淘宝网
Taobao.com

淘宝高访问量系统优化那些事

君山
共享业务平台事业部-商品详情
2013.8.21

个人简介

- 许令波，花名君山，
- 微博：@淘宝君山，
- 邮箱：junshan@taobao.com
- http://xulingbo.net
- 2009年毕业加入淘宝，关注性能优化领域，参与了淘宝高访问量Web系统主要的优化项目，如模板引擎的改造、静态化、CDN化等。著有《深入分析Java Web技术内幕》一书。

内容简介

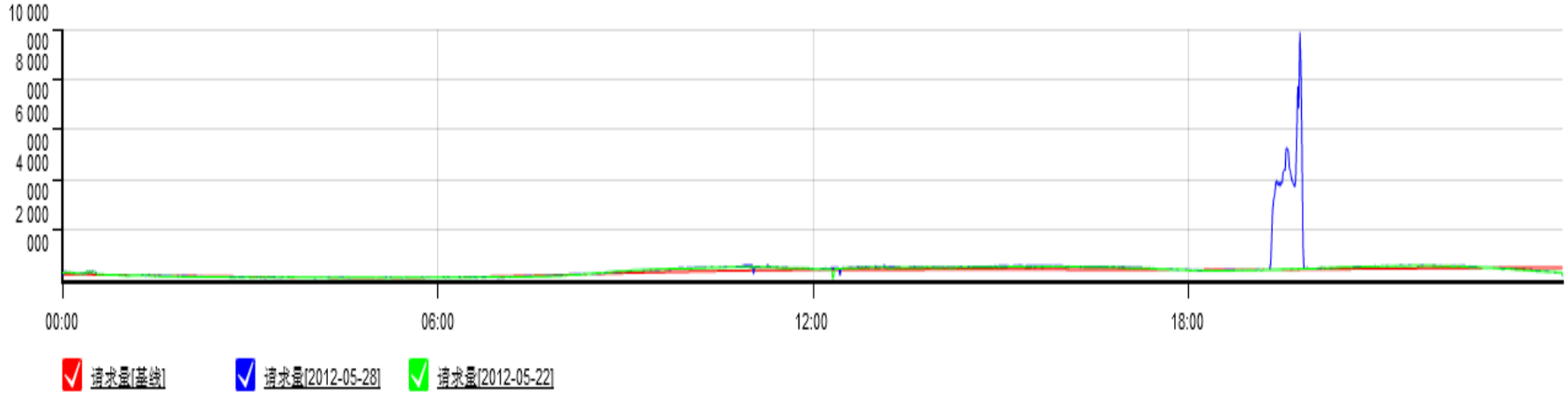
- ***我们面临的问题与挑战***
- 淘宝高访问量系统优化历程
- 优化事例介绍
- 优化实践总结
- 总结

面临的问题与挑战（1）

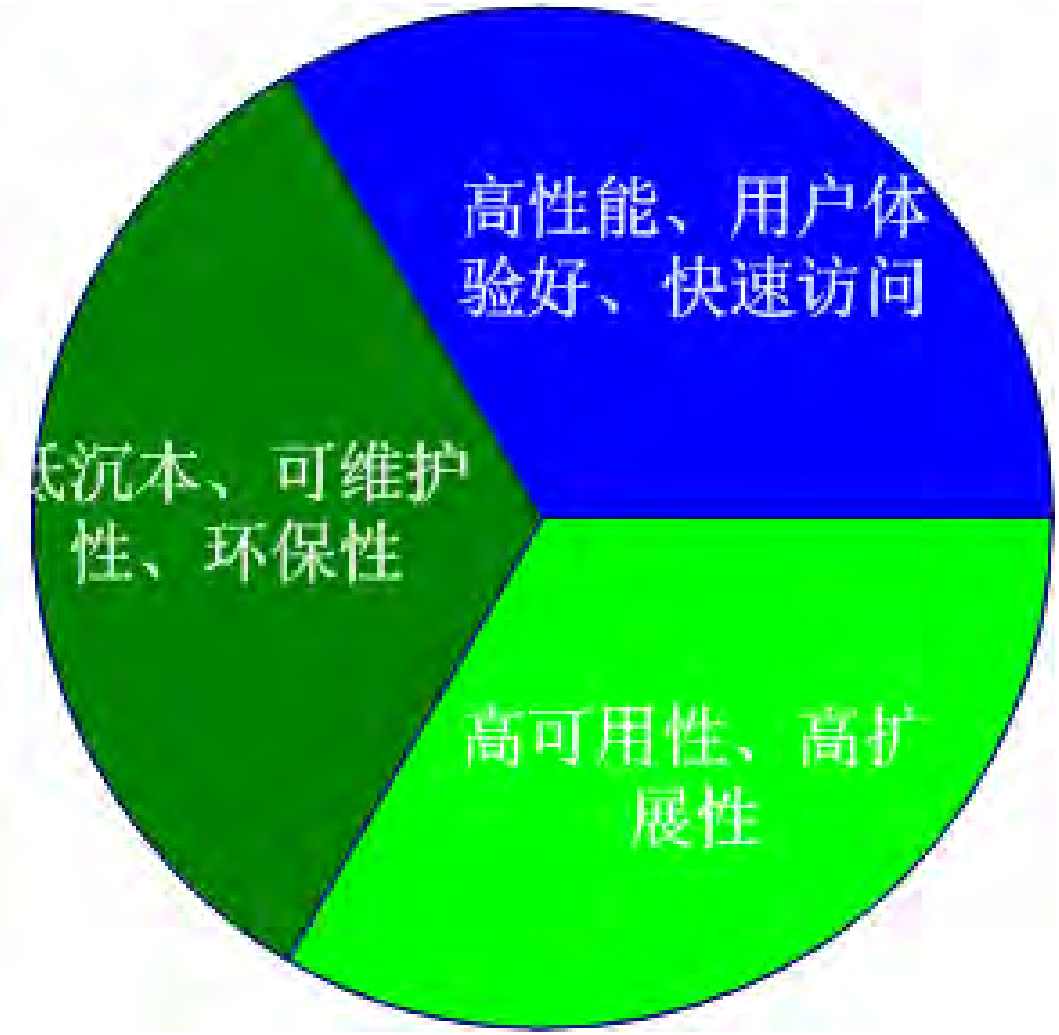
- Alexa全球排名为13
- 根据Alexa统计日均PV约有25亿，日均独立IP访问约有1.5亿
- 访问量仍在增长，硬件成本持续增加
- 机房、带宽、电力有点伤不起了

