分享，移动，生活

开放、动态网络分享、综合性网络服务

PDF-XChange
Click to buy NOW!
www.docu-track.com

开放平台的资源分配与多级缓存体系的优化

飞信开放平台技术总监
互联网产品首席架构师
孙朝晖
http://weibo.com/steadwater

2011年9月

SACC2011

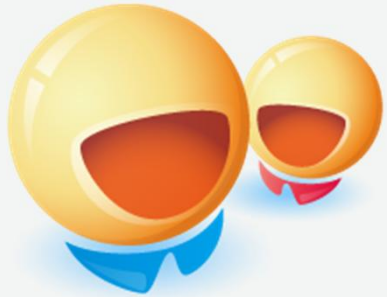
私人广告

steadwater
http://weibo.com/steadwater
北京,朝阳区
博客：http://space.feixin.10086.cn/52885975
飞信互联网产品首席架构师，关注一切互联网行业动态

- 首先希望遭到关注并通过微博交流
- 本人职责
  - “飞信开放平台”总体技术架构设计
  - 飞信互联网相关产品的技术规划
  - 飞信技术社区建设，特别欢迎与同仁广泛交流

目录

- 飞信开放平台对合作伙伴的资源分配
- 缓存系统在SNS类应用中的地位
- 飞信SNS应用中缓存体系总体架构
- 资源类缓存体系建设的技术特点
- 数据类缓存体系建设的技术特点
- 缓存体系的监视方法



