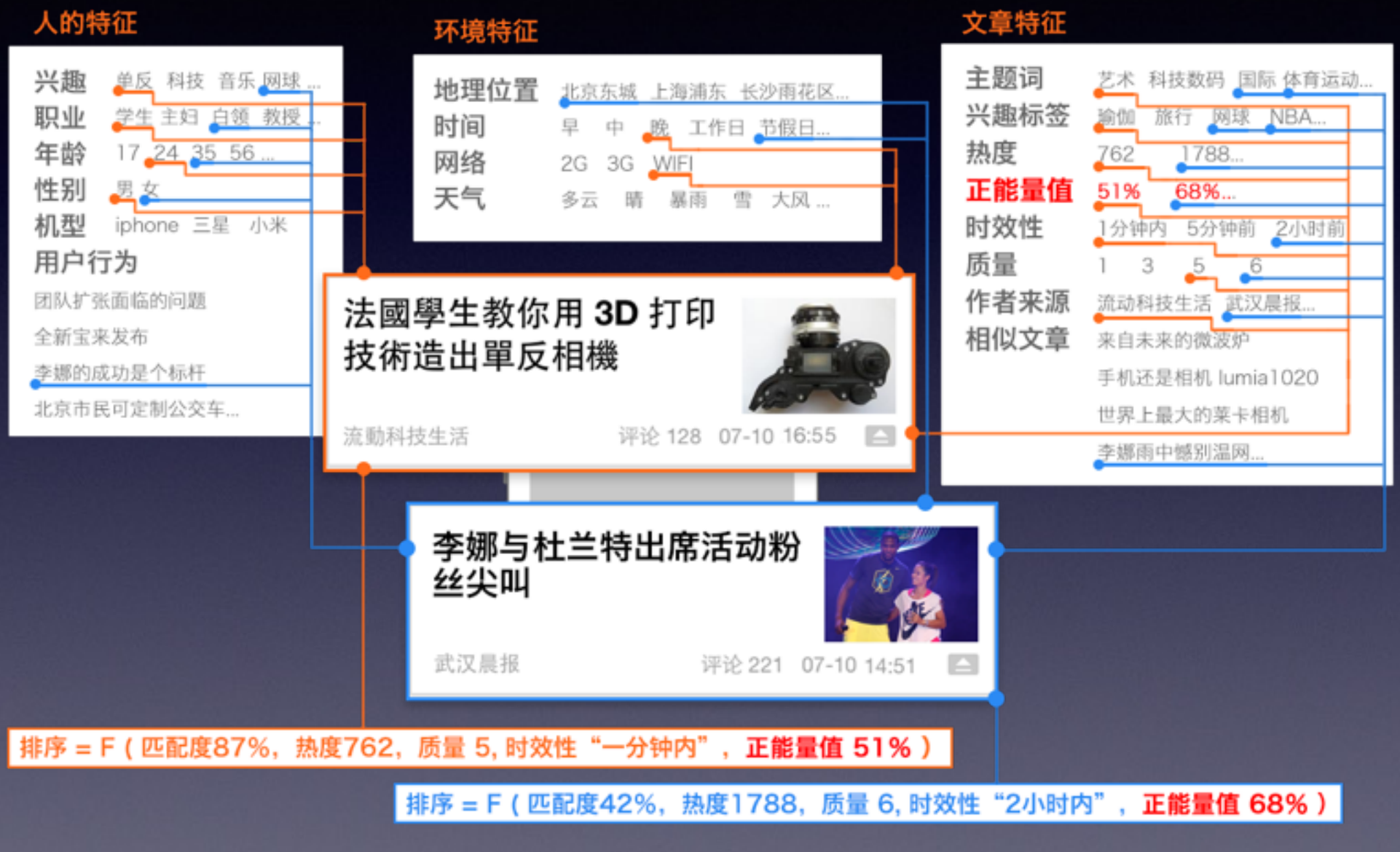
今日头条 User Profile 系统架构实践

丁海峰

推荐系统是怎样工作的



- 高质量的用户特征是做好推荐的关键之一

什么是好的推荐效果

- 点击率，但不仅是点击率
- 内容高质量，丰富多样，有惊喜感，能够帮助用户探索兴趣，快速反馈又不能过度灵敏，etc.
- 长期目标
  - 用户：有兴趣，有收获，愿意长期使用
  - 生态：鼓励良币，驱逐劣币

