面试增强计划
之
面试技巧

大明

给不会面试的人一个指引
给会面试的人一个参考

主要内容

- 准备简历
- 准备项目
- 设计面试方案
  - 猜测面试官的问题
  - 话术模板
  - 面试技巧
- 面试 checklist

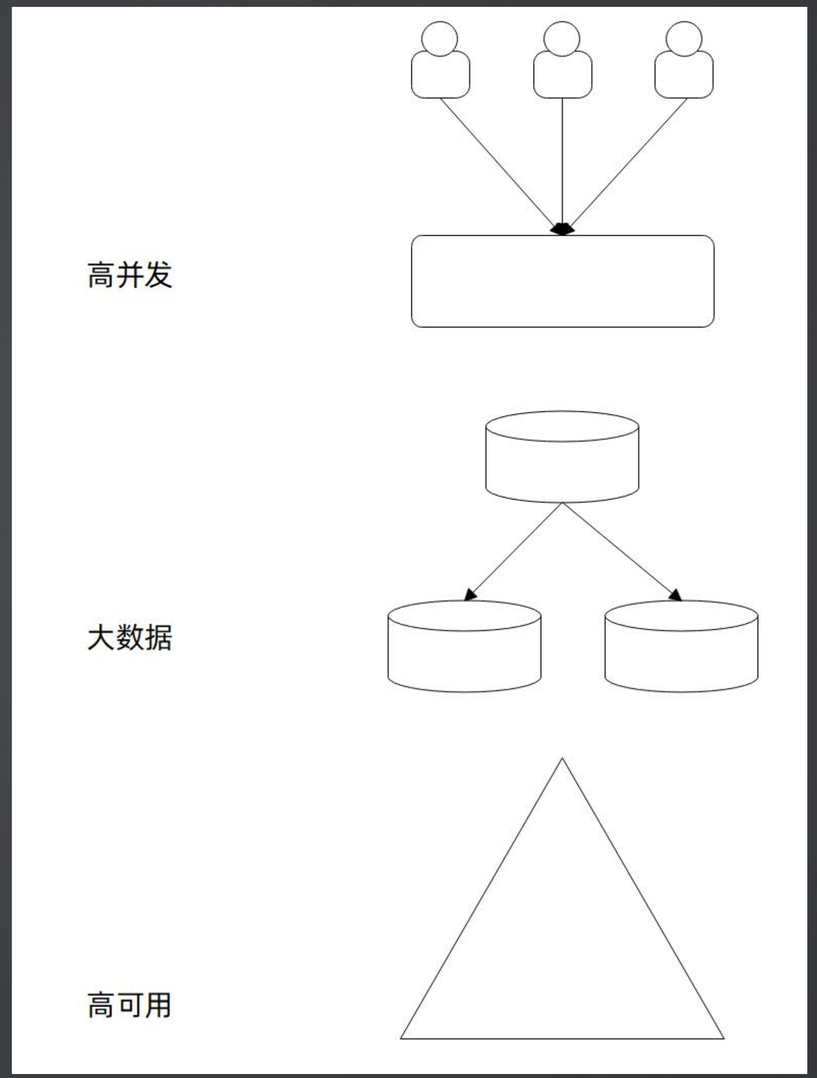
准备简历

准备简历——互联网方向什么简历有吸引力

业务上：相似性。即你做过类似的东西，比如说你面支付岗位，那么你有支付系统的背景；比如说你面订单岗位，那么你有订单系统的背景...

技术上：高并发（高性能）、高可用。

- 高并发：高并发一般和性能优化结合，非常显眼；
- 高可用：高可用强调的很难出问题，即便出问题了损失也最小，恢复也最快：
  - 中间件高可用：也就是你要开启类似于 MySQL 主从、Redis Cluster 之类的功能
  - 应用治理：服务治理，或者单体应用其实也可以使用类似的技术来治理
  - 容错：不管是业务层面上的容错还是技术层面上的容错

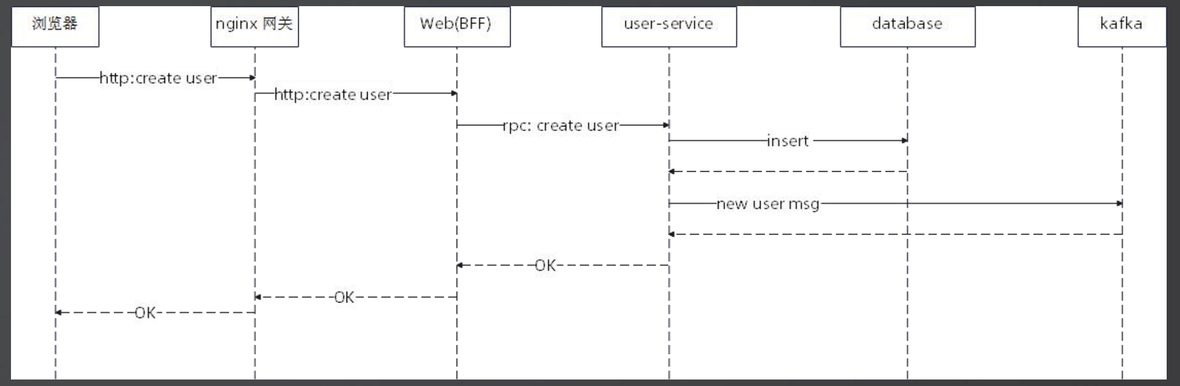


高并发——性能优化非常重要

最好刷的跟高并发相关的就是性能优化。简历一定要带关键字性能优化。

简历上可以提及的跟性能优化的点可以是：

- 业务相关：比如说优化不合理的业务流程
- 引入缓存：比如说引入Redis，或者引入本地缓存-Redis双重机制，或者缓存预热等
- 数据库相关优化：数据库本身优化、锁优化和 SQL 优化，乐观锁优化
- 代码层面优化：Go内存优化（逃逸分析、对象池）、Go并发优化（锁优化，原子操作，goroutine泄露，Go协程池）
- 异步：引入消息队列
- 架构层面优化：引入任何一个新的中间件，都可以看做是优化，比如说引入 NoSQL
- 其它常见的：布隆过滤器，bit array 等



性能优化是一个全方位的、全链路的事情。也就是说仔细思思考自己做的项目，从前端一直到后端数据库，你做了一些什么事情来优化性能

中小型企业的优化方案——缓存优化

如果你们公司没有引入 Redis，那么你就可以考虑说引入 Redis/memcache 等。除此以外还有一些可行的简单的点：

- 缓存预热：比如说你们公司缓存很关键，那么你可以说在应用启动的时候提前加载缓存；或者说利用灰度发布来做预热；
- 缓存永不过期：有一些缓存你是可以考虑永不过期的，比如说每天的排行榜；内容生产的草稿库和线上库分离；
- 尝试引入 Canal 来落地 CDC 方案：利用阿里开源的 Canal 来监听 binlog，然后去更新缓存数据
- 双层缓存：本地缓存 + Redis 缓存，你需要解释清楚怎么搞一致性的问题。在性能极其严苛的场景下，是只能大部分走本地缓存，少部分走 Redis 缓存，极少部分真的发起了数据库查询

这四个方案都挺好落地的（只要你在公司处于核心研发地位），所以你们可以实验一下，不然面试聊到细节你可能答不出来。

中小型企业的优化方案 ——SQL优化

对于很多中小型企业来说，性能问题都是出现在数据库查询上，基本上不外乎写了复杂的查询，索引没建好。一般来说，都是早期数据量不大的时候随便写的 SQL，在后面业务有一定的发展之后就出现性能问题了。

