
ENGLISH WIKIPEDIA ON HADOOP CLUSTER

Final Report



MAY 4, 2016

CS 4624 MULTIMEDIA/HYPertext/INFORMATION ACCESS; DR. FOX
Virginia Tech, Blacksburg VA, 24061

Client: Shivam Maharshi
Student: Steven Stulga

Table of Contents

| | | |
|-----|---|----|
| 1 | Executive Summary..... | 4 |
| 2 | User’s Manual..... | 5 |
| 2.1 | Purpose..... | 5 |
| 2.2 | Requirements..... | 5 |
| 2.3 | Inventory..... | 8 |
| 3 | Developer’s Manual..... | 9 |
| 3.1 | Design..... | 9 |
| 3.2 | Data Path..... | 10 |
| 3.3 | Timeline..... | 11 |
| 3.4 | Implementation..... | 11 |
| 3.5 | Prototype..... | 11 |
| 3.6 | Refinement..... | 18 |
| 3.7 | Testing..... | 19 |
| 3.8 | Future Work..... | 20 |
| 4 | Acknowledgements..... | 21 |
| 5 | Appendix..... | 22 |
| 5.1 | Wikipedia Document Type Definition..... | 22 |
| 5.2 | data-config.xml..... | 23 |
| 5.3 | Avro Schema File..... | 24 |
| 5.4 | Extract and Import Script..... | 25 |
| 6 | References..... | 26 |

Table of Figures

| | | |
|--------------|--|----|
| Figure 2.2.1 | Solr Query For All Documents..... | 6 |
| Figure 2.2.2 | Solr Text Query..... | 6 |
| Figure 3.1 | Wikipedia Cluster Architecture..... | 9 |
| Figure 3.2 | Wikipedia Data Flow..... | 10 |
| Figure 3.5.1 | Hadoop Namenode Dashboard Web Interface..... | 15 |
| Figure 3.5.2 | Hadoop Datanode Dashboard Web Interface..... | 15 |
| Figure 3.5.3 | Solr Dashboard Web Interface..... | 17 |
| Figure 3.5.4 | Solr Index Job Response..... | 18 |

Table of Tables

| | | |
|-------------|--|----|
| Table 2.3.1 | Inventory of wikipedia_hadoop.zip..... | 8 |
| Table 3.3 | Project Timeline..... | 11 |

1 Executive Summary

To develop and test big data software, one thing that is required is a big dataset. The full English Wikipedia dataset would serve well for testing and benchmarking purposes.

Loading this dataset onto a system, such as an Apache Hadoop cluster, and indexing it into Apache Solr, would allow researchers and developers at Virginia Tech to benchmark configurations and big data analytics software. This project is on importing the full English Wikipedia into an Apache Hadoop cluster and indexing it by Apache Solr, so that it can be searched.

A prototype was designed and implemented. A small subset of the Wikipedia data was unpacked and imported into Apache Hadoop's HDFS. The entire Wikipedia Dataset was also downloaded onto a Hadoop Cluster at Virginia Tech. A portion of the dataset was converted from XML to Avro and imported into HDFS on the cluster.

Future work would be to finish unpacking the full dataset and repeat the steps carried out with the prototype system, for all of Wikipedia. Unpacking the remaining data, converting it to Avro, and importing it into HDFS can be done with minimal adjustments to the script written for this job. Continuously run, this job would take an estimated 30 hours to complete.

2 User's Manual

2.1 Purpose

The purpose of this project is to download the complete English Wikipedia, specifically the text articles, convert them to a more useful and compressed data format, and upload them to a cluster where searches and analytics can be supported. The data of interest is only text, and will include the history of revisions on all the Wikipedia articles. Upon completion, Wikipedia will serve as a valuable offline dataset for testing analytics against Apache Hadoop and Apache Solr.

The client, Shivam Maharshi, and many different research groups at Virginia tech, would then have access to this dataset and be able to run analytics on English Wikipedia articles, using Hadoop and Solr, which would otherwise be impossible when accessing the online version. The Wikipedia index with Solr will be a useful benchmark when compared to research and experimental indexing or searching software. Further, local research projects could run large numbers of searches with much better performance than if using remote services.

2.2 Requirements

There are three main aspects of the project. These aspects are the English Wikipedia data, Hadoop Distributed Filesystem (HDFS) on the Hadoop Cluster, and the Apache Solr index.

2.2.1 English Wikipedia Data

The Wikimedia Foundation (WMF) dumps the entire English version of all Wikipedia articles about once a month. Several versions of the dump are hosted. One of the dumps contains the current version of all articles, without a history of revisions. Other versions include the complete articles, along with the complete version history and discussions, which is the dump that will be imported into the Hadoop Cluster.

This dump is called 'pages-meta-history' by the WMF. The format of the data uses the Extensible Markup Language (XML). The data is offered with two different compression options: 7zip or bzip2. The compression that will be chosen is 7zip, because the files that must be downloaded are generally smaller than their bzip2 versions. The size of the full compressed dataset is about 107 gigabytes. The size of the full raw XML dataset is approximately 1.8 terabytes. The Wikipedia Document Type Definition (DTD) of the format is found in the Appendix.