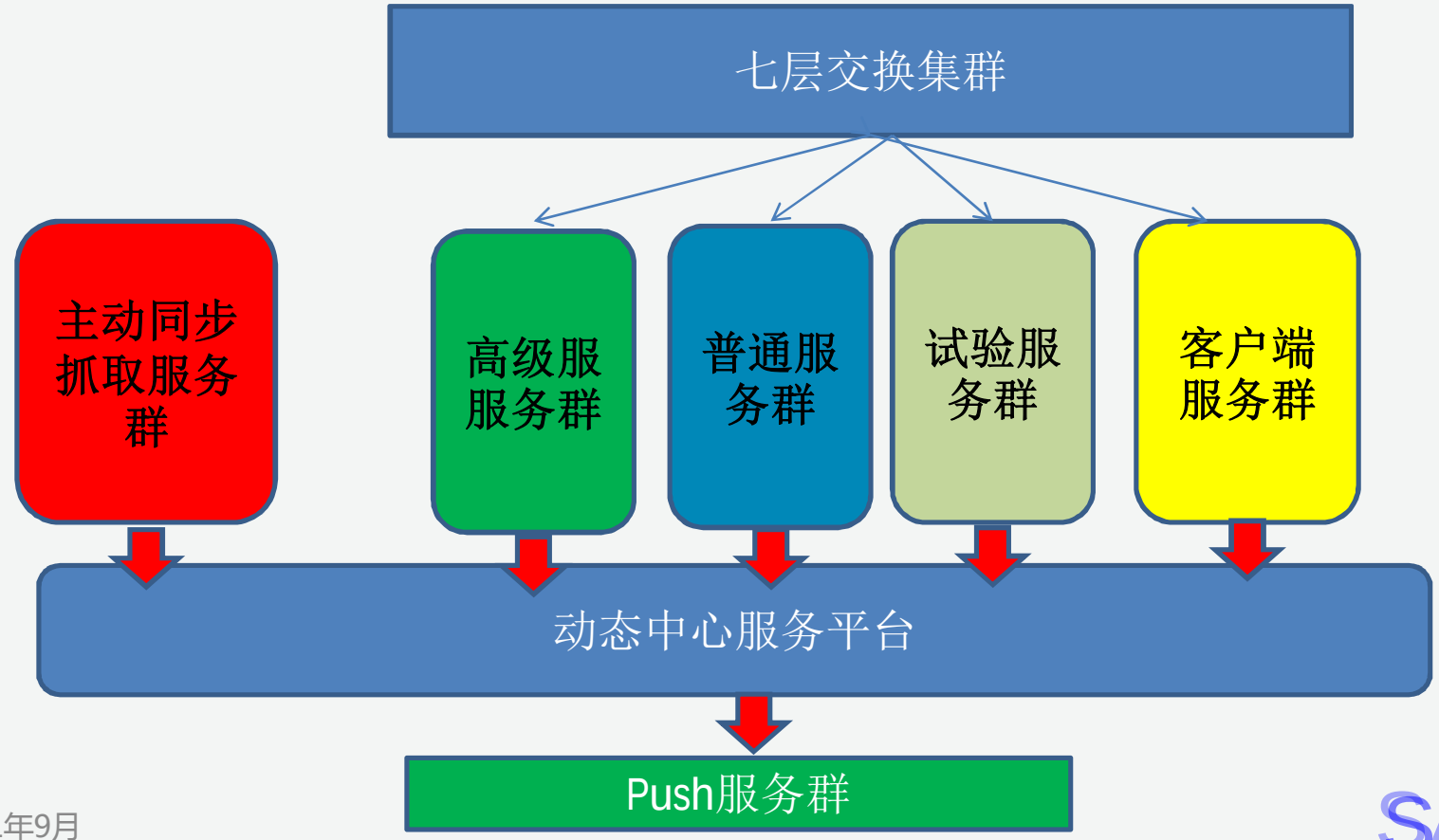
飞信产品体系中的SNS与开放平台

飞信开放平台合作伙伴的数据通信方式

- 飞信主动同步类型
  - 飞信利用第三方服务开放平台功能拉取TimeLine，并发布Feed（如新浪、腾讯微博）
- 飞信被动同步类型
  - 第三方服务调用飞信开放平台API将动态主动推送到飞信开放平台上（多数互联网合作伙伴）
- 双方相互同步类型
  - 双方相互向对方推送动态（如移动微博等）
- 客户端类型
  - 飞信以及第三方开发的PC，手机客户端，收发API据需要

飞信开放平台对OPEN API的整体分配策略

- 飞信开放平台通过基于RESTFUL的OPENAPI 提供数据通信接口，根据不同的限制区域和服务级别，分成不同的服务器群集



飞信开放平台资源分配

- 每IP 并发限制，通过Nginx7层交换完成
  - limit_zone one $binary_remote_addr 10m;
- 每应用，每用户的每小时访问频度通过应用程序控制
  - 试验区应用采用先回写计数后响应策略
    - 计数器资源存储，先Redis，定期刷新DB
  - 其它区域采用先响应后会写计数策略
    - 普通服务器集群 计数检查→响应->回写计数，存储资源先redis,定期刷新DB
    - VIP集群， 状态检查 ->响应 异步回写->计数->通知状态，状态资源管理 先shmop 后 Redis

缓存体系的重要性

- 承担了80%以上请求处理过程中的资源与数据访问需求
- 每台Web Server承担4000RPS 峰值首页加载，其中的关键性能保障
- 保证数据库运行安全和持久化数据的安全
- 减少服务器带宽消耗

