

程序员 系统设计

关注者
41

被浏览
4,088

Windson Yang 也关注了该问题



程序员如何提高系统设计的能力？

关注问题

写回答

邀请回答

好问题 3

添加评论

分享

...

查看全部 2 个回答



Windson Yang

osjobs.net - 海外兔（求职课程 | 经验分享）

20 人赞同了该回答

本文首发于专栏：[编程超能力培训班](#)

本文章由 [ResumeJob](#) 撰写，**ResumeJob 能帮助你审视简历，模拟面试，重新规划你的职业生涯**。我们团队在国内外公司面试过数百名的工程师，浏览过千份简历，志在帮助程序员找到合适自己的工作，如果你想知道简历哪里出现问题或者想在面试前进行模拟面试的话，欢迎联系我们，了解更多细节欢迎浏览 [resumejob.github.io](#)。

求职系列文章：

- [程序员如何写一份更好的简历](#)
- [程序员如何准备技术面试](#)
- [程序员国外求职指南](#)
- [系统设计面试 101](#)
- [程序员面试推荐书籍](#)

概述

最简单快速学习系统设计的方法是通过 AWS，Google Cloud 等文档进行学习，例如要了解如何设计高可用的数据库，可以参考微软的 cosmos db 的文档，里面描述了它的设计思想和使用场景。本文会以一个例子讲述系统设计的流程。

面试流程

这篇文章并不是教读者如何构建一个高可用的系统，而是着重于阐述系统面试的具体流程以及常见的错误，系统设计流程总体可以分成 5 个 部分，总时长约一个小时

- 确定范围（5分钟）
- 粗略计算（5分钟）
- 整体架构（5分钟）
- 组件架构（20分钟）
- 架构优化（20分钟）

1. 确定范围

系统面试开始之时，面试官可能只会提出一个问题：

“如何设计 Uber 这样的应用？” “如何设计 Whatsapp 这样的应用？”

如果你刚参加工作或者刚毕业，会对如何设计一个这样的系统束手无策，要设计高可用，易扩展的系统需要广阔的计算机知识以及大量的项目经验累积，也往往需要一个团队配合设计以及多个版本的迭代。所以，面试官并不期待你可以在一小时内把 Uber 或者 Whatsapp 的全部功能都设计出来。系统设计面试真正考核的是

- 把抽象问题转变成实际工程中能够解决的简单问题
- 利用自己现有的知识设计一个基本可用的系统
- 观察现有方案可能出现的瓶颈并提出解决方案



关于作者



Windson Yang

osjobs.net - 海外兔（求职...

kenshinji 也关注了他

回答

14

文章

35

关注者

1,488

已关注

发私信

被收藏 40 次

JAVA学习

0 人关注

小啊小二郎呀 创建

程序员

0 人关注

痴人妄语 创建

看了是否又能坚持？

0 人关注

小田同学 创建

编程

0 人关注

空指针 创建

个人修为

0 人关注

卖铁壳的罗先生 创建

相关问题

作为一名程序员，你觉得最重要的能力是什么？ 96 个回答

程序员应该怎么样才能让别人知道自己的能力强呢？ 12 个回答

对于职场新人（程序员），如何在工作中提升自己？ 2 个回答

赞同 20

▼

1 条评论

分享

收藏

喜欢

...

