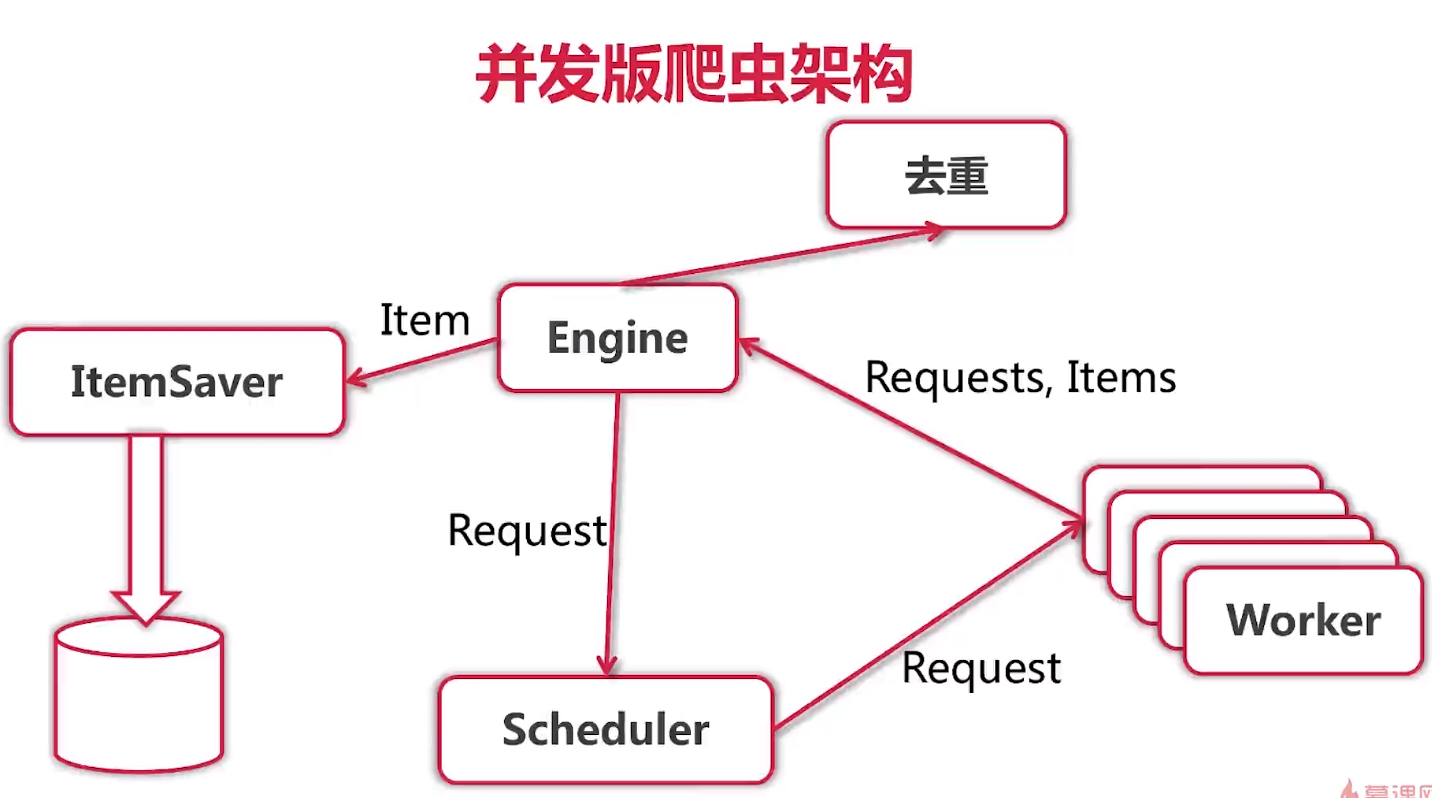
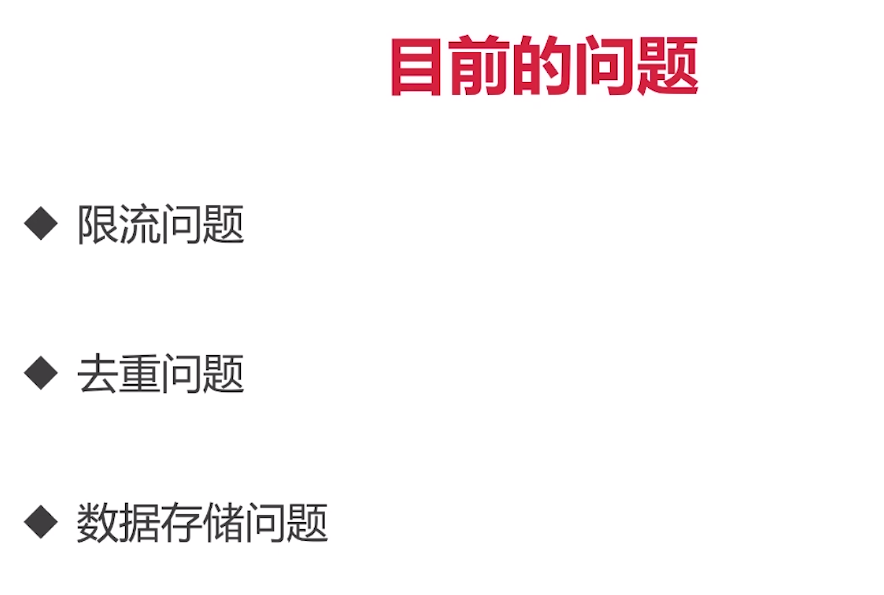
# 并发版爬虫架构

