搜索引擎相关技术
Crawler & Typeahead

课程版本:v6.0 主讲人:东邪

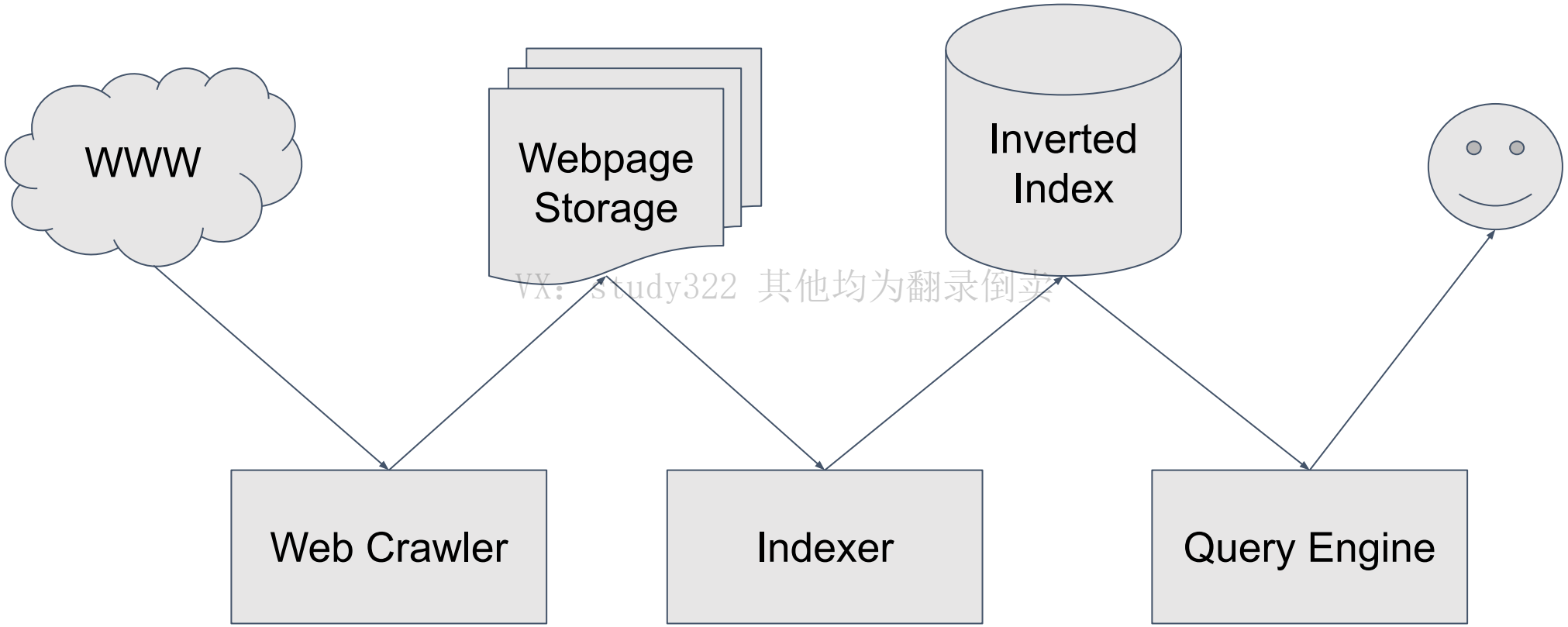
扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
知乎: http://zhuanlan.zhihu.com/jiuzhang
官网: http://www.jiuzhang.com

课程大纲

- 搜索引擎相关技术概要
  - 倒排索引 Inverted Index
  - 中文分词 Chinese Word Segmentation
- 设计爬虫系统
  - 什么是生产者消费者模型
  - 如何设计一个可以工作的爬虫
  - 如何设计一个更 Robust 的爬虫
- 设计 Typeahead
  - 前端的 Typeahead 设计
  - 后端的 Typeahead 设计 (Google Suggestion)

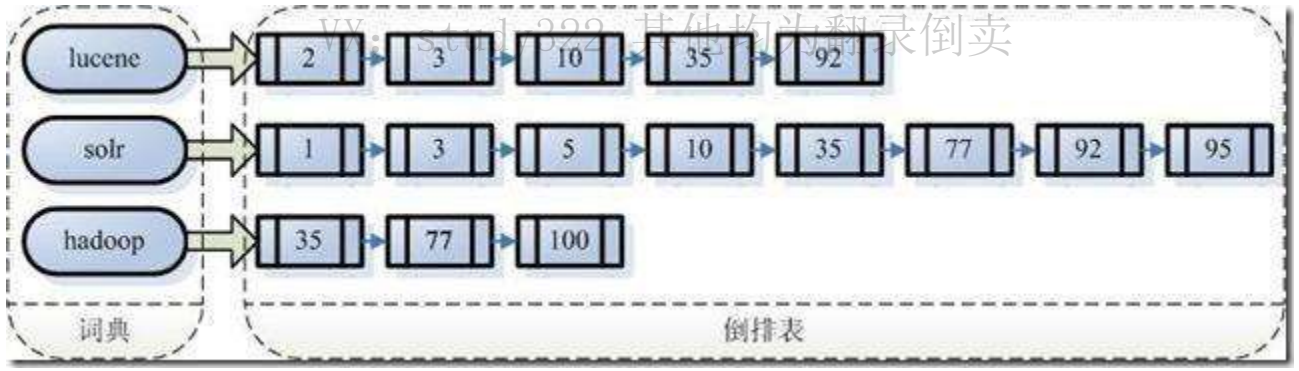
搜索引擎技术概要



倒排索引 Inverted Index

从 word 指向 doc id list 的索引(index)

同理，Forward Index 指的是从 doc id 指向 word list 的索引。



中文分词 Chinese Word Segmentation

“化妆和服装”

