淘宝网前台应用性能优化实践

自我介绍

- 丁宇，阿里花名叔同
- 4年行业软件和3年互联网应用开发经验，对构建高性能、高可用、可扩展的Web应用兴趣浓厚
- 现为淘宝网技术部综合业务平台技术专家，负责性能优化领域的工作
- Weibo：淘宝叔同

议程

1. 应用性能分析

2. 基础设施优化

3. 应用自身优化

4. 前端性能优化

I. 应用性能分析

- 介绍前台应用
- 度量关键指标
- 查找应用瓶颈

介绍前台应用

- 面向用户的Web应用
- 商品详情、店铺等
- 流量较大、机器较多

前台应用特点

- 模板渲染页面
  - 使用Velocity引擎
  - 页面较大(大于100KB)
- 外部调用较多
  - 数据来自外部，不保存数据
  - DB、HSF、Tair、Search、TFS ...
- 磁盘读写较少
- 吞吐量相对低

度量关键指标

- 吞吐量
  - 线上环境压测获取
  - HttpLoad、Nginx引流
- 响应时间、 页面大小
  - 分析访问日志获取
- 每请求内存数
  - GC回收的内存数除以吞吐量

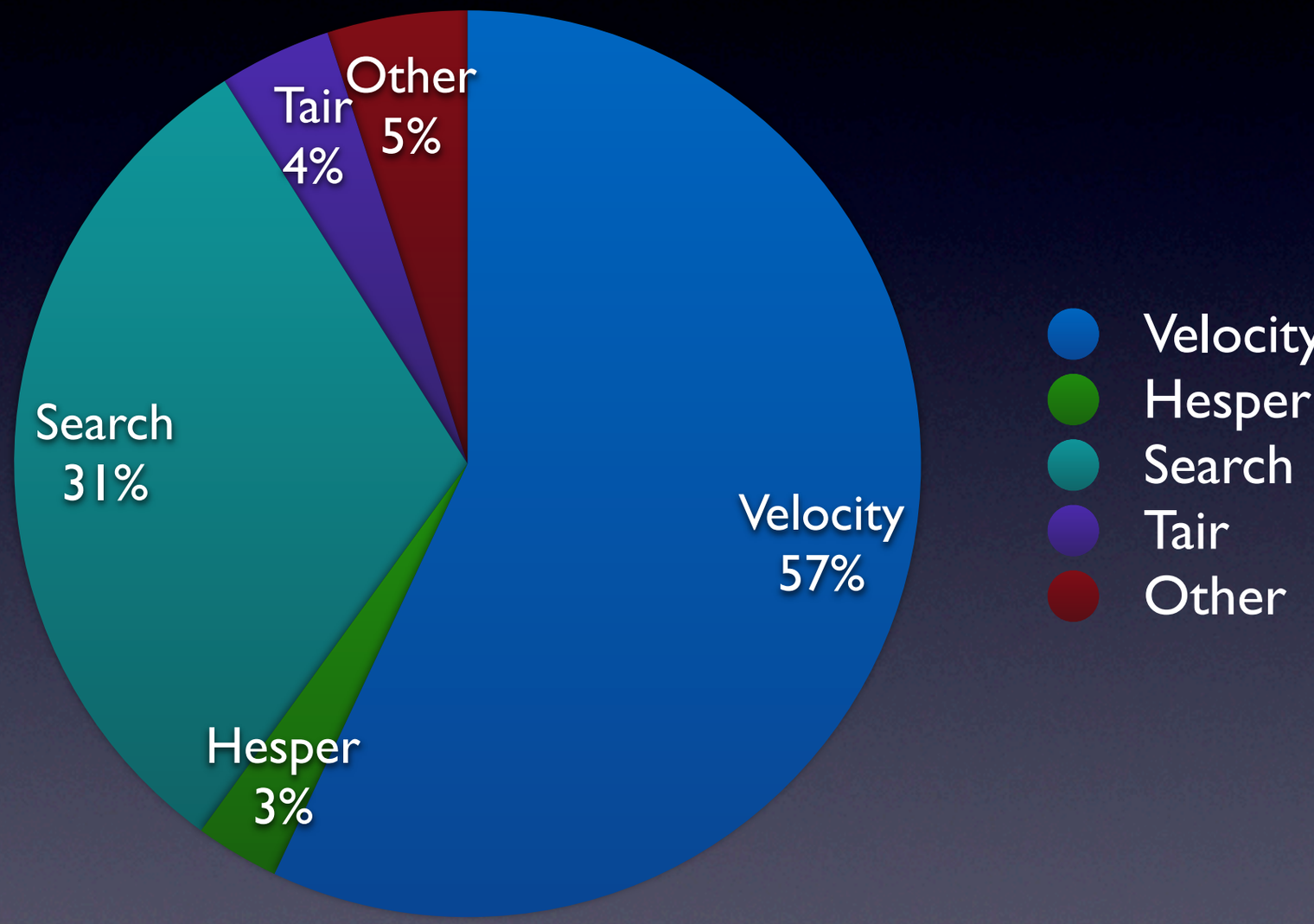
查找应用瓶颈

- 关于瓶颈
  - 瓶颈是系统中比较慢的部分，在瓶颈完成前，其他部分需要等待
- 2/8原则
