XXXXX

底线

- 语言基础
  - Object类下的常用方法
  - hashCode()
  - clone()
  - 接口 与 抽象类 的区别
  - 值传递 / 引用传递 对比
  - 迭代器的使用
  - 基础数据类型，Boxing & UnBoxing
  - final finally 含义
  - 【基础】序列化 / 反序列化
  - Java异常处理 -- 两种Exception ， Error
- 集合类
  - LinkedList / ArrayList / Arrays 比较
  - HashMap实现
  - HashMap / HashTable / ConcurrentHashMap 比较
  - List接口、Set接口和Map接口的区别
- 设计模式
  - 单例
  - 观察者

    ##基础
- 并发编程
  - 阻塞/非阻塞的区别
  - 同步/异步的区别
  - 阻塞IO、非阻塞IO、多路复用IO、异步IO
  - Java 下 进程 / 线程 区别和实现
  - 进程状态转换，对应Java内的状态变化
  - Java线程池的实现
  - IO NIO 比较
- 内存管理
  - Java内存分区
  - 堆 栈 内存区别
  - 哪些内存区域会发生OOM
  - GC的分类
  - 说一种日常使用的GC
- 集合
  - CAS 实现
- 编译器
  - 多态的实现
  - 泛型的理解
  - synchronized / volatile 语义，原理
  - 指令重排序，内存屏障，缓存行失效问题
  - 类加载机制，双亲委派模型，类库举例，过程
- 设计模式
  - 工厂
  - 装饰者
  - ThreadLocal

    ##进阶
- 并发编程
  - CountDownLatch / CyclicBarrier 原理 场景
  - Fork / Join框架（JDK 7+）
- 内存管理
  - 比较 CMS/G1
  - 哪些GC不会发生STW
  - OOM问题定位

- 集合类
  - comparable / comparator 区别
- 编译器
  - 指令重排序，内存屏障，缓存行失效问题
  - 反射，性能优化
  - Class.ForName 作用
- 设计模式
  - 桥接
- 网络库 *
  - Netty
  - 网络模型，线程池
  - 处理TCP 粘包拆包的方法
  - ByteBuf设计
  - 零拷贝
  - EventLoopGroup设计
  - ChannelHandler
  - poll epoll 内核实现 与 Netty封装

Network

- 【底线】OSI网络体系结构与TCP/IP协议模型?
- 【底线】TCP和UDP区别?
- 【进阶】RUDP如何实现?
- 【基础】处理TCP 粘包拆包的方法
- 【底线】TCP几次握手几次挥手？为什么要三次握手和四次挥手
  - 【基础】tcp为什么可靠？为什么需要TIME_WAIT(2MSL等待状态)？为什么建立连接是三次握手，而关闭连接却是四次挥手呢?
  - 三次握手的最主要目的是保证连接是双工的，可靠更多的是通过重传机制来保证的。
  - time_wait，为实现TCP这种全双工（full-duplex）连接的可靠释放，为使旧的数据包在网络因过期而消失
  - 【基础】可靠性：数据包校验，对失序数据包重排序，丢弃重复数据，应答机制，超时重发，流量控制
- 【基础】TCP的拥塞处理
  - 防止过多的数据注入网络中，这样可以使网络中的路由器或链路不致过载。拥塞控制和流量控制不同，前者是一个全局性的过程，而后者指点对点通信量的控制
  - 慢启动
  - 拥塞避免
  - 快重传
  - 快恢复
- 【进阶】SYN攻击？#netstat –nap | grep SYN_RECV
- 【基础】DDOS攻击?
- 【基础】Http和Https的区别?
- 【底线】Get与POST的区别？URLEncoder?
- 【基础】对称加密与非对称加密
- 【基础】从输入网址到获得页面的过程?
- 【基础】ARP协议是什么？作用?
- 【基础】TCP和UDP单个包数据最大大小?
  - UDP 包的大小就应该是 1500 – IP头(20) – UDP头(8) = 1472(Bytes)(避免分片重组)；
  - TCP 包的大小就应该是 1500 – IP头(20) – TCP头(20) = 1460 (Bytes)；
  - 理论上UDP最多可以发送65535– IP头(20) – UDP头(8)＝65507字节
- 【基础】websocket
- 【进阶】Oauth2.0原理和过程
- 【基础】cookie和session
- 【基础】tcpdump 使用
- 【进阶】科学上网