SQL 优化你要注意掌握：

- explain 的用法，各个字段解读
- 索引相关：包括底层，以及怎么设计和优化索引

要准备一个 SQL 优化的案例！

中小型企业的优化方案 ——代码优化

Go 代码性能优化就两个：内存分配优化和并发优化，其余不值一提。

- Go内存分配优化：核心就是内存逃逸和对象池。
  - 内存逃逸优化：其实属于那种听上去逼格很高，但是实际上效果非常有限的。比如说业务SQL优化一下，几十几百毫秒省出来了，但是Go内存分配优化来优化区，可能也就是优化了1毫秒。一般是使用 go build -gcflags '-m' 命令来看哪里发生了逃逸，然后尝试优化掉；
  - 对象池：这个最简单，我们在 eorm 里面就用了buffer pool，你可以理解为一种特殊形态的对象池。一般来说，如果你有什么接口是要处理比较大批量数据的，可以考虑使用这个方案

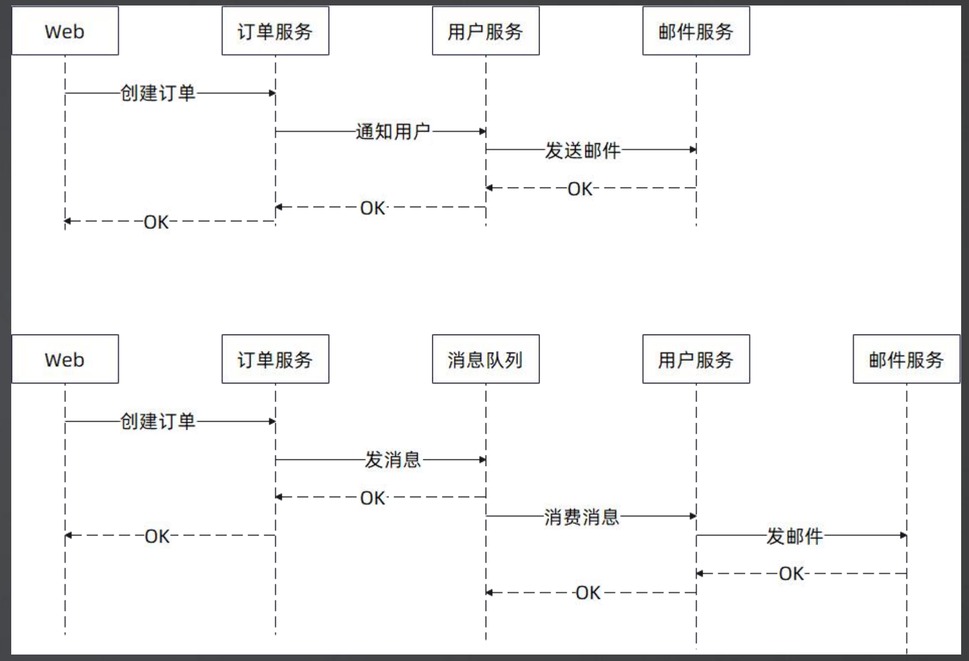
- 并发优化：主要思路有有锁改无锁；写锁改读写锁；原子操作（CAS也可以看做是乐观锁）；全局锁。并发优化这个在业务开发里面比较少用

一般我建议是用过对象池，知道内存逃逸大概怎么定位就可以在建立里面写Go程序性能优化了。
并发优化都是看场景的，不强求。

中小型企业的优化方案——异步

一般来说，对于一些复杂业务，比如说原本你是同步调用的，你可以改为丢消息到消息队列，然后消费者去消费。

比如说常见的创建了订单之后要同步调用发通知给用户，那么这个过程就可以通过引入消息队列，在原本的创建订单流程里面就不需要发起同步调用了。



中小型企业的优化方案 ——架构层面优化

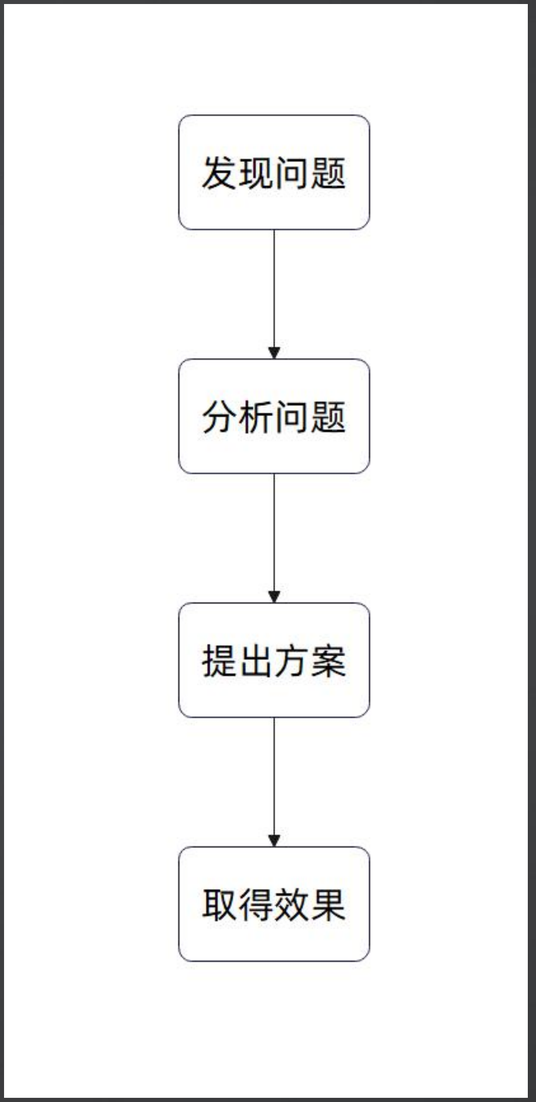
这种优化范围就很广泛，稍微有点常见的：

- 引入 Redis 或者消息队列：前面讲过
- 数据库引入读写分离：这个在数据规模增长到一定地步之后是需要考虑的
- 引入 NoSQL：比如说你们经常存储的 json 字段，可以考虑引入 NoSQL 的东西，比如说 mongo DB
- 引入 CDN 处理静态资源
- 引入 nginx 反向代理
- 引入微服务网关
- ...

核心就是你引入了一个组件，那么你要考虑清楚为什么引入，以及引入带来的效果

性能优化——每一个案例都要准备好四步

- 发现问题：比如说，你怎么知道你的某个接口性能很差？这方面一般是通过监控和告警来发现的
- 分析问题：比如说你发现接口性能差是因为一个SQL写得有问题，那么你是怎么分析这个SQL 的；
- 提出方案：要记住，不能上来就是你怎么优化的。你要先说你有什么候选方案，然后你最终挑了一个
- 取得效果：你可以说你用了比较多手段，前几个效果都不太好，最后一个效果最好。然后优化前后的具体性能数字