拆分方法1：化妆 | 和 | 服装

拆分方法2：化妆 | 和服 | 装

维特比算法 Viterbi Algorithm

https://www.zhihu.com/question/20136144

https://en.wikipedia.org/wiki/Viterbi_algorithm

LintCode 练习题

https://www.lintcode.com/problem/word-break

https://www.lintcode.com/problem/word-break-ii

设计爬虫系统

Design Web Crawler

Scenario 场景

爬虫的目的是要做什么？

爬虫的目的

抓取互联网上的“所有”网页内容信息

并存储下来供索引器(Indexer)建立倒排索引(Inverted Index)

如何从一个网页找到另外一个网页

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>W3Cschool - 学技术查资料，从w3cschool开始!</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<meta name="renderer" content="webkit" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<meta name="keywords" content="w3cschool,w3cschool在线教程,技术文档,编程入门教
程,w3school,W3C,HTML,HTML5,CSS,Javascript,jQuery,Bootstrap,PHP,Java,Sql" />
<meta name="description" content="w3cschool是一个专业的编程入门学习及技术文档查询网站，提供包括
HTML,CSS,Javascript,jQuery,C,PHP,Java,Python,Sql,Mysql等编程语言和开源技术的在线教程及使用手册，是类国外
w3schools的W3C学习社区及菜鸟编程平台。" />
<link rel="stylesheet" type="text/css" href="//www.w3cschool.cn/statics/css/w5.css" />
<link rel="stylesheet" href="//www.w3cschool.cn/statics/css/lrtk.css" type="text/css" />
<meta name="google-site-verification" content="tnYGDmBAeTYS_84U_dz6Z4rWznCb0wFD4XxtBRNM9qg" />
<meta name="msvalidate.01" content="DF7F554D393F4224E25D986A1CDA3066" />
<link rel="shortcut icon" href="//www.w3cschool.cn/statics/images/favicon.ico" />
</head>
<body id="homefirst" class="index-body">
<!--引入头部-->
<!--header start-->
<div id="header_item">
<div id="header_index">
<div id="header-l">
<a href="http://www.w3cschool.cn" title="w3cschool"><img
src="http://statics.w3cschool.cn/images/w3c/logo.png" alt="W3Cschool" /></a>
<ul class="header-menu">
<li><a href="http://www.w3cschool.cn/tutorial" title="编程入门教程">编程入门教程</a></li>
<li><a href="http://www.w3cschool.cn/manual" title="开源技术文档">开源文档</a></li>
<li><a href="http://www.w3cschool.cn/examples" title="编程实例">编程实例</a></li>
<li><a href="http://www.w3cschool.cn/topic/" title="问答">问答</a></li>