  - 20%的代码执行会消耗80%的资源

查找应用瓶颈

- CPU、IO、Memory
  - Thread、File、Socket、GC
- 一般性瓶颈
  - CPU计算
    - 字符串查找、替换、拼接
    - 编码、解码、压缩、解压
  - 外部调用
    - IO开销、序列化、反序列化

查找模块瓶颈



时间消耗分析，使用TProfiler工具

查找代码瓶颈

- 找出代码瓶颈
  - 使用TProfiler工具
  - VisualVM、JProfiler

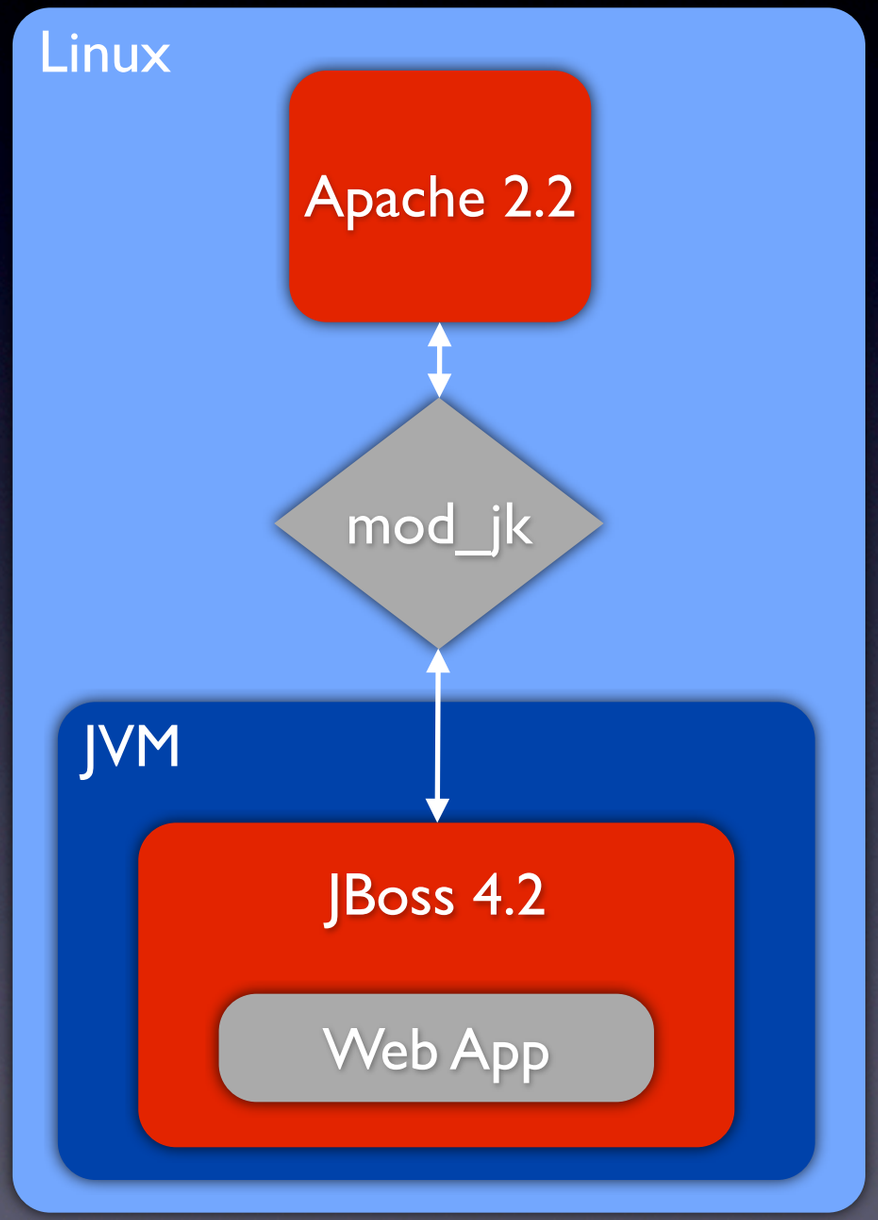
方法信息	执行时间	执行次数	总时间
com/xxx/web/core/NewList:execute()	61	3102	190067
com/xxx/web/core/PerformScreenTemplate:performScreenModule()	18	4822	87822
com/xxx/biz/core/DefaultSearchAuctionManager:doMultiSearch()	43	708	30357
com/xxx/biz/core/DefaultSearchCatRedirectManager:doSearch()	4	1248	4552

