自我

小米：**小米金融app的服务端的需求的开发**（新版本的个人中心接口（支持多app、修改写死数据、修改埋点信息、红点）、重构组件处理逻辑（支持多app、新增排序组件处理、抽象组件处理逻辑组织成责任链方便维护））。**CMS的需求的开发**（支持多app、支持编辑组件顺序、）。控制层和业务层用thrift进行调用，定义接口、Bean、枚举

Boss直聘：**推荐平台的需求的开发。**（接入内部员工信息、增加推荐实验插件参数校验、统一trace系统的业务线（将trace提供的接口对业务线的分类与平台上的细分业务线进行映射，然后实现可配置）、压测系统的种子收集新分类）

优惠券项目：模板（创建优惠券模板、异步生成优惠券码）。分发（校验优惠券领取条件、生成优惠券信息）。结算（根据传来的优惠券，根据优先级排序，放进责任链处理，责任链从处理器管理获取对应处理器进行处理）。

学校：带宽调度。大背景就是解决数据中心广域网上的传输任务的流量调度问题。让网络在时间、链路、需求的约束下能够容纳更多的传输任务。将传输切分为时隙，以时隙为最小粒度计算预留带宽和路径。贪心的顺序占用，优化线性方程。

规划：先在实战中继续磨练积累技术，熟悉业务，建立起更深更广的理解吧。

JDK

面向对象：封装。继承。多态：接口抽象类父类，覆写重写重载。重写和重载。多态实现（动态链接）

抽象类和接口：前者是

基本数据类型：8个 大小 范围

访问控制符：4个：public都可、protected不能不同包（子类可不同包）、defailt不能子类不能不同包、private只能同一个类内

内部类：成员、静态、方法、匿名

异常：错误（代码无法处理）。异常（可以通过tc来捕获并处理）：受检和非受检

JDK8新特性：函数式接口，Lambda，流式编程

static：修饰类，修饰方法，修饰变量，修饰导包

final、finally、finalize：final修饰变量、方法、类、参数。finally在异常处理中使用。finalize方法在对象被清理时执行一次。

Object：getClass、hashCode、equals、clone、wait、notify、notifyAll、finalize、toString

克隆：浅克、深克。

封箱拆箱：基础类型转为包装类型。除了浮点类的包装类型初始化缓存池，存储[-128-127]的对象

序列化：

反射：获取Class对象，可以获取类信息，可以执行类方法，可以创建类的实例

JVM

类加载：5步：加载（class文件转为字节码载入内存）、连接（验证（字节码符合类加载器）、准备（分内存赋零值）、解析（符号转直接引用））、初始化（收集初始化代码，先父类后子类）。

双亲委派：类加载器收到类加载请求时，先托付给父加载器，如果父加载器不行，才尝试自己。破坏机制：自定义一个类加载器，继承ClassLoader类，重写loadClass方法，用这个加载器加载指定类，用反射获取实例来执行。

卸载时机：类加载器被回收、实例都被回收、class对象没有被引用了

JVM内存中结构：5个。程序计数器、虚拟机栈、本地方法栈、堆、方法区

GC：GCROOTS：栈中引用的、静态变量常量引用的对象，活动的线程对象。标记-清除、复制、整理，基于对象特点分代。

MGC触发：对象优先分配到伊甸园。如果没有空间了就准备进行MinnorGC。先进行**分配担保**：MinorGC前检查老年代大于年轻代才证明GC是安全的。如果年轻代大，就看是否设置了允许担保失败。如果允许继续检查老可用是否大于历届平均晋升，如果大就还尝试MinorGC，否则直接FullGC。

进入老年代时机：MinnorGC时新对象放不到幸存者区、大对象（可设置）、年龄晋升（经过MGC一定次数）、动态年龄判断：对象进入幸存者区后，对象年龄和以下的容量总和超过阈值，就将大于年龄的对象加入老年代

内存分配过程：

CMS：五步，两个STW：初始标，并发标，重新标，并发清，并发重置。缺点cpu敏感，浮动垃圾，碎片

G1：可以预测停顿时间，设置最大时间，根据价值维护区域优先队列。四步，三个stw：初始标、并发标、最终标、筛选回收

OOM：栈溢出、堆溢出、元空间溢出、直接内存溢出，不能创建新线程.

