



下载APP



## 04 | 网页爬虫设计：如何下载千亿级网页？

2022-02-23 李智慧

《李智慧·高并发架构实战课》

课程介绍 >



讲述：李智慧

时长 20:54 大小 19.15M



你好，我是李智慧。

在互联网早期，网络爬虫仅仅应用在搜索引擎中。随着大数据时代的到来，数据存储和计算越来越廉价和高效，越来越多的企业开始利用网络爬虫来获取外部数据。例如：获取政府公开数据以进行统计分析；获取公开资讯以进行舆情和热点追踪；获取竞争对手数据以进行产品和营销优化等等。

网络爬虫有时候也被称为网络机器人，或者网络蜘蛛。我们准备开发一个全网爬虫，爬取全（中文）互联网的公开网页，以构建搜索引擎和进行数据分析。爬虫名称为“Bajie（八戒）”。



Bajie 的技术挑战包括：如何不重复地获取并存储全网海量 URL？如何保证爬虫可以快速爬取全网网页但又不会给目标网站带来巨大的并发压力？接下来我们就来看看 Bajie 的需求与技术架构。

## 需求分析

Bajie 的功能比较简单，这里不再赘述。

## 性能指标估算

因为互联网网页会不断产生，所以全网爬虫 Bajie 也是一个持续运行的系统。根据设计目标，Bajie 需要每个月从互联网爬取的网页数为 20 亿个，平均每个页面 500KB，且网页需存储 20 年。

Bajie 的存储量和 TPS（系统吞吐量）估算如下。

### 每月新增存储量

估计平均每个页面 500KB，那么每个月需要新增存储 1PB。

$$20\text{亿} \times 500KB = 1PB$$

### 总存储空间

网页存储有效期 20 年，那么需要总存储空间 240PB。

$$1PB \times 12\text{个月} \times 20\text{年} = 240PB$$

### TPS

Bajie 的 TPS 应为 800。

$$20\text{亿} \div (30 \times 24 \times 60 \times 60) \approx 800$$

## 非功能需求

Bajie 需要满足的非功能需求如下。

1. 伸缩性：当未来需要增加每月爬取的网页数时，Bajie 可以灵活部署，扩大集群规模，增强其爬取网页的速度。也就是说，Bajie 必须是一个分布式爬虫。
2. 健壮性：互联网是一个开放的世界，也是一个混乱的世界，服务器可能会宕机，网站可能失去响应，网页 HTML 可能是错误的，链接可能有陷阱.....所以 Bajie 应该能够面对各种异常，正常运行。
3. 去重：一方面需要对超链接 URL 去重，相同的 URL 不需要重复下载；另一方面还要对内容去重，不同 URL 但是相同内容的页面也不需要重复存储。
4. 扩展性：当前只需要爬取 HTML 页面即可，将来可能会扩展到图片、视频、文档等内容页面。

此外，Bajie 必须是“礼貌的”。爬虫爬取页面，实际上就是对目标服务器的一次访问，如果高并发地进行访问，可能会对目标服务器造成比较大的负载压力，甚至会被目标服务器判定为 DoS 攻击。因此 Bajie 要避免对同一个域名进行并发爬取，还要根据目标服务器的承载能力增加访问延迟，即在两次爬取访问之间，增加等待时间。


并且，Bajie 还需要遵循互联网爬虫协议，即目标网站的 robots.txt 协议，不爬取目标网站禁止爬取的内容。比如 www.zhuhu.com 的 robots.txt 内容片段如下。

[复制代码](#)

```
1 User-agent: bingbot
2 Disallow: /appview/
3 Disallow: /login
4 Disallow: /logout
5 Disallow: /resetpassword
6 Disallow: /terms
7 Disallow: /search
8 Allow: /search-special
9 Disallow: /notifications
10 Disallow: /settings
11 Disallow: /inbox
12 Disallow: /admin_inbox
13 Disallow: /*?guide*
```