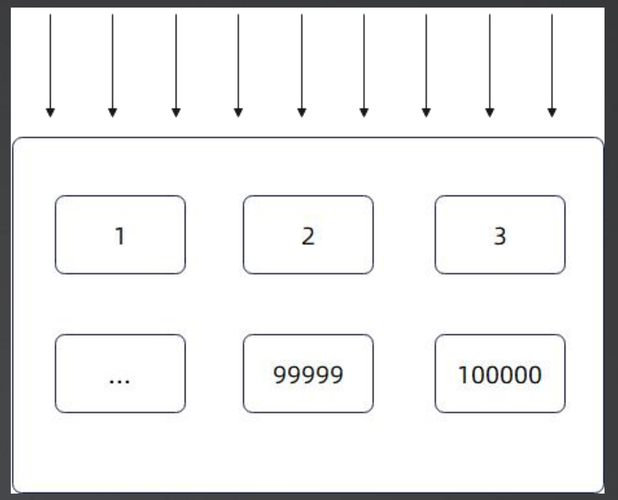
高并发——不是只有百万QPS才是高并发

高并发三个字非常具有欺骗性。假如说有一个系统QPS有一千万，但是有十万台机器在支撑着，这能算是高并发吗？其实不太算，因为单机才一百QPS，毛毛雨。

但是如果我这个系统 QPS 有一千万，都要经过同一个网关，或者数据都存在同一个数据库，那么能算高并发吗？这就是真正的高并发！

所以高并发的核心是你有一些节点，没办法通过水平扩展的方式来降低并发。

如右图，真的称得上是网关和数据库，中间的微服务节点，因为节点数量太多，所以实际上每个节点的并发都不高。



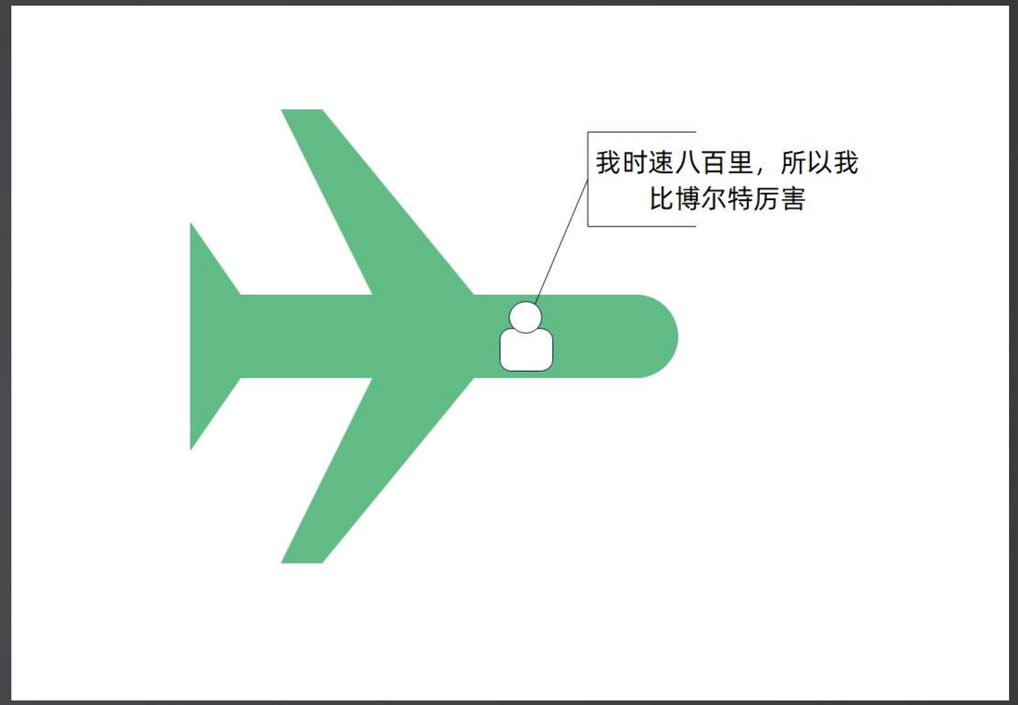
高并发——单机高并发

因为大部分同学没有工作在大规模集群上的经验，所以基本上没有什么大规模分布式集群的高并发经验。

不是我针对谁，绝大多数大厂出来的人觉得自己有高并发经验也是在瞎扯，Redis 能撑住 10W QPS，然后你利用 Redis 搞了一个能撑住 9W QPS 的系统就觉得自己有高并发经验了，这不是瞎扯吗？

你们的目标就是：我有单机高并发的经验。

所以你要尝试优化你的应用的性能，达到极限。至于多高算高，这个真的和业务相关的。我有一些复杂的业务，单机五百我已经觉得我尽力了；有一些简单的增删改查项目单机能到 3W，不过都是靠 Redis。



真正的高并发，是研发出 Redis 或者 Kafka 这种中间件的高并发。业务上组合使用，说实在的，只能算是充分发挥了这些中间件的性能。所以你坐飞机时速八百公里，不是你跑得快，是飞机造得好

高可用 —— 中间件高可用方案

要注意以下：

- Redis 如果是单机，你就要有想法把它改造为Sentinel 或者 Cluster 模式
- 数据库如果是单机，你就要考虑主从结构和分库分表
- 消息队列使用集群模式
- ... 其它注意参考对应的高可用方案

我们都不是开发这种中间件的人，所以只要学会用这些高可用方案就可以了

高可用 —— 应用治理

做好以下几点：

- 超时控制-重试-幂等：发起了 HTTP 调用，RPC调用，尤其是和第三方打交道（如微信之类的）
- 负载均衡：如果你业务有一些特性，那么负载均衡可能不一样，但是绝大多数都轮询打天下
- 熔断、限流、降级：Web 本身，关键路径上的关键服务
- 隔离：比如说根据业务价值来进行分组，保证 VIP 用户不受影响等；JAVA 之类的线程池隔离等；
- 监控和告警：logging, tracing, metrics 和告警一个不能少，高端方案是利用这些监控数据搞自动化治理；

你们如果现在没做，那么就在面试前准备做一做。或者至少你要准备一个方案，面试的时候就说你准备这么改造。

实际上，如果你入职的时候就有意识地准备下一次跳槽， 那么工作两三年你肯定能把这些都做好。

高可用 —— 容错

你要考虑好以下几点：

- 任何一个第三方中间件崩溃了你的系统要保持运作正常：例如说 Redis 崩溃了容易引起 MySQL 过载，所以你要保护好你的系统。注意，可以是有损的保护
- 你业务的任何一个环节失败了怎么办：典型的就是一致性问题，比如说写缓存成功了，但是写数据库失败了；比如说调用支付接口成功了，处理回调失败了...
- 快速恢复：无可奈何的情况下，系统就是崩溃了，那也要有方案能够快速地、自动地恢复

简历其它关键字

在简历里面提到热点主题、热点技术的关键字。记住，这些关键字就是腐肉，吸引面试官那只苍蝇

- 关键字：高并发、大数据、服务治理、分库分表、分布式事务、分布式锁、秒杀……

除此以外，你在简历没有提，但是也要防着面试官问你这种热点主题，做好准备：

- 每一个主题相关的八股文背好：例如分库分表里面的分库分表的常见策略、主键等问题。
- 每一个技术主题，要准备一个案例。案例可以是你做的，也可以是你同事做的，也可以是你公司里的，也可以是网上看到的。例如分库分表，你就至少要背下来一个方案，涵盖技术选型到落地的整个方案。

简历写法 —— 总结（中小型企业适用版）

- 如果你做了任何的性能优化，那么在简历和你做性能优化的那个项目里面加上性能优化四个字
- 即便你们公司没有高并发，但是你做了性能测试，发现在4核8G这种机器上能有大几百的QPS，那么也可以说是高性能
- 如果你公司有个几百万上千万用户，那么你就加上高并发三个字
- 如果你做了全方位的治理（从前端到数据库），那么就加上高可用三个字
- 如果你做的东西可以跟任何的热点主题，热点趋势相关，加上对应的关键字

你的目标就是：我是一个熟练掌握构建大规模高并发、高可用、大数据量分布式系统的人才。

如果是业务，那就是：我是一个深挖支付（换成你对应的领域）领域有十年经验的领域专家。

注意，不管是高并发还是高可用，你都是有一整套方案的！

在自我介绍、项目介绍的时候一定要记得说这些关键字！

技能/个人优势

- 重要的在前面，不重要的在后面。中小型企业跳槽适用版（建议版，1-3年工作经验可参考）：
  - 王炸：精通构建大规模高性能高可用的分布式系统
  - 语言相关：熟练掌握 Go，Go并发编程、网络编程，深入了解 GO 运行时原理；熟练掌握 Gin，GORM 等框架使用，深入了解其原理；
  - 数据库：熟悉 MySQL，精通 SQL 优化；熟悉MySQL读写分离；了解分库分表
  - 缓存：熟练使用 Redis，熟悉 Redis 高性能和高可用方案；掌握缓存模式，熟练解决缓存穿透、雪崩、击穿等问题；
  - 消息队列：熟练使用 Kafka，掌握 Kafka高性能和高可用方案，熟练解决消息幂等、有序等问题
  - 方法论：熟练掌握DDD，TDD，熟练运用设计模式
  - 其它中间件：Zookeeper, Nacos, Etcd ... 全部罗列一遍
  - 其余技能：前端技能，linux 命令行，git

