技术设计101

zhengche@

自我介绍

- Math, CS, Cloud, AI
- 非典型程序员
- 编程语言爱好者

提纲

授之以鱼不如授之以渔不如授之以为何渔

- 软件工程「概述」
- 如何写设计文档&数据模型设计
- 架构设计
- Q&A

软件工程「概述」

软件工程师的职责

软件工程师的职责

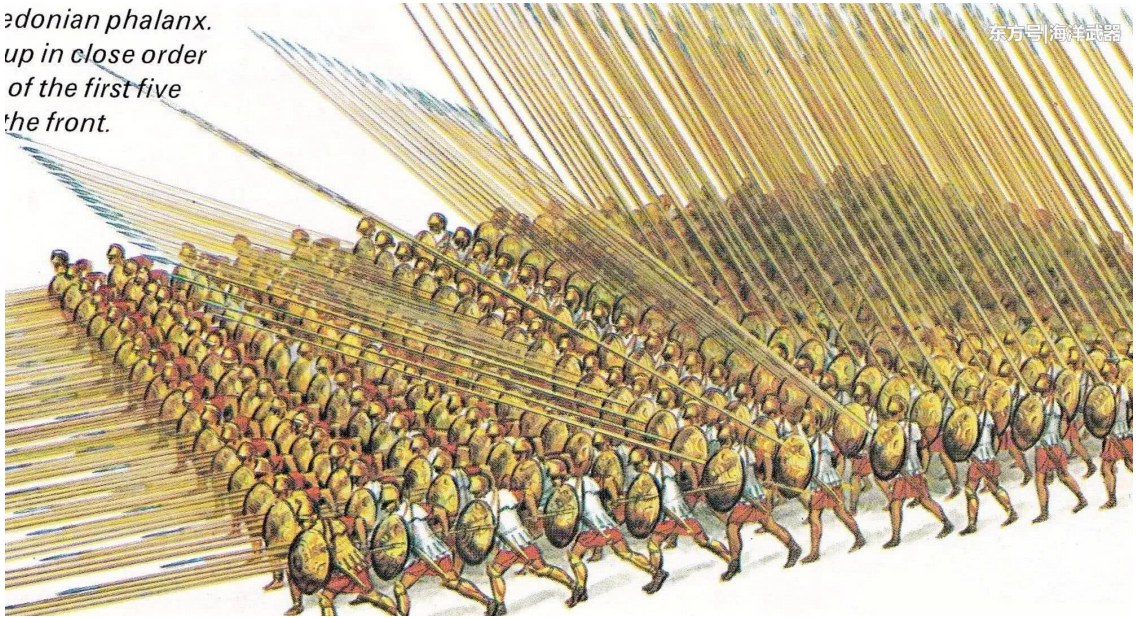
- 工程师
- **解决问题**
- 不要被工具限制想象力

软件工程为什么难

- 《A philosophy of software design》
- 庞杂

软件工程为什么难

- 依赖关系与不确定性



软件工程怎么解

- 通过拆分和重组解耦依赖关系
  - 架构拆分，微服务拆分，代码分层，依赖注入，对象设计
- 通过各类白盒和黑盒的工具降低不确定性
  - 白盒：技术设计，框架，中间件，PaaS
  - 黑盒：测试，监控，防腐层

下课！

写在前面

- 技术是会更新的，职责是会变大的，但方法是相似的
- 不要被工具定义了能力
- 端到端的解决问题

如何写设计文档

为什么要写

为什么要写

- 帮助自己梳理逻辑
- 作为媒介，与他人（包括未来的自己）沟通
  - 关联方
  - 评审方
  - ...

