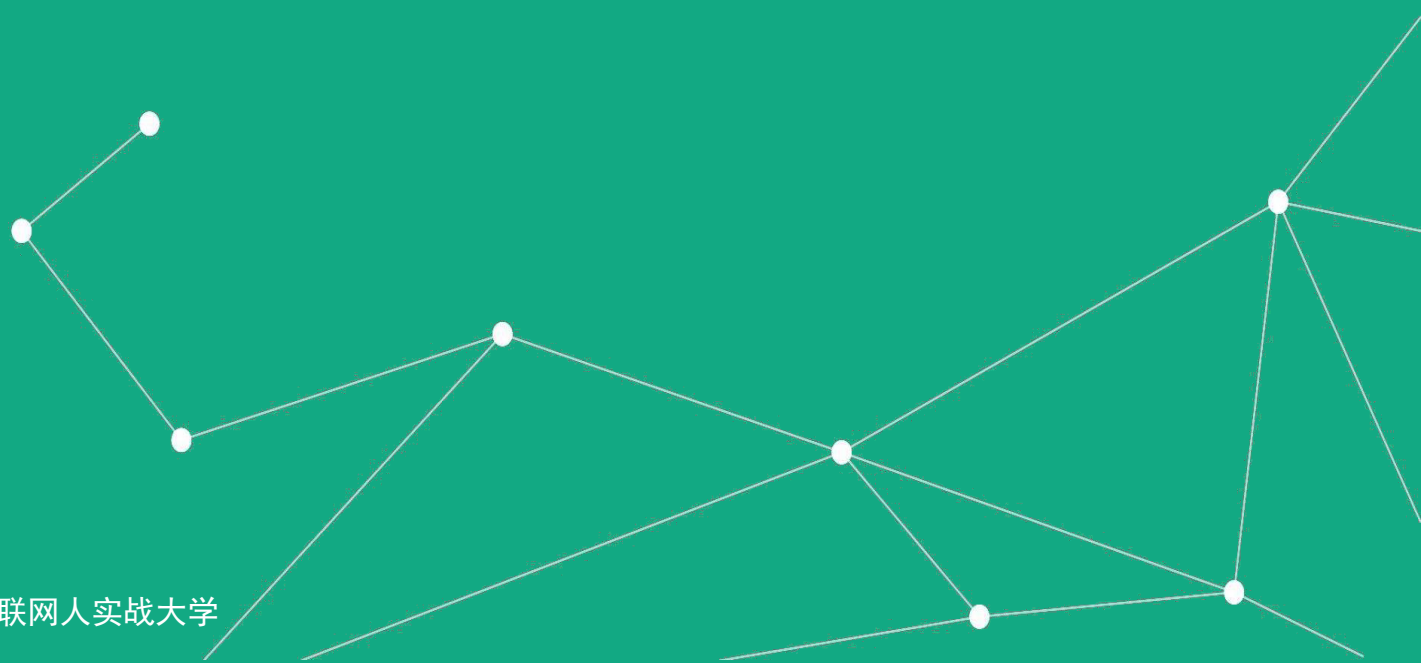
课时34

如何为海量计数场景设计缓存服务？

1. 计数常规方案
2. 海量计数场景
3. 海量计数服务架构
4. 海量计数服务收益

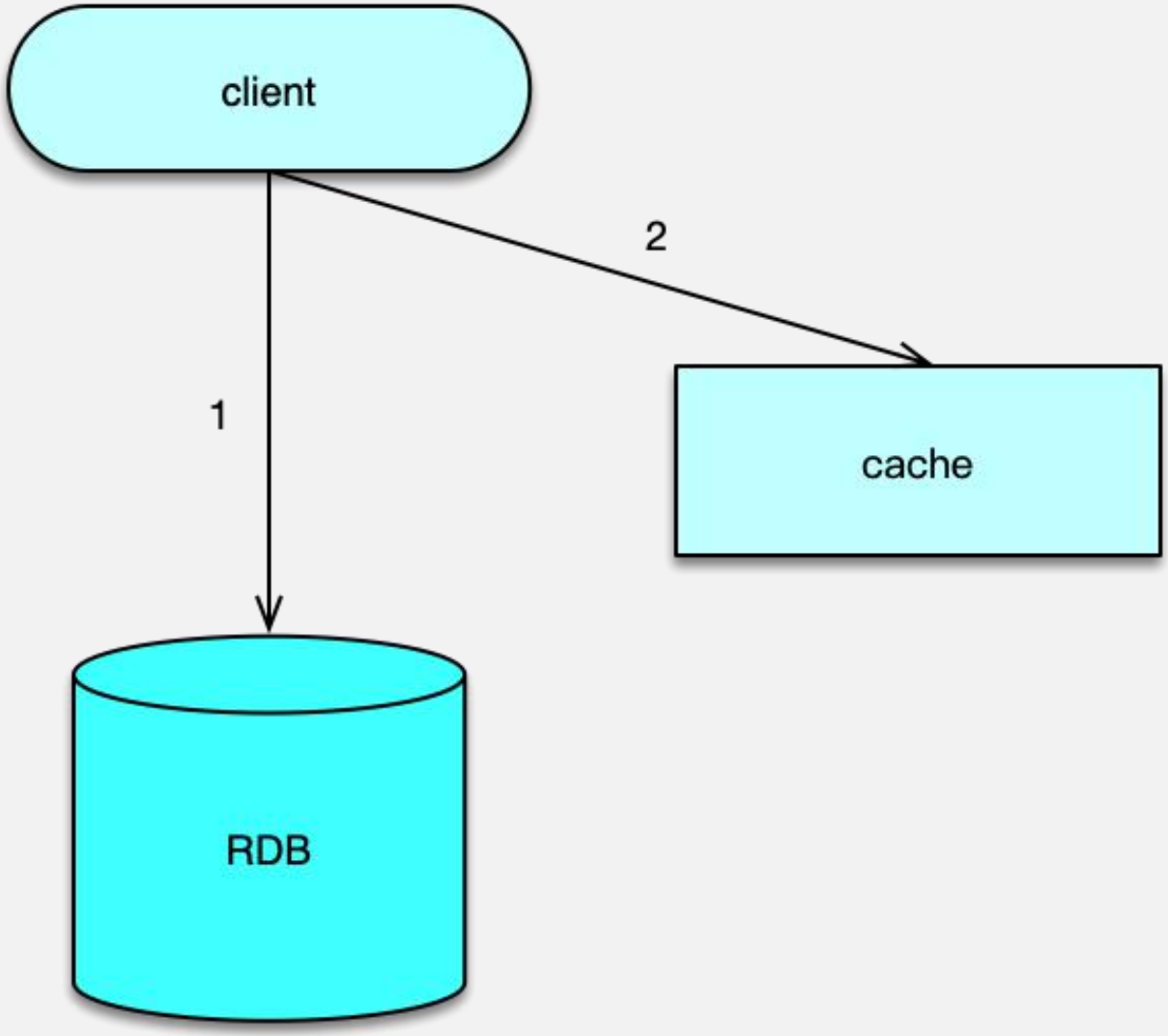


海量计数缓存解析

拉勾 LAGOU.COM

计数常规方案

- 常规计数服务架构
  - Value 存为long型，有效内存8字节
  - Cache + DB 存储方案