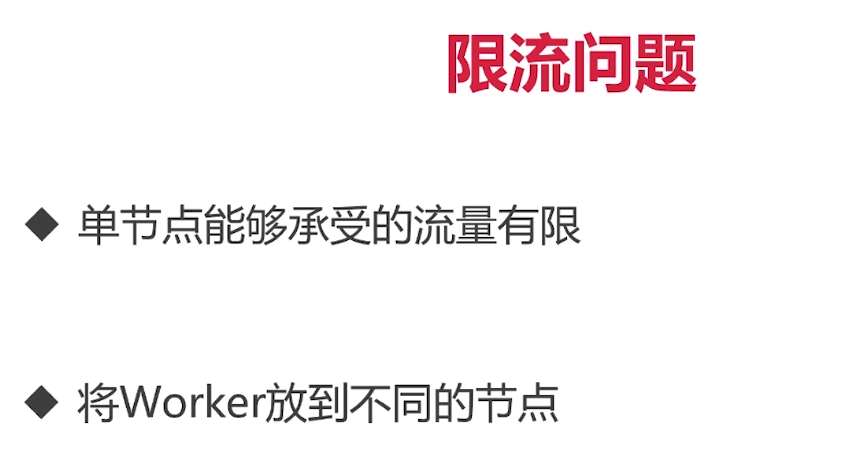


# 根据现有问题设计分布式系统

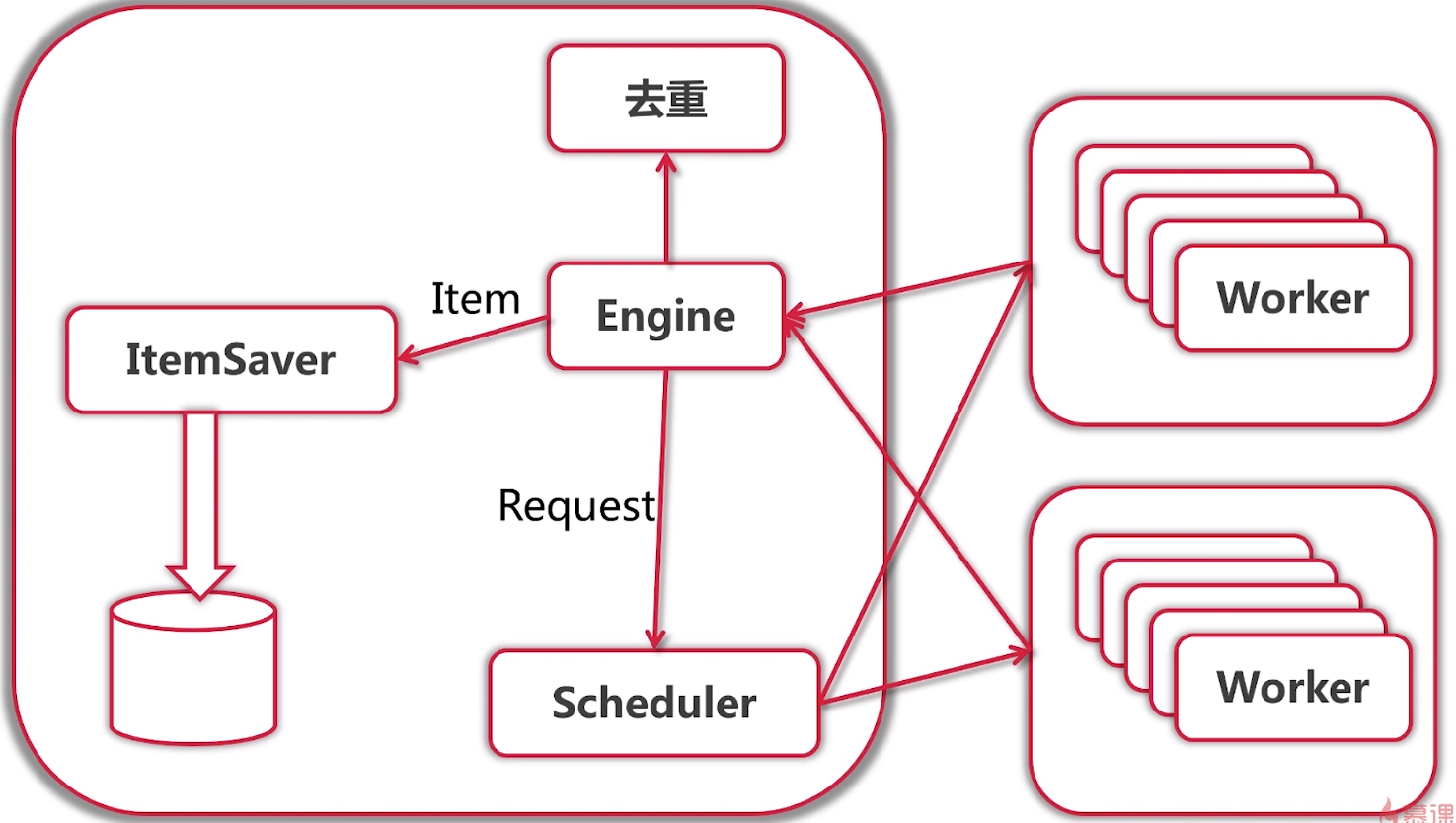
不是根据并发版爬虫架构设计分布式架构，而是根据现有问题设计



