc
0.1 ms 36.31 %	▼ com/taobao/point/platform/common/tair/MultiClusterPromotionTairManager.get(Ljava/io/Serializable;I)Ljava/lang/Object;
0.1 ms 36.31 %	▼ com/taobao/tair/impl/mc/MultiClusterTairManager.get(ILjava/io/Serializable;)Lcom/taobao/tair/Result;
0.1 ms 36.31 %	▼ com/taobao/tair/impl/DefaultTairManager.get(ILjava/io/Serializable;)Lcom/taobao/tair/Result;
0.1 ms 36.31 %	▼ com/taobao/tair/impl/DefaultTairManager.sendRequest(ILcom/taobao/tair/comm/TairClient;Lcom/taobao/tair/pack
0.1 ms 36.31 %	▼ com/taobao/tair/comm/TairClient.invoke(ILcom/taobao/tair/packet/BasePacket;Ljava/lang/String;)Ljava/lang/Ob
0.1 ms 36.31 %	▼ com/taobao/tair/packet/ResponseGetPacket.decode()Z
0.1 ms 36.31 %	▼ com/taobao/tair/comm/DefaultTranscoder.decode([BIII)Ljava/lang/Object;
0.1 ms 36.31 %	▼ java/io/ObjectInputStream.readObject()Ljava/lang/Object;
0.1 ms 36.31 %	▼ java/io/ObjectInputStream.readObject0(Z)Ljava/lang/Object;
0.1 ms 36.31 %	▼ java/io/ObjectInputStream.readSerialData(Ljava/lang/Object;Ljava/io/ObjectStreamClass;)V
0.1 ms 36.31 %	▼ sun/reflect/GeneratedMethodAccessor130.invoke(Ljava/lang/Object;[Ljava/lang/Object;)Ljava/lang/O
0.1 ms 36.31 %	▼ java/util/ArrayList.readObject(Ljava/io/ObjectInputStream;)V
0.1 ms 36.31 %	▼ java/io/ObjectInputStream.readObject(Ljava/lang/Object;
0.1 ms 36.31 %	▼ java/io/ObjectInputStream.readObject0(Z)Ljava/lang/Object;
0.1 ms 36.31 %	▼ java/io/ObjectInputStream.readClassDesc(Z)Ljava/io/ObjectStreamClass;
0.1 ms 36.31 %	▼ java/io/ObjectInputStream.readNonProxyDesc(Z)Ljava/io/ObjectStreamClass;
0.1 ms 36.31 %	▼ com/taobao/tair/comm/TairObjectInputStream.resolveClass(Ljava/io/ObjectStreamClas
0.1 ms 36.31 %	▼ java/lang/Class.forName0(Ljava/lang/String;ZLjava/lang/ClassLoader;Ljava/lang/Clas
0.1 ms 36.31 %	▶ java java lang Class forName0

代码优化实践：增加cache

- 性能提升5%

```
private Class
```

```
public TairObj  
    throws  
    super(in);  
    this.class  
}
```

```
@Override  
protected Clas
```

```
    throws  
    if(classLo  
    return
```

```
try {  
    String  
    return  
} catch (C  
    return
```

```
public class TairObjectInputStream extends ObjectInputStream {  
  
    private final static ConcurrentHashMap<ClassLoader, ConcurrentHashMap<String, Class>> Loader2Cache = new ConcurrentHashMap<>();  
  
    private ClassLoader classloader = null;  
  
    public TairObjectInputStream(InputStream in, ClassLoader classLoader)  
        throws IOException {  
        super(in);  
        this.classloader = classLoader;  
    }  
  
    @Override  
    protected Class<?> resolveClass(ObjectStreamClass desc)  
        throws IOException, ClassNotFoundException {  
        if (classLoader == null) {  
            classLoader = TairObjectInputStream.class.getClassLoader();  
        }  
  
        try {  
            String name = desc.getName();  
            ConcurrentHashMap<String, Class> cache = Loader2Cache.get(classLoader);  
            if (cache == null) {  
                cache = new ConcurrentHashMap<String, Class>();  
                ConcurrentHashMap<String, Class> old = Loader2Cache.putIfAbsent(classLoader, cache);  
                if (old != null) {  
                    cache = old;  
                }  
            }  
            Class clazz = cache.get(name);  
            if (clazz == null) {  
                clazz = Class.forName(name, false, classLoader);  
                cache.putIfAbsent(name, clazz);  
            }  
            return clazz;  
        } catch (ClassNotFoundException e) {  
            return super.resolveClass(desc);  
        }  
    }  
}
```