    - 先变更计数DB
    - 然后变更计数缓存 memcache/redis
    - DB 容易成为瓶颈
    - 适合中小规模计数服务

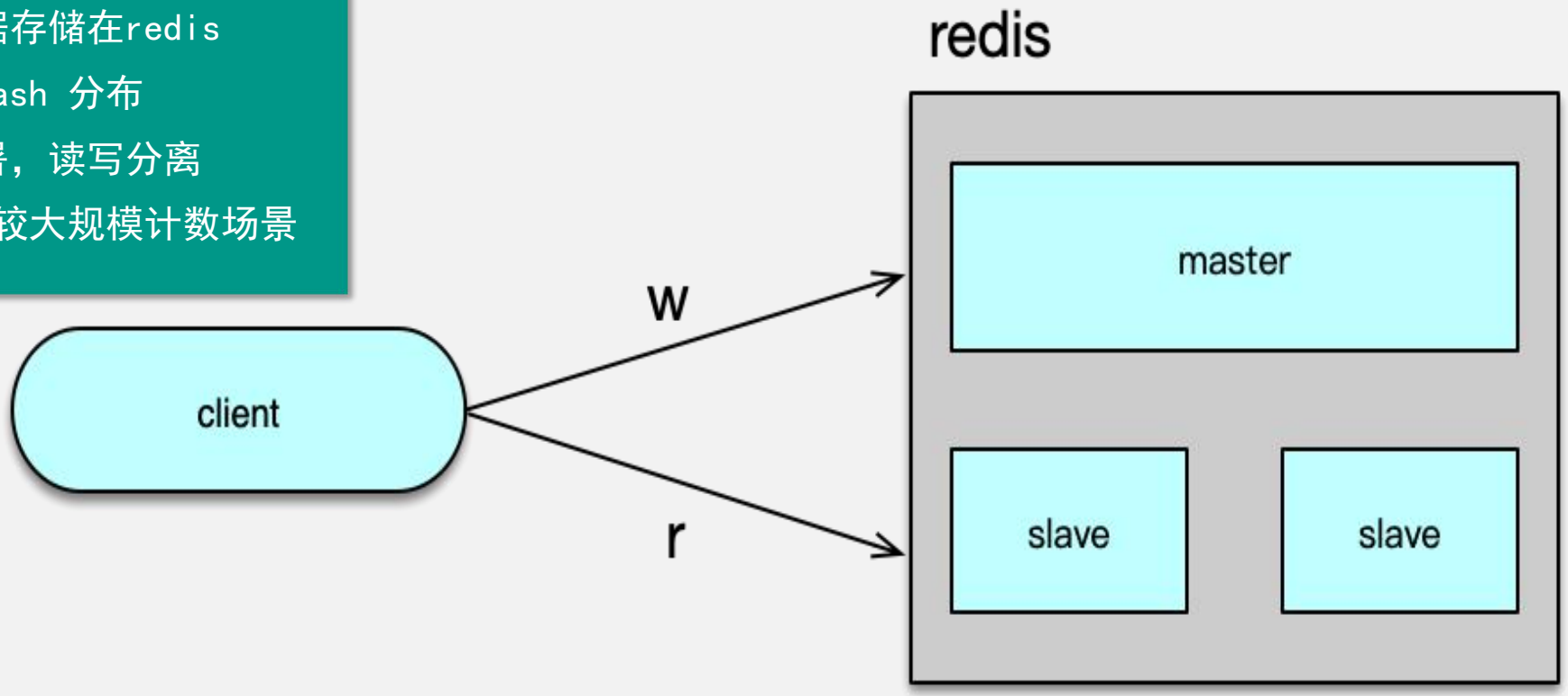


海量计数缓存解析

拉勾 LAGOU.COM

计数常规方案

- 常规计数服务架构
  - Redis 存储方案
    - 全部数据存储在redis
    - 按key hash 分布
    - 主从部署，读写分离
    - 适合中/较大规模计数场景

