新浪微博redis实践

刘东辉
@Alfejik

大纲

- 应用场景
- redis优化实践
- redis使用建议
- Q&A

应用场景-业务

- **关系**
- **计数**
- 存在性判断
- 通知提醒

应用场景-挑战

read万亿级

write千亿级

内存数十TB

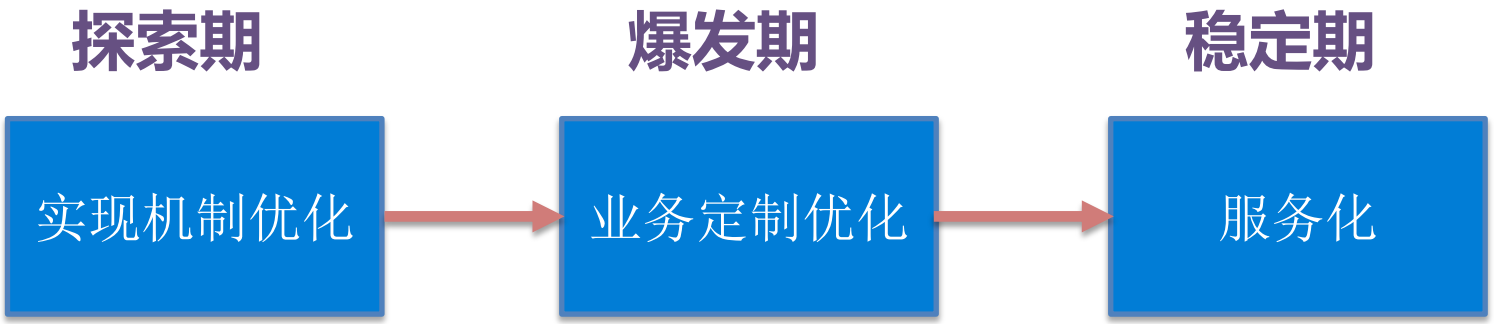
业务数百个 数十TB

VS

性能

成本

redis优化实践-阶段

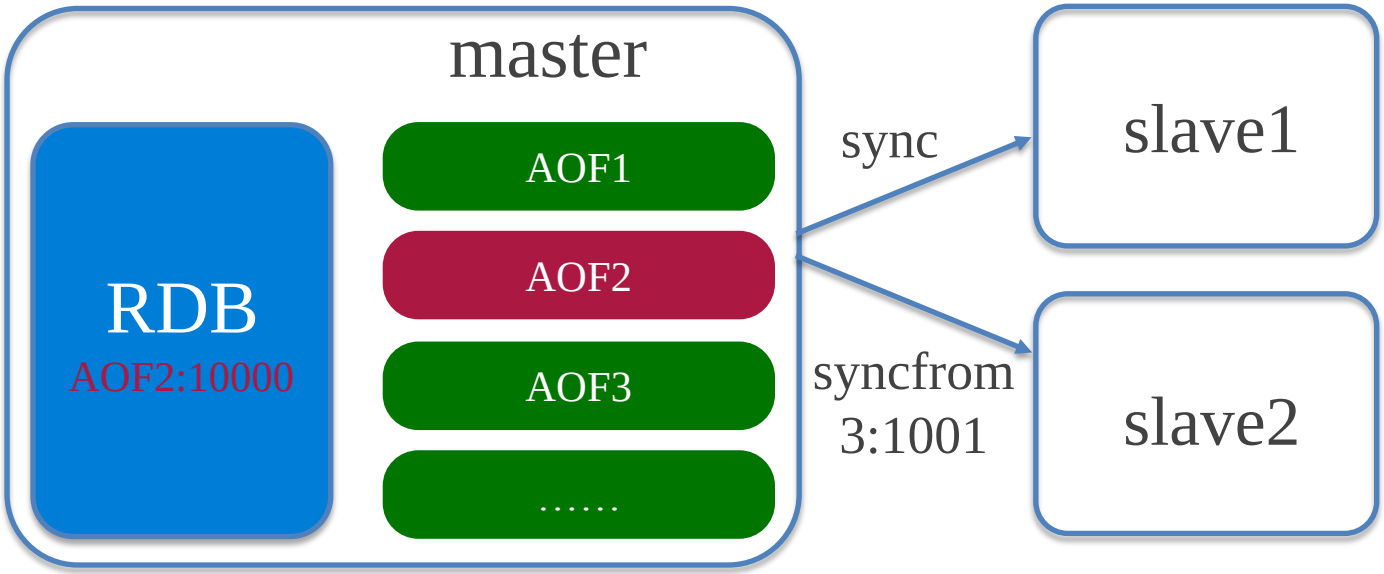


官方实现问题

- 性能问题
