

Course: Big Data Analytics

Professor: Dr. Zeng, Yinghui

Final Project: A Nutch-Based Search Engine with web-interface

Team Member: Guanghan Ning, Trung Nguyen

0. Scope of the Final Project:

Before choosing to work on Web Search Engine, we were inspired mainly by the Nutch paper in which Nutch search engine is described as an open-source project that is rarely available to software developers for researching, developing and deploying a web search engine. Nutch search engine was built on top of Hadoop and Hadoop Distributed File System (HDFS). And it also supports Hbase as one of options for storing data distribute on HDFS. Besides, Nutch is a medium-scale search engine which is suitable for organizational purposes, instead of commercial ones as other popular web search engines such as Google, Bing or Yahoo!. For that reason, unfortunately, those commercial web search engines are kept secret from curious developers like us. Therefore, our first purpose for our final project is to download the Nutch source code, install it, configure it and run it on local machine with Hbase at the first phase in order to fully understand how it works and what components or modules it has as the way we take to seek a deep understanding of what we learn from the course of Big Data Analytics. Based on our knowledge of Nutch Search engine, we will improve existing modules or develop new features for it as our second phase or future plan. We hope we could develop a new version or our own version of search engine based on Nutch to deploy it as commercial web search engine.

1. Introduction and Background:

Internet is our everyday source for all kinds of information, like news, broadcast, music, pictures, or even movies. We are immersed in the flood of big data every day. However, the information we need is only a small portion of it and for different individuals the importance of information is hardly the same. A search engine will make it convenient for us to absorb information efficiently. The search engine will do MapReduce for us humans to get the related and useful pages.

A Search engine consists of several parts: A web crawler, a web indexer, and a web searcher. Nutch is an open-source web crawler software project, which is highly extensible and scalable. It provides useful open source code for building a search engine.

The project aims at deploying Nutch on a commodity cluster provided by IBM and using hadoop to perform distributed computing on the web-crawling, indexing and

searching. But instead of using Nutch's default database, we are planning on using HBase to store the crawled data, process and transform the data that has been read into the HBase.

The canonical HBase use case is webtable, used for a search engine. HBase is a distributed column-oriented storage system or database for managing structured data; it is built on top of HDFS. It is an application appropriate for real-time read/write random access to very large datasets: petabytes of data across thousands of commodity servers. HBase is not relational and does not support SQL or any full relational data model. Given proper problem space, however, HBase can do what an RDBMS cannot do: host very large, sparsely populated tables on clusters made from commodity hardware. It provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of the data represented in the underlying storage.

By using the technology described above, we are able to build a search engine of our own. It is based on Nutch, whose injector is re-written or modified so that the HBase could be used as the database to store crawled data. Web searcher and web-interface is supported by Nutch as well, so that we can search text from a web interface.

2. Objectives:

We are using Nutch - an open-source effort to build a web search engine, as our web crawler to download the data from the internet, and we are using HBase to store the database, and perform MapReduce on the database, and we are going to build a simple web interface as front-end layer for users to search text. The project deploys Nutch on a commodity cluster provided by IBM and uses hadoop to perform distributed computing on the web-crawling, indexing and searching. Our search engine will be a focused search engine, which searches for a specific field of interest, e.g., books, film reviews, or soccer game results.

We are absorbing film information from the website IDMB (www.idmb.com), and store the data as our database, in order to meet the need of the users who search for a film review.

Also, we crawled IGN (www.ign.com) for game reviews. And we test the searching engine by searching key words from the games.

Other websites, like the *rotten tomatoes*, and the *metacritic.com*, are then crawled in order to incorporate and versatile our database.

3. System Requirements and Use Cases:

3.1 Tools:

3.1.1 Server

IBM cloud server (clusters we created) in Canada

OS: Red Hat Enterprise Linux v6. Copper - 64 bit (vCPU: 2, RAM: 4 GiB, Disk: 60 GiB)

OR

lewis, a cluster of multi-core compute servers operated by University of Missouri Bioinformatics Consortium.

OS: Platform OCS Linux 4.1.1 (based on Redhat Linux).

3.1.2 Programming Tools

HBase, Eclipse

3.1.3 Software/platform

Nutch, BigInSights

3.2 Why we chose BigInSights:

A search engine usually requires large space for storing data, as the tables will be very large. HBase is highly scalable and is quite fit for the database. The IBM cloud cluster provides up to ten nodes for us to use, and the disk for each node is extensible. So we choose IBM academic skills cloud as our server as it provides abundant space. Besides, InfoSphere BigInSights platform including Hadoop and HBase is pre-installed on every node, which is quite convenient for us to use.

3.3 Why we chose Nutch:

Building a web search engine from scratch is not eligible for us, for not only is the software required to crawl and index websites complex to write, but it is also a challenge to run in distribution on Hadoop. Since Hadoop has its origins in Apache Nutch, running it with Hadoop will not be a problem. Besides, many parts of the software that are not related to distributed computing and this course are already written and documented. We can easily read the documentations and deploy it. And once we do, we can then focus on the data storage and data analysis part. Storage is provided by HDFS and analysis by Mapreduce.

3.4 Why we chose HBase as the database:

HBase is a distributed column-oriented storage system or database for managing structured data; it is built on top of HDFS. It is an application appropriate for real-time read/write random access to very large datasets. HBase can do what an RDBMS cannot do: host very large, sparsely populated tables on clusters made from commodity hardware. It provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of the data represented in the underlying storage. Since the webpages to be crawled can be considered huge dataset, and the tables should be easily extensible, HBase well fits this demand.

Row Key	column-family1		column-family2			column-family3
	column1	column2	column1	column2	column3	column1
key1						
key2						
key3						

Fig. 3.1 HBase Storage structure. Column Oriented.

3.5 Description of input data, input format:

The input data comes from the websites listed previously. The whole websites are crawled as the input data. Then the data from the websites are stored in the HBase in a specific fashion, which is designed to fetch film reviews or game reviews more easily. The collection of this intermediate data forms our database, from where we do the searching to test. Once a key word is typed in the web interface search engine, the searcher and indexer will search and query the webpage that we are looking for. The listed pages can be considered the final output. And we check if the search result is satisfactory.

4. Design and Methodology:

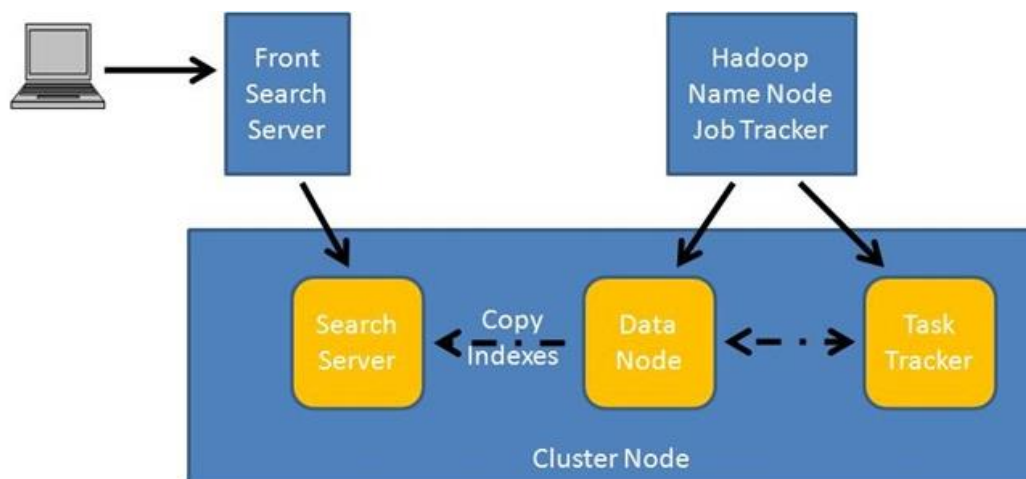


Fig. 4.1. Structure Overview.

As shown in Fig. 4.1 is the overview of the distribution of machines and their jobs.

