hbase 上搭建广告实时数据处理平台

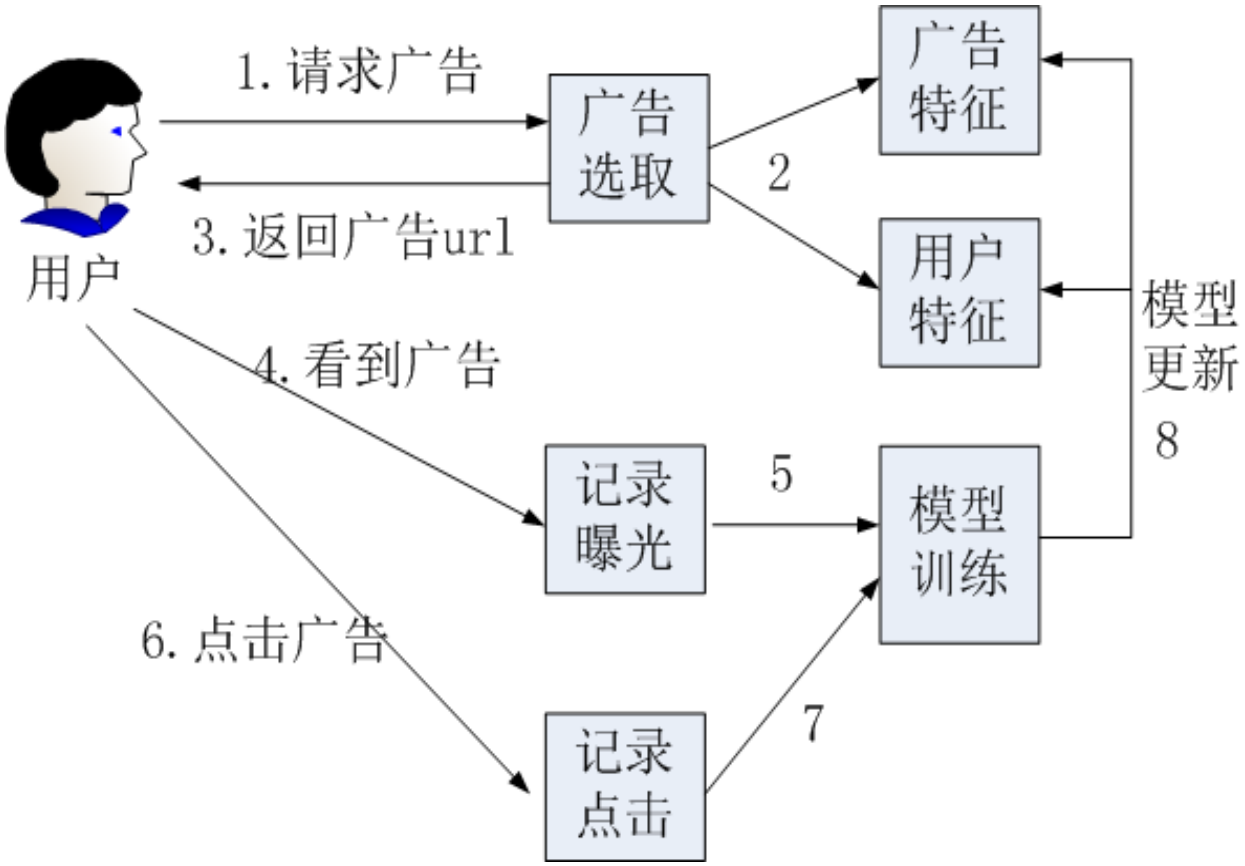
腾讯广点通 WRAPENGINE 项目

李锐

广点通

- 基于腾讯社交体系的效果广告平台
- 流量
  - QQ
  - QQ 空间
  - 微信
  - 移动联盟
  - 等等
- 智能精准定向
  - 用户特征
- http://e.qq.com

背景：广告数据流


1. 请求广告
广告选取
广告特征
2
用户特征
3. 返回广告url
用户
模型更新
8
4. 看到广告
记录曝光
5
模型训练
6. 点击广告
记录点击
7

背景

- 实时 storm+ 离线 mapreduce 计算，多维度分析

- 难点：如何把大量的信息从广告选取阶段传递至广告点击的模型训练？
  - Proto buffer
  - 实时日志关联

技术选型 (1)

方案一：

- 广告请求时把信息编码到广告 url 里面。
- 曝光和点击的时候再上报。

- 缺点：
  - 广告 url 过长，“偷”用户流量，移动端优质 APP 难以接受。
  - 不能放入过多的信息。

技术选型 (2)