## 限流问题



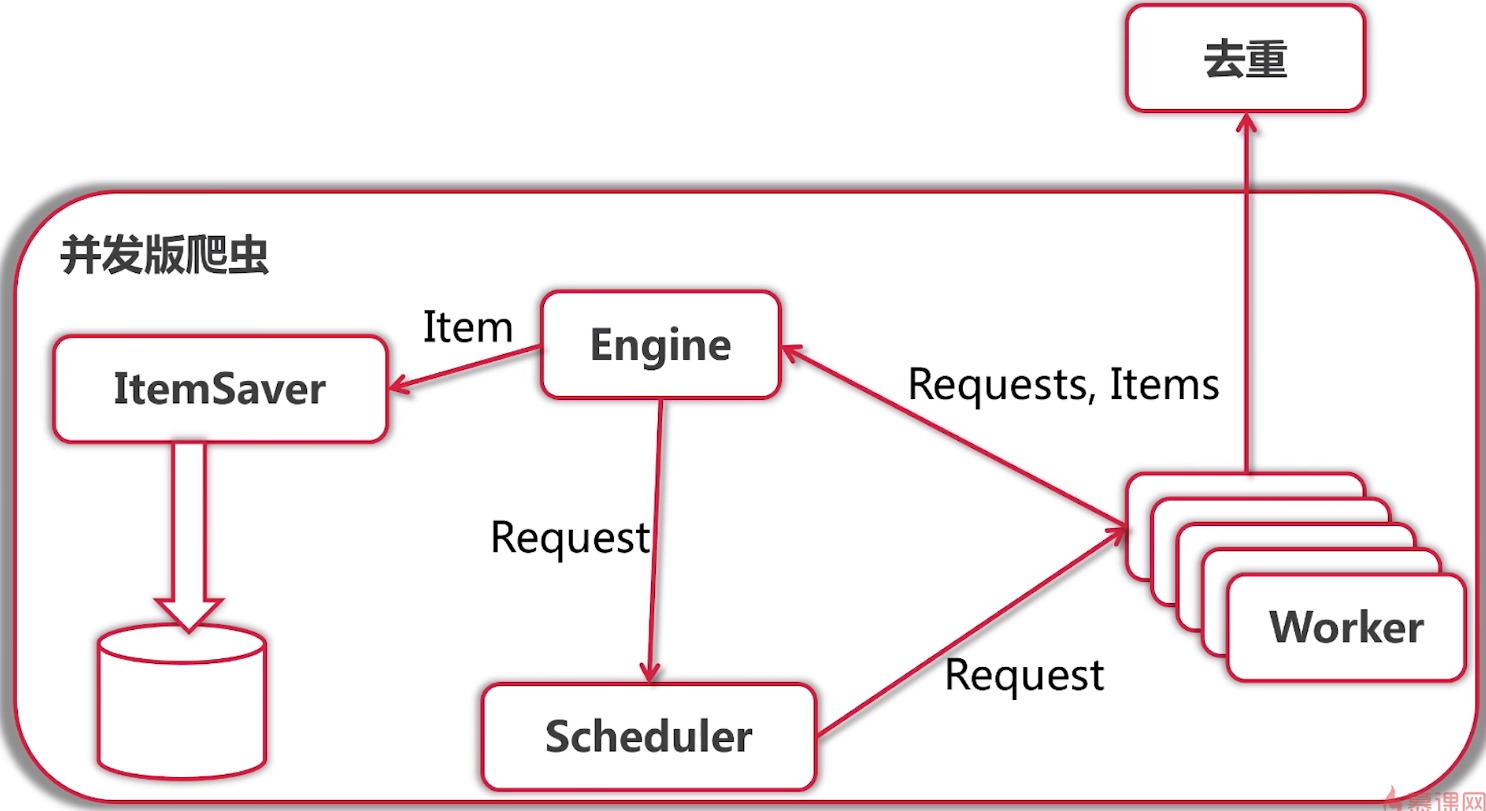
把worker放到多个节点解决限流问题



## 去重问题

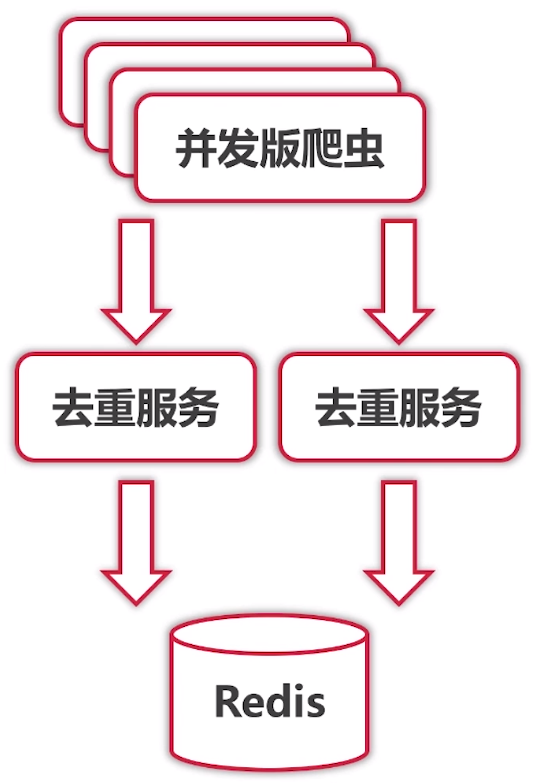