准备项目

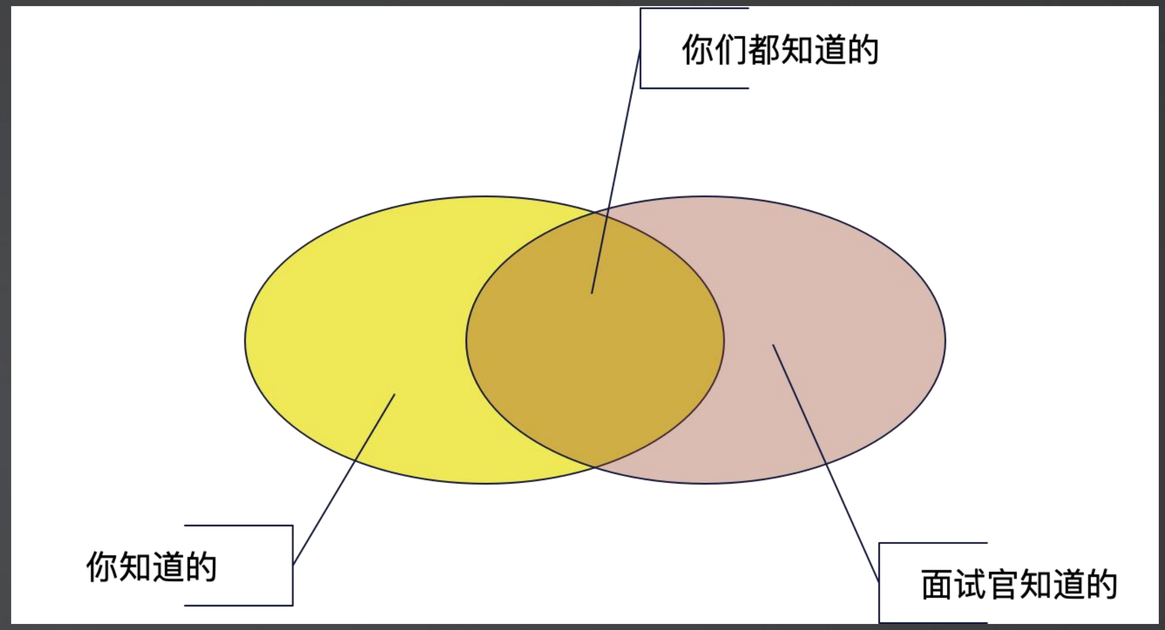
项目经验——选取项目

- 你主导的核心项目：这一部分是面试的重点
- 公司的其他项目：
  - 解释特定的技术主题：例如你虽然没有实际解决过高并发问题，但是你们公司有类似的系统，那么你就深入去理解清楚你们公司内部的设计。类似的主题还有大数据、分库分表、服务治理。
  - 用于举例子说明某些中间件的用法：例如说明 Kafka、Redis 之类的中间件。

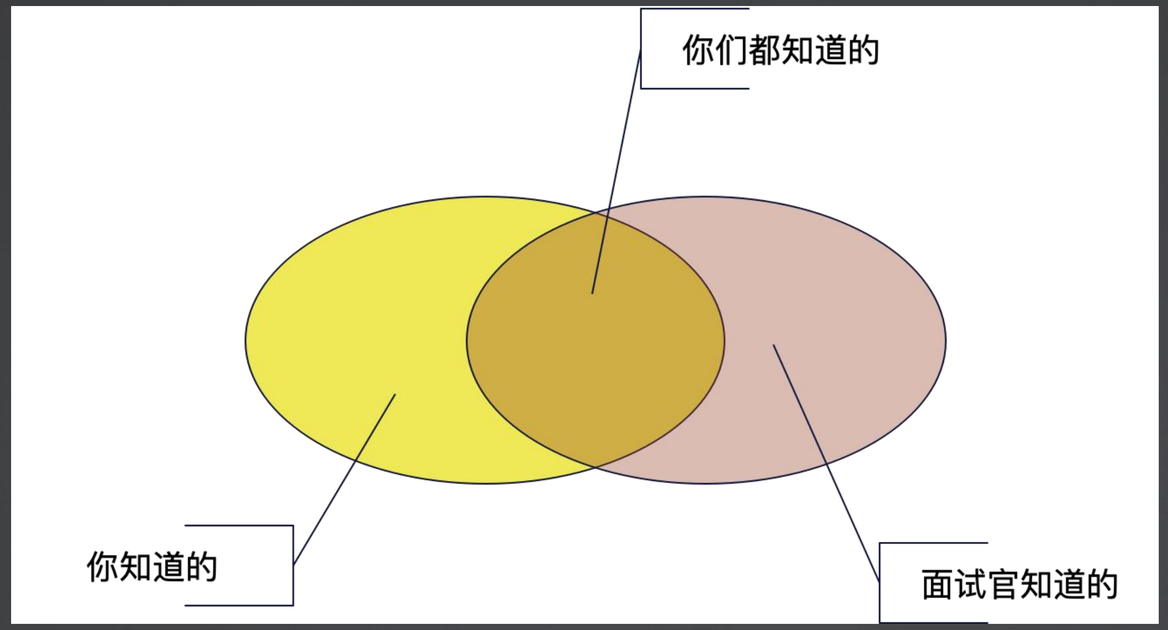
核心是：如果你对一个项目很了解，那么面试官是区分不出来你是核心，还是就贡献了一行代码。

项目经验——准备事项

- 核心困难：一句话说清楚
  - 必须是面试官能够理解的。
  - 准备多几个困难点，然后根据面试的岗位、面试官偏好选择一个困难作为自己项目的核心困难点。
- 解决方案
- 取得效果
- 总结方法论



描述项目——核心困难



面试官知道，你不知道的，问到了你就寄了。

你知道的，但是面试官不知道的，你说了也等于白说：

- 他无法理解你解决的问题有多难
- 他无法理解你的解决方案有多棒

描述项目——核心困难

如何构造合适的项目难点：

- 当前有什么技术热点你的项目就有什么难点；
- 你面试的公司需要什么样的人才，你就解决过什么样的问题；
- 传统的热点：高并发、大数据，以及衍生出来的服务治理、分库分表你都有；
- 你可以一个项目有多个难点，也可以不同项目覆盖不同难点。