需要怎样的用户特征

- 人口学：性别、年龄、地域，etc.
- 内容特征：category, topic, keyword, entity, etc.
  - 喜欢 & 不喜欢
  - 短期 & 长期
- 协同特征：相似用户
- 其它：e.g. 逼格

My Profile

用户基本信息

性别	展开>>
male	0.9452

年龄段	展开>>
24-30	0.3068

用户订阅来源		
越玩越野	2015-03-31 20:05	麻省理工科
图虫人像摄影	2015-01-19 14:15	图虫人文摄
拍娃党	2014-11-06 13:20	什么值得买

用户喜欢分类信息:

用户喜欢顶级分类(impr decay) @召回			
news_tech	1250.8073	news_world	389.8
news_travel	270.3227	news_society	255.4
news_military	107.3003	digital	74.39

兴趣分类来源(long term) @召回		
科技:36氪	1080.8857	科技:CSDN资
科技:虎嗅网	537.5638	科技:DoNew
科技:虎嗅	417.8447	科技:界面
科技:iDoNews	371.3731	科技:人人都是
国内:新京报	340.3811	财经:36氪

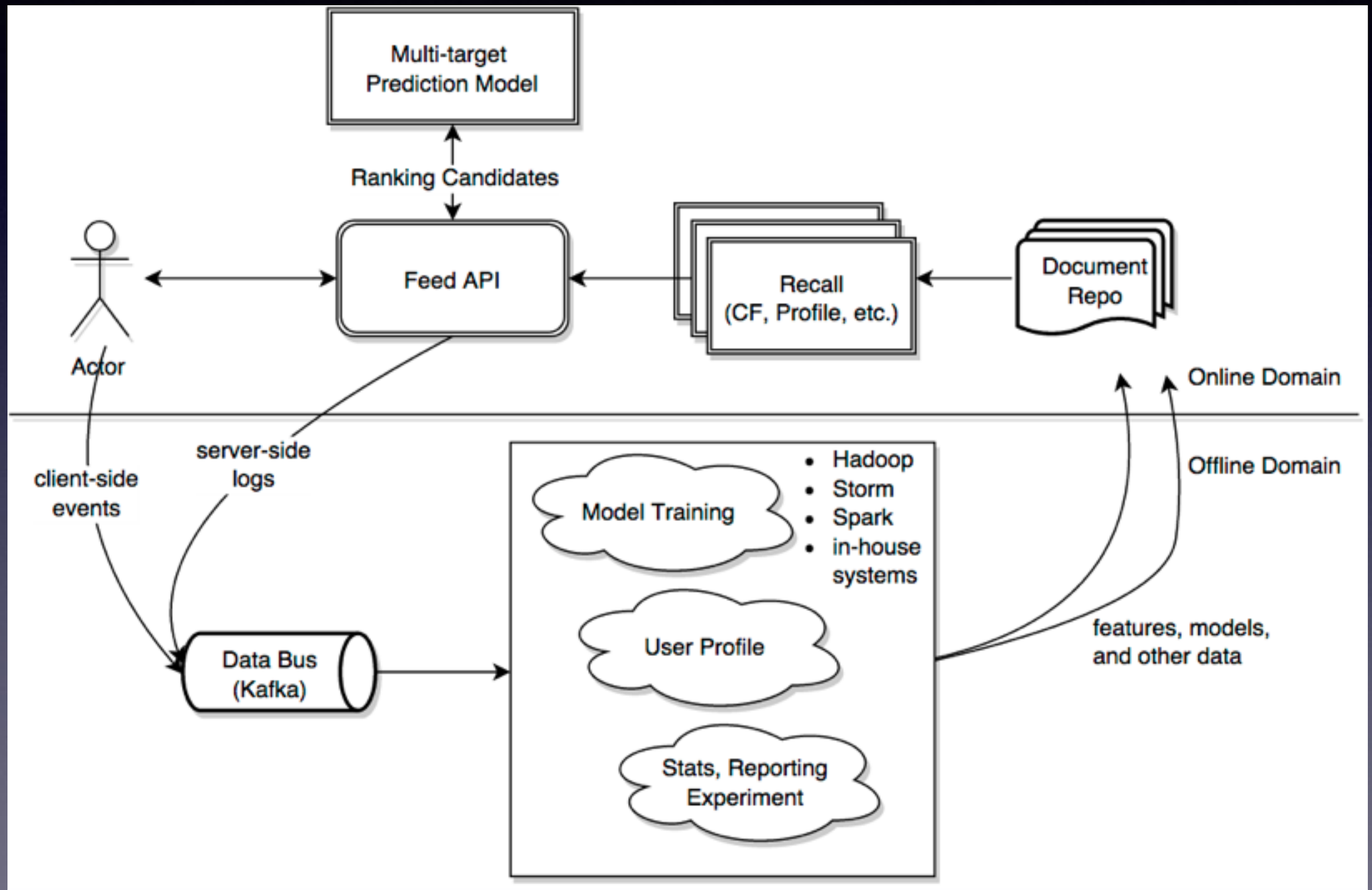
兴趣汽车品牌 @召回			
丰田汽车	288.9407	大众汽车	144.20
日产汽车	56.7518	福特汽车	47.662
本田技研工业	24.0639	起亚汽车	22.765
现代汽车	17.3621	路虎	15.920

