**SEARCH-ENGINE : SPORTS(CRICKET)**

**Crawling : Harsh Verma**

**TECH STACK USED:**

1. Apache Nutch-1.15

2. Solr-7.3.1d

3. Selenium-2.42.2

4. Firefox-29.0

5. Java 8.0

**Apache Nutch is an Internet search engine software, web Crawler, powerful for vertical search engine :**

For crawling web pages associated with Cricket Sports, The Apache Nutch Framework was utilized for crawling as well as feeding the Fetched Content from crawling to the Solr Framework hosted on localhost for indexing the fetched web pages as well as creating web graphs for implementing Page Rank and the HITS algorithms.

**Why Use Nutch ?**

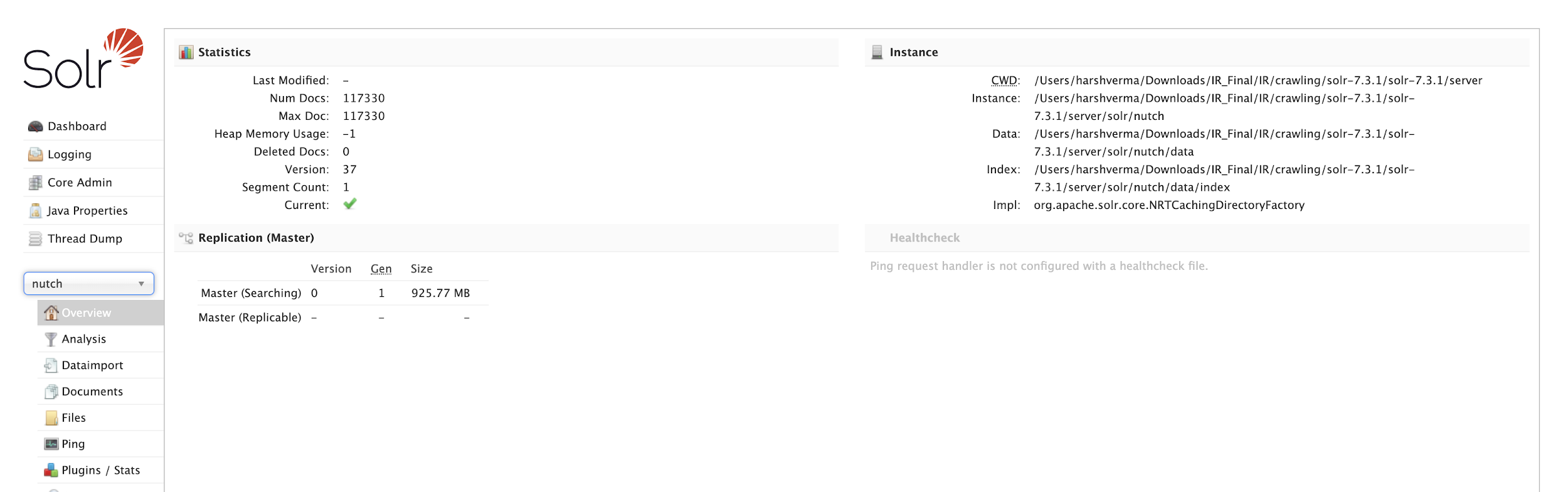
* Production ready Web Crawler, Scalable , Tried and Tested
* Fine grained Configurations
* Relying on Apache hadoop data structure, batch processing
* MultiThreaded.
* Allows Custom Implementation for parse, index and scoring.
* Pluggable Indexing (solr, mongodb, elastic search, etc)
* Automation on checking broken links
* Handle Duplication
* User friendly ,Url Filters, Normalization