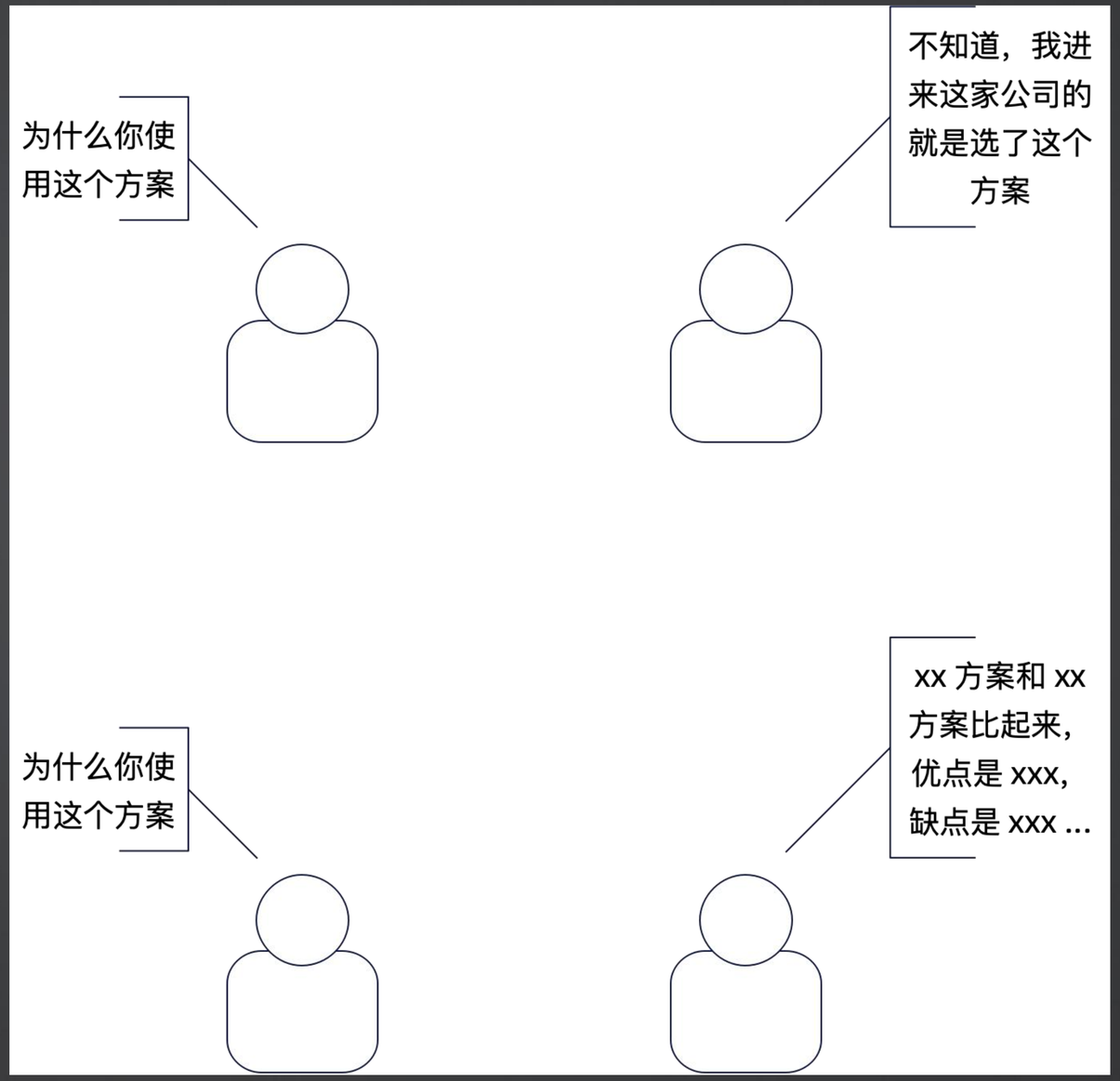
所有的这些难点，你都需要整理成文本，在面试之前背好。

描述项目——解决方案

解决方案要符合以下特征：

- 符合你的业务特征，尤其是极其针对核心困难；
- 微创新，或者中创新：即你的方案不能太死板、古老，也不能太先进；
- 解决方案一定是你经过深思熟虑之后做出来的选择。

即便技术选型确确实实不是你做的，你也得说出个一二三出来，非常忌讳上来说我不知道，这是架构师搞的。



描述项目——解决方案

你要在面试前，至少准备一个其他的可行的方案。

训练营例子：

我们训练营里面使用的是 Redis 来实现分布式锁，而实际上还可以考虑使用别的，例如 zookeeper，那么你就要强调 Redis 有什么特点、zookeeper 有什么特点，而你们公司业务特点是 XXX，所以你最终选择了 Redis。

相当于，在分布式锁这个主题之下，你要非常了解其它可行的方案在你们公司的场景下，会有什么优劣。

你不对比，怎么知道哪个好呢？

描述项目——解决方案

对你当前的项目，不管方案是不是你设计的，你都需要考虑改进点。

改进点最好是围绕当前的技术热点来描述：

- 比如说公司用的是单机 Redis，你就可以考虑说引入 Redis 集群
- 比如说公司还没分库分表，你就说数据量增长很快，要考虑引入分库分表
- 比如说公司在服务治理上还不够完善，你就说要补充什么服务治理措施
- ...

另外，所有的改进点，你都要真的有一个方案，比如说你说可以考虑优化性能，那么怎么优化、可以从哪些方面优化，不要空口白牙说可以改进！

描述项目——取得效果

取得的效果，一定是可量化的！

你要提前准备好公司内部系统的各项指标：

- 响应时间
- 错误率
- BUG 率
- 最高并发数
- 数据量
- ……

和人相关的东西是不好衡量，比如说研发效率之类的。

好量化的，就准备数据；不好量化的，就准备小故事。

项目介绍误区

- 罗列功能：具体功能点不用列，你把你做得做出彩的，最核心的简单写一下就可以。尤其是增删改查，后台管理不要写！
- 描述基操：大篇幅描述自己是核心成员，什么捋需求，做设计，搞测试，但凡你是项目核心成员，就都做这些事情，完全没有竞争力
- 长长长描述项目背景：太长不看。这应该是面试官不懂的时候你再给它解释，简历就要简洁，用一种普通人都能理解的话把项目的困难点讲清楚就可以。具体背景，比如说什么系统升级、公司并购、办公室斗争，不用写！
- 大量使用专有词汇：比如说在简历里面用你们公司的黑话行话，但凡你觉得面试官可能没听过，也不能望文生义的，都要换一种说法！

要注意，你写简历不是真的写自己做了什么，而是写面试官对你做的什么感兴趣！就仿佛讲故事，你不能是流水账，你得抑扬顿挫、起承转合。

利用课程知识来刷 Go 项目经验

你们可以通过将我部分演示的代码进一步深化，然后做成工具库，写进去简历里面，作为你们的 Go 项目。我们的这一类项目普遍是技术含量比较高的，比增删改查的项目经验会更好一点。

- 并发队列：你可以在公司内部开发一个并发队列库，或者在自己 github 上放一个项目。注意解释为什么有 channel 还要有这个并发队列
- 协程池：参考 ekit 里面的 pool.TaskPool
- 分布式锁：基于 Redis 的那个，我们在缓存里面完整演示过
- 基于内存的消息队列：在讲 channel 里面我见过，你们需要加上可观测性和容错（优雅退出、持久化到数据库等）的设计
- grpc/gin/gorm + 基于 Redis 的熔断限流和降级：将我在熔断、限流和降级里面的东西弄一份出来；
- grpc/gin/gorm + 基于 Prometheus 采集数据的熔断限流和降级：最后一次作业的那种思路，利用 prometheus 采集的数据来做服务治理（包含负载均衡，客户端治理和服务端治理）
- 缓存模式：将我们的缓存模式挪出去，比如说为 go-cache，或者 beego cache 模块加上这种模式

设计面试方案

猜测面试官可能问的问题

面试官问的问题，大部分都是有迹可循的：

- 由关键字引出来的，关键字基本上就是各种技术词汇。
- 由某个问题衍生出来的。你可能根本没有提到相关的关键字，但是面试官可能就问到。

关键字

关键字是最直接、最容易引起面试官提问的。

- 你在简历中写了某些关键字
- 你在自我介绍、项目介绍中说了某些关键字
- 你在面试过程中，为了解释一个问题而提到的另外一个关键字

- 熟悉 Mysql 数据库原理，对 Mysql 分布式存储和缓存有丰富的设计和使用经验。具备数据库设计和优化经验，熟练使用表索引、分库、分表等。有单表千万级的查询优化经验。
- 熟悉 Nosql 技术，精通 Memcached/Redis/MongoDB 等常用的 Nosql 解决方案。了解各自的优缺点以及使用场景。
- 掌握 Zookeeper/Kafka/RocketMQ/RabitMQ 等中间件的应用。