飞信SNS应用中的缓存体系

资源缓存体系

- 浏览器缓存
- CDN
- 7层交换前部缓存
- 图片服务器的Web应用缓存

减少数据流量，提高Web加载速度，提升使用体验

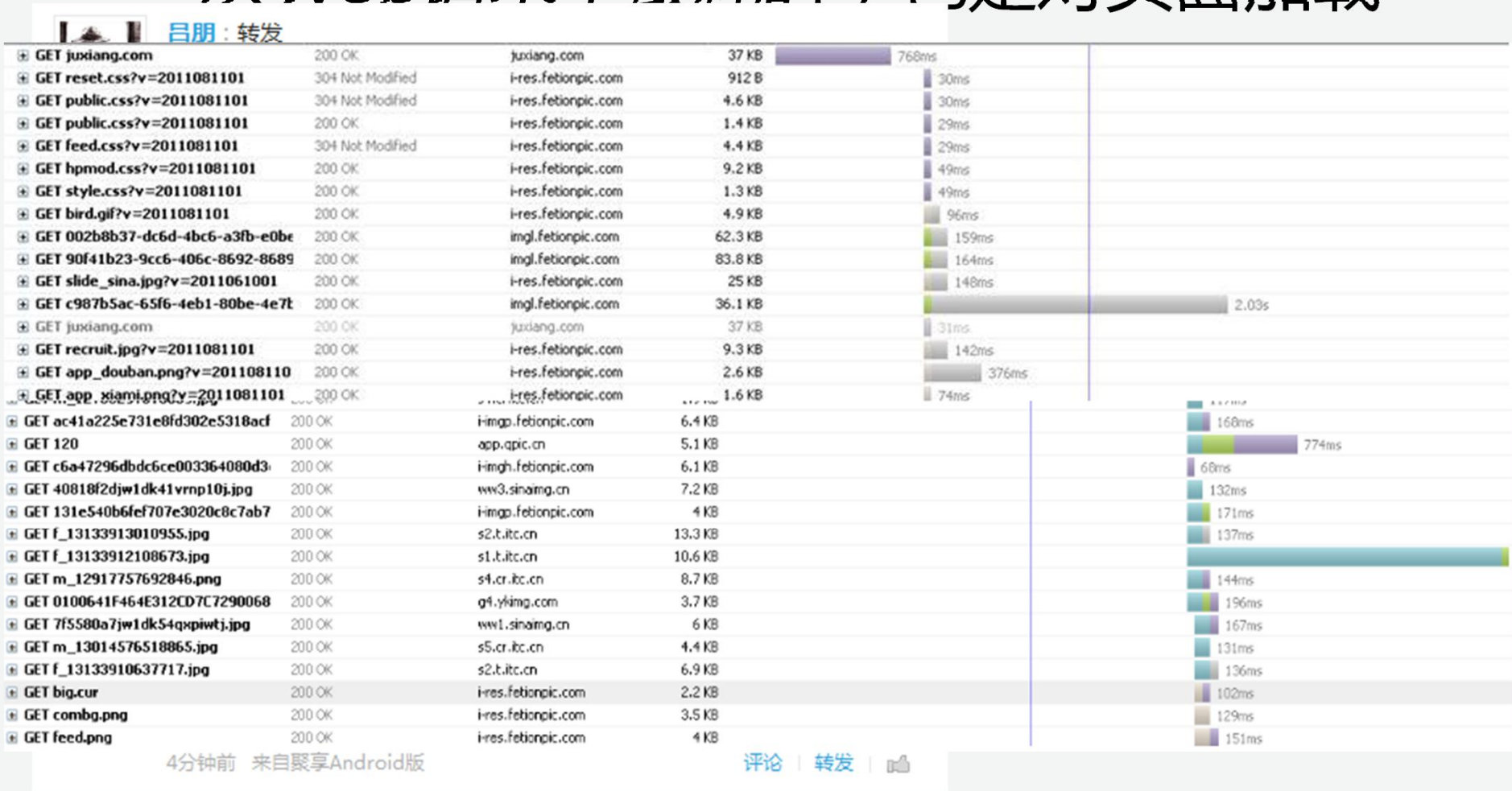
数据缓存体系

- 浏览器本地缓存
- Web服务器输出缓存
- Web Server 数据缓存
- Redis缓存服务器
- 中间件服务器本地缓存
- 数据库服务器的BufferPool