Zhihu 约定 Bing 爬虫可以访问和不可以访问的路径都列在 robots.txt 中，其他的 Google 爬虫等也在 robots.txt 中列明。

robots.txt 还可以直接禁止某个爬虫，比如淘宝就禁止了百度爬虫，淘宝的 robots.txt 如下。

 复制代码

```
1 User-agent: Baiduspider
2 Disallow: /
3 User-agent: baiduspider
4 Disallow: /
```

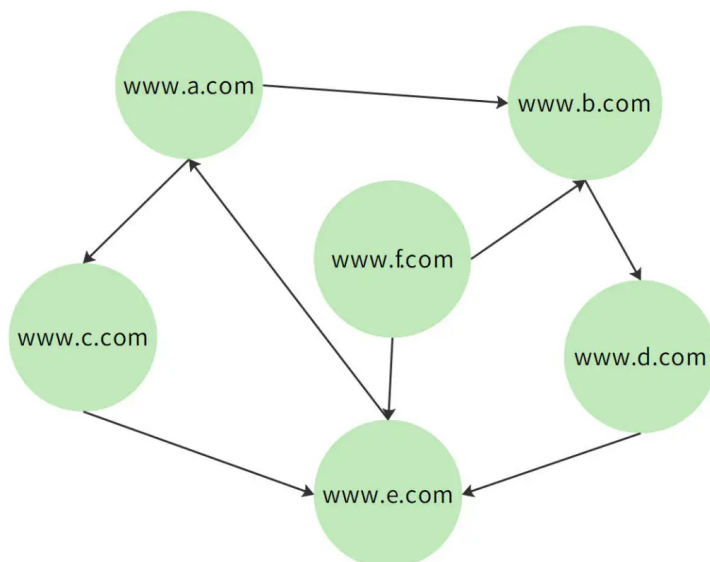
淘宝禁止百度爬虫访问根目录，也就是禁止百度爬取该网站所有页面。

robots.txt 在域名根目录下，如 `www.taobao.com/robots.txt`。Bajie 应该首先获取目标网站的 robots.txt，根据爬虫协议构建要爬取的 URL 超链接列表。

## 概要设计

Bajie 的设计目标是爬取数千亿的互联网网页，那么 Bajie 首先需要得到这千亿级网页的 URL，该如何获得呢？

全世界的互联网页面事实上是一个通过超链接连接的巨大网络，其中每个页面都包含一些指向其他页面的 URL 链接，这些有指向的链接将全部网页构成一个有向（网络）图。如下图所示，每个节点是一个网页，每条有向的边就是一个超链接。