Jmap导出堆转储文件，或者打印GC信息包括那些区域GC了多少次

内存泄漏和溢出：溢出是什么，泄漏：jvm无法清除实际上已经无用的对象，导致内存不足。ThreadlocalMap与线程保定，其中key为弱引用，为了让没有主动

操作系统

操作系统是什么：管理计算机硬件和软件资源，屏蔽了复杂度。

什么是内核态用户态：根据进程访问资源的特性，分为两种状态。内核态进程直接读取用户程序数据，进程可以通过系统调用进入内核态来访问计算机所有资源。

进程和线程的区别：进程是（代码在数据上的执行过程，资源单位，包含多线程），线程是（调度执行单位，可以共享进程的公共资源），从四个方面说（资，调，销（前者涉及页置换，对处理器缓存影响大），空（前者内存隔离，后者可共享进程资源），通（前者要用IPC，后者可通过读写所在进程进行间接通信））

进程状态：五种（创建、就绪、运行、阻塞、终止），再引入挂起状态

进程通信：文件、**管道**：父进程到子进程建立，单向。**命名管道**：去除父子限制，消息队列，**信号量**，**共享内存**：分配空闲区域，把逻辑地址和物理内存的共享区域去映射，同步靠信号量，socket

进程调度算法：三种系统。交互系统：分时、优先级、多级反馈

内存管理：逻辑和物理，CPU需要虚拟地址值，MMU转换，过主线在内存中取返回。**连续内存分配**：空闲分区表：首次适配、最优、最差。**非连续内存分配**：**分段**：根据内容构成（栈堆程序数据代码），段表项base，limit。**分页**：内存分为大小同单元，逻辑页，物理帧，页表项：标志位帧号。段页。虚拟内存。

内存置换算法：虚拟内存需要。FIFO、LRU（最近最久未用）、LFU（最不常用）、CLOCK

死锁：概念，4个必要条件（互斥、非抢占、请求保持、循环等待）。死锁预防（禁止一个必要条件）、避免（获取资源检测安全状态）、检测（记录获取锁，遍历检查是否存在环）、消除（重启、撤销进程、回退进程）

JUC

Java线程状态之间的关系：6个状态（准备、运行、阻塞、等待、超时等待、终止）

线程安全：什么是。怎么保证安全。隔离（局部、TL）、共享区域加同步锁、乐观锁

线程怎么通信：通过访问公共变量（volatile、锁，原子类）、等待通知机制、join()方法、同步工具类

Java中有什么锁：宏观分类悲观乐观，悲观有：synchronized、lock。

volatile实现：可见性（原理：读（缓存一致性：写主存会让核心中缓存对应副本值无效，发现无效后会重新读取、总线加锁），写（lock前缀要求强制把写缓存写回主存））。防止指令重排（原理：在字节码中合适的位置插入内存屏障）。

CAS：是什么。ABA问题。解决。通过带时间戳的原子引用。

synchronized：java关键字，实现访问资源的同步。本质就是线程对一个对象的monitor的获取和释放。用在什么地方，进出，同步队列，等待通知模式。监视器中有阻塞队列和等待队列，就绪线程通过cas和所有队列之外的线程竞争锁，导致**非公平**。锁升级，四个状态，说怎么加锁解锁和场景。

AQS：抽象队列同步器。提供模板方法来简化同步工具的开发。用volatile修饰的int类型变量表示同步状态，用FIFO队列表示等待队列维护阻塞线程。本身提供了操作同步状态，加入队列，进入等待、唤醒线程等方法，让子类实现tryAcquire来判断什么情况下获取和释放资源成功还是失败。

可重入锁大概实现：定义了两个子类来实现AQS中的独占和共享模型的获取资源释放资源。共享模式中通过加入先判断获取资源线程是否为队列中的第一个来实现公平。

常用同步器：CountDownLatch：使用countDown()来倒数，归零前await()方法阻塞。CyclicBarrier：线程执行await()会阻塞，当阻塞数量达到指定数量就会开始执行。Semaphore：设置数量。执行acquire()数量-1，为0时阻塞；执行release()数量+1，为执行数值时阻塞。

线程池：Executors创建三种，newFixedThreadPool(int nThreads)：核心和最大为n，存活时间0，LinkedBlockingQueue没有上限；newSingleThreadExecutor()：上面的n为1；核心0最大int.max，存活60s，SynchronousQueue上限为int.max。ThreadPoolExecutor类的**7个参数**。流程。**ctl变量**前三位表示状态和后29位表示线程数。mainLock保证添加worker的原子性（不能超过约束）。**worker**表示线程包含第一个任务task，继承AQS（AQS独占锁+不可重入来反应当前执行任务状态）实现Runnable。复用关键方法runWorker(Worker w)。没达到核心数添加任务时会创建线程执行任务，**为了预热**。关闭方法：shutdown中断未执行任务的线程，shotdownNow中断所有线程返回任务列表。

线程方法：sleep，yiled、join（把被join的线程作为锁对象，判断其为运行状态后执行wait进入锁的等待队列，当要终止时会nf自己）、interrupt。Object方法：wait、notify、notifyAll

ThreadLocal：数据结构（线程有TLM，key为TL为弱，value为强）。为什么弱（key没有外部引用（线程任务中）后，可以被检测并移除value）。内存泄漏（没有remove，并且TLM不在setgetremove）。