  - bgsave、bgrewrite引起redis服务阻塞
- 可用性问题
  - 持久化过程中内存耗尽，导致redis进程崩溃
  - 全量复制网络阻塞，从库reload数据服务不可用
- 运维复杂问题
  - 版本升级及管理复杂度高

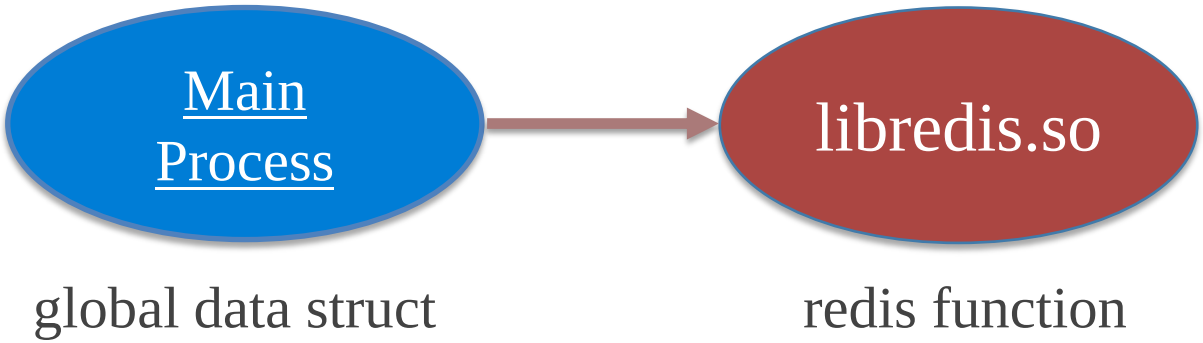
持久化&复制优化

- aof write：主线程—>bio
- 落地时间：不可控—>cronsave
- 落地方式：rdb，aof —> rdb + aof(rotate)
- 复制：全量复制—>独立线程 + 完全增量复制