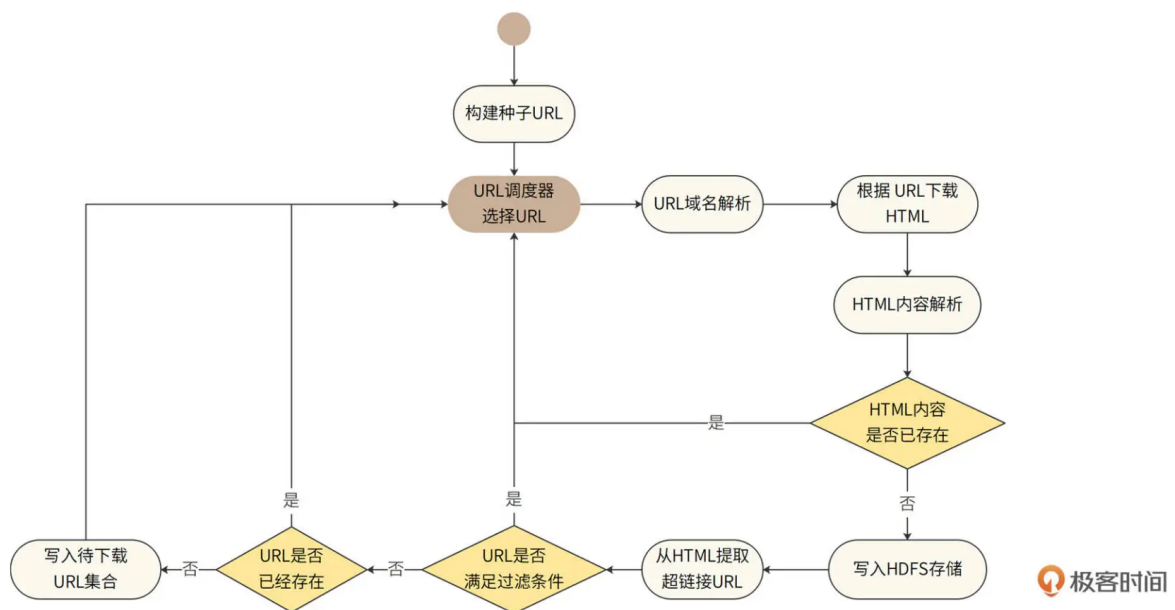
 极客时间

上图中，`www.a.com` 包含两个超链接，分别是 `www.b.com` 和 `www.c.com`，对应图中就是节点 `www.a.com` 指向节点 `www.b.com` 和节点 `www.c.com` 的边。同样地，`www.b.com` 节点也会指向 `www.d.com` 节点。

如果从这个图中的某个节点开始遍历，根据节点中包含的链接再遍历其指向的节点，再从这些新节点遍历其指向的节点，如此下去，理论上可以遍历互联网上的全部网页。而**将遍历到的网页下载保存起来**，就是爬虫的主要工作。

所以，Bajie 不需要事先知道数千亿的 URL，然后再去下载。Bajie 只需要知道一小部分 URL，也就是所谓的种子 URL，然后从这些种子 URL 开始遍历，就可以得到全世界的 URL，并下载全世界的网页。

Bajie 的处理流程活动图如下。



极客时间

首先 Bajie 需要构建种子 URL，它们就是遍历整个互联网页面有向图的起点。种子 URL 将影响遍历的范围和效率，所以我们通常选择比较知名的网站的主要页面（比如首页）作为种子 URL。

然后，URL 调度器从种子 URL 中选择一些 URL 进行处理。后面将在详细介绍中说明 URL 调度器的算法原理。

Bajie 对选择出来的 URL 经过域名解析后，下载得到 HTML 页面内容，进而解析 HTML 页面，分析该内容是否已经在爬虫系统中存在。因为在互联网世界中，大约有一分之一的内容是重复的，下载重复的内容就是在浪费计算和存储资源。如果内容已存在，就丢弃该重复内容，继续从 URL 调度器获取 URL；如果不存在，就将该 HTML 页面写入 HDFS 存储系统。