The methodology is, deploying both Nutch and Hadoop on the cloud, which is consisted of one master node and five slave nodes. The architecture is designed in such a way in order to make the most of commodity hardware. The master node takes

the responsibility of storing metadata and assigning data nodes tasks. The data nodes stores the data crawled from the Internet.

The web crawler crawls the web pages specified by the user. It is a program which browses the web in a methodical and automated manner. And it not only keeps a copy of all the visited pages for later processing but also indexes these pages to make the search narrower. Its process includes locating, fetching, and storing pages on the web.

The web indexer does the job of indexing to make the search narrower. An inverted index is a representation for the document collection over which user queries will be evaluated. The pages with higher query scores will be listed above in the search result.

The Web searcher deals query processing. When a user searches something, the searcher will query pages that contain key words that are being searched.

BigTable will store the initial web pages crawled. Then through Mapreduce, the same pages will be filtered.

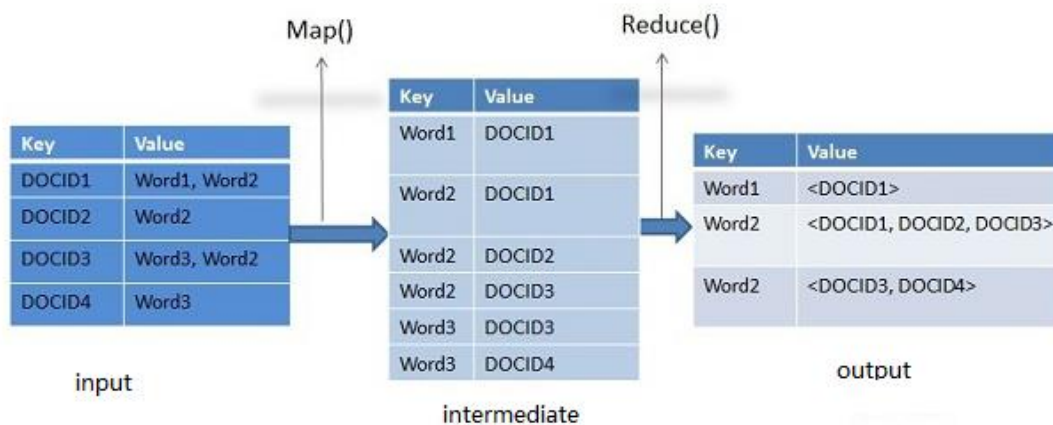


Fig. 4.2. MapReduce on the initial web pages.

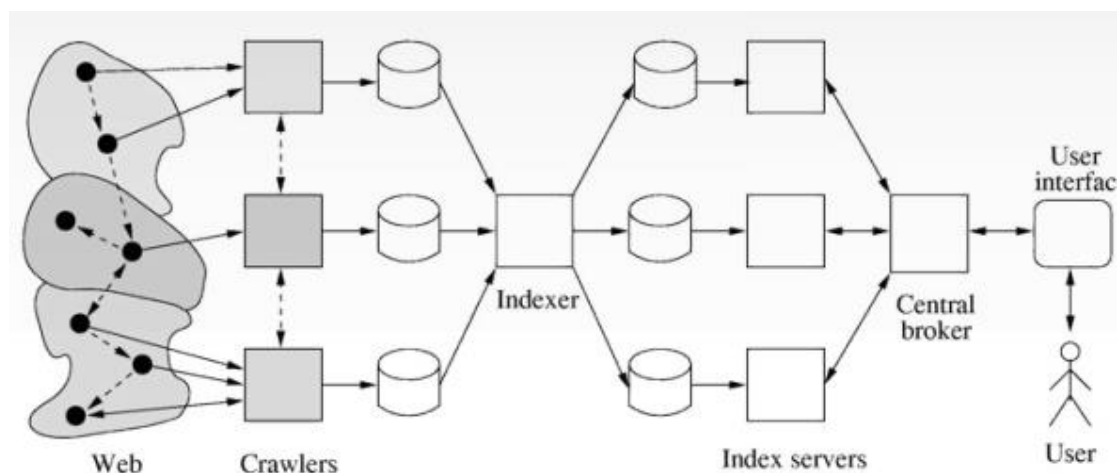


Fig. 4.3 Main components of a search engine: 1) crawler; 2) indexer; 3) query processing.

There are some advantages of distributed web crawling: (1). Higher crawling throughput with lower latency; (2). Improved network politeness with less overhead on routers; (3). Better coupling with distributed indexing/search.

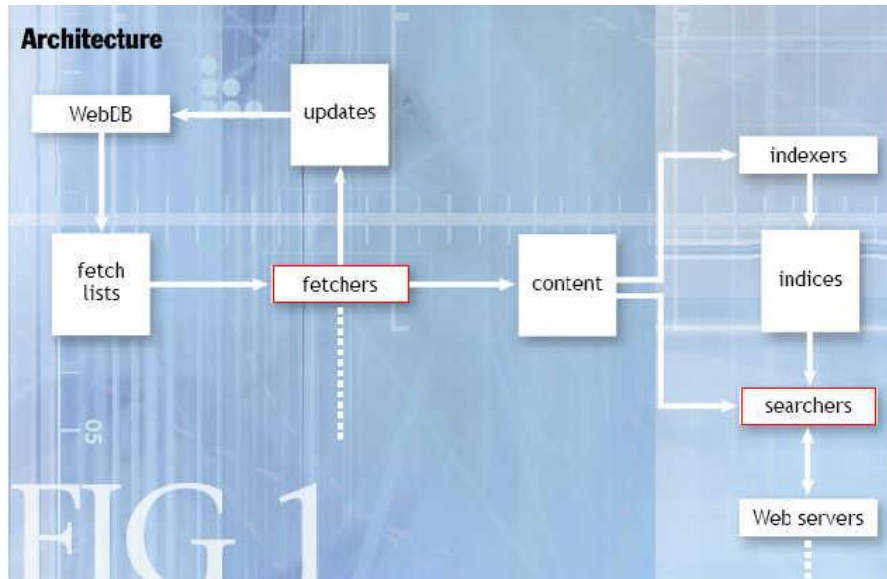


Fig. 4.4. Architecture of Nutch.

5. Implementation and Execution Results:

5.1 Implementation:

As we mentioned in the previous sections, a search engine consists of three components including crawler which is in charge of downloading web pages and store them into Hbase tables, indexer which is responsible for creating inverted index files or producing words in web pages as keys in Hbase tables, and searcher or querier which is in charge of querying data from Hbase tables.

For this project, we will use MapReduce in Hadoop to write Map and Reduce function for storing urls read for seed.txt file into Hbase tables as an example for getting to know how Nutch works.

For Injector module, we have two main classes as following:

5.1.1 Class *UrlMapper* is responsible for storing url data into Hbase tables

```
/**
 * Map web pages information including urls, web content, length, ... into Hbase tables
 */
public static class UrlMapper extends Mapper<LongWritable, Text, String, WebPage> {
    private URLNormalizers urlNormalizers;
    private int interval;
    private float scoreInjected;
    private URLFilters filters;
    private ScoringFilters scfilters;
    private long curTime;
```

```

    @Override
    protected void setup(Context context) throws IOException, InterruptedException {
    }

    protected void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {};
}

```

5.1.2 Class InjectorJob is responsible for getting urls from seed.txt file, and then uses UrlMapper class to store that data into Hbase tables.

```

public class InjectorJob extends NutchTool implements Tool {

    /**
     * Map information relating to web pages downloaded from the Internet into Hbase tables
     */
    protected void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {};

    /**
     * Default constructor function
     */
    public InjectorJob() {};

    /**
     * Constructor function with configuration information
     */
    public InjectorJob(Configuration conf) {};

    /**
     * Map function to do injection job
     */
    public Map<String, Object> run(Map<String, Object> args) throws Exception {};

    /**
     * Do downloading web pages from input list of urls and inserting them into Hbase tables
     */
    public void inject(Path urlDir) throws Exception {};

    /**
     * Run module Injector
     * This function overrides run function in the NutchTool class
     */
}