面临的问题与挑战（2）

- 双11/双12的大型促销活动
- 秒杀/活动等突发流量冲击
- 各种爬虫频繁抓取数
- 网站经常受到攻击



面临的问题与挑战（3）



内容简介

- 我们面临的问题与挑战
- ***淘宝高访问量系统优化历程***
- 优化事例介绍
- 优化实践总结
- 总结

淘宝高访问量系统优化历程

- 2009系统拆分、静态文件合并、前端页面异步化和JSON化
- 2010去DB依赖、引入缓存、提升单机QPS、关注用户体验
- 2011 优化进入深区Velocity、BigRender
- 2012 静态化改造
- 2013 统一cache、CDN化、网络协议

（1）系统拆分

- 存在的问题
  - 打包部署困难
  - 多人开发代码容易冲突
  - 权限控制不便
- 解决
  - 五彩石项目
  - 中心化商品、用户、店铺、交易

（2）静态文件合并

- 遇到的问题
  - 前端性能不佳
  - 页面中存在众多脚本和样式
  - 阻塞式加载
- 解决
  - 整合页面中inline的js\css到外部文件
  - 静态文件合并请求
  - 将iframe改为jsonp调用

（3）异步化、JSON化

- 弊端
  - HTML页面大
  - 服务端RT比较长
- 解决办法
  - 部分请求异步化
  - 列表页面JSON输出
  - 建立了异步系统

（4）提升单机性能

- 前端系统去除DB的直接依赖
- 后端系统中心化、平台化、服务化
- 大量使用K/V缓存，分布式缓存数据
- 建立较完善的监控系统，跟踪和预测系统性能趋势

（5）关注用户端体验

- 前端优化
- 脚本延迟加载
- BigRender优化

（6）后端深度优化

- Velocity模板引擎优化
  - 开发Sketch模板引擎
  - Vm转成Java
  - 去除空格等无效字符
  - 性能提升30%
- 协程

（7）静态化改造

- 遇到的瓶颈
  - 单纯Java系统已经难于解决
  - 单系统可能需要上千台机器、难以接受
  - 大促、攻击等
- 解决
  - 动态系统静态化改造
  - 建立统一的cache集群
  - 进一步CDN化

内容简介

- 我们面临的问题与挑战
- 淘宝高访问量系统优化历程
- ***优化事例介绍***
- 优化实践总结
- 总结

事例（1，Velocity模板引擎优化）

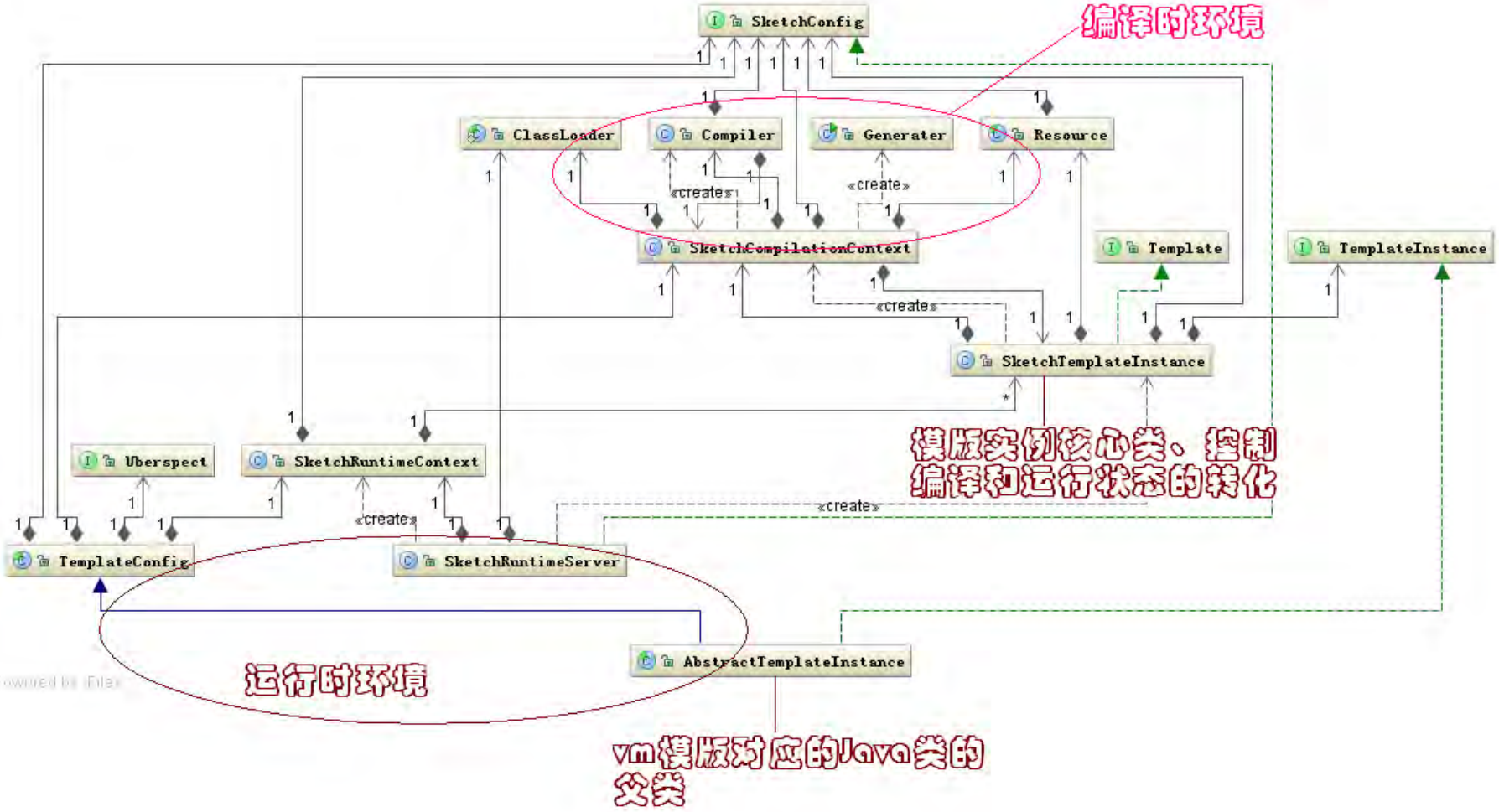
- 面临的问题
  - Velocity是动态解释性语言，执行效率较差
  - 页面复杂，反射调用非常多
  - 发现模板渲染占用了60%以上的CPU时间。
  - 整个页面输出比较大，平均在80KB左右。