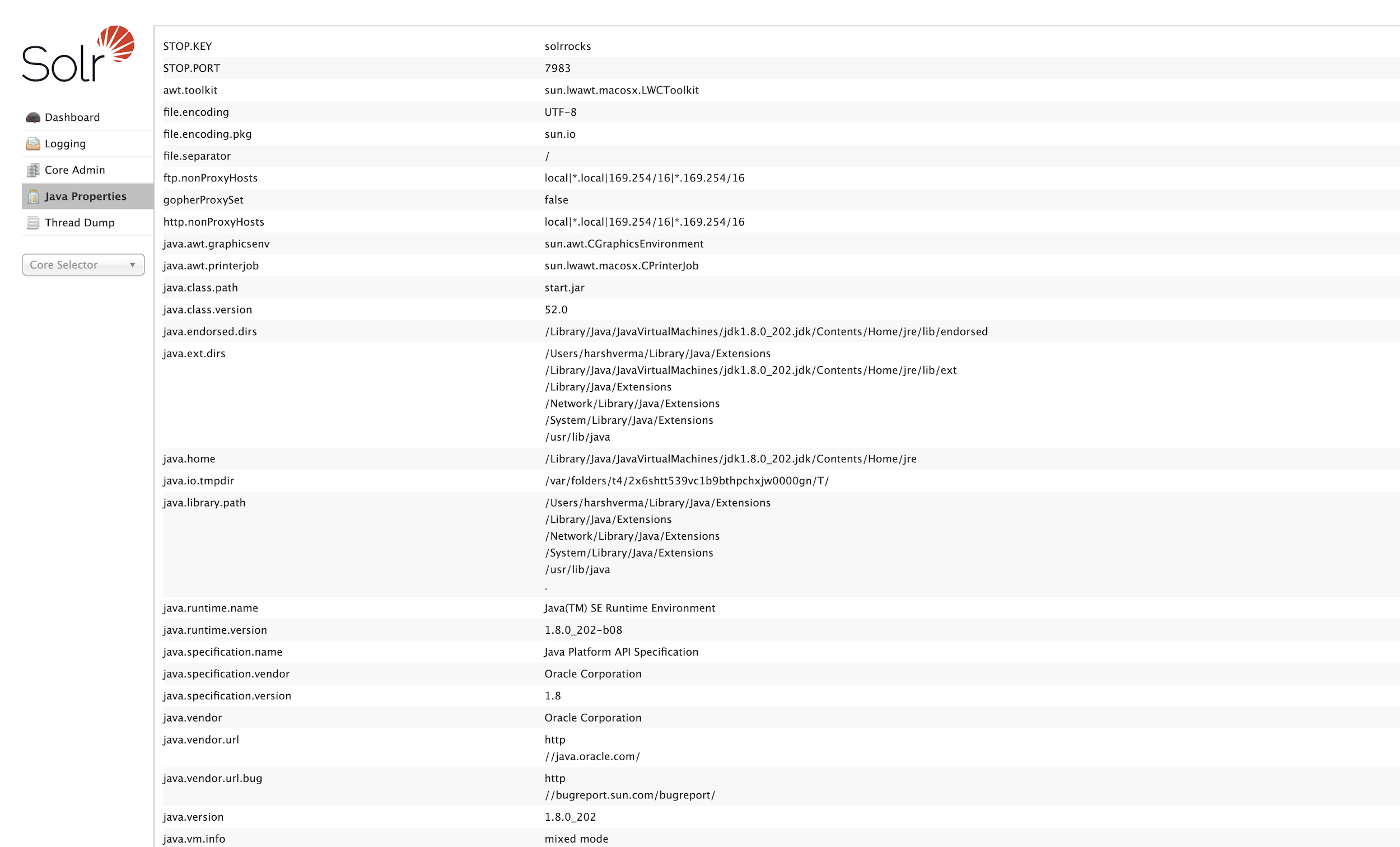
**Crawled Data Numbers:**

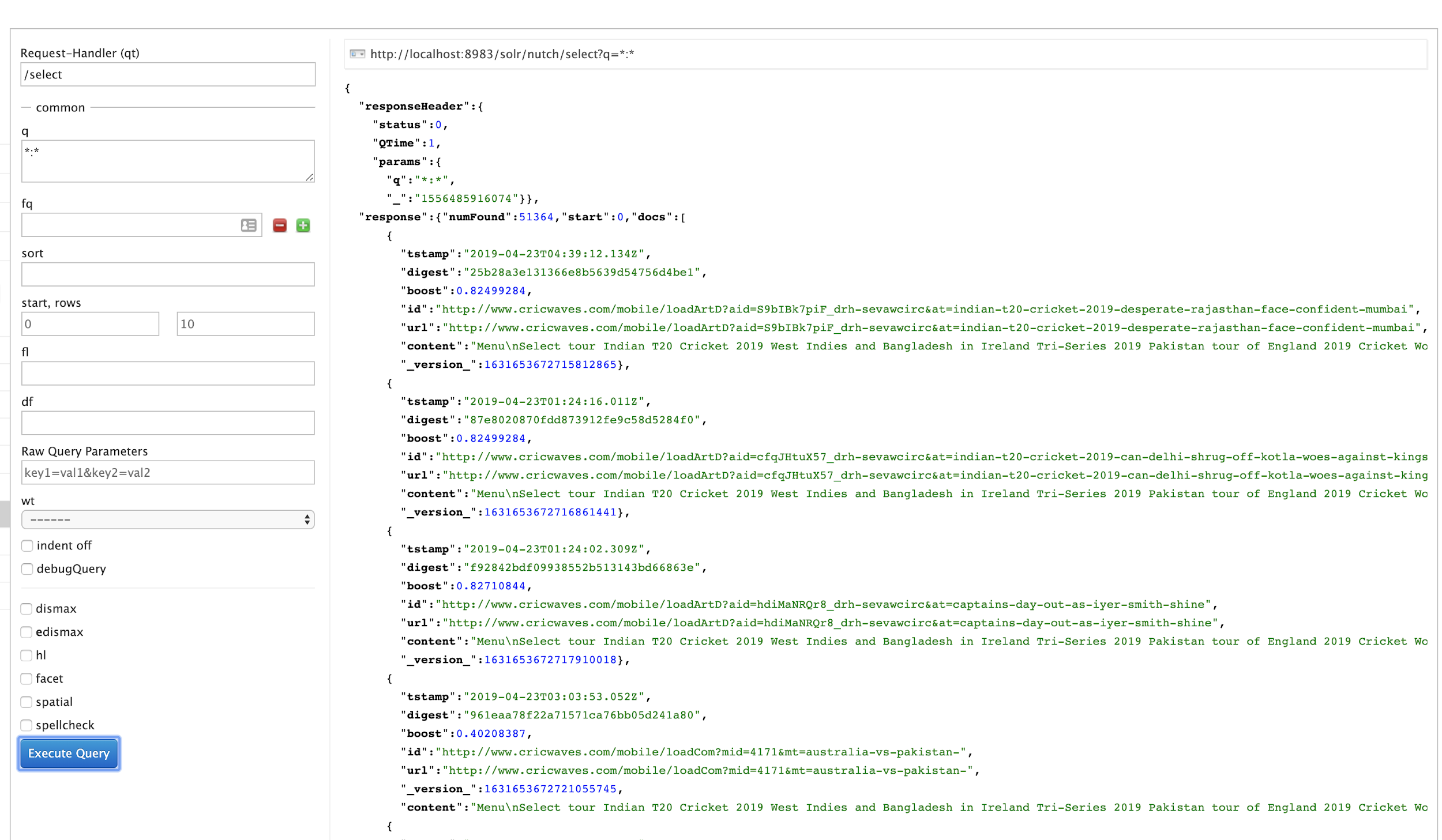
**The number of pages crawled are : 1,17,330**