```

```

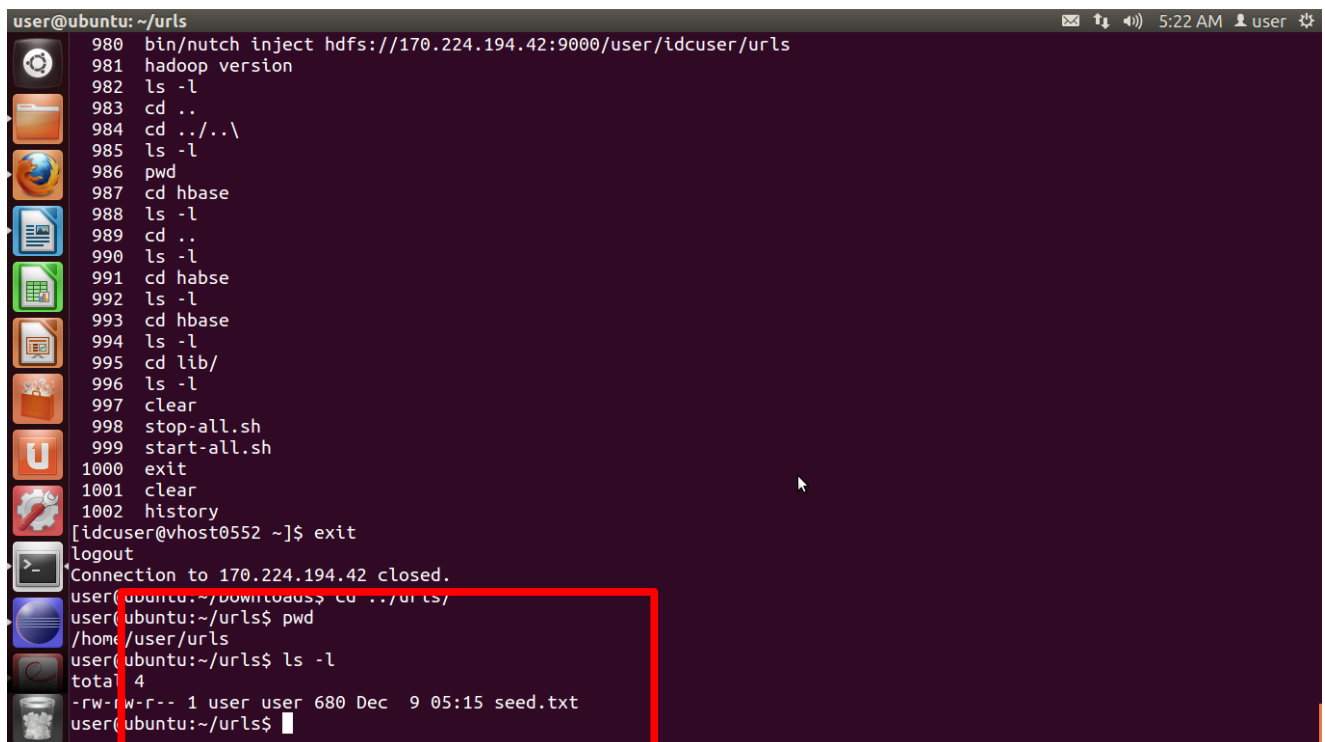
@Override
public int run(String[] args) throws Exception {};

/**
 * Main entry to run Injector to download web pages and inject them into Hbase tables
 */
public static void main(String[] args) throws Exception {
    int res = ToolRunner.run(NutchConfiguration.create(), new InjectorJob(), args);
    System.exit(res);
}
}

```

5.2 Execution Results:

seed.txt file is stored in /home/user/urls folder, and the file has 17 urls as following pictures:



```

user@ubuntu: ~/urls
980 bin/nutch inject hdfs://170.224.194.42:9000/user/idcuser/urls
981 hadoop version
982 ls -l
983 cd ..
984 cd ../../\
985 ls -l
986 pwd
987 cd hbase
988 ls -l
989 cd ..
990 ls -l
991 cd habse
992 ls -l
993 cd hbase
994 ls -l
995 cd lib/
996 ls -l
997 clear
998 stop-all.sh
999 start-all.sh
1000 exit
1001 clear
1002 history
[idcuser@vhost0552 ~]$ exit
logout
Connection to 170.224.194.42 closed.
user@ubuntu: ~/downloads$ cd ../urls/
user@ubuntu: ~/urls$ pwd
/home/user/urls
user@ubuntu: ~/urls$ ls -l
total 4
-rw-rw-r-- 1 user user 680 Dec  9 05:15 seed.txt
user@ubuntu: ~/urls$

```

Fig. 5.2. Seed.txt that contains the URLs of websites to be crawled.

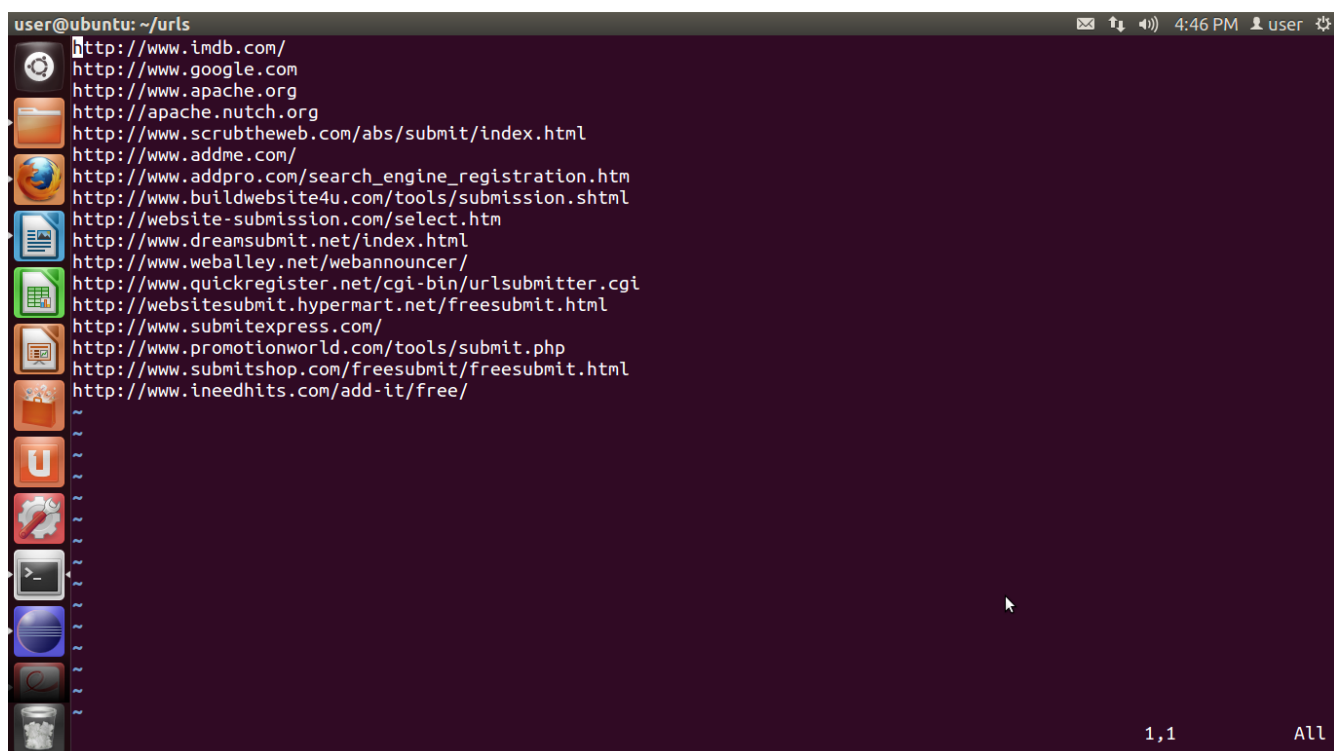


Fig. 5.3. URLs where websites are represented will be crawled.