2. 基础设施优化

- 软件升级
- JVM调优
- 模板引擎优化

软件升级

- Apache -> Tenginx
  - 吞吐量提升10%+
- JBoss 4 -> Tomcat 7
  - 吞吐量提升10%-
- JVM 1.6 23 -> 1.6 32
- OS 32bit -> 64bit
  - 加大Java Heap，吞吐量提升70%
- Kernel升级
  - 内核升级参数调整，吞吐量提升40%



JVM调优

- Minor GC暂停25ms， Major GC暂停500ms
- 减少Young晋升到Tenured的对象， 减少Major GC
- 合理分配堆大小、Survivor Space Size、 Tenuring Threshold
- 借助工具 TBJMap 分析JVM堆每一个分区的内容
- 优化系统代码， 及时回收对象、减少内存使用、减小页面大小
- JVM性能表现的最佳状态是没有Major GC

模板引擎优化

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%

- Char 转 byte - 性能提升100%
- 解析执行改编译执行 - 性能提升10%

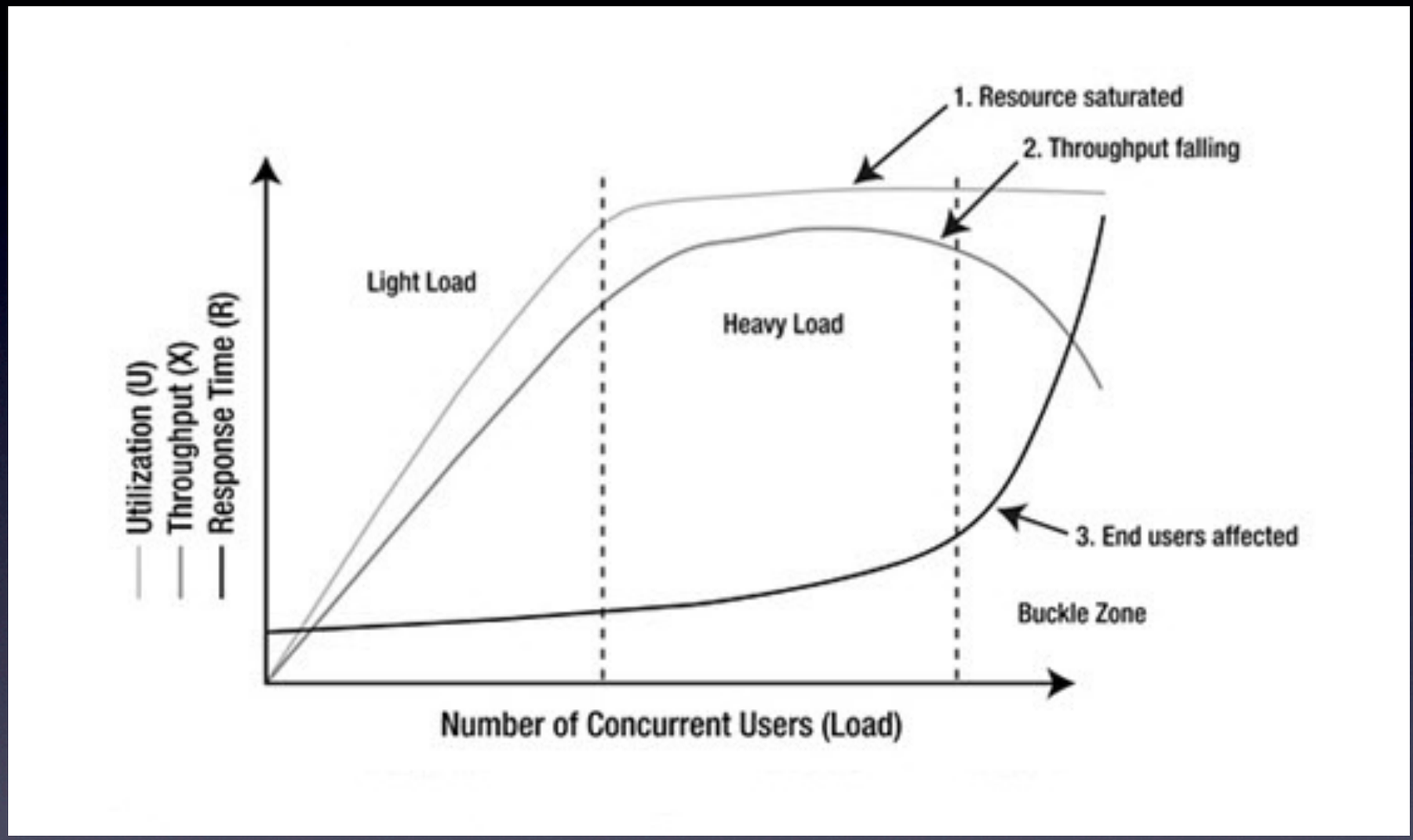
3. 应用自身优化

- 压缩模板大小
- 设置最佳并发
- 代码瓶颈优化
- 外部调用优化
- CPU优化
- 架构优化

压缩模板大小

- 模板大小和吞吐量成反比
- 删除空行、多余空格
- 长URL压缩、 用URL别名
  - 去掉’http:’
- 业务上去重
  - 案例：JSON去重减小50%

设置最佳并发



最佳并发 = ((CPU时间+CPU等待时间) / CPU时间) * CPU数量

代码瓶颈优化

- 找到影响性能的关键点进行优化