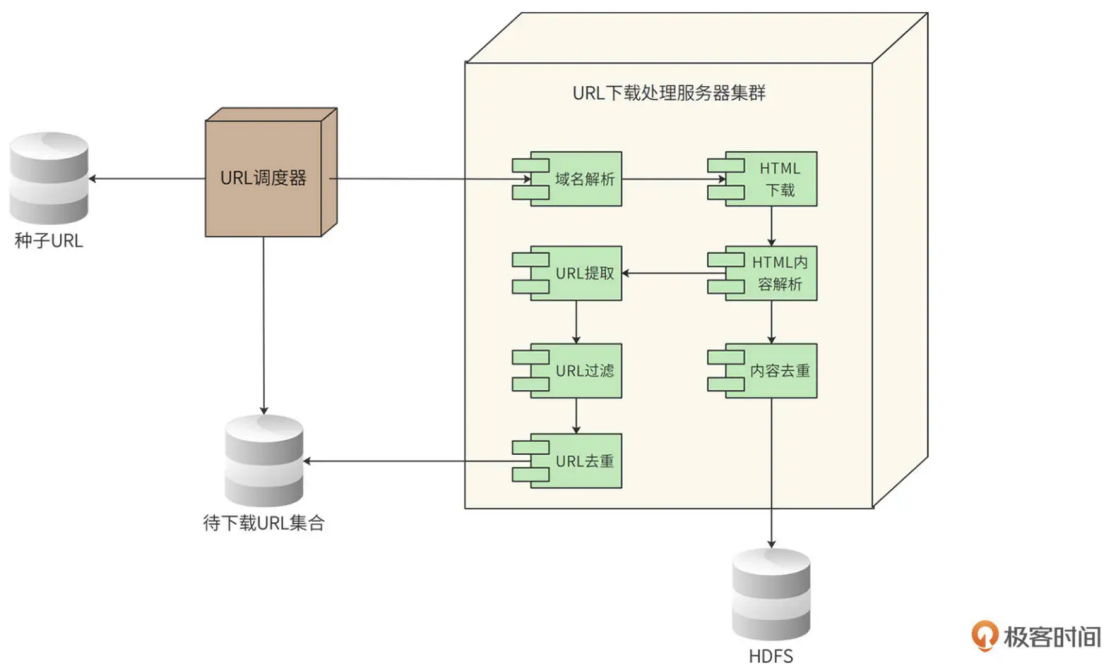
然后，Bajie 进一步从已存储的 HTML 中提取其内部包含的超链接 URL，分析这些 URL 是否满足过滤条件，即判断 URL 是否在黑名单中，以及 URL 指向的目标文件类型是否是爬虫要爬取的类型。

如果 HTML 中的某些 URL 满足过滤条件，那么就丢弃这些 URL；如果不满足过滤条件，那么，进一步判断这些 URL 是否已经存在，如果已经存在，就丢弃该 URL，如果不存在，就记录到待下载 URL 集合。URL 调度器从待下载 URL 集合中选择一批 URL 继续上面的处理过程。

这里需要注意，想判断 URL 是否已经存在，就要判断这个 URL 是否已经在待下载 URL 集合中。此外，还需要判断这个 URL 是否已经下载得到 HTML 内容了。只有既不是待下载，也没被下载过的 URL 才会被写入待下载 URL 集合。

可以看到，在爬虫的活动图里是没有结束点的，从开始启动，就不停地下载互联网的页面，永不停息。其中，**URL 调度器是整个爬虫系统的中枢和核心，也是整个爬虫的驱动器**。爬虫就是靠着 URL 调度器源源不断地选择 URL，然后有节奏、可控地下载了整个互联网，所以 **URL 调度器也是爬虫的策略中心**。

据此，Bajie 的部署图如下。



Bajie 系统中主要有两类服务器，一类是 URL 调度器服务器；一类是 URL 下载处理服务器集群，它是一个分布式集群。



URL 调度器从种子 URL 或待下载 URL 集合中载入 URL，再根据调度算法，选择一批 URL 发送给 URL 下载处理服务器集群。这个下载处理服务器集群是由多台服务器组成的，根据需要达到的 TPS，集群规模可以进行动态伸缩，以实现需求中的伸缩性要求。

每台 URL 下载处理服务器先得到分配给自己的一组 URL，再启动多个线程，其中每个线程处理一个 URL，按照前面的流程，调用域名解析组件、HTML 下载组件、HTML 内容解析组件、内容去重组件、URL 提取组件、URL 过滤组件、URL 去重组件，最终将 HTML 内容写入 HDFS，并将待下载 URL 写入待下载 URL 集合文件。

## 分布式爬虫

需要注意的是，URL 下载处理服务器采用分布式集群部署，主要是为了提高系统的吞吐能力，使系统满足伸缩性需求。而 URL 调度器则只需要采用一台高性能的服务器单机部署即可。

事实上，单机 URL 调度器也完全能够满足目前 800TPS 的负载压力，以及将来的伸缩要求。因为 800TPS 对于 URL 调度器而言其实就是每秒产生 800 个 URL 而已，计算压力并不大，单台服务器完全能够满足。

