

你知道的越多，你不知道的越多

点赞再看，养成习惯

本文 **GitHub** <https://github.com/JavaFamily> 上已经收录，有一线大厂面试点思维导图，也整理了很多我的文档，欢迎Star和完善，大家面试可以参照考点复习，希望我们一起有点东西。

絮叨

本来是没这期的，按道理更新也应该是在周一更新消息队列的幂等，分布式事务相关的文章，但是这篇暖男我实在忍不住了，不是发自己的文章，是帮课代表发一下，她本科是北京交通大学，也是电子科技大学的研究生。

她看了我的系列，做了个笔记📝，我一看，偶买噶！。

这是什么仙女啊，这是我第一次有这种感觉，这笔记有着前所未有的新鲜感，细节的勾勒，让整个笔记更显出奥妙....有些不太成熟的话语，跟我文章的骚气十分搭配，将Redis的性能衬托的更为出色，这才呈现出完美的课代表笔记。

这也是我第一次看到看个渣男的文章都做笔记的，这笔记让我有了初恋的味道，这我以后可得好好写了，不然辜负了课代表这样认真的妹子，到时候到杭州来找我：敖丙你个渣男乱写，害我没拿Offer！我要杀了你！

我也不多BB了不影响大家食用了，课代表说了以后我写的其他技术栈的笔记一样会贡献出来，代价就是要嫁给我，呸呸呸，代价就是我以后帮他介绍大厂朋友内推下，看看简历呀，解答下职场问题啊什么的。（根本就是举手之劳啊，我血赚？）

我一听我不能忍啊，我气得拍桌子，不行你以后不懂的知识点我包了，我也不懂的咳咳我看完书再包？

哈哈开玩笑的，总之课代表的精神大家包括我都应该好好学学，这种人活该她拿SSP的Offer。

Tip: SSP （Special Offer 优秀生源Offer渠道）



Redis

基础知识

- 异步队列
 - list还有个指令叫blpop，在没有消息的时候，它会阻塞住直到消息到来
 - pub/sub主标题订阅者模式，可以实现 1N 的消息队列，实现生产一次，消费多次
 - pub/sub主标题订阅者模式，消费者下线的情况下，生产的消息会丢失，得使用专业的消息队列如RocketMQ
- 延时队列
 - sortedset，拿时间戳作为score，消息内容作为key调用zadd来生产消息，消费者用zrangebyscore指令获取N秒之前的数据轮询进行处理。
- 持久化
 - RDB做镜像全量持久化，AOF做增量持久化。因为RDB会耗费较长时间，不实时时，在停机的时候会导致大量丢失数据，所以需要AOF来配合使用。在redis实例重启后，会使用RDB持久化文件重新构建内存，再使用AOF重放近期的操作指令来实现完整恢复重启之前的状态。
 - Redis本身的机制是 AOF持久化开启且存在AOF文件时，优先加载AOF文件；AOF关闭或者AOF文件不存在时，加载RDB文件；加载AOF/RDB文件成功后，Redis启动成功；AOF/RDB文件存在错误时，Redis启动失败并打印错误信息。
- 机器断电对数据丢失的影响
 - AOF日志sync属性的配置，如果不要求性能，在每条写指令时都sync一下磁盘，就不会丢失数据。但是在高性能的要求下每次都sync是不现实的，一般都用使用定时sync，比如1s1次，这个时候最多就会丢失1s的数据。
- RDB原理
 - Fork是指redis通过创建子进程来进行RDB操作，cow指的是copy on write，子进程创建后，父子进程共享数据段，父进程继续提供读写服务，写脏的页面数据会逐渐和子进程分离开来。
- Pipeline 好处
 - 可以将多次IO往返的时间缩减为一次，前提是pipeline执行的指令之间没有因果相关性。
 - 才主题
- 集群的同步机制
 - Redis可以使用主从同步，从主同步
 - 第一次同步时，主节点做一次bgsave，并将后续修改操作记录到内存buffer，待完成后将RDB文件全量同步到复制节点
 - 复制节点接受完成后将RDB镜像加载到内存，加载完成后，再通知主节点
 - 后续的增量数据通过AOF日志同步即可，有点类似数据库的binlog
- 集群的高可用
 - Redis Sentinel 着眼于高可用，在master宕机时会自动将slave提升为master，继续提供服务。
 - Redis Cluster 着眼于扩展性，在单个redis内存不足时，使用Cluster进行分片存储。