- 案例：某应用每个请求都抛异常吞异常
  - 去掉后， CPU使用率提高30%
  - 吞吐量提升近30%
  - 吞异常危害很大，需要从根源发现
    - Taobao JDK异常监测补丁

外部调用优化

- 并行RPC、并行搜索
  - 会降低响应时间，不会提升吞吐量
- 合并外部调用
  - 去冗余调用、合并接口
- 使用更优的序列化协议
  - Protocol Buffers
  - Kryo 比 PB 快20%

CPU优化

- GZip压缩级别
- 类中Field排序
  - 频繁使用的放一起
  - Cache Line优化
- 批量处理数组
  - 按行处理不要按列处理
  - 使用批量接口
- 使用乐观策略

架构优化

- 动态资源静态化
  - 分析页面，更新迟缓占比较大的部分放CDN
  - 案例：页面大小200KB -> 100KB
- 后台依赖前台化
  - 降低响应时间、提高稳定性
- 后端渲染前端化
  - 数据远小于页面，页面布局比较规则
  - 案例：响应时间减少25%、页面大小减小60%

架构优化

- DB依赖缓存化
  - 需要Center应用维护缓存
  - 对象缓存
  - 页面片段缓存
  - 整页缓存
  - HTTP缓存
  - 提高命中率

4. 前端性能优化

- 度量关键指标
- 前端优化实践

度量关键指标

- 页面下载时间
- 开始渲染时间
- 首屏时间
- domReady
- onLoad

- TSlow
- 阿里度
- 阿里测
- 页面埋点
  - Navigation Timing
- PhantomJS

前端优化实践

- Yahoo 34条军规
- 首屏优先，渲染优化
  - BigRender
- 延迟加载，按需加载
- Keep Alive、mod_pagespeed

Q&A