事例（1，Velocity模板引擎优化）

- 解决思路
  - 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

事例（1，Velocity模板引擎优化）

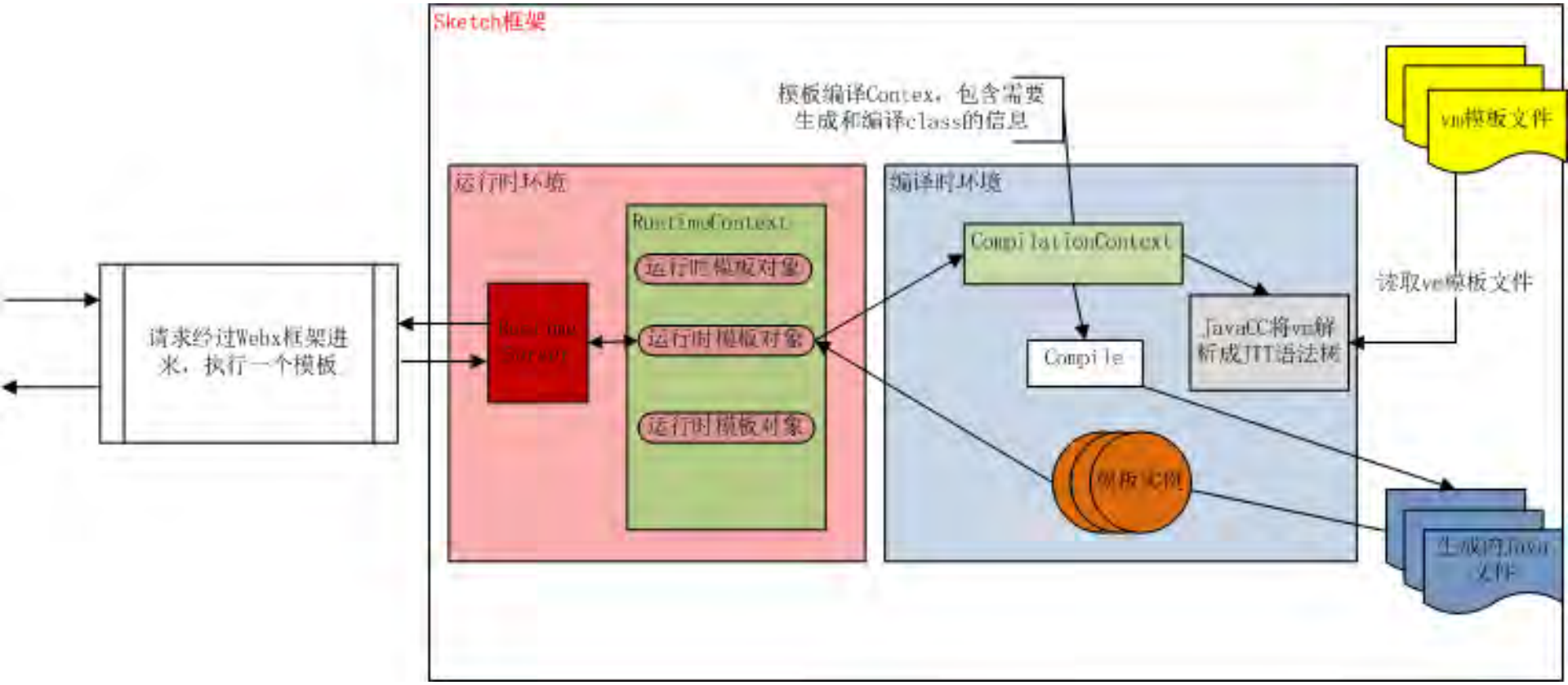
- 系统结构



```
private Object _foreach_3414368_43072917262352(final PageContext pageContext, final I _I) throws Exception {
    final ContextAdapter context = pageContext.getContext();
    final PageWriter out = pageContext.getOut();
    Iterator _it = _COLLE((_I.exampleDO == null ? null : ((Mode) _I.exampleDO).getItemList()));
    int _VelocityCount = 1;
    while (_it.hasNext()) {
      Object _i = _it.next();
      pageContext.addForVarsDef("_i", _i);
      pageContext.addForVarsDef("_VelocityCount", _VelocityCount);
      out.write(_S0);
      if (EPUT.is(EPUT.eq(EPUT.mod(_i, 2), 0))) {
        context.put("str", "偶数");
        out.write(_S0);
      } else {
        context.put("str", "奇数");
        out.write(_S0);
      }
      out.write(_S0);
      out.write(_EVTCK(context, "$i", _i));
      out.write(_S1);
      out.write(_EVTCK(context, "$str", context.get("str")));
      out.write(_S2);
      _VelocityCount++;
    }
    return Boolean.TRUE;
  }
```