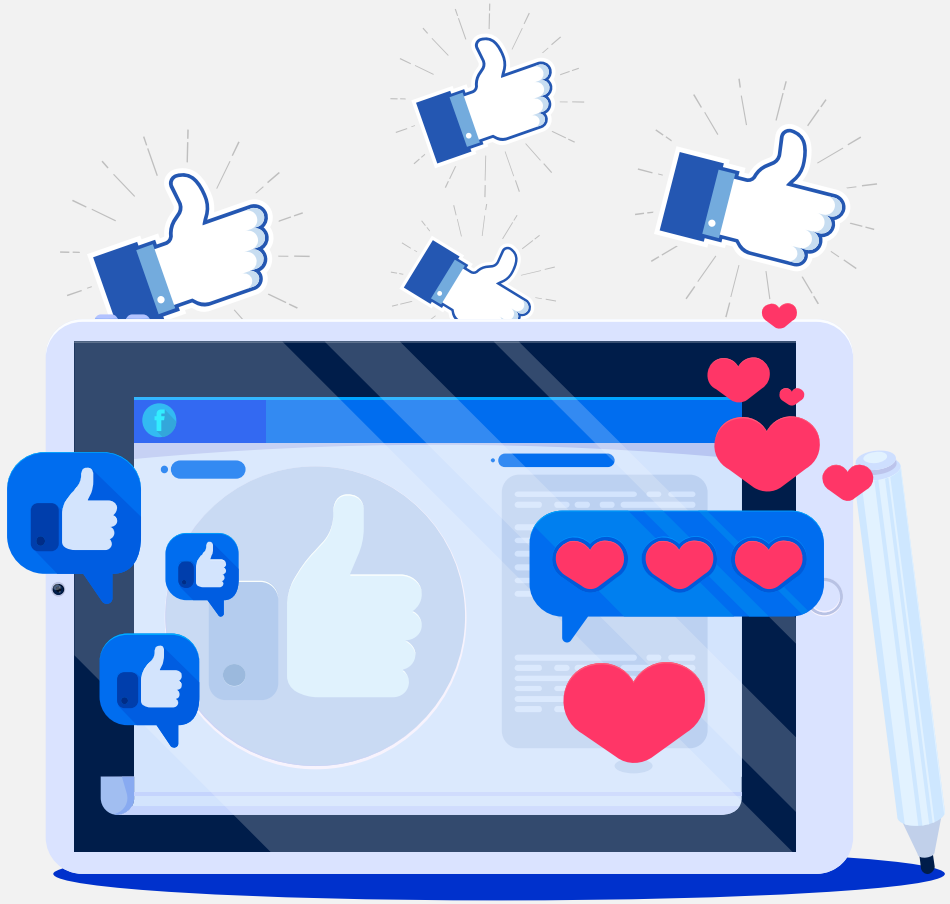


海量计数缓存解析

拉勾 LAGOU.COM

海量计数场景

- 海量计数业务特点（微博为例）
  - 海量计数对象
    - 用户维度，日活2亿+，月活近5亿
    - Feed 维度，总量千亿级，日增亿级
  - 每个对象多个计数
    - 用户维度，关注、粉丝、feed等
    - Feed维度：转发、评论、赞、阅读、表态等



海量计数缓存解析

拉勾 LAGOU.COM

海量计数场景

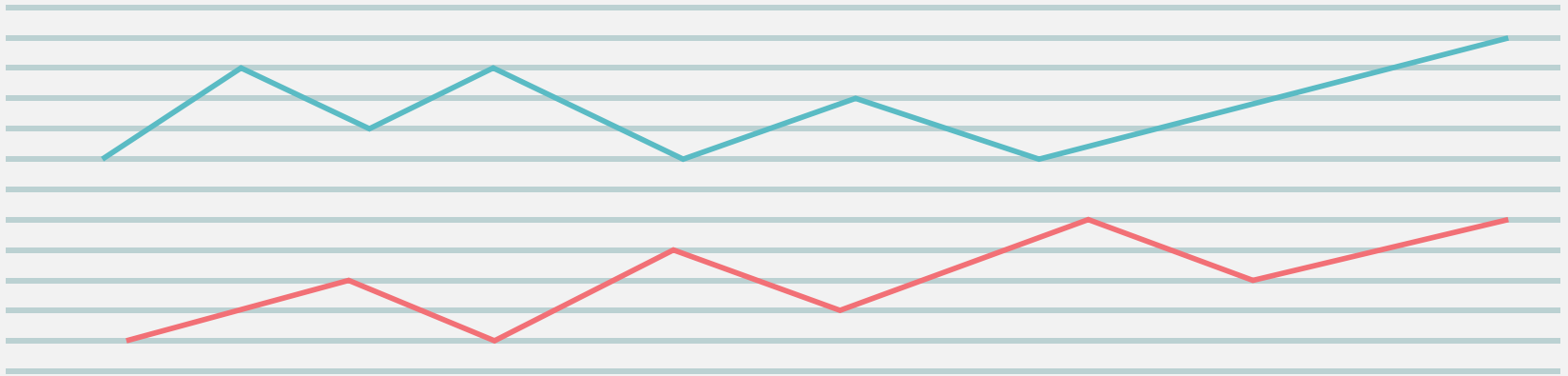
- 海量计数业务特点（微博为例）
  - 单次请求多对象，每个对象多个计数
    - 用户维度，关注、粉丝、feed等
    - Feed维度：转发、评论、赞、阅读等
  - 总请求量大，百万级QPS

海量计数缓存解析

拉勾 LAGOU.COM

海量计数场景

- 常规计数方案在海量计数场景的问题
  - Memcache + DB
    - DB 更新存在瓶颈
    - 单次请求key数多，一旦穿透DB，耗时长
  - Redis 存储方案
    - Redis 全内存存储，历史数据百T级，日增近百G，成本过大