写什么

- 理清功能逻辑
  - 目标是让没有背景的人能够理解
  - 从数据，到逻辑，到对接；
  - 记得沉淀设计原则，可选框架：架构图，ER图，流程图
- 找出难点，着重讨论
  - 目标是让技术决策的不确定性降低
  - 业务系统：权限，对未来的扩展性；C端：并发，限流
- 检验是否达标
  - 目标是没有红线问题
  - 标准：稳定性，安全性，性能...
- 确定工作计划
  - 目标是把本团队资源以及对接方资源与研发任务对齐

写什么

- 理清功能逻辑
  - 目标是让没有背景的人能够理解
  - 从**数据**，到逻辑，到对接；
  - 记得沉淀设计原则，可选框架：架构图，ER图，流程图
- 找出难点，着重讨论
  - 目标是让技术决策的不确定性降低
  - 业务系统：权限，对未来的扩展性；C端：并发，限流
- 检验是否达标
  - 目标是没有红线问题
  - 标准：稳定性，安全性，性能...
- 确定工作计划
  - 目标是把本团队资源以及对接方资源与研发任务对齐

目录

=====技术设计文档=====

需求拆解

技术评审记录

业务逻辑拆分

系统设计

架构设计

数据库设计

前后端接口设计

关联系统对接

数据安全

独立指标

鉴权模型

稳定性、性能、资源需求

新老兼容

时间规划

开发计划

上线计划

版本追踪

举个例子

我们来做个流程类项目，如下

- 背景：小宿的团队有很多项目，项目的主R和同学不固定，希望能够找一个项目制的反馈工具
  - 管理员会创建项目，项目有开始日期和截止日期
  - 项目负责人可以加人，人可以是小数，人整体分配 应该是100%
  - 项目结束会发起一次主R和同学的互评，评价包括信息和评分，结果走一层审批
  - 人的管理者希望有一个看板看大家的情况，绩效期会把这个信息导入绩效系统做参考
- 问题一：prd和技术设计的关系

举个例子

- 我们来做个流程类项目，比如敏捷绩效
- 问题一：prd和技术设计的关系
  - 相关但是不同，prd侧重的是用户，技术设计侧重的是机器，和体系

举个例子

- 我们来做个流程类项目，比如敏捷绩效
- 问题一：prd和技术设计的关系
  - 相关但是不同，prd侧重的是用户体验，技术设计侧重的是数据结构和实现
- 问题二：数据模型设计怎么做
- 问题三：需要做哪些调研和对齐

数据模型设计的几个建议

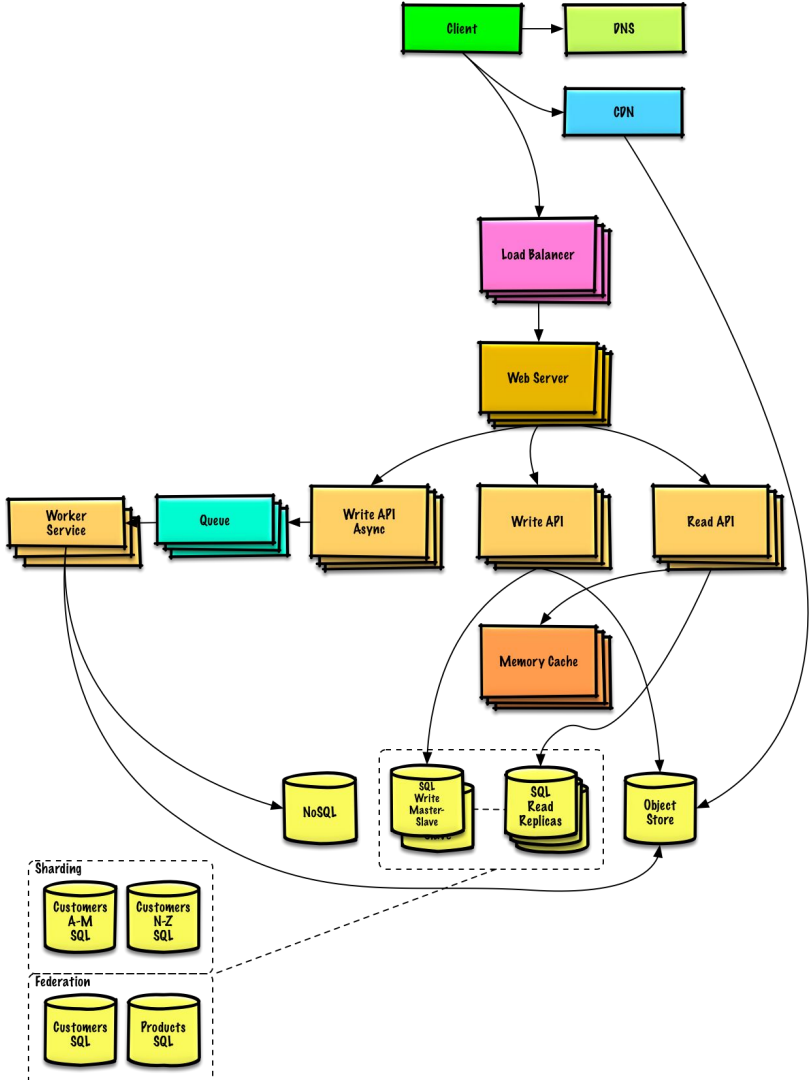
- 先关注写，再关注读
- 关系比字段更重要
  - 尝试去挑战哪些关系可能被破坏
- 多看看其他系统的表结构设计：camunda等等