Basic DataStructure & Algorithm

基础算法分类：二分、快排、BFS、DFS、贪心、分治、DP、枚举、概率、回溯、图论（最小生成树，网络流，拓扑排序）、数据结构（并查集、堆）

代表题目

- 【基础】10个经典算法题
  - 【堆】TOP N问题（头条上最火的文章）
  - 【代码功底】生成n阶螺旋数组
  - 【数学】大整数加法
  - 【树】验证二叉搜索树，蛇形打印二叉树，判断一个树是否是另一个树的子树
  - 【DP】
  - 最长递增子序列（LIS）
  - 上楼梯
  - N 切分
  - 【贪心】找零钱问题
  - 【数据结构】实验LRU Cache – 数据结构+基本设计
  - 【链表】多个有序链表的合并（可从2个问起）
  - 【图】图的BFS/DFS 实现
  - 2–sum, 3–sum, 4–sum
- 【底线】几个底线题目
  - 快排
  - 链表中倒数第k个结点
  - 反转链表
  - 二进制中1的个数
  - 二叉树层序遍历
  - 数组中超过一半的数字
  - 底线是必须知道hashmap求解
  - 如果不知道经典解法，考虑减分
  - 二叉树的深度
  - 二叉树 path sum

常见题目汇总

- 基础算法原理
  - 【底线】快排
  - 【基础】非负边最短路径（Dijkstra）
  - 【基础】负边最短路径（Bellman–Ford）
  - 【基础】任意两点最短路径（Floyd）
  - 【基础】最小生成树（prim）
  - 【底线】BFS，DFS
  - 【基础】有向图连通性
  - 【基础】有向无环图判断
  - 【底线】最大公约数（欧几里德算法）
- 【基础】基础算法题 （底线满足，基础 1–Pass / 3–Candidates）
  - 二分
  - X的平方根（开K次方根）https://marvel.bytedance.net/#/question/details/618
  - 数组峰值：https://marvel.bytedance.net/#/question/details/967
  - 数据结构
  - 链表 单链表每隔K个元素进行反转：https://marvel.bytedance.net/#/question/details/1236
  - 头条上最火的文章（TOP N问题）https://marvel.bytedance.net/#/question/details/1021
  - 旋转链表：https://marvel.bytedance.net/#/question/details/955
  - 单链表(奇数位升序，偶数位降序)的排序 https://marvel.bytedance.net/#/question/details/776
  - 给定单链表，求离终点距离为 k 的节点： https://marvel.bytedance.net/#/question/details/586
  - 扑克牌堆栈队列操作：https://marvel.bytedance.net/#/question/details/757
  - 给定数组求数组一个区间，该区间满足区间内最小数乘上区间内数字和结果最大 https://marvel.bytedance.net/#/question/details/666
  - 数组中第K小的数：
  - 求解朋友关系中的朋友圈数量
  - 实现LRU cache
  - 2（n）个有序链表合并
  - 2–sum，3–sum，或者k–sum?
  - 数据结构–矩阵
  - 斜45度打印二维矩阵: https://marvel.bytedance.net/#/question/details/863
  - 生成n阶螺旋数组：https://marvel.bytedance.net/#/question/details/825