The reason that the WMF distributed its dump as XML rather than HTML is because the HTML articles that are familiar to Wikipedia users consist of dynamic content that is

pieced together on the server side before being sent to the users. XML is a more static format that can be produced easily as a onetime dump and yet reflects all of the data in the text of the English Wikipedia. XML format is also relatively easy to work with.

For the purposes of this project, the XML data will be converted to the Apache Avro data format. Apache Avro is a binary data serialization format, which uses schemas to define the elements of the data. Schemas are defined in JavaScript Object Notation (JSON). Apache releases libraries for many languages, such as Java or Python, which allow programs to easily work with Avro. Avro is often more compressed than formats, such as XML, and facilitates distributed processing on a Hadoop cluster, as well as large block sizes that are needed when handling files in HDFS.

There are many external open-source scripts that can be found online for converting Wikipedia's XML data dumps into Avro format. One such script, "xml-avro", has been chosen for this project. The source for this script is found on Github and the program is implemented in Java.

Once the script is executed on the Wikipedia data and it is confirmed that the data is correctly converted from XML to Avro format, then it must be loaded into the Hadoop cluster on HDFS.

2.2.2 HDFS on the Hadoop Cluster

Apache Hadoop is an implementation of a Map/Reduce system for distributed computing across clusters of compute nodes. Map/Reduce is a framework for scheduling and performing operations on a set of data spread across a cluster, and then compiling the results together. Hadoop uses HDFS to split the data among the nodes in a cluster. This is useful for data where easily parallelized operations can be performed, or for datasets that are very large. The Wikipedia data must be stored across multiple disks and accessed through Hadoop's HDFS interface, for these reasons. Once the data is imported to HDFS on the cluster, it is ready to be indexed by Apache Solr.

2.2.3 Apache Solr Index

Apache Solr is a search engine platform designed for high volume and advanced text searches. It works by utilizing Apache Lucene, a library for information retrieval. Apache Solr wraps Lucene into an enterprise application, and adds functionality to it, such as more advanced searches, as well as faceting and other analysis.

The Wikipedia dataset ultimately must be indexed by Apache Solr so that quick searches on keywords will return Wikipedia articles. To index the data, a Solr schema will be created, which tells Solr how to create indexes from each document, and what parts of the documents the results of a query should provide.

Figure 2.2.1 – Solr Query For All Documents

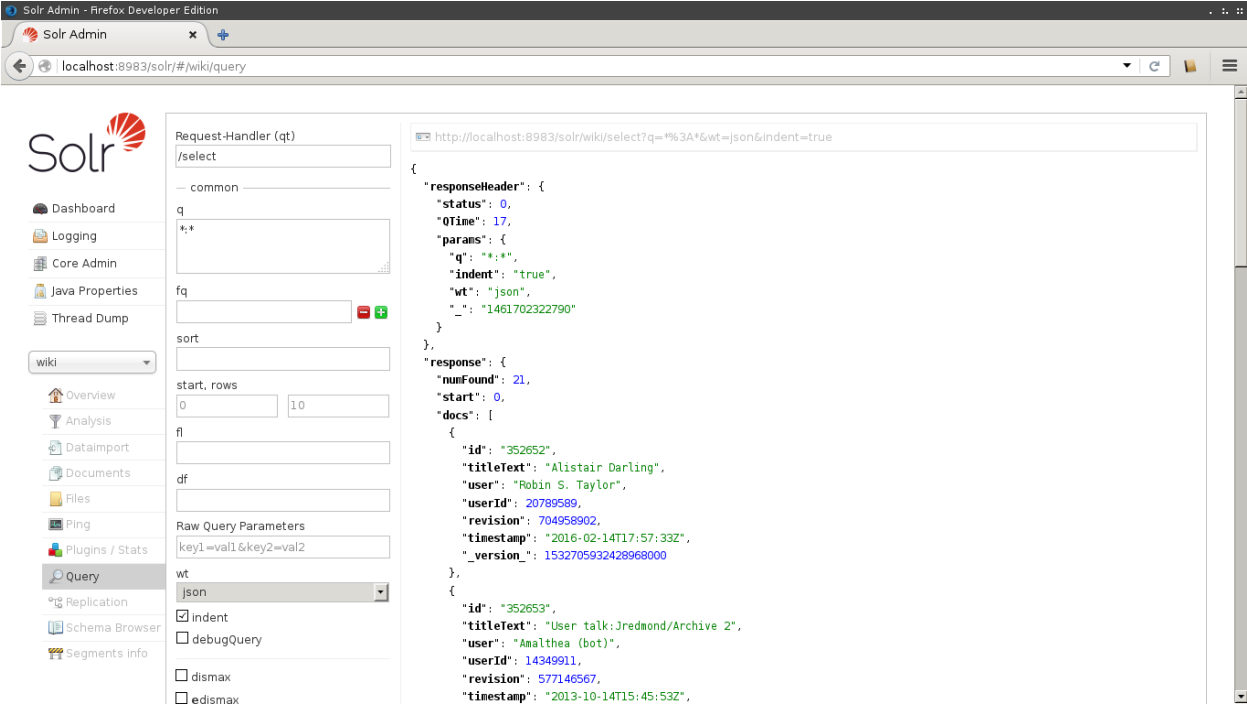
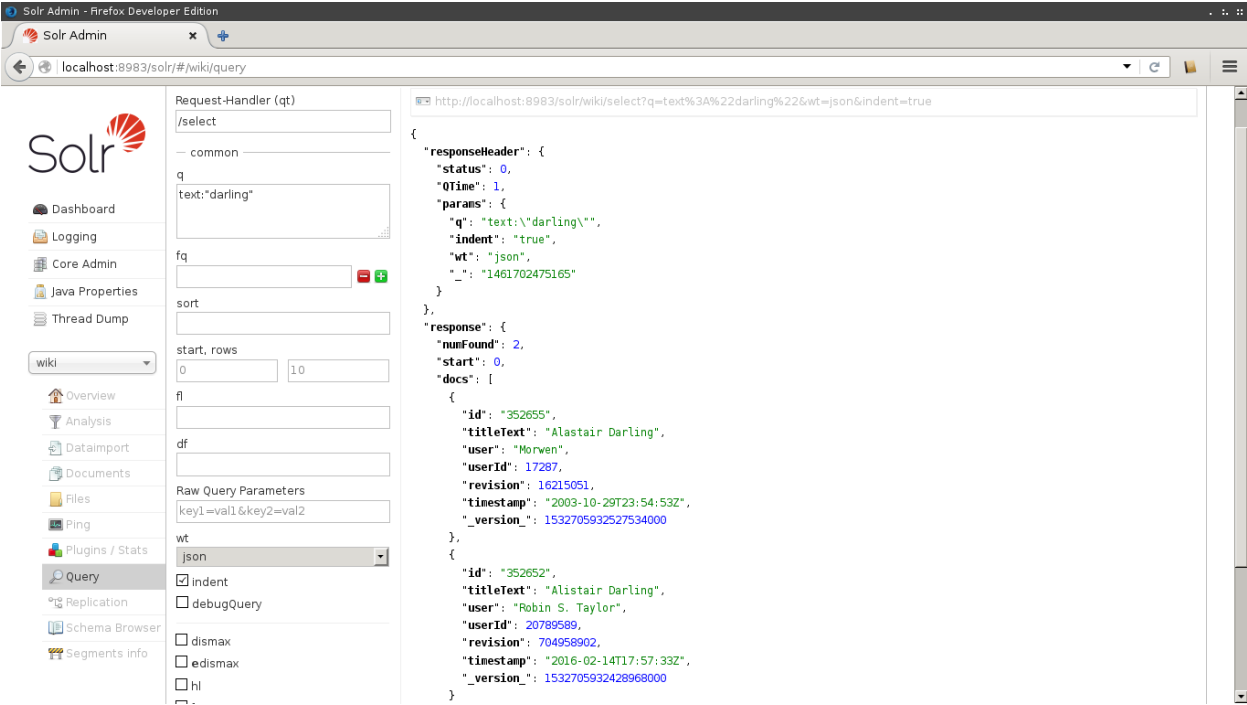