架构设计

为什么要做架构设计

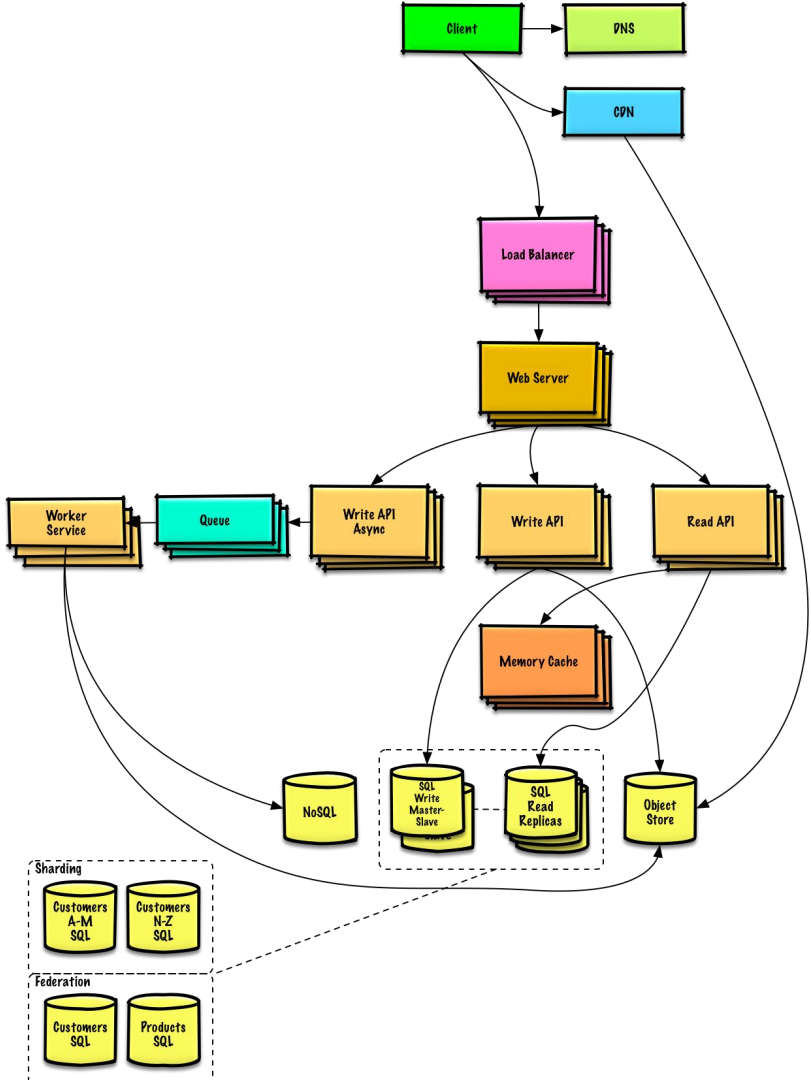
- 在服务层面完成职能的**拆分**
- 理清各个服务的职责和**边界**

举个例子


Client
DNS
CDN
Load Balancer
Web Server
Worker Service
Queue
Write API Async
Write API
Read API
Memory Cache
NoSQL
SQL Write Master-Slave
SQL Read Replicas
Object Store
Sharding
Customers A-M SQL
Customers N-Z SQL
Federation
Customers SQL
Products SQL