- 螺旋打印二位数组：https://marvel.bytedance.net/#/question/details/816
- 数据结构-树
- 对称树判断：https://marvel.bytedance.net/#/question/details/998
- 二叉树转换成双向链表：https://marvel.bytedance.net/#/question/details/927
- 验证二叉搜索树（求二叉树中最大的二叉搜索树）：https://marvel.bytedance.net/#/question/details/979
- 序列化和反序列化二叉树：https://marvel.bytedance.net/#/question/details/969
- 二叉树 Path Sum https://marvel.bytedance.net/#/question/details/1235
- 实现字典树：https://marvel.bytedance.net/#/question/details/1049
- 求二叉树是否存在和值为N的路径 http://marvel.bytedance.net/#/question/details/867
- 蛇形打印二叉树：https://marvel.bytedance.net/#/question/details/859
- 输出二叉树左视角能看到的节点 http://marvel.bytedance.net/#/question/details/772
- 数据结构-图
- 海洋中大陆块数：https://marvel.bytedance.net/#/question/details/898
- 安卓解锁密码数：https://marvel.bytedance.net/#/question/details/826
- 有向图判断环：https://marvel.bytedance.net/#/question/details/118
- 单词接龙：
- 数学
- 用户在线峰值：https://marvel.bytedance.net/#/question/details/1227
- 区间合并：https://marvel.bytedance.net/#/question/details/1050
- 大整数加法（乘法）：
- 36进制正数加法：https://marvel.bytedance.net/#/question/details/861
- 股票买卖问题：https://marvel.bytedance.net/#/question/details/926
- 给出n个点，求最多有多少点在同一条直线上。
- 给定数字下一个比它大的数：https://marvel.bytedance.net/#/question/details/784
- 素数判断：https://marvel.bytedance.net/#/question/details/1032
- 计算 a的n次方 % b；
- 字符串
- 【进阶】编辑距离：https://marvel.bytedance.net/#/question/details/1054
- 最长无重复子串：https://marvel.bytedance.net/#/question/details/976
- 【进阶】贪心
- 找零问题：https://marvel.bytedance.net/#/question/details/1053
- 长度为k的最小字典子序列：https://marvel.bytedance.net/#/question/details/685
- 【进阶】DP
- 矩阵中最长递增路径：https://marvel.bytedance.net/#/question/details/1052
- 环节点的不同走法：https://marvel.bytedance.net/#/question/details/922
- 最大连续子序列之和，最长递增子序列（LIS），最长公共子序列（LCS）
- 有N件物品和一个容量为V的背包，第i件物品的费用是c[i]，价值是w[i]，求解将哪些物品装入背包可使价值总和最大
- 青蛙跳石子问题：https://marvel.bytedance.net/#/question/details/616
- 概率：
- 扔 n 个骰子，向上面的数字之和为 S。给定 Given n，请列出所有可能的 S 值及其相应的概率。
- 轮流抛硬币问题：https://marvel.bytedance.net/#/question/details/1224

Database-mysql

- 【基础】ACID是什么?
- 【基础】delete和truncate删除数据的区别?
- 【基础】你熟悉的mysql引擎有哪些？innodb和myisam的区别?
- 【底线】索引是如何实现的？为什么?
- 【进阶】如何优化慢SQL
- 【基础】有哪些事务隔离级别？分别有哪些问题?
- 【进阶】mysql的事务是如何实现的？mvcc机制是什么?
- 【基础】mysql有哪些锁?
- 【基础】什么是binlog?
  - 【进阶】有哪些模式，区别是什么?
- 【基础】mysql主从复制原理及流程？单库写多的场景如何提高从库复制效率?
- 【基础】online schema change如何实现？原理是什么?
- 【进阶】不小心删除了一些数据？如何快速恢复?
- 【基础】什么场景下需要分库分表？有什么用?
- 【进阶】你理解的mysql proxy需要起到哪些作用?