事例（1，Velocity模板引擎优化）

- 运行时状态



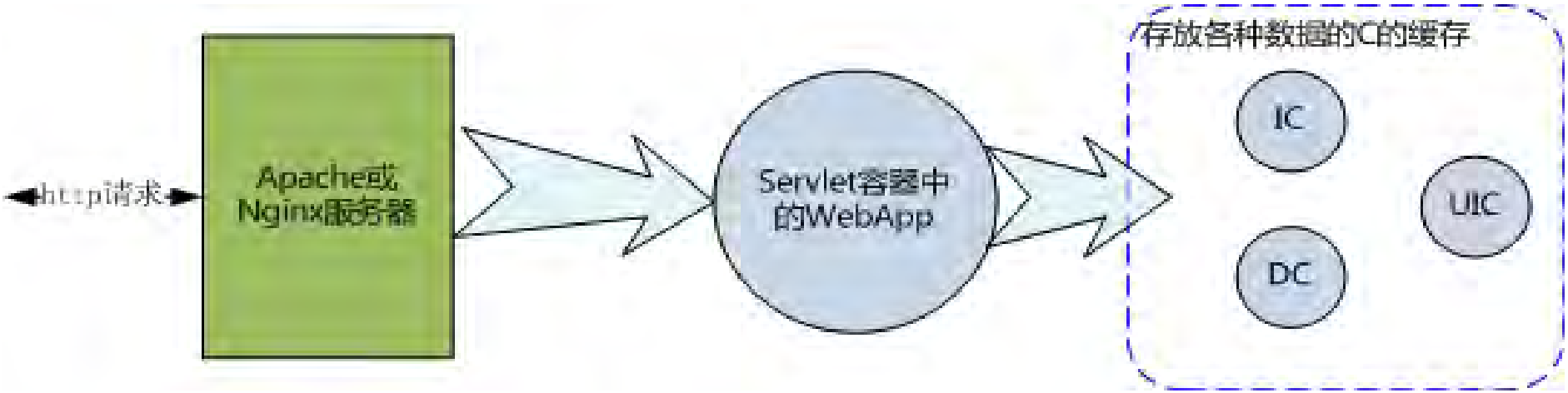
事例（1，Velocity模板引擎优化）

- 成果

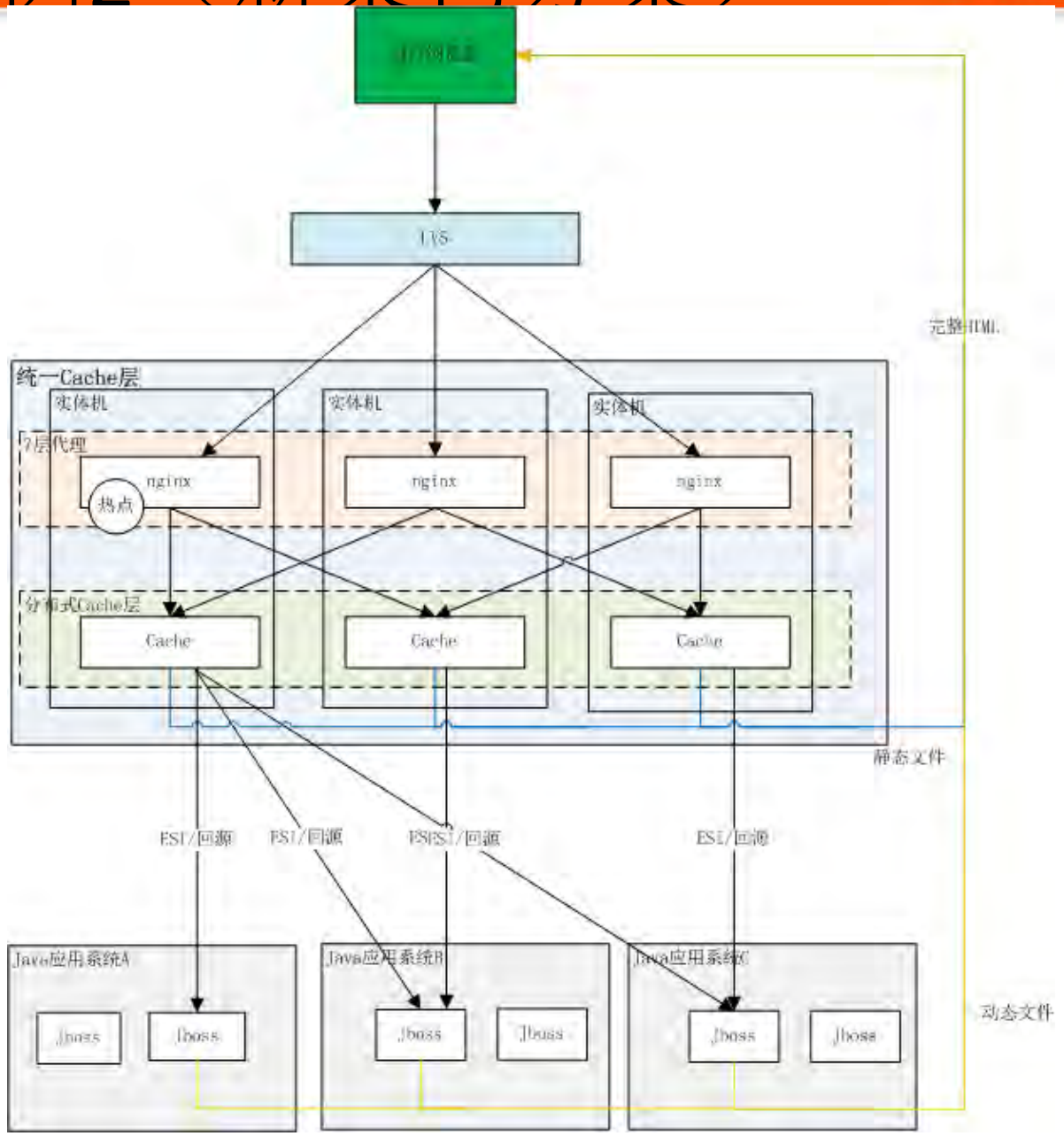
页面大小	优化前 QPS	优化前 RT(ms)	优化后的 QPS	优化后 RT(ms)	提升%
47355	319.05	109.7	455.87	76.776	43%
48581	306.85	114.061	445.39	78.582	45%
55735	296.65	117.983	437.46	80.007	47%
63484	193.69	180.698	302.55	115.684	56%
83152	180.88	193.498	236	148.305	30%
92890	170.68	205.064	214.27	163.342	26%
99732	103.64	337.707	161.46	216.77	56%
144292	108.76	321.81	148.18	236.199	36%
67144	148.49	235.714	268.07	130.565	81%
79703	124.51	281.1	243.64	143.657	96%
92537	123.85	282.595	190.8	183.44	54%
127047	117.52	297.829	164.1	213.284	40%
129479	105.36	332.197	155	225.8	47%