more..

- 对象作为HashMap的key
- web.xml配置版本信息可以减少启动时annotation的扫描时间
- **Logger创建没有使用static修饰符导致线程阻塞**
- **少用Thread.getStackTrace()**
- **正则运算尽量cache**

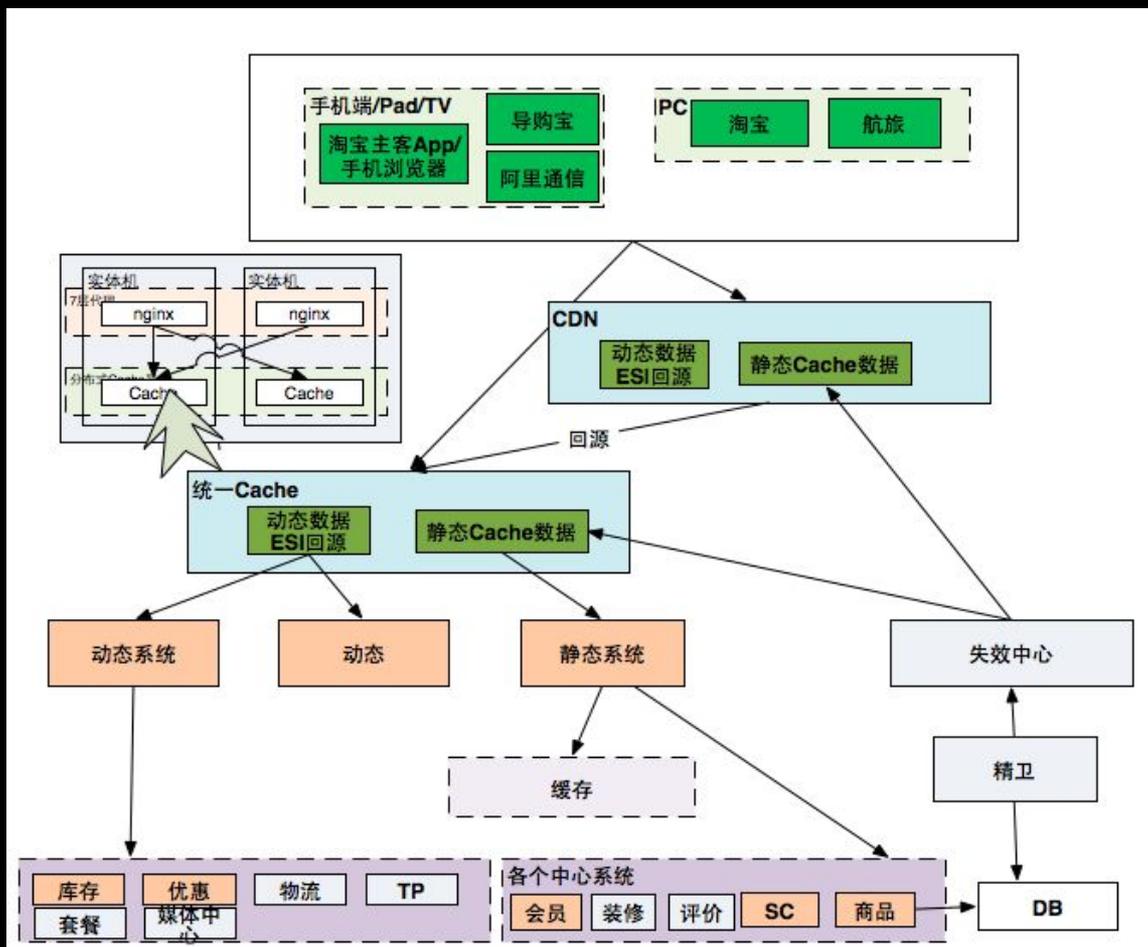
架构优化

- 数据的动静分离
- 读写的分层校验

数据动静分离

- 系统的静态化是读系统性能优化的终极必杀器
- 让用户的请求尽量不要经过Java系统
- 让静态数据放在离用户最近的地方
- 让动态数据尽可能的小

架构优化实践：读系统的静态化



架构优化实践:商品详情的静态化

- 每天支持30+亿的PV
- 可以支持峰值100w的QPS
- 静态请求单机10000+（物理机）
- 动态请求单机1500+（16核）

读写数据的分层校验

看一下全球最大的秒杀系统如何实现

读写数据的分层校验

- 从一个普通的详情页面跳转过来

天猫 Tmall.com

搜索

首页 宝贝详情

自动化数据——商城宝贝+大型秒杀宝贝+即将开始 暂无评论

天猫 Tmall.com

auctestb305店铺

宝贝与描述相符: ★★★★★
卖家的服务态度: ★★★★★
卖家发货的速度: ★★★★★

掌柜: auctestb305
客服:
公司名称: EAUMFXCF公司
所在地: 河北, 保定

店铺说明: 服务条款

[进入店铺](#)

[收藏本店铺](#)

一口价: **345.00** 元

运 费: 卖家承担运费

宝贝类型: 全新

秒杀数量: 件 (限1件)

2210年10月20日 **10:00** 开始秒杀 [即将开始...](#)

秒杀订单将直接使用默认收货地址, 请提前设置正确。 [秒杀流程](#)

[分享给好友](#) [收藏该宝贝](#)

