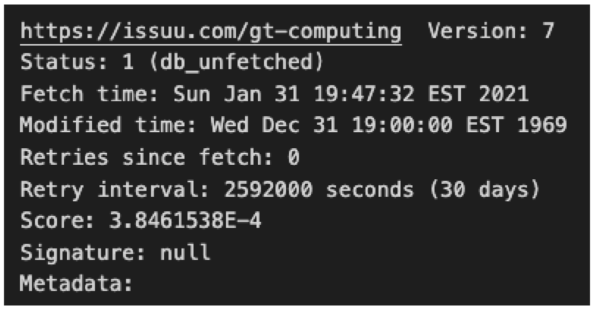
Homework 1
Yuyi Qu
Session: CS6675
Problem 1. Hand-on Experience with a Web Crawler
Option 1.1: Experience with an Open Source Crawlers

Introduction
A web crawler is a program that automatically browses and further explores the internet. It is essential for a web search engine to build its search index. In this homework, I use Apache Nutch, which is a well matured and production ready web crawler, to crawl thousands of web pages. I then build an index using the crawled pages via Apache Solr. Finally, my program can take in a query and return the web pages that are related to this query. The crawl statistics are plotted and by analyzing the crawl speed, a prediction is made about the time needed to crawl a relatively larger number of pages.

Design of Web Archive
In order to store the web pages crawled, the web archive is composed of a crawl database, a link database, and a set of segments, each of which records the crawling data that are crawled as a unit.

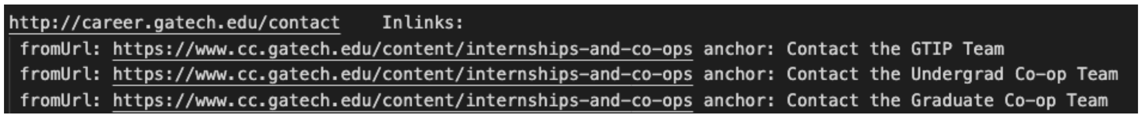
- Crawl database
  The crawl database stores the URLs known to the crawler, both fetched or not fetched yet. For each URL, the database stores the metadata of it, such as the fetching time, retry interval, its status, etc. An example of one entry in the crawl database is shown in Figure 1.



```
https://issuu.com/gt-computing  Version: 7
Status: 1 (db_unfetched)
Fetch time: Sun Jan 31 19:47:32 EST 2021
Modified time: Wed Dec 31 19:00:00 EST 1969
Retries since fetch: 0
Retry interval: 2592000 seconds (30 days)
Score: 3.8461538E-4
Signature: null
Metadata:
```

Figure 1. Screenshot of one entry stored in the crawl database.

- Link database
  The link database stores the URLs known to the crawled, and for each of which, it stores the source URLs and the anchor text (the clickable text on the source page that leads to this page). An example of one entry in the link database is shown in Figure 2.



```
http://career.gatech.edu/contact    Inlinks:
 fromUrl: https://www.cc.gatech.edu/content/internships-and-co-ops anchor: Contact the GTIP Team
 fromUrl: https://www.cc.gatech.edu/content/internships-and-co-ops anchor: Contact the Undergrad Co-op Team
 fromUrl: https://www.cc.gatech.edu/content/internships-and-co-ops anchor: Contact the Graduate Co-op Team
```

Figure 2. Screenshot of one entry stored in the link database.