Figure 2.2.2 – Solr Text Query



2.3 Inventory

The included, "wikipedia_hadoop.zip", archive contains the prototype, as well as some scripts used on the head node. The inventory of this archive is as follows.

Table 2.3.1 – Inventory of Wikipedia_hadoop.zip

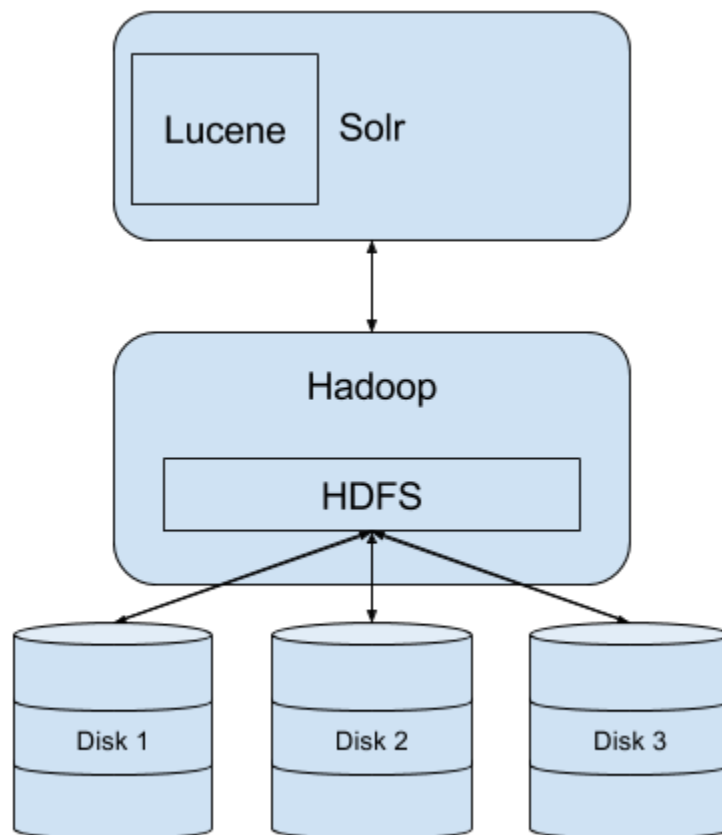
| | |
|-----------------------------|---|
| 7z/ | 7z archive of wikipedia XML dump segment |
| XML/ | XML wikipedia dump segment |
| Avro/ | Avro wikipedia dump segment |
| p7ip_15.14.1/ | p7zip utility used to extract 7z archives |
| hadoop-2.6.4/ | Apache Hadoop installation |
| solr-5.3.1/ | Apache Solr installation |
| solr-5.3.1/server/solr/wiki | Apache Solr core used to index wikipedia XML |
| xml-avro/ | xml-avro conversion tool |
| dtd2xsd.pl | Perl tool to convert DTD file to XSD |
| wikipedia.dtd | DTD file for wikipedia XML data |
| enwiki.xsd | XSD generated from dtd2xsd.pl tool |
| export-0.10.xsd | wikipedia data XSD file |
| xml.avsc | Avro Schema created by xml-avro conversion tool |
| links.txt | Contains URL for every archive in wikipedia dataset |
| Unpack_import.sh | Script for extracting and importing XML files into HDFS |