方案二：

- 在 storm 里面实现流式关联

- 缺点：
  - storm 数据无法落地，需要有额外的存储方案解决 storm 重启的问题
  - 20% 的曝光在广告请求 20min 以上，内存无法缓存如此长时间的请求数据。

技术选型 (3)

方案三：

- 在 storm 里面实现流式关联
- 中间数据写到磁盘排序文件，比如 sstable
- 追加日志
- 排序文件的合并
- 读写数据缓存
- 过期数据淘汰

- 缺点：
  - 开发周期过长，跟 hbase 功能重合

技术选型 (4)

方案四：

- 广告请求信息写入 hbase
- 广告曝光和点击时查询 hbase，做日志关联

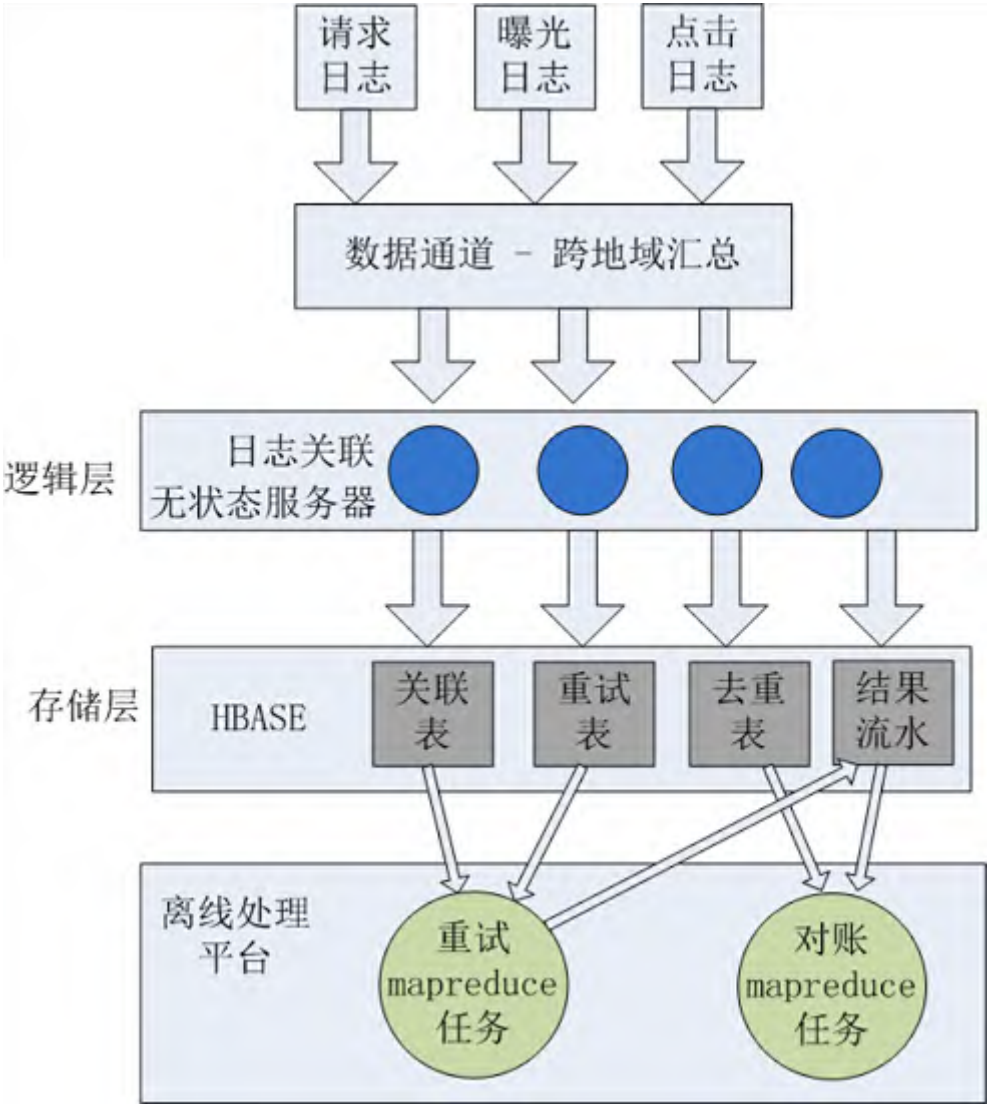
- 优点：
  - 大部分的查询能命中 cache

技术选型 (5)