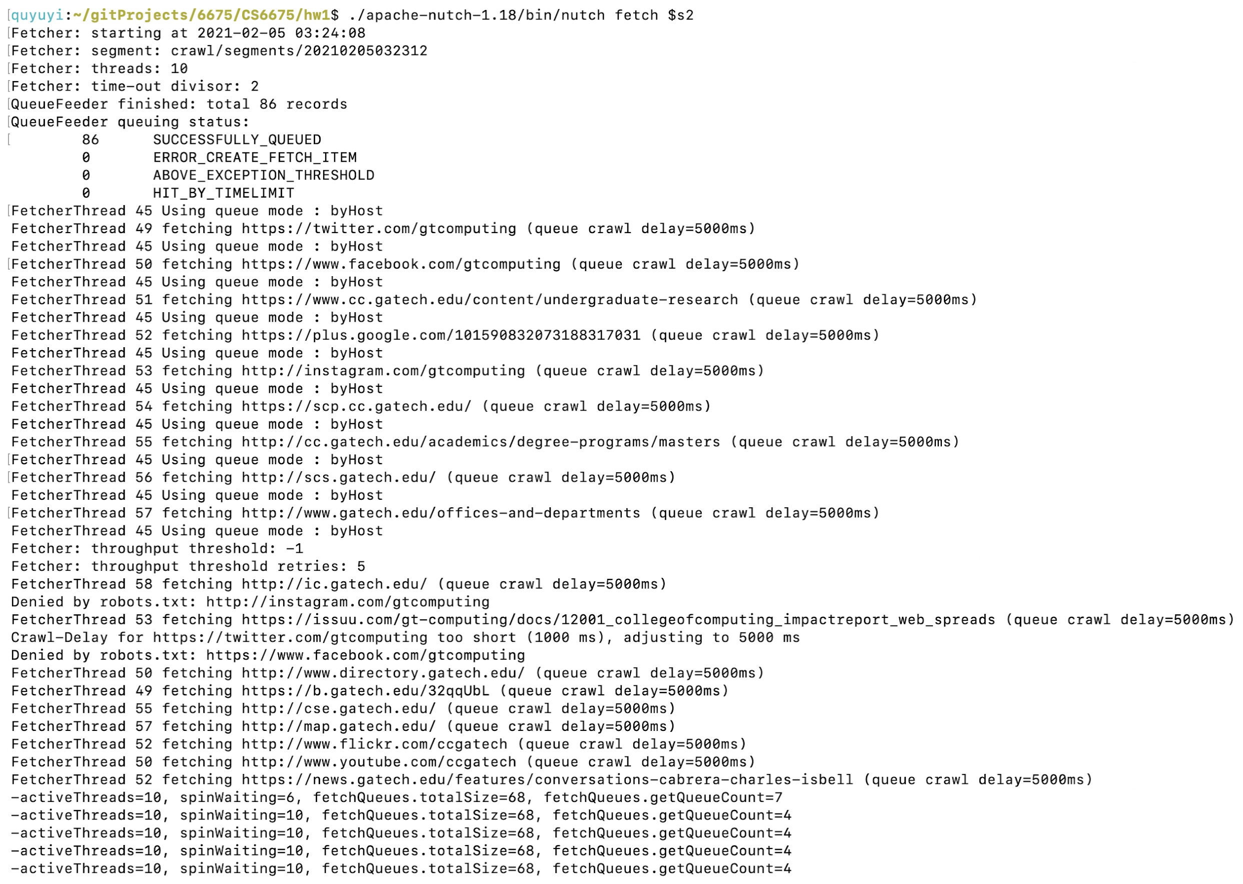
Each segment is further composed of crawl_generate, crawl_fetch, content, parse_text, parse_data, parse_crawl.

- crawl_generate: This stores the fetch list due to be fetched for this segment generated from the crawl database.
- crawl_fetch: This stores the status of fetching each URL.
- content: This stores the content retrieved from each URL.
- parse_text: This is obtained by parsing the text in the content for each URL.
- parse_data: This is obtained by parsing the outlink URLs and metadata in the content for each URL.
- parse_crawl: This contains the outlink URLs, which will be used to update the crawl database in the next round.

The keywords extracted for building the index includes the tokenized words of the title and parsed text content of each URL.