第一阶段，确定范围就显得特别重要了。我们需要先与面试官讨论该系统

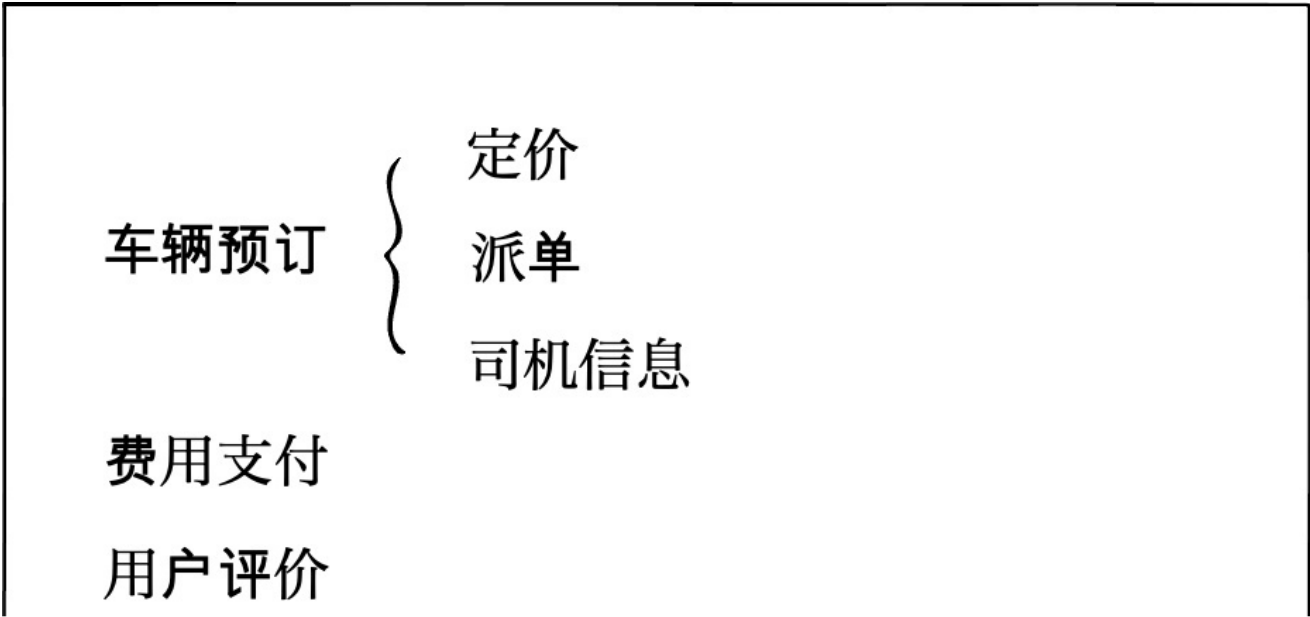
- 需要实现的功能点
- 用户量，数据存储量，两者的增长量以及其他限制条件

1.1 需要实现的功能点

如果你之前没有使用过这些应用（Uber, Whatsapp），不要害怕通过提问来确认功能点。（这并不奇怪，2016年，美国有三分之一人没听过 Uber 或者 Lyft）。用来确认功能点的问题有：

用户可以使用这个应用做什么？ 这个应用有什么与众不同的功能？

设计架构本身花不了多少时间，我们值得投资更多时间在了解清楚问题中。**切记，不要没有理解问题就动手设计。**本文以设计 Uber 为例，Uber 包括多项功能，前端工程师关注**显示地图，司机位置更新，日志传输**。后端工程师关注**车辆预订（包含定价，派单，司机信息），费用支付，用户评价，日志分析**。我们先把想到的功能点都列在白板上，这里以后端工程师为例：