After executing Injector module, we got the result as following:

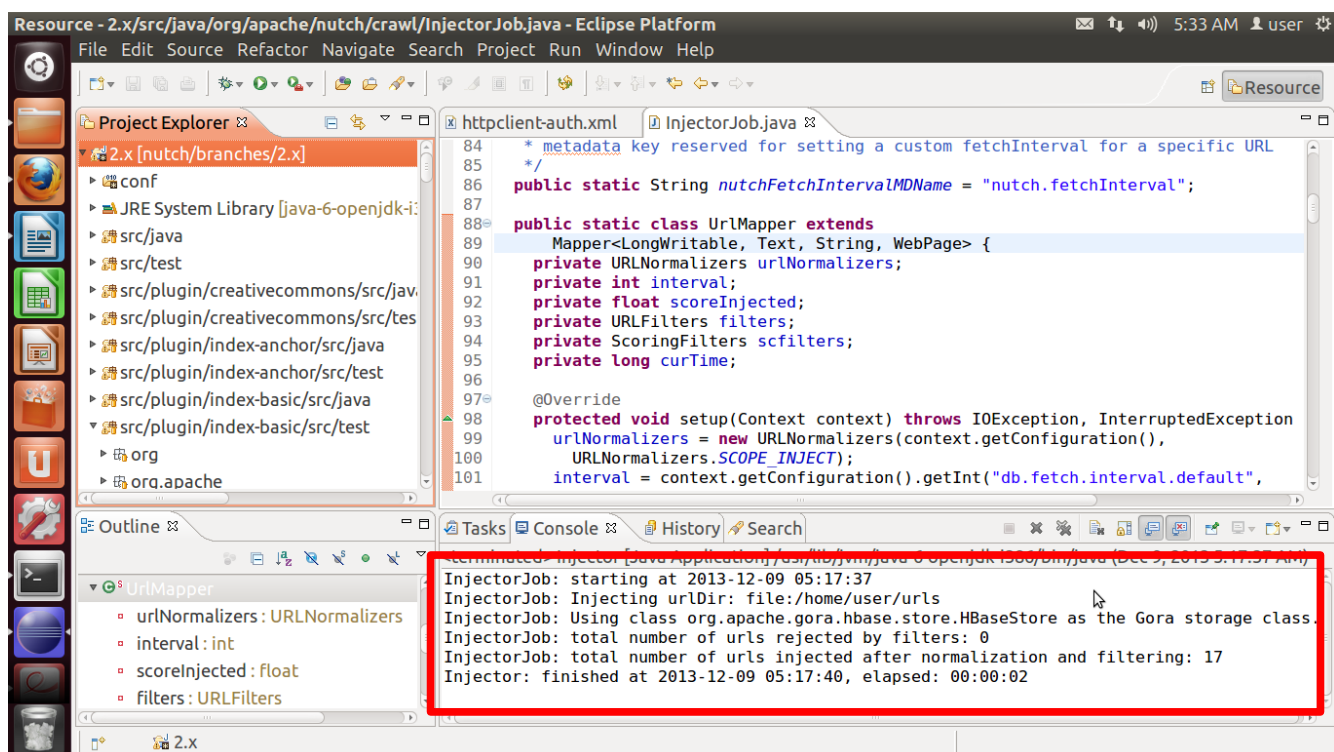


Fig. 5.4. Injector result.

6. Testing and Validation:

To test *InjectorJob* module, we will input a list of urls into seed.txt as previous section. Subsequently, we will execute the module from Eclipse to get the result. To verify whether or not the module insert the list of urls into Hbase table correctly, we will use Hbase Shell to test it as following.

```
user@ubuntu: ~/urls
hbase(main):001:0> list
TABLE
webpage
1 row(s) in 0.5710 seconds

hbase(main):002:0> describe 'webpage'
DESCRIPTION
{NAME => 'webpage', FAMILIES => [{NAME => 'f', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'h', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'il', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'mk', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'mtdt', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'ol', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'p', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 's', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}]}
1 row(s) in 0.0690 seconds

hbase(main):003:0> describe 'webpage'
DESCRIPTION
{NAME => 'webpage', FAMILIES => [{NAME => 'f', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'h', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'il', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'mk', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'mtdt', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'ol', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'p', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 's', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '1', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}]}
1 row(s) in 0.0690 seconds
```

Fig. 6.1 Describe the “webpage” table, to check the data structure with which data is stored in HBase.

```
user@ubuntu: ~/urls
hbase(main):006:0> count 'webpage';
hbase(main):007:0> count 'webpage'
18 row(s) in 0.0900 seconds

hbase(main):008:0> scan 'webpage'
ROW COLUMN+CELL
com.addme.www:http/ column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
com.addme.www:http/ column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
com.addme.www:http/ column=mk:injmrk_, timestamp=1386631442970, value=y
com.addme.www:http/ column=mk:dist, timestamp=1386631442970, value=0
com.addme.www:http/ column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
com.addpro.www:http/search_en column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
com.addpro.www:http/search_en column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
com.addpro.www:http/search_en column=mk:injmrk_, timestamp=1386631442970, value=y
com.addpro.www:http/search_en column=mk:dist, timestamp=1386631442970, value=0
com.addpro.www:http/search_en column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
com.buildwebsite4u.www:http/t column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
com.buildwebsite4u.www:http/t column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
com.buildwebsite4u.www:http/t column=mk:injmrk_, timestamp=1386631442970, value=y
com.buildwebsite4u.www:http/t column=mk:dist, timestamp=1386631442970, value=0
com.buildwebsite4u.www:http/t column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
ools/submission.shtml
```

Fig. 6.2. Scan to display data of all columns. (a)

```

user@ubuntu: ~/urls
/free/
com.ineedhits.www:http/add-it column=mk:dist, timestamp=1386631442970, value=0
/free/
com.ineedhits.www:http/add-it column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
/free/
com.promotionworld.www:http/t column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
ools/submit.php
com.promotionworld.www:http/t column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
ools/submit.php
com.promotionworld.www:http/t column=mk:_injmrk_, timestamp=1386631442970, value=y
ools/submit.php
com.promotionworld.www:http/t column=mk:dist, timestamp=1386631442970, value=0
ools/submit.php
com.promotionworld.www:http/t column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
ools/submit.php
com.scrubtheweb.www:http/abs/ column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
submit/index.html
com.scrubtheweb.www:http/abs/ column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
submit/index.html
com.scrubtheweb.www:http/abs/ column=mk:_injmrk_, timestamp=1386631442970, value=y
submit/index.html
com.scrubtheweb.www:http/abs/ column=mk:dist, timestamp=1386631442970, value=0
submit/index.html
com.scrubtheweb.www:http/abs/ column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
submit/index.html
com.submitexpress.www:http/ column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
com.submitexpress.www:http/ column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
com.submitexpress.www:http/ column=mk:_injmrk_, timestamp=1386631442970, value=y
com.submitexpress.www:http/ column=mk:dist, timestamp=1386631442970, value=0
com.submitexpress.www:http/ column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
com.submitshop.www:http/frees column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
ubmit/freesubmit.html
com.submitshop.www:http/frees column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD

```

Fig. 6.2. Scan to display data of all columns (b)

```

user@ubuntu: ~/urls
com.website-submission:http/s column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
elect.htm
net.dreamsubmit.www:http/inde column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
x.html
net.dreamsubmit.www:http/inde column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
x.html
net.dreamsubmit.www:http/inde column=mk:_injmrk_, timestamp=1386631442970, value=y
x.html
net.dreamsubmit.www:http/inde column=mk:dist, timestamp=1386631442970, value=0
x.html
net.dreamsubmit.www:http/inde column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
x.html
net.hypermart.websitesubmit:h column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
ttp/freesubmit.html
net.hypermart.websitesubmit:h column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
ttp/freesubmit.html
net.hypermart.websitesubmit:h column=mk:_injmrk_, timestamp=1386631442970, value=y
ttp/freesubmit.html
net.hypermart.websitesubmit:h column=mk:dist, timestamp=1386631442970, value=0
ttp/freesubmit.html
net.hypermart.websitesubmit:h column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
ttp/freesubmit.html
net.quickregister.www:http/cg column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
i-bin/urlsubmitter.cgi
net.quickregister.www:http/cg column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
i-bin/urlsubmitter.cgi
net.quickregister.www:http/cg column=mk:_injmrk_, timestamp=1386631442970, value=y
i-bin/urlsubmitter.cgi
net.quickregister.www:http/cg column=mk:dist, timestamp=1386631442970, value=0
i-bin/urlsubmitter.cgi
net.quickregister.www:http/cg column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
i-bin/urlsubmitter.cgi
net.thebestfree.www:http/prom column=f:fi, timestamp=1386587859623, value=\x00'\x8D\x00