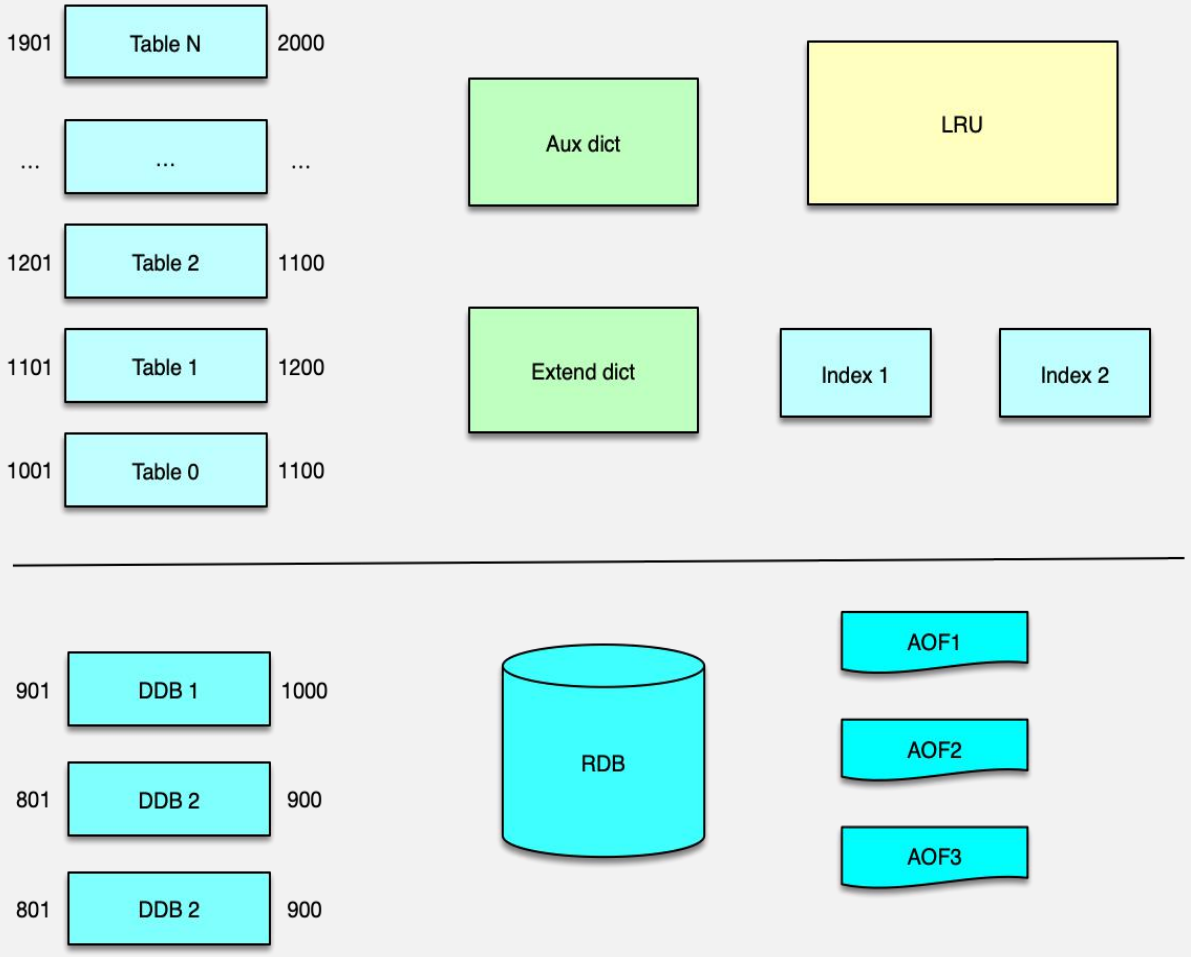


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务架构

- 基于Redis开发，兼容Redis协议
- 存储分为内存存储及磁盘存储两部分
- 内存预分配N块相同大小的Table空间，每个Table 1GB左右
- Table内key是微博id，value是自定义的多个计数

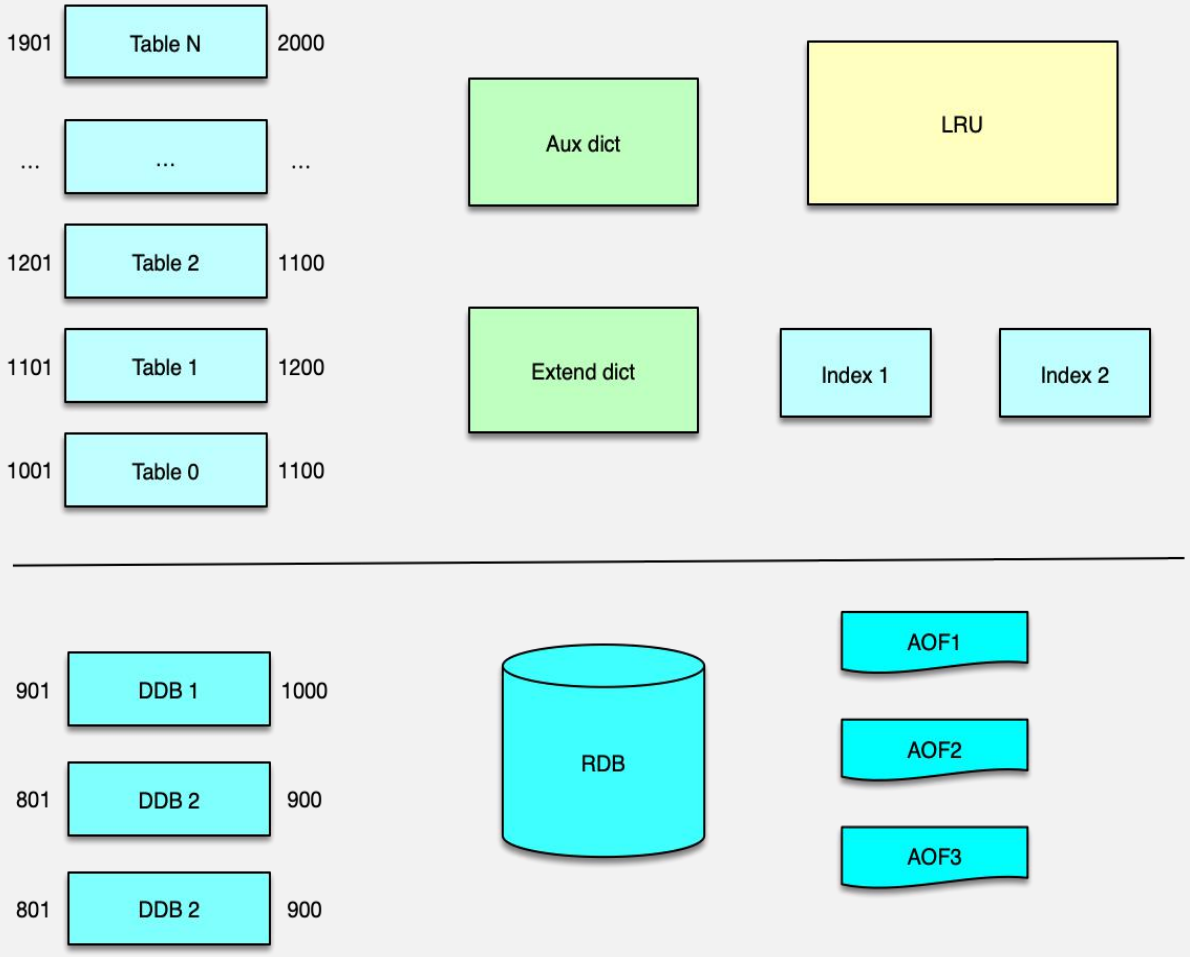


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务架构

- Id 递增，每个Table只存储一定范围内的id
- 当Table填充率超过阀值，滚动使用下一个Table当预分配Table使用完毕，内存分配新的Table
- 内存占用达到阀值，将内存中id范围最小的Table落盘到SSD磁盘，落盘Table文件称为DDB
- 落盘的DDB，会将文件索引记录在内存，方便查询