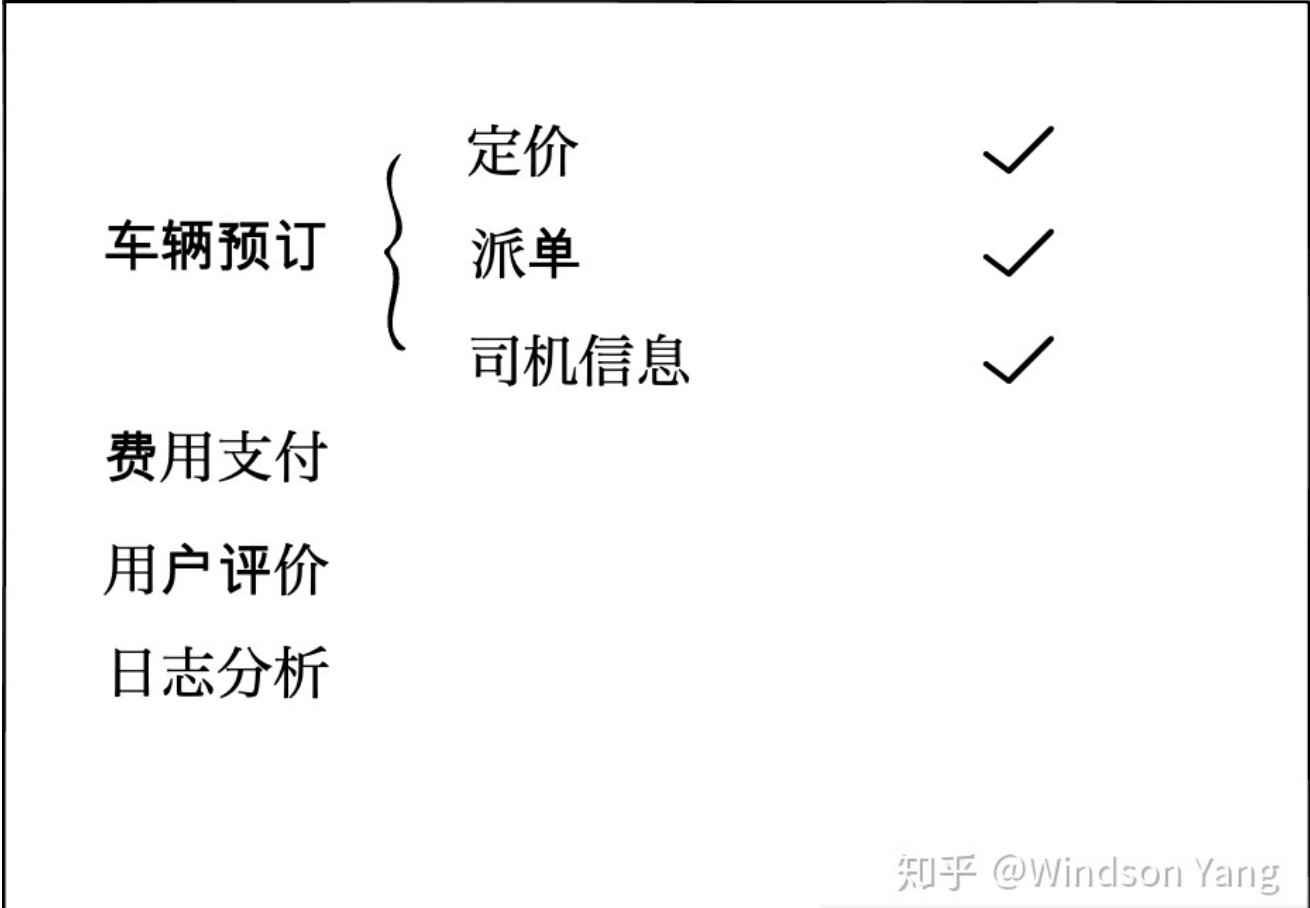


继续浏览内容



进行确认：

“我先从这些功能开始进行设计可以吗？”



知乎 @Windson Yang

有哪些对程序员很重要，但大部分学校基本不培养的能力？ 31 个回答



相关推荐

- 

如何快速成长为后端架构师
宋文峰
★★★★★
- 

怎样才能成为很厉害的程序员
★★★★★
- 

谈谈工程师常常忽视的软件开发能力
★★★★★

UCloud 优刻得，科创板上市！

海外云服务器 全球19大机房同价

2核4G
450元/年

4核8G
750元/年

[立即抢购](#) [广告](#)

刘看山 · 知乎指南 · 知乎协议 · 知乎隐私保护指引
应用 · 工作 · 申请开通知乎机构号

打开

继续

儿童色情信息举报专区
证照中心
联系我们 © 2021 知乎

我们必须经过面试官对设计的功能点认可才能进行接下来的系统设计。



1.2 用户量，数据存储量，两者的增长量以及其他限制条件

接下来我们需要了解

“每秒有多少用户预订车辆？有多少数据需要存储？用户每个月的增长量是多少？对响应时间以及服务器数量有没有限制？”

面试官可能会助攻一把：

“车辆预订系统每秒有 1万次 请求，每天有 100G 数据需要存储，用户每月增长 10%，服务器不能超过 20台。”

我们在感谢面试官之后把这些重要数据记录在白板。不过面试官也可能简单地回答：

“每天有 100万 人次预订车辆，数据库需要存储每次预订的所有信息，包括订单号，起点，终点，路径经纬度等。”