3 Developer's Manual

3.1 Design

The system architecture for this project is very standard and similar to other systems using Apache Hadoop and Apache Solr. A diagram of the design is in Figure 3.1. At the highest level of the system is Solr, which will receive queries and deliver responses. Solr uses the Lucene library, to help index the data. It must communicate with Hadoop, which uses HDFS to store data across multiple disks. The disks are where the Wikipedia data lies. This architecture was chosen because of the size of the dataset; it is beneficial to store it in a distributed filesystem like HDFS. To index and search the data with Apache Solr, it is necessary to interface Apache Solr with Apache Hadoop.

Figure 3.1 - Wikipedia Cluster Architecture



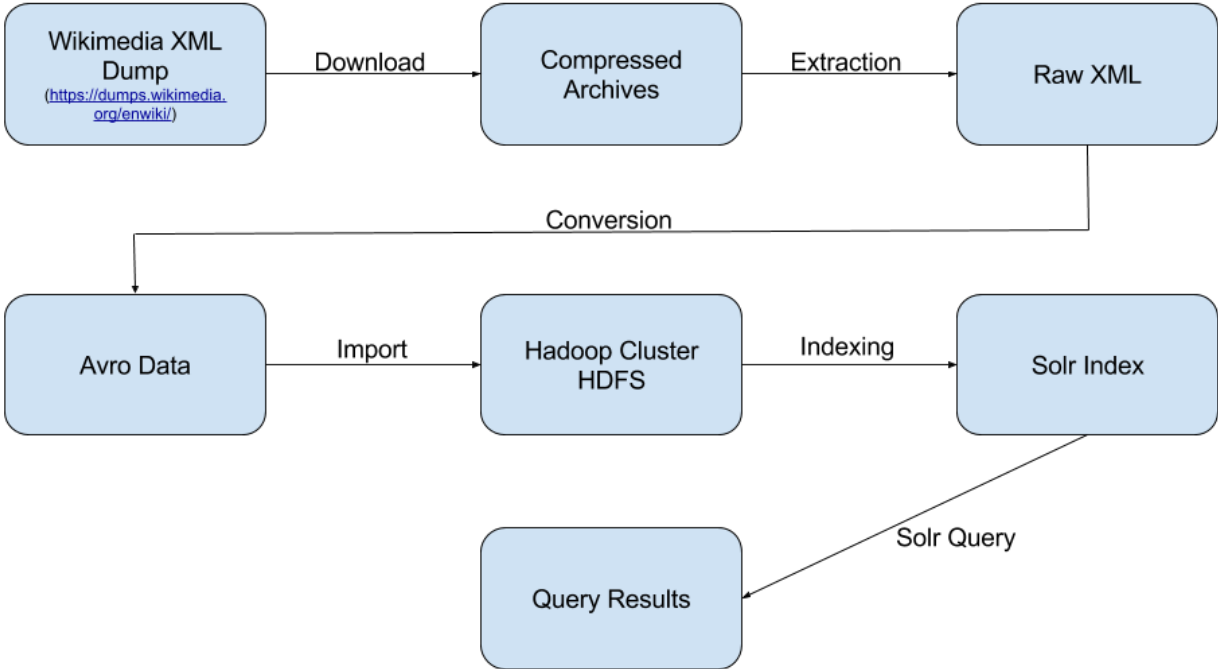
Particularly important is the Apache Solr schema. This schema must be designed so that Apache Solr indexes all of the relevant information that could be used in queries and to match documents. The Wikipedia DTD in the Appendix describes all of the fields in each document of the dataset.

There are existing Apache Solr schemas on the Internet that are used to index XML formatted Wikipedia data. One such schema is listed as an example on the official Apache Solr Wiki. This example schema will be adapted so that queries can be made on all fields, including the document and revision content.

3.2 Data Path

The data path for the project is outlined in Figure 3.2. It involves downloading the Wikipedia data dump from Wikimedia.org. The dump used was from March 5, 2016, and can be found at <https://dumps.wikimedia.org/enwiki/20160305/>. The data used is the “All pages with complete edit history” collection. This dump is compressed with 7z format, so it must be extracted into raw XML. After it is extracted, the data gets converted from XML to Apache Avro. This Avro data is then imported into the HDFS on the Hadoop cluster. Once the data is imported, it can be indexed by Solr. Queries made to Solr will return query responses, with results in JSON format.