<li><a href="http://www.w3cschool.cn/position/positionList" title="职位进阶">职位进阶<!—<i class="new-icon">
</i>--></a></li>
</ul>
</div>
```


指向了其他网页的链接信息

存储 HTML 还是存储文本内容？

用户搜索的一般是文本内容，那是否可以只存储文本内容呢？

存储 HTML 还是存储文本内容？

用户搜索的一般是文本内容，那是否可以只存储文本内容呢？

存 HTML，文本信息在不同的位置权重不同

标题和正文的权重不一样

且还需要保存 <a href=”/course/”> 这样的链接信息

爬虫的模型

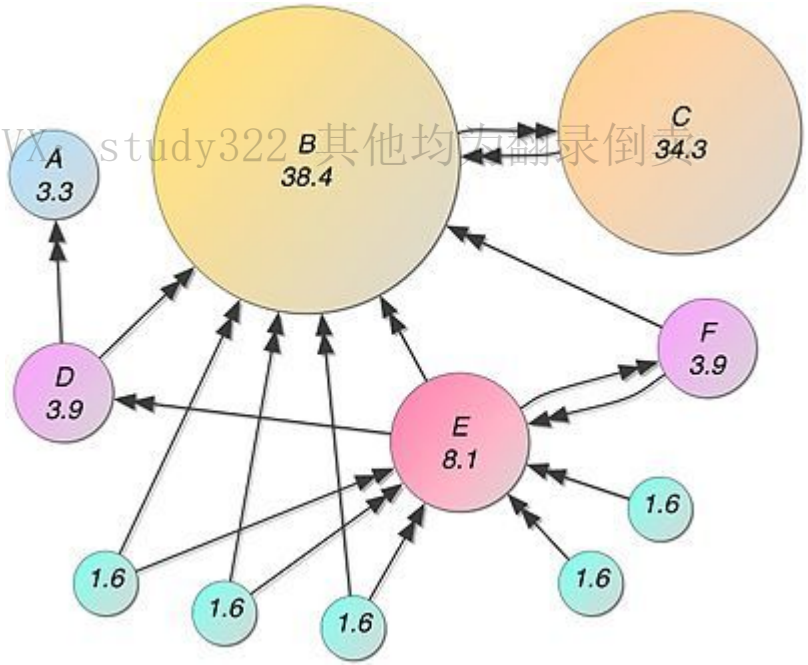
URL(链接)可以认为是图中的节点

根据URL抓取下来的网页中的其他URL即体现了URL之间的关联性，可以看做一条图中的有向边

Page Rank

Google 的立身之本

被更多其他网页指向的网页，具有更高的价值



从哪些网页开始爬取？

选择哪些 URL 作为我们的种子网页

从哪些网页开始爬取？

选择哪些 URL 作为我们的种子网页

种子链接 (seed urls) 通常是一些新闻类网站

或者 Alexa 上的 Top 100 sites

爬取目标

Google: 60 trillion web pages in the world

一个月之内将全世界所有网页抓取一次(20m webpages per sec)

存储下所有网页需要 600pb (10k per webpage)

爬取目标(面试要求)

10天之内抓取下 1B 网页 (1k webpages per sec)

需要 10T 的存储 (10k per webpage)

Service 服务

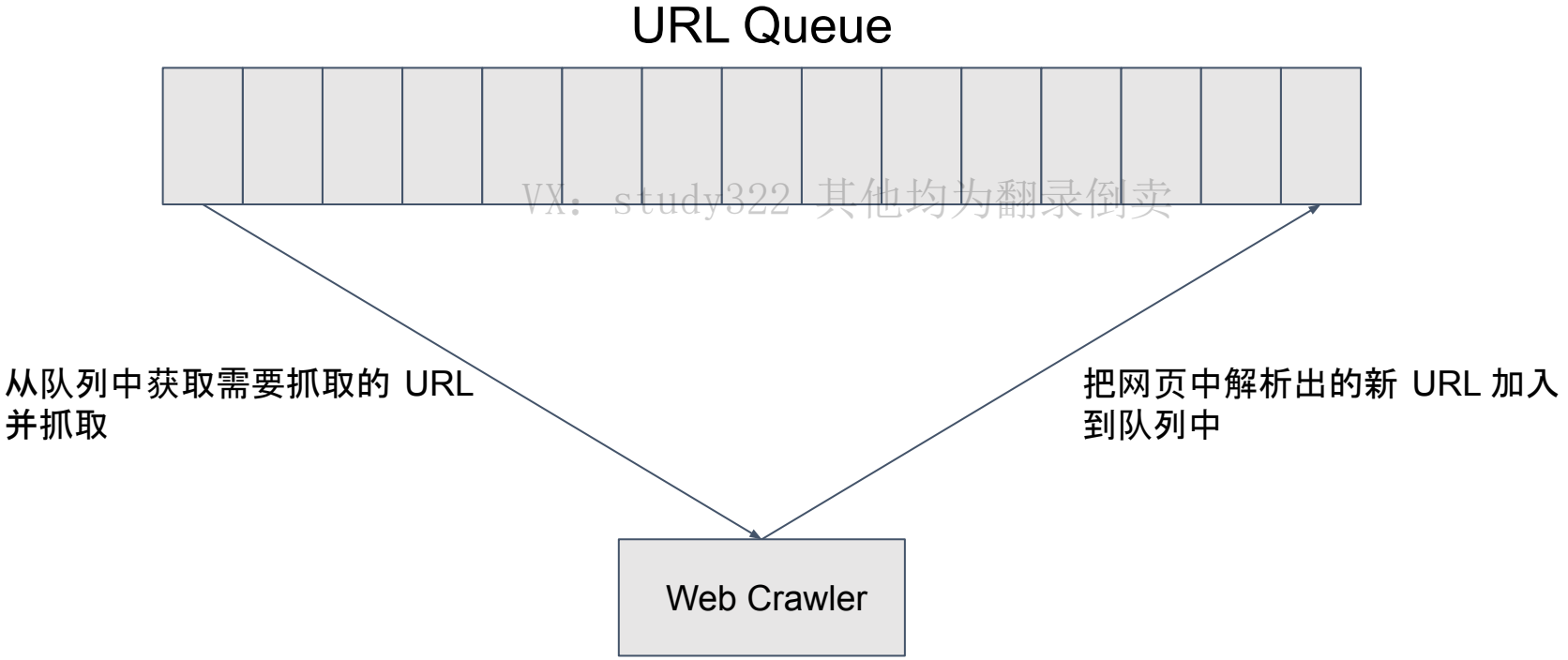
在一个基础版的 Web Crawler 中我们只需要一个 Crawler Service

使用什么算法进行爬取？

DFS or BFS?

生产者消费者模型

在 Web Crawler 中，Crawler 即是 Producer 又是 Consumer



单进程(single process)爬虫是否可行？

最大瓶颈是什么？

单进程(single process)爬虫是否可行？

最大瓶颈是什么？

single process 会因为 network 的原因大部分时间处于 idle 状态

一般来说，平均 download 一篇 webpage 需要 2s

那么 single process 的性能只能做到 0.5 webpage / s

是否进程数越多越好？

既然一个 process 可以做到 0.5 webpage / s

是不是 2k 个 processes 就可以做到 1k webpages / s ？

是否进程数越多越好？

既然一个 process 可以做到 0.5 webpage / s

是不是 2k 个 processes 就可以做到 1k webpages / s ？

不行，过多的 context switch 会导致 CPU 利用率下降

更好的办法是，我们可以用 20 台机器，每台机器启动 100 个 processes，每个 process 单独执行一个爬虫程序

Storage 存储

网页如何存储？

BFS 中的 Queue 如何存储？

BFS 中的 HashSet 如何存储？

网页如何存储

爬虫爬取到的网页，如何存储？

是存在爬虫所在的机器上还是存储在其他地方？

网页如何存储

DFS (Distributed File System)

若是直接在爬虫上存储会使得数据管理混乱

且破坏了爬虫 Stateless 的属性

问：系统设计中还有哪些常见的 stateless 的东西？

BFS 中的队列如何存储？

是直接在内存中开一个 Queue 么？

BFS 中的队列如何存储？

是直接在内存中开一个 Queue 么？

直接在内存中存储会导断电时数据丢失

应该使用 Message Queue，如 Redis, Kafka, RabbitMQ

BFS 中的 HashSet 如何存储？

HashSet 的作用是即避免一个网页被重复抓取

BFS 中的 HashSet 如何存储？

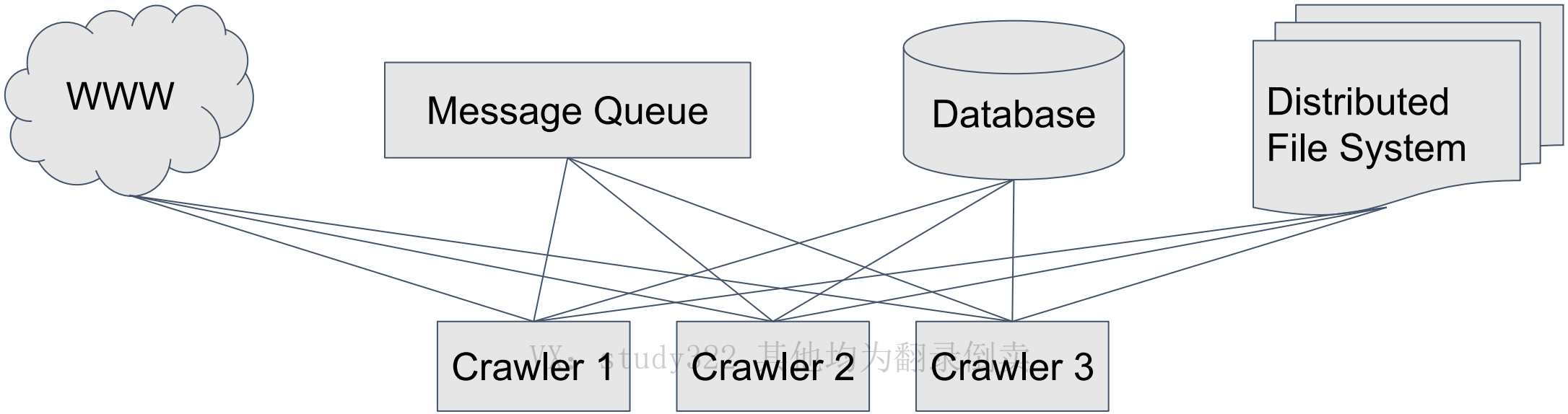
HashSet 的作用是即避免一个网页被重复抓取

存储在数据库中

可以是效率比较高的 key-value 的数据库

除了是否被取过的信息，还可以同时存储其他的一些信息

可行解 Work Solution