**The number of web pages crawled & fetched are : 1,20,120**

**Screenshot of Documents Stats in solr :**



**Solr Settings Screenshot :**

**Searching Document in Solr:**

**Apache Nutch generates 3 folders during the crawling operation:**

1. **CRAWLDB:** it maintains the information about URLs such as the fetch status, fetching schedule, metadata, etc.

2. **LINKDB**: For each URL, the LINKDB maintains the incoming and outgoing URLs for that URL which are further used to facilitate PAGE RANKING algorithm and the HITS algorithm.

3. **SEGMENTS:** contains multiple subdirectories within it. During Crawling, the crawl script creates multiple directory to store information for Crawl Fetching, Crawl Content, Crawl Parsing, Parsed Data and Parsed Text

**The Crawling Method can be described by the following methods:**

**STEP 1: INJECTOR**

1) The injector takes all the URLs of the seeds.txt file and adds them to the crawldb.

2) As a central part of Nutch, the crawldb maintains information on all known URLs (fetch schedule, fetch status, metadata).

**$NUTCH\_RUNTIME\_HOME/bin/nutch inject crawl/crawldb urls**

**STEP 2: GENERATOR**

Based on the data of crawldb, the generator creates a fetchlist and places it in a newly created segment directory

**$NUTCH\_RUNTIME\_HOME/bin/nutch generate crawl/crawldb crawl/segments -topN 130**

**STEP 3: FECTHER**

The fetcher gets the content of the URLs on the fetchlist and writes it back to the segment directory.

This step usually is the most time-consuming one , This respects the robots rules (robort.txt and robots directives)

**$NUTCH\_RUNTIME\_HOME/bin/nutch fetch $s1**

**STEP 4: PARSER**

Now the parser processes the content of each web page and for example omits all html tags.

If the crawl functions as an update or an extension to an already existing one (e.g. depth of 3), the updater would add the new data to the crawldb as a next step.

**$NUTCH\_RUNTIME\_HOME/bin/nutch parse $s1**

**STEP 5: LINK INVERTER**

Before indexing, all the links need to be inverted, which takes into account that not the number of outgoing links of a web page is of interest, but rather the number of inbound links.

This is quite similar to how Google PageRank works and is important for the scoring function.

The inverted links are saved in the linkdb.

**$NUTCH\_RUNTIME\_HOME/bin/nutch invertlinks crawl/linkdb -dir crawl/segments**

**STEP 6 and 7: SOLR INDEXER**

Using data from all possible sources (crawldb, linkdb and segments), the indexer creates an index and saves it within the Solr directory.

For indexing, the popular Lucene library is used.

Now, the user can search for information regarding the crawled web pages via Solr.

**bin/nutch index crawl/crawldb/ -linkdb crawl/linkdb/ crawl/segments/\*/ -filter -normalize –deleteGone**

**The command used for creating the Dump is:**

**bin/nutch readlinkdb Crawling/LinkDB -dump LinkDBDUMP**

This generates a Dump containing the incoming and outgoing links for each URL.

**APACHE NUTCH CONFIGURATION**

**Modifications that were made to the nutch configuration file**

**Path:**apache-nutch-1.1.15/conf/nutch-site.xml

**Plugins**: protocol-http|protocol-httpclient|urlfilter-regex|index-(basic|more)|query- (basic|site|url|lang)|indexer-solr|nutch-extensionpoints|protocol-httpclient|urlfilter-regex|parse-(text|html|msexcel|msword|mspowerpoint|pdf)|summary-basic|scoring-opic|urlnormalizer-(pass|regex|basic)protocol-http|urlfilter-regex|parse-(html|tika|metatags)|index-(basic|anchor|more|metadata)

**Fetcher Server Delay (**The number of seconds the fetcher will delay between successive requests to the same server): 1