Figure 3.2 - Wikipedia Data Flow



3.3 Timeline

Table 3.3 – Project Timeline

| | |
|----------|---|
| March 14 | Apache Hadoop and HDFS are installed on prototype machine; Portion of Wikipedia data is downloaded for prototype |
| March 24 | Full Wikipedia dataset is downloaded onto the head node of the cluster |
| March 31 | Testing for converting XML data to Avro |
| April 7 | Testing extracting full Wikipedia dataset on head node Apache Solr is installed on prototype machine; configurations and testing Apache Solr |
| April 11 | Apache Solr prototype configured to index Wikipedia XML data |
| April 27 | Imported Avro data into Hadoop Cluster |
| Future | Index Avro data with Apache Solr on Hadoop Cluster |

3.4 Implementation

This design will be carried out on the Virginia Tech Hadoop cluster, hadoop.dlib.vt.edu.

The compressed XML data was downloaded to the head node with a small script that utilized the Linux utility “wget”. The download was automated and monitored, to make sure that each of the pieces of the XML dump were completely situated on the machine. The total download time took little over 24 hours to complete.

Next the source code for the Linux utility “p7zip” was cloned from the Internet and built. This utility is an open source command-line port of 7zip for Linux, so that the downloaded 7z archives were able to be decompressed into the full raw XML. There was not enough space on the head node to decompress all of the 7z files in the dataset at once, so this process had to be done piecemeal, separately for each file.

The “xml-avro” Github project was cloned onto the head node and compiled. This script caused memory errors when trying to handle the large XML files, so it was tested on the head node, as well as a machine with 16GB of RAM, and different JVM heap configurations to try to convert all of the XML to Avro. For files 8GB and larger, the memory error could not be solved.

The future steps are to convert and import the remaining XML data. Finally, indexing is needed of the Avro data with Apache Solr from HDFS.

3.5 Prototype

The prototype for this project was the main focus and most important part. The steps taken and lessons learned in the prototype will be repeated and used on the cluster

with the full Wikipedia dataset. Once these steps are created and implemented, the creation of the full system will be eased and quickened.

This prototype was designed and implemented on a local machine, specifically a Dell laptop. This prototype fleshed out each of the steps taken on the head node, before performing them on the local machine.

On the prototype machine, Apache Hadoop and Apache Solr instances were installed and configured. The prototype system was implemented only working with a small section of the English Wikipedia data. The smallest 7z archive offered by Wikimedia was selected to be used. Specifically, the archive file “enwiki-20160305-pages-meta-history4.xml-p000352652p000352689.7z” from the dump occurring on March 3, 2016. This archive is 364KB large. The command-line used to download this file was:

```
~ $ mkdir 7z
~ $ cd 7z
~/7z $ wget https://dumps.wikimedia.org/enwiki/20160305/enwiki-20160305-  
pages-meta-history4.xml-p000352652p000352689.7z
~/7z $ cd ..
```

The open-source command-line port of 7-Zip, p7zip, was needed to extract the 7z archive. The latest version of the source code for this tool was downloaded, and built into binaries. The command lines necessary for the binary 7za was run with the “x” switch in order to extract the full XML file. The exact command-lines used for this job were:

```
~ $ wget https://downloads.sourceforge.net/project/p7zip/p7zip/15  
.14%20.1/p7zip\_15.14.1\_src\_all.tar.bz2
~ $ tar -xf p7zip_15.14.1_src_all.tar.bz2
~ $ cd p7zip_15.14.1
~/p7zip_15.14.1 $ make all
~/p7zip_15.14.1 $ cd ..
~ $ mkdir XML
~ $ p7zip_15.14.1/bin/7za x 7z/enwiki-20160305-pages-meta-  
history4.xml-p000352652p000352689.7z -oXML/
```

The external tool xml-avro was then downloaded from Github and built. The tool, xml-avro can be run with an XML file and an XSD file as inputs. XSD is an XML Schema Definition file, which defines the documents, elements, and fields inside of an XML file. This is explicitly states the schema of the data for when converting to Avro. The tool would then output an Avro Schema File, and the actual Avro data. The tool can also be run without the XSD file, to generate an Avro file using a more generic schema. Two different XSD files for the Wikipedia XML were trailed with the xml-avro converter. The