同时 URL 调度器也不需要考虑单服务器宕机导致的可用性问题，因为爬虫并不是一个实时在线系统，如果 URL 调度器宕机，只需要重新启动即可，并不需要多机部署高可用集群。

相对应地，每个 URL 在 URL 下载处理服务器上的计算负载压力要大得多，需要分布式集群处理，也因此大规模爬虫被称为分布式爬虫，Bajie 就是一个分布式爬虫。

## 详细设计

Bajie 详细设计关注 3 个技术关键点：URL 调度器算法、去重算法、高可用设计。

### URL 调度器算法

URL 调度器需要从待下载 URL 集合中选取一部分 URL 进行排序，然后分发给 URL 下载服务器去下载。待下载 URL 集合中的 URL 是从下载的 HTML 页面里提取出来，然后进行过滤、去重得到的。一个 HTML 页面通常包含多个 URL，每个 URL 又对应一个页面，因此，URL 集合数量会随着页面不断下载而指数级增加。

待下载 URL 数量将远远大于系统的下载能力，**URL 调度器就需要决定当前先下载哪些 URL。**

如果调度器一段时间内选择的都是同一个域名的 URL，那就意味着我们的爬虫将以 800 TPS 的高并发访问同一个网站。目标网站可能会把爬虫判定为 DoS 攻击，从而拒绝请求；更严重的是，高并发的访问压力可能导致目标网站负载过高，系统崩溃。这样的爬虫是“不礼貌”的，也不是 Bajie 的设计目标。

前面说过，网页之间的链接关系构成一个有向图，因此我们可以按照图的遍历算法选择 URL。图的遍历算法有深度优先和广度优先两种，深度优先就是从一个 URL 开始，访问网页后，从里面提取第一个 URL，然后再访问该 URL 的页面，再提取第一个 URL，如此不断深入。

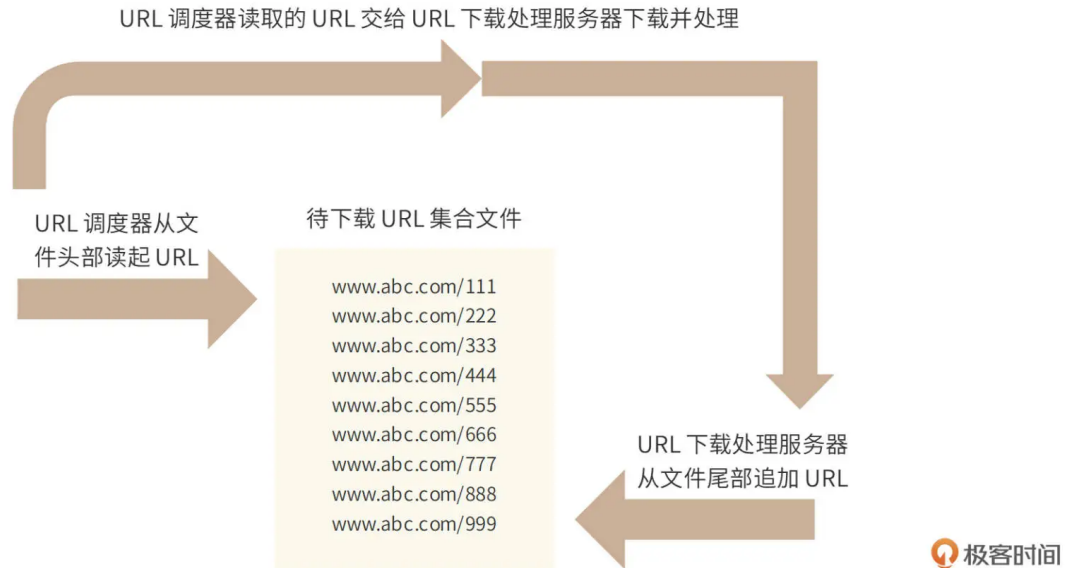
深度优先需要维护较为复杂的数据结构，而且太深的下载深度导致下载的页面非常分散，不利于我们构建搜索引擎和数据分析。所以我们没有使用深度优先算法。