读写数据的分层校验

- 整个页面是cache在用户浏览器
- 如果强制刷新整个页面，也会请求到CDN
- 实际有效请求只是“刷新抢宝”按钮

自动化数据——商城宝贝+大型秒杀宝贝+即将开始 暂无评论

天猫 Tmall.com

auctestb305店铺

宝贝与描述相符：★★★★★
卖家的服务态度：★★★★★
卖家发货的速度：★★★★★

掌柜：auctestb305
客服：
公司名称：EAUMFYCF公司
所在地：河北 保定

店铺说明：服务条款

[进入店铺](#)

[收藏本店铺](#)



一口价：345.00 元
运费：卖家承担运费
宝贝类型：全新

秒杀数量： 件 (限1件)

2012年11月23日 14:35 开始秒杀 [刷新抢宝](#)

1. 秒杀订单将使用默认收货地址，请提前设置正确。 [秒杀流程](#)
2. 点击“刷新抢宝”按钮刷新，会比用 F5 键和浏览器刷新更快速。

[分享给好友](#) [收藏该宝贝](#)

读写数据的分层校验

- 防止被秒杀器刷掉
- 通过答题分散用户的写请求，控制并发数



The screenshot shows a product listing for a dress. The main image shows a woman in a white dress lying on a white chair. To the right, the price is listed as 345.00元. Below the price, there is a section for the captcha challenge. The question is: "让我来考考你: 请输入[基金丰和]的第3个字的全拼(小写字母)". The answer field is empty. Below the answer field is a button labeled "立即秒杀".