first was an XSD file hosted from mediawiki.org, which is referred to by the Wikipedia XML. The second XSD file trialed was generated from Wikipedia’s DTD. This DTD can be found in the Appendix, 5.1 Wikipedia Document Type Definition, and was named “enwiki.dtd”. To generate an XSD from the DTD, an external Perl script from w3.org was used. The steps trailed are:

```
~ $ mkdir Avro
~ $ wget https://www.mediawiki.org/xml/export-0.10.xsd
~ $ wget https://www.w3.org/2000/04/schema\_hack/dtd2xsd.pl
~ $ perl dtd2xsd.pl -prefix xs -ns
http://www.w3.org/2001/XMLSchema\_enwiki.dtd > enwiki.xsd
~ $ git clone https://github.com/elodina/xml-avro.git
~ $ cd xml-avro
~/xml-avro $ mvn package
~/xml-avro $ java -jar target/xml-avro-0.0.1-SNAPSHOT.jar export-0.10.xsd
XML/enwiki-20160305-pages-meta-history4.xml-
p000352652p000352689 enwiki.asvc Avro/ enwiki-20160305-pages-
meta-history4.avro-p000352652p000352689
~/xml-avro $ java -jar target/xml-avro-0.0.1-SNAPSHOT.jar enwiki.xsd
XML/enwiki-20160305-pages-meta-history4.xml-
p000352652p000352689 enwiki.asvc Avro/ enwiki-20160305-pages-
meta-history4.avro-p000352652p000352689
~/xml-avro $ java -jar target/xml-avro-0.0.1-SNAPSHOT.jar
ly.stealth.xmlavro.simple.Converter avro XML/enwiki-20160305-
pages-meta-history4.xml-p000352652p000352689 Avro/ enwiki-
20160305-pages-meta-history4.avro-p000352652p000352689
~/xml-avro $ cd ..
```

However, neither an XSD file generated from Wikipedia’s DTD, nor the XSD file hosted, were compatible with the xml-avro conversion tool. The XSD generated from Wikipedia’s DTD file required the proper XML Schema Definition namespace to be manually inserted for each element, but there were attributes that were missing the required “name” and “ref” members. The errors output for the official XSD showed that the schema is using the attribute ‘xml:space=“preserve”’. This attribute gives information about how to handle whitespace of the text in that element. This attribute is not supported by the xml-avro script, and therefore the XSD could not be used to convert to Avro. Fortunately, using a generic schema was successful for converting the XML into Avro. The schema is found in the Appendix, 5.3 Avro Schema File. This Avro schema is more generic and includes less metadata than the XML, however the information from each document is still included in the Avro records. The other uncompressed XML documents in the Wikipedia dataset reach file sizes from 8GB to over 60GB. These very large XML files create complications when trying to convert from XML to Avro, because of their massive size and memory resource requirements during the conversion. Converting files of this size presented a “java.lang.OutOfMemoryError: Java heap space” exception. To attempt to remedy this exception, the maximum size of

the JVM's heap was increased using the "-Xmx" flag with different heap sizes. Running this conversion on several different machines did not solve the issue.

The next step was to download and unpack Apache Hadoop:

```
~ $ wget http://apache.mirrors.pair.com/hadoop/common/hadoop-2.6.4/hadoop-2.6.4.tar.gz
~ $ tar -xf hadoop-2.6.4.tar.gz
~ $ cd hadoop-2.6.4
~/hadoop-2.6.4 $ bin/hadoop
```

To configure Hadoop and prepare the environment, some steps needed to be taken. The first was to edit the file "~/hadoop-2.6.4/etc/hadoop/hadoop-env.sh" and update the line "export JAVA_HOME=\$JAVA_HOME" to accurately reflect the correct location of the Java installation on the machine. Next, was to edit "~/hadoop-2.6.4/etc/hadoop/core-site.xml" to reflect:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

and "~/hadoop-2.6.4/etc/hadoop/hdfs-site.xml" to reflect:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

Apache Hadoop requires the "sshd" service to run and be configured correctly so that it can communicate through ssh to the localhost without a login prompt. To configure "sshd", the commands were run:

```
~ $ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
~ $ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
~ $ chmod 0600 ~/.ssh/authorized_keys
```

Now, Apache Hadoop can be started and information loaded into the HDFS. Hadoop starts up a web interface, that can be accessed at <http://localhost:50070> following the commands:

```
~/hadoop-2.6.4 $ bin/hdfs namenode -format
~/hadoop-2.6.4 $ sbin/start-dfs.sh
~/hadoop-2.6.4 $ bin/hdfs dfs -mkdir /solr
~/hadoop-2.6.4 $ bin/hdfs dfs -put ../Avro/enwiki-20160305-pages-meta-history4.avro-p000352652p000352689 /solr/
~/hadoop-2.6.4 $ bin/hdfs dfs -ls /solr
```