为何不用：

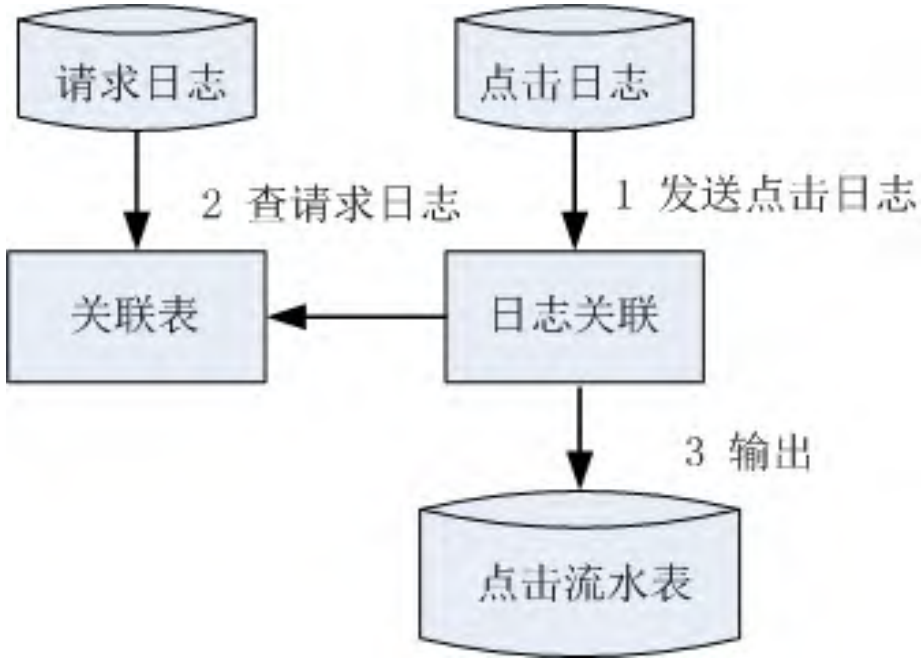
- Impala
- Spark
- Cassandra

技术方案 – 整体框架


请求日志
曝光日志
点击日志
数据通道 – 跨地域汇总
逻辑层
日志关联无状态服务器
存储层
HBASE
关联表
重试表
去重表
结果流水
离线处理平台
重试mapreduce任务
对账mapreduce任务