在不清楚所需数据范围的时候，需要进入**粗略估算**阶段

2. 粗略估算

2.1 吞吐率

粗略估算是每名工程师都需要学习的技巧，它帮助我们把现有的数据转换成项目需要的数据。吞吐率是衡量服务器性能的一个常用指标，意思为服务器每秒钟能够处理的请求数量。Uber 每天有 100万 人次车辆预订，我们可以假定每次预订客户端发送两个 HTTP 请求，加上一些背景知识：

- 每天有 86400 秒

继续浏览内容

知乎

发现更大的世界

打开

Chrome

继续

服务器配置 CPU：Intel(R) Xeon(R) CPU 1.60GHz 内存：4GB 硬盘转速：15k/min

软件	并发用户	总请求数	请求内容	缓存措施	吞吐率
Nginx/0.7	1000	50000	151字节静态文件	无	10556 req/s
Apache/2.2	100	1000	151字节静态文件	无	6364 req/s
Apache/2.2	100	1000	1.2m静态文件	sendfile()	590 req/s
APC Cache	100	1000	PHP动态页面	PHP 动态页面缓存	51 req/s
Apache/2.2	100	1000	PHP动态页面	无	51 req/s

我们可以看到，**由于请求不同的内容，使用不同的软件以及设置不同的缓存策略**都会给吞吐率带来极大的影响。Uber 业务逻辑比较多，我们估计单台服务器每秒能够处理 50次 请求车辆预订请求，预计用户月增长量为 10%，所以总共需要 8台 后端服务器来处理车辆预订功能。

2.2 数据请求

假设每次车辆预约产生 3次 数据库查询以及 1次 数据库写入。每秒总共产生 900次 数据库查询以及 300次 数据库写入。那么这又需要多少台数据库支撑呢？你猜到了，这也是一个复杂的问题，涉及到**查询热点的分布，数据库缓存的设置，索引设置**等等。这里有两篇文章分析介绍了一些案例：

- [High Performance MySQL](#)
- [Using mysql as nosql](#)

我们保守估计单台数据库可以支撑每秒 2000次 查询以及 500次 写入。由于数据库对数据持久化以及可用性要求非常高，我们通常都需要**添加主从热备，读写分离以及故障切换功能**。因此我们在之后的瓶颈分析再进行讨论。在这里，暂时我们先使用一台数据库。

2.3 存储量

每天 100万 次预订，每次预订大概产生 10KB 左右的信息。每天产生 10G 的数据，加上其他功能大概每天 20G 左右。现在的硬盘都是以 TB 为单位了，所以存储量不需要担心。粗略估计结束了，把计算完的数据记录下来：

