NutchTutorial

Introduction

Nutch is a well matured, production ready Web crawler. Nutch 1.x enables fine grained configuration, relying on Apache Hadoop data structures, which are great for batch processing. Being pluggable and modular of course has it's benefits, Nutch provides extensible interfaces such as Parse, Index and Scoring Filter's for custom implementations e.g. Apache Tika for parsing. Additonally, pluggable indexing exists for Apache Solr, Elastic Search, SolrCloud, etc. We can find Web page hyperlinks in an automated manner, reduce lots of maintenance work, for example checking broken links, and create a copy of all the visited pages for searching over. This tutorial explains how to use Nutch with Apache Solr. Solr is an open source full text search framework, with Solr we can search pages acquired by Nutch. Apache Nutch supports Solr out-the-box, simplifying Nutch-Solr integration. It also removes the legacy dependence upon both Apache Tomcat for running the old Nutch Web Application and upon Apache Lucene for indexing. Just download a binary release from here.

Learning Outcomes

By the end of this tutorial you will

- Have a configured local Nutch crawler setup to crawl on one machine
- Learned how to understand and configure Nutch runtime configuration including seed URL lists, URLFilters, etc.
- Have executed a Nutch crawl cycle and viewed the results of the Crawl Database
- Indexed Nutch crawl records into Apache Solr for full text search

Any issues with this tutorial should be reported to the Nutch user@ list.

Table of Contents

Steps

> This tutorial describes the installation and use of Nutch 1.x (e.g. release cut from the master branch). For a similar Nutch 2.x with HBase tutorial, see Nutch2Tutorial.

Requirements

- Unix environment, or Windows-Cygwin environment
- Java Runtime/Development Environment (JDK 11 / Java 11)
- (Source build only) Apache Ant: https://ant.apache.org/

Install Nutch

Option 1: Setup Nutch from a binary distribution

- Download a binary package (`apache-nutch-1.X-bin.zip`) from here.
- Unzip your binary Nutch package. There should be a folder `apache-nutch-1.X`.
- `cd apache-nutch-1.X/`
  From now on, we are going to use `${NUTCH_RUNTIME_HOME}` to refer to the current directory (`apache-nutch-1.X/`).

Option 2: Set up Nutch from a source distribution

Advanced users may also use the source distribution:

- Download a source package (`apache-nutch-1.X-src.zip`)
- Unzip
- `cd apache-nutch-1.X/`
- Run `ant` in this folder (cf. RunNutchInEclipse)
- Now there is a directory `runtime/local` which contains a ready to use Nutch installation.
  When the source distribution is used `${NUTCH_RUNTIME_HOME}` refers to `apache-nutch-1.X/runtime/local/`. Note that
- config files should be modified in `apache-nutch-1.X/runtime/local/conf/`
- `ant clean` will remove this directory (keep copies of modified config files)

Option 3: Set up Nutch from source

See UsingGit#CheckingoutacopyofNutchandmodifyingit

Verify your Nutch installation

- run "`bin/nutch`" - You can confirm a correct installation if you see something similar to the following:

```
Usage: nutch COMMAND where command is one of:
readdb            read / dump crawl db
mergedb           merge crawldb-s, with optional filtering
readlinkdb        read / dump link db
inject            inject new urls into the database
generate          generate new segments to fetch from crawl db
freegen           generate new segments to fetch from text files
fetch             fetch a segment's pages
...
```

Some troubleshooting tips:

- Run the following command if you are seeing "Permission denied":

```
chmod +x bin/nutch
```

- Setup `JAVA_HOME` if you are seeing `JAVA_HOME` not set. On Mac, you can run the following command or add it to `~/.bashrc`:

```
export JAVA_HOME=/System/Library/Frameworks/JavaVM.framework/Versions/11/Home
# note that the actual path may be different on your system
```

On Debian or Ubuntu, you can run the following command or add it to ~/.bashrc:

```
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
```

You may also have to update your /etc/hosts file. If so you can add the following

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost.localdomain localhost LMC-032857
::1             ip6-localhost ip6-loopback
fe80::1%lo0     ip6-localhost ip6-loopback
```

Note that the `LMC-032857` above should be replaced with your machine name.

Crawl your first website

Nutch requires two configuration changes before a website can be crawled:

1. Customize your crawl properties, where at a minimum, you provide a name for your crawler for external servers to recognize
2. Set a seed list of URLs to crawl

Customize your crawl properties

- Default crawl properties can be viewed and edited within {{conf/nutch-default.xml }}- where most of these can be used without modification
- The file `conf/nutch-site.xml` serves as a place to add your own custom crawl properties that overwrite `conf/nutch-default.xml`. The only required modification for this file is to override the `value` field of the {{http.agent.name }}
    - i.e. Add your agent name in the `value` field of the `http.agent.name` property in `conf/nutch-site.xml`, for example:

