救急

redis 跳表

- 实现有序集合**(ZSET)**：Redis 使用跳表作为有序集合的底层实现之一（与哈希表结合使用）
- 多层结构：由多层链表组成，底层包含所有元素，上层是索引层

- 每层向前遍历，遇到比目标大的节点就「下沉」到下一层

最高的节点是最大的值

分布式锁宕机了怎么办？

redisson 默认会为锁设置一个 30 秒的租期（leaseTime） 没续上就没锁了

强制设置 **TTL**	即使客户端不主动释放，锁最终会过期

rabbitmq为什么可靠高性能

四种交换机类型实现灵活高效的消息分发

轻量级进程和原生并发支持，单节点可处理10万+ QPS

消息、队列、交换机均可持久化到磁盘

生产者确认(publisher confirm)和消费者ACK双重保证

堆和栈的区别

- 栈：
    - 存储基本数据类型（int, float等）
    - 对象引用（指针）
    - 方法调用的上下文（局部变量、返回地址等）
- 堆：
    - 存储对象实例
    - 数组等复合数据结构
    - 静态变量/全局变量（部分语言）

GC的几种常见算法

引用计数（**Reference Counting**）

原理： 每个对象维护一个引用计数，引用增加时计数加一，减少时减一，计数为 0 时回收该对象。
优点： 实现简单，适用于实时系统。
缺点： 无法处理循环引用（A 引用 B，B 又引用 A）。

标记-清除（**Mark-Sweep**）

原理：

- 标记阶段（**Mark**）： 从 GC Root 开始遍历，标记所有可达对象。
- 清除阶段（**Sweep**）： 未标记的对象被释放。
  优点： 能处理循环引用。
  缺点： 可能产生内存碎片，影响分配效率。

标记-整理（**Mark-Compact**）

-改进自标记-清除，解决内存碎片问题。
原理：

- 先执行标记阶段。
- 再将存活对象向一端移动，重新整理地址，最后清理无效对象。
  优点： 避免内存碎片，提高分配效率。
  缺点： 移动对象成本较高。

复制算法（**Copying**）

原理： 将内存分成两块，使用时只用其中一块，当 GC 发生时，把存活对象复制到另一块，并释放原内存。
优点： 速度快，无碎片。
缺点： 需要 2 倍内存，浪费空间。

分代回收（**Generational GC**）

新生代（对象存活时间短）采用 复制算法。

老年代（对象存活时间长）采用 标记-整理 或 标记-清除。
优点： 针对不同对象优化，提高效率。
缺点： 需要额外的调优。

G1和CMS区别

CMS 是 标记-清除 算法，回收老年代，低延迟但会产生内存碎片。G1 是 分区回收，标记-整理方式，减少碎片，能预测 GC 停顿时间，更适合大内存应用。JDK 9 之后 G1 逐步取代 CMS。