那广度优先算法如何呢？广度优先就是从一个 URL 开始，访问网页后，从中得到 N 个 URL，然后顺序访问这个 N 个 URL 的页面，然后再从这 N 个页面中提取 URL，如此不断深入。显然，广度优先实现更加简单，获取的页面也比较有关联性。

图的广度优先算法通常采用**队列**来实现。首先，URL 调度器从队列头出队列（dequeue）取一个 URL，交给 URL 下载服务器，下载得到 HTML，再从 HTML 中提取得到若干个 URL 入队列（enqueue）到队列尾，URL 调度器再从队列头出队列（dequeue）取一个 URL.....如此往复，持续不断地访问全部互联网页，这就是互联网的广度优先遍历。

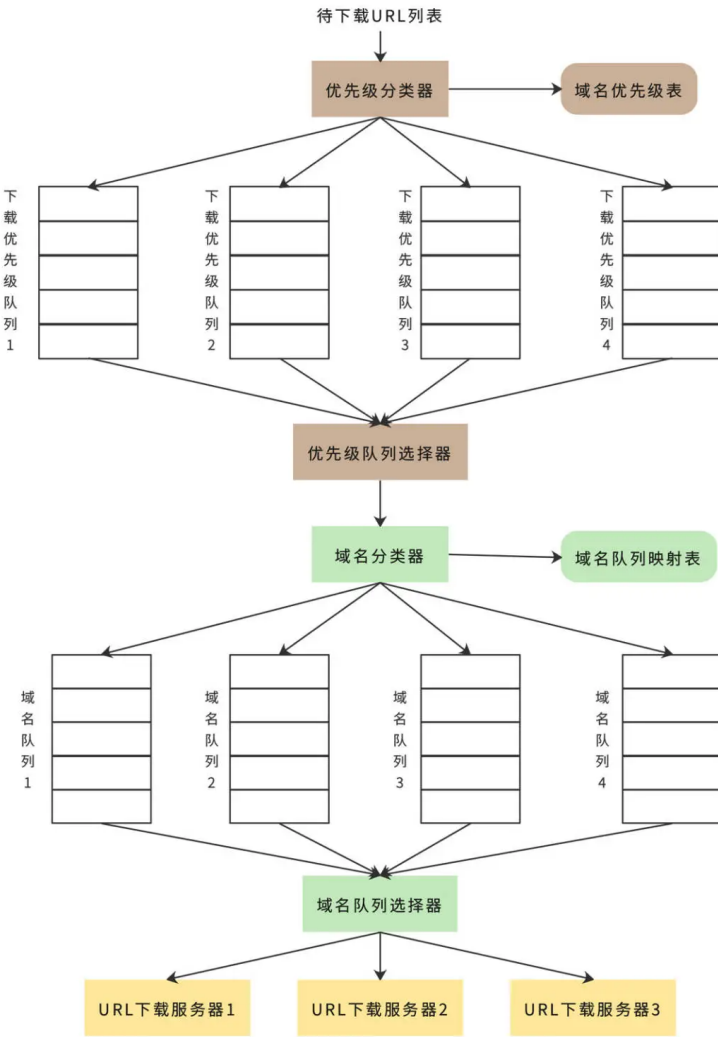
事实上，由于待下载 URL 集合存储在文件中，URL 下载服务器只需要向待下载 URL 集合文件尾部追加 URL 记录，而 URL 调度器只需要从文件头顺序读取 URL，这样就天然实现了先进先出的广度优先算法，如下图。