```python
def crawler_task(url):
    webpage = http_request.download(url)
    distributed_file_system.save(url, webpage)

    for next_url in extract_urls(webpage):
        if not database.exists(next_url):
            message_queue.add_task('crawler_task', next_url)
            database.insert(next_url)
```

Scale 拓展

设计一个更加 Robust 的爬虫

Robots 协议

又称为爬虫协议

https://www.zhihu.com/robots.txt

有很多网站的 robots 协议中会限制爬虫的访问频率

Robots 协议不是一个强制协议，是一个软性约定

robots.txt

https://www.zhihu.com/robots.txt

```
User-agent: Googlebot
Disallow: /login
Disallow: /logout
Disallow: /resetpassword
Disallow: /terms
Disallow: /search
Disallow: /notifications
Disallow: /settings
Disallow: /inbox
Disallow: /admin_inbox
Disallow: /*?guide*

User-agent: Googlebot-Image
Disallow: /login
Disallow: /logout
Disallow: /resetpassword
Disallow: /terms
Disallow: /search
Disallow: /notifications
Disallow: /settings
Disallow: /inbox
Disallow: /admin_inbox
Disallow: /*?guide*

User-agent: EasouSpider
Request-rate: 1/2 # load 1 page per 2 seconds
Crawl-delay: 10
Disallow: /login
Disallow: /logout
Disallow: /resetpassword
Disallow: /terms
Disallow: /search
Disallow: /notifications
Disallow: /settings
Disallow: /inbox
Disallow: /admin_inbox
Disallow: /*?guide*
```



```
User-agent: MSNBot
Request-rate: 1/2 # load 1 page per 2 seconds
Crawl-delay: 10
Disallow: /login
Disallow: /logout
Disallow: /resetpassword
Disallow: /terms
Disallow: /search
Disallow: /notifications
Disallow: /settings
Disallow: /inbox
Disallow: /admin_inbox
Disallow: /*?guide*

