BLOG **Welcome to our BLOG.**

Get the latest scoop or just follow and discuss our latest studies.

Mar
13

Apache Nutch Overview

By Emir DIZDAREVIC // Lucene Search // No Comments

Apache Nutch is a crawler written in Java. It is continuously updated and its pluggable architecture makes it quite adaptable, which is the reason why we chose Apache Nutch for data extraction. The documentation is not really up to date but the source is good enough to figure out how to customize its behavior.

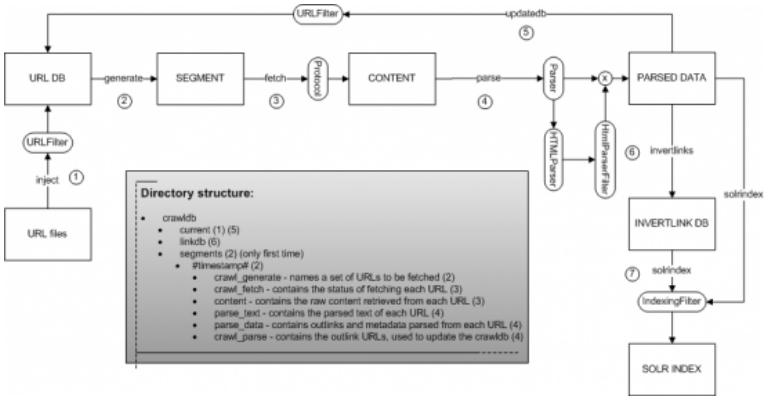
The whole processing of data extraction in Apache Nutch is done trough plugins. Plugins are a set of extension which corresponds to the extension points defined by Nutch. The supported extension points by are: OnlineCluster, IndexingFilter, Ontology, Parser, HtmlParserFilter, Protocol, QueryFilter, URLFilter and NutchAnalyzer. In most cases it will be enough to extend on of following plugins:

- HtmlParserFilter – Permits one to add additional metadata to HTML parses
- IndexingFilter – Permits one to add metadata to the indexed fields. All plugins found which implement this extension point are run sequentially on the parse

More information about the Nutch plugin architecture can be found at http://wiki.apache.org/nutch/AboutPlugins.

This article is not indented to teach someone how to use Apache Nutch. Whoever is unfamiliar with Apache Nutch usage and configuration we may refer you to the official documentation: http://wiki.apache.org/nutch/. It will give different perspective on Nutch's flow, something that we found missing while initially learning Nutch.

Notice: This is written for Apache Nutch 1.4



Apache Nutch Basic Workflow:

1. **Inject** – Injects URLs into the Nutch URL database. If the database does not exist it will be created. Injected URLs are stored under <current> directory.
   - `Command: bin/nutch inject <crawldb> <url_dir>`
   - `Example: bin/nutch inject crawldb urls`
2. **Generate** – You start by generating a segment which tells Nutch which and how to fetch the urls. Nutch creates folder <#timestamp#> (timestamp when it was created) under the <segments_dir>. Inside that dir he creates the folder "crawl_generate" where Nutch puts list of URLs to be fetched.
   - `Command: bin/nutch generate <crawldb> <segments_dir> [-force] [-topN N] [-numFetchers numFetchers] [-adddays numDays] [-noFilter] [-noNorm] [-maxNumSegments num]`
   - `Example: bin/nutch generate crawldb crawldb/segments -topN 1000`
3. **Fetch** – Instruct Nutch to fetch the URLs we generated in the previous step. Nutch fetches the resources using one of the defined Protocol extensions (http, ftp, file etc.) and places the data into the "content" and "crawl_fetch" folders inside the <segments_dir>/#timestamp# directory.
   - `Command: bin/nutch fetch <segment> [-threads n]`

OUR SERVICES

AUTOMATED QUALITY ASSURANCE
BIG DATA (HADOOP)
LUCENE SEARCH
MOBILE DEVELOPMENT

BLOG CATEGORIES

Automated QA / Testing (5)
BI and Analytics (2)
Big Data (8)
Hadoop as service (3)
Lucene Search (5)
Regression and Monitoring (1)
Testing Extensions (2)
Testing Tools (4)
Uncategorized (1)

RECENT POSTS

Alternating Nutch flow to store fetched data to CSV
Precise data extraction with Apache Nutch
Apache Nutch Overview
Bad vs Good Search Experience
Regression Suite
Apache Solr – Slow queries and frequent terms
Amazon Elastic MapReduce – Part 2 (Amazon S3 Input Format)
HBase backup, anyone?
Hadoop powered BI and Analytics – Part 2
Hadoop powered BI and Analytics – Part 1