把去重拆分出来成为一个服务，虽然worker去访问去重服务这里会卡住，必须等待结果返回程序才能走下去。但是worker有很多个goroutine，影响并不大。



去重的服务可以有多个

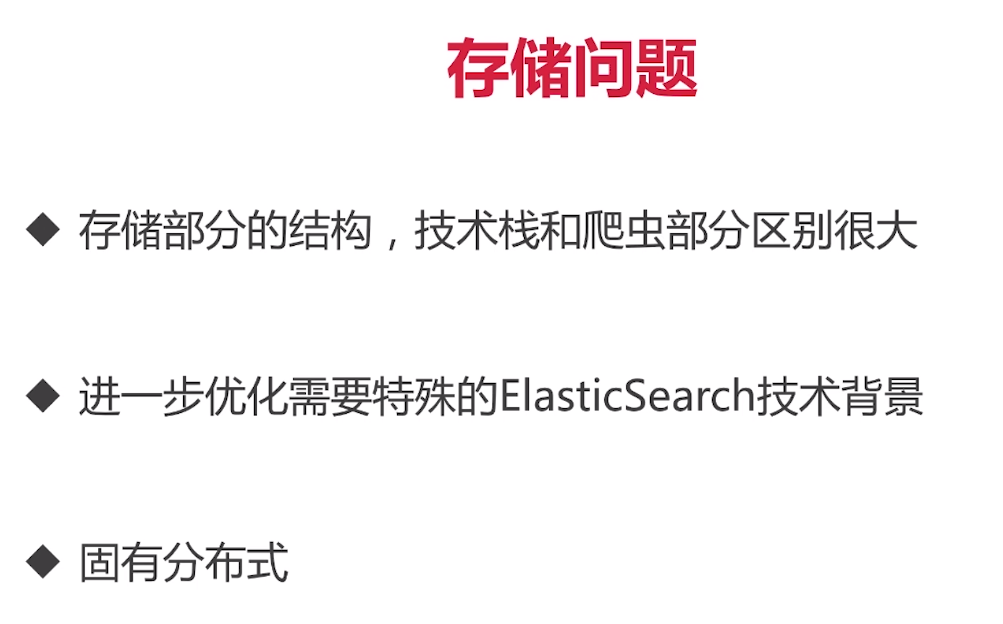
去重的数据存Redis里面



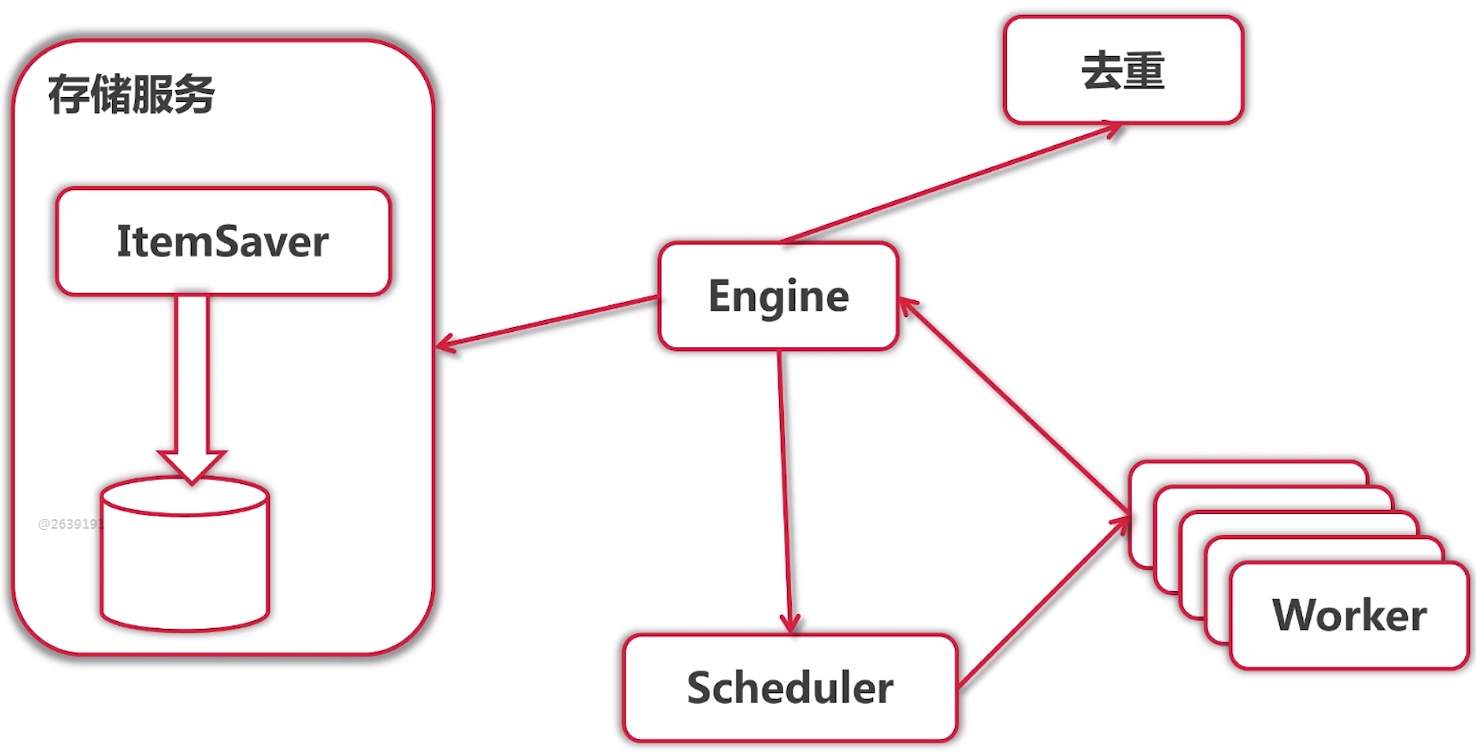
## 存储问题

存储部分所用的技术栈和爬虫部分区别很大，数据量大到一定程度的时候