User-Agent: *
Disallow: /
```

如何限制爬虫访问某个网站的频率？

单纯的使用先进先出的 Queue 会使得一个网站短时间内被抓取次数过多，从而导致爬虫被封等问题

如何限制爬虫访问某个网站的频率？

让 Crawler 只做 Consumer，不负责产生新的抓取任务

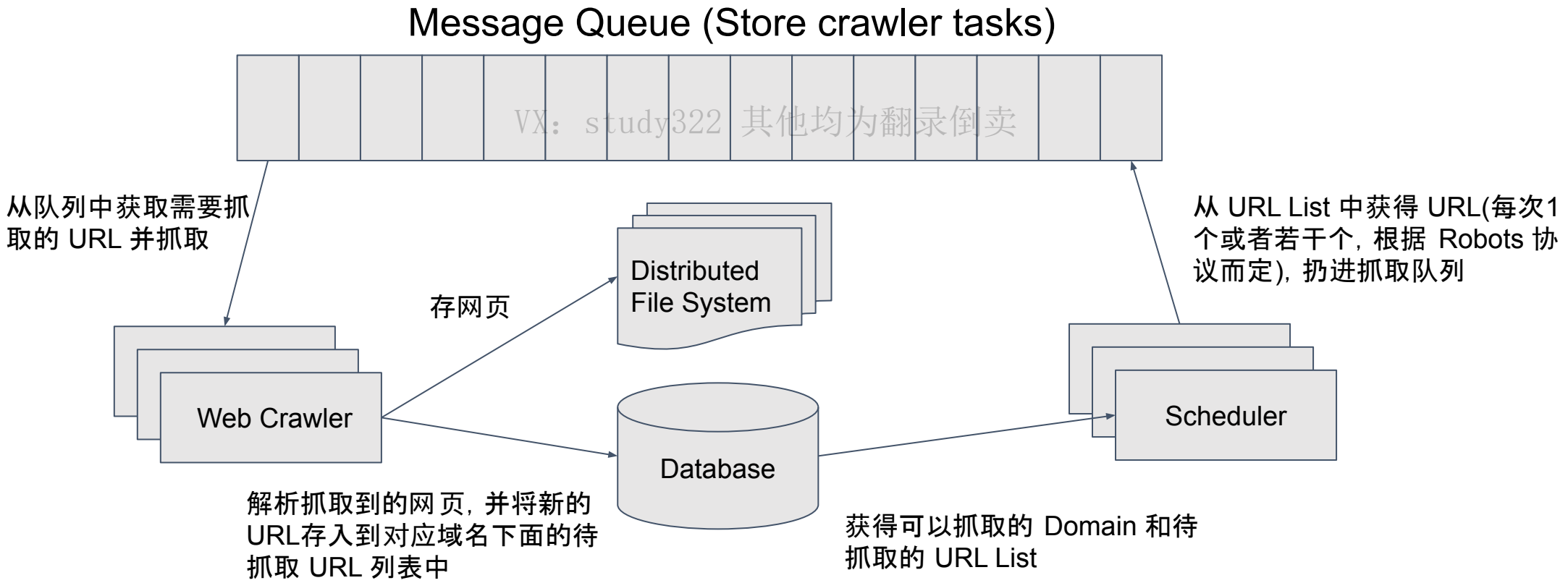
新增一个 Scheduler (Producer) 负责调度和生产抓取任务

在 Database 中记录每个网站下一次可以友好抓取的时间

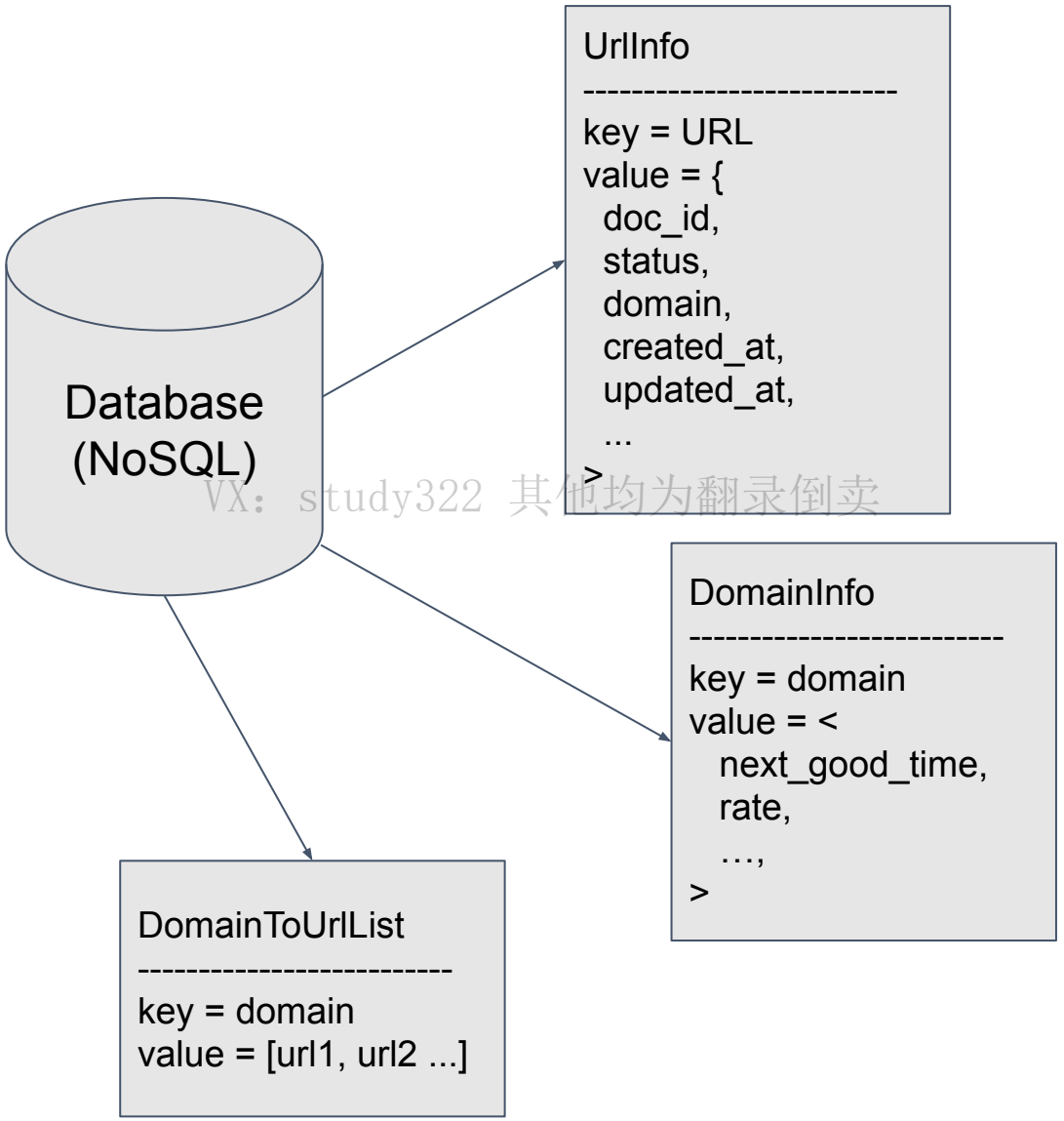
DB 中增加 key=domain value=url_list 的存储结构，存放每个域名下面待抓取的 URL List

DB 中增加每个 domain 下次什么时候可以被抓取的信息记录

Scheduler 的代码，循环遍历每个 domain，遇到可以抓取的 domain，就把其中的一个 url 丢到抓取队列中



Database 中存储的数据



伪代码(主要看流程和设计，不用管具体的算法细节)


```python
# crawler code
# 只负责抓取网页，解析新 URLs，
# 将新的 URLs 存储到所属 domain 的 URL List 里
def crawler_task(url):
    webpage = http_request.download(url)
    distributed_file_system.save(url, webpage)

    for next_url in extract_urls(webpage):
        if not database.url_existed(next_url):
            domain = fetch_domain(next_url)
            database.append_url(domain=domain, url=next_url)