车辆预订

请求 300 req/s

数据请求

读 900 req/s

写 300 req/s

数据存储

每个月 600G

知乎 @Windson Yang

3. 整体架构

我想大多数求职者都没有设计过 Uber，也没写过车辆预订的相应业务代码。但是没有关系，系统

继续浏览内容

知乎

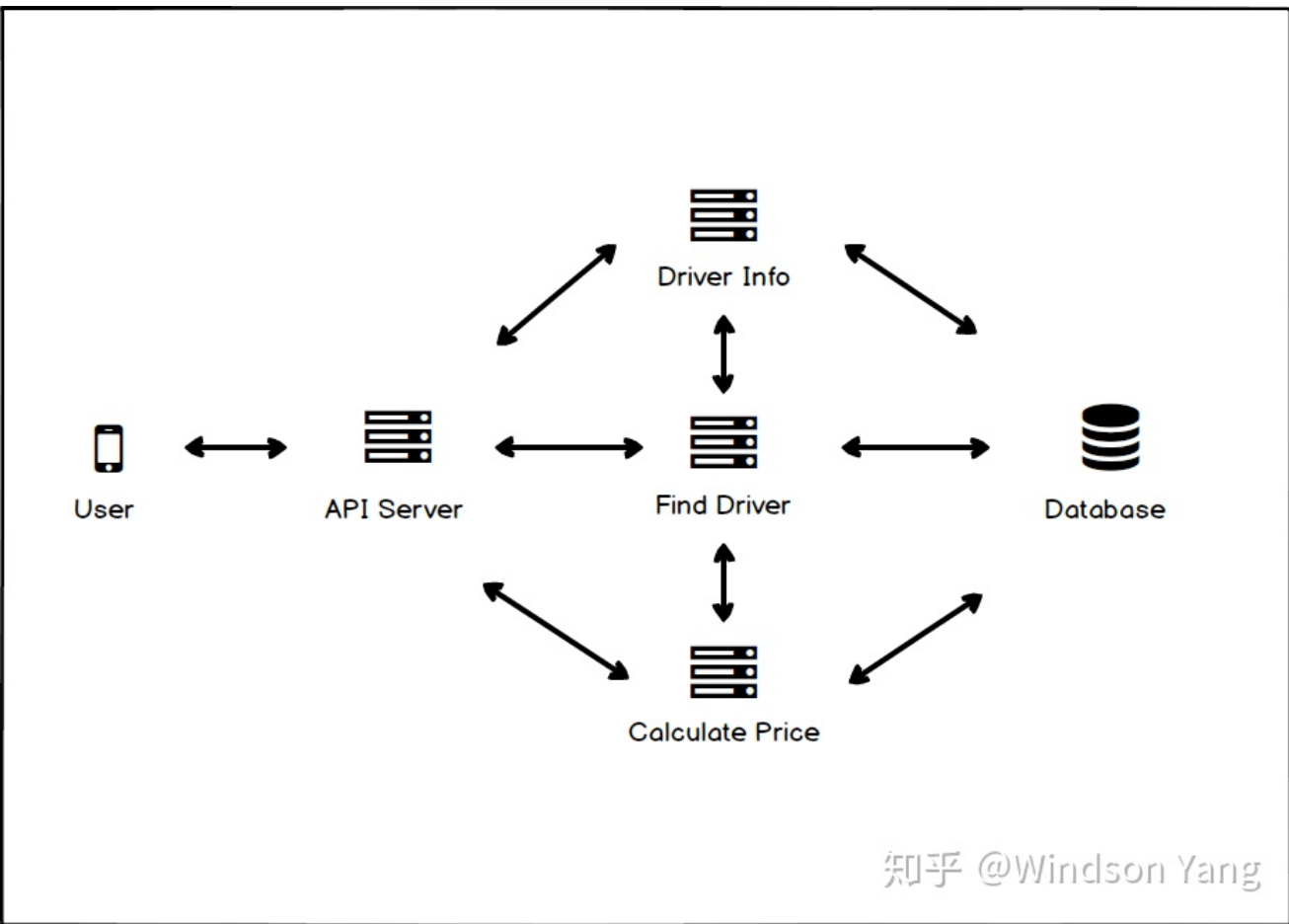
发现更大的世界

打开

Chrome

继续

错）。综合上面的内容，我们先画出一个粗略的整体架构。



- 1. 用户请求 API 服务器
- 2. API 服务器请求相应的业务服务器（业务服务器可能会互相



- 户附近有多少司机)
- 3. 查询数据库
- 4. 返回结果给用户

我们先从这个整体架构开始讨论每个功能的细节。

4. 组件架构

这里我们阐述定价，派单，司机信息三个功能的架构设计。

4.1 定价

定价背后的逻辑比较复杂，主要由于商业原因而不是技术原因，Uber 也一直在调整它的定价策略而达到利润最大化。不过，我们可以先设计一个粗略的版本，首先，分析影响价格的因素，