需要请专门的数据库人才来优化，所以需要把数据存储部分拆分出来成为一个服务。

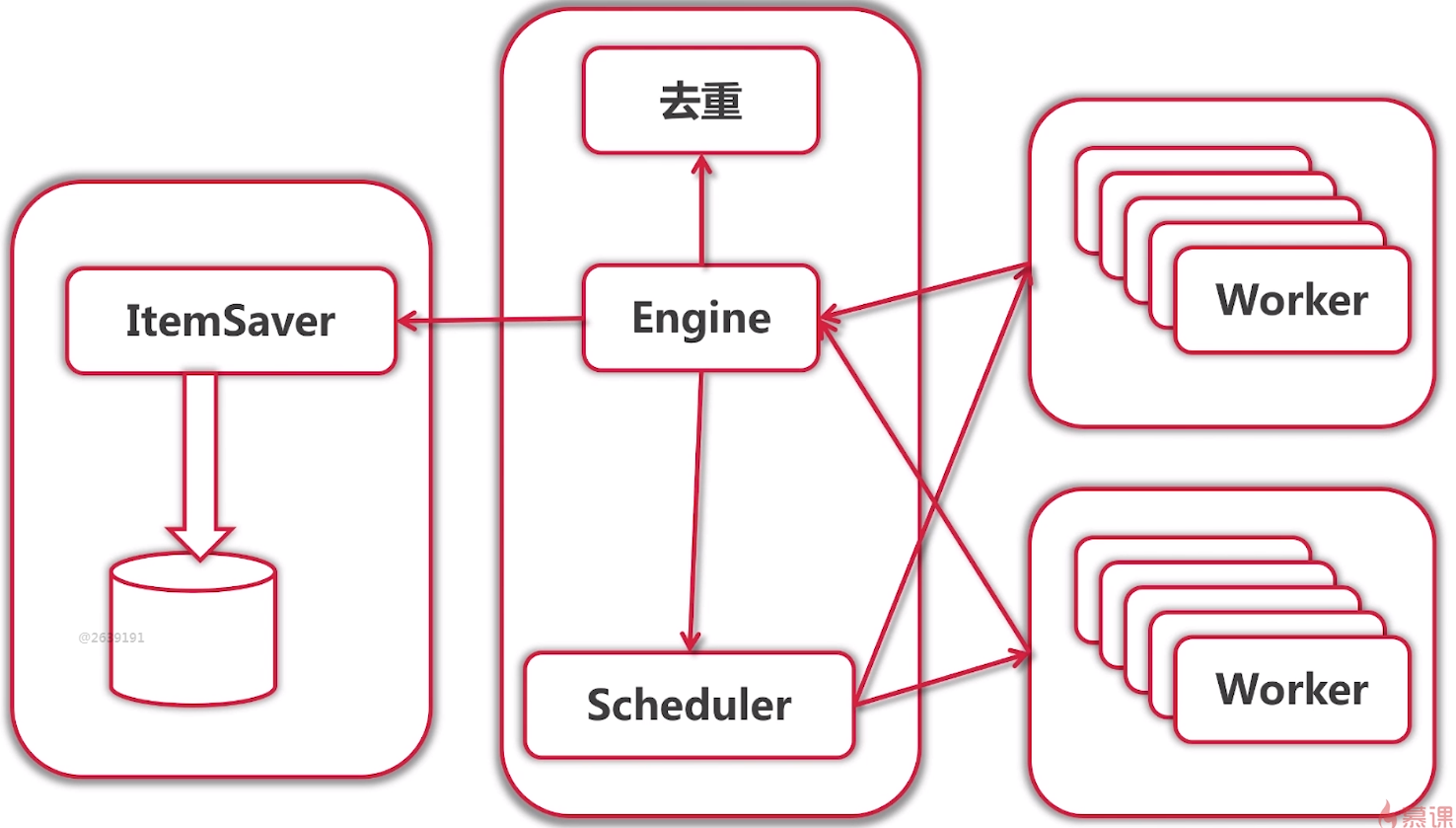
因为人的能力不同，每个服务可能会需要不同的人来开发、维护，这就是人的固有式分布。



