【自我介绍】

**面试官**您好，我叫**胡景峰**。

**本科**就读于 合肥工业大学 计算机学院

**硕士**就读于 东南大学 软件学院（明年6月毕业）

本科和硕士阶段的 **课程成绩排名** 均为前5%

硕士组内是以 **虚拟现实与人机交互** 为大方向

我的科研方向主要是 **渲染+边缘计算** 做一些优化

个人技术栈以**Java**为主，熟悉Spring、MySQL、Redis等

平时有写文档或博客**反思总结**的习惯

是一个**长期主义**者，相信 微小而正确的习惯 带来的 复利效应

【英文介绍】

Hello, my name is Jingfeng Hu.

I **completed undergraduate studies** in Computer Science at Hefei University of Technology.

I’m currently **pursuing Master's degree** in Software Engineering at Southeast University. （and will graduate in 2026.）

I have earned **several honors**, including:

* National Scholarship
* Title of Excellent Graduate

and prizes from competitions like

* China Software Cup
* Mathematical Modeling Contest

I usually use **Java** and I’ve learned about:

* Concurrency Mechanisms
* TCP/IP and HTTP protocols
* MySQL and Redis

I **value teamwork**, **enjoy challenges**, and am committed to **continuous learning and improvement**. I believe my **technical skills and positive attitude** will allow me to **quickly adapt and contribute effectively to our team**.

【项目经历】

吃了么外卖平台

这是一个**网络教程项目**，主要想通过项目实战**理解并应用Spring**。

1. Spring IOC、AOP
2. 分页查询
3. 订单状态定时处理

分页查询

MyBatis的PageHelper实现

使用 **LIMIT 子句，借助偏移量、每页记录数**实现分页查询

项目中使用了MyBatis的PageHelper：

1. 查询前调用 **startPage**将当前页码、每页记录数存储在ThreadLocal
2. 实现拦截器接口，当查询方法被调用时，检查分页参数
3. 在原始SQL中添加相应的分页语句
4. 执行分页查询
5. 结果被封装到**PageInfo**中，包含总记录数、总页数、当前页码

深度分页查询

当分页到比如1k页以上时，会出现**深度分页查询**，导致：

1. 需要扫描大量的数据页才能定位到指定页的数据
2. 临时结果集占用过多内存，使用磁盘存储数据，慢

**优化方案：**

1. 覆盖索引避免回表：如果查询只需要 id、name 列，且这两列在一个联合索引中，那么可以创建覆盖索引来优化查询。
2. WHERE子句过滤数据实现基于主键的游标分页：使用上一页的最后一条记录的主键ID作为下一页查询的条件。
3. 优化数据库的缓存大小、应用层Redis缓存 、分库分表减少单表数据量

公共字段自动填充

自定义注解+ AOP面向切面编程实现

采用**自定义注解**+ **AOP面向切面编程**自动填充**创建时间、更新时间**等字段

1. 定义**枚举** FieldFill标识INSERT还是UPDATE
2. 自定义**注解**@AutoFill
3. 创建**切面类**AutoFillAspect获取方法参数并通过反射设置公共字段的值
4. 在需要自动填充的 Mapper 方法上，**添加** @AutoFill 注解

订单状态定时处理

Spring Task实现定时任务

* 超时未支付订单自动取消
  1. 待支付 && 下单时间 < 当前时间-15min
  2. @Scheduled(cron = "0 \* \* \* \* ?")
* 自动完成长时间处于派送中的订单  
  （比如tb确认收货，用户已收货但未确认）
  1. 派送中 && 派送开始时间 < 当前时间-60min
  2. 凌晨1点是为了避开日志、备份等系统维护任务
  3. @Scheduled(cron = "0 0 1 \* \* ?")

1. 启动类@EnableScheduling开启定时任务
2. 任务方法添加@Scheduled，结合Cron表达式定义执行时间  
   秒 分 时 日 月 周 [年]

Spring Task潜在问题和解决

1. **默认单线程执行，任务阻塞**
   1. 配置线程池 ThreadPoolTaskScheduler
   2. 添加 @Async 注解使用异步执行
2. **异常未处理导致任务停止**
   1. try-catch
   2. 实现 ErrorHandler 接口，统一处理定时任务中的异常
3. **不支持分布式环境，任务重复执行**
   1. 分布式锁
   2. 引入分布式任务调度框架，如 Quartz、XXL-Job

对菜品信息进行缓存、数据一致性问题、Redis

明显不可取的方案

**方案1：先写缓存、再写数据库（×）**

**方案2：先写数据库、再写缓存（×）**

**方案3：先删缓存、再写数据库（×）**

* 苍穹外卖：在增删改时删除匹配特定模式的Redis键

可根据实际情况选用的方案

**方案4：先写数据库再删缓存**

* **缺点：**时间窗口内数据不是最新的
* **适用：**对一致性要求不高，追求性能  
  **社交媒体平台：**用户发布内容后可能会编辑，但大多用户对延迟不敏感