Runtime Screenshots

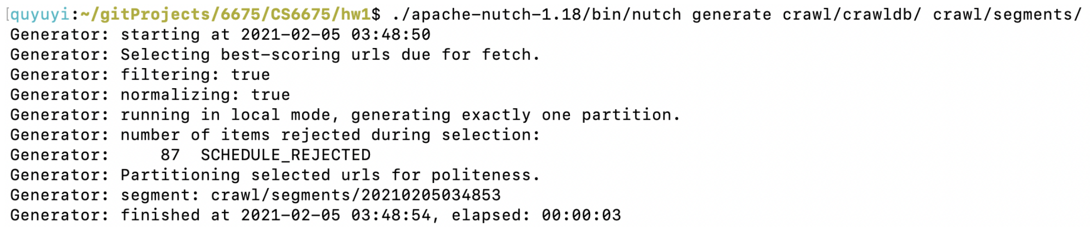
Figure 3. shows the screen of the CLI when running nutch fetch for the second round. We can see that 86 records are successfully queued, which are records extracted from the first round and are due to be fetched in this round.



```
quyuyi:~/gitProjects/6675/CS6675/hw1$ ./apache-nutch-1.18/bin/nutch fetch $s2
Fetcher: starting at 2021-02-05 03:24:08
Fetcher: segment: crawl/segments/20210205032312
Fetcher: threads: 10
Fetcher: time-out divisor: 2
QueueFeeder finished: total 86 records
QueueFeeder queuing status:
        86      SUCCESSFULLY_QUEUED
        0       ERROR_CREATE_FETCH_ITEM
        0       ABOVE_EXCEPTION_THRESHOLD
        0       HIT_BY_TIMELIMIT
FetcherThread 45 Using queue mode : byHost
FetcherThread 49 fetching https://twitter.com/gtcomputing (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 50 fetching https://www.facebook.com/gtcomputing (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 51 fetching https://www.cc.gatech.edu/content/undergraduate-research (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 52 fetching https://plus.google.com/101590832073188317031 (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 53 fetching http://instagram.com/gtcomputing (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 54 fetching https://scp.cc.gatech.edu/ (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 55 fetching http://cc.gatech.edu/academics/degree-programs/masters (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 56 fetching http://scs.gatech.edu/ (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
FetcherThread 57 fetching http://www.gatech.edu/offices-and-departments (queue crawl delay=5000ms)
FetcherThread 45 Using queue mode : byHost
Fetcher: throughput threshold: -1
Fetcher: throughput threshold retries: 5
FetcherThread 58 fetching http://ic.gatech.edu/ (queue crawl delay=5000ms)
Denied by robots.txt: http://instagram.com/gtcomputing
FetcherThread 53 fetching https://issuu.com/gt-computing/docs/12001_collegeofcomputing_impactreport_web_spreads (queue crawl delay=5000ms)
Crawl-Delay for https://twitter.com/gtcomputing too short (1000 ms), adjusting to 5000 ms
Denied by robots.txt: https://www.facebook.com/gtcomputing
FetcherThread 50 fetching http://www.directory.gatech.edu/ (queue crawl delay=5000ms)
FetcherThread 49 fetching https://b.gatech.edu/32qqUbL (queue crawl delay=5000ms)
FetcherThread 55 fetching http://cse.gatech.edu/ (queue crawl delay=5000ms)
FetcherThread 57 fetching http://map.gatech.edu/ (queue crawl delay=5000ms)
FetcherThread 52 fetching http://www.flickr.com/ccgatech (queue crawl delay=5000ms)
FetcherThread 50 fetching http://www.youtube.com/ccgatech (queue crawl delay=5000ms)
FetcherThread 52 fetching https://news.gatech.edu/features/conversations-cabrera-charles-isbell (queue crawl delay=5000ms)
-activeThreads=10, spinWaiting=6, fetchQueues.totalSize=68, fetchQueues.getQueueCount=7
-activeThreads=10, spinWaiting=10, fetchQueues.totalSize=68, fetchQueues.getQueueCount=4
-activeThreads=10, spinWaiting=10, fetchQueues.totalSize=68, fetchQueues.getQueueCount=4
-activeThreads=10, spinWaiting=10, fetchQueues.totalSize=68, fetchQueues.getQueueCount=4
-activeThreads=10, spinWaiting=10, fetchQueues.totalSize=68, fetchQueues.getQueueCount=4
```

Figure 3. Screenshot of running $ bin/nutch fetch <segment>.