Selenium Driver: firefox

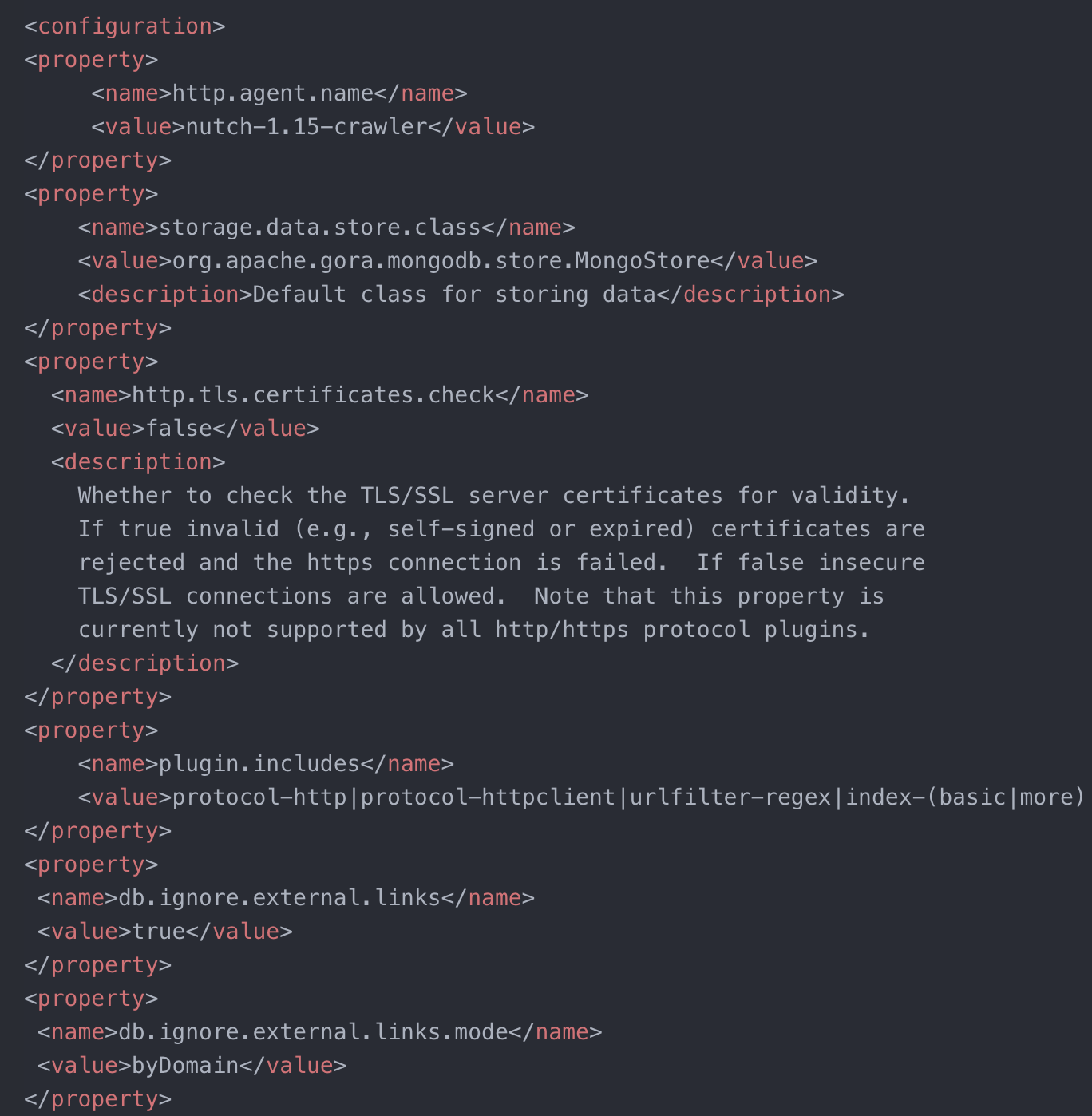
**Http Redirect Max** The maximum number of redirects the fetcher will follow when trying to fetch a page. If set to negative or 0, fetcher won't immediately follow redirected URLs, instead it will record them for later fetching: 1

