

**调度器(Scheduler)**

调度器从引擎接受request并将他们入队，以便之后引擎请求他们时提供给引擎。

**下载器(Downloader)**

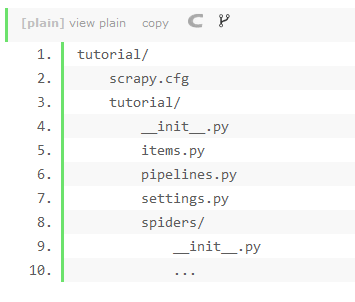
下载器负责获取页面数据并提供给引擎，而后提供给spider。

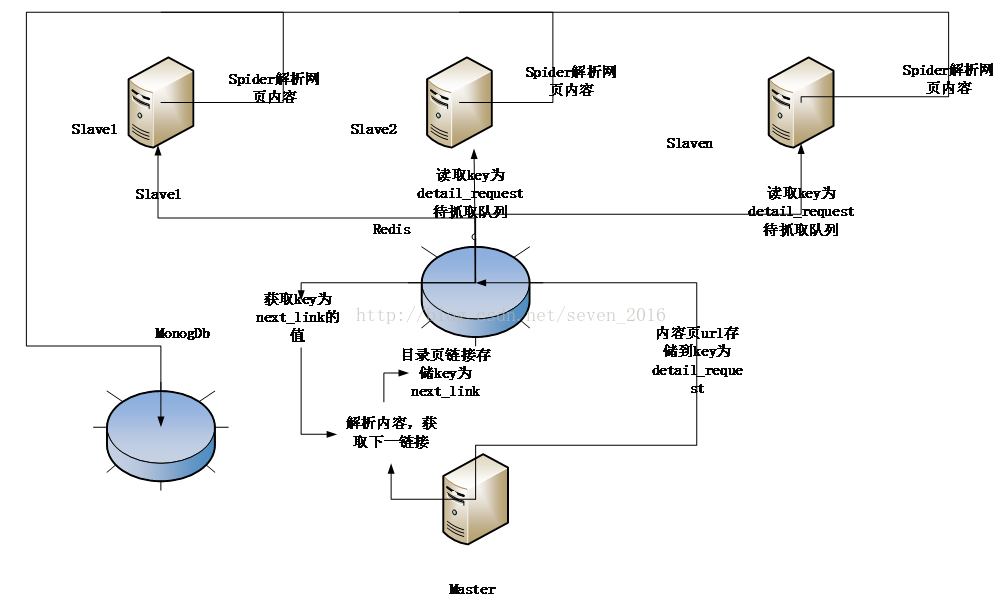
**Spiders**

Spider是Scrapy用户编写用于分析response并提取item(即获取到的item)或额外跟进的URL的类。 每个spider负责处理一个特定(或一些)网站。 更多内容请看 [Spiders](https://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/spiders.html#topics-spiders) 。

**Item Pipeline**

Item Pipeline负责处理被spider提取出来的item。典型的处理有清理、 验证及持久化(例如存取到数据库中)。 更多内容查看 [Item Pipeline](https://scrapy-chs.readthedocs.io/zh_CN/0.24/topics/item-pipeline.html#topics-item-pipeline) 。

* 
* scrapy.cfg：项目的配置文件
* tutorial/：项目的Python模块，将会从这里引用代码
* tutorial/items.py：项目的items文件
* tutorial/pipelines.py：项目的pipelines文件
* tutorial/settings.py：项目的设置文件
* tutorial/spiders/：存储爬虫的目录



　　　　scrapy-redis实现分布式，其实从原理上来说很简单，这里为描述方便，我们把自己的核心服务器称为master，而把用于跑爬虫程序的机器称为slave。

我们知 道，采用scrapy框架抓取网页，我们需要首先给定它一些start\_urls，爬虫首先访问start\_urls里面的url，再根据我们的具体逻辑，对里面的元素、或者是其他的二级、三级页面进行抓取。而要实现分布式，我们只需要在这个starts\_urls里面做文章就行了。