Figure 3.5.1 – Hadoop Namenode Dashboard Web Interface

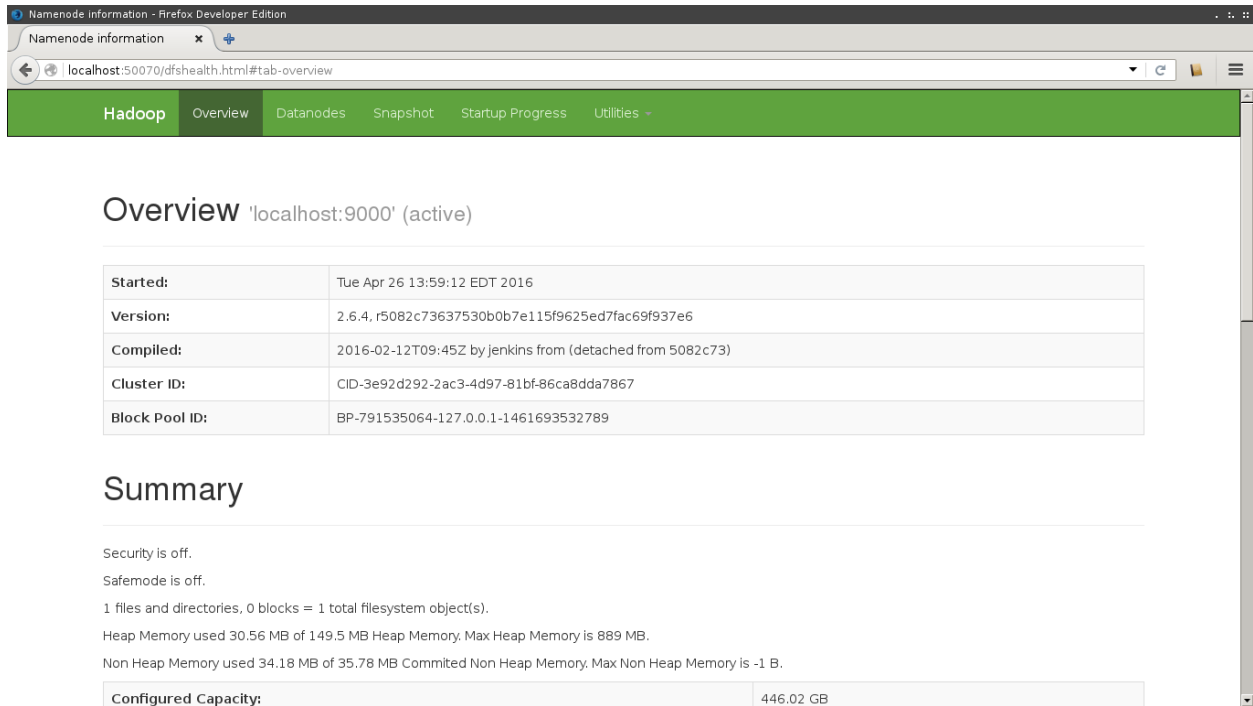
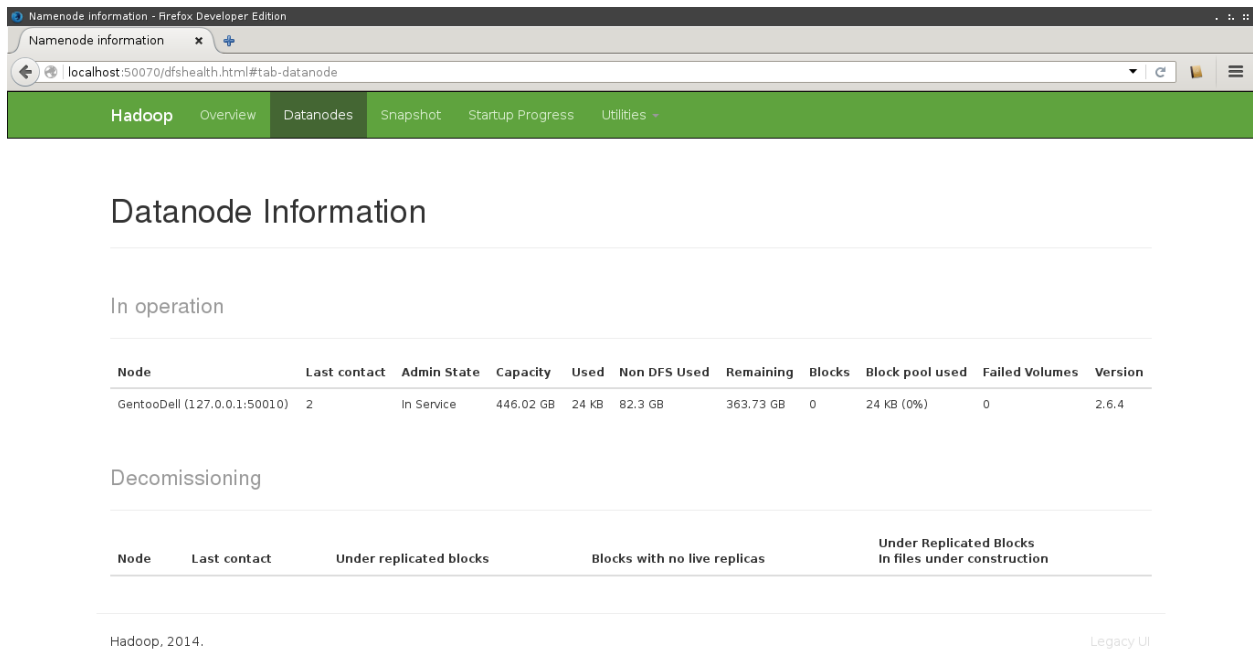


Figure 3.5.2 – Hadoop Datanode Dashboard Web Interface



For the prototype, I configured Apache Solr to index an XML file from the Wikipedia dataset as a proof of concept. Creating the schema file used by Apache Solr to index the data was done with the following commands:

```
~ $ wget
    http://mirrors.gigenet.com/apache/lucene/solr/5.3.1/solr-
    5.3.1.tgz
~ $ tar -xf solr-5.3.1.tgz
~ $ cd solr-5.3.1
~/solr-5.3.1 $ bin/solr start
~/solr-5.3.1 $ bin/solr create -c wiki
~/solr-5.3.1 $ cp -r
    server/solr/configsets/data_driven_schema_configs/conf
    server/solr/wiki/
~/solr-5.3.1 $ mv server/solr/wiki/conf/managed-schema
    server/solr/wiki/conf/schema.xml
```

Next some configurations were modified.