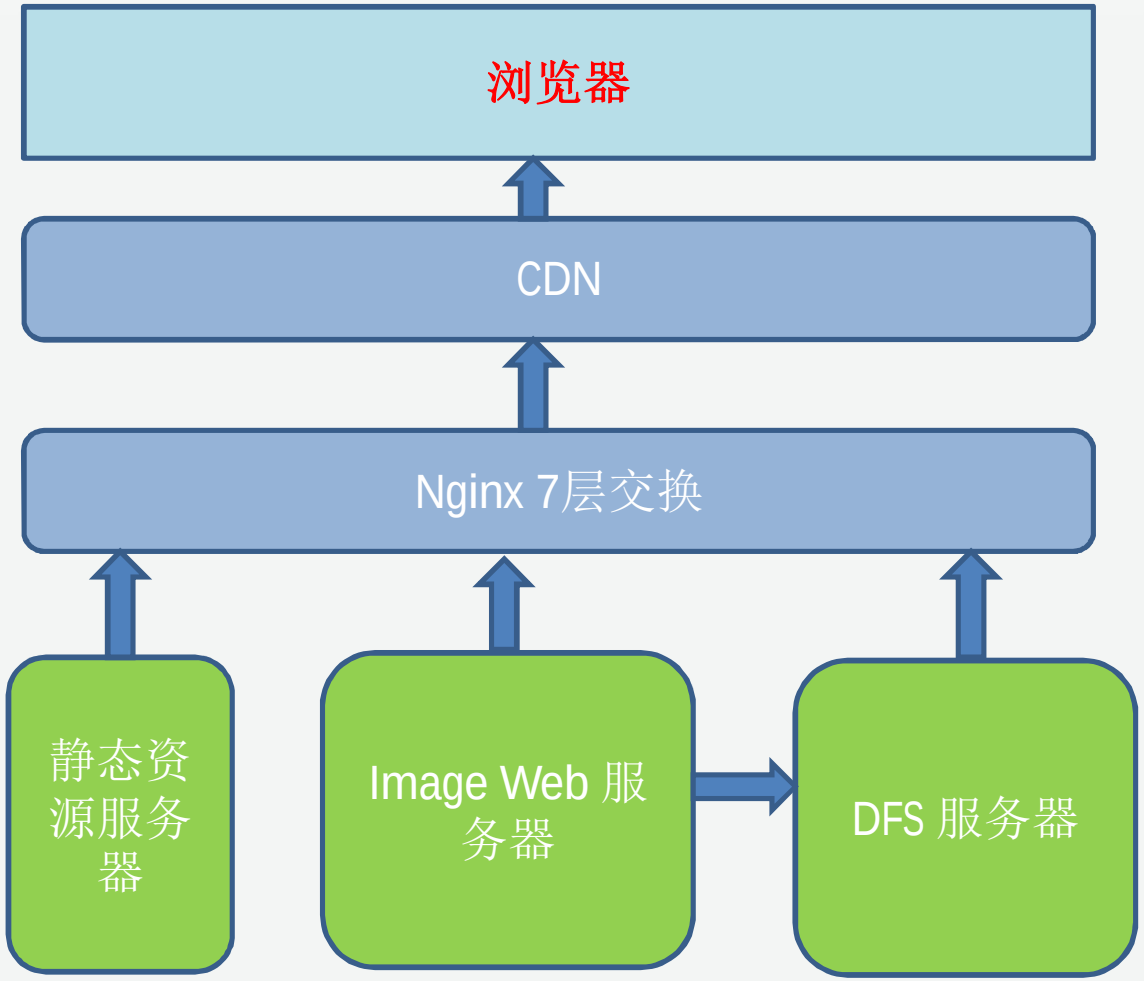
减少数据库直接读取，减少重复计算，降低计算负荷

资源缓存体系的重要作用

- 一次Web请求中最耗时，也是对页面加载



资源缓存的技术体系

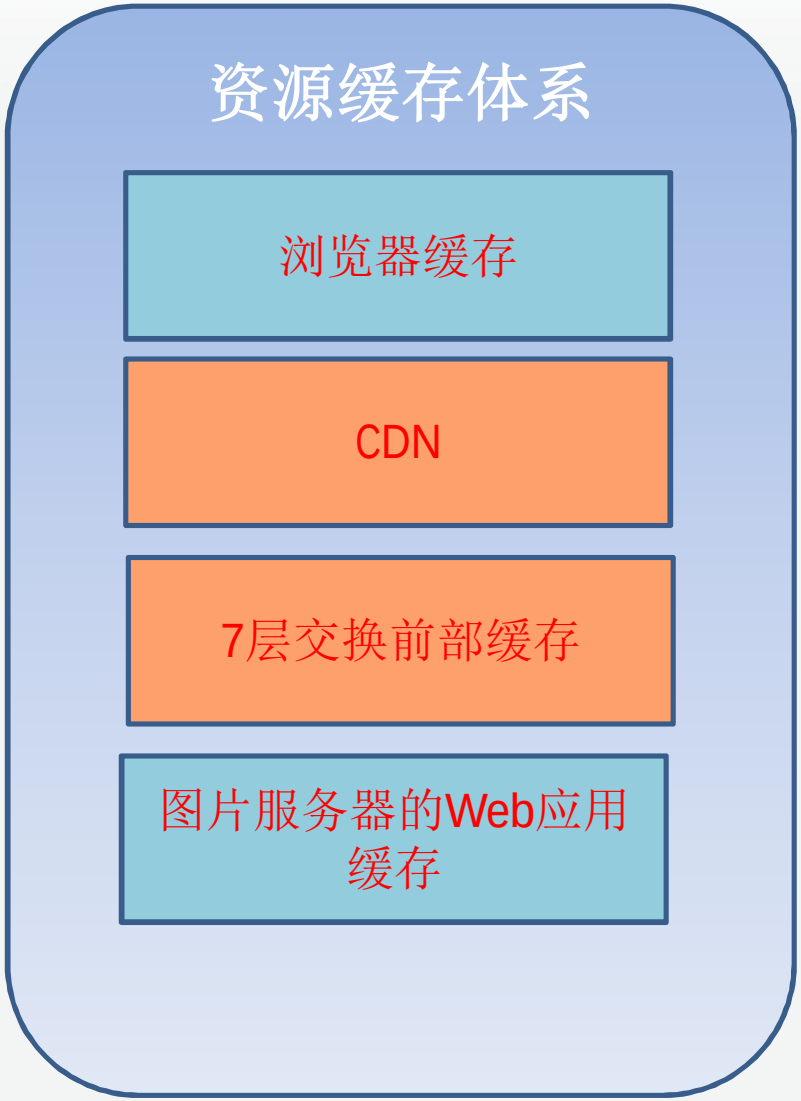


浏览器资源缓存的主要技术

- 资源统一预加载

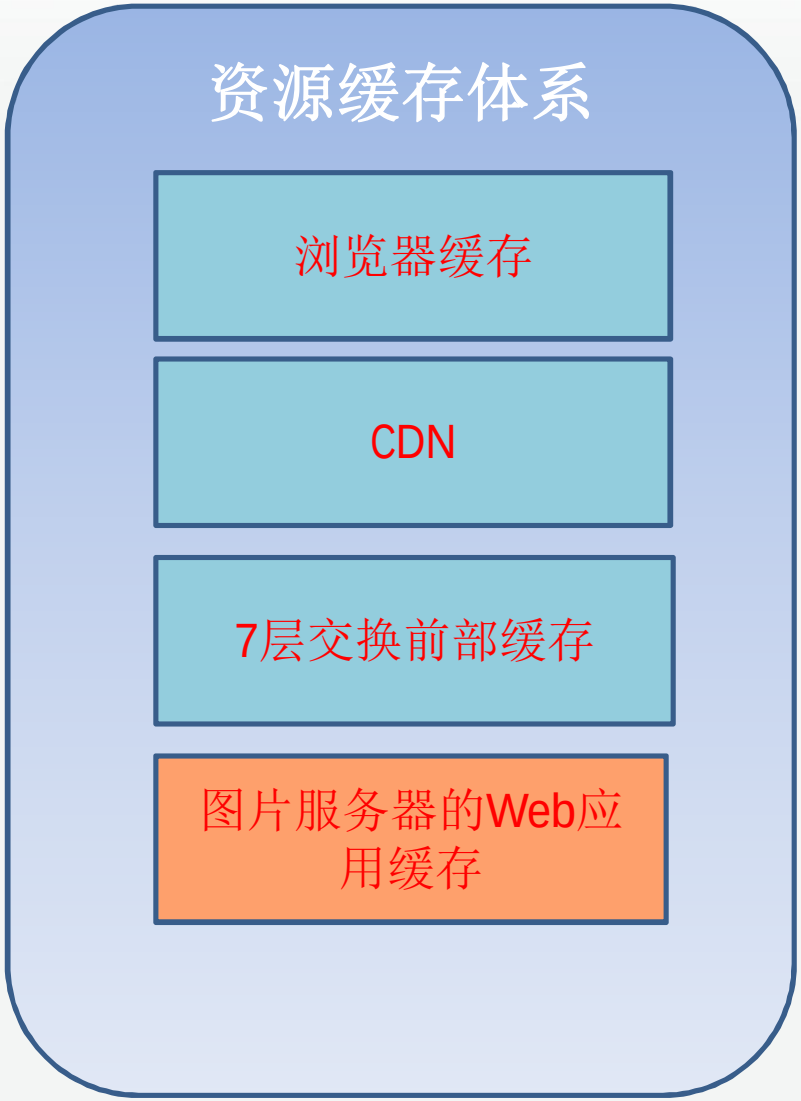
（典型场景：用户头像）

浏览器资源缓存的主要技术