把ItemSaver拿出来作为一个服务



## 最终架构

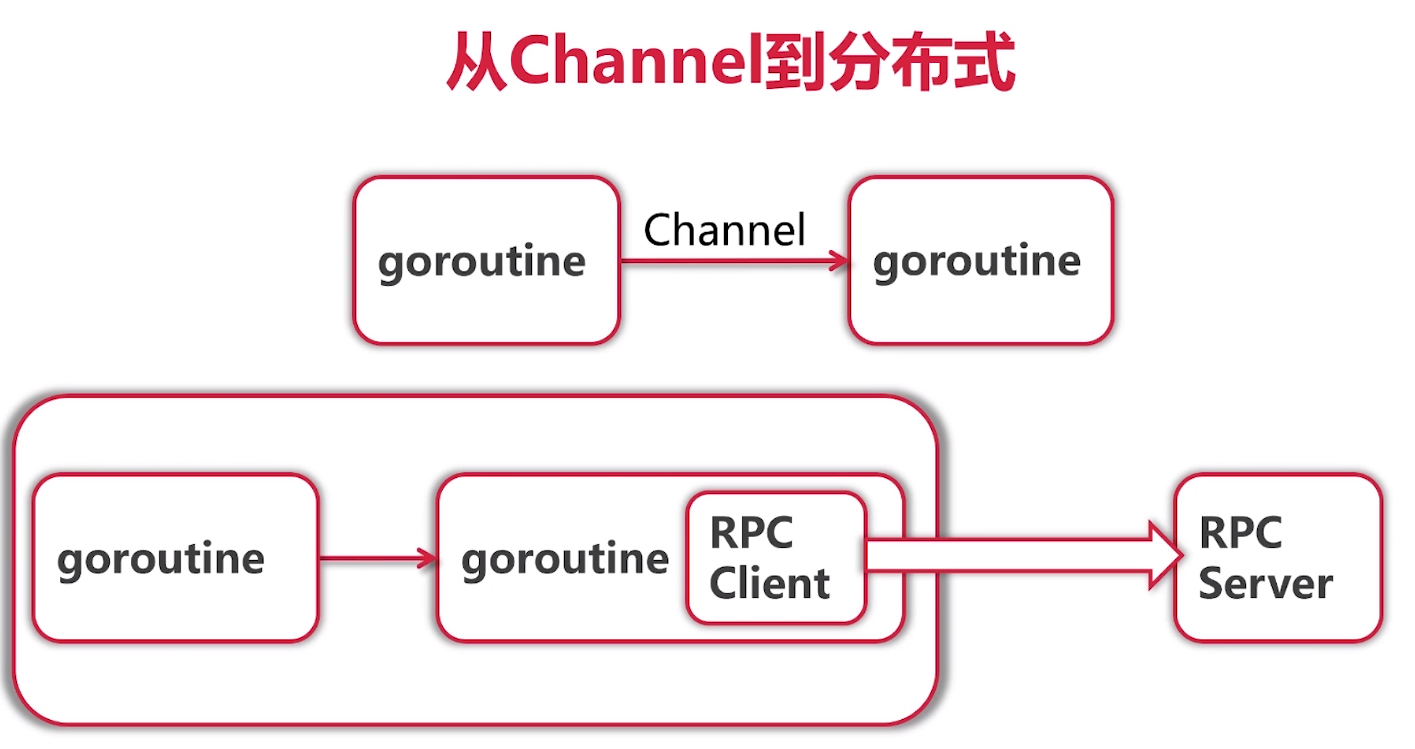


# 从channel到分布式

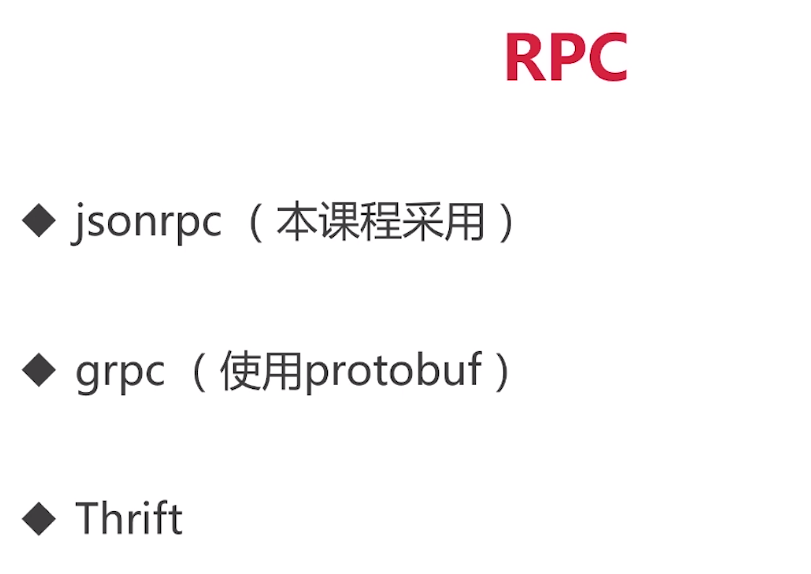
并发版各模块之间通信是用channel，非常像分布式系统，所以go语言是天生支持分布式系统的