注意这一个特点，后面你要掌握利用这个特性来撰写简历，准备自我介绍、项目介绍的话术。

这些描述可能引发面试官问什么问题？

- 熟悉 Mysql 数据库原理，对 Mysql 分布式存储和缓存有丰富的设计和使用经验。具备数据库设计和优化经验，熟练使用表索引、分库、分表等。有单表千万级的查询优化经验。
- 熟悉 Nosql 技术，精通 Memcached/Redis/MongoDB 等常用的 Nosql 解决方案。了解各自的优缺点以及使用场景。
- 掌握 Zookeeper/Kafka/RocketMQ/RabitMQ 等中间件的应用。

2、 了解 go 语言

3、 熟悉分布式系统和主从服务器机制及原理，拥有相关开发经验，拥有微服务开发经验

4、 对数据结构与算法有较好的理解。

6、 熟悉 Linux 的常用指令和部署

编程语言：熟练 Python、Go 语言和 SQL，了解 C、HTML/CSS 和 JavaScript 等编程语言

开发框架：Gin、FastAPI、Flask、Django

数据库：MySQL、Redis、PostgreSQL

其他：掌握基础数据结构和算法的基本原理，掌握计算机网络、多线程和多进程的基本原理和编程

由某个问题衍生出来

你不一定非得提到了某个关键字，只要你的内容能够让面试官联想到相关的内容，他就可能问出来。

- 从相似性的角度衍生出来：也就是两个问题之间有一定的相似性。比如：
  - 你在讲 Redis 的字典结构的时候，他问你 Go/Java 的 map 结构，然后要求你做比较
  - 你在讲 Kafka 的时候，他问你 RabbitMQ/RocketMQ，然后要求你做比较
- 从相反的角度衍生出来：也就是当你说到某个点的时候，他会故意问你一个相反的点，没有提前准备的话，很容易卡主。比如：
  - 你大谈索引和 SQL 优化，他突然问你索引有什么代价
  - 当你一直讲分库分表的时候，他突然问你不分库分表可不可以
- 从递进的角度衍生出来：也就是你提到了一个比较普通的话题，但是从这个话题会引申到一个比较高级的话题。一般来说“一杆子打到底”的面试策略就是按照这个思路来的。比如：
  - 当你谈到服务注册与发现的时候，他问你注册中心崩了怎么办
  - 当你谈到原子操作的时候，他问你原子操作在计算机上是怎么实现的

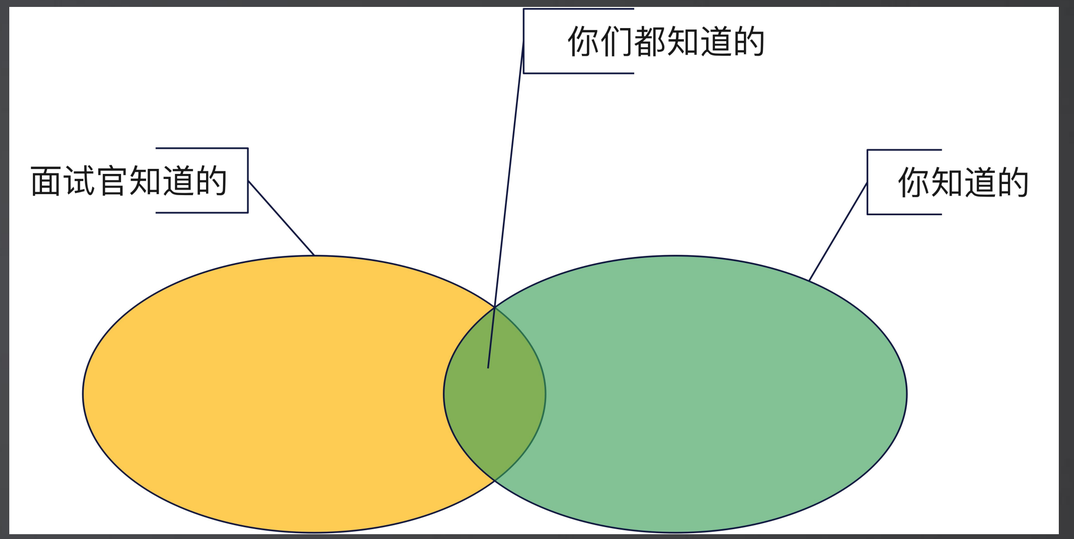
设计答案以引导面试官

既然你已经知道，你简历上的任何一个字，面试过程中说出的每一句话，都有可能让面试官追问下去。

追问有两个可能：

- 问到你知道的
- 问到你不知道的

如果你不做任何引导，那么大概率问到你不知道的。



设计答案以引导面试官

针对一些你觉得大概率面试官会问的问题，你要提前设计好答案。

你要根据相似、相反、递进三个关系来设计引导点。

比如说：

- 关键字：有意识地在回答中留下一些关键字
- 话中留一个小尾巴：比如说你基本回答了他的问题之后，本身你是可以进一步解释的，但是这个进一步，你就泛泛而谈一下就可以
- 故意对比：使用相反的例子大概描述一下

注意1，你的引导点必须大概率是面试官也知道的，如果面试官都不知道，他根本不会问你。比如说如果你在我面前设计的引导点是前端相关的内容，我直接就忽略，因为我根本不会。

注意2，你设计的答案要提前写好，不要相信自己的临场应变能力。

设计答案以引导面试官

例子：

- （回答MySQL自增主键）.....（引导点）在分库分表中，设计主键生成策略也要考虑自增的问题。（相似性，关键字）
- （回答 Java 的 map 结构）.....但是有些 map 结构的扩容就不太一样，它是渐进式扩容，比如说 Redis。（相反性，关键字 + 对比）
- （回答 MySQL 的隔离级别）.....MySQL 的隔离级别，在 InnoDB 引擎上的实现关键，是利用了 MVCC 中的 Read View 结构。（递进，关键字 + 话说一半）
- （自我介绍）.....擅长性能优化，包括 SQL 性能调优.....（直接使用关键字）

设计亮点以赢得竞争优势

什么样的才可以构成你面试中的亮点？

理论上：人无我有，人有我精。这个已经有点难做到了，因为这年头卷王太多了。

实践中：

- 微创新方案：你在传统方案的基础上改进了一点，但是这个改进要具备一定的泛用性，也就是说，完全针对你的业务的改进，面试效果并不好。
- 偏门冷僻奇诡的方案：出奇制胜，也就是你的方案是业界很少采用的，或者它本身就透着一种不正统的味道的方案。
- 综合性强的方案：面试的主题都是割裂的，如果你的方案能够综合体现多个主题，那么就有很强竞争能力。

首选微创新方案，这也就是要求你在平时工作的时候，要发挥你的想象力，不要太拘泥于传统。亮点的本质是证明你比别人强，所以能达到类似效果的都可以。

话术模板

自我介绍模板

自我介绍可以看做是一个简历的补充、详细版。自我介绍是一个很好的开始引导面试官的切入点。记住，自我介绍不是让你背一下简历。自我介绍核心是凸显你想被面试官问的项目、技术点。模板：

- 你好，我叫 XXX。（基本信息不用说，你简历都有，比如说性别这种）
- 我之前在 XXX 工作，核心职责是 XXX，主要成就是 XXX。（注意不要流水账，如果有什么公司内部的奖励，或者绩效很高，记得提一下。你这里最多介绍近期两个公司，注意关键字）
- 我也积极参与开源，主要成就是 XXX （可选，注意关键字，即便是你自己的个人项目，也可以介绍一下）
- 近期关注到贵公司在招聘 XXX，我认为我和这个岗位比较契合，相比其它候选人，我认为我的优势在于 XX （注意关键字）
- .. 后面随便补一点你觉得值得一提的其它事情