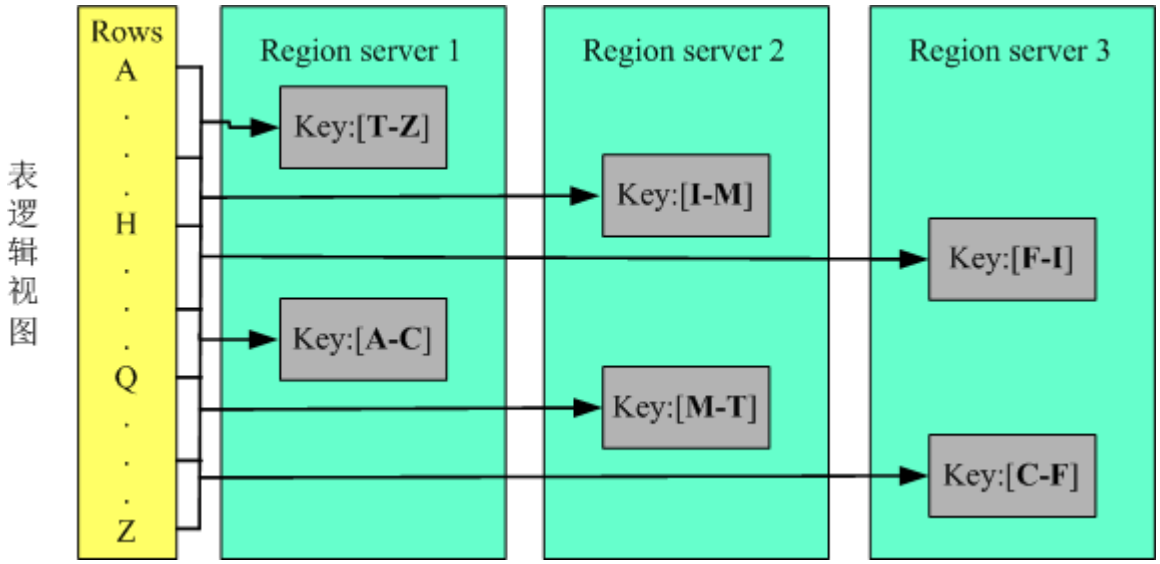
技术方案 – 整体框架

- 每次广告请求分配唯一 id
- 关联表：
  - 收集请求日志，按 id 为 key 写到 hbase 关联表
  - 曝光和点击日志，按 id 查关联表



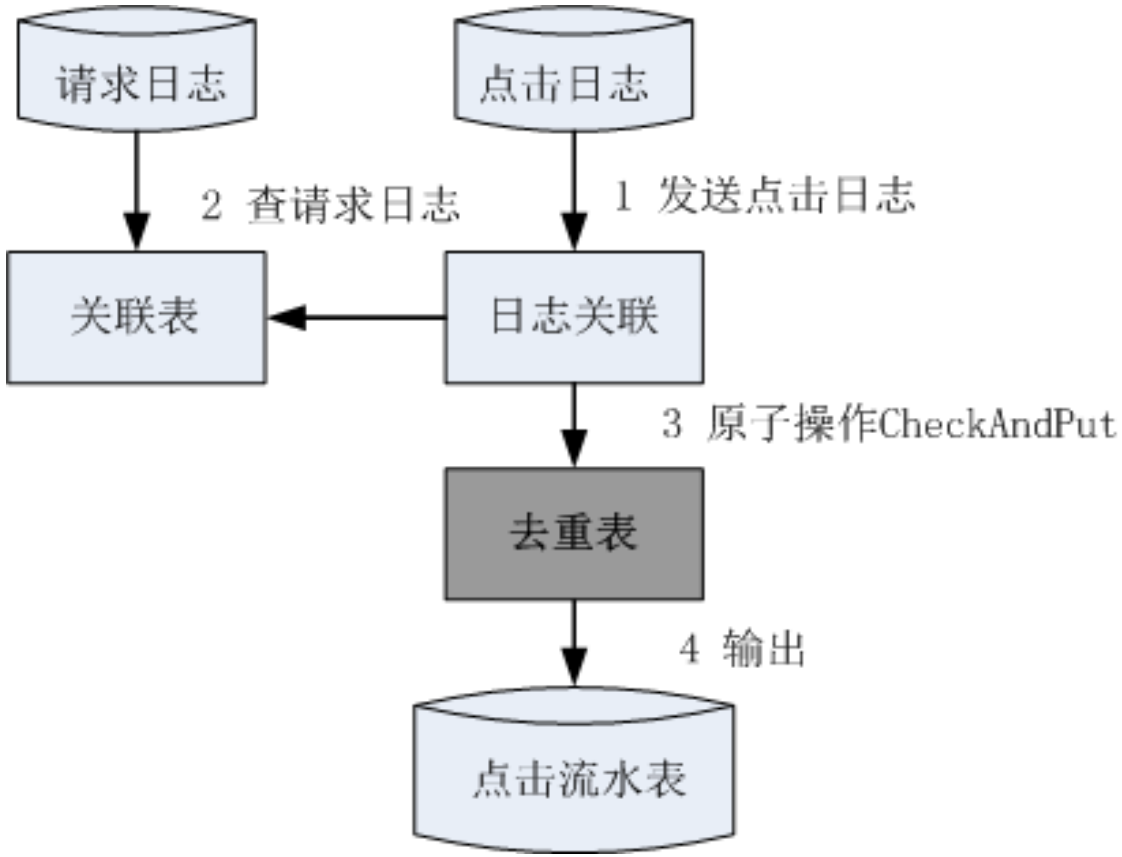
技术方案 – key 的设计

- 避免热点
- Key 前缀加分桶打散数据
- 支持按时间段小批量构建模型
- Key 中间放时间戳



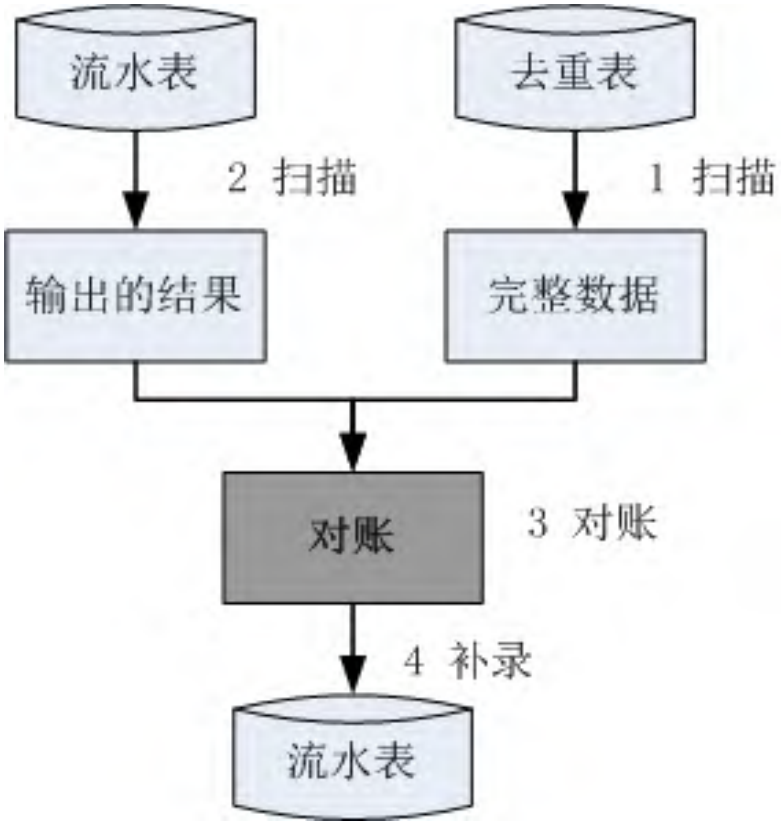
