进阶训练营——课程导论

邓明

课程目标

上:能够胜任技术专家、架构师的岗位;

中:掌握了通往技术专家、架构师的路径;

下:换一个更加好的工作;

课程会如何使用 AI ？

思路启发者

在灵感枯竭或者无从下手的时候，让 AI 来寻找思路

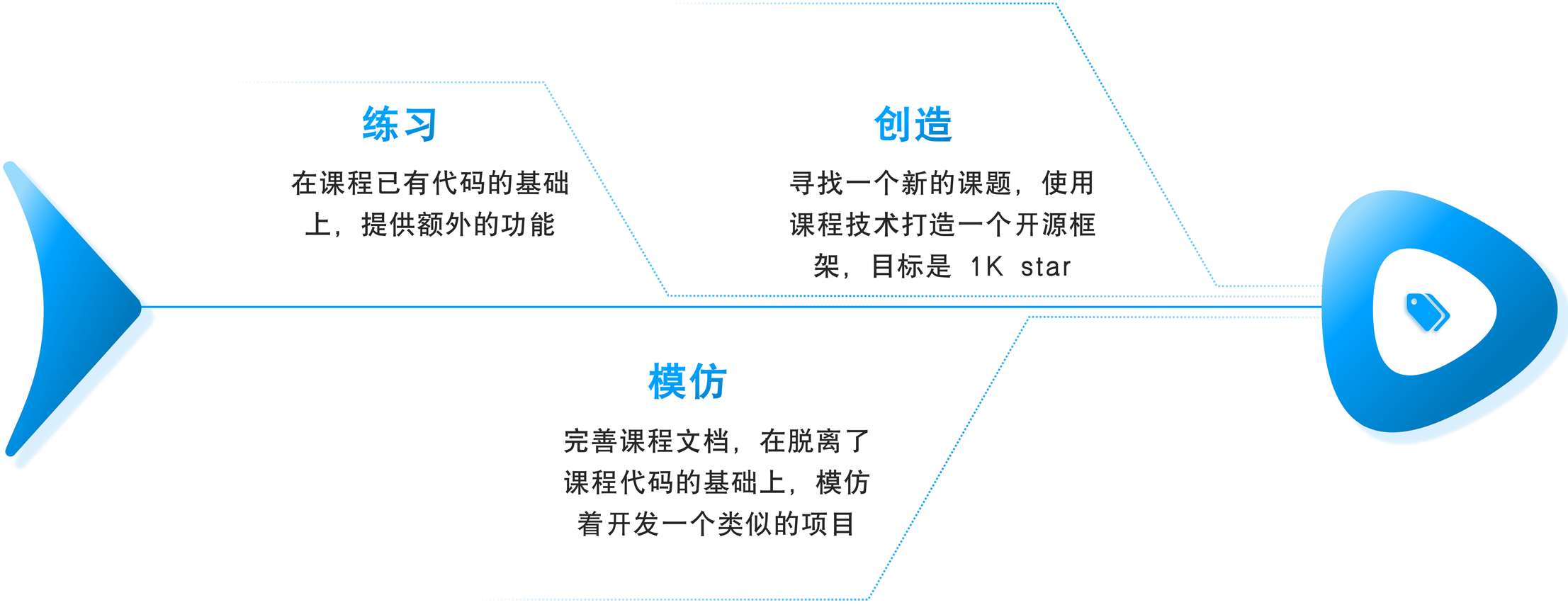
王牌参谋

针对问题进行分析
提出具体的建议

百晓生

一个万能的、无所不知的资料收集者

如何学习本课程


练习
在课程已有代码的基础上，提供额外的功能
创造
寻找一个新的课题，使用课程技术打造一个开源框架，目标是 1K star
模仿
完善课程文档，在脱离了课程代码的基础上，模仿着开发一个类似的项目

课程核心

高并发	会详细分析各种支撑高并发的技巧，你在实践中按需选用
高可用	设计各种高可用措施的交钱、套路
大数据	如何分库分表，如何在大数据的情况下保证高并发、高可用
数据一致性	深入阐述各个方面的并发更新和部分失败引发的问题，以及对应的解决思路

高可用的整体套路

服务自身治理和容错

第三方中间件的治理和容错

第三方服务治理和容错

考虑到第三方崩溃、性能下降后，系统如何保持稳定运行

第三方中间件治理和容错

考虑到第三方中间件崩溃、性能下降、网络不通，如何确保业务还是能正常运作

服务治理方案

不管你是出去面试，还是你真的要在实践中搞服务治理，你需要从这些方面制定的一个服务治理方案。

- 服务注册与发现：主要是容错，也就是考虑在注册中心崩溃的情况下保证服务依旧可用；
- 熔断/限流/降级：常规的方法可以用在实践中，但是用在面试中就不够分量；
- 隔离/分组/路由：针对技术场景、业务特征设计隔离机制
- 重试策略：不要只会简单的传统的指数退避，在公司刷不了 KPI，面试中没优势；