集合

接口之间的关系：迭代器（Collection（List（ArrayList、LinkedList）、Set（HashSet、TreeSet）、Queue（Deque（ListedList、ArrayDeque））、Map（HashMap、HashTable）、ConcurrentMap（ConcurrentHashMap）、SortedMap（TreeMap））

list：两个实现类，底层结构，区别，方法的时间复杂度，线程安全的

map：put，扩容（先扩容在添加。数组未初始化。元素个数大于阈值。1.8扩容时机+1大于阈值且节点不为空），1.7 1.8区别（引入红黑树、扩容区别）

并发map：1.7put，扩容，size：统计每个segment中元素数量，如果modCount变化，则再重新尝试，尝试两次，如果还是变化则锁住Segment。1.8put，扩容，size

布隆：多哈希+bitmap

MySQL

存储引擎区别：InnoDb支持事务，支持行锁，支持外键，支持索引

索引的分类：聚簇非聚簇，叶子节点不同。单列（主键、唯一、普通）、组合

索引的结构：B+树特点，多路查找树，非叶子只存索引，索引在子节点中冗余，叶子节点相连

优化：什么时候用索引，不用索引，索引失效的情况，要注意地：最左，联表，前缀索引。。检查执行计划

更新过程：执行器调用存储**引擎接口查询数据**，引擎检查数据页是否在内存如果不在在磁盘中查询后**读入内存返回给执行器**，执行器进行计算然后调用引擎**进行更新**，引擎更新内存数据页，**记录undo、redolog**并告知执行器已经**准备**好，执行器**记录binlog**然后调用引擎**提交**，引擎将**redolog标记为提交**，返回更新成功。

redolog：侧重于维持原子性和持久性，用来恢复数据。innoDB引擎的日志，记录物理数据的修改。先写入缓存（循环写入），后台线程根据不同策略写入磁盘。

binlog：侧重于数据的转移和同步，用于主从同步和基于时间点还原。是mysql的日志，记录逻辑修改。先写缓存，然后落盘。

事务特性：A原子性：redologundo、C一致性、I隔离性：MVCC和锁、D持久性：redolog。

事务隔离级别：四个：未提交读、已提交读能解决脏读、可重复读解决不可重复读和部分幻读、串行化解决幻读，意思，不能避免什么问题，InnoDB怎么实现

锁：共享（允许多个事务获取数据的共享锁，对数据只读）、排他（不允许其他事务获取锁）、间隙（RR、）、临键

MVCC：读不加锁。通过undolog生成readview，得到事务所能看到的数据版本。事务中select时会创建一个当前事务版本的readview，如果是度提交事务中之后的每次select都重新生成一个readview，如果是可重复度，则之后的select会用最开始生成的readview。

explain：可以帮助选择建更合适索引。type：system > const > eq_ref > ref > range > index > ALL。key：实际用到的索引。Extra：额外信息。Using filesort：用没用到外部排序

网络

OIS7层协议：应用、表示、会话、传输、网络、数链、物理

TCPIP5层协议：应用、传输、网络、数据链路、物理

dns：服务器：本地、根、顶级、权限。数据同步同tcp，域名解析用udp

http：结构：请求（请求行（方法 URI 版本）、头部资源、空行、请求体）。响应（状态行（版本 状态码 短语）、头部字段、空行、响应体）。功能：长连接（复用tcp，2.0解决头部阻塞）

https：https+加密（混合加密）+认证（证书）+完整性保护（数字摘要）。1.客发连，版本、加密法、随机1 2. 服发加密法、随机2。3.服发数字证书（公钥、CA签名、服信息）， 4.客验证，生成会话密钥，公钥加密发送。5.服私钥解密得到会话秘钥。6.客验证密钥，开始使用。

TCP：和UDP区别 三次挥手（排除历史连接、确定序号） 四次握手（双方传完） 保证可靠 流量控制（滑动窗口，窗口大小） 拥塞控制（慢开始（窗+窗，直到上限）、拥塞避免（窗+1。超时拥塞后上限改为窗半，窗1）、快重传（接收三个重复ack就重发）、快恢复（快重传后窗半，拥塞避免））

getpost：功能、数据位置（uri后面；请求体（允许二进制数据））和大小、安全性（get参数可看、可缓存可回退，post参数不可）

cookie session： 两者都是为了在无状态的HTTP是实现保存会话。前者是将数据保存在浏览器，请求时带上Cookie值。后者是为请求会话生成的状态缓存，在服务器生成和存储，把id放到请求放的Cookie存储来识别会话。sessionId或者其他标识信息还可以放在头部字段中、url后面