```
<property>
 <name>http.agent.name</name>
 <value>My Nutch Spider</value>
</property>
```

- ensure that the `plugin.includes` property within `conf/nutch-site.xml` includes the indexer as `indexer-solr`

Create a URL seed list

- A URL seed list includes a list of websites, one-per-line, which nutch will look to crawl
- The file `conf/regex-urlfilter.txt` will provide Regular Expressions that allow nutch to filter and narrow the types of web resources to crawl and download

Create a URL seed list

- `mkdir -p urls`
- `cd urls`
- `touch seed.txt` to create a text file `seed.txt` under `urls/` with the following content (one URL per line for each site you want Nutch to crawl).

```
http://nutch.apache.org/
```

(Optional) Configure Regular Expression Filters

Edit the file `conf/regex-urlfilter.txt` and replace

```
# accept anything else
+.
```

with a regular expression matching the domain you wish to crawl. For example, if you wished to limit the crawl to the `nutch.apache.org` domain, the line should read:

```
+^https?://([a-z0-9-]+\.)*nutch\.apache\.org/
```

This will include any URL in the domain `nutch.apache.org`.

NOTE: Not specifying any domains to include within regex-urlfilter.txt will lead to all domains linking to your seed URLs file being crawled as well.

Using Individual Commands for Whole-Web Crawling

NOTE: If you previously modified the file `conf/regex-urlfilter.txt` as covered here you will need to change it back.

Whole-Web crawling is designed to handle very large crawls which may take weeks to complete, running on multiple machines. This also permits more control over the crawl process, and incremental crawling. It is important to note that whole Web crawling does not necessarily mean crawling the entire World Wide Web. We can limit a whole Web crawl to just a list of the URLs we want to crawl. This is done by using a filter just like the one we used when we did the `crawl` command (above).

Step-by-Step: Concepts

Nutch data is composed of:

1. The crawl database, or crawldb. This contains information about every URL known to Nutch, including whether it was fetched, and, if so, when.
2. The link database, or linkdb. This contains the list of known links to each URL, including both the source URL and anchor text of the link.
3. A set of segments. Each segment is a set of URLs that are fetched as a unit. Segments are directories with the following subdirectories:
    - a *crawl_generate* names a set of URLs to be fetched
    - a *crawl_fetch* contains the status of fetching each URL
    - a *content* contains the raw content retrieved from each URL
    - a *parse_text* contains the parsed text of each URL
    - a *parse_data* contains outlinks and metadata parsed from each URL
    - a *crawl_parse* contains the outlink URLs, used to update the crawldb

Step-by-Step: Seeding the crawldb with a list of URLs

Bootstrapping from an initial seed list.

This option shadows the creation of the seed list as covered here.

```
bin/nutch inject crawl/crawldb urls
```

Now we have a Web database with your unfetched URLs in it.

Step-by-Step: Fetching

To fetch, we first generate a fetch list from the database:

```
bin/nutch generate crawl/crawldb crawl/segments
```

This generates a fetch list for all of the pages due to be fetched. The fetch list is placed in a newly created segment directory. The segment directory is named by the time it's created. We save the name of this segment in the shell variable `s1`:

```
s1=`ls -d crawl/segments/2* | tail -1`
echo $s1
```

Now we run the fetcher on this segment with:

```
bin/nutch fetch $s1
```

Then we parse the entries:

```
bin/nutch parse $s1
```

When this is complete, we update the database with the results of the fetch:

```
bin/nutch updatedb crawl/crawldb $s1
```

Now the database contains both updated entries for all initial pages as well as new entries that correspond to newly discovered pages linked from the initial set.

Now we generate and fetch a new segment containing the top-scoring 1,000 pages:

```
bin/nutch generate crawl/crawldb crawl/segments -topN 1000
s2=`ls -d crawl/segments/2* | tail -1`
echo $s2

bin/nutch fetch $s2
bin/nutch parse $s2
bin/nutch updatedb crawl/crawldb $s2
```

Let's fetch one more round:

```
bin/nutch generate crawl/crawldb crawl/segments -topN 1000
s3=`ls -d crawl/segments/2* | tail -1`
echo $s3

bin/nutch fetch $s3
bin/nutch parse $s3
bin/nutch updatedb crawl/crawldb $s3
```

By this point we've fetched a few thousand pages. Let's invert links and index them!

Step-by-Step: Invertlinks

Before indexing we first invert all of the links, so that we may index incoming anchor text with the pages.