兴趣汽车价格 @召回			
35-50万	630.7637	25-35万	541.1
70-100万	217.4961	15-20万	120.0
5万以下	7.7931		

算法

- 点击加权 & 未点击惩罚
- 热门点击降权
- 时间衰减
- 噪声过滤：spam，标题党等
- 其它精细的调优

System Overview



Our Challenges

- 存量用户量大，用户行为数据量巨大
- 期望快速反馈
- Online serving storage: 读写吞吐高，时延低且可预期

一些数字

- 用户行为数据
  - 历史存量：500TB+ （压缩后）
  - 每日新增：1TB+ （压缩后）
    - 高峰时段：400K msg/s （Overall）
- Profile Server
  - feature 数量：200+
  - 容量：单副本 12TB
  - 请求次数：1.2M qps

Batch Approach

- Batch 计算，MySQL 存储
- Daily Mapreduce Workflow
  - 对每日活跃用户，抽取该用户过去两个月的展示和动作，从0开始重建该用户的 user profile

问题

- CPU 密集，大量重复计算
- 高写入吞吐，MySQL 瓶颈
- 更新不及时，用户动作反馈到 user profile 可能会长达两天

Streaming Approach

- A Storm Topology
- mini-batch processing
- 固定 10 min 时间窗口
- 获取用户上一次 profile 状态作为 base
- 做时间衰减
- 使用新增的展示和动作更新 profile

Pros.