我们在master上搭建一个redis数据库（注意这个数据库只用作url的存储，不关心爬取的具体数据，不要和后面的mongodb或者mysql混淆），并对每一个需要爬取的网站类型，都开辟一个单独的列表字段。通过设置slave上scrapy-redis获取url的地址为master地址。这样的结果就是，尽管有多个slave，然而大家获取url的地方只有一个，那就是服务器master上的redis数据库。

并且，由于scrapy-redis自身的队列机制，slave获取的链接不会相互冲突。这样各个slave在完成抓取任务之后，再把获取的结果汇总到服务器上（这时的数据存储不再在是redis，而是mongodb或者 mysql等存放具体内容的数据库了）

这种方法的还有好处就是程序移植性强，只要处理好路径问题，把slave上的程序移植到另一台机器上运行，基本上就是复制粘贴的事情。

Nutch 是一个开源Java 实现的搜索引擎。它提供了我们运行自己

的搜索引擎所需的全部工具。包括全文搜索和Web爬虫。

Java unix hbase solr

爬虫crawler和查询searcher。Crawler主要用于从网络上抓取网页并为这些网页建立索引。Searcher主要利用这些索引检索用户的查找关键词来产生查找结果。两者之间的接口是索引，所以除去索引部分，两者之间的耦合度很低。

现在Nutch分为两个版本，1.x和2.x。1.x最新版本为1.7，2.x最新版本为2.2.1。两个版本的主要区别在于底层的存储不同。

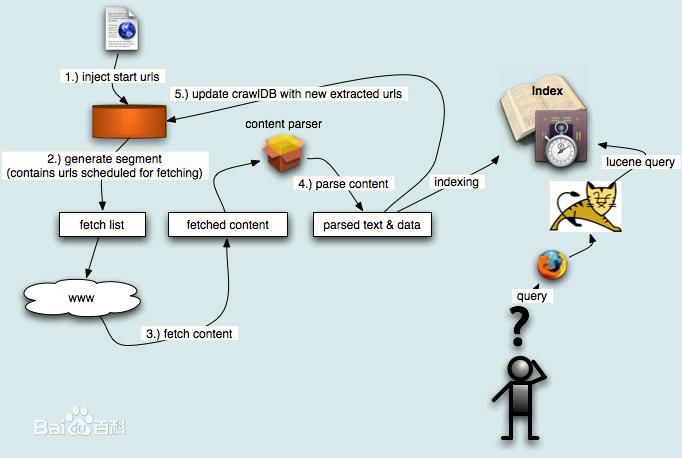
1.x版本是基于Hadoop架构的，底层存储使用的是HDFS，而2.x通过使用Apache Gora，使得Nutch可以访问HBase、Accumulo、Cassandra、MySQL、DataFileAvroStore、AvroStore等NoSQL。

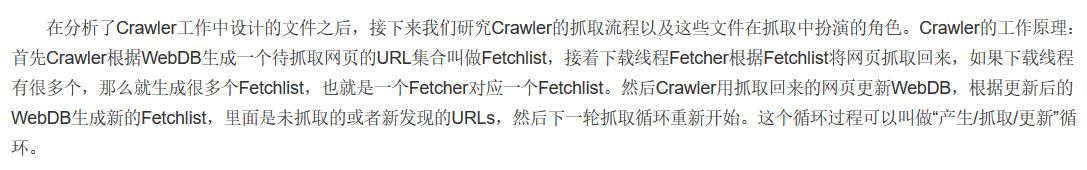
Crawler的重点在两个方面，Crawler的工作流程和涉及的数据文件的格式和含义。数据文件主要包括三类，分别是web database，一系列的segment加上index，三者的物理文件分别存储在爬行结果目录下的db目录下webdb子文件夹内，segments文件夹和index文件夹。那么三者分别存储的信息是什么呢？

一次爬行会产生很多个segment，每个segment内存储的是爬虫Crawler在单独一次抓取循环中抓到的网页以及这些网页的索引。Crawler爬行时会根据WebDB中的link关系按照一定的爬行策略生成每次抓取循环所需的fetchlist，然后Fetcher通过fetchlist中的URLs抓取这些网页并索引，然后将其存入segment。Segment是有时限的，当这些网页被Crawler重新抓取后，先前抓取产生的segment就作废了。在存储中。Segment文件夹是以产生时间命名的，方便我们删除作废的segments以节省存储空间。