```

Fig. 6.2. Scan to display data of all columns. (c)

```

user@ubuntu: ~/urls
net.thebestfree.www:http/prom column=f:fi, timestamp=1386587859623, value=\x00'\x8D\x00
o/submit.htm
net.thebestfree.www:http/prom column=f:ts, timestamp=1386587859623, value=\x00\x00\x01B\xD7\x13\xA3\xF2
o/submit.htm
net.thebestfree.www:http/prom column=mk:_injmrk_, timestamp=1386587859623, value=y
o/submit.htm
net.thebestfree.www:http/prom column=mk:dist, timestamp=1386587859623, value=0
o/submit.htm
net.thebestfree.www:http/prom column=s:s, timestamp=1386587859623, value=?\x80\x00\x00
o/submit.htm
net.weballey.www:http/webanno column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
uncer/
net.weballey.www:http/webanno column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
uncer/
net.weballey.www:http/webanno column=mk:_injmrk_, timestamp=1386631442970, value=y
uncer/
net.weballey.www:http/webanno column=mk:dist, timestamp=1386631442970, value=0
uncer/
net.weballey.www:http/webanno column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
uncer/
org.apache.www:http/ column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
org.apache.www:http/ column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
org.apache.www:http/ column=mk:_injmrk_, timestamp=1386631442970, value=y
org.apache.www:http/ column=mk:dist, timestamp=1386631442970, value=0
org.apache.www:http/ column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
org.nutch.apache:http/ column=f:fi, timestamp=1386631442970, value=\x00'\x8D\x00
org.nutch.apache:http/ column=f:ts, timestamp=1386631442970, value=\x00\x00\x01B\xD9\xAC\xA7\xBD
org.nutch.apache:http/ column=mk:_injmrk_, timestamp=1386631442970, value=y
org.nutch.apache:http/ column=mk:dist, timestamp=1386631442970, value=0
org.nutch.apache:http/ column=s:s, timestamp=1386631442970, value=?\x80\x00\x00
18 row(s) in 0.9850 seconds
hbase(main):009:0>

```

Fig. 6.2 Scan to display data of all columns. (d)

After using Hbase Shell to query the content of an Hbase table *webpage*, the result meets with our expectation.

7. Problems and solutions:

7.1. Problems

Nutch is able to run on two modes including local and distributed mode as Hadoop does. The local mode is mainly for testing and development purpose. On the other hand, distributed mode is for executing or deploying the web search engine on a cluster of machines, so that users can do searching, which is served by the Nutch Search Engine.

To run Nutch in local mode, we use Nutch 2.2.1 – the newest and stable version of Nutch web search engine which also supports Hbase as a Database system for storing big data crawled from the Internet. Nutch 1.7 is also a stable version for users to work on, however, it does not support Hbase. Thus, Nutch 1.7 is not a choice for us. On the other hand, Nutch 2.2.1 supports Hbase 0.90.4, which allows us to store huge amount of crawled data into Hbase based big tables. In order to run Nutch 2.2.1 on a local machine for debug purpose, we need first to install, configure and run Hadoop on that machine. For our final project we chose Hadoop 1.2.1. At the first time, we installed Hbase-0.94.13 on our local machine along with Hadoop 1.2.1 for running Nutch 2.2.1 on. Subsequently, we built the Nutch 2.2.1 source code downloaded with ant, and has been using Eclipse Indigo 3.7.2 to import the code for further development. However, when we first executed Injector module of Nutch search engine we got an error which says “InjectorJob: org.apache.gora.util.GoraException: java.lang.RuntimeException: java.lang.IllegalArgumentException: Not a host:port

pair...”. As far as we know, Nutch Search Engine uses storage backend Gora to set option for database such as Hbase.

To run Nutch in the distributed mode on a cluster of machines, we use IBM cloud which was provided from the course. We created one Master node and 5 slave nodes for deploying Nutch 2.2.1. On IBM cloud, Master and slave nodes support Hadoop 1.0.3, and on the Master node, Hbase version 0.94.0-security is supported. In addition, zookeeper-3.4.3 is also installed along with Hadoop 1.0.3.

Having Nutch 2.2.1 configured and built on Master node by using apache-ant, we execute Nutch 2.2.1 by running a command - “bin/nutch inject hdfs://170.224.194.42:9000/user/idcuser/urls”. However, we got the same error, “InjectorJob: org.apache.gora.util.GoraException: java.lang.RuntimeException: java.lang.IllegalArgumentException: Not a host:port pair...”.

7.2. Solutions

To solve the problem happened on the local machine, all we have to do is reinstall Hbase0.94.0, and instead, we changed to version 0.90.4 of Hbase to match with version of Hbase which is supported in Nutch. After reinstalling Hbase0.94.0, we ran Nutch 2.2.1 by using Eclipse, and it works just fine.

For the problem on IBM cloud, it turned out that it is not as easy as we used to assume it might be. We were trying several options to work the problem out. The first one is we decided to download Hbase0.90.4, and modify HBASE_HOME on the Master in order to run Hbase0.90.4 instead of Hbase0.94.0. However, it did not work out. Thus, we tried another option by copying Hbase0.90.4.jar file into default HBASE_HOME of the Master node, and ran it again. Nevertheless, we faced the same error. Then we sought the last resort by deleting all files in the default HBASE_HOME on the Master node, and moved all files of Hbase0.90.4 to the same location of Hbase0.94.0. We kept the same configuration files as they were in Hbase0.94.0. We still have the same error when we run the Nutch 2.2.1. We know Hbase is built on the top of Hadoop and it communicates with Zookeeper for managing and operating distributed files on HDFS.

So we draw a conclusion that to run Nutch 2.2.1 on a cluster of machines. We need to setup our own cluster with Hbase0.90.4 running on it. Or another option, we would run Nutch on other public cloud to see what is going on.

For that reason, with time limitation in this project, we just stop at running Nutch 2.2.1 on local machine and try to understand its features or the way it gains, stores and processes big data.

8. Conclusions:

We find out that the engine works well as a focused search engine, which can search information of a specific interest. This provides related and useful information for users, helps people absorb information more efficiently.

Future work can be done to add advanced search mode. Where we can search based on some logic, e.g., search results must include some key words, and may include some other key words, or must not include some key words. In addition, the search engine may search filter certain pages for censorship. Or search films whose average review score is above a certain value.

9. References:

- [1] <http://nutch.apache.org/>
- [2] <http://wiki.apache.org/nutch/NutchHadoopTutorial>
- [3] http://www.google.com/intl/en_us/insidesearch/howsearchworks/crawling-indexing.html
- [4] <http://hbase.apache.org/>
- [5] <http://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands/>
- [6] <http://wiki.apache.org/nutch/RunNutchInEclipse>
- [7] [http://wiki.apache.org/nutch/NutchTutorial#A3.3. Using the crawl script](http://wiki.apache.org/nutch/NutchTutorial#A3.3.Using_the_crawl_script)
- [8] Mike Cafarella and Doung Cutting, "Building Nutch: Open Source Search," ACM Queue, April 2004.
- [9] "Modern Information Retrieval", R. Baeza-Yates & B. Ribeiro-Neto, Addison-Wesley, 1999.
- [10] "Managing Gigabytes: Compressing and Indexing Documents and Images", I.H. Witten, A. Moffat, and T.C. Bell .Morgan Kaufmann, San Francisco, second edition, 1999.
- [11] "Modeling the Internet and the Web: Probabilistic Methods and Algorithms", Pierre Baldi, Paolo Frasconi, and Padhraic Smyth, John Wiley & Sons; May 28, 2003.
- [12] "Mining the Web: Analysis of Hypertext and Semi Structured Data", Soumen Chakrabarti, Morgan Kaufmann, 2002

Appendix List

A file list, together with all programs, scripts, and configuration files.