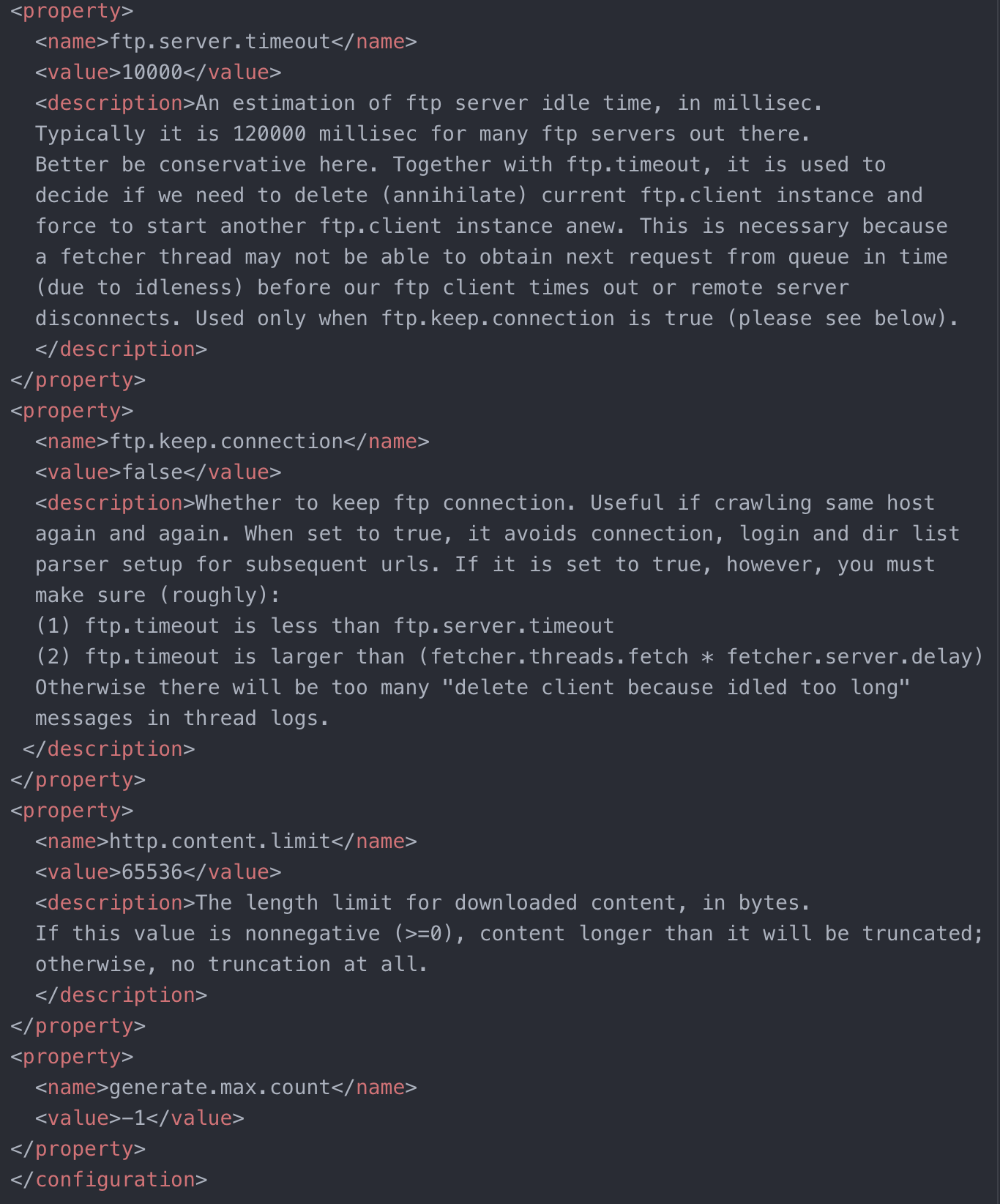
**Ignore outlinks to the same hostname**: False

Ignore outlinks to the same domain: False

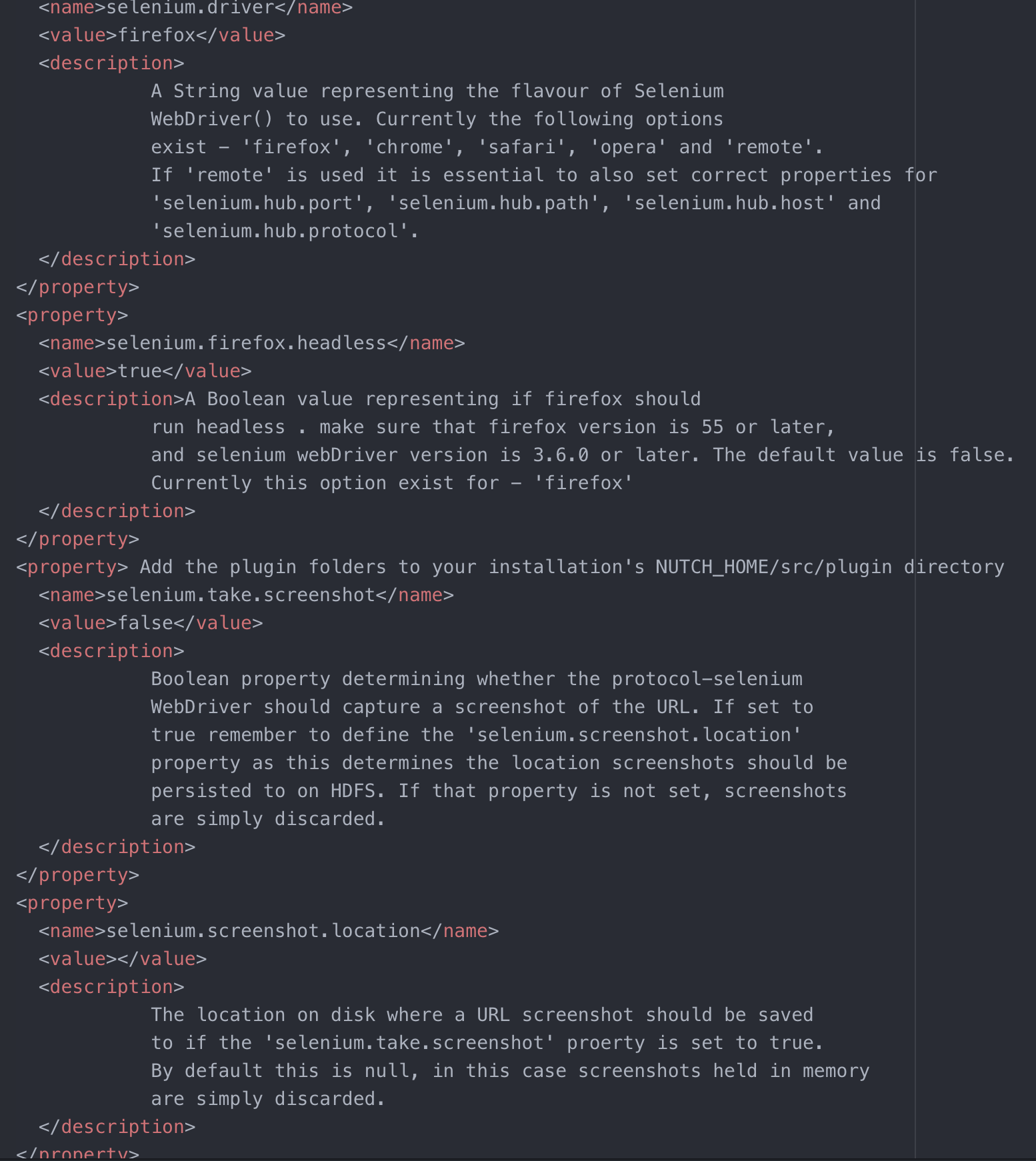
Limit to only a single outlink to the same page: False