**方案5：延时双删：删缓存→写数据库→延迟再删缓存**

* **缺点：**时间窗口内可能会读取旧数据并回写缓存
* **适用：**读多写少、对一致性要求高  
  **商品详情页：**用户频繁浏览商品信息，更新操作相对较少

**方案6：先写数据库再通过Binlog异步更新缓存**

* **理由：**高并发场景删除缓存可能导致击穿并不合适
* **缺点：**需要引入Canal、存在一定的延迟
* **适用：**高并发、对数据一致性要求高的分布式系统  
  电商平台的库存管理：库存数据频繁变动，且对一致性要求高

Spring IoC

依赖注入3种方式

控制反转IoC是**思想，**依赖注入DI是实现方式

1. **构造器注入（推荐）**：适用于强制依赖的注入
2. **Setter注入**：适用于可选依赖或个人开发的模块的注入
3. **@Autowired字段注入（不推荐）**：因为无法通过构造器或Setter方法修改依赖关系，降低了代码的可维护性和可测试性。

Bean 的作用域

1. **singleton（默认）：** 单例bean
2. **prototype：**适用于验证码这种短期使用的、有状态、非线程安全的对象
3. **request：** 每个请求（如表单数据处理）创建一个实例
4. **session：** 每个会话（如用户信息缓存）创建一个实例
5. **application/global-session：** 在Web应用启动时（如统计数据）创建一个实例
6. **websocket：** 每个WebSocket会话（连接状态管理）创建一个实例

Bean 的生命周期

1. 容器通过**反射**根据**配置文件**实例化 Bean
2. **依赖注入**
3. **初始化**
   1. **Aware回调注入**应用：使Bean获取上下文信息、其它 Bean，用于日志标识
   2. **BeanPostProcessor#postProcessBeforeInitialization**  
      应用：创建动态代理实现AOP
   3. **Initialization**  
      应用：缓存预热、数据库连接池初始化
      * + @PostConstruct
        + InitializingBean#afterPropertiesSet
        + initMethod
   4. **BeanPostProcessor#postProcessAfterInitialization**
4. **使用**
5. **销毁** @PreDestroy调用DisposableBean#destroy

Spring AOP

AOP面向切面编程

通过代理的方式，拦截穿插，再调用真正方法。

把权限校验、日志记录这种通用功能模块化，避免代码重复。

切面 = 切点 + 通知

* **切面：**封装了权限校验、日志记录通用功能
* **切点：**要织入的方法
* **通知：**拦截后要执行的操作

动态代理：JDK vs CGLIB

**JDK 动态代理**

* SpringFramework的默认实现
* 基于接口实现，通过反射机制实现
* 注意没有接口的话会有报错

**CGLIB 动态代理**

* SpringBoot2的默认实现
* 基于类继承，通过ASM 字节码生成工具生成子类
* 注意不能代理 final 类和 final 方法

12306 铁路购票系统

GitHub找的**开源项目二开**，主要想对**高并发**有一个实践和应对。

1. 分布式：分库分表
2. 数据一致性、缓存穿透问题
3. 对于节假日抢票，TOKEN令牌限流
4. RocketMQ延时消息 取消超时未支付订单

通过复合分片算法完成订单数据分库分表，支持订单号和用户查询

为什么要分库分表

冷热数据的架构方案：账号只能查看最近一个月内的订单购买记录

订单分库分表

需求：

* 用户要能查看自己的订单
* 支持订单号精准查询

选择订单号和用户ID作为复合分片键

每次查询时需要带着用户和订单两个字段，非常不方便：

采用基因法把 用户 ID 后六位 冗余到 订单号 里（tb订单）

按用户 ID 后六位分库分表：

分 2 个库order\_0和order\_1，以及对应业务 16 张表

分片键定义为用户 ID 和订单号：

shardingColumns: user\_id,order\_sn

只要查询中携带这两个字段，我们就取用户 ID 后六位进行查找分片表的位置：

首先判断 SQL 是否包含用户 ID，如果包含直接取用户 ID 后六位

如果对订单中的 SQL 语句不包含用户 ID 那么就要从订单号中获取后六位，也就是用户 ID 后六位

雪花算法生成订单号（有序、唯一、性能）

64 位的长整型（long），通常采用以下结构：

* 1 位：符号位（通常为 0）
* 41 位：**时间戳**（毫秒级）
* 10 位：**机器 ID**
* 12 位：**序列号**

分布式组件、SpringCloud、Nacos

封装缓存组件库，避免注册时用户名全局唯一带来的缓存穿透问题

缓存穿透（不存在）

请求不存在于数据库的数据 造成 数据库负载、缓存内存耗尽、用户体验差

缓存穿透解决方案

1. 缓存不存在的 Key并把值设为Null，设置短暂过期时间比如60秒  
   **缺点：**尝试但没注册一个不存在的用户名，该值60s内都不可被注册
2. 分布式锁 保证 只有一个线程访问数据库  
   **缺点：**其他用户注册请求缓慢或超时
3. **布隆过滤器** 存已注册用户名  
   布隆过滤器中没有的一定不存在 = 可用  
   布隆过滤器中有，再查缓存或数据库  
   **缺点：**不能删除元素，注销用户名无法再次使用。