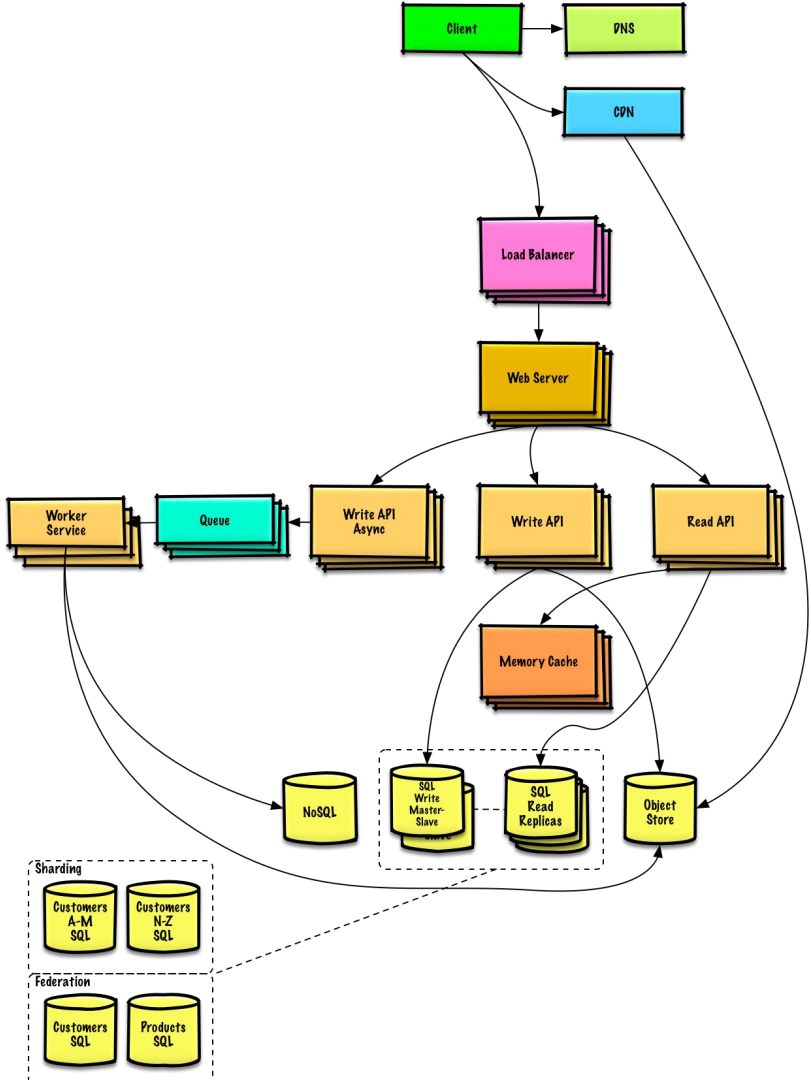
举个例子

- 每一块的职责是什么



举个例子

- 单体架构哪里不好 aka 分布式架构的价值



常见的注意事项

- 状态和存储是最复杂的东西，没事别碰 :)
- 先设计好存储部分，选好工具（数据库，ES，HBase）
- 缓存是用正确性换性能，B端场景适用性要论证

写在最后

- 希望大家不仅知其然更知其所以然
- **端到端**的为你面对的问题负责
- 如果只设计一部分，设计**数据结构**
- **多看多想**其他的设计，技术设计就有技巧但更需要临摹和练习

写在最后

- 推荐几本书：
  - 老生常谈：重构，代码大全，设计模式
  - 这10年的大趋势，可以读很久的书
    - microservices.io，designing data intensive application，the sre workbook
  - 案例集合；the architecture of open source software
  - 随笔：软件随想录
  - 代码开发：effective *系列

写在最后

I still don’t know very much

-- 20 Things I’ve Learned in my 20 Years as a Software Engineer

作业

- 找茬会
  - 选一个没有文档的项目，写一份**新的**设计文档
  - 重点讲哪里现在设计的好，哪里设计的不好

Q&A