# scheduler code
# 无线循环，不断的检查是否有可以抓取的 domain 和 urls
def scheduler(scheduler_id):
    while True:
        for domain in database.filter_domains(key=scheduler_id):
            if not good_time_to_crawl(domain):
                continue
            # 这里取一个 url，但如果 robots 协议允许
            # 也可以多取几个，但一般不会把 Pending 的 Urls 全都取出来
            url = database.get_url_from(domain)
            message_queue.add_task('crawler_task', url)
            database.update_next_available_crawl_time(domain)
```

在中国抓取美国的网页会比较慢

类似的问题：为什么 Google 退出中国之后要在香港部署服务器？

在中国抓取美国的网页会比较慢

在不同的地区部署 Crawler 系统

每个地区只抓取所在地区被分配的 domain 下的网页

可以通过 domain 的 whois 信息来确定网站所属地区

如何处理网页的更新和失效？

很多时候网页会不断更新，爬虫也需要不断的抓取同一个网页

有的时候可能网站挂了一小段时间，此时网页正好无法被抓取

如何处理网页的更新和失效？

增加对每个 URL 的信息记录

记录下这个 URL 下一次需要被重新抓取的时间

可以通过 Expenential Backoff 的方式计算

Exponential Backoff

如何处理网页更新

- URL 抓取成功以后，默认 1 小时以后重新抓取
- 如果 1 小时以后抓取到的网页没有变化，则设为 2 小时以后再次抓取
  - 2小时以后还是没有变化，则设为 4 小时以后重新抓取，以此类推
- 如果 1 小时以后抓取到的网页发生变化了，则设为 30 分钟以后再次抓取
  - 30分钟以后又变化了，设为 15 分钟以后重新抓取。
- 设置抓取时间的上下边界，如至少 30 天要抓取一次，至多 5 分钟抓取一次

如何处理网页失效

- URL 抓取失效以后，默认 1 小时以后重新抓取
- 如果 1 小时以后依然抓不到，则设置为 2 小时
- 其他步骤类似网页更新的情况

如何使用 AWS 的机器 40 小时抓取 2.5 亿网页

http://www.michaelnielsen.org/ddi/how-to-crawl-a-quarter-billion-webpages-in-40-hours/

Design Typeahead

Typeahead vs Google Suggestion

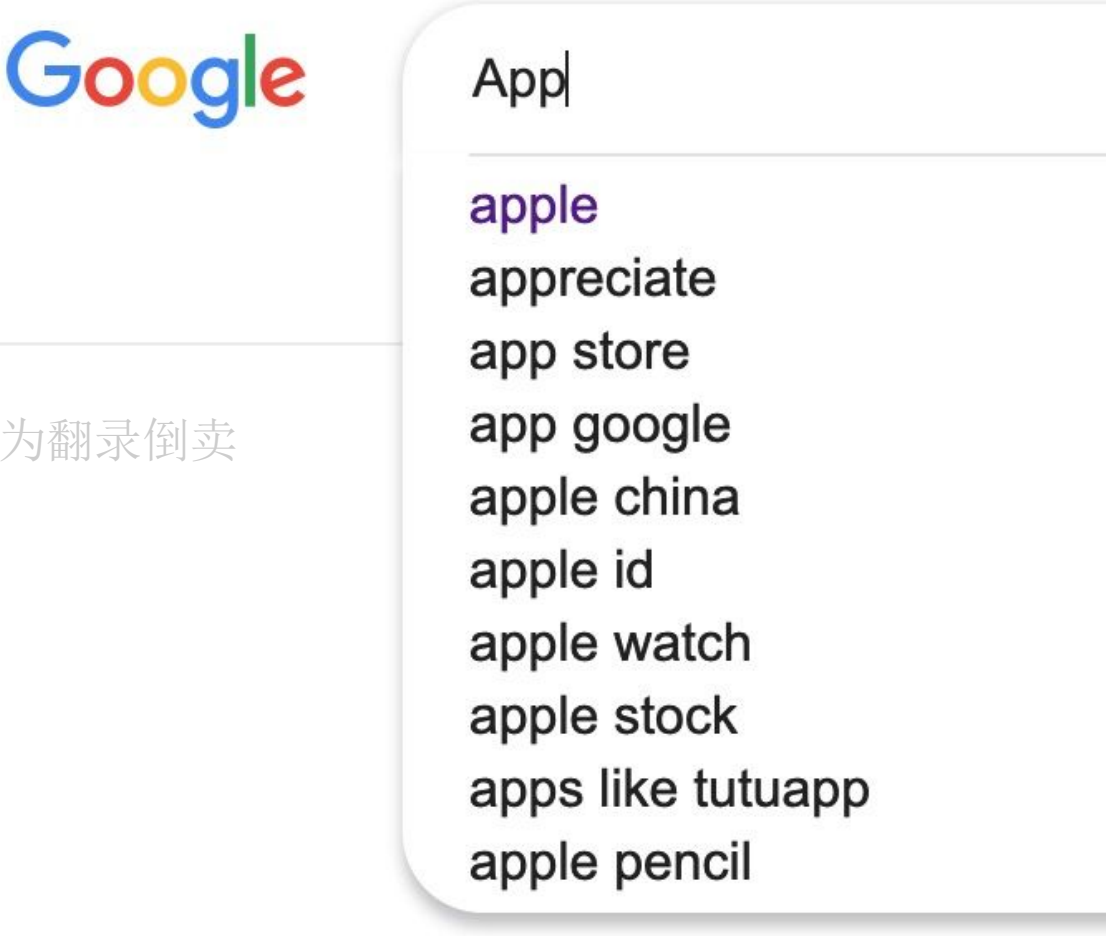
Typeahead.js 是 Twitter 开源的一个前端插件

支持输入一个前缀后，返回匹配这个前缀的 items

Google Suggestion 是搜索时提供的 Query 建议

是一个后端的系统

在系统设计中通常都是让你设计后端系统



Scenario 场景

定义输入和输出

推算 QPS

输入与输出

用户输入一段想要搜索的内容的前缀，返回可能匹配的 Top 10 Suggest Queries

尽量返回被其他人搜索得较多的 Queries