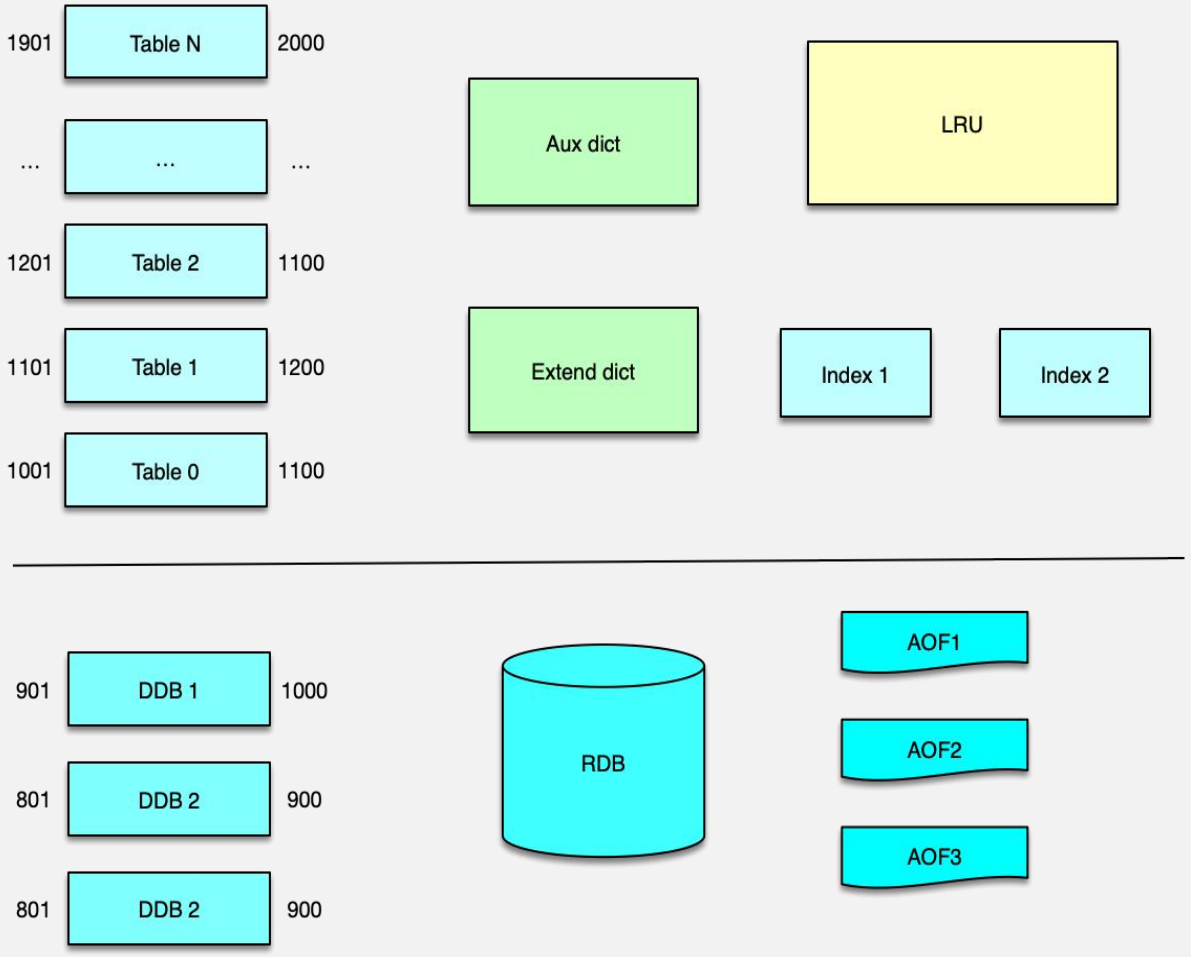


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务架构

- 通过 Schema 支持多列，每个key的value对应多个计数，每个计数的内存占用按schema分配，精确到bit
- 如果key有计数超过设置的阀值，则迁移到aux dict
- 每个Table负责一定范围的id，id范围超过容量当Table的填充率达到阀值，则新计数插入到新Table
- 旧Table 又插入大量数据，超过容量限制，则将新key插入到Extend dict