一口价: 345.00 元

运 费: 卖家承担运费

宝贝类型: 全新

秒杀数量: 件 (限1件)

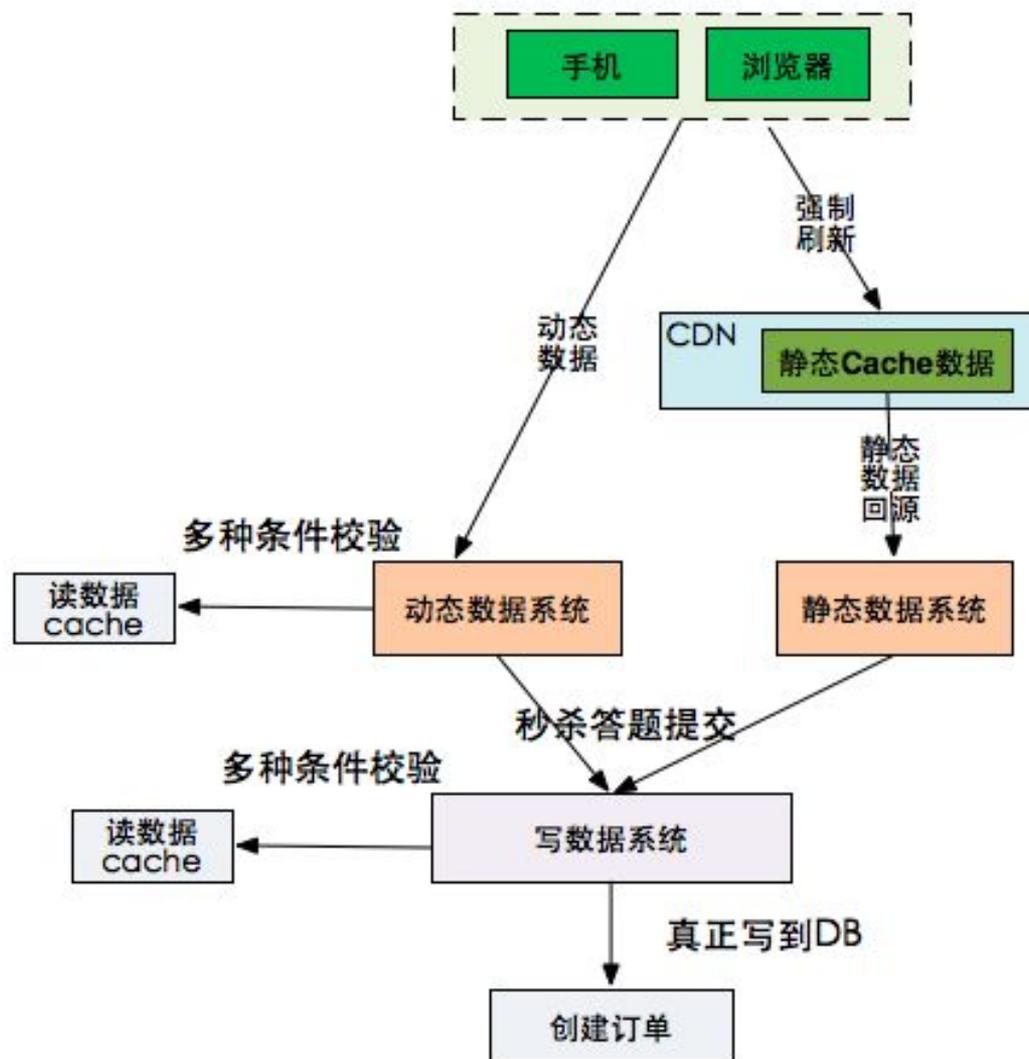
问题: 让我来考考你: 请输入[基金丰和]的第3个字的全拼(小写字母)

答案:

立即秒杀

分享给好友 收藏该宝贝

秒杀系统的执行逻辑



读写数据的分层校验总计

- 先做数据的动静分离
- 将99%的数据缓存在客户端浏览器
- 将动态请求的读数据cache在web端
- 对读数据不做强一致性校验
- 对写数据进行基于时间的合理分片
- 对写请求做限流保护
- 对写数据进行强一致性校验

