韦宇

性别：男

年龄：21

电话：18933549212

学历：本科

邮箱：3658043236@qq.com

求职意向：测试开发

CSDN: https://blog.csdn.net/wy990880?type=blog

教育经历

广东药科大学　　计算机科学与技术　本科　　2022-09 ~ 2026-06

专业技能

- **后端基础：**熟悉Java及Golang编程语言，有一年使用经验，掌握Java**集合框架**、异常、多线程、反射等核心机制
- **MySQL：**熟悉MySQL基础原理、**存储引擎**、**索引原理**、MVCC、事务等机制、具备一定的SQL性能调优能力
- **Redis：**熟悉Redis底层数据结构、分布式锁、线程模型、**内存淘汰策略**等机制，熟悉**缓存击穿**、**穿透**、**雪崩**概念
- **计算机网络：**熟悉TCP、UDP、**HTTP**、**HTTPS**等网络协议，掌握**TCP三次握手**、四次挥手、流量控制等机制
- **操作系统：**熟悉**进程**、线程、虚拟内存、**I/O多路复**用等，掌握进程间通信和多线程同步技术
- **AI Conding工具：**熟练运用如**Cursor**、通义灵码、**ChatGPT**、Claude等AI开发大模型工具
- **测试:**了解测试理论和测试用例设计方法，了解功能测试，接口测试，性能测试
- **框架:**熟悉使用常见测试框架和 Java 框架，如 Selenium,Appuim,NGTest
- 能够编写自动化测试用例用于结构测试，根据测试用力和技术文档编写测试用例

实习经历

美的集团　　Java开发实习生　　2024-11 ~ 2025-04

实习描述：参与开发数据供应链部门**APS系统（高级计划与排程系统）**，与美的集团**iPass**（集成化生产管理平台）、以及MES（制造执行系统）深度集成，实现**供应链计划→生产排程→车间执行**全链路数据协同

工作职责：

- 主要参与项目核心链路开发、负责如工单下达、工艺路线管理、车间排产全链路模块开发、完成核心代码编写工作
- 协助推进接口性能优化工作，参与技术方案评审并针对线程安全问题提出优化建议

主要工作成果：

- **多线程异步任务开发：**基于 CompletableFuture结合分页机制（PageSize=1000）实现数据拉取的并发处理，实现了**单批次1200条数据平均处理时间从7s优化至1.8秒。**同时通过CountDownLatch控制并发+Spring编程式事务手动提
- 交解决多线程事务下的部分事务回滚问题，保证事务的原子性。
- **日志记录模块开发**：参与优化高并发场景下的日志记录模块，通过 **ThreadLocal**确保多线程环境下日志记录的线程安全性，支持（Debug、Info、Error）不同日志级别的**动态切换与记录**，确保日志记录的线程安全性与高性能
- **优化慢SQL：**通过**慢查询日志**和 EXPLAIN 分析，针对生产计划的执行依赖于任务的优先级排序，建立时间和任务优先级的**联合索引**，消除了file sort的影响，解决了需要频繁进行SQL查询的性能问题，**查询时间从秒级优化到毫秒级**

项目经验

AsyncScheduler（异步调度框架）

项目背景：学校实验室的医学AI训练场景需要对收集到的图像进行数据采集、清洗、特征提取、分布式存储等多个步骤。为了提高开发效率，我抽象为多个异步任务并开发了一个基于Java的多阶段异步任务框架。

个人职责：

- **负责架构设计**：**采用生产者-消费者模式**。整体框架分为**Flow Server（服务层）**和 **Worker（执行层）**。Flow Server层通过web接口向外部提供主要服务，包括查询任务、创建任务、占据任务等。Worker层提供HTTP服务。主要接口有创建任务、拉取任务、轮询任务状态等。Worker层负责消费任务。
- **数据库表设计：设计主要的三张数据库表**：任务信息表、配置表、位置表。方便任务快速注册和进行任务管理，**实现低耦合**
- **任务调度设计：**支持**按相对优先级来调度任务**。综合创建时间、更新时间、重试间隔（**采用渐进式间隔重试策略**）进行相对优先级排序
- **服务治理设计：**服务治理通过轮询的方式来发现超时任务并重置其状态，支持通过轮询的方式来判断是否达到分表的阈值并实现分表逻辑
- **架构优化设计：**多机竞争**由Mysql行级锁优化为Redis分布式锁**，下阶段考虑引入MQ，将任务拉取和执行解耦交给MQ

技术难点：

- **任务排序规则设置：**框架抽象出了一个 **order_time** 排序字段来对任务进行排序，受到**任务创建时间**（基础排序）、**任务修改时间**、任务优先级、**任务失败次数的影响**。实现逻辑统一并解决了排序规则和多个字段耦合的问题
- **分表方案设计：实现**基于记录数量进行分表的方案：任务治理服务会定时检查任务中的记录数量，超过阈值之后触发分表。此时新的任务创建中新表中，但是仍然从旧表中调度任务、直到旧表任务调度完成。这是我们团队对于架构设计的前瞻性的一个设计思想
- **多机竞争方案及优化问题：**多个Worker去拉取任务容易拉到同一批任务。一开始这里在Worker侧引入Redis分布式锁来解决，任务冲突率解决90%，但Worker拉取和执行任务偶尔可能会CPU飙高至80%左右的问题。考虑**引入MQ**进行水平扩展效果