所有的措施、手段都有非常明显的套路，在课程里面我就是要教会你们各种套路。

当然，过于传统的内容我会大概提一下，但是会认为你们多少都知道，例如说令牌桶漏桶啥的。

第三方服务容错

也是具有非常明显的套路，三板斧的问题：

- 通过多种机制判定服务是否有问题：
  - 心跳
  - 响应时间监控：要综合考虑绝对值，以及相应的增长率
  - 错误率：可以对错误进一步细分，结合绝对比率和增长率来判定；
- 第三方服务崩溃之后：
  - 切换另外一个第三方
  - 停止发送请求：等恢复之后再尝试
- 恢复策略：
  - 灰度问题

第三方中间件容错

也就是综合考虑你的依赖的关键第三方中间件如 MySQL 之类的崩溃了怎么办。也是有非常明显的套路的，你的解决方案其实没啥选择：

- MySQL 崩溃：
  - 依赖 Redis 等缓存；
  - 数据转储到消息队列、本地文件；
- Redis 崩溃：
  - MySQL + 限流顶住
  - 切换 Redis 集群
  - 本地缓存顶住
- 消息队列崩溃：
  - 切换备用集群
  - 转储到别的地方
- ...

高并发的整体套路

高并发也是有套路的：

- 分而治之：也就是无状态类的服务，就是加机器就能解决。前提是类似入口 Nginx 这种能撑住，当然 Nginx 这种也可以搞多个接入点；
- Redis 与缓存方案：基本上高并发的读写，严重依赖于 Redis 这种缓存；
- 使用异步：可以借助消息队列，也可以直接转储到数据库；
- 使用批量：批量操作也能显著提高系统性能；
- 网络和传输优化：比如说优化应用层协议等；
- 环境调优：主要是硬件、软件层面上的调优；

其余例如说熔断、限流、降级等并不是真的支撑高并发，而是说防止高并发冲垮系统，并不是真的能正常处理那么高的并发请求。

制定特殊的缓存方案

大部分人的缓存方案都毫无特色，在实践中不是不能用，而是在答辩的时候、面试的时候没有优势。

- 如何提高缓存命中率？
- 如何动态计算过期时间？
- 如何定制淘汰策略？
- 如何解决缓存一致性问题？
  - 并发更新怎么解决？
  - 部分失败怎么解决？
- 如何应付 Redis 崩溃？
- 如何支撑超高并发？
  - 分 key 等……

大厂高并发 —— 法拉利贴膜

大部分大厂搞的高并发解决方案，我愿称之为法拉利贴膜：

- 法拉利：Kafka 和 Redis 等能撑住极高并发的中间件；
- 贴膜：大厂员工在这上面搞的一些操作；

而后他们就宣称自己解决了高并发的问题。就仿佛，你给法拉利贴了一个膜，然后你说自己最高时速 350km/h。

大数据

本课程主要针对的是分库分表，而不是大数据平台建设。分库分表讨论的也就是几个点而已：

- 分库分表方案：即按照什么分，主要是考虑解决典型查询场景；
- 主键：如何生成主键；
- 容量规划：分多少库，分多少表，怎么算；
- 跨表查询：非典型查询的解决方案；
- 分布式事务：跨库事务可以认为是分布式事务的一种，所以都要考虑解决；
- 数据迁移：在单表拆分分库分表的时候要考虑数据迁移的问题；

数据一致性

所有的数据一致性问题，都归结为两个：

并发更新

- 同一节点多个线程更新同一个数据
- 分布式环境下不同节点更新同一个数据

部分失败

- 更新 MySQL 之后更新 Redis 失败了
- 订单落库成功了，但是消息发送失败

结论：分布式环境下，强一致性是想 P 吃。

制作课程的局限性

- 时长：4 周 * 5 = 20 小时讲清楚一个大项目，是一个非常大的挑战，所以我在课堂里面只讲最核心、最重要的内容。部分省略的细节，如果你无法理解，可以提问，也可以跟我反馈；
- 知识毒药：我在课程中会尽力从一个经验不足、未操刀过大项目的研发角度出发，设计内容。但是我本身经验比较充足，所以容易出现我认为“这个东西一眼就能看穿”，但是你觉得“这是怎么想到的”的情况，记得跟我反馈；
- 未曾讲解的代码：有一些代码我觉得会很容易写出来，但是你同样可能认为完全看不懂，那么同样可以跟我说；
- 经验不足：这种高级的课程我也是第一次制作，经验不足，难免有缺漏；

正统方法论 VS 我的歪路子

整个课程会涉及到非常多的软件工程方法论、系统设计方法论。那么课程会：

- 会涉及这些传统的、正统的方法论：
  - DeepSeek 对这些方法论了然于胸，你学不学已经没那么关键了；
- 但是以我在实践中采用的方法论为核心，相比传统：
  - 简化：会去除掉很多防呆防甲方防耍赖的相关内容；
  - 变种：引入一些额外的措施、改变方法论的落地策略；

整体来说，我会认为我的歪路子更加适合当下互联网、更加适合当下 AI 时代特点，并且在效率和质量之间取得一个平衡。

课程迭代

我会针对反馈持续补充视频。

小福利：模拟面试

为了保证你出去拿这些项目面试能够面过去，我会在第一个项目完成之后，开启模拟面试的活动。

活动形式：

- 报名参加模拟面试
- 提供你的简历，而且项目经历必须是课程项目，其余则随意
- 公开直播模拟面试，我会扮演一个苛刻挑剔的面试官来捶打你的项目经历
- 其余同学围观

THANKS