Redis

1. 【底线】Redis有哪些数据结构?
    1. 字符串String、字典Hash、列表List、集合Set、有序集合SortedSet。
2. 【基础】Redis分布式锁
    1. setnx+expire直接使用set的参数一次设置避免死锁
3. 【基础】Redis如何做持久化的?
    1. bgsave做镜像全量持久化，aof做增量持久化。因为bgsave会耗费较长时间，不够实时，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实例重启时，会使用bgsave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。
4. 【基础】bgsave的原理是什么?
    1. fork和cow。fork是指redis通过创建子进程来进行bgsave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。
5. 【基础】Redis采取的过期策略
    1. 【进阶】expire机制如何实现
    2. 懒汉式删除+定期删除
6. 【基础】MySQL里有2000w数据，redis中只存20w的数据，如何保证redis中的数据都是热点数据
    1. redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。redis 提供 6种数据淘汰策略：
    2. voltile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰
    3. volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰
    4. volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰
    5. allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰
    6. allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰
    7. no-enviction（驱逐）：禁止驱逐数据
7. 【进阶】Memcache与Redis的区别都有哪些?
    1. Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。memcache支持简单的数据类型，String。
    2. Redis支持数据的备份，即master-slave模式的数据备份。
    3. Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用,而Memecache把数据全部存在内存之中
    4. Redis的速度比memcached快很多
    5. Memcached是多线程，非阻塞IO复用的网络模型；Redis使用单线程的IO复用模型。
8. 【基础】pipeline的好处是什么
9. 【进阶】zset底层实现的数据结构
    1. 跳跃表+map

MQ

- 【底线】
    - 简单介绍对消息队列的理解?
    - 说说常用的消息队列都有哪些？用过哪些?
    - 为什么使用消息队列？使用场景(与接口调用方式对比，利用消息队列进行系统交互)：
    - 解耦：简单点讲就是一个事务，只关心核心的流程。而需要依赖其他系统但不那么重要的事情，有通知即可，无需等待结果。换句话说，基于消息的模型，关心的是“通知”，而非“处理”。
        - 上游系统A不需要关注下游服务B是否可用；下游服务B异常，A不需要单独存储数据，再进行重试；
        - 也不用关注B业务变更，需要取消同步通知。
    - 异步：解决非核心影响，同步调用多个下游。
    - 削峰：服务高峰时段抗不住。
- 【基础】
    - 使用消息队列有什么缺点?
    - 系统可用性降低：系统引入的外部依赖越多，越容易挂掉，本来你就是A系统调用BCD三个系统的接口就好了，人ABCD四个系统好好的，没啥问题，你偏加个MQ进来，万一MQ挂了咋整？MQ挂了，整套系统崩溃了，你不就完了么。
    - 系统复杂性提高：硬生生加个MQ进来，你怎么保证消息没有重复消费？怎么处理消息丢失的情况？怎么保证消息传递的顺序性？头大头大，问题一大堆，痛苦不已
    - 一致性问题：A系统处理完了直接返回成功了，人都以为你这个请求就成功了；但是问题是，要是BCD三个系统那里，BD两个系统写库成功了，结果C系统写库失败了，咋整？你这数据就不一致了。
    - 如何保证消息不被重复消费？如何保证消息队列的幂等性?
    - 什么场景会造成消息重复消费？网络传输等等故障，确认信息没有传送到消息队列，导致消息队列不知道自己已经消费过该消息了，再次将消息分发给其他的消费者。

    - 如何解决?
        - 唯一键：生产消息带上唯一id、通过消息体某个特征判断
        - 存储消息唯一键(redis 或 mysql)
        - 通过唯一键判断是否消费过
    - 如何解决数据一致性？CAP理论
- 【进阶】
    - 消息队列如何选型?
    - 消息消费是push还是pull
    - kafka、activemq、rabbitmq、rocketmq都有什么优点和缺点啊?
    - 如何保证消息队列是高可用的?
    - 如何保证消费的可靠性传输?
    - 生产者弄丢数据
    - 消息队列弄丢数据
    - 消费者弄丢数据
    - 如何保证消息的顺序性?
    - 秒杀系统设计