但是，广度优先搜索算法可能会导致爬虫一段时间内总是访问同一个网站，因为一个 HTML 页面内的链接常常是指向同一个网站的，这样就会使爬虫“不礼貌”。

通常我们针对一个网站，一次只下载一个页面，所以 URL 调度器需要将待下载 URL 根据域名进行分类。此外，不同网站的信息质量也有高低之分，爬虫应该优先爬取那些高质量的网站。优先级和域名都可以使用不同队列来区分，如下图。



首先优先级分类器会根据网页内容质量将域名分类（后面专栏会讲 PageRank 质量排名算法），并为不同质量等级的域名设置不同的优先级，然后将不同优先级记录在“域名优先级表”中。

接下来，按照广度优先算法，URL 列表会从待下载 URL 集合文件中装载进来。根据“域名优先级表”中的优先级顺序，优先级分类器会将 URL 写入不同的队列中。

下一步，优先级队列选择器会根据优先级使用不同的权重，从这些优先级队列中随机获取 URL，这样使得高优先级的 URL 有更多机会被选中。而被选中的 URL 都会交由域名分类器进行分类处理。域名分类器的分类依据就是“域名队列映射表”，这个表中记录了不同域名对应的队列。所以域名分类器可以顺利地将不同域名的 URL 写入不同的域名队列中。

最后，域名队列选择器将轮询所有的域名队列，从其中获得 URL 并分配给不同的 URL 下载服务器，进而完成下载处理。

## 去重算法

爬虫的去重包括两个方面，一个是 URL，相同 URL 不再重复下载；一个是内容，相同页面内容不再重复存储。去重一方面是提高爬虫效率，避免无效爬取；另一方面提高搜索质量，避免相同内容在搜索结果中重复出现。URL 去重可以使用**布隆过滤器**以提高效率。

内容去重首先要判断内容是否重复，由于爬虫存储着海量的网页，如果按照字符内容对每一个下载的页面都去和现有的页面比较是否重复，显然是不可能的。

Bajie 计算页面内容的 MD5 值，通过判断下载页面的内容 MD5 值是否已经存在，判断内容是否重复。

如果把整个 HTML 内容都计算 MD5，那么 HTML 中的微小改变就会导致 MD5 不同，事实上，不同网站即使相同内容的页面，也总会改成自己的 HTML 模板，导致 HTML 内容不同。

所以，比较内容重复的时候，需要将 HTML 里面的有效内容提取出来，也就是提取出去除 HTML 标签的文本信息，针对有效内容计算 MD5。更加激进的做法是从有效内容中抽取一段话（比如最长的一句话），计算这段话的 MD5，进而判断重复。

而一个内容 MD5 是否存在，需要在千亿级的数据上查找，如果用 Hash 表处理，计算和内存存储压力非常大，我们将用布隆过滤器代替 Hash 表，以优化性能。

## 高可用设计

Bajie 的可用性主要关注两个方面，一是 URL 调度器或 URL 下载处理服务器宕机，二是下载超时或内容解析错误。

由于 Bajie 是一个离线系统，暂时停止爬取数据的话，不会产生严重的后果，所以 Bajie 并不需要像一般互联网系统那样进行高可用设计。但是当服务器宕机后重启时，系统需要能够正确恢复，保证既不会丢失数据，也不会重复下载。

所以，URL 调度器和 URL 下载处理服务器都需要记录运行时状态，即存储本服务器已经加载的 URL 和已经处理完成的 URL，这样宕机恢复的时候，就可以立刻读取到这些状态数

据，进而使服务器恢复到宕机前的状态。对于 URL 下载处理服务器，Bajie 采用 Redis 记录运行时状态数据。

此外，为了防止下载超时或内容解析错误，URL 下载处理服务器会采用多线程（池）设计。每个线程独立完成一个 URL 的下载和处理，线程也需要捕获各种异常，不会使自己因为网络超时或者解析异常而退出。