从channel入手把系统改造成分布式系统

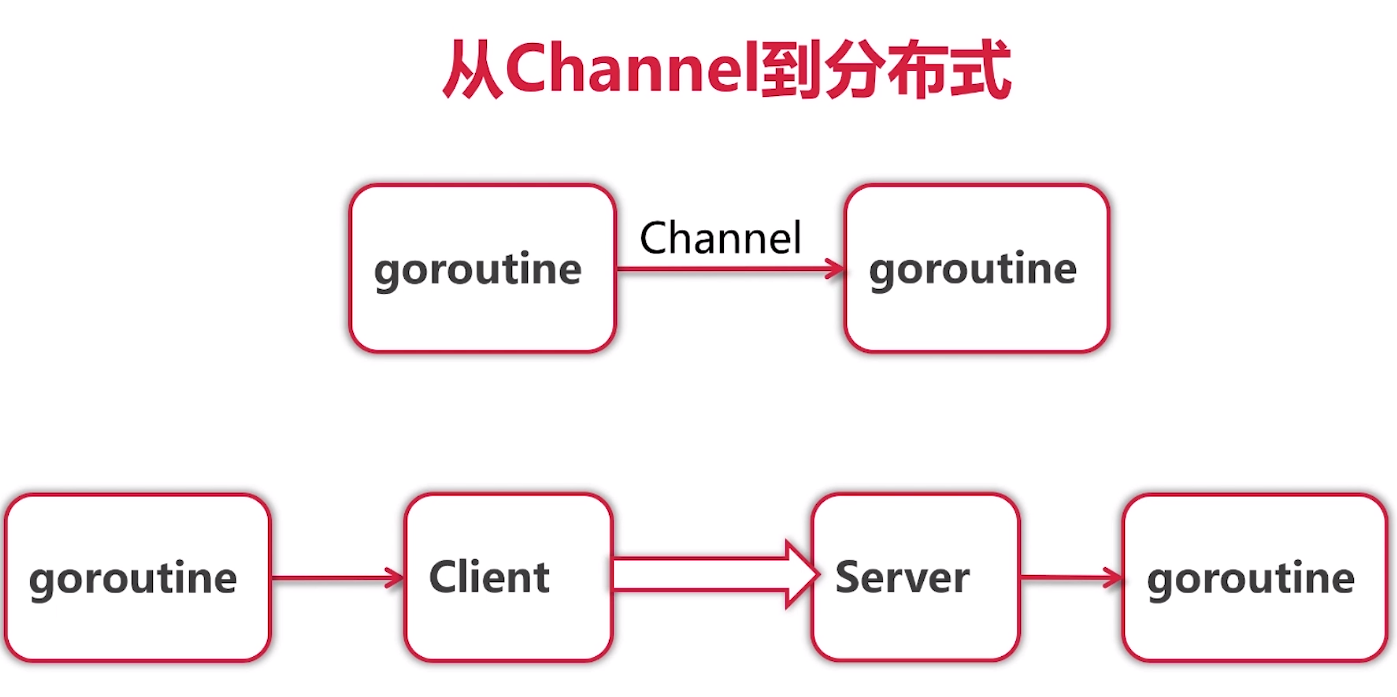
## 通过RPC实现



RPC是业界标准，还可以跨语言实现分布式



## 通过网络进行channel数据流



可以通过以下开源项目实现，是自有协议。当然也可以根据自己的需求自己实现。



## 通过中间件实现