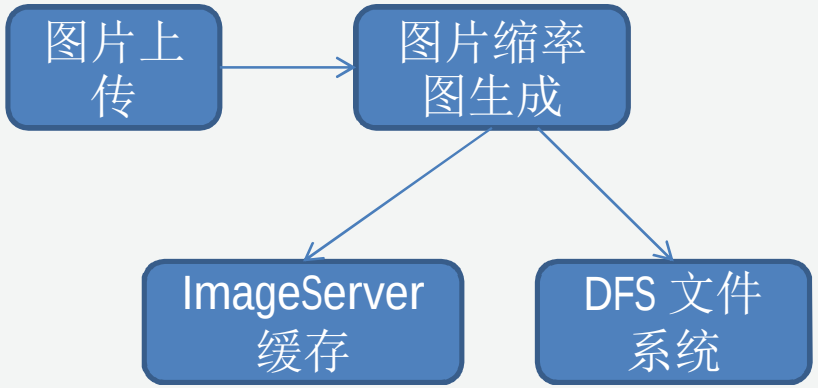


- 动态与静态分开域名，采用不同的Cache-Control header测试，配合CDN
- 采用基于基于Nginx7层交换，前部缓存主要是针对分布式文件系统存储的图片进行优化，减少网络跳数
- 由于ImageServer本身有缓存存在，避免出现二次跳转

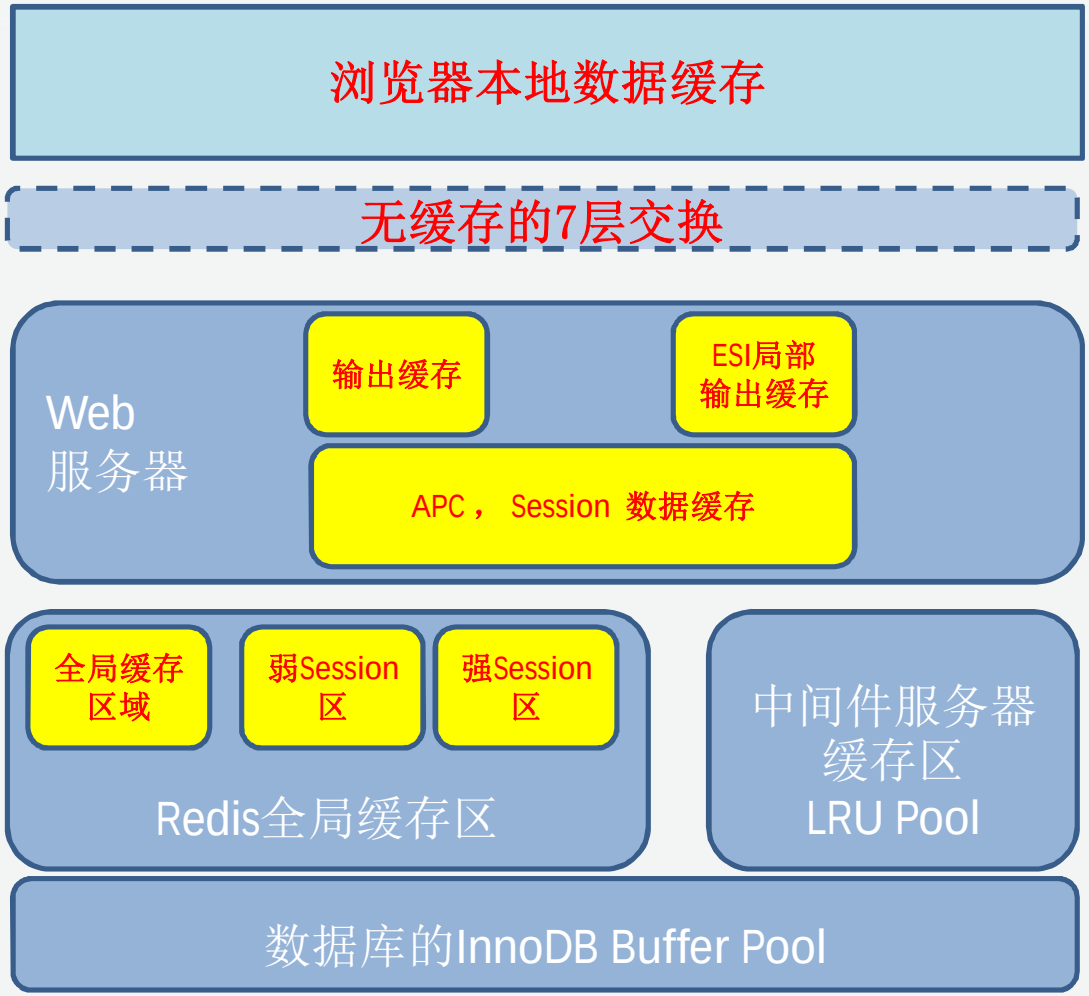
浏览器资源缓存的主要技术



- 图片服务器缓存主要用来处理最近缩略图，因为最近处理的图片往往是最近需要的图片，不等DFS同步完成，就需要使用



数据类缓存体系的总体技术架构



本地浏览器缓存的应用

- 本地浏览器缓存：
  存放用户本地JS计算使用的数据
  应用场景：@自动提示需要的姓名与拼音缓存

浏览器本地Cache使用的技术

- JS开发技术

  使用IE的UserData对象

  使用FireFox，Chrome支持的SessionStorage
- 另一种选择：FlashCache
  - FlashCache 的浏览器兼容性较好
  - 可以设计出更加复杂的加密方法
- 期待HTML5标准的大规模导入