Figure 4. shows the screenshot of running butch generate to generate the list of URLs to be crawled in next round from the updated crawldb. We can see that 87 SCHEDULE_REJECTED, since these are URLs in the crawldb which are fetched already and need not be re-fetched for now. This command creates a new fetch segment named using the current timestamp. For the fetch result or parse result for this round will be stored in this segment directory.



```
[quyuyi:~/gitProjects/6675/CS6675/hw1$ ./apache-nutch-1.18/bin/nutch generate crawl/crawldb/ crawl/segments/
Generator: starting at 2021-02-05 03:48:50
Generator: Selecting best-scoring urls due for fetch.
Generator: filtering: true
Generator: normalizing: true
Generator: running in local mode, generating exactly one partition.
Generator: number of items rejected during selection:
Generator:     87  SCHEDULE_REJECTED
Generator: Partitioning selected urls for politeness.
Generator: segment: crawl/segments/20210205034853
Generator: finished at 2021-02-05 03:48:54, elapsed: 00:00:03
```

Figure 4. Screenshot of running $ bin/nutch generate <crawldb> <segment>.

Crawl Statistics

The results are obtained on MacBook Pro (13-inch, 2019, Four Thunderbolt 3 ports), with processor being 2.4 GHz Quad-Core Intel Core i5, and memory being 16 GB 2133 MHz LPDDR3. A total of 5 rounds are run with about 4 hours. A total of 7582 pages are attempted to be crawled, and 5994 of them are successfully crawled. A total of 51162 pages are extracted from the crawling process.

Figure 5. shows the plot of #URLs crawled (blue) and #URLs successfully crawled (red) with respect to the time used. Failure to crawl a URL occurs when the page is gone, redirected or other exception. The figure shows no clear trend in the change of the crawl speed. I think the crawl speed is dependent on the outgoing links on a page. Since the crawler needs to extract as much as possible URLs that are linked from a page, the more links there are, the more time it will spend on the crawling on this page. Besides, due to the configuration of the robots.txt, the crawler needs to wait a certain amount of time between two extraction requests, which will also affect the crawl speed. Therefore, the average crawl speed is 7582 URLs crawled / 201 min = 37.8 pages/min.

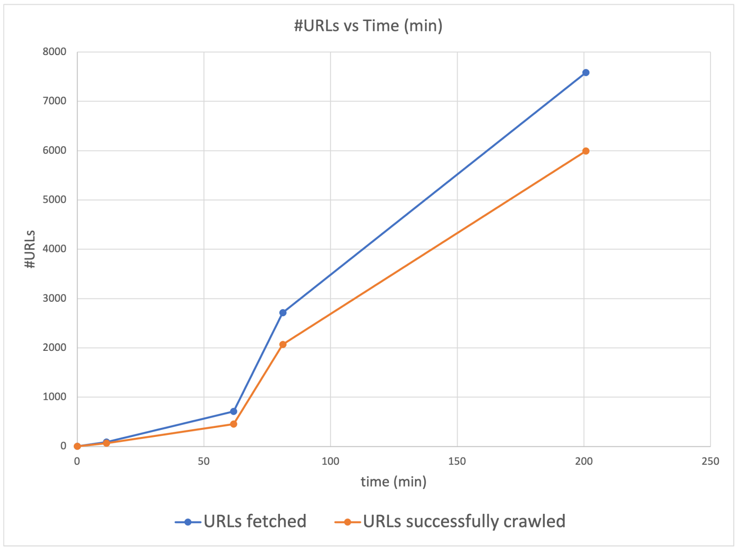


Figure 5. #URLs crawled and #URLs successfully crawled vs time.

Figure 6. shows the plot of #URLs extracted with respect to the time used. As time proceeds, the speed of extracting URLs is increasing. I think this is because as the crawldb gets to know more are more URLs, it can explore more new URLs instead of coming across the same URLs extracted before. Figure 7. shows the plot of the ratio of #URLs crawled to #URLs extracted with respect to time. We can see there is a trend that the ratio decreases as time proceeds. This is because the time used to extract a new URL is way less than the time to crawl a URL, and by crawling a URL, several URLs will be extracted from this single one URL. Therefore the growth rate of #URLs crawled should be way less than the growth rate of the #URLs extracted.