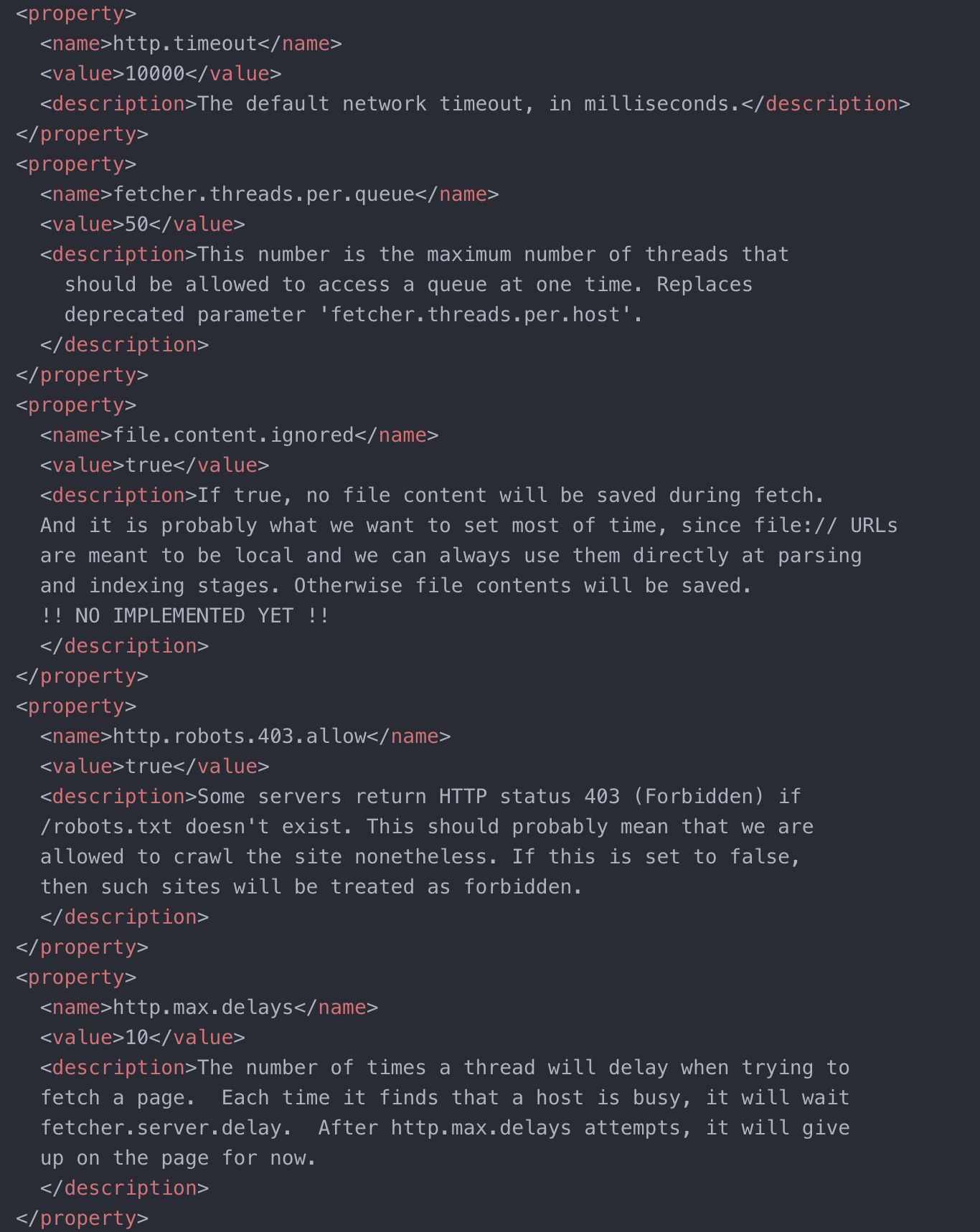
**This number is the maximum number of threads that should be allowed to access a queue at one time:** As per machine we set it to 40