技术方案 – EXACT ONCE

- 如何保证点击等计费信息不重不丢?
  - 上游发送数据失败时，进行重试。
  - logjoin 输出到下游的时候通过 hbase 进行去重



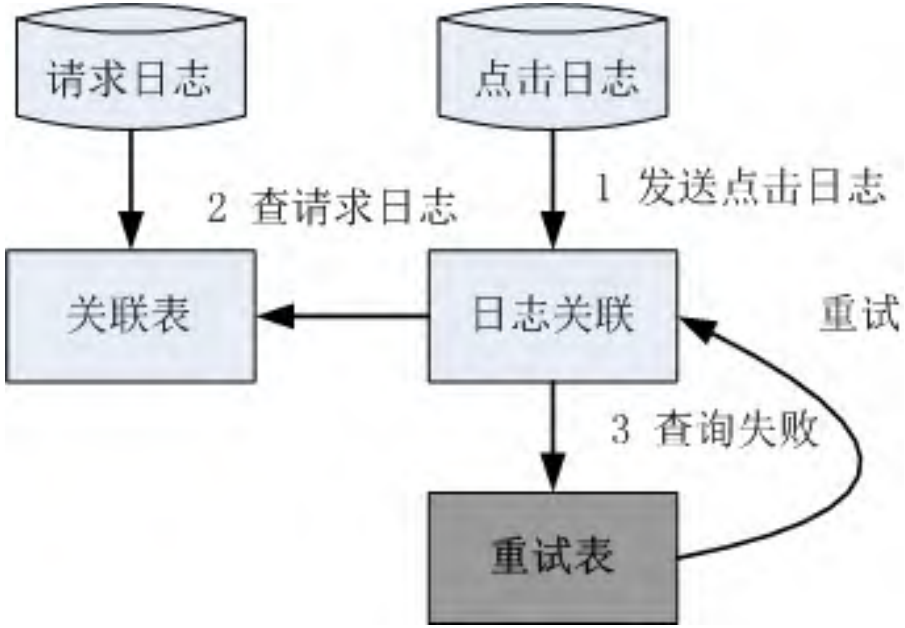
技术方案 – EXACT ONCE

- 后台离线对账。



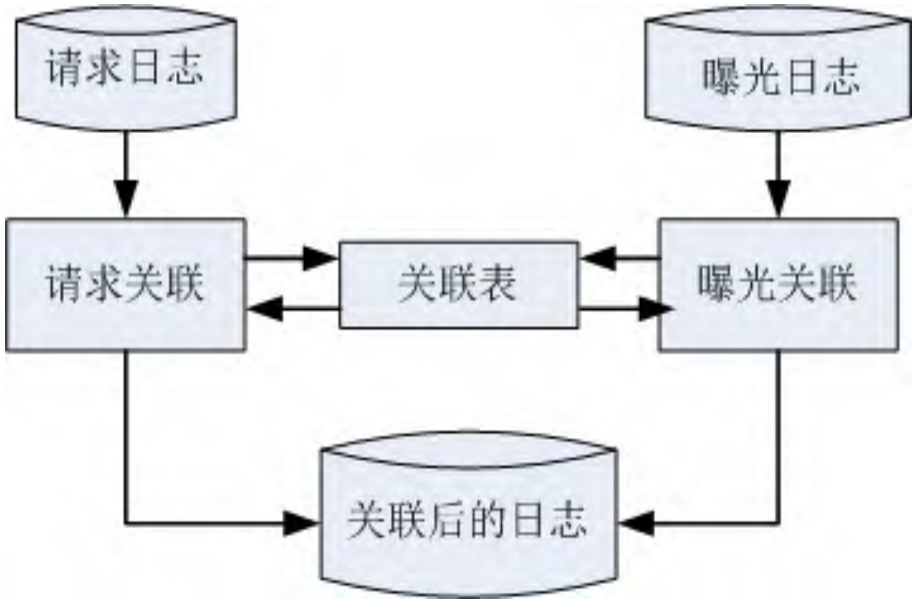
技术方案 – 处理数据乱序

- 由于跨地域传输，机器偶尔卡住等原因，有时曝光和点击比请求日志早到。