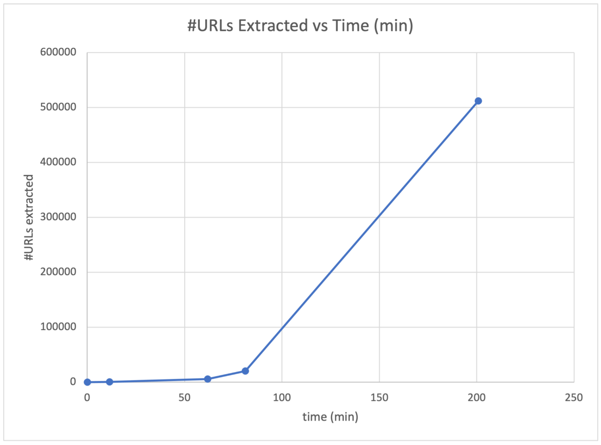


Figure 6. #URLs extracted vs time.

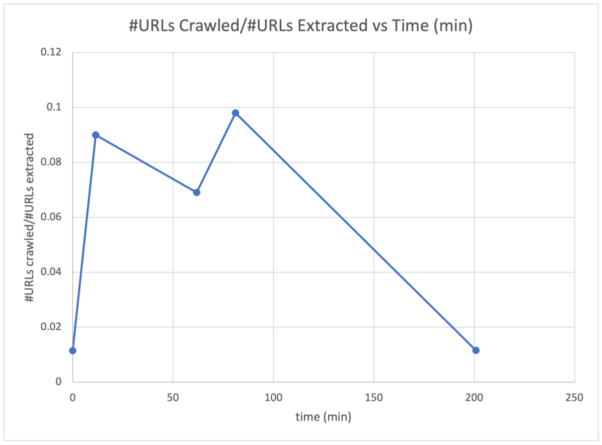
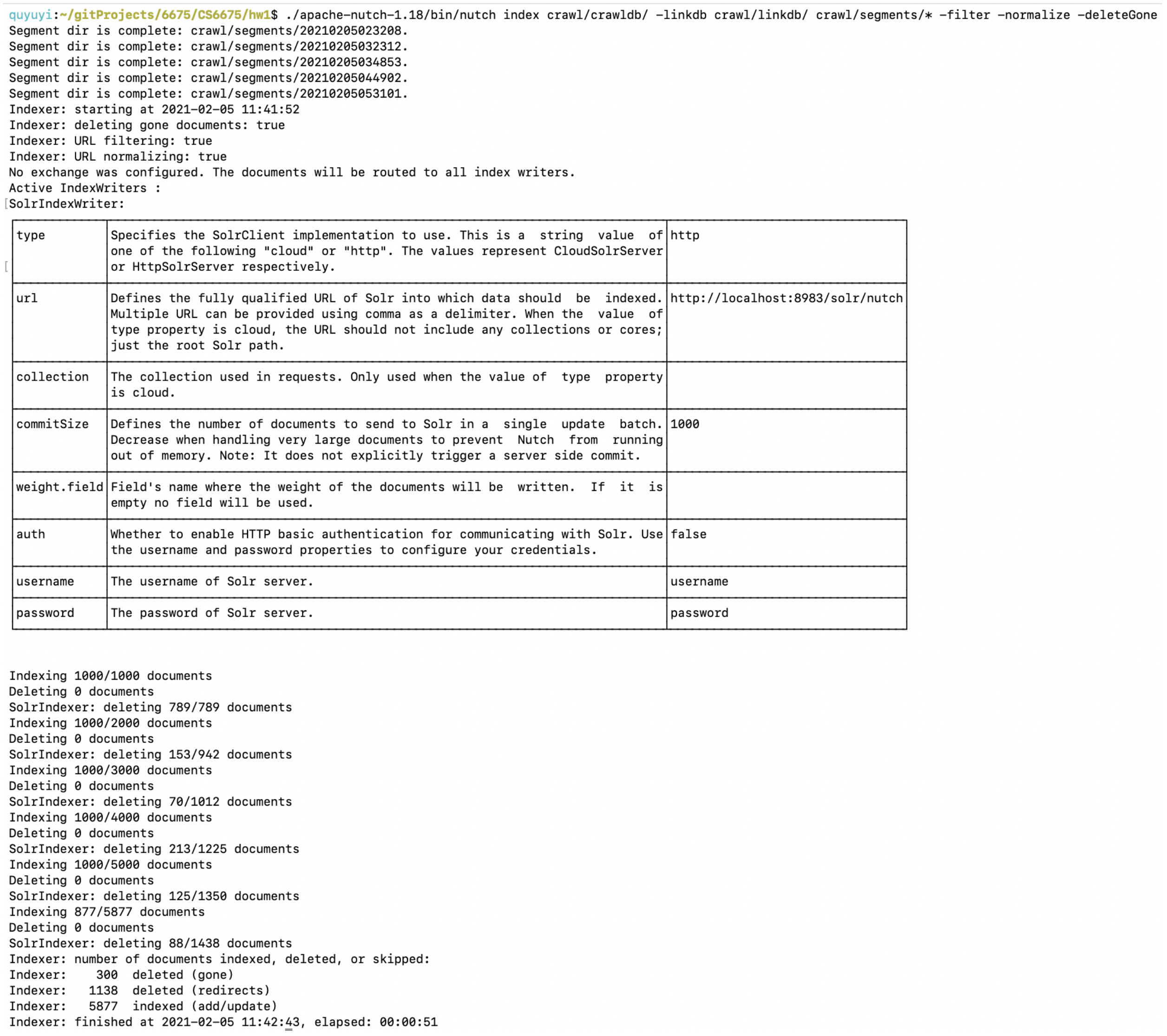