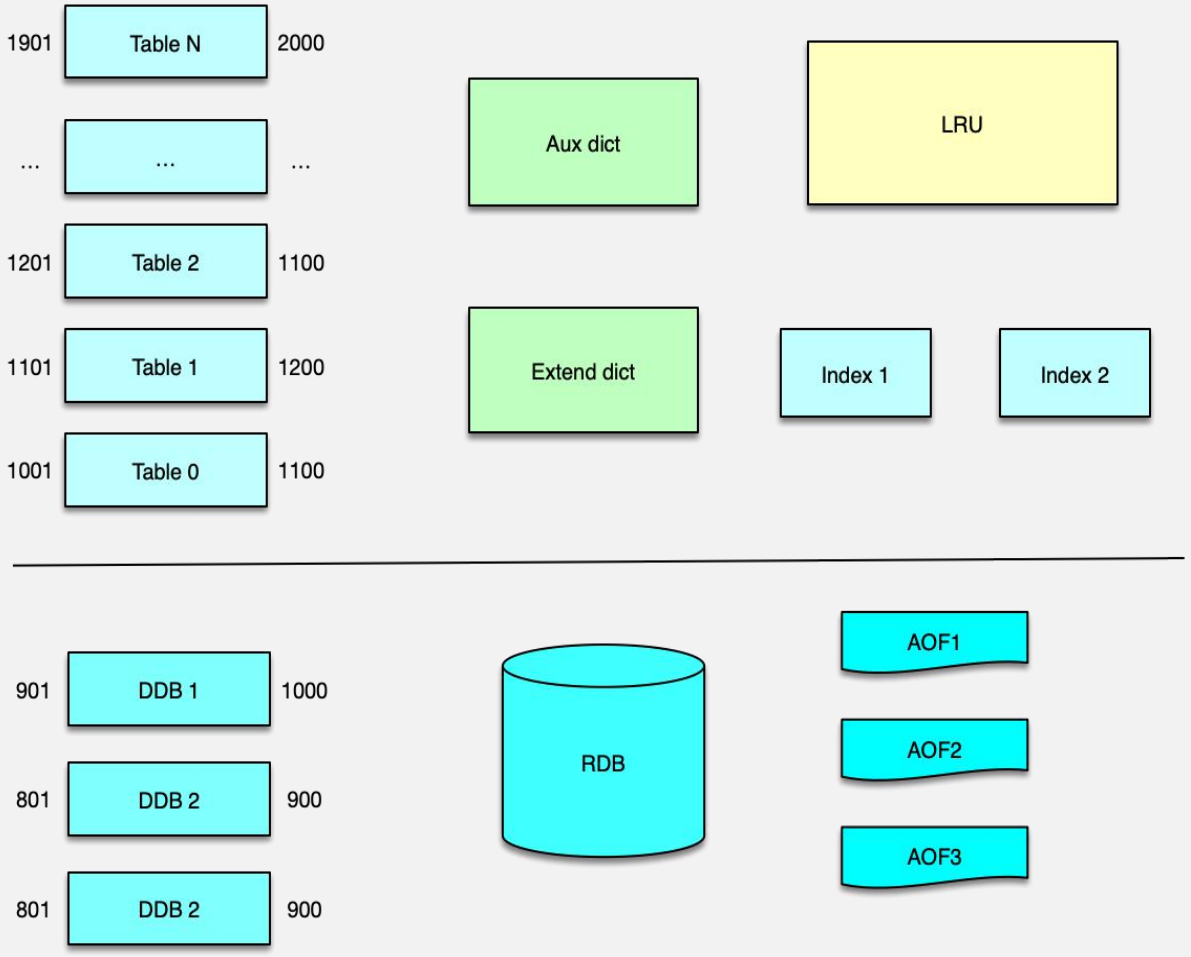


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务架构

- DDB 数据查询通过多线程异步并行执行，磁盘数据加载不影响业务正常访问
- DDB中冷数据重新加载后，存放到LRU，方便后续再次访问
- 数据快照采用RDB + 滚动 AOF策略，RDB记录构建时对应的AOF文件id 及 pos
- 全增量复制，slave断开重连时，上报同步AOF的id及位置，master 将对应文件位置之后的内容同步