链路优化

- 链路优化的目标是整体提升用户访问体验（低延时）
- 从用户的浏览器/APP
- 网关/CDN
- 服务端(web系统/服务层/数据层)

用户访问链路

- <http://www.webpagetest.org/video/compar>

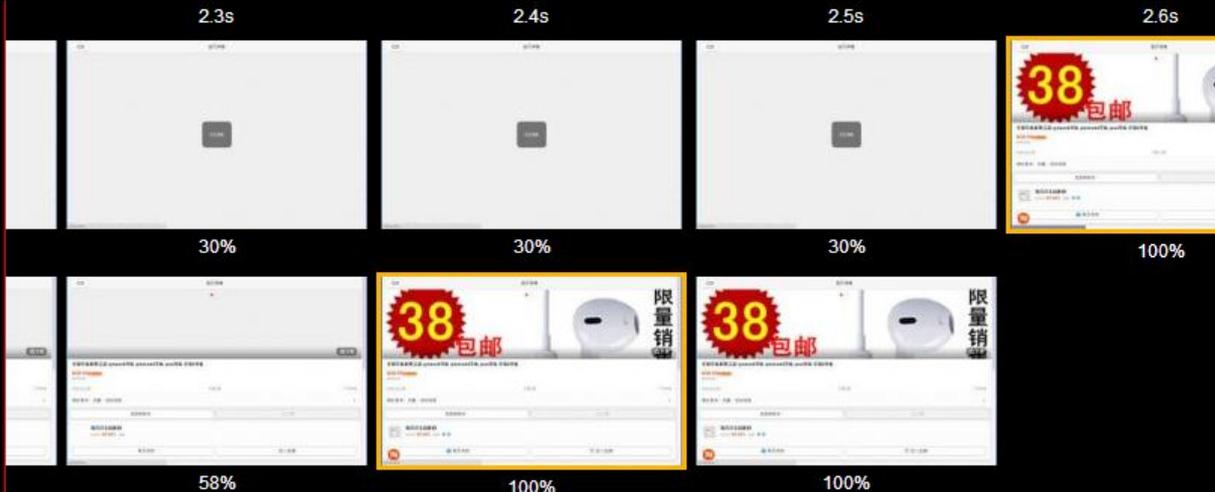
8 Z2
||

平均数据											
任务名称	整体性能 (s)	首屏用时 (s)	基础文档DNS用时 (s)	基础文档TCP用时 (s)	基础文档响应用时 (s)	基础文档下载用时 (s)	基础文档SSL握手用时 (s)	有效监测次数 (次)	错误次数 (次)	可用性 (%)	
[IL]pc 加速 100k	0.456	0	0.203	0.057	0.055	0.14	0	1494	7	99.534	
[IL]pc 回源 100k	0.409	0	0.152	0.047	0.056	0.155	0	1496	2	99.866	

平均数据										双击最大化
任务名称	整体性能 (s)	整体速度 (KB/s)	DNS用时 (s)	TCP用时 (s)	响应用时 (s)	下载用时 (s)	有效监测次数 (次)	错误次数 (次)	可用性 (%)	
[M]淘宝无线 API 统一	0.62	5.025	0.209	0.139	0.25	0.021	4434	116	97.451	
[M]淘宝无线 API 加速	0.773	4.255	0.286	0.148	0.316	0.023	5312	146	97.325	

1: h5.m.taobao.com/awp/core/detail.htm?id=36240908311 (Edit)

2: beta1.item.taobao.com/node?id=36240908311 (Edit)