Index是Crawler抓取的所有网页的索引，它是通过对所有单个segment中的索引进行合并处理所得的。Nutch利用Lucene技术进行索引，所以Lucene中对索引进行操作的接口对Nutch中的index同样有效。但是需要注意的是，Lucene中的segment和Nutch中的不同，Lucene中的segment是索引index的一部分，但是Nutch中的segment只是WebDB中各个部分网页的内容和索引，最后通过其生成的index跟这些segment已经毫无关系了。

Web database，也叫WebDB，其中存储的是爬虫所抓取网页之间的链接结构信息，它只在爬虫Crawler工作中使用而和Searcher的工作没有任何关系。WebDB内存储了两种实体的信息：page和link。Page实体通过描述网络上一个网页的特征信息来表征一个实际的网页，因为网页有很多个需要描述，WebDB中通过网页的URL和网页内容的MD5两种索引方法对这些网页实体进行了索引。Page实体描述的网页特征主要包括网页内的link数目，抓取此网页的时间等相关抓取信息，对此网页的重要度评分等。同样的，Link实体描述的是两个page实体之间的链接关系。WebDB构成了一个所抓取网页的链接结构图，这个图中Page实体是图的结点，而Link实体则代表图的边。





1. 创建一个新的WebDb (admin db -create).

2. 将抓取起始URLs写入WebDB中 (inject).

3. 根据WebDB生成fetchlist并写入相应的segment(generate).

4. 根据fetchlist中的URL抓取网页 (fetch).

5. 根据抓取网页更新WebDb (updatedb).

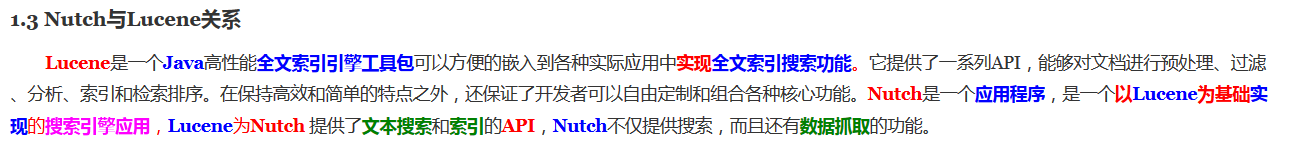
6. 循环进行3－5步直至预先设定的抓取深度。

7. 根据WebDB得到的网页评分和links更新segments (updatesegs).

8. 对所抓取的网页进行索引(index).

9. 在索引中丢弃有重复内容的网页和重复的URLs (dedup).

10. 将segments中的索引进行合并生成用于检索的最终index(merge).



Nutch 搜索引擎是一个基于 Java 的开放源代码的搜索引擎。Nutch 搜索引擎处理流程包括抓取流程和搜索流程，如图 1 所示。相应地 Nutch 也分为2部分，抓取器和搜索器。 在抓取流程中，抓取器也叫蜘蛛或者机器人，以广度优先搜索（BFS）的方式从企业内部网或者互联网抓取网页。这个过程涉及到对 CrawlDB 和 LinkDB 数据库的操作。然后 Nutch 解析器开始解析诸如 HTML、XML、RSS、PDF等不同格式的文档。最后 Nutch 索引器针对解析结果建立索引并存储到 indexDB 和 SegmentsDB 数据库中，以供搜索器搜索使用。

在搜索流程中，搜索应用使用输入关键词调用 Nutch 搜索接口（Nutch Query Interface）。应用可通过网页上的输入框输入相应关键词。搜索接口解析搜索请求为 Lucene 全文检索引擎可以识别的格式。Nutch 索引器将会调用 Lucene 引擎来响应请求在 indexDB 上展开搜索。最后搜索接口收集从索引器返回的URL、标题、锚和从 SegmentsDB 返回的内容。所有上述内容将被提供给排序算法进行排序。排序完成后，搜索接口将返回命中的搜索结果。 由于构建在 Hadoop 分布式文件系统之上， Nutch 对CrawlDB, LinkDB, SegmentsDB 和 IndexDB 数据库的操作都是通过调用 M/R(map/reduce) 函数完成的。这使得 Nutch 具有了集群扩展能力。