**Screenshot of Nutch Config Properties: HTTP Agent and Plugin Properties**

**FTP Properties and Http Content Settings:**

**Firefox Driver and Dynamic Content Property**



**Http Timeout and Robot Property Settings:**

**STATISTICS FOR CRAWLED DATA:**

**Nutch provides an api called readdb with stats as argument, which gives the stats of the data crawled :**

|  |  |
| --- | --- |
| **TOPIC** | **NUMBER** |
| Crawled Links | 1,17,330 |
| Db\_unfetched | 2,10,120 (Depth 2) |
| Db\_gone | 4320 |

**Common Issues Faced and Resolved:**

1. **Nutch do not fetch AJAX/JavaScript driven dynamic HTML content :**

**->** In order to crawl webpages that rely on JavaScript/AJAX to dynamically load content we used the Selenium.

**-> The average rate Selenium used to crawl the pages is 500 pages / hour**

-> So to crawl more than 1,00,000 pages it took around **200+ Hours, so we used multiple machines to achieve it in 10 days parallely.**

**2. Choosing Specific Version of Apache Nutch :**

I started with apache nutch 2.X , it has web-api where in we can schedule jobs and monitor. But 2.X is not stable and does not have as many features as 1.X like webgraph api and many more. Hence, I switched to nutch 1.5

**3. Filters :** Pages like blogs dummy pages, help pages where being crawled which are not

needed, and where not crawled future with regex url filter.

**4. Selenium with Firefox compatibility issue:**

Selenium is automates browser. Nutch loads its pages in selenium’s browser. So, It’s very important to find the compatible version of browser with particular version of selenium. Version mentioned in requirements are compatible versions with nutch 1.5.

**5. Selenium has an threading issue of memory leak:** So due to this, many times browser gets stuck on a pages and do not terminate. Throwing port locked exception**.** Hence I run a command in cron for every 15-30 mins to kill firefox. This releases the locked ports, to be used by active threads.

**SEEDS LIST**

Initially we gave 95 URLs as seed URL to start our crawling

**The sample list of seed URLs are as follows:**

http://www.iplt20.com/

http://www.icc-cricket.com/

http://www.cricbuzz.com/

http://www.cricket.com.au/

http://www.royalchallengers.com/

http://www.bcci.tv/

http://www.cricketworld.com/

http://chennaisuperkings.com/

http://kkr.in/

http://www.cricketnmore.com/

http://www.mumbaiindians.com/

http://cricwaves.com/

http://www.pcb.com.pk/

http://ecb.co.uk/

http://www.kxip.in/

http://cricketweb.net/

http://rajasthanroyals.com/

http://www.srilankacricket.lk/

http://www.lords.org/

http://www.cricket365.com/

http://www.islandcricket.lk/

http://cricket.af/

http://yorkshireccc.com/

http://www.lastmanstands.com/

http://cricket.co.za/

http://www.indiancricketfans.com/

http://www.kiaoval.com/

http://cricschedule.com/

http://www.windiescricket.com/

http://www.cricketvictoria.com.au/

http://mumbaicricket.com/mca/

http://www.cricbay.com/

http://www.glamorgancricket.com/

http://emiratescricket.com/

http://cricketcrowd.com/

http://www.kentcricket.co.uk/

http://zimcricket.org/

http://pcboard.com.pk/

http://hycricket.org/

http://howstat.com.au/cricket/home.asp

http://waca.com.au/

http://cricschedule.com/

http://cricruns.com/

http://www.cricket.co.uk/

http://www.worldcricketcentre.com/

http://www.20-20.in/

http://www.indiatimes.com/sports/

http://www.in.com/sports/cricket/

http://www.hotstar.com/sports/cricket

http://sports.ndtv.com/cricket

http://crickettimes.com/series/2312/IPL-2019/

http://www.cricwaves.com/cricket/news/articles/

http://www.onlinecricketbetting.net/blog/

http://www.cricadium.com/category/ipl/2019-ipl-12/

http://www.cricadium.com/category/cricket-world-cup/world-cup-2019/

http://www.thefulltoss.com/

http://caribbeancricket.com/news

http://www.cricindeed.com/

http://www.cricketnews.net.in/

http://1tip1hand.com/

http://www.thecricketblog.com/

http://www.cricfirst.com/

http://www.cricnmore.com/

http://en.wikipedia.org/wiki/Cricket

http://en.wikipedia.org/wiki/Indian\_Premier\_League

http://cricket.yahoo.net/

http://sportzwiki.com/cricket

http://betting.betfair.com/cricket/

http://www.edailysports.com/category/cricket/

http://www.sportseon.com/cricket/

<http://www.thecricketblog.info/>

**HANDLING DUPLICATE DATA**

Apache Nutch handles duplication of the crawled data via its properties setting and commands

By default Nutch use the **org.apache.nutch.crawl.MD5Signature** class to calculate the digest of an URL, this class calculates the digest using the MD5Hash function of the raw binary content of the page, if no content is found then the URL is used.