A user manual.

10. User Manual.

Before installing and running Nutch 2.2.1, we need to make sure the following prerequisite should be met.

10.1. Prerequisites

1. Installing, configuring and running successfully Hadoop on your machine. We use Hadoop 1.2.1, a new and stable version of Hadoop. We are able to download it from <http://hadoop.apache.org>.
2. Make sure you install and run successfully Hbase version **0.90.4** on your machine because Nutch 2.2.1 works fine just with this version of Hbase. You can download Hbase0.90.4 from <http://hbase.apache.org/>.
3. Install apache-ant to build java project. You can download ant from <http://ant.apache.org> . We use Apache-ant version 1.8.2 to build Nutch project.
4. Install Eclipse to import Nutch source code to develop the Search Engine. For this project, we use Eclipse Indigo 3.7.2. You are also required to install IvyDE plugin (you can download it here <http://ant.apache.org/ivy/ivyde/download.cgi>) and m2n plugin (you can download it here <http://marketplace.eclipse.org/content/maven-integration-eclipse>) for Eclipse.
5. Install svn subversion plugin for Eclipse or standalone subversion on your local machine, then it allows you to check out source code and check in, update source directly from your local machine to the Nutch repository.

10.2. Checkout and Build Nutch

1. Get the latest source code from SVN using terminal. For Nutch 1.x (ie.trunk) run this:

```
svn co https://svn.apache.org/repos/asf/nutch/branches/2.x
cd 2.x
```

2. At this point you should have decided which data store you want to use. See the [Apache Gora](#) documentation to get more information about it. Here are few of the available options of storage classes:

```
org.apache.gora.hbase.store.HBaseStore
```

```
org.apache.gora.cassandra.store.CassandraStore
org.apache.gora.accumulo.store.AccumuloStore
org.apache.gora.avro.store.AvroStore
org.apache.gora.avro.store.DataFileAvroStore
```

In “*conf/nutch-site.xml*” add the storage class name. eg. Say, you pick HBase as datastore, add this to “*conf/nutch-site.xml*”:

```
<property>
  <name>storage.data.store.class</name>
  <value>org.apache.gora.hbase.store.HBaseStore</value>
  <description>Default class for storing data</description>
</property>
```

3. In *ivy/ivy.xml*: Uncomment the dependency for the data store that you selected. eg. If you plan to use HBase, uncomment this line:

```
<dependency org="org.apache.gora" name="gora-hbase" rev="0.3"
conf="*->default" />
```

4. Set the default datastore in *conf/gora.properties*. eg. For HBase as datastore, put this in *conf/gora.properties*:

```
gora.datastore.default=org.apache.gora.hbase.store.HBaseStore
```

5. Add “*http.agent.name*” and “*http.robots.agents*” with appropriate values in “*conf/nutch-site.xml*”. See *conf/nutch-default.xml* for the description of these properties. Also, add “*plugin.folders*” and set it to {*PATH_TO_NUTCH_CHECKOUT*}/*build/plugins*. eg. If Nutch is present at “/home/user/Desktop/2.x”, set the property to:

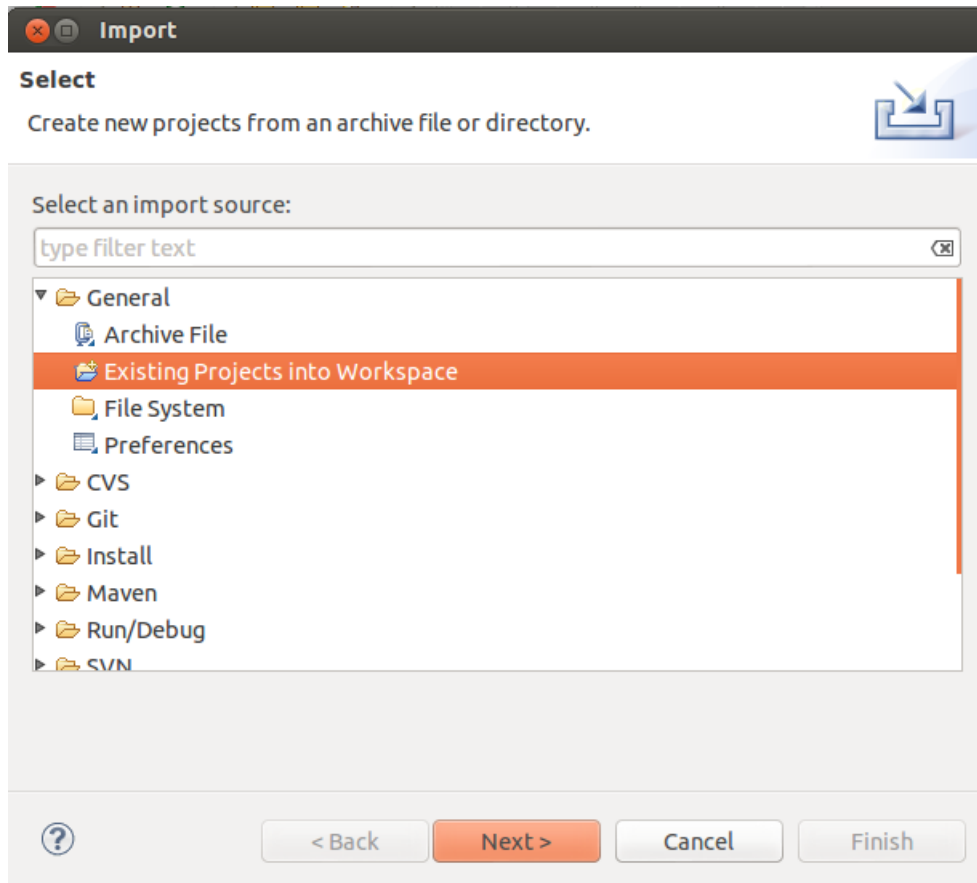
```
<property>
  <name>plugin.folders</name>
  <value>/home/user/Desktop/2.x/build/plugins</value>
</property>
```