Redis

5大基本数据类型、6种底层结构（简单动态字符串、整数数组、双向链表、哈希、压缩列表（表头尾部存储字段）、跳表（链表上增加多级索引））

渐进性哈希：为了避免进行扩容时发生长时间的阻塞。扩容进行链表转移时，每个请求的操作涉及到的链表才进行转移。

持久化：**快照**（bgsave，让子进程对内存数据进行全量持久化）。**AOF**（写后日志）。AOF**重写**（子进程生成快照，然后把快照的aof日志写入临时文件，主进程继续处理请求，同时写操作记录aof也到重写缓冲，子进程写完后，把重写缓冲加入临时文件，然后替换之前的aof文件）。混合方式：定期快照，之前记录aof。

键过期：**惰性删除**（访问key是检查过期）+ **定期扫描**（默认每10秒从过期字典中随机20个key，删除其中过期的，如果超过1/4则再扫描）

内存淘汰策略：报错、普通lru、过期lru、普通随机、过期随机、过期ttl、普通lfu、过期lfu。**lru原理**：24位存储时间戳，需要淘汰时，随机采样获取n个key，然后删除最久没使用的。**lfu原理**：16位存时间戳，8位存频率。随机采样n个可以，删除频次最低的。

主从复制：一般主从用**读写分离**。同步机制：1.从与主**建立连接**。2.主bgsave出**rdb**发给从并且同时把之后的操作记录到缓存，从清空然后加载rdb。3.主把缓存中的**aof**记录发送给从，从执行。

哨兵机制：负责主从模式下的故障转移。哨兵集群交互通过发布订阅（哨兵在主库发布自身，然后订阅其他哨兵）。哨兵主要任务：**监控**（ping主、从判断下线。利用集群从主观下到客观下线）、**选主**（筛选+打分）、**通知**（被客户端订阅，客户端可获知主库下线、主库切换等信息）

穿透（空值、布隆）、**击穿**（做高可用、提高性能）、**雪崩**（随机过期）

数据一致：读写缓存（保证操作缓存和DB具有原子性）。只读缓存（新增不存在的key的数据不用处理。修改数据时：1.**先写DB后删缓存**。2.**先删缓存后写DB**：使用延时双删（写DB等一会再删除缓存一次）防止缓存一直使用旧数据）。为了保证两个都处理，可以进行重试。

并发访问：保证客户端的读并修改的操作并发安全，有两种方法：加锁（分布式锁）+原子操作（转为单命令（decr、set key value ex 10 nx）、lua脚本）

设计模式

观察者：被观察者要管理观察者集合，提供通知功能执行观察者的update方法。观察者实体继承观察者接口实现update方法，完成在被通知后的处理逻辑（比如获取被观察者的数据）。

责任链：定义问题类。处理者抽象类定义下一个处理者属性、模板方法处理问题（其中包含解决问题、如果失败传递给下一个处理者、如果成功怎么做。这些方法也定义出来让子类实现）。管理类来注册处理者，处理到来问题。

分布式

CAP：一致性（任意节点能够读到最新数据）、可用性（操作能够得到相应正确且不超时的结果）、分区容忍性（对分区的容忍能力，分区容忍度越高，网络分区的增加对一致性和可用性的影响越小）

BASE：对CAP理论中的一致性和可用性做权衡。BA：基本可用。允许降低部分可用性。SS：软状态。允许数据存在中间状态，即允许系统在多个节点上的数据副本存在延时。EC：最终一致性。数据不会一直处于软状态，最后在期限内会达到数据的一致性。

复杂均衡：数据传输分散到多台服务器。实现：重定向（发给前置机，然后重定向让客户端发送到指定服务器）、反向代理（通过前置机进行重新转发，实现：交换机、nginx软件）。算法：轮询、原地址哈希、最小连接、一致性哈希（保证在新增和删除节点时映射的稳定。将一个哈希值区域组织成环形，根据节点将环形划分成多个区间，对请求哈希值取余映射到区间。如果节点移除则区间也会合并）

Spring

IOC：类之间的依赖关系解耦。

AOP：动态代理。基于接口、基于多态继承，重写回调。拦截器链。

拦截器：interceptor，依赖StringMVC，基于反射，对dispatcher分发请求之后的controller前后进行处理。

过滤器：filter，依赖servlet容器，基于函数回调，对http请求的整个流程进行拦截。

IO模型

阻塞非阻塞：被调用者是否直接返回。同步异步：调用者是否需要等待结果

四个模型：BIO（同步阻塞）NIO（同步非阻塞）AIO（异步非阻塞）

消息队列

队列模型

发布订阅模型：生产者发布、消费者订阅、broker把消息放置到主题中，主题分为多个分区，分区内消息有序。让不同消费者组消费，可以消费多次，通过每个消费者组维护的标记。组内的消费者可以并发消费不同的分区。

3种问题：消息丢失：三个阶段分析、确认。重复消费：几个级别。消息积压：分析问题消费还是生产