Figure 7. #URLs crawled/extracted vs time.

Search

After crawling about thousands of pages, I use Apache Solr to build a search index. Figure 8. is the screenshot of running the index command. We can see that out of 5932 documents that are fetched, 5877 documents are indexed, and the rest are deleted due to page gone or page redirects. Start running the Solr server and navigate to http://localhost:8983/solr/ and select the nutch core, there is a GUI to do search query as well as view the search results.

Figure 9 shows searching the keyword “algorithm” in the title field of the URLs. The right side is the response in json format. We can see that out of 5877 documents indexed, there are 29 matched documents which contain the “algorithm” in their title. We can also search the “content” field of the URLs, since it is also indexed by Solr.

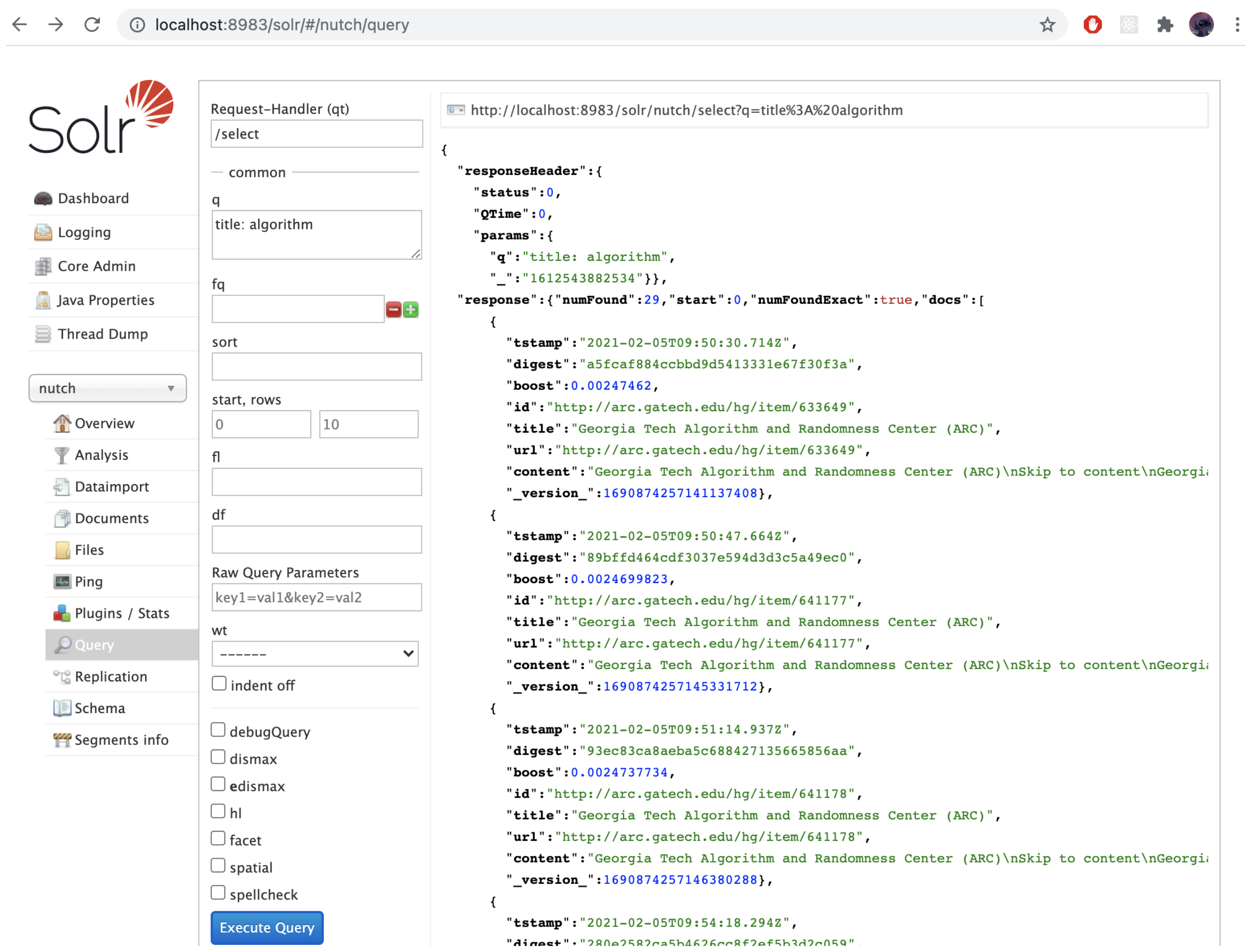


```
quyuyi:~/gitProjects/6675/CS6675/hw1$ ./apache-nutch-1.18/bin/nutch index crawl/crawldb/ -linkdb crawl/linkdb/ crawl/segments/* -filter -normalize -deleteGone
Segment dir is complete: crawl/segments/20210205023208.
Segment dir is complete: crawl/segments/20210205032312.
Segment dir is complete: crawl/segments/20210205034853.
Segment dir is complete: crawl/segments/20210205044902.
Segment dir is complete: crawl/segments/20210205053101.
Indexer: starting at 2021-02-05 11:41:52
Indexer: deleting gone documents: true
Indexer: URL filtering: true
Indexer: URL normalizing: true
No exchange was configured. The documents will be routed to all index writers.
Active IndexWriters :
SolrIndexWriter:
```

type	Specifies the SolrClient implementation to use. This is a string value of one of the following "cloud" or "http". The values represent CloudSolrServer or HttpSolrServer respectively.	http