6. Run this command:

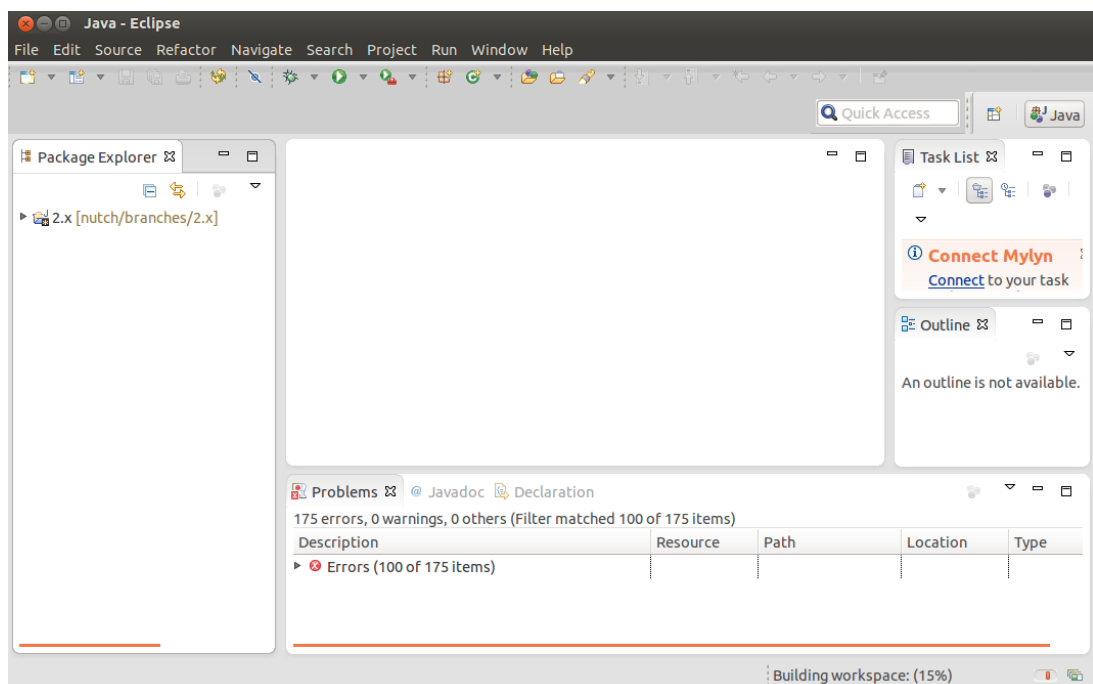
```
ant eclipse
```

10.3. Load project in Eclipse

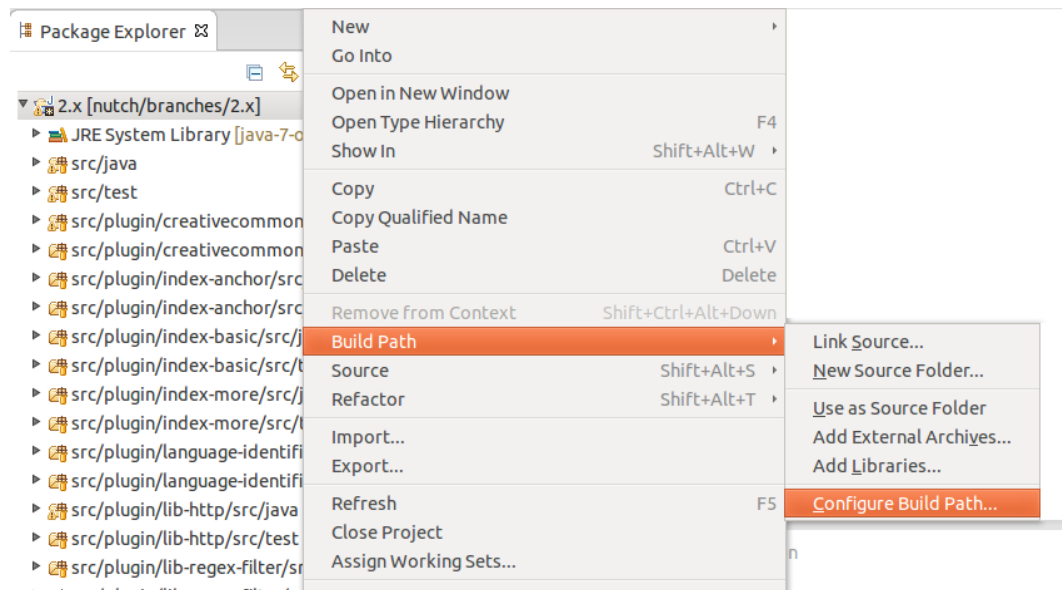
1. In Eclipse, click on “File” -> “Import...”
2. Select “Existing Projects into Workspace”



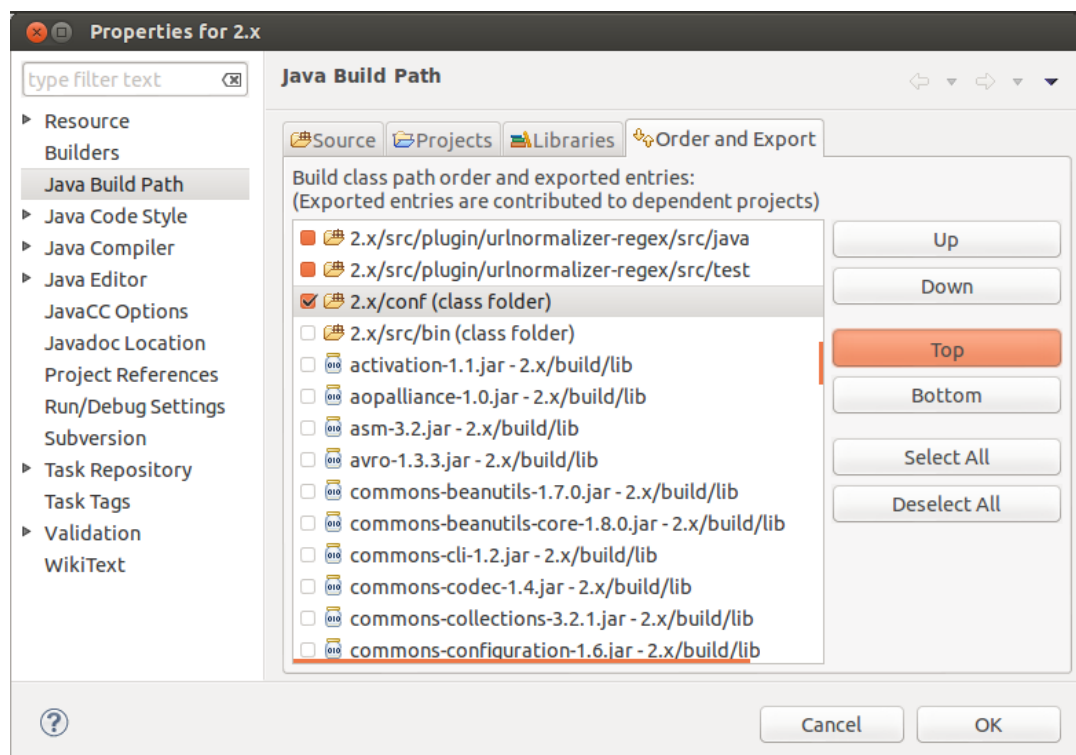
3. In the next window, set the root directory to the location where you took the checkout of nutch 2.x. Click “Finish”.
4. You will now see a new project named 2.x being added in the workspace. Wait for a moment until Eclipse refreshes its SVN cache and builds its workspace. You can see the status at the bottom right corner of Eclipse.



5. In Package Explorer, right click on the project “2.x”, select “Build Path” -> “Configure Build Path”



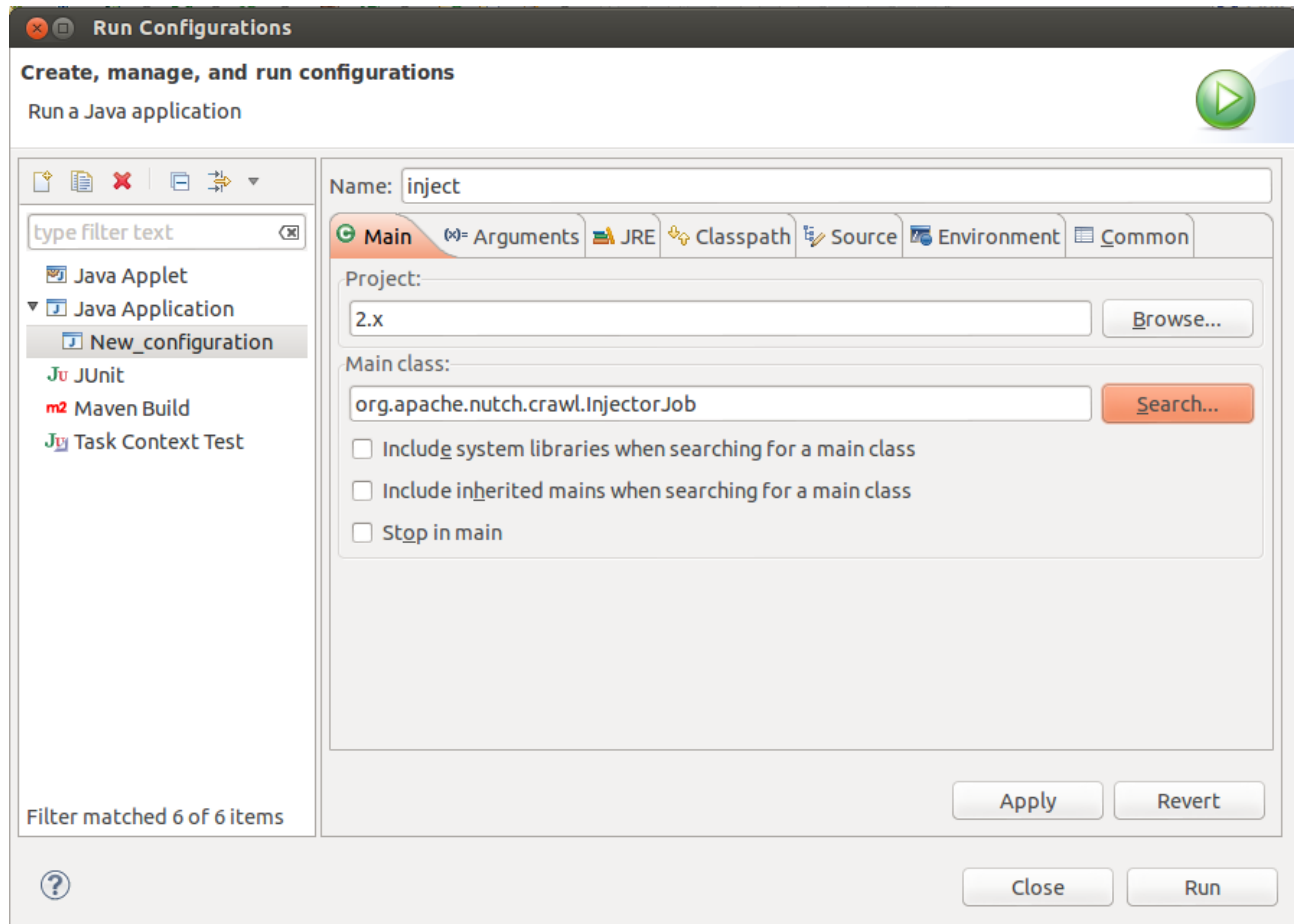
6. In the “Order and Export” tab, scroll down and select “2.x/conf” (or trunk/conf). Click on “Top” button. Sadly, Eclipse will again build the workspace but this time it won’t take much.