- 当请求日志延迟时，缓存点击日志到磁盘，然后离线重试。



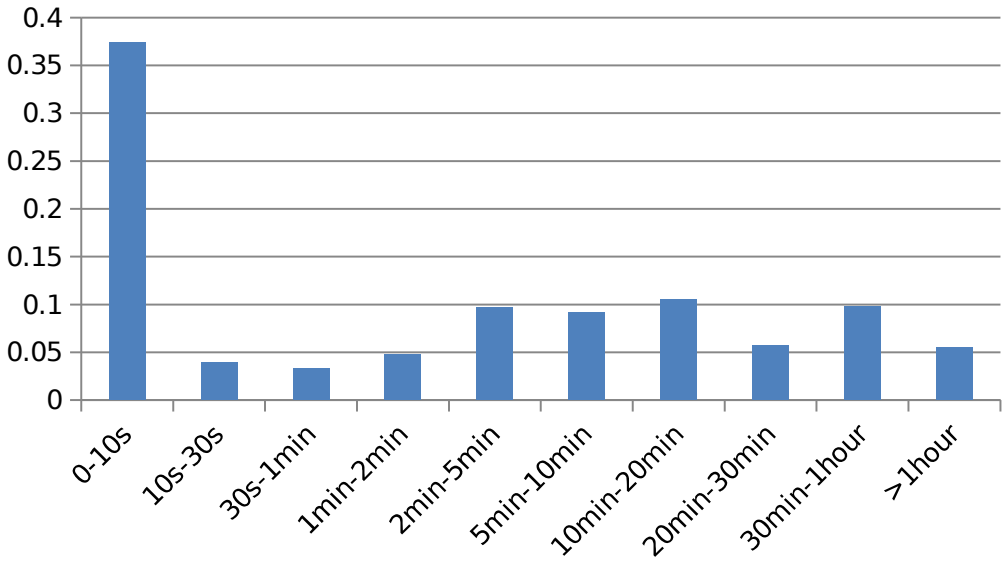
技术方案 –处理数据乱序

- 约 1/10 的请求晚于曝光日志
- 让请求、曝光日志晚到的触发关联操作
- 通过 checkandput 原子操作保证数据不重复



性能优化 – 系统特点

- 访问特性：
- 写 hbase： 每天 300 亿 +
- 读 hbase： 每天 200 亿 + ，而且都是随机读！
- 但是：
  - 只有 100 多亿读操作是预期读到数据的
  - 大部分数据写入到读取的时间延迟很小