- 用户动作反馈周期大大缩短：2天 -> 10分钟
- 减少重复计算，节省大量计算资源

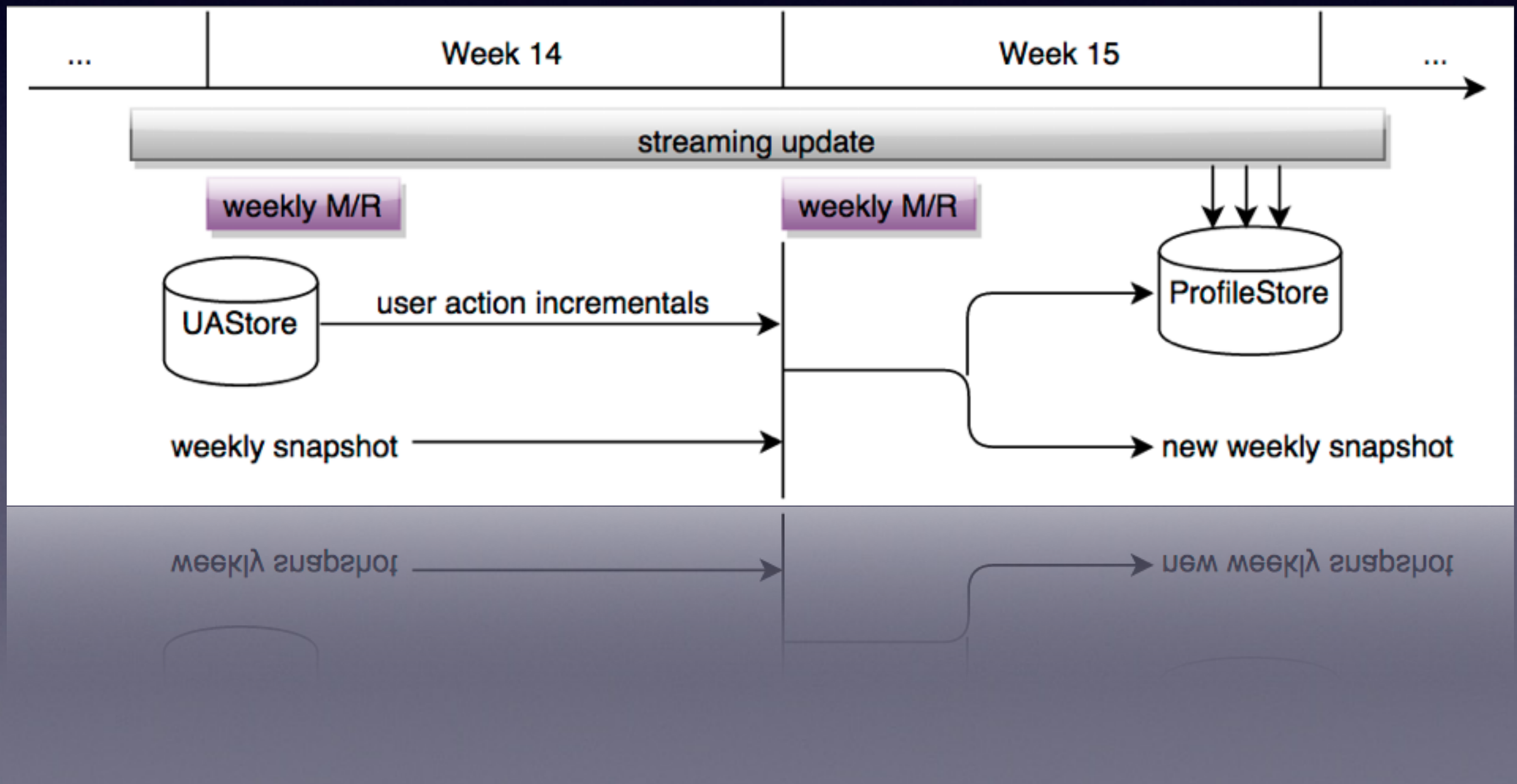
Cons.

- 统计类特征会有延迟，即时值 不等于 最终值
  - 点击延迟：用户可能在展示之后一段时间才会点击
  - 热点文章点击降权：热点文章，在文章发布初期点击的用户被错误的认为点击了冷门文章
  - 文本特征延迟：spam 标题党等特征判定会有延迟
- 算法上线可能会有异常，需要回滚 user profile
  - batch 更容易，覆盖新数据即可
  - streaming 计算需要 replay 长时间的历史数据，开销反而更大

Hybrid Approach

- 在 Streaming 更新的基础上，引入周级的 Batch 校准
- 以上一次 Batch 计算产出的 user profile 快照作为 base，replay 其后产生的用户展示、动作并更新 user profile

Data pipeline



算法接口抽象

- Storm & Mapreduce 计算模型有差异，但核心算法一致
- 抽象核心算法接口，算法实现保持一致，避免维护两份不同的 Code
  - `update_profile(base_profile, impressions, actions) => new_profile`
- Re-thinking: Spark & Spark Streaming

UserActionStore

- 基于 HBase 实现的，实时、可随机读写、可扩展的用户行为存储
- 以 (hash, user_id, timestamp) 为 RowKey
- 访问接口
  - `UAStore.get_impressions(uid, start_time, end_time)`
  - `UAStore.get_actions(uid, start_time, end_time)`