## 小结

架构设计是一个权衡的艺术，**不存在最好的架构，只存在最合适的架构**。架构设计的目的是解决各种业务和技术问题，而解决问题的方法有很多种，每一种方法都需要付出各自的代价，同时又会带来各种新的问题。架构师就需要在这些方法中权衡选择，寻找成本最低的、代价最小的、自己和团队最能驾驭得住的解决方案。

因此，架构师也许不是团队中技术最好的那个人，但一定是**对问题和解决方案优缺点理解最透彻**的那个人。很多架构师把高可用挂在嘴上。可是，你了解你的系统的高可用的目的是什么吗？你的用户能接受的不可用下限在哪里？你为高可用付出的代价是什么？这些代价换来的回报是否值得？

我们在 Bajie 的设计中，核心就是 URL 调度器。通常在这样的大规模分布式系统中，核心组件是不允许单点的，也就是不允许单机部署，因为单机宕机就意味着核心功能的故障，也就意味着整个系统无法正常运行。

但是如果 URL 调度器采用分布式集群架构提高可用性，多服务器共同进行 URL 调度，就需要解决数据一致性和数据同步问题，反而会导致系统整体处理能力下降。而 Bajie 采用单机部署的方式，虽然宕机时系统无法正常运行，但是只要在运维上保证能快速重新启动，长期看，系统整体处理能力反而更高。

此外，对于一个千亿级网页的爬虫系统而言，最主要的技术挑战应该是海量文件的存储与计算，这也确实是早期搜索引擎公司们的核心技术。但是，自从 Google 公开自己的大数据技术论文，而 Hadoop 开源实现了相关技术后，这些问题就变得容易很多了。Bajie 的海量文件存储就使用了 Hadoop 分布式文件系统 HDFS，我会在后面的《常见海量数据处理技术回顾》这一讲详细讨论它。

## 思考题

一个设计良好的爬虫需要面对的情况还有很多，你还能想到哪些文中没提及的情况？最好也能和我聊聊对应的设计方案。

欢迎在评论区分享你的思考，或者提出对这个设计文档的评审意见，我们共同进步。

分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

生成海报并分享

👍 赞 2    💡 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇    03 | 短 URL 生成器设计：百亿短 URL 怎样做到无冲突？

## 精选留言 (3)

💬 写留言



林海

2022-02-24

大佬，您今晚吃的酱骨饭嘛，我好像在下沉广场见到您了，没敢上去认🤔

编辑回复: 不要怂，冲冲冲！



hdddh

2022-02-23

老师，后续可否加上物理部署图，可能理解会更深刻一些，比如几个机房，每个机房怎么样的

作者回复: 现实中，会在项目前期做高并发的架构设计与开发，但是不会做高并发的物理部署，一则用户量是逐渐增加的，开始的时候不知道需要部署多少服务器，其次服务器的并发承载能力是压测和不断优化得到的，在设计期并不知道。物理部署图在设计阶段没有意义和价值。

多机房部署也是一种设计，后面有案例专门讨论。



**peter**

2022-02-23

请教老师几个问题啊：

Q1：“可扩展”与“可伸缩”的区别是什么？

“可扩展”是指软件开发方面，“可伸缩”是指部署方面，即增删机器方面，对吗？

Q2：怎么知道目标服务器的负载能力？

Q3：爬虫如果不理会robots.txt，会怎么处理？...

展开 ∨

作者回复: 1 是的

2 压测

3 不理睬没关系，但是爬来的数据就没法用来做公开搜索引擎

4 不知道

5 可以啊，URL调度器产生的URL快慢就可以控制，也依赖下载服务器的处理能力

6 会删除，一般不会丢弃，将来还要分析用的

7 爬虫就是一个HTTP client，各种语言都可以。python多是因为Python在数据处理领域用的多，用到爬虫的地方也多。实际上大规模爬虫开发的重头是下载HTML后的处理过程，Java或者C++的处理能力更强

8 是个文件

9 用HDFS存储

10 域名，多个域名可以合用一个队列。集合中的域名是有限的，列举的完