事例2（详情系统的静态化改造）

- 详情系统当前架构面临的挑战
  - 能使用缓存的地方已经使用了
  - 能想到的都做了
  - 当前的Detail架构已经达到瓶颈



事例2（新架构方案）

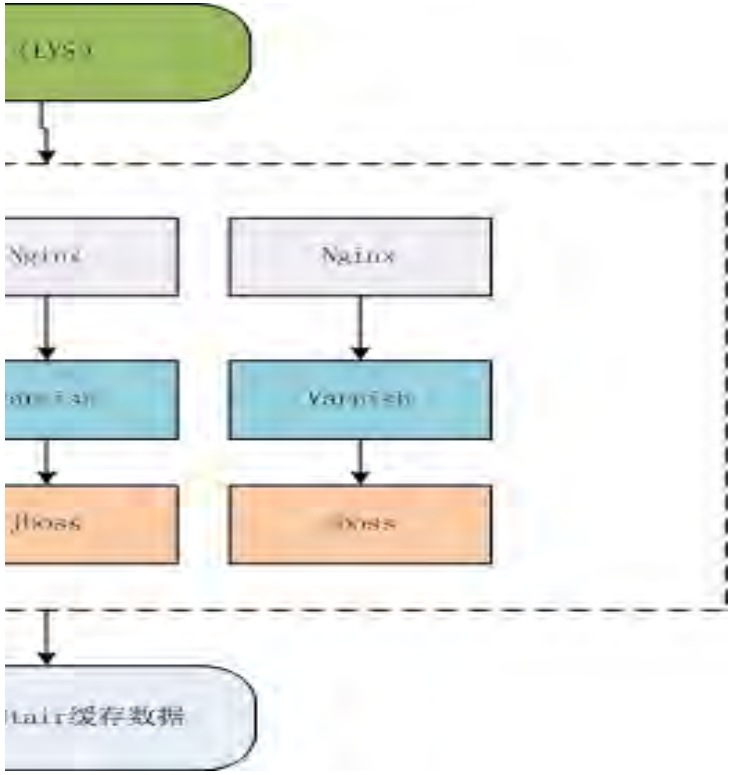

LVS
完整HTML
统一Cache层
实体机
实体机
实体机
7层代理
nginx
热点
nginx
nginx
分布式Cache层
Cache
Cache
Cache
静态文件
ESI/回源
ESI/回源
ESI/回源
ESI/回源
Java应用系统A
Java应用系统B
Java应用系统C
Jboss
Jboss
Jboss
Jboss
Jboss
Jboss
动态文件

事例2（新架构方案选择）

- 是否一致性Hash分组？
- 动静分离是ESI or CSI？
- 是否使用物理机？
- 谁来压缩?
- 网卡选择?

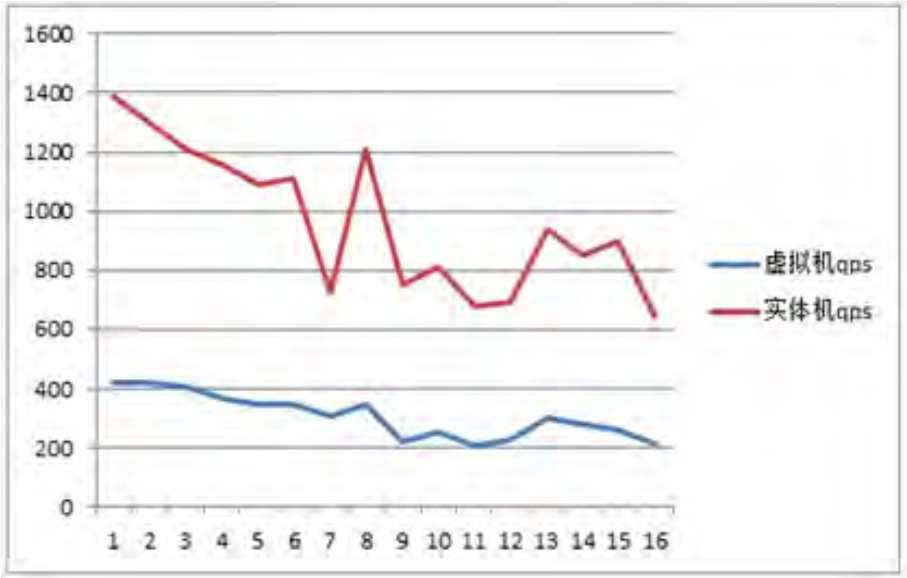
事例2（方案1 Tengine+cache+Jboss（vm））

- 优点
  - 没有网络瓶颈，不需要改造网络
  - 机器数多，也没有网卡瓶颈
  - 机器数增多故障风险减少
- 缺点
  - 机器数多，缓存命中率下降
  - 缓存分散，失效难度增加·
  - cache和Jboss都会争抢内存