缓存雪崩、击穿、穿透

- 雪崩
 - 大面积的缓存失效，打崩了DB
 - 同一时间大面积失效，那一瞬间Redis跟没有一样，那这个数量级别的请求直接打到数据库几乎是灾难性的，你想想如果打挂的是一个用户服务的库，那其他依赖他的库所有的接口几乎都会报错，如果没做熔断等策略基本上就是瞬间挂一片的节奏
 - 解决办法：1、批量往Redis存数据的时候，把每个Key的失效时间都加个随机值就好了，这样可以保证数据不会在同一时间大面积失效
 - 2、Redis是集群部署，将热点数据均匀分布在不同的Redis库中也能避免全部失效的问题
 - 3、设置热点数据永不过期，有更新操作就更新缓存就好了
- 穿透
 - 用户不断的发起缓存和数据库中均不存在的数据的请求，导致数据库压力过大，严重会击垮数据库
 - 解决办法：1、增加参数校验
 - 2、从网关层Nginx增加配置项，对单个IP每秒访问次数超出阈值的IP都拉黑。
 - 3、布隆过滤器（Bloom Filter）这个也能很好的防止缓存穿透的发生，他的原理也很简单就是利用高效的数据结构和算法快速判断出你这个Key是否在数据库中不存在，不存在你return就好了，存在你就去查了DB刷新KV再return。
- 击穿
 - 一个Key非热点数据，在不停的扛着大并发，大并发集中对这一个点进行访问，当这个Key在失效的瞬间，持续的大并发就穿破缓存，直接请求数据库，就像在一个完好无损的桶上凿开了一个洞
 - 解决办法：1、设置热点数据永不过期
 - 2、增加互斥锁

持久化

- RDB
 - 冷备
 - RDB 持久化机制，是对 Redis 中的数据进行周期性的持久化。
 - 优点：RDB对Redis的性能影响非常小，是因为在同步数据的时候他只是fork了一个子进程去做持久化的，而且他在数据恢复的时候速度比AOF来的快。
 - 缺点：RDB都是快照文件，都是默认五分钟甚至更久的时间才会生成一次，这意味着你这次同步到下次同步这中间五分钟的数据都很可能全部丢失，AOF则最多丢一秒的数据；RDB在生成数据快照的时候，如果文件很大，客户端可能会暂停几毫秒甚至几秒
- AOF
 - 热备
 - AOF 机制对每条写入命令作为日志，以 append-only 的模式写入一个日志文件中
 - 优点：AOF是一秒一次去通过一个后台的线程sync操作，那最多丢这一秒的数据。AOF在对日志文件进行操作的时候是以append-only的方式去写的，他只是追加的方式写数据，自然就少了很多磁盘寻址的开销
 - 缺点：一样的数据，AOF文件比RDB还要大

哨兵

- 哨兵+主从 实现redis 集群高可用
- 集群监控：负责监控 Redis master 和 slave 进程是否正常工作。
- 消息通知：如果某个 Redis 实例有故障，那么哨兵负责发送消息作为报警通知给管理员。
- 故障转移：如果 master node 挂掉了，会自动转移到 slave node 上。
- 配置中心：如果故障转移发生了，通知 client 客户端新的 master 地址。

主从同步

- 启动一台slave 的时候，他会发送一个psync命令给master，如果是这个slave第一次连接到master，他会触发一个全量复制。master就会启动一个线程，生成RDB快照，还会把新的写请求都缓存存在内存中，RDB文件生成后，master会将这个RDB发送给slave的，slave拿到之后做的第一件事就是写进本地的磁盘，然后加载进内存，然后master会把内存里面缓存的那些新命名都发给slave。

内存淘汰机制

- 过期策略
 - 定期删除 默认100ms就随机抽一些设置了过期时间的key，去检查是否过期，过期了就删了。
 - 惰性删除 我不主动删，我删，我等你来查询了我看看你过期没，过期就删了还不给你返回，没过期就怎么样就怎么样。
- 定期没有删除、我也没有去查询、怎么办？淘汰机制

分布式锁

- 以基于 Zookeeper 实现分布式锁。每个系统通过 Zookeeper 获取分布式锁，确保同一时间，只能有一个系统实例在操作某个 Key
- 以上用来解决因多个系统同时操作（并发）Redis带来的数据问题

最经典的KV、DB读写模式

- 缓存+数据库读写的模式
 - 读的时候，先读缓存，缓存没有的话，就读数据库，然后取出数据后放入缓存，同时返回响应
 - 更新的时候，先更新数据库，然后再删除缓存

与memcache区别

- Redis 相比 Memcached 来说，拥有更多的数据结构，能支持更丰富的数据操作
- 在 redis3.x 版本中，便能支持 Cluster 模式，而 Memcached 没有原生的集群模式，需要依靠客户端来实现往集群中分片写入数据
- Redis 只使用单核，而 Memcached 可以使用多核，所以平均每一个核上 Redis 在存储小数据时比 Memcached 性能更高