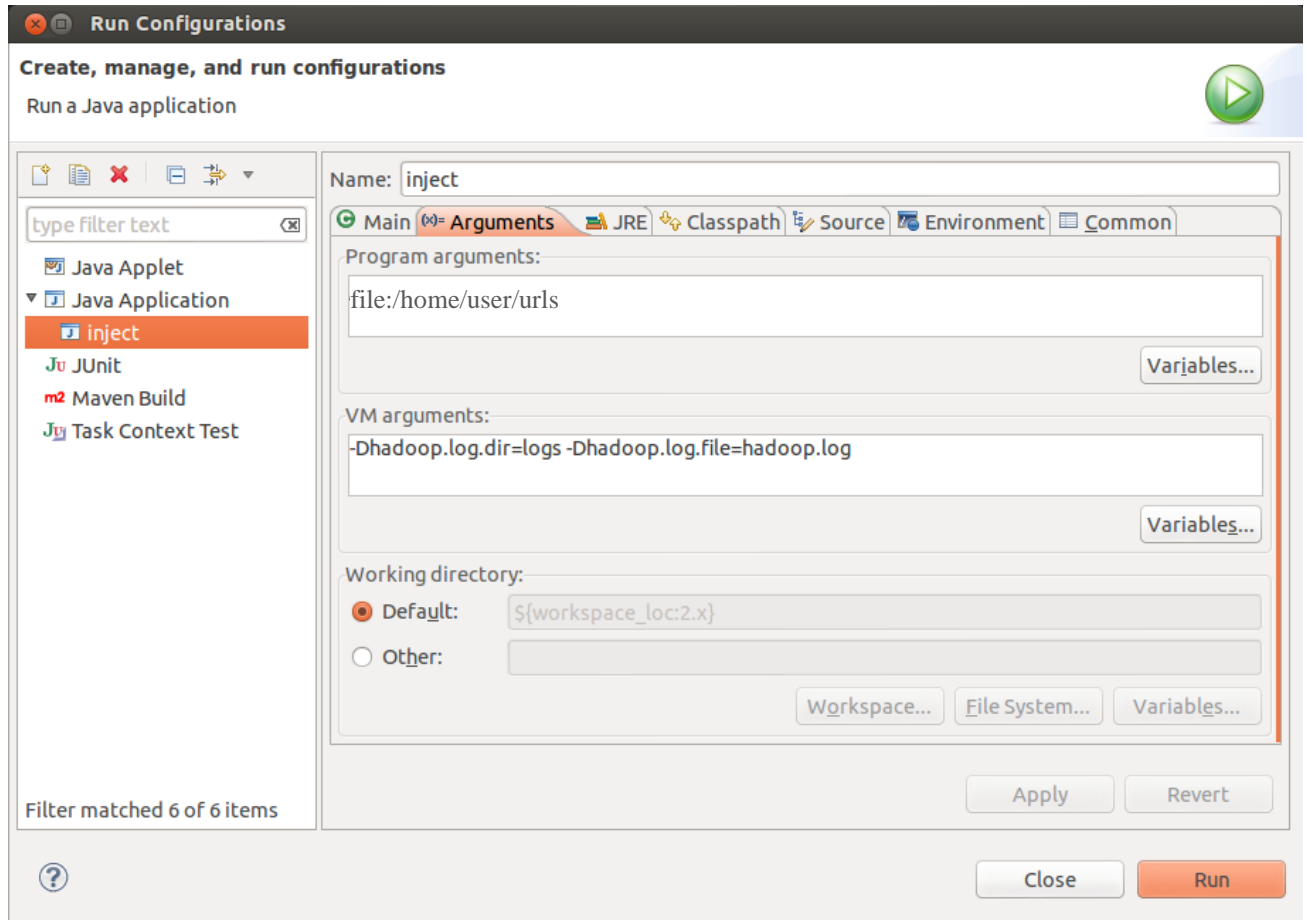
10.4. Create Eclipse launcher

Now, let's get geared to run something. Let's start off with the inject operation. Right click on the project in "Package Explorer" -> select "Run As" -> select "Run Configurations". Create a new configuration. Name it as "inject".

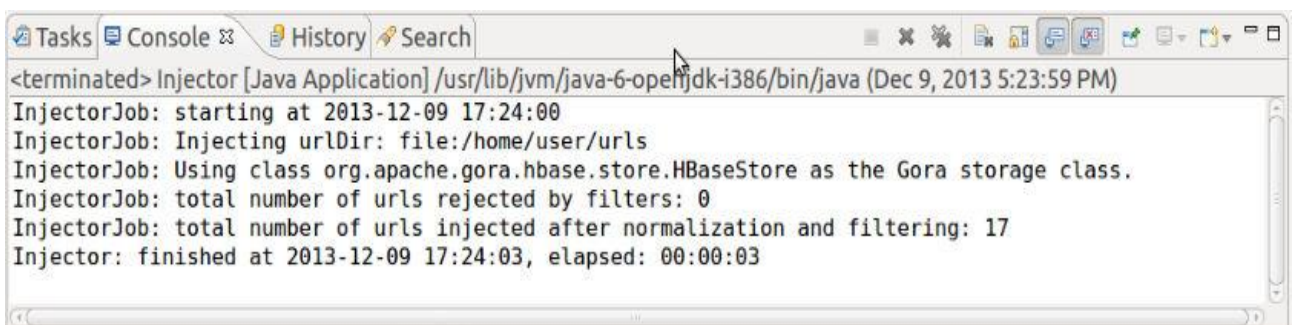
- Set the main class as: `org.apache.nutch.crawl.InjectorJob`, this class we just updated from the last sector.



In the arguments tab, for program arguments, provide the path of the input directory which has seed urls. Set VM Arguments to `"-Dhadoop.log.dir=logs -Dhadoop.log.file=hadoop.log"`



Click "Apply" and then click "Run". If everything was set perfectly, then you should see inject operation progressing on console.



11.File List:

Nutch includes a lot of files, and the same with Hadoop. Therefore, only typical files that are related to this subject are listed here.

Crawler.java
InjectorJob.java
URLFilter.java
URLFilters.java
Crawl-tests.xml
Hbase-site.xml
Nutch-default.xml
Nutch-site.xml
.....

The *apache-nutch-2.2.1-src* is attached in order for verification. Please follow the manual guide described above. Other pre-requirements like Hadoop and Eclipse may be needed, which in case you don't have, download links are provided in the manual.