The **DeduplicationJob** first groups fetched URLs by the digest (in your case both URLs should have the same signature/digest) and marks all the URLs as duplicated, except the one with the highest score, if both (or more) URLs have the same digest and the same score, then the one with the latest timestamp is used instead

**HYPERLINK INFORMATION FOR INDEXING AND RELEVANCE MODEL**

Crawldb of apache nutch has all the data that has been crawled. A small python script was run all the segments , and that script executed readseg api to get data from all the segments.

This api when called with parameters of to get title, url and content , created a dump for each segment.

A Python script was ran on this data to create, individual record files with Recno, Title, Url, Outlinks and Page Content which was then shared with Indexing person(Vatsal).

The content gathered by the python script from the data present in the segments has all the information such as:

* Record Number
* Title
* Urls
* Outlinks
* Page Content

**Python file Path:** apache-nutch-1.15/crawl\_parse/src/parse\_crawl.py

This output is then passed further for indexing and generating relevance model

**SOLR :**

Every version of Nutch is built against a specific Solr version , **So the compatible version with Nutch-1.15 is Solr-7.3.1 .** We used Solr to monitor the data.

To start Solr

**./bin/solr start**

**Admin Console:** <http://localhost:8983/solr/#/>

In addition, filters, normalizers and plugins allow Nutch to be highly modular, flexible and very customizable throughout the whole process.

**SELENIUM**

Nutch do not fetch AJAX/JavaScript driven dynamic HTML content

In order to crawl webpages that rely on JavaScript/AJAX to dynamically load content we used the Protocol-Selenium Plugin.

This plugin will load the pages that you're crawling in Selenium so that JavaScript will be handled properly.

So we used selenium -2.42.2 with Nutch which is the compatible version

With selenium -2.42.2 we used Firefox-29.

**DISCUSSION**

We started implementing the crawler with Scrapy and Beautiful soup. But, the issue that we faced was we were not able to get dynamic content from the websites as most of the websites uses JavaScript / Ajax to load the content. We decided to use open source crawler Apache-Nutch which was able to get dynamic content using a selenium plugin and firefox.

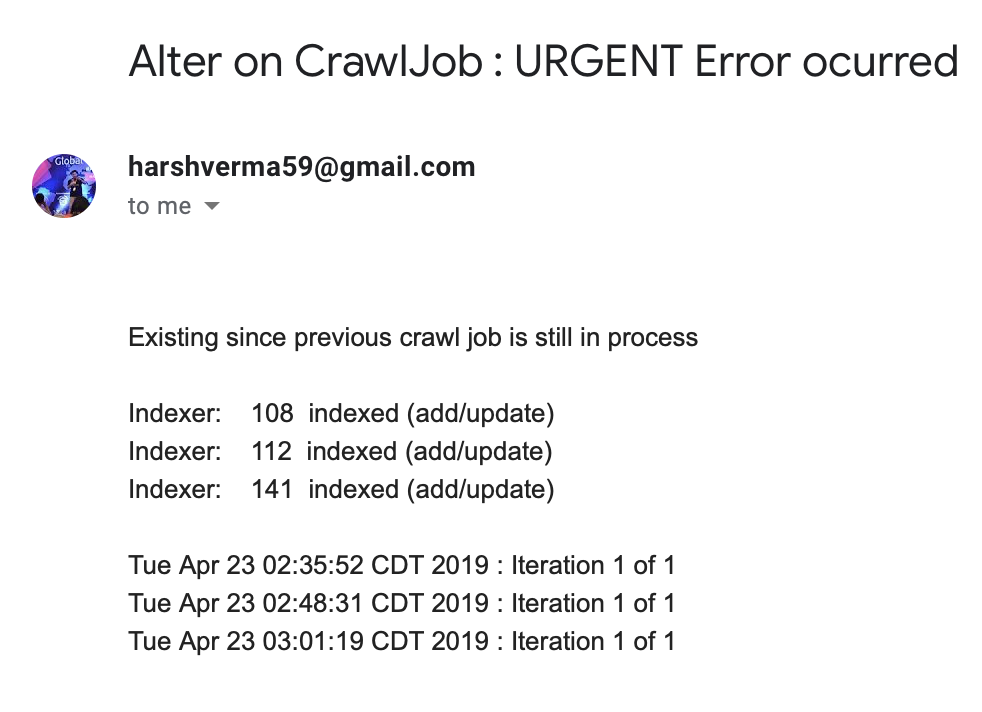
**Monitoring for Nutch:**

Apache Nutch 1.X doesn’t have web-ui , to monitor the jobs and logs error and manage crawling.

Hence, I created a email script where after every couple of crawl jobs, it reads the logs and send email to me, giving details of the jobs like :

* Number of jobs ran
* Number of urls processed.
* Amount of time each job took.
* Urls where Error Occurred , along with error
* Report failed Jobs.

**Screenshot of Email ->>>**



**CONCLUSION**

Apache Nutch is highly scalable, robust and relatively feature rich crawler. Quality – crawling can be biased to fetch "important" pages first. If we use Apache Nutch on AWS, Google Cloud Computing it can be clustered among 100’s of machine at a time which can result in to fast performance or can be run in distributed mode using big-data mapreduce format.