- 路程长度
- 需求量（预约人数与周围司机数量比值）
- 预订时间
- 地区消费水平


这个表会包含非常多的因素，不过我们可以猜测里面最主要的三个因素以及它们的权重有以下的大小关系：

$$W(\text{路程长度}) > W(\text{需求量}) > W(\text{预约时间})$$

重复一次，这并不是真实的情况，只是我猜测的关系式，定价可能是通过一个公式计算出来：

$$\text{Price} = F(\text{路程长度}) * K(\text{需求量}) * G(\text{预约时间})$$

继续浏览内容

 **知乎**
发现更大的世界

打开

 **Chrome**

继续

- 1. 目的地离起始点1.5公里
- 2. 附近2公里内有4名其他用户在预约，两名空闲司机（需求量为2）
- 3. 早高峰时段

预约车辆，那么预估价格为

$$10 * 1.4 * 1.2 = 16.8 \text{ 元}$$

这样看起来，一个粗略的定价功能就实现了，我们在白板中写下数据库结构以及 API 示例。这个功能用 NoSQL 实现会更容易，不过现阶段我们先使用 SQL 表：

距离表

- distance_range
- price

需求表

- demand_range
- price

时间表

- time_range
- price



API 示例

API 示例主要描述输入以及输出即可，以下为 Python 代码，可以略过实现，不过注释要写清楚：

```
def calculate_price(latitude, longitude, current_time):  
    ...  
  
    input:  
        latitude (float), longitude (float): current user location  
        current_time (timestamp): user request timestamp  
  
    output:  
        price (float): estimated price  
    ...  
  
    pass
```


4.2 派单

接收到预约请求之后，系统会派单给合适的司机，假设系统只会派单给 **2公里 内在线以及空闲的司机**。这里的三个要点 “2公里内” ， “在线” ， “空闲” ，三个要点中，第一个和第三个比较容易，只需要在查询在线司机的时候附带筛选条件即可。假设数据库中存储了司机的信息，里面包含