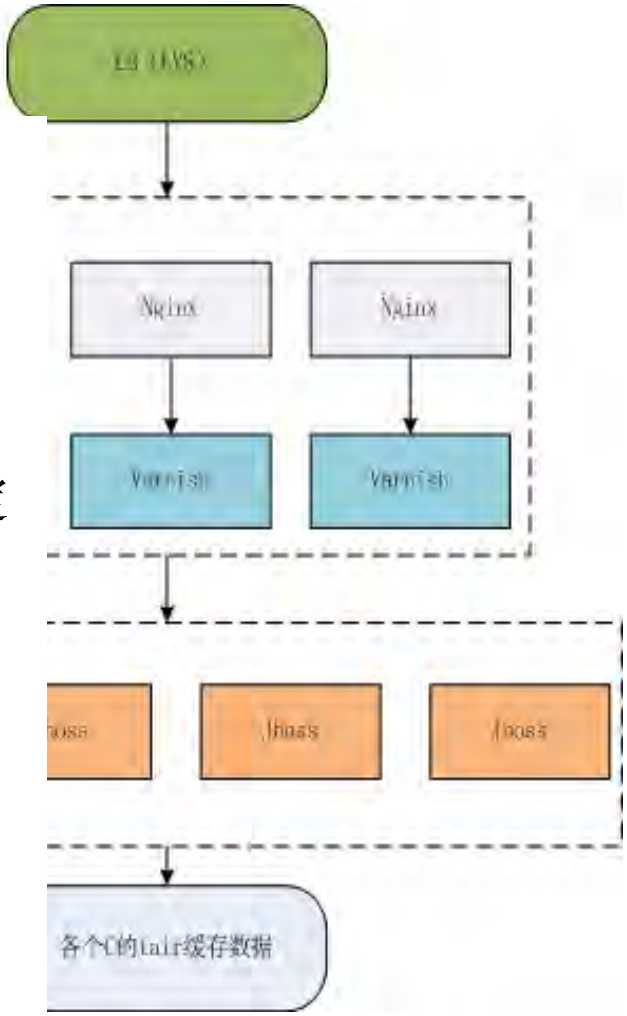
事例2（方案2 Tengine+cache+Jboss实体 ）

- 实体机和虚拟机对比
  - 实体机是虚拟机的3倍
- 中庸之美
  - 既没有网络瓶颈也能使用大内存
  - 减少Varnish机器，提升命中率
  - 提升命中率，能减少Gzip压缩
  - 减少CC失效压力



事例2（方案3 Tengine+cache前置）

- 优点
  - 与Java应用隔离，方便PE维护
  - 方便其他系统接入
- 缺点
  - 缓存集中，系统的稳定性有一定挑战



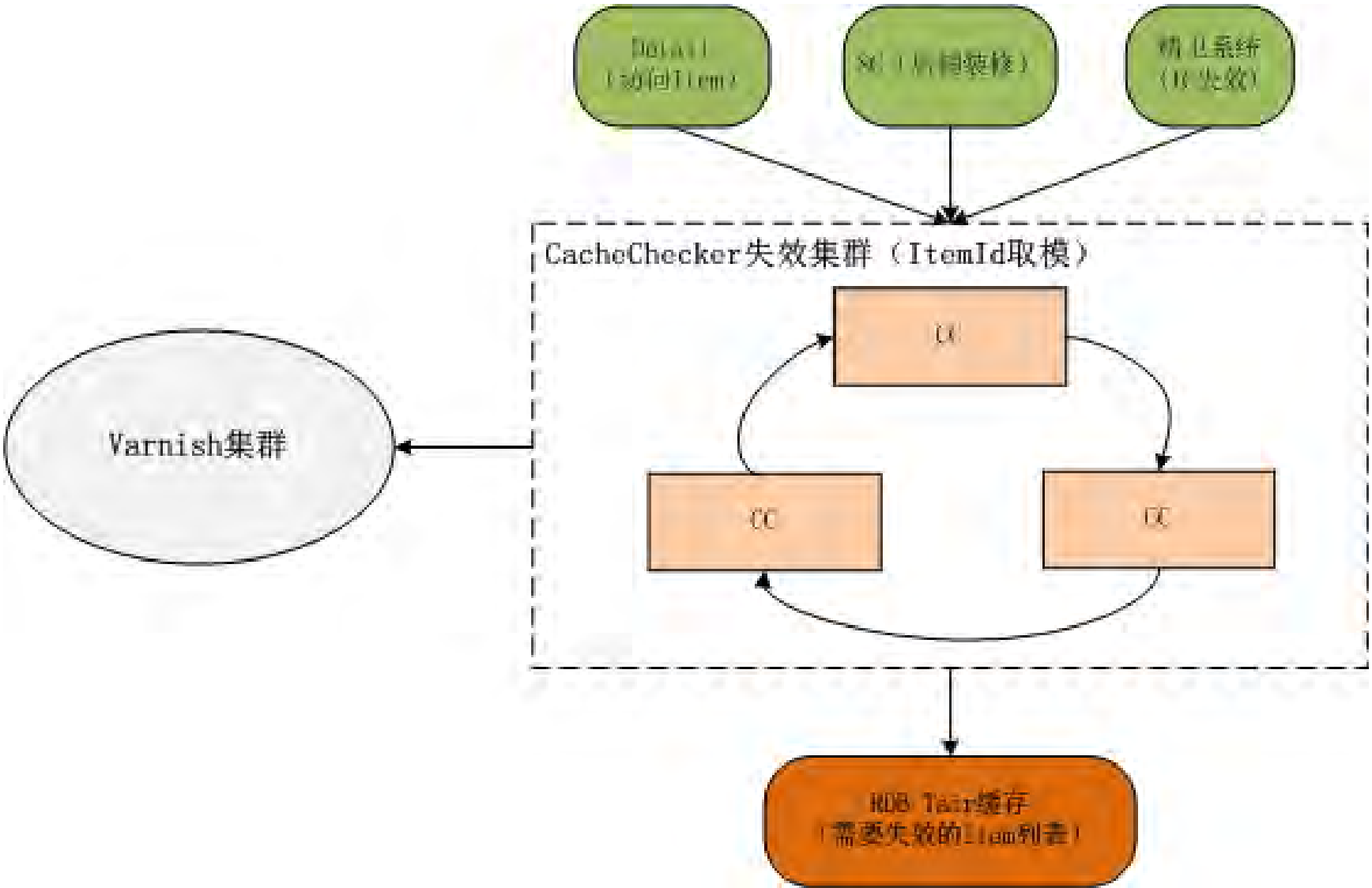
事例2（动态系统如何改造-动静分离）

- URL固定（/item.htm?id=xxxx）
- 去掉浏览者相关
  - Atpanel打点
  - 用户id等Cookie信息
- 去掉时间相关
  - 定时上架
  - 秒杀
- 去掉地域相关
  - 运费模板
- 去掉Cookie