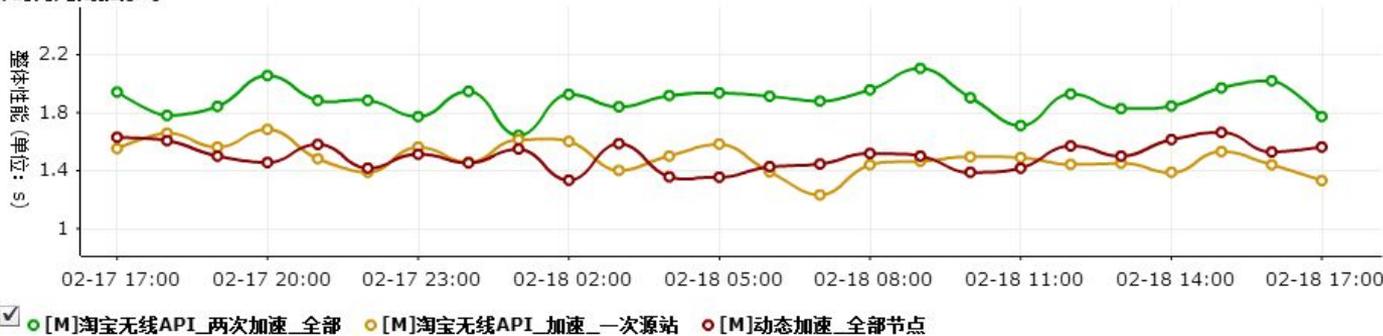
用户访问链路

- http://www.webpagetest.org/video/compare.php?tests=140318_M5_7GV%2C140318_Z2_7CJ&thumbSize=200&ival=100&end=full
- 服务端响应时间只占整个请求路径上的很小一部分
- PC上更重要的是优化首屏的加载
- 无线端更多的是优化中间的管道

无线端请求合并

- 无线环境下做请求合并的收益是比较大的，所以会将当前的两次请求再服务端做ESI合并为一个请求。

任务/时间/折线图[M]



平均数据									
任务名称	整体性能 (s)	投影DNS用时 (s)	投影TCP用时 (s)	投影响应用时 (s)	投影下载用时 (s)	有效监测次数 (次)	错误次数 (次)	可用性 (%)	元素个数 (个)
[M]淘宝无线API_两次加速_全部	1.89	0.382	0.302	0.63	0.077	4521	15	99.669	2
[M]淘宝无线API_加速_一次源站	1.486	0.459	0.244	0.349	0.036	5670	18	99.684	1
[M]动态加速_全部节点	1.492	0.476	0.215	0.333	0.057	5669	23	99.596	1

数据量大小

- 无线环境下数据大小对性能影响比PC更加明显，PC从20k到80k增加了100ms，而无线从20k到80k增加了700ms。所以无线控制页面大小对性能影响很大

平均数据										
任务名称	整体速度 (KB/s)	整体性能 (s)	DNS用时 (s)	TCP用时 (s)	响应用时 (s)	下载用时 (s)	字节数 (KB)	有效监测次数 (次)	错误次数 (次)	可用性 (%)
[M]淘宝无线 API 统一_20k	32.386	0.831	0.07	0.142	0.26	0.359	17.293	4343	45	98.974
[M]淘宝无线 API 加速_20k	27.46	1.148	0.399	0.128	0.266	0.356	17.127	5198	46	99.123
平均数据										
任务名称	整体速度 (KB/s)	整体性能 (s)	DNS用时 (s)	TCP用时 (s)	响应用时 (s)	下载用时 (s)	字节数 (KB)	有效监测次数 (次)	错误次数 (次)	可用性 (%)
[M]淘宝无线 API 统一_80k	65.903	1.873	0.238	0.144	0.29	1.201	80.783	4964	60	98.806
[M]淘宝无线 API 加速_80k	77.888	1.563	0.139	0.123	0.25	1.052	82.688	4298	41	99.055

平均数据										
任务名称	整体性能 (s)	首屏用时 (s)	基础文档DNS用时 (s)	基础文档TCP用时 (s)	基础文档响应用时 (s)	基础文档下载用时 (s)	基础文档SSL握手用时 (s)	有效监测次数 (次)	错误次数 (次)	可用性 (%)
[IL]pc 加速_20k	0.33	0	0.196	0.031	0.06	0.042	0	1471	8	99.459
[IL]pc 回源_20k	0.303	0	0.164	0.049	0.05	0.041	0	1489	1	99.933