学习思路

- 事前
 - Redis 高可用，主从+哨兵，Redis cluster，避免全盘崩溃。
- 事中
 - 本地 ehcache 缓存 + Hystrix 限流+降级，避免MySQL被打死。
- 事后
 - Redis 持久化 RDB+AOF，一旦重启，自动从磁盘上加载数据，快速恢复缓存数据。

能总结得这么全面连我都忍不住点赞了！

总结

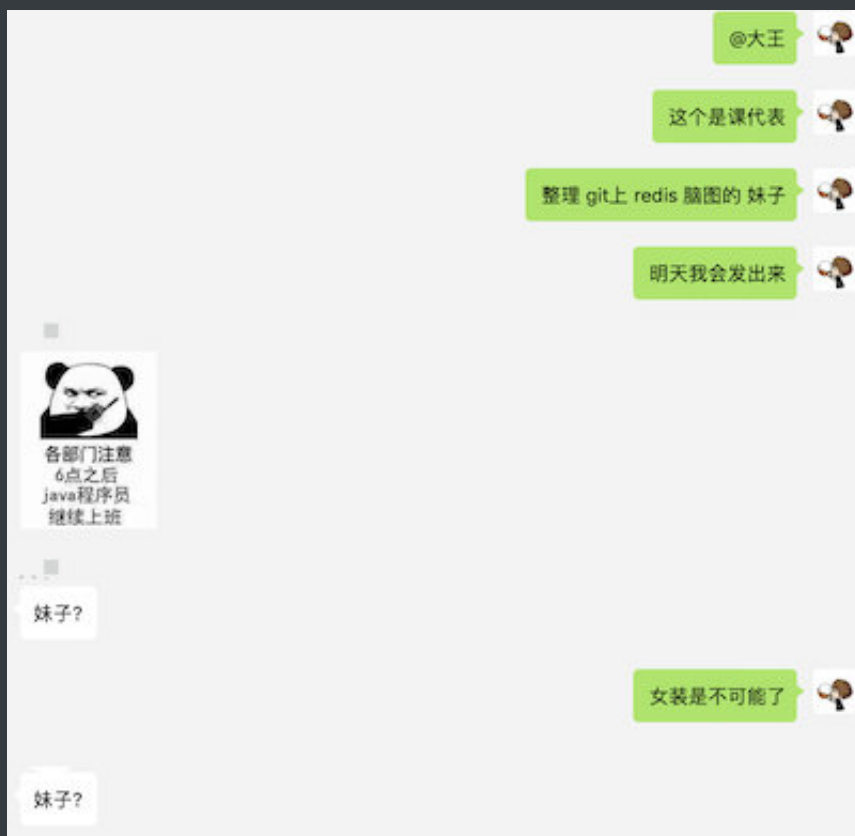
里面很多细节的点还是需要完善的，课代表最近上课很忙帅丙我呢除了周末也没时间，不过会不断完善到Git的，大家也可以去公众号回复「课代表」获取思维导图原稿。

其实我真的很欣赏课代表这样的精神的，她这样的举动触动了~~我~~，想想自己大学时候的样子，我忍不住给了自己两嘴巴子，我但凡有课代表一半的努力都不至于沦落到今天这样，等我冷静下来，走到了窗边，眺望头上若影若现的月亮，我的眼角又湿了！

花絮

人才群里的人才真的都是人才，一周两更高产似母猪了我都，还天天催更不过我也认了，课代表进去差点把人家吓走，这么好的课代表吓走了我哪里找第二个？

GitHub上有我联系方式和入群方式 <https://github.com/JavaFamily>





点关注，不迷路

好了各位，以上就是这篇文章的全部内容了，能看到这里的人呀，都是人才。

我后面会每周都更新几篇一线互联网大厂面试和常用技术栈相关的文章，非常感谢人才们能看到这里，如果这个文章写得还不错，觉得「敖丙」我有点东西的话 求点赞👍 求关注❤️ 求分享👥 对暖男我来说真的 非常有用!!!

创作不易，各位的支持和认可，就是我创作的最大动力，我们下篇文章见！

敖丙 | 文 【原创】

如果本篇博客有任何错误，请批评指教，不胜感激！

文章每周持续更新，可以微信搜索「三太子敖丙」第一时间阅读和催更（比博客早一到两篇哟），本文 **GitHub** <https://github.com/JavaFamily> 已经收录，有一线大厂面试点思维导图，也整理了很多我的文档，欢迎Star和完善，大家面试可以参照考点复习，希望我们一起有点东西。