url	Defines the fully qualified URL of Solr into which data should be indexed. Multiple URL can be provided using comma as a delimiter. When the value of type property is cloud, the URL should not include any collections or cores; just the root Solr path.	http://localhost:8983/solr/nutch
collection	The collection used in requests. Only used when the value of type property is cloud.	
commitSize	Defines the number of documents to send to Solr in a single update batch. Decrease when handling very large documents to prevent Nutch from running out of memory. Note: It does not explicitly trigger a server side commit.	1000
weight.field	Field's name where the weight of the documents will be written. If it is empty no field will be used.	
auth	Whether to enable HTTP basic authentication for communicating with Solr. Use the username and password properties to configure your credentials.	false
username	The username of Solr server.	username
password	The password of Solr server.	password

```
Indexing 1000/1000 documents
Deleting 0 documents
SolrIndexer: deleting 789/789 documents
Indexing 1000/2000 documents
Deleting 0 documents
SolrIndexer: deleting 153/942 documents
Indexing 1000/3000 documents
Deleting 0 documents
SolrIndexer: deleting 70/1012 documents
Indexing 1000/4000 documents
Deleting 0 documents
SolrIndexer: deleting 213/1225 documents
Indexing 1000/5000 documents
Deleting 0 documents
SolrIndexer: deleting 125/1350 documents
Indexing 877/5877 documents
Deleting 0 documents
SolrIndexer: deleting 88/1438 documents
Indexer: number of documents indexed, deleted, or skipped:
Indexer:    300  deleted (gone)
Indexer:   1138  deleted (redirects)
Indexer:   5877  indexed (add/update)
Indexer: finished at 2021-02-05 11:42:43, elapsed: 00:00:51
```