```
bin/nutch invertlinks crawl/linkdb -dir crawl/segments
```

We are now ready to search with Apache Solr.

Step-by-Step: Indexing into Apache Solr

Note: For this step you should have Solr installation. If you didn't integrate Nutch with Solr. You should read here.

Now we are ready to go on and index all the resources. For more information see the command line options.

```
Usage: Indexer (<crawldb> | -nocrawldb) (<segment> ... | -dir <segments>) [general options]

Index given segments using configured indexer plugins

The CrawlDb is optional but it is required to send deletion requests for duplicates
and to read the proper document score/boost/weight passed to the indexers.

Required arguments:

        <crawldb>       path to CrawlDb, or
        -nocrawldb      flag to indicate that no CrawlDb shall be used

        <segment> ...   path(s) to segment, or
        -dir <segments> path to segments/ directory,
                        (all subdirectories are read as segments)

General options:

        -linkdb <linkdb>        use LinkDb to index anchor texts of incoming links
        -params k1=v1&k2=v2...  parameters passed to indexer plugins
                                (via property indexer.additional.params)

        -noCommit       do not call the commit method of indexer plugins
        -deleteGone     send deletion requests for 404s, redirects, duplicates
        -filter         skip documents with URL rejected by configured URL filters
        -normalize      normalize URLs before indexing
        -addBinaryContent       index raw/binary content in field `binaryContent`
        -base64         use Base64 encoding for binary content

Example:
   bin/nutch index crawl/crawldb/ -linkdb crawl/linkdb/ crawl/segments/20131108063838/ -filter -normalize -
deleteGone
```

Step-by-Step: Deleting Duplicates

Duplicates (identical content but different URL) are optionally marked in the CrawlDb and are deleted later in the Solr index.

MapReduce "dedup" job:

- Map: Identity map where keys are digests and values are CrawlDatum records
- Reduce: CrawlDatums with the same digest are marked (except one of them) as duplicates. There are multiple heuristics available to choose the item which is not marked as duplicate - the one with the shortest URL, fetched most recently, or with the highest score.

```
Usage: bin/nutch dedup <crawldb> [-group <none|host|domain>] [-compareOrder <score>,<fetchTime>,<httpsOverHttp>,
<urlLength>]
```

Deletion in the index is performed by the cleaning job (see below) or if the index job is called with the command-line flag `-deleteGone`.

For more information see dedup documentation.

Step-by-Step: Cleaning Solr

The class scans a crawldb directory looking for entries with status DB_GONE (404), duplicates or optionally redirects and sends delete requests to Solr for those documents. Once Solr receives the request the aforementioned documents are duly deleted. This maintains a healthier quality of Solr index.

```
Usage: bin/nutch clean <crawldb> [-noCommit]
Example: bin/nutch clean crawl/crawldb/
```

For more information see clean documentation.

Using the crawl script

If you have followed the section above on how the crawling can be done step by step, you might be wondering how a bash script can be written to automate all the process described above.

Nutch developers have written one for you 🙂, and it is available at bin/crawl. Here the most common options and parameters:

```
Usage: crawl [options] <crawl_dir> <num_rounds>

Arguments:
  <crawl_dir>                           Directory where the crawl/host/link/segments dirs are saved
  <num_rounds>                          The number of rounds to run this crawl for