性能优化 – 内存 cache

- 增大写 cache， cache 命中率 95%
- 依靠 Bloom filter 过滤掉大部分读磁盘需求
- 部分访问频繁的小表配置为全内存表

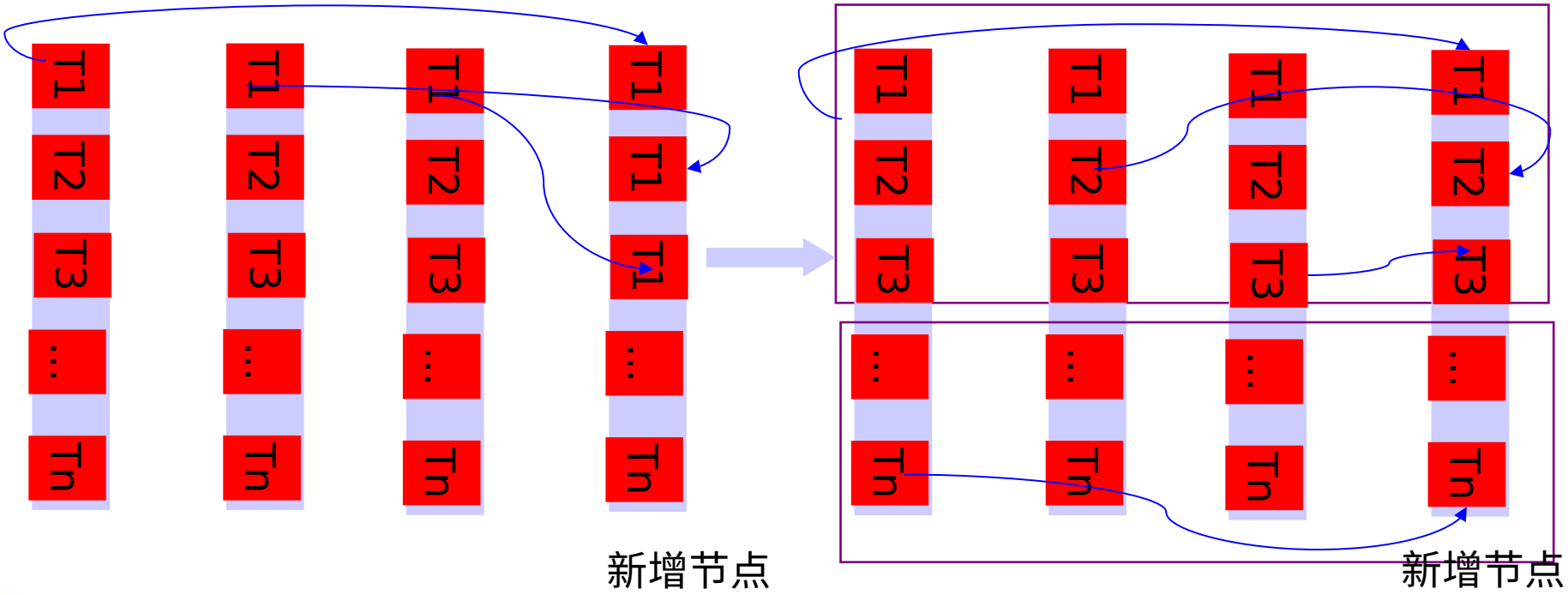
性能优化 – 优化负载均衡算法

- 问题描述

  对于时间序列的表，在新增节点或是节点故障重新加入时，热点 Region 会分布到一个 Regionserver 上。

- 优化点

  针对时间序列的 balancer 算法优化。按时间分段，随机选取



性能优化 – 避免单点故障导致堵塞

- 问题描述
  为了更好的性能，我们使用批量操作读写 hbase 。
  在同步的接口中，会启动与服务器个数相同的线程池，等待服务器的返回结果。
  随着服务器数目增加，客户端线程数目增加非常多。
  当有一个服务器处理缓慢时，会拖慢所有客户端的所有线程。
- 目前的解决办法：客户端把所有批处理操作按照 Region 归类分发。
- 更好的解决办法：采用异步的客户端。