Figure 8. Screenshot of Statistics of Solr.



localhost:8983/solr/#/nutch/query

Request-Handler (qt): /select
q: title: algorithm
fq:
sort:
start, rows: 0 10
fl:
df:
Raw Query Parameters: key1=val1&key2=val2
wt: ------
indent off

http://localhost:8983/solr/nutch/select?q=title%3A%20algorithm

```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"title: algorithm",
      "_":"1612543882534"}},
  "response":{"numFound":29,"start":0,"numFoundExact":true,"docs":[
      {
        "tstamp":"2021-02-05T09:50:30.714Z",
        "digest":"a5fcaf884ccbbd9d5413331e67f30f3a",
        "boost":0.00247462,
        "id":"http://arc.gatech.edu/hg/item/633649",
        "title":"Georgia Tech Algorithm and Randomness Center (ARC)",
        "url":"http://arc.gatech.edu/hg/item/633649",
        "content":"Georgia Tech Algorithm and Randomness Center (ARC)\nSkip to content\nGeorgi
        "_version_":1690874257141137408},
      {
        "tstamp":"2021-02-05T09:50:47.664Z",
        "digest":"89bffd464cdf3037e594d3d3c5a49ec0",
        "boost":0.0024699823,
        "id":"http://arc.gatech.edu/hg/item/641177",
        "title":"Georgia Tech Algorithm and Randomness Center (ARC)",
        "url":"http://arc.gatech.edu/hg/item/641177",
        "content":"Georgia Tech Algorithm and Randomness Center (ARC)\nSkip to content\nGeorgi
        "_version_":1690874257145331712},
      {
        "tstamp":"2021-02-05T09:51:14.937Z",
        "digest":"93ec83ca8aeba5c688427135665856aa",
        "boost":0.0024737734,
        "id":"http://arc.gatech.edu/hg/item/641178",
        "title":"Georgia Tech Algorithm and Randomness Center (ARC)",
        "url":"http://arc.gatech.edu/hg/item/641178",
        "content":"Georgia Tech Algorithm and Randomness Center (ARC)\nSkip to content\nGeorgi
        "_version_":1690874257146380288},
      {
        "tstamp":"2021-02-05T09:54:18.294Z",
        "digest":"280e2582ca5b4626cc8f2ef5b3d2c059",
```

Figure 9. Screenshot of Search on Solr.

Discussion

The crawl database keeps track of the URLs known to the crawler with different status. By inspecting the crawldb, I found several status types such as db_unfetched, db_fetched, db_gone, db_redir_temp, db_redir_perm. “db_gone” are used to mark crawls that are denied by robots.txt of the URL. “redir” is used to mark that the URL is redirected. Another parameter in crawldb is retry interval, which marks the interval before refetch. Most retry interval is 2592000 seconds (30 days), which makes sense since most web pages don’t change that often and it saves the resources of the crawler by not refetching them too often.

When inspecting the log of the fetch command, I saw there is a crawl delay for each URL. It is the directive specified in robots.txt, which is used for the URL to communicate to the crawlers to not crawl too often and thus avoid being overwhelmed by crawl requests and remain available to normal requests.

As calculated in the crawl statistics section, the average crawl speed is 37.8 pages/min. Therefore, in order to crawl 10 million pages, the time needed for the crawler is 183.7 days. In order to crawl 1 billion pages, the time needed for the crawler is 50.3 years. This is a quite long time. The growing and changing speed of the Internet is absolutely much larger than the crawl speed, therefore it is of great necessity to build faster crawlers. For example, we can build distributed crawlers to collaborate for better crawling speed.

Appendix
Source code with README can be found at https://github.com/quyuyi/CS6675/tree/master/hw1.