Options:
  -i|--index                            Indexes crawl results into a configured indexer
  -D                                    A Nutch or Hadoop property to pass to Nutch calls overwriting
                                        properties defined in configuration files, e.g.
                                        increase content limit to 2MB:
                                          -D http.content.limit=2097152
                                        (distributed mode only) configure memory of map and reduce tasks:
                                          -D mapreduce.map.memory.mb=4608    -D mapreduce.map.java.opts=-
Xmx4096m
                                          -D mapreduce.reduce.memory.mb=4608 -D mapreduce.reduce.java.opts=-
Xmx4096m
  -w|--wait <NUMBER[SUFFIX]>            Time to wait before generating a new segment when no URLs
                                        are scheduled for fetching. Suffix can be: s for second,
                                        m for minute, h for hour and d for day. If no suffix is
                                        specified second is used by default. [default: -1]
  -s <seed_dir>                         Path to seeds file(s)
  -sm <sitemap_dir>                     Path to sitemap URL file(s)
  --hostdbupdate                        Boolean flag showing if we either update or not update hostdb for each
round
  --hostdbgenerate                      Boolean flag showing if we use hostdb in generate or not
  --num-fetchers <num_fetchers>         Number of tasks used for fetching (fetcher map tasks) [default: 1]
                                        Note: This can only be set when running in distributed mode and
                                              should correspond to the number of worker nodes in the cluster.
  --num-tasks <num_tasks>               Number of reducer tasks [default: 2]
  --size-fetchlist <size_fetchlist>     Number of URLs to fetch in one iteration [default: 50000]
  --time-limit-fetch <time_limit_fetch> Number of minutes allocated to the fetching [default: 180]
  --num-threads <num_threads>           Number of threads for fetching / sitemap processing [default: 50]
  --sitemaps-from-hostdb <frequency>    Whether and how often to process sitemaps based on HostDB.
                                        Supported values are:
                                          - never [default]
                                          - always (processing takes place in every iteration)
                                          - once (processing only takes place in the first iteration)
```

The crawl script has lot of parameters set, and you can modify the parameters to your needs. It would be ideal to understand the parameters before setting up big crawls.

Setup Solr for search

Every version of Nutch is built against a specific Solr version, but you may also try a "close" version.

Nutch	Solr
1.19	8.11.2
1.18	8.5.1
1.17	8.5.1
1.16	7.3.1
1.15	7.3.1
1.14	6.6.0
1.13	5.5.0
1.12	5.4.1

To install Solr 8.x (or upwards):

- download binary file from here
- unzip to `$HOME/apache-solr`, we will now refer to this as `${APACHE_SOLR_HOME}`
- create resources for a new "nutch" Solr core

```
mkdir -p ${APACHE_SOLR_HOME}/server/solr/configsets/nutch/
cp -r ${APACHE_SOLR_HOME}/server/solr/configsets/_default/* ${APACHE_SOLR_HOME}/server/solr/configsets
/nutch/
```

- copy the Nutch's schema.xml into the Solr `conf` directory
    - (Nutch 1.15 or prior) copy the schema.xml from the conf/ directory:

      ```
      cp ${NUTCH_RUNTIME_HOME}/conf/schema.xml ${APACHE_SOLR_HOME}/server/solr/configsets/nutch/conf/
      ```

    - (Nutch 1.16 and upwards) copy the schema.xml from the indexer-solr source folder (source package):

      ```
      cp .../src/plugin/indexer-solr/schema.xml ${APACHE_SOLR_HOME}/server/solr/configsets/nutch/conf/
      ```

      or indexer-solr plugins folder (binary package):

      ```
      cp .../plugins/indexer-solr/schema.xml ${APACHE_SOLR_HOME}/server/solr/configsets/nutch/conf/
      ```

      Note for Nutch 1.16: due to NUTCH-2745 the schema.xml is not contained in the 1.16 binary package. Please download the schema.xml from the source repository.
    - You may also try to use the most recent schema.xml in case of issues launching Solr with this schema.
- make sure that there is no managed-schema "in the way":

  ```
  rm ${APACHE_SOLR_HOME}/server/solr/configsets/nutch/conf/managed-schema
  ```

- start the solr server

  ```
  ${APACHE_SOLR_HOME}/bin/solr start
  ```

- create the nutch core

  ```
  ${APACHE_SOLR_HOME}/bin/solr create -c nutch -d ${APACHE_SOLR_HOME}/server/solr/configsets/nutch/conf/
  ```

After that you need to point Nutch to the Solr instance:

- (Nutch 1.15 and later) edit the file `conf/index-writers.xml`, see IndexWriters
- (until Nutch 1.14) add the core name to the Solr server URL: `-Dsolr.server.url=http://localhost:8983/solr/nutch`

Verify Solr installation

After you started Solr admin console, you should be able to access the following links:

```
http://localhost:8983/solr/#/
```

You should be able to navigate to the `nutch` core and view the managed-schema, etc.

Whats Next

You may want to check out the documentation for the Nutch 1.X REST API to get an overview of the work going on towards providing Apache CXF based REST services for Nutch 1.X branch.