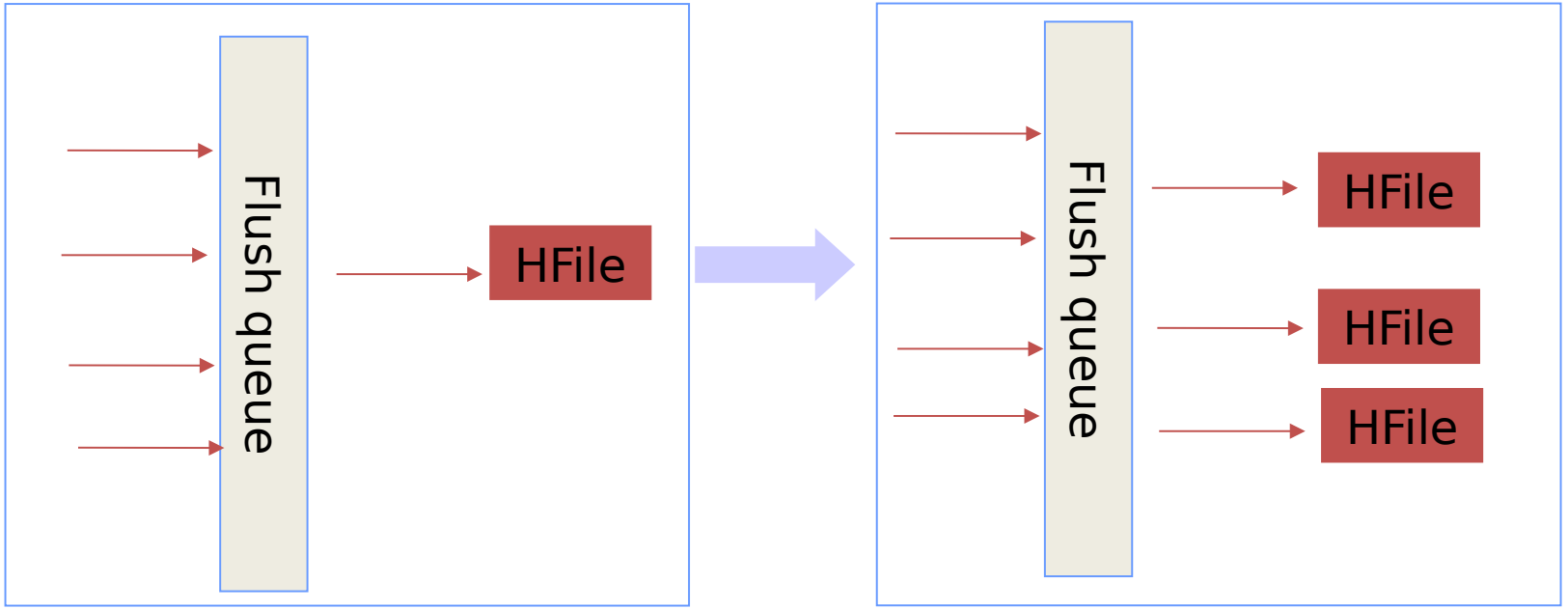
性能优化 – 避免 rpc 堵塞

- 问题描述

  在 put 请求过大的情况下，服务器的 flushsize 队列积压，导致 RPC 出现阻塞

- 优化点

  使用多线程进行 flush



性能优化 – 减少网络开销

- 问题描述

  网络带宽消耗过大

- 优化点

  调整 Memstore 的大小以及 Hlog 等参数减少 flush 小文件的个数，减少 compation 的需求

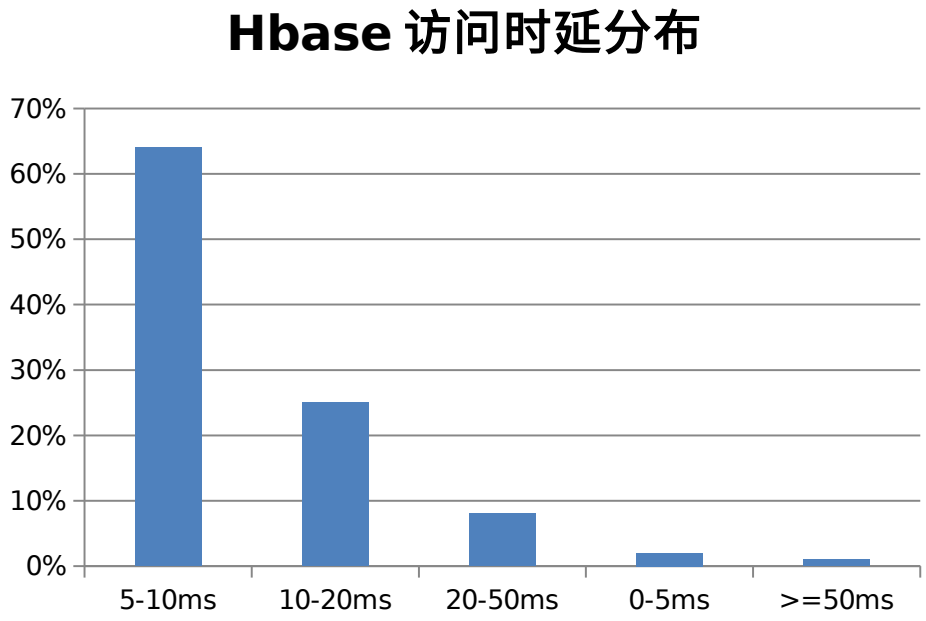
  关闭大表的 major compation

  调整 compation 的阈值参数和线程数目

  采用针对单个 region 的 compaction 进行操作，通过外部配置，尽量在流量低峰进行 compation 操作

应用情况

- 日志 200+ 亿 / 天
- Hbase 集群 200+ 台



Thanks!

李锐
腾讯广点通
Phone: 13810095271
Email: rli@tencent.com