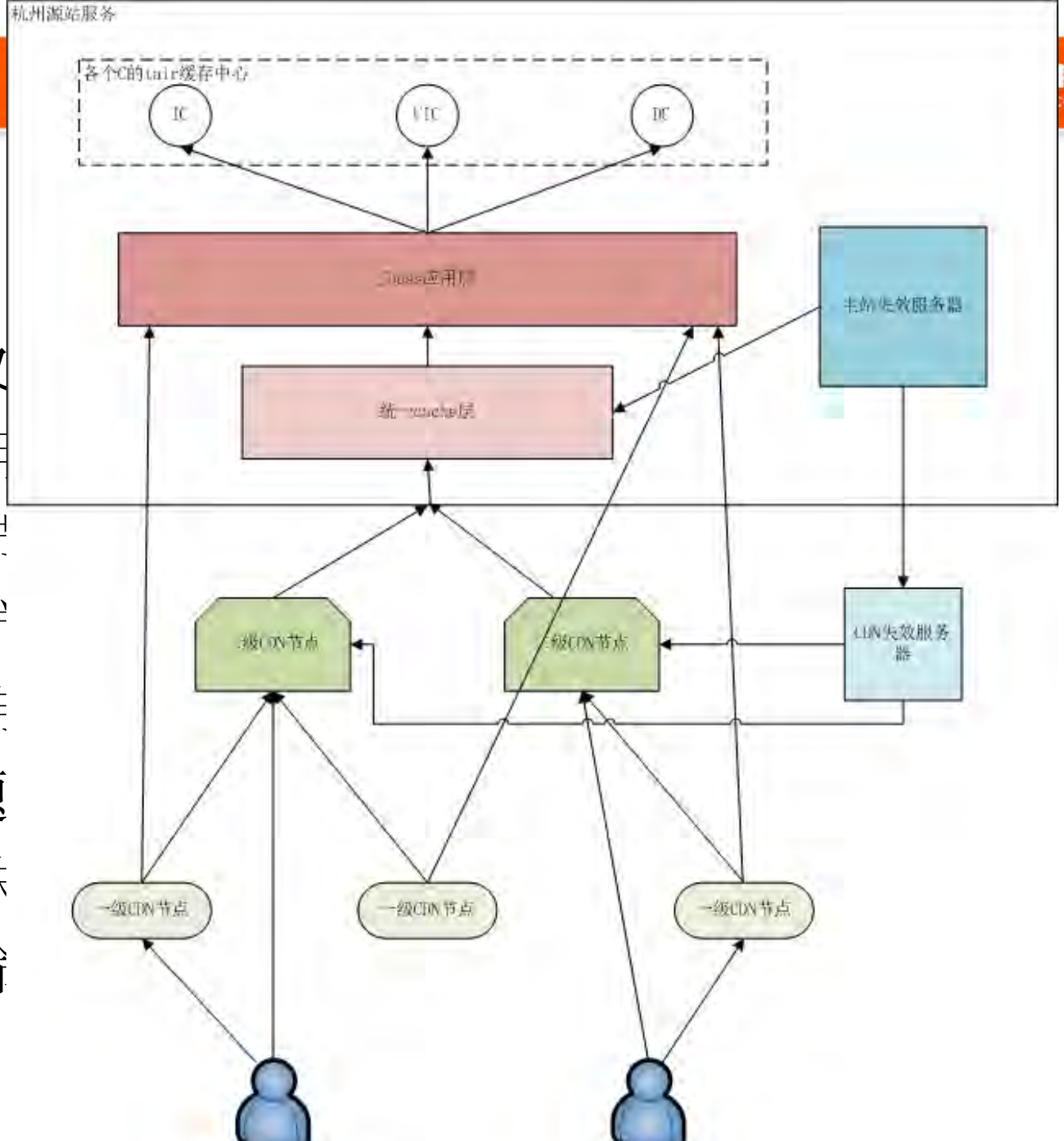
事例2（动态内容结构化）

```
window.TB.Detail.common.data.idata={
    item:{
        id:1,//item Id
        status:0
    },
    seller:{
        id:1,//sellerId
        status:0
    },
    shop:{
        id:1,//id>0 has shop
        eshop:true//是否是旺铺
    },
    //This Field need To be Injected by Ajax
    viewer:{
        id:1,
        buySum:2//买家信用
    }
}
```

事例2（失效方案）



- 好处
  - 用
  - 节
  - 扩
  - 带
- 问题
  - 失
  - 命


杭州源站服务
各个IDC的tair缓存中心
IDC
VIDC
IDC
Tomcat应用层
主站失效服务器
统一cache层
二级CDN节点
二级CDN节点
CDN失效服务器
一级CDN节点
一级CDN节点
一级CDN节点

内容简介

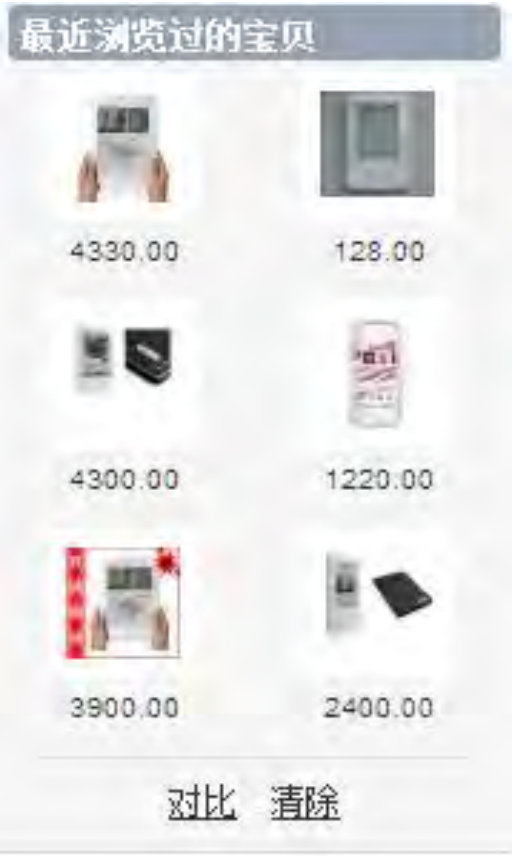
- 我们面临的问题与挑战
- 淘宝高访问量系统优化历程
- 优化事例介绍
- ***优化实践总结***
- 总结