Web 服务器的输出缓存

- 采用PHP中 ob_XXX 函数控制
- Web服务器数据缓存与7层交换前部缓存相比
  - 输出到文件
  - 具有更多的可控性
  - 用于缓存静态或者半静态化页面

Web 服务器的输出缓存

- ESI 局部输出缓存
  - 在Web服务器的Nginx ESI Module直接实现，降低7层复杂性
  - 实现局部静态化，与反向代理可配合
  - 典型应用场景：主页主体结构与局部结构分离

Web服务器数据缓存

- APC
  - 存储基于Shmop
  - 代码缓存
  - 典型应用场景：全局统一的不易变内容
    例如：全局配置（频度限制等）
- Session
  - 本地Session仅用于保存短声明周期过程数据
  - 存储基于文件
  - 典型场景：OAuth认证过程中间Token

Web服务器数据缓存

- APC
  - 存储基于Shmop
  - 代码缓存
  - 典型应用场景：全局统一的不易变内容
    例如：全局配置（频度限制等）
- Session
  - 本地Session仅用于保存短声明周期过程数据
  - 存储基于文件
  - 典型场景：OAuth认证过程中间Token

Redis缓存体系

- Redis缓存与数据区域对应，隔离影响范围，防止全面雪崩
- Redis分成了三个区域

- 全局区域
  - 特点：全局共用，对各Web，中间件Server等价，重建成本低
  - 典型应用：短连接的地址映射缓存
  - 构建方法：多Server多进程一致性Hash，无Persist，无复制

Redis缓存体系

- 弱Session区
  - 特点：与用户相关，要求一致性低，重建成本低
  - 典型应用：每用户的Session

    典型对象：好友列表、隐私设置，50条最新动态（满足Ajax轮循加载）
  - 构建方法：采用多对一复制方案，无永久存储，多主一备份，备份多实例

Redis缓存体系

- 强Session区
  - 特点：与用户相关，要求一定程度的一致性，构建成本高，存储占用量不高
  - 典型应用：用户的在线队列，用户调整动态同步的抓取优先级
  - 构建方法：采用DiskStore方案，一对一复制

中间件服务器本地计算缓存

- 主要应用场景：
  - 记录异步调用的Transaction
  - 写入缓冲：例如在处理新动态通知中，在多个动态分发完成后，汇总通知条数，想事件服务器分发事件通知，避免大量频繁通知
- 主要技术
  - 采用基于Apache Common Pool的对象池
  - 采用LRU 淘汰策略

数据库存储的要点

- 索引与Feed存储分离
- 索引（TimeLine、Topic）等采用Mysql，Feed采用MySQL handlerSocket
- 使用MySQL handlerSocket 充分利用Buffer Pool
  - 调大 innodb_buffer_pool_size 至内存70%
  - 采用innodb_flush_method=O_DIRECT减少操作系统缓存和SWAP区域占用
  - 利用Secondary Buffer Pool补丁充分利用SSD（Percona 的XtraDB 已经整合）

缓存体系监视—注意软性问题

- 关键是建立因果图谱，多角度监视，提早发现问题
  - 举例：DFS服务器的IOPS提高->检查反向代理前部缓存命中率
  - 举例：基于Xhprof记录的PHP Feed服务RPC调用，平均事件增加，说明缓存命中率降低 -> 监视Redis服务器内存占用和工作并发网络连接
  - 举例：好友数据库服务的IOPS升高->是否所有Redis服务器故障，导致某个分区的缓存失效
- 在完整的缓存体系中对线上状态进行监视

总结

- 资源缓存设计很重要，资源速度决定加载速度
- 数据缓存对于计算资源的合理分配很重要，避免重复计算
- 缓存体系永远是优化的重点

感谢！