项目介绍模板

项目介绍就是围绕我们之前提过的四个步骤，一定要注意凸显自己的贡献：

项目 XXX 是我们公司的核心项目，xxx （这里补充一些业务大概的数据，如果数据很惨就不要提）。我主要负责核心模块开发（十个人有九个都是），以及解决项目的核心难点 xxx。我做了 xxx ，最终取得的成果是 xxx。也因此，我拿到了 S 绩效（或者别的奖励）

比如说：

xxx 是我们公司的核心系统，主要解决和付款、退款等问题。该项目的核心难点是在保证高性能和高可用的前提下保证数据一致性。为此我设计了一个基于 SAGA 的分布式事务框架，在使用该框架之后我们的 TPS 能够达到单机10K。和原来比起来，可用性达到9999，并且每日数据核对，不一致数据下降为原本的1%。因此，我拿到了最佳设计奖。

项目介绍模板

例子（调用第三方接口）：

XXX 系统是我们公司的核心系统，它主要解决对接第三方支付平台，如微信和支付宝等问题。它的核心难点是要保证高可用下的数据一致性（一般小体量的公司并发数都不太高）。为此我将整个模块做成了一个对外的网关，为业务方屏蔽了第三方支付平台的差异。并且在这基础上，加入了统一的服务治理、容错、鉴权等措施，可用性提高到了 9999，和原本比起来出错比率下降了90%。同时我还设计了一个支持测试的mock服务，使得开发和测试效率提高了20%。为此我绩效拿到了 S。

例子（开放平台）：

XXX 平台是我们公司的核心系统，它主要是提供公开API给合作伙伴使用。该项目的核心难点是高可用和安全性（身份认证和权限校验）问题。为此，我使用了 OAuth2 的方案，叠加了熔断、限流等全面的服务治理措施，保证了安全性和高可用。为此我绩效拿到了 S。

项目介绍模板

例子（数据库相关）：

XXX 项目是我们公司的项目，它主要负责解决 XXX 问题。它的核心困难在于随着业务增长，已有的数据库成为了性能瓶颈。为此我在公司内部推动了将这个数据库进行分库分表。我主要确定了技术选型，设计了分库分表的方案（包含分款分表键选择、容量计算），以及数据迁移和校验方案。在经过 X 个月之后我们顺利完成了分库分表，目前运作良好，瓶颈问题彻底解决，可用性和性能都大幅提升，达到了 xxx 。为此我拿到了 S 绩效。

例子（已有系统重构）：

XXX 项目是我们公司的核心项目，它主要负责解决 xxx 问题。该项目是一个历史遗留项目，性能、可扩展性和代码质量都极差。我在摸清楚整个项目之后推动了整个项目的重构。我考虑到这个系统的特性是 xxx，所以我引入了 xxx 和 xxx，以从架构层面上改善整个系统。同时我引入了 CI，静态代码检查，推行了代码规范，提高了整个系统的代码质量。也引入了可观测性工具，搭建了全面的监控和告警平台。和原来比起来，目前接入新功能需要的资源只有原本的三分之一，性能提升了 xxx，可用性提到了 xxx。我们这个项目在公司内部被点名表扬...

为什么用 X 不用 Y 模板

类似的问法还有：

- 为什么用 X 不用 Y
- 为什么用 X/你为什么用 X
- 为什么不用 X

这种核心就是交叉对比回答。一般的模板都是：

X 的优点是 xxx，Y 的优点是 xxx，而我们公司的业务特点是 xxx。相比 Y，X 更加契合我们的场景（这边你可以具体解释为什么契合，为什么不契合），所以我们选择了 X。

可以简单介绍一下 X 吗

和前一个问题差不多，但是这边侧重点在于回答一些关键字。这些关键字就是你希望面试官面你的。而后再结合一下你们公司的使用场景。

比如说：可以简单介绍一下 Redis 吗？

Redis是现在被大规模使用的缓存，它有丰富的数据结构，包括 xxx。Redis本身性能非常高，也提供了包括 Sentinel 和 Cluster 在内的高可用方案。在我们公司，Redis 主要被用于 xxx，也会被拿来做分布式锁。

类似这种中间件的介绍，注意抓住两个点：主要场景、高性能和高可用特性、你们公司的应用场景。

或者，你们刷的面经有什么特点，你就介绍什么特点

你的项目有什么难点

类似的问题：你遇到最大的问题/挑战是什么？

要点是：用面试官能理解的一句话解释清楚难点，然后讲至少两个的候选方案，再讲自己采用的方案，最后总结效果和可能改进的地方。例如：

我们这个项目的核心难点是 XXX。正常来说，在业界要解决类似的问题都可以考虑采用 xxx 或者 yyy。相比之下，xxx 方案的缺陷在于 aaa，我们并不太能用这个方案（也可以说你们试了这个方案，但是效果不太好）。所以我们选择了 yyyy。落地之后（一般来说，落地的困难都没什么好说的，总不能说你遇到了什么BUG）取得的效果是 xxx。不过目前来看，这个方案的局限性也是有的，比如说 bbbb，我已经在计划改进这一点了，比如说用 ccc来改进（随便提一嘴改进，不需要详细说）

你遇到的最大的BUG是什么

和难点类似。差别在于你要回答清楚你是怎么发现、排查分析、可选解决方案、实施以及取得效果。

一般，我建议使用性能问题的 BUG。比如说 OOM，SQL慢查询，或者 goroutine 泄露等。

面试技巧

面试技巧

- 先射箭，后画靶子
- 高屋建瓴讲方法论，脚踏实地讲落地
- 统计上讲数据，细微处讲故事
- 你问的都是我想要你问的

先射箭，后画靶子

是指，你已经提前准备好了一些话题，那么在回答一些开放性质的问题的时候，就要把话题朝着你提前准备好的素材上引导。

要知道，有一些问题你是绕不开的，面试官几乎必面的：

- 你项目有什么难点？
- 你项目有什么亮点？
- 你做这个有什么收获？
- 我觉得这个项目很普通，为什么你觉得你做得很好？
- 你在学习 GO 语言的时候，你觉得哪个最难？

所以你可以总结涉及这一类亮点/痛点/难点之类的问题，你都可以用你提前准备的素材来回答。

先射箭，后画靶子

一般的策略是用提前准备好的热点主题。例如用训练营的知识：

1) 我这个项目的难点是用好分布式锁，众所周知，分布式锁在使用的时候需要注意 xxx，为了解决这些问题，我采用了 xxx，所以目前效果还不错，同时在这些方面 xxx 上还可以考虑改进，比如说采用 xxx……

2) 我这个项目比较难的是实现一个支撑高并发的 GO 优先级队列，因为 GO 里面缺乏其它语言的 await 调用，这迫使我们必须考虑使用别的方案来协调入队出队。最开始我们设想使用 CGO 的路子，但是 xxx……

3) 我这个项目对可用性要求非常高，所以我们从负载均衡、超时控制、熔断、限流、降级、监控和告警几个方面进行了综合了服务治理……

这些热点主题肯定会面的，所以你趁着这种开放性问题引导面试到你提前准备好的内容上，既可以消耗面试时间，又可以从容不迫刷出亮点。

高屋建瓴讲方法论

面试，不能单纯就事论事，你需要适当拔高，也就是总结出一些方法论出来，这主要是为了体现你是一个擅长思考、总结的人。

训练营例子：

……(假如说你已经解释清楚了怎么去做全链路超时控制，开始总结链路元数据）。全链路超时控制可以看做是一个链路元数据的例子，与之类似的还有 A/B 测试、链路追踪、全链路压测等，核心都是依赖于在整个链路中传递元数据，而后利用这些元数据来做一些什么……