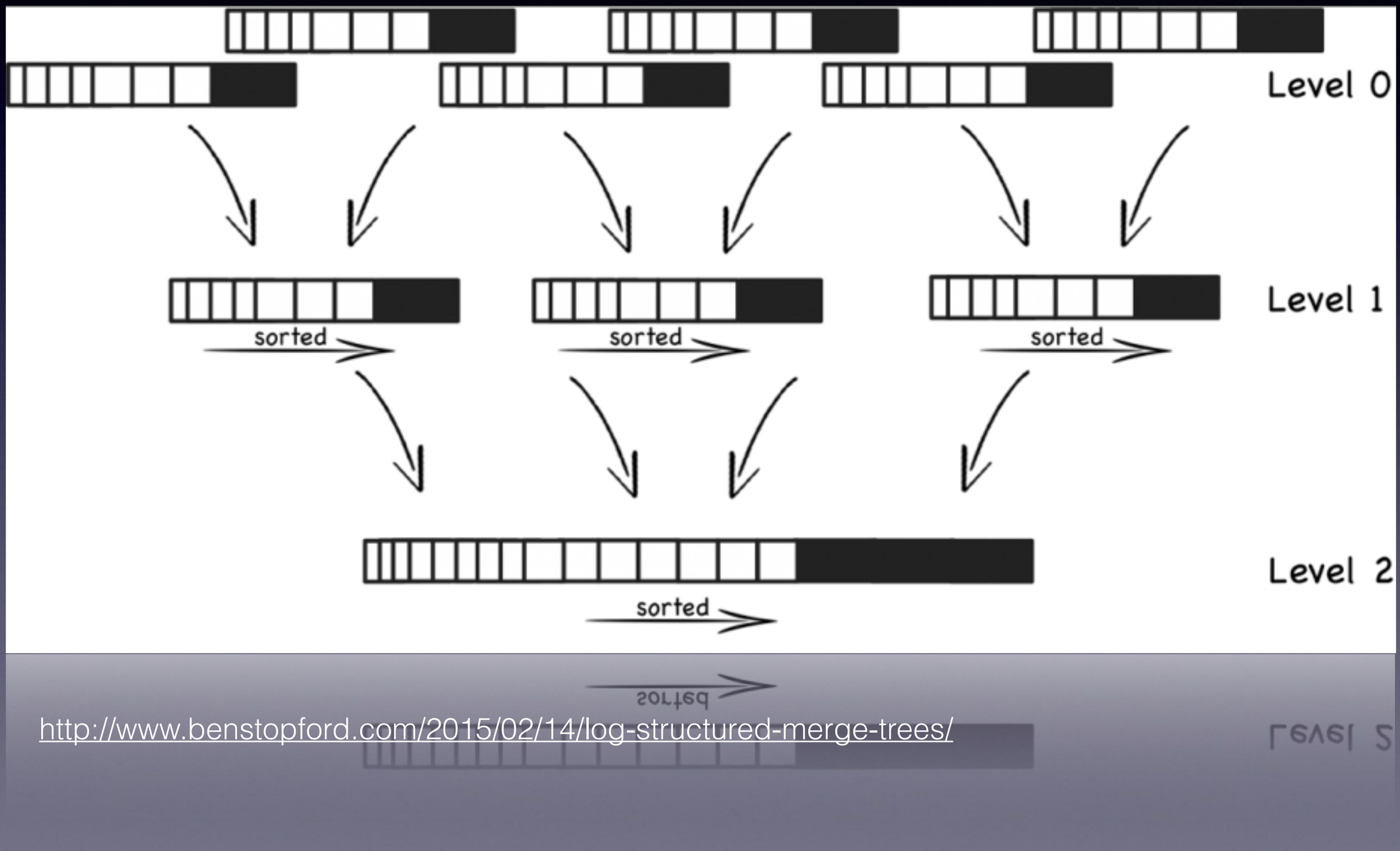
ProfileStore

- 需求
  - 读请求稳定低延迟：serve 在线访问请求
  - 高写入吞吐：batch/流式更新
  - 数据多副本
  - 数据量巨大，需要可水平扩展

Springdb

- twemproxy + rocksdb
- 主从同步 双副本
- twemproxy reload 配置实现主从切换
- 重写 compaction 策略，降低写放大系数
- latency 长尾调优，减少超时

LSM-Tree Compaction



http://www.benstopford.com/2015/02/14/log-structured-merge-trees/

Fight with Write-Amp

- rocksdb: LSM-Tree, compaction, 写放大10x~, SSD 寿命
- Our Solution:
  - 限制只使用 L0~L1，减少 compaction 层次
    - Customized Level Style Compaction
      - L0 小文件 compaction => L0
      - L0 大文件 full compaction => L1
    - 写放大 10x~ => 2~3x，读放大、空间放大可接受
- https://github.com/facebook/rocksdb/issues/210

Performance

- 数据量：压缩后 12TB 数据 x 2副本
- QPS：read 140K，write 55K (cache 后)
- 时延：avg 500us, pct99 5ms
- 机器：16台机器，SSD，存储瓶颈

Lessons we learned

- Batch + Streaming 是一种常用的模式
  - 合适的基础设施和业务抽象，减少重复
- 深入理解 workload，选择合适的存储系统
  - Rocksdb on SSD rocks!

Thanks for listening.

Joint work with many others.

Q&A Time