推算 QPS

DAU = 500M

假设每天每人搜索 6 次，每次平均输入 10 个字符

用户每输入一个字符，都会访问一次 Suggestion API 来获得 Top 10 Queries

Average QPS = 500M * 6 * 10 / 86400 = 30B / 86400 ~ 340k

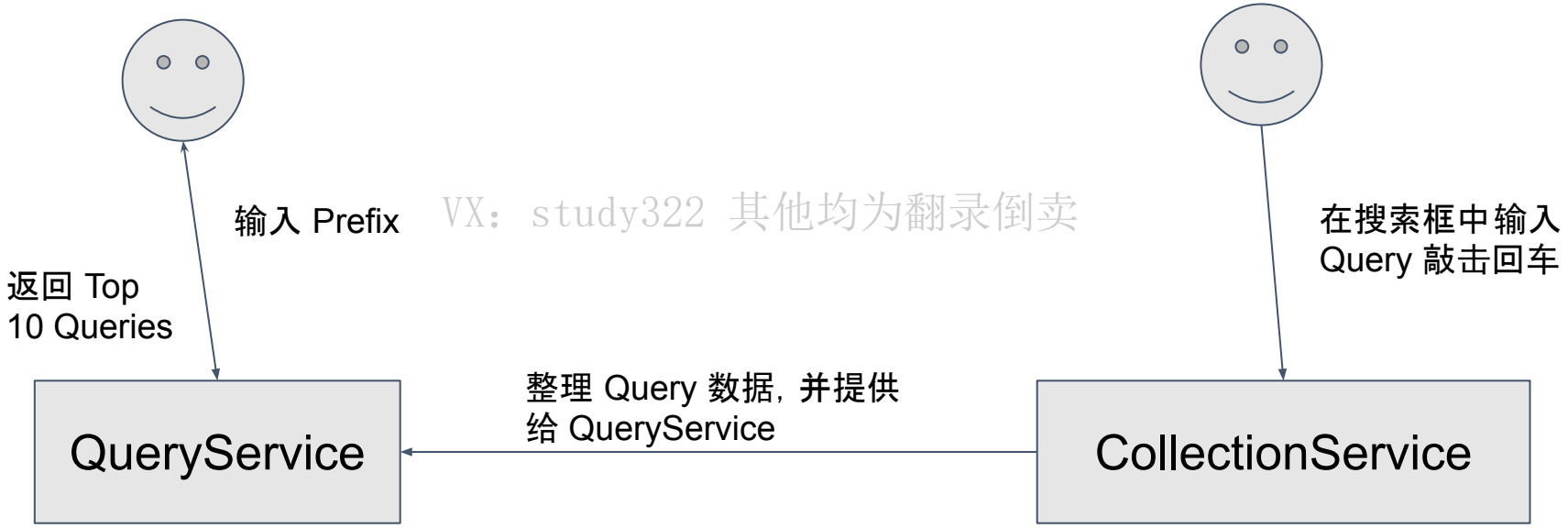
Peak QPS ~ Average QPS * 3 ~ 1M

Service 服务

QueryService: 提供 Top 10 Queries 的查询

CollectionService: 记录用户的 Queries 提供给 QueryService

Service 服务



问：为什么不是 QueryService 请求 CollectionService 来实时计算 Top 10？

Storage 存储

QueryService - 什么结构支持前缀查询？

Storage 存储

QueryService - 什么结构支持前缀查询？

Trie / Prefix Tree

有支持 Trie 的数据库么？

前缀树 Trie / Prefix Tree

- 好处：
  - 节省空间
- 坏处：
  - 没有现成的支持该结构的数据库
- 在 LintCode 上练习 Trie，了解 Trie 的实现原理
  - https://www.lintcode.com/problem/implement-trie/

哈希表 HashTable / HashMap

- 好处：
  - 现成的 Key-value Storage 很多，如 RocksDB, Redis
- 坏处：
  - 空间耗费相对于 Trie 稍大

如果使用 Key-value Storage

key 可以是用户输入的 query prefix

那么 value 是什么？

如果使用 Key-value Storage

key 可以是用户输入的 query prefix

那么 value 是什么？

Top 10 Queries

如何计算 Top 10 Queries

CollectionService 负责统计每个 Query 的搜索次数

定期遍历所有 Queries，将每个 Query 通过打擂台的方式加入到其所有 Prefix 的 Top 10 Queries 中

- 如 apple 这个词如果被搜索了 1b 次
- 那 apple 需要分别被加入到 a, ap, app, appl, apple 这5个 prefix 的 Top 10 Queries 中
- 如果某个 prefix 已经存在了被搜索次数更多的其他 10 个 Queries，就无需再加入了

最后相当于我们需要存储下这样一些 key-value：

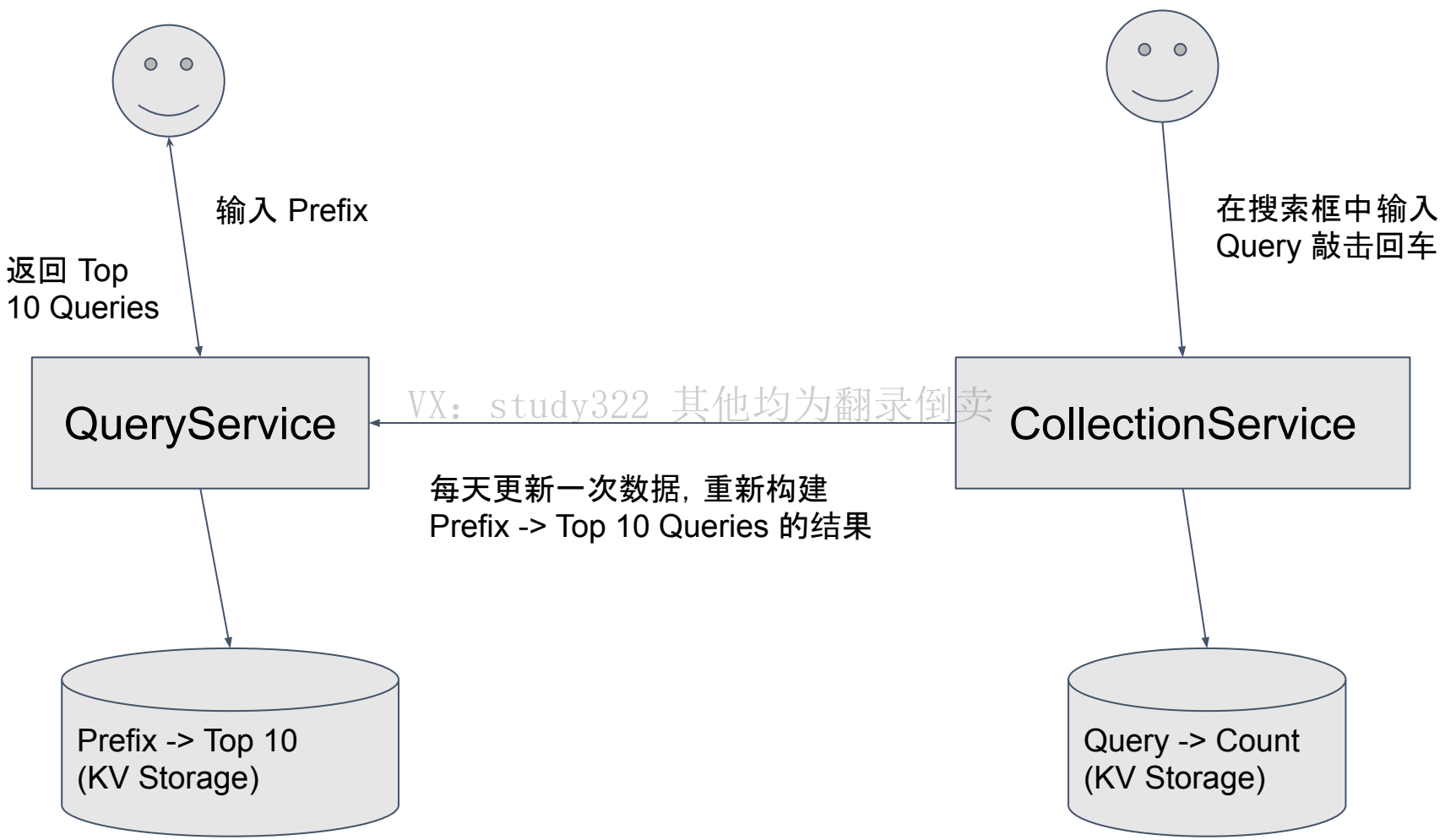
- a: [“amazon”, “aws”, “air china”, “apple”, “airbnb”, “adidas”, ...]
- ap: [“apex”, “apple china”, “apple id”, …]
- …

Storage 存储

CollectionService - 收集每个 Query 被搜索的次数

也可以使用 key-value Storage, key=query, value=被搜索次数

可行解 Work Solution



Scale 拓展

Make it robust!

如何优化 CollectionService

并不是每条 Query 都会成为 Top 10