- 基本信息，包括车型，车牌，评分
- 最后在线时的经纬度
- 最后在线时的时间
- 现在是否空闲

那么就可以根据 经纬度以及是否空闲先筛选出附近可用司机的列表，然后再根据司机的车型、评

继续浏览内容

 **知乎**
发现更大的世界

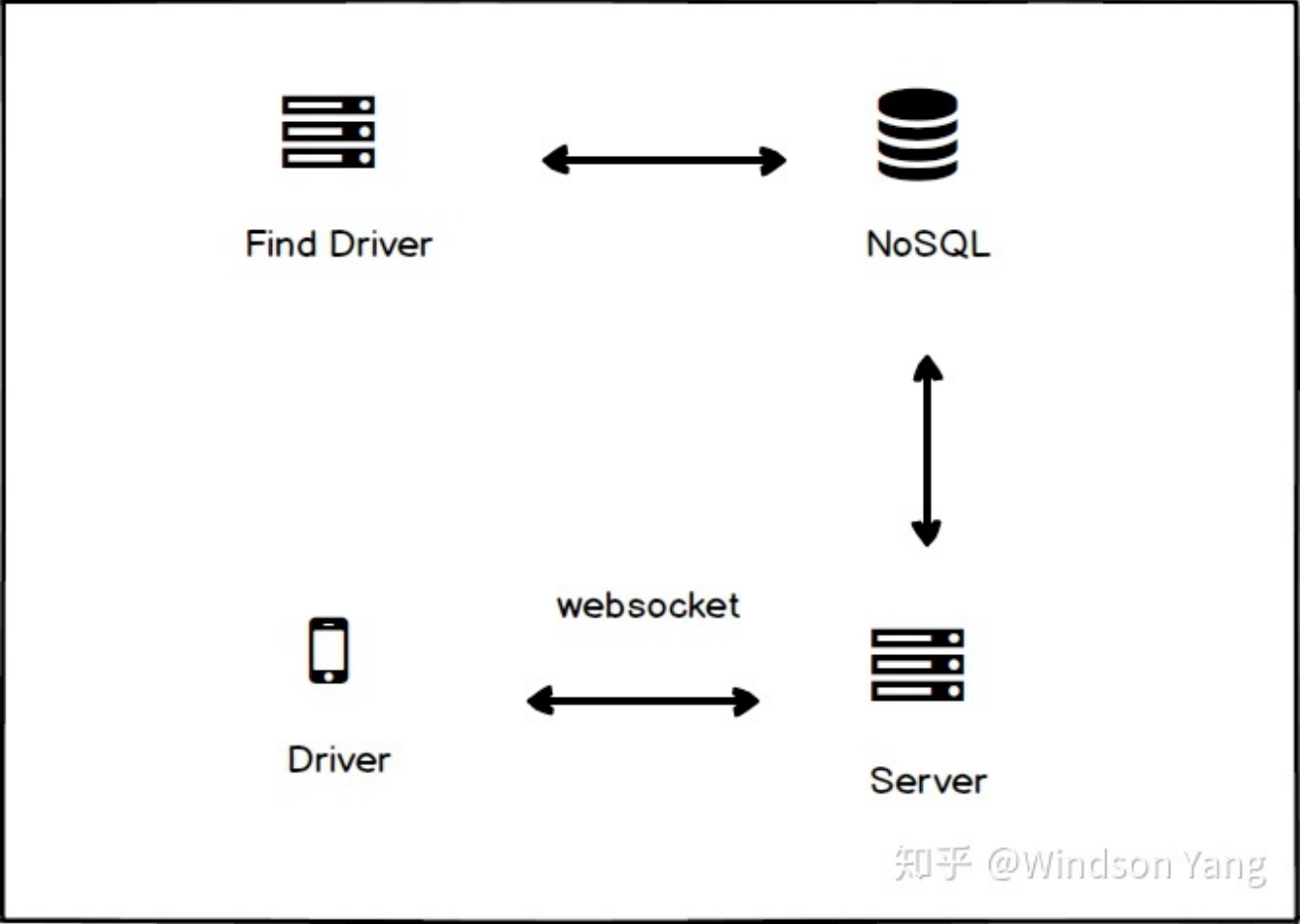
打开

 **Chrome**

继续

反馈不及时，司机可能因为网络原因 X分钟 没请求服务器导致错过了派单，其次是司机下班了，但是系统在 X分钟 内依然派单过去。最后是服务器需要处理大量的司机客户端请求，对系统的性能有影响。这种解决方法适合在初期使用，因为实现简单，容易维护。

第二种解决方法是使用双向连接，例如 Websocket，服务器可以直接查询客户端连接是否被关闭，司机是否在线。服务器同样在数据库维护着一个列表，每次用户连接服务器，服务器在列表中添加司机，一旦连接断了，则从列表中删除。我们在白板中写下具体架构以及数据库架构：



司机表

- driver_id
- car_number
- ranking
- last_online_time
- free
- ...

继续浏览内容



打开



继续

```
output:
    driver_id (list of int): use driver_id to request driver's info later
    ...
pass
```

4.3 司机信息

派单之后需要在用户的客户端显示司机的基本信息，实时位置以及车辆方向。此时 Websocket 双向链接的优势就显现出来了，服务端可以实时获取到司机的当前位置发送给客户端，然后客户端把司机的位置渲染在地图的道路上。这里面的难点非常多，像滴滴使用的是 FLP 融合定位，在GPS有问题的时候，使用 WIFI，道路匹配，车辆航位推算等方式来解决这个问题。具体细节如果你了解的话可以进行进行阐述，在这里我只是简单地当成客户端从服务端获取司机的经纬度然后在地图上渲染。

4.4 数据存储

那么我们之前记录需要存储的数据该存储在哪里呢？服务器内存？文件？还是数据库？如果是数据库的话是 SQL 还是 NoSQL。几个基本原则是：

- 暂时不知道瓶颈在哪里，但是数据又需要持久化存储，我们优先使用 SQL 数据库。（Facebook 在有5亿日活量的时候，主要还是用 MySQL）。
- 关系性不强，需要频繁修改的地方（例如上面的司机在线列表）我们可以使用 NoSQL 存储在内存中。
- 大型并且很少需要查询的文件，例如日志，可以存储在文件系统