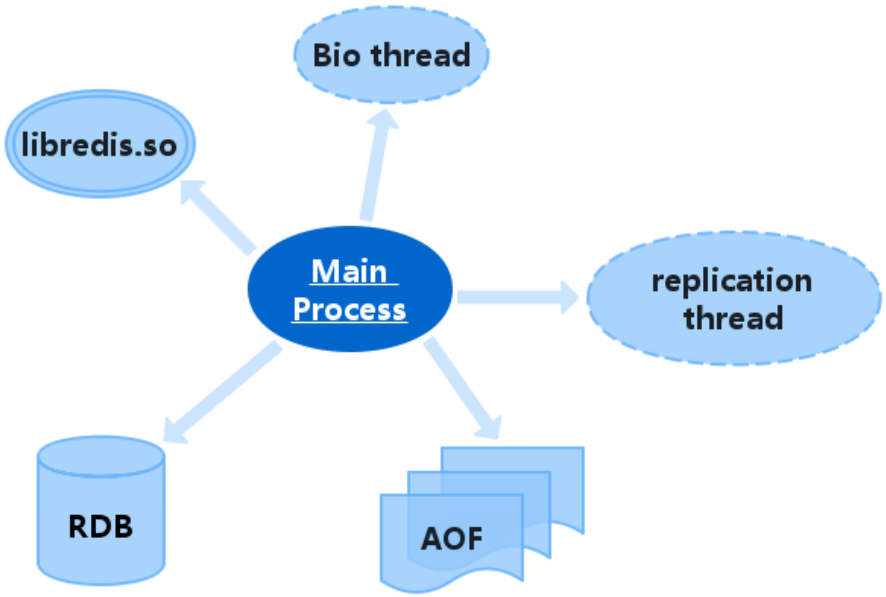
redis内部机制优化-实现机制优化

- **升级优化：重启 —> 动态升级**



小结

- 无阻塞落地
- 增量复制
- 在线热升级



特定业务场景问题

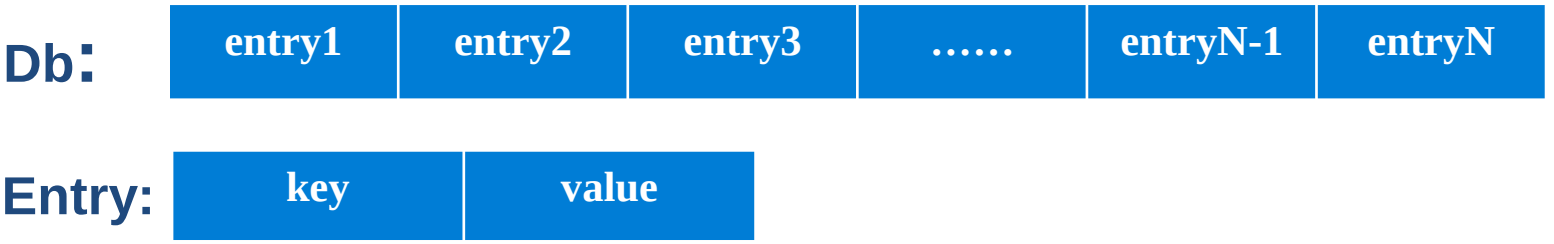
- 容量问题
  - 快速增长的关系业务成本压力
- 计数类需求
  - 多样的计数类场景需要专门的解决方案
- 判断类需求
  - 频繁的判断需求性能及成本压力

关系类业务

- storage —> cache
- 放弃原生hash结构，定制longset数据结构
- 固定长度开放寻址hash数组
- Miss后client批量O(1)回写