CONTACT US

Our sales and support staff are standing by

+387.33.716550

Contact Us

- **Example:** bin/nutch fetch crawldb/segments/#timestamp#

4. **Parse –** Now we instruct Nutch to parse the fetched data which he places into three folders "crawl_parse", "parse_data" and "parse_text" inside the <segments_dir>/#timestamp# directory. After the resource (page, document, image etc.) has been fetched Nutch delegates the resource in form of a byte array to the configured Parser extension. The Parser extensions are mapped trough the MIME type of the fetched resource. The output of the parser is the parsed text, metadata, and the newly acquired URLs. If you use HtmlParser as Parser extension you can add the HtmlParserFilter extension to filter/manipulate/drop the parsed data.

- **Command:** bin/nutch parse
- **Example:** bin/nutch parse crawldb/segments/#timestamp#

5. **Updatedb –** Update the url database with the newly acquired urls.

- **Command:** bin/nutch updatedb <crawldb> (-dir <segments> | <seg1> <seg2> ...) [-force] [-normalize] [-filter] [-noAdditions]
- **Example:** bin/nutch updatedb crawldb crawldb/segments/#timestamp#

6. **Invertlinks –** Create invert link database required for indexing. You can instruct Nutch to normalize and filter urls.

- **Command:** bin/nutch invertlinks <linkdb> (-dir <segmentsDir> | <seg1> <seg2> …) [-force] [-noNormalize] [-noFilter]
- **Example:** bin/nutch invertlinks crawldb/linkdb -dir crawldb/segments

7. **Solrindex -** Index the crawled data with Apache Solr. Nutch delegates all data collected in parsing to the IndexingFilter extension which generates the data to be indexed. The output of the filter is a NutchDocument which again is delegated to Nutch. Nutch then decides if the data should be indexed based on the mapping file which defines which NutchDocument fields will be mapped to SolrDocument fields is read by Nutch.

- **Command:** bin/nutch solrindex <solr url> <crawldb> [-linkdb <linkdb>] (... | -dir <segments>) [noCommit]
- **Example:** bin/nutch solrindex http://127.0.0.1:8983/solr/ crawldb -linkdb crawldb/linkdb crawldb/segments/*

Notice: "bin/nutch" is relative to the NUTCH_HOME directory.

After injecting starting points, steps 2-5 are crawling loop and, if topN parameter in generate step is large enough to fatch all links, each repeat is going one depth more. Following script automates the process of generating/fetching/parsing/updatingdb:

Command: gfpu topN repeatTimes
Example:gfpu 1000 5

```
#!/bin/bash

# Get arguments
TOPN="$1"
REPEATTIMES="$2"

if [ $# = 0 ]; then
    echo "Usage: gfpu topN repeatTimes"
    exit 1
fi

for (( i=1; i&lt;=$REPEATTIMES; i++ ))
do

    echo "ITERATION: $i"
    echo "GENERATING"
    bin/nutch generate crawldb crawldb/segments -topN $TOPN
    seg=`ls -d crawldb/segments/* | tail -1`

    echo "FETCHING"
    bin/nutch fetch $seg

    echo "PARSING"
    bin/nutch parse $seg

    echo "UPDATING DB"
    bin/nutch updatedb crawldb $seg

    # FIX FOR DIFFERENT FOLDER STRUCTURE FOR inject AND updatedb
    cur=`ls -d crawldb/current/* | tail -1`
    if [ "$cur" != "crawldb/current/part-00000" ]
    then
        shopt -s dotglob
        mv $cur/part-00000/* $cur
        shopt -u dotglob
        rm -f -r $cur/part-00000
    fi
done
```

These and additional scripts can be downloaded from this repository: https://github.com/ATLANTBH/nutch-plugins

Steps 6. and 7. are indexing steps. Creating inverted links DB is optional and needed if incoming links are required. Nutch has built in support only for SOLR.

When it comes to general web crawling and indexing, Apache Nutch with it's out of the box plugins makes a great job. However, we run into few limitations:

- It parses web page and index entire content as single field. There is no built in support to extract data from a web page.
- There is no separation between pages that should be visited from one that should be indexed.
- There is no nice extension point for non SOLR indexing.

To address this problem we wrote a couple of plugins that will be covered in the next article:

Come and visit us at
Zmaja od Bosne 74
71000 Sarajevo
Bosnia and Herzegovina

Keep in touch
→ Contact Us

http://www.atlantbh.com/precise-data-extraction-with-apache-nutch/.

Career:
phone: +387.33.716557
email: jobs@atlantbh.com

Contact:
phone: +387.33.716550
fax: +387.33.716551
email: contact@atlantbh.com