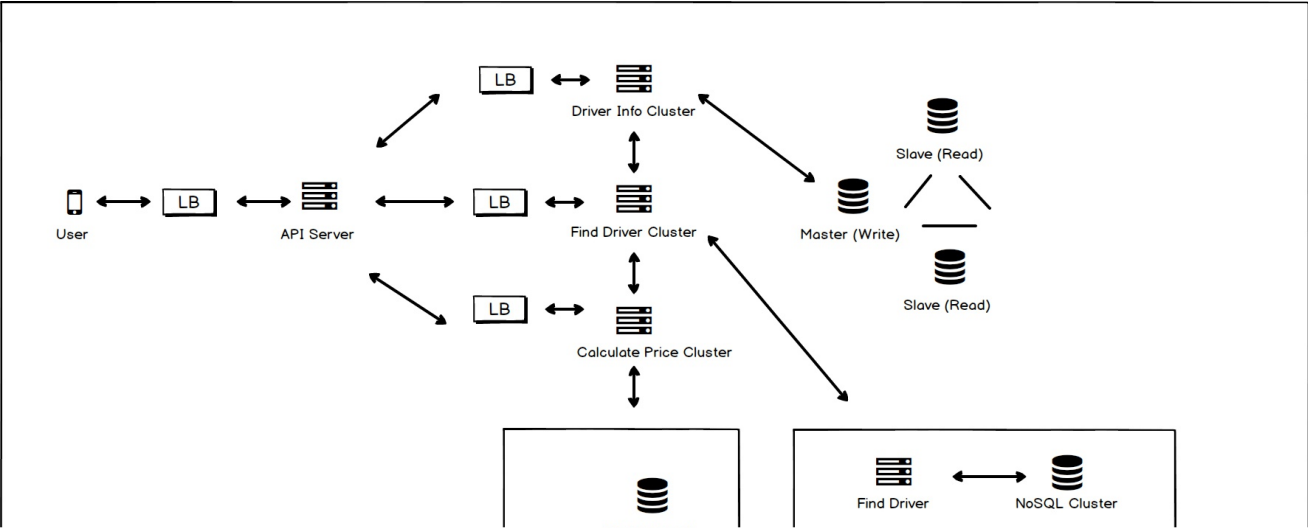
5. 架构优化



完成架构之后，我们需要进行架构优化，首先是业务逻辑上的优化，例如，要根据用户当前经纬度找到两公里内的司机，原始的方式把可以用户经纬度把所有在线司机的经纬度——对比，然后找出距离少于两公里的司机。这个时间复杂度是 $O(n)$ ， n 代表在线司机的数量。更好的方法是把地图分成两公里为单位的一个个格子，然后找与用户同一个格子的司机。时间复杂度马上降到 $O(1)$ 了。其次是系统架构上的优化，需要我们观察系统瓶颈，消除单点故障，保证高可用以及水平扩展。基本的优化思路如下（需要根据实际情况修改）

- 服务器均使用服务器集群
- 服务器集群前加上负载均衡
- 计算耗时久的数据使用缓存存储，
- 关系型弱且需要频繁修改的数据使用 NoSQL 存储，
- 数据库使用主从热备，读写分离，以及故障切换

在原本的架构中，定价功能可以使用 NoSQL 缓存价格的哈希表以及在线司机列表，派单功能可以使用 Websocket 进行双向连接，再添加上服务器负载均衡以及数据库主从热备，最终我们可以得到这个架构图。



继续浏览内容

知乎

发现更大的世界

打开

Chrome

继续

下保证高可用，这时候就需要利用我们平时积累的架构方案作出优化。就像算法题中，了解基础的数据结构能够帮助我们快速解决问题，系统设计也需要我们了解这些常用的软件架构以及使用场景，这里我列出了一些学习资源，你可以了解下。

- [Inside NGINX: How We Designed for Performance & Scale](#)
- [Introduction to Redis](#)
- [Redis Sentinel Documentation](#)（重点）
- [Netflix: What Happens When You Press Play?](#)
- [How does Hadoop work and how to use it?](#)
- [Thorough Introduction to Apache Kafka](#)

总结

系统设计资源非常多，理解常用的架构以及模式就能解决大部分的问题，同时，别忘记借助面试官的提示来解决问题。最后希望你能在系统面试中好好表现，:D



知乎 @Windson Yang

如果你觉得这篇文章帮助到你，希望我们能继续分享自己的技术知识和求职经验，请帮我们点赞吧。

编辑于 2020-02-15

更多回答



梦涯
Google Robotist

系统设计的能力很难通过自学提高的，最好是在工作中或者项目中学习。当你见到了一些系统框架，或者当一个问题复杂到你不得不用一些框架的时候，你就会自己去寻找合适的框架。

实践出真知。只靠读一些系统设计的书是远远不够的。

发布于 2020-10-04

继续浏览内容



打开



继续