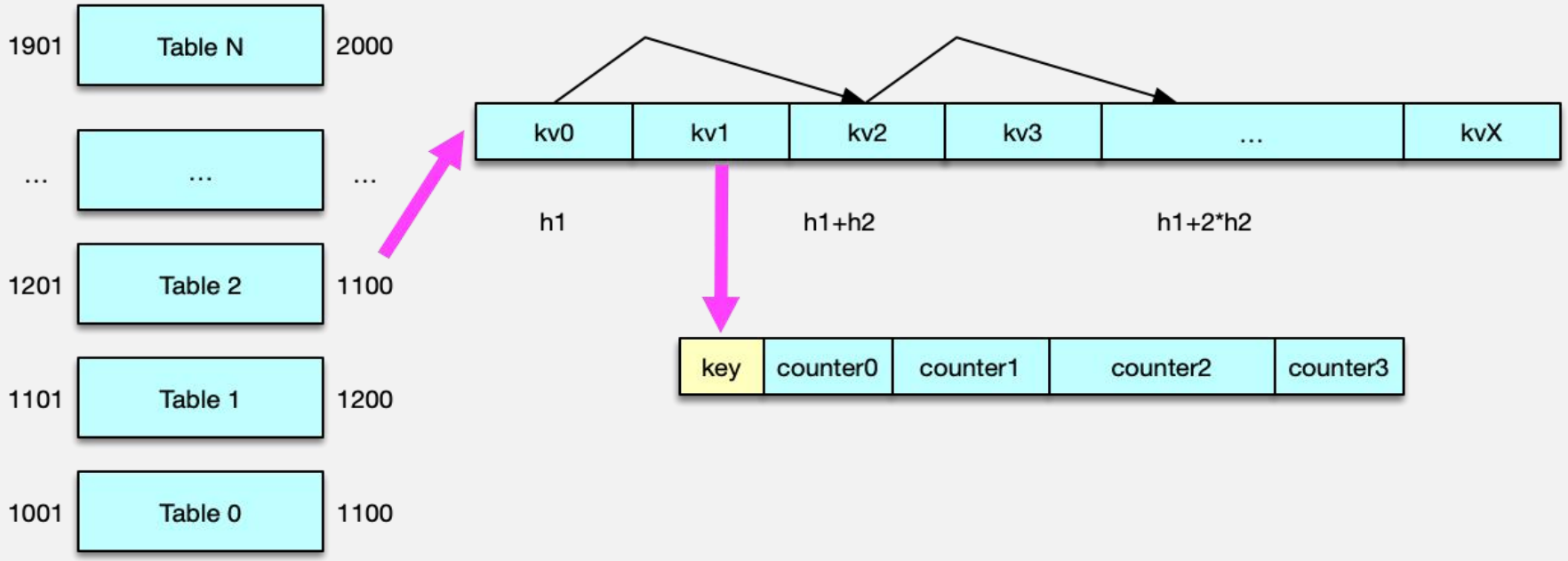


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务架构

- Table 是一个一维开放数组，每个kv占用内存相同
- 每个kv内部，key和多个计数紧凑型布局

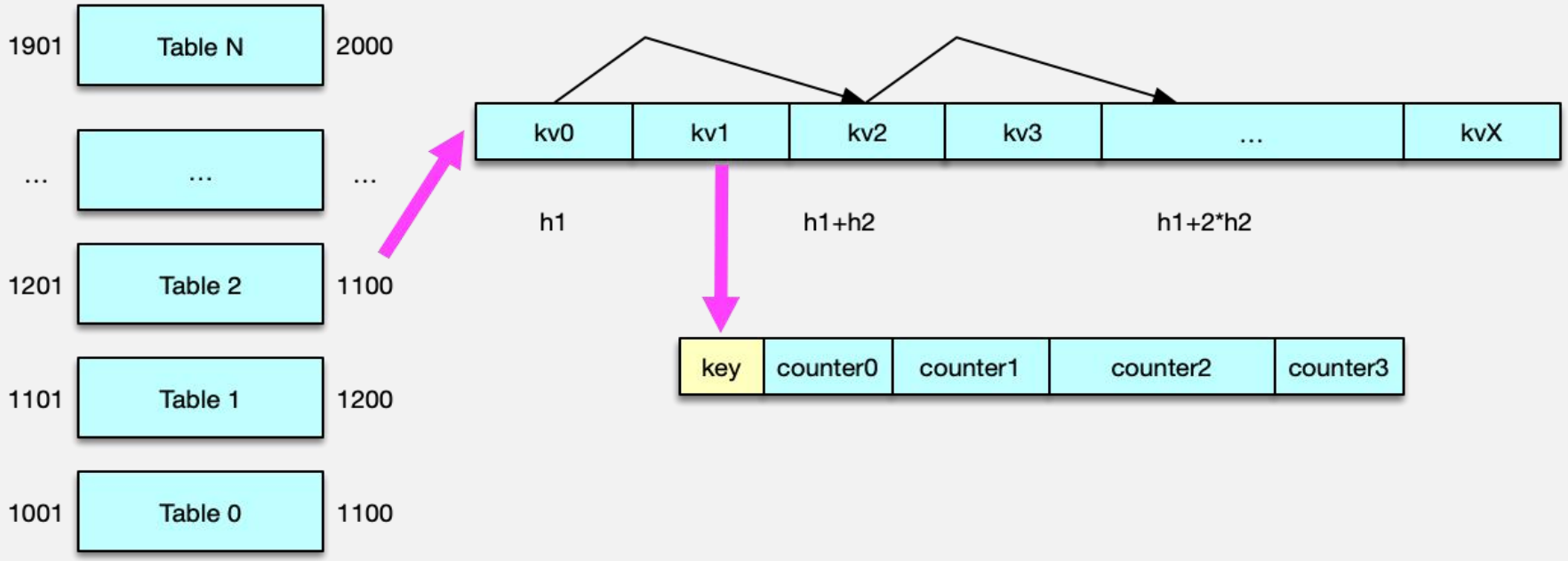


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务架构

- Key定位
  - 首先根据Table id范围确定key所在内存Table
  - 然后根据double-hash计算h1、h2，用h1+N*h2来查找位置，设置N最大值作为最大查询次数

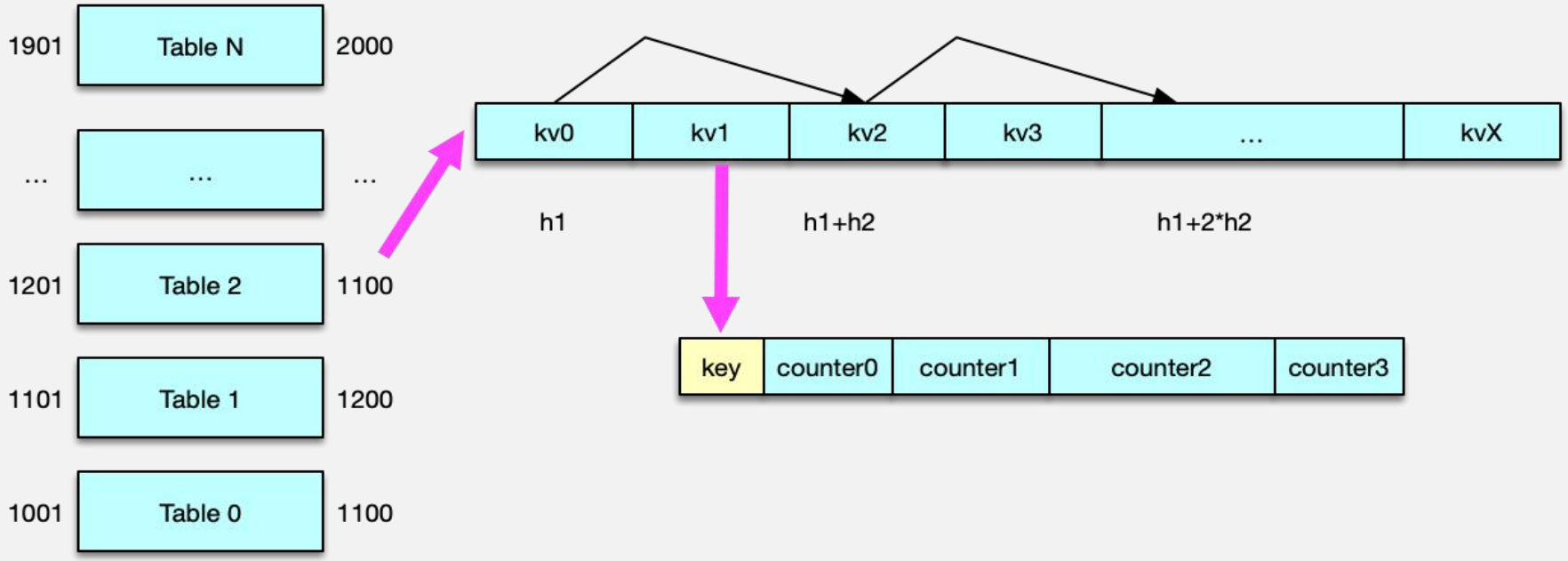


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务架构

- Key定位
  - 更新，查询位置为空直接插入，若key相同，则增减，否则继续向后查询
  - 查找，如果查询的位置为空，立即停止；若key不同，则N加1，继续查询