……(假如说你已经解释清楚了 Web 或者 ORM 的回调/hook/middleware，然后开始总结一般的 AOP 设计）大多数的中间件都会设计类似的接口，例如在 GORM 里面有 hook、xx 里面有 xx。它们的设计都是类似的，大体上都可以看作是洋葱模式或者责任链模式，即便没有类似机制我们也可以考虑通过装饰器来封装已有接口，达成类似的目标，比如说 xxx……

脚踏实地讲落地

面试，也不能狂吹法螺，在讲到落地的时候一定要具体，具有可行性。

在讲落地的时候要注意：有细节、有具体的步骤。

训练营例子，你在谈到如何进行分库分表的时候：

- 讲清楚多少表多少库，多少表
- 你是如何选择分库分表键（这里可以总结一般的选择分库分表键的原则）
- 你是如何做数据迁移和验证的
- ……

这中间，可以补充你遇到了什么千奇百怪的问题，并且最终是如何解决的。

注意，如果项目不是你做的，那么你就会在这一步被坑。

摆数据

摆数据要注意：

- 数据要有冲击力：比如说节省公司成本 N 千万
- 前后数据对比要强烈：比如说性能有数量级提升
- 数据在合理的范畴内：可以夸大，但是不要过分夸大

这些数据要提前准备好。比如说常见的面试题就是如何优化性能，那么你要提前准备好优化性能的方案，以及前后的性能对比。

训练营例子：

我之前尝试优化过一个性能问题。在 ORM 上，不考虑数据库本身性能的话，瓶颈主要在构造 SQL 和处理查询结果集上，构造 SQL 采用了buffer pool结合 GO 内存逃逸分析，使得内存分配减少了 50%；而在处理结果集上，用 unsafe 取代了反射，CPU 消耗和内存分配都减少了 30%.....

要有冲击力！

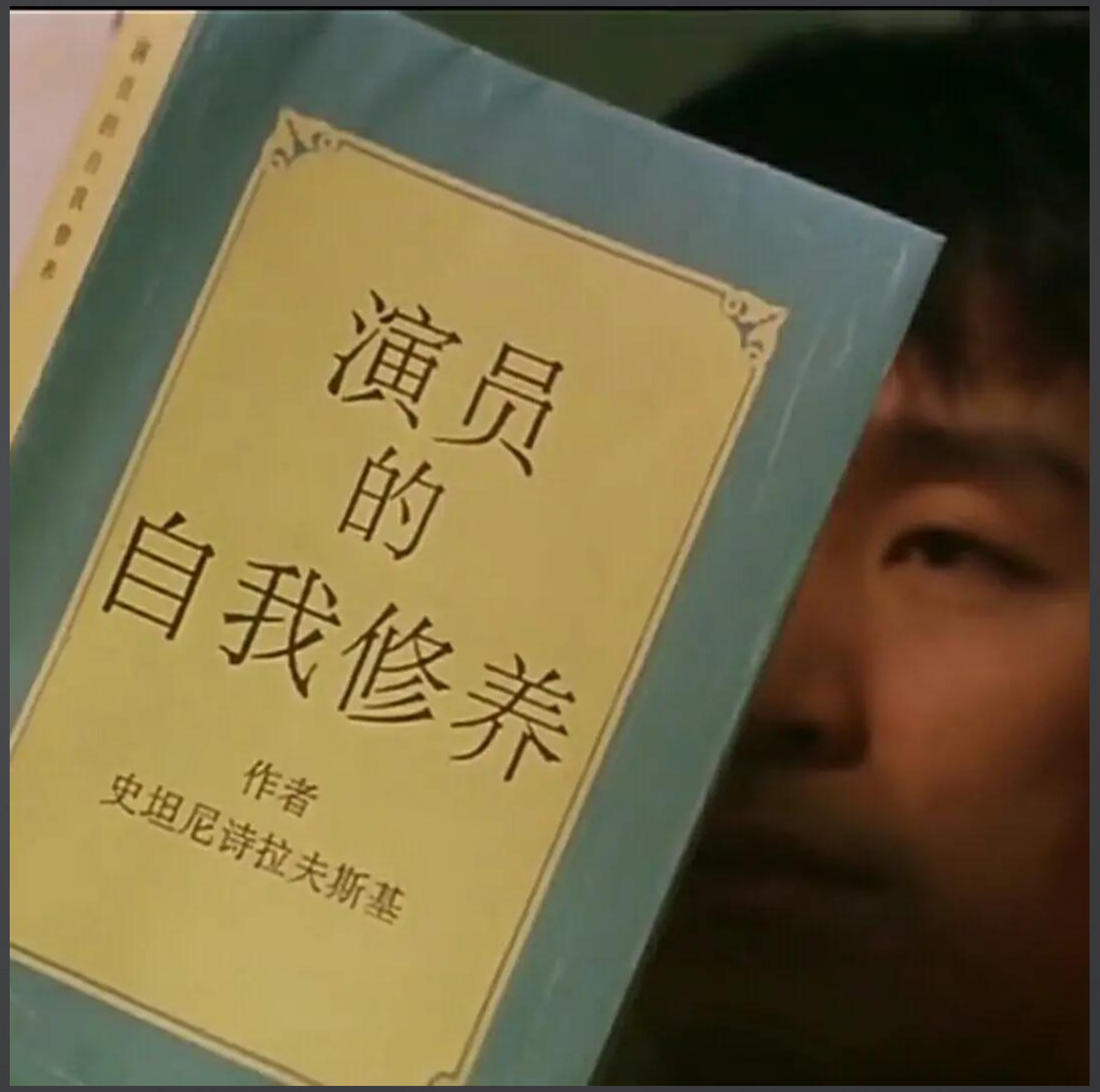
讲故事

注意一个核心点：相比冷冰冰的数字，人更加喜欢听故事。

当你在准备小故事的时候，一定要考虑是否能够引起面试官的共鸣。

例如说在讲到你改造已有系统的时候，你可以说在改造前，天天出 BUG，然后被老板骂得狗血淋头，然后还背了一个 P0（C）。在经过你的改造后，现在已经很稳定了，都可以请长假出去玩而不必担心系统崩溃……

描述要生动形象，要配合表情和肢体语言，并且多准备几个小故事，看情况使用不同的小故事。



积极引导 + 暗示

- 虚张声势：对于一个主题如果你不太熟悉，要注意虚张声势，假装自己在这个地方非常了解，以至于面试官下意识觉得不需要在这个主题上问你问题。慎用，因为你可能弄巧成拙。
- 能而示之不能：如果一个主题你很了解，那么你可以假装自己不太熟悉，比如说面试的时候含糊其辞，那么面试官就会逮着你往死里问，这就是正中下怀了。
- 知己知彼：如果你有内推，那么要让那个内推的人帮你深入打听你这家公司（最好是细化到组）主要做什么，已经做成的东西，正在做的东西，难点...如果他们有文档，你可以让内推的人给你看看，了解他们的设计方案（尤其是BUG）。而后在面试的时候你假装不经意地提起类似的话题，那么面试官肯定会追问。
- 先发制人：前面提到的先射箭后画靶子，也就是先发制人的体现，即你主动提起某个话题。
- 明修栈道暗度陈仓：在回答一个面试题的时候，特意在答案里面提到另外一些技术方面的关键字（语气可以稍微强调一下），如果面试官对这些技术感兴趣，他就会追着你问。

注意：你说出来的每一个技术词汇，都可能成为面试官问下去的导火索

引导总结

关键字是那块腐肉，面试官是那只苍蝇

面试中总结

立人设、重引导

Q & A

THANKS