The first file worked with was “server/solr/wiki/conf/solrconfig.xml”.

The lines added were:

```
<lib dir="${solr.install.dir:../../../../..}/dist/" regex="solr-
dataimporthandler-.*\.jar" />

<schemaFactory class="ClassicIndexSchemaFactory"/>

<requestHandler name="/dihupdate"
class="org.apache.solr.handler.dataimport.DataImportHandler" startup="lazy">

    <lst name="defaults">

        <str name="config">data-config.xml</str>

    </lst>

</requestHandler>
```

The lines deleted were:

```
<schemaFactory class="ManagedIndexSchemaFactory">

    <bool name="mutable">true</bool>

    <str name="managedSchemaResourceName">managed-schema</str>

</schemaFactory>
```

We then need to create a file “server/solr/wiki/conf/data-config.xml”, which contents can be found in the Appendix, 5.2 data-confix.xml. After creating that file, we must modify “server/solr/wiki/conf/schema.xml” to include only the following field elements:

```
<field name="_version_" type="long" indexed="true" stored="true"/>
<field name="id" type="string" indexed="true" stored="true"
required="true"/>
```



```

<field name="title"      type="string"  indexed="true" stored="false"/>
<field name="revision"  type="int"    indexed="true" stored="true"/>
<field name="user"      type="string" indexed="true" stored="true"/>
<field name="userId"    type="int"    indexed="true" stored="true"/>
<field name="text"      type="text_en" indexed="true" stored="false"/>
<field name="timestamp" type="date"    indexed="true" stored="true"/>
<field name="titleText" type="text_en" indexed="true" stored="true"/>
<copyField source="title" dest="titleText"/>

```

Once these modifications are complete, Solr must be restarted by running:

```

~/solr-5.3.1 $ bin/solr stop
~/solr-5.3.1 $ bin/solr start

```

Visiting the URL <http://localhost:8983/solr/wiki/diupdate?command=full-import> will kick off the job to index the documents. Once this has been done, the index can be queried in the Solr web dashboard, <http://localhost:8983/solr/#/wiki/query>.

Figure 3.5.3 – Solr Dashboard Web Interface

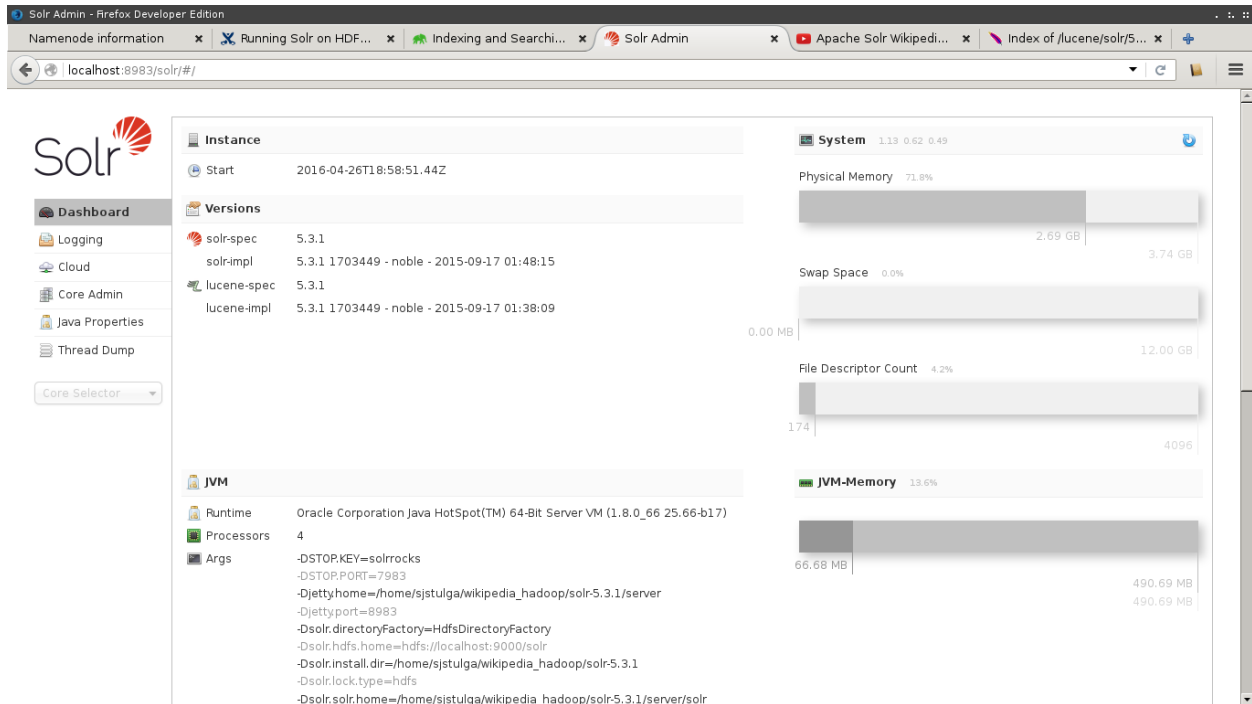