会浪费很多存储在记录永远不会成为 Top 10 的 Queries 上

如何优化 CollectionService

不记录所有的 Queries，以 1/10000 的概率来记录

即 if get_random(10000) == 0 则对应的 Query 计数+1

否则就直接扔掉

因为我们不关心具体的 Query 次数，只需要一个相对的排名

该是 Top 10 的还是 Top 10

优化 Prefix → Top 10 的构建速度

循环遍历每一个 Query 然后打擂台的方式非常慢

假设计算资源足够，有什么办法可以优化效率？

优化 Prefix → Top 10 的构建速度

循环遍历每一个 Query 然后打擂台的方式非常慢

假设计算资源足够，有什么办法可以优化效率？

Map Reduce

Map: <apple: 1b> → <a: apple,1b> <ap: apple, 1b> …

Reduce: 遍历同一个 Prefix 下的 Queries, 筛选 Top 10

用户输入速度很快怎么办

没有必要都获取 a, ap, app, appl, apple 的 top 10 queries

用户输入速度很快怎么办

没有必要都获取 a, ap, app, appl, apple 的 top 10 queries

在 frontend 设置一个 delay

当用户停止输入超过 200ms 时，才发送请求

如何优化响应速度

要应对 1M+ 的 QPS

我们还需要进一步极尽所能的优化这个系统的响应速度

Backend Cache

每次更新 prefix -> top 10 的数据时都主动更新 Backend Cache

避免更新带来的 Cache miss

Frontend-cache & Prefetch

Frontend-cache 即在前端进行缓存

Pre-fetch 即预加载一些用户可能搜索的 prefix 和 top 10

使用 Frontend-cache / Pre-fetch 的方法避免过多的 round-trip

前端缓存 Frontend-cache

自 H5 以后 Web Browser 中有 local Storage 可以用来作为前端缓存的 cache(key-value storage)

用户曾经输入过的 prefix 和对应的 top 10 queries 都可以在这里被 cache 起来，避免重复检索

可以通过设置 expiration 的方式来避免 top 10 queries 是一些过时的结果

预加载 Pre-fetch

如用户在输入 ap 以后，极有可能想要搜索的是以 app 开头的 query

因此可以假设用户之后会输入 app 的情况下先将 app 的 top 10 queries 也一起返回到前端并 cache 起来

预加载并不会减少后端系统内部的查询次数(且反而会增加)，但是可以减少前端和后端之间的通信次数。一般来说，后端系统内部的通信要远快于用户浏览器端(前端)与后端的通信。因此这个优化是值得的。

如何获得实时热门 Queries

每天重新计算一次 Prefix → Top 10 则可能会有一些延迟

且一些短期内的热门的搜索应该获得更高的权重

如何获得实时热门 Queries

构建一套一样的系统，只是查询的内容是最近 2 小时内的热门搜索

这套系统每 2 小时更新一次数据

用户的请求需要汇总普通搜索结果和热门搜索结果

汇总时可以用一些算法来提高热门搜索的权重(2小时内被搜索的次数肯定会员小于历史被搜索次数)

网页搜索结果的展示也有类似的架构