最终解决方案

**布隆过滤器 + Redis Set缓存**

1. 布隆过滤器不存在 = 数据库没有 = 可用

2. Redis Set缓存存在 = 已注销 = 可用

3. 查询数据库到底有没有

使用 BinLog 配合 RocketMQ 消息队列完成 MySQL 数据库与 Redis 缓

存之间的数据最终一致性。

明显不可取的方案

**方案1：先写缓存、再写数据库（×）**

**方案2：先写数据库、再写缓存（×）**

**方案3：先删缓存、再写数据库（×）**

* 苍穹外卖：在增删改时删除匹配特定模式的Redis键

可根据实际情况选用的方案

**方案4：先写数据库再删缓存**

* **缺点：**时间窗口内数据不是最新的
* **适用：**对一致性要求不高，追求性能  
  **社交媒体平台：**用户发布内容后可能会编辑，但大多用户对延迟不敏感

**方案5：延时双删：删缓存→写数据库→延迟再删缓存**

* **缺点：**时间窗口内可能会读取旧数据并回写缓存
* **适用：**读多写少、对一致性要求高  
  **商品详情页：**用户频繁浏览商品信息，更新操作相对较少

**方案6：先写数据库再通过Binlog异步更新缓存**

* **理由：**高并发场景删除缓存可能导致击穿并不合适
* **缺点：**需要引入Canal、存在一定的延迟
* **适用：**高并发、对数据一致性要求高的分布式系统  
  电商平台的库存管理：库存数据频繁变动，且对一致性要求高

Canal

通过 RocketMQ 延时消息实现购票十分钟未支付情况下取消订单功能

RocketMQ延时消息实现

* 订单生成时，向 RocketMQ 发送一条延时消息，设置十分钟
* **十分钟后消息投递给消费者**，如果仍为未支付，则取消并释放库存。

订单延时关闭功能技术选型

#### **定时任务（xxl-job、PowerJob、shardingsphere-elasticjob）**

* + **延时精度问题：**调度器的轮询 + 分发延迟导致的
  + **高并发问题：**大量定时任务同时执行 系统负载过大
  + **分库分表问题：**因为根据创建时间查询扫描一批订单进行关闭没有携带分片键，存在读扩散问题。

#### **Redis 过期监听 / Redisson的RDelayedQueue**

1. **延时精度问题：**Redis 的过期监听通过定时器实现
2. **宕机重启问题：**设置了过期时间&&未过期的订单无法正确关闭
   1. Redis 5.0前如果宕机或者重启，这个消息不会重复投递
   2. Redis 5.0推出了 Stream 功能，有了持久化功能

#### **RabbitMQ**

* **延时精度：**RabbitMQ基于消息的Time-To-Live实现延时消息，消息过期检测是惰性触发的，等下一次投递消息、消费消息才“顺便”检查
* **高并发问题：**消息堆积
* **重复消息问题：**由于网络原因或其他不可预知的因素，可能会导致消息重复发送到订单队列。如果没有处理好消息的幂等性，可能会导致订单重复关闭的问题，从而造成数据不一致或其他异常情况。
* **部署难度：**Rabbitmq是一个大型中间件，光docker部署就上1个G了，而且还封闭，性能比不过Rocketmq

**RocketMQ**

* 时间轮算法
* 设置消息的 delayLevel 来指定延迟级别

【荣誉奖项】

“中国软件杯”大学生软件设计大赛国家三等奖

* 基于**金蝶云低代码平台**这一完全陌生的技术栈，通过阅读文档短时间内零基础学习开发完成**图书管理系统**
* 锻炼了**在未知领域中快速学习与落地**的能力
* 培养了**协调资源、带领团队**的组织能力

全国大学生数学建模竞赛省二等奖

* 与队友**连续奋战三天三夜**，从问题建模、算法实现到论文撰写，过程中深刻体会到**持续专注推进一件复杂任务**所带来的成就感。
* 锤炼了数理逻辑与编程建模能力
* 提升了**在高压下保持思维清晰、推进任务执行**的能力

【反问环节】

* 能谈一谈您作为一个公司老员工**对公司的感受**，和其他公司不同之处？
* 我觉得我这次表现不是很好，您可以给一些**评价或建议**吗？
* 这个岗位为什么还在招人？实习生主要工作？