平均数据										
任务名称	整体性能 (s)	首屏用时 (s)	基础文档DNS用时 (s)	基础文档TCP用时 (s)	基础文档响应用时 (s)	基础文档下载用时 (s)	基础文档SSL握手用时 (s)	有效监测次数 (次)	错误次数 (次)	可用性 (%)
[IL]pc 加速_100k	0.456	0	0.203	0.057	0.055	0.14	0	1494	7	99.534
[IL]pc 回源_100k	0.409	0	0.152	0.047	0.056	0.155	0	1496	2	99.866

cache到CDN上

- 直接从CDN上获取Cache后的数据性能很好，40k以下的页面只要600ms左右。相当是直接回源一倍的性能

平均数据										
任务名称	整体性能 (s)	整体速度 (KB/s)	DNS用时 (s)	TCP用时 (s)	响应用时 (s)	下载用时 (s)	有效监测次数 (次)	错误次数 (次)	可用性 (%)	字节数 (KB)
[M]无线_cache_广东_2k (结束)	0.32	7.096	0.001	0.097	0.216	0.006	110	0	100.000	1.561
[M]无线_cache_广东_20k (结束)	0.534	45.17	0.001	0.099	0.213	0.221	104	2	98.113	15.685
[M]无线_cache_广东_40k (结束)	0.67	65.403	0.001	0.09	0.197	0.381	112	1	99.115	30.167
[M]无线_cache_广东_80k (结束)	1.26	99.633	0.001	0.095	0.21	0.954	151	5	96.795	73.463

一些结论

- 无线环境下一次网络请求要明显好于2次以上，减少网络请求次数对首屏加载性能影响比较明显
- 无线环境文件大小与PC环境下文件大小对性能影响的差异不同，无线环境下数据大小对性能影响比PC更加明显，PC从20k到80k增加了100ms，而无线从20k到80k增加了700ms。所以无线控制页面大小对性能影响很大
- CDN直接Cache性能提升很大，所以尽量数据Cache到CDN同样对无线是有效的
- 小数据情况下动态加速和直接回主站比较没有明显优势，加上当前动态加速链路还在调优，所以当前无线数据直接回统一cache比较理想。待动态加速更加成熟后再走CDN，当前统一Cache和CDN已经做到了动态切换，所以往CDN也没有成本

more...

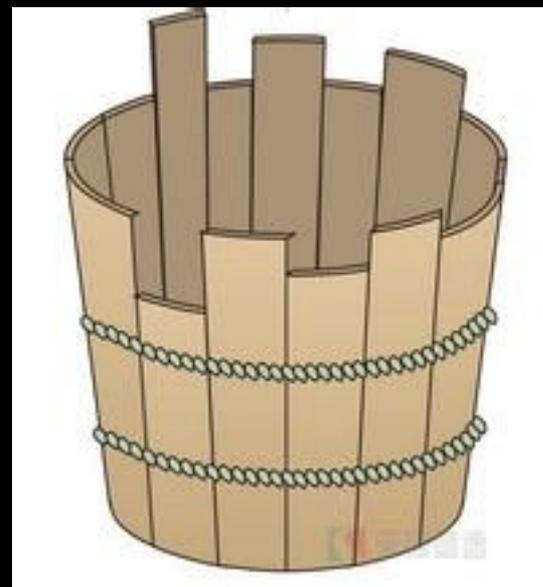
- 域名的收敛
- DNS本地cache
- SPDY长连
- 图片本地cache
- 合理的预加载机制