优化实践总结

- 前端优化实践总结
- Java端优化
- 网络优化

前端优化实践总结

- 保证首屏加载速度
  - 首屏HTML尽量小（1~4k）
  - 控制首屏请求数量
- 按需加载
  - 控制页面长度
- 前端缓存
  - 304
  - 客户端存储



Java系统优化实践总结

- 字符编码
- 减少反射调用
  - asm
- 并发控制
  - 协程
  - hystrix

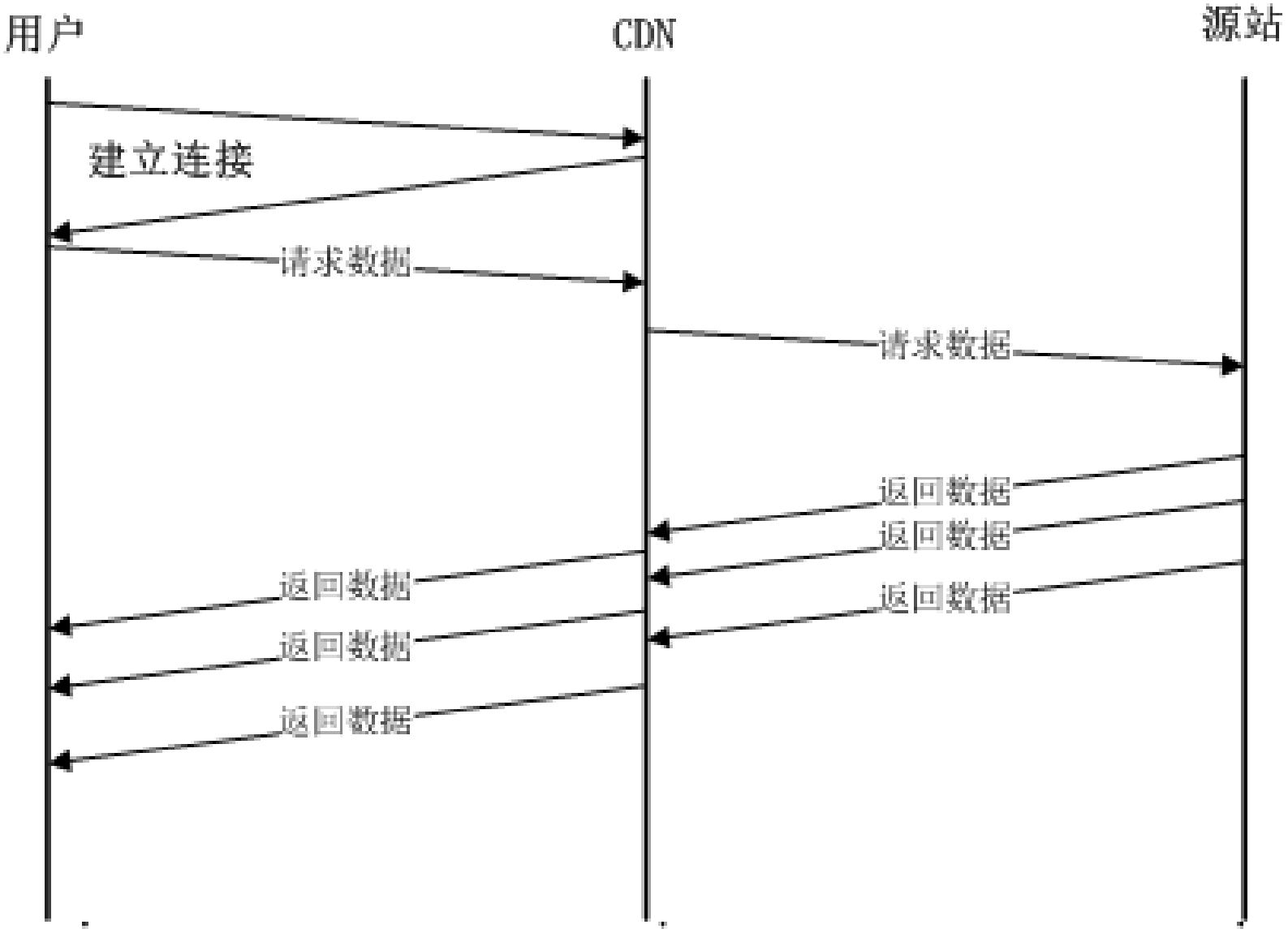
网络优化实践总结

- TCP协议栈优化
- CDN动态加速
- Webp图片优化

TCP协议栈优化

- 调整TCP窗口
- 一次HTTP请求需要多少个RTT？
- 5%的请求的平均响应时间>1s,更早的发现丢包，让重传的速度变快

CDN动态加速



Webp图片优化

WebP 格式压缩率
Quantity: 30472

quantiles	Diff(%)	Avg(%)
5.0%	-90.0	-92.4
10.0%	-88.0	-90.9
20.0%	-84.0	-88.6
30.0%	-80.0	-86.5
40.0%	-74.0	-84.0
50.0%	-69.0	-81.5
60.0%	-65.0	-79.1
70.0%	-60.0	-76.8
80.0%	-55.0	-74.4
90.0%	-45.0	-71.7
95.0%	-35.0	-70.1
99.0%	-18.0	-68.4
99.9%	3.0	-67.9
99.99%	57.0	-67.9
99.999%	80.0	-67.8
100%	80.0	-67.8

内容简介

- 我们面临的问题与挑战
- 淘宝高访问量系统优化历程
- 优化事例介绍
- 优化实践总结
- ***总结***

总结

- 优化是从做大量日常开始的
  - 最了解系统的人最适合做
- 从业务推动到优化到主动追求技术提升
  - 提前预研
  - 技术追求（横向比较）
- 优化是一个长期的持续的过程
  - 用最简单的方式达到最好的效果
  - 用麻烦的方式达到比较好的效果
  - 通过复制让更多的系统享受到更好的效果
- 优化要从多个层次集合形成合力
  - 前端
  - 服务器端
  - 网络等
- 建立监控机制

淘宝网
Taobao.com

Q & A

谢谢