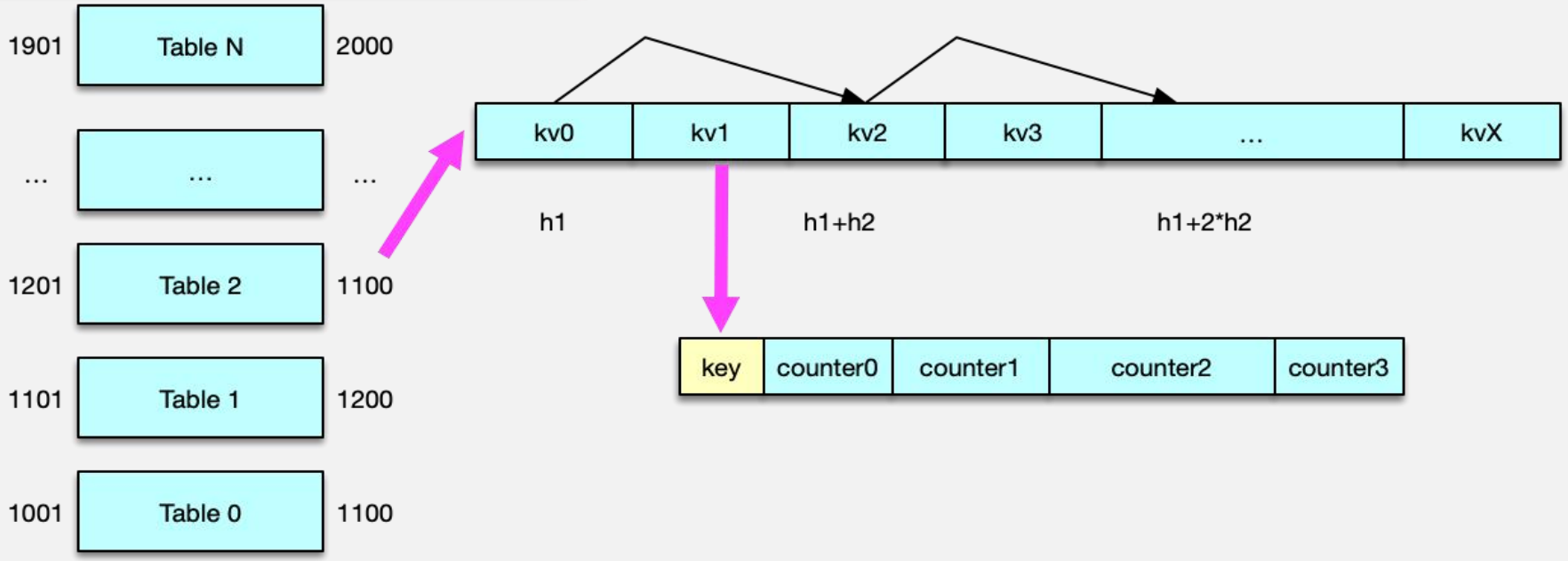


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务收益

- 内存占用只有Redis的10%以下
  - Key 直接存long型id
  - 多个计数合并为一个value，紧凑型存储
  - 每个计数设计适当大小size，超过阀值独立存储

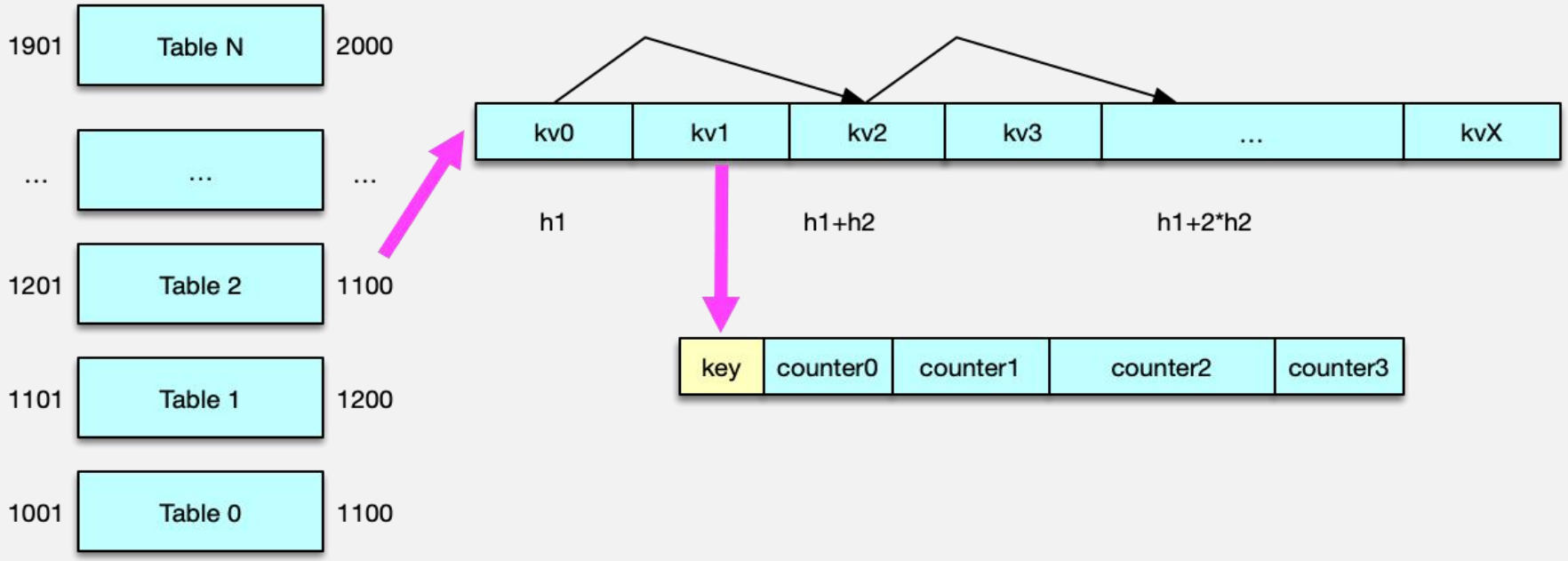


海量计数缓存解析

拉勾 LAGOU.COM

海量计数服务收益

- 查询性能提升3-5倍
  - 一个key存储多个计数，一次查询即可全部返回



Next：课时35《如何为社交feed场景设计缓存体系？》