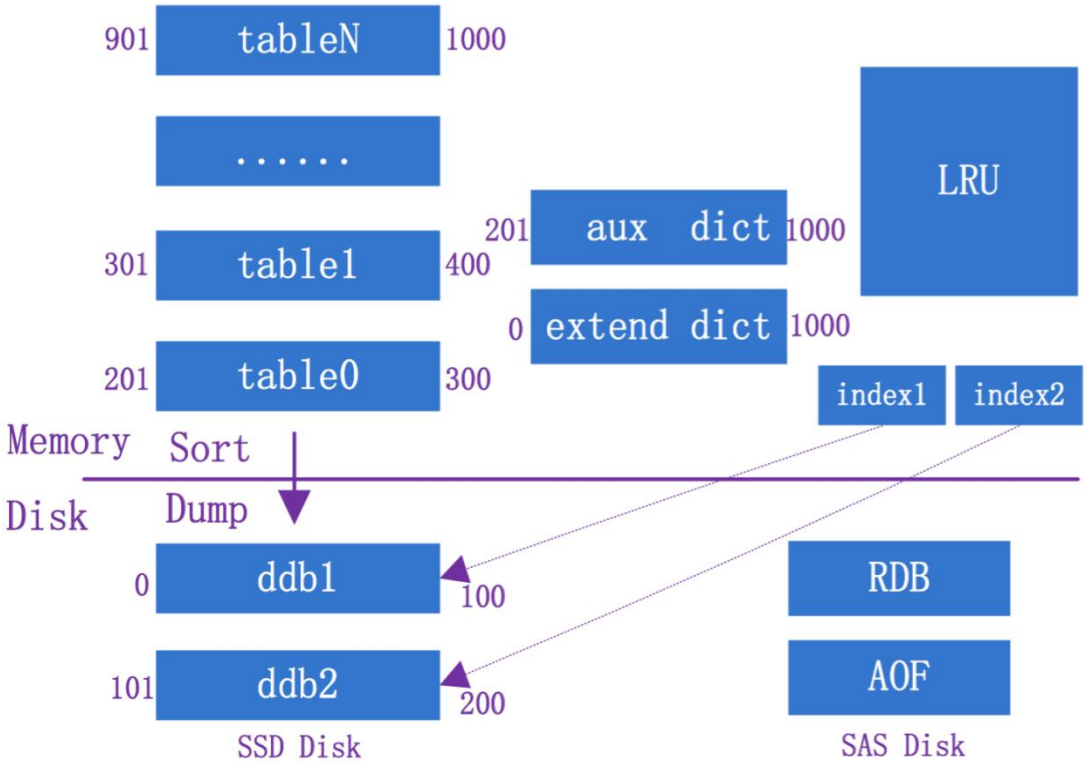
计数类业务

- kv定长存储
- double hash解决冲突



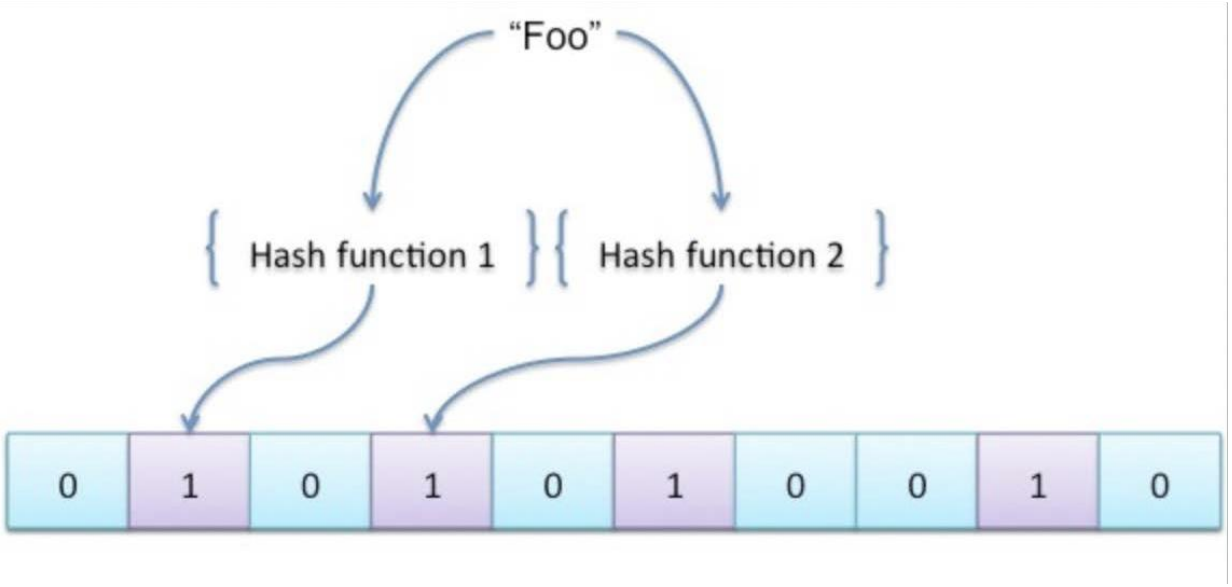
计数类业务

- schema支持多列
- 冷热数据分离
- lru提升性能
- 异步线程访问磁盘数据



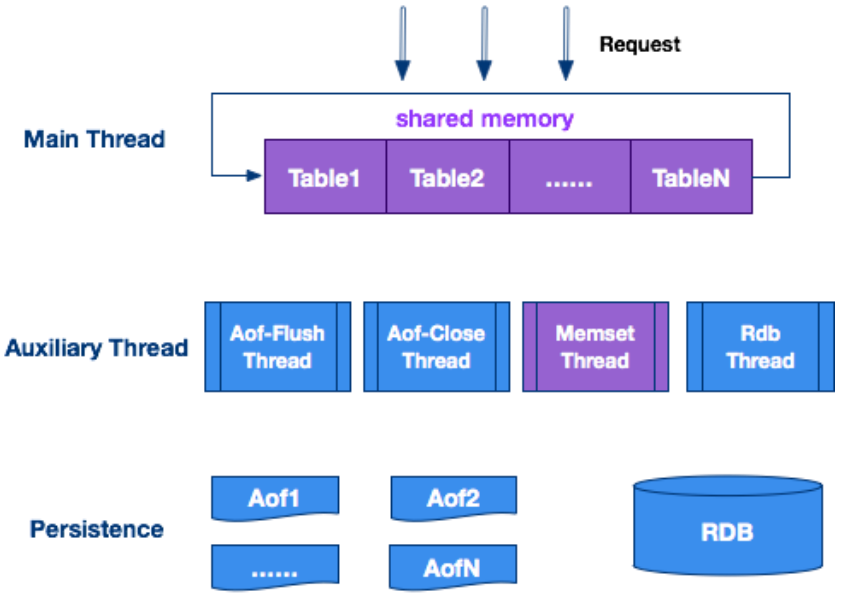
判断类业务

- table分段预分配，段内bloomfilter
- 每条kv 1.2byte(1%误判率)

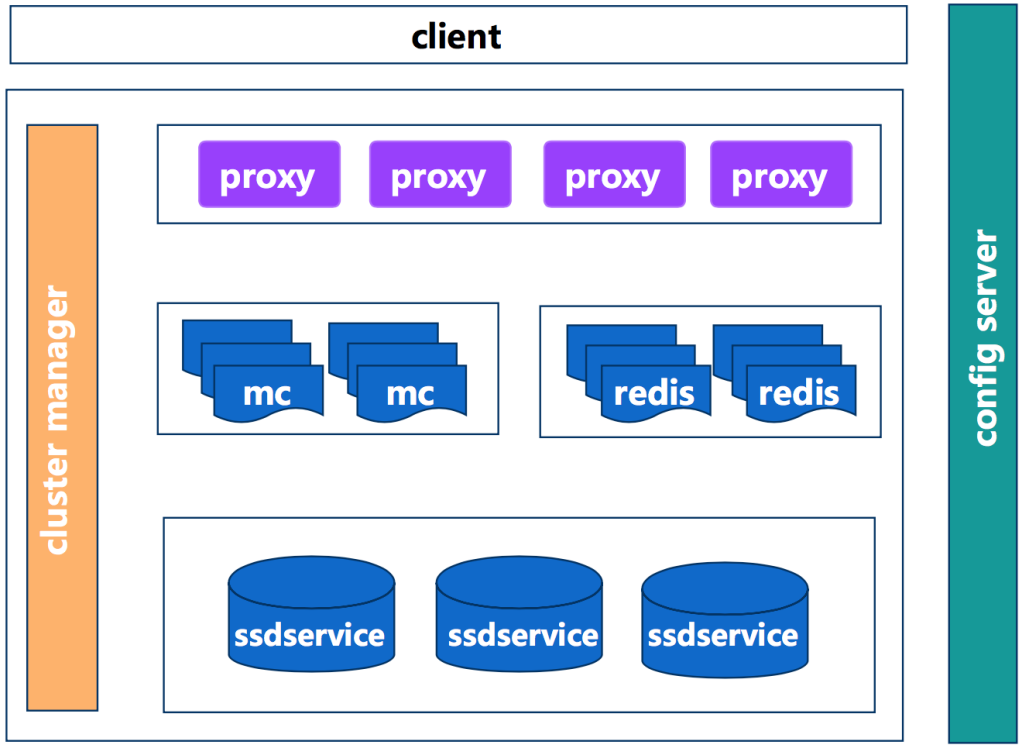


判断类业务

- redis协议
- Table分段预分配，段内bloomfilter
- 共享内存存储数据
- rdb+aof持久化



- cache服务化
  - 配置服务化
    - 统一配置管理
  - 访问proxy化
    - 屏蔽资源细节
    - 简化业务使用
  - 运维服务化
    - 统一管理cache生命周期
    - 融合HA机制
    - 业务SLA保证



redis使用用及优化建议

- redis不是万能的：合理的业务选型
- 明确redis业务使用规范
- 按照业务线独立部署：避免混用
- 线上版本尽量统一
- 拥抱需求，持续优化

Q&A

以微博之力 让世界更美！

weibo.com