我们的经验

- 知道短板
- 减少数据大小
- 数据分级
- 减少中间环节、增加预处理

发现短板

- 光速
- 网速
- 网络结构（交换机/网卡）
- TCP/IP
- 虚拟机（内存/cpu/io...）
- 应用



减少数据大小

- HTML
- 图片
- JSON
- Java对象
- 请求数



数据分级



数据分级

- 首屏为先
- 重要信息为先
- 次要信息异步加载

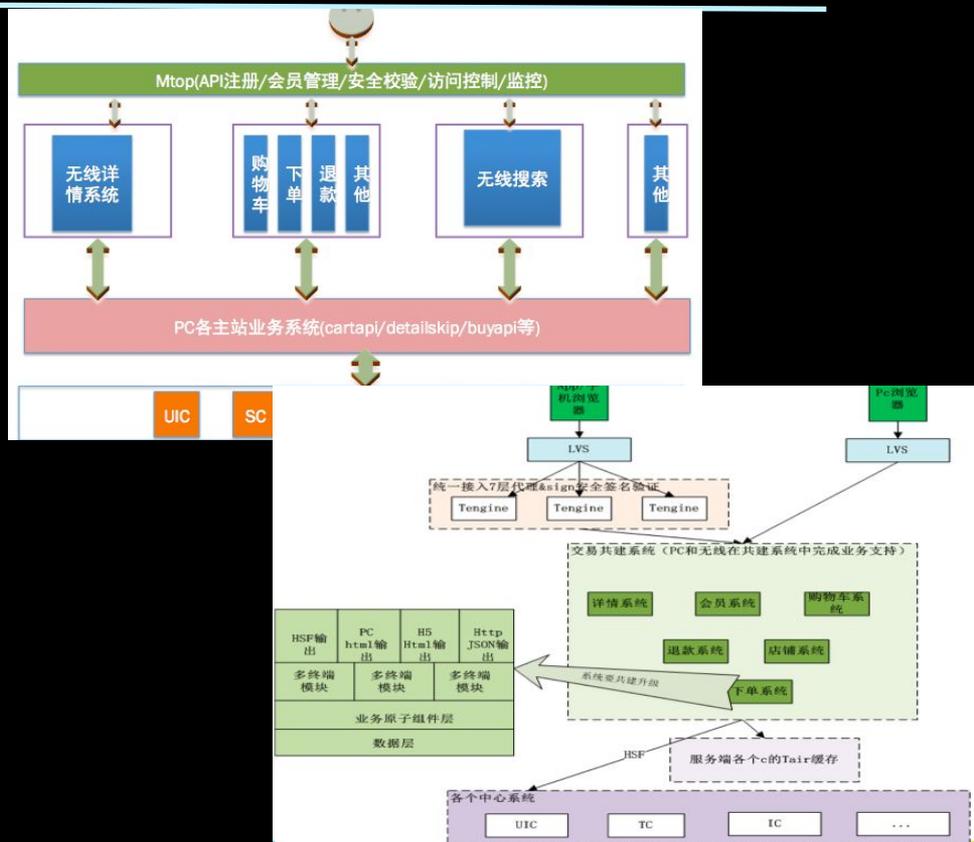


减少中间环节



减少中间环节

- 减少中间代理
- 减少字符到字节的转换
- 将变的转换为不变的，增加预处理



回顾

1. 面临的挑战

- 流量挑战
- 业务复杂度挑战

2. 我们走过的路

- 代码优化
- 架构优化
- 链路优化

3. 我们的经验

- 知道短板
- 减少数据大小
- 数据分级
- 减少翻译、增加预处理

总结

1. 一定要做应用基线
 - 性能基线（何时性能突然下降了？）
 - 成本基线（去年双11用了多少台机器？）
 - 链路基线（我们的系统发生了那些变化？）
2. 必须持续有人关注系统的性能
 - 代码级（提升编码质量）
 - 业务（改掉不合理的调用）
 - 架构和链路级（改进架构）
3. 更通用和批量的解决问题
 - 整合系统之间的调研链路（合并部署）
 - 提升整体机器使用率（弹性部署）



- 微博:@淘宝君山
- webchat:xulingbo0201
- 邮箱:xulingbo0201@163.com
- <http://xulingbo.net>

