搜索引擎Nutch源码解析

# 简介

Nutch主要分为两个部分：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。

Crawler和Searcher两部分尽量分开的目的主要是为了使两部分可以分布式配置在硬件平台上，例如将Crawler和Searcher分别放在两个主机上，这样可以提升性能。

爬虫，Crawler：

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？

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。

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。

Crawler工作流程：

在分析了Crawler工作中设计的文件之后，接下来我们研究一下Crawler的抓取流程以及这些文件在抓取中扮演的角色。Crawler的工作原理主要是：首先Crawler根据WebDB生成一个待抓取网页的URL集合叫做Fetchlist，接着下载线程Fetcher开始根据Fetchlist将网页抓取回来，如果下载线程有很多个，那么就生成很多个Fetchlist，也就是一个Fetcher对应一个Fetchlist。然后Crawler根据抓取回来的网页WebDB进行更新，根据更新后的WebDB生成新的Fetchlist，里面是未抓取的或者新发现的URLs，然后下一轮抓取循环重新开始。这个循环过程可以叫做“产生/抓取/更新”循环。

指向同一个主机上Web资源的URLs通常被分配到同一个Fetchlist中，这样的话防止过多的Fetchers对一个主机同时进行抓取造成主机负担过重。另外Nutch遵守[Robots Exclusion Protocol](http://www.robotstxt.org/wc/exclusion.html)，网站可以通过自定义Robots.txt控制Crawler的抓取。

在Nutch中，Crawler操作的实现是通过一系列子操作的实现来完成的。这些子操作Nutch都提供了子命令行可以单独进行调用。下面就是这些子操作的功能描述以及命令行，命令行在括号中。

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).

Crawler详细工作流程是：在创建一个WebDB之后(步骤1), “产生/抓取/更新”循环(步骤3－6)根据一些种子URLs开始启动。当这个循环彻底结束，Crawler根据抓取中生成的segments创建索引（步骤7－10）。在进行重复URLs清除（步骤9）之前，每个segment的索引都是独立的（步骤8）。最终，各个独立的segment索引被合并为一个最终的索引index（步骤10）。

其中有一个细节问题，Dedup操作主要用于清除segment索引中的重复URLs，但是我们知道，在WebDB中是不允许重复的URL存在的，那么为什么这里还要进行清除呢？原因在于抓取的更新。比方说一个月之前你抓取过这些网页，一个月后为了更新进行了重新抓取，那么旧的segment在没有删除之前仍然起作用，这个时候就需要在新旧segment之间进行除重。

另外这些子操作在第一次进行Crawler运行时可能并不用到，它们主要用于接下来的索引更新，增量搜索等操作的实现。这些在以后的文章中我再详细讲。

个性化设置：

Nutch中的所有配置文件都放置在总目录下的conf子文件夹中，最基本的配置文件是[conf/nutch-default.xml](http://svn.apache.org/viewcvs.cgi/*checkout*/lucene/nutch/trunk/conf/nutch-default.xml?rev=179640)。这个文件中定义了Nutch的所有必要设置以及一些默认值，它是不可以被修改的。如果你想进行个性化设置，你需要在conf/nutch-site.xml进行设置，它会对默认设置进行屏蔽。

Nutch考虑了其可扩展性，你可以自定义插件plugins来定制自己的服务，一些plugins存放于plugins子文件夹。Nutch的网页解析与索引功能是通过插件形式进行实现的，例如，对HTML文件的解析与索引是通过[HTML document parsing plugin](http://lucene.apache.org/nutch/apidocs/org/apache/nutch/parse/html/package-summary.html), parse-html实现的。所以你完全可以自定义各种解析插件然后对配置文件进行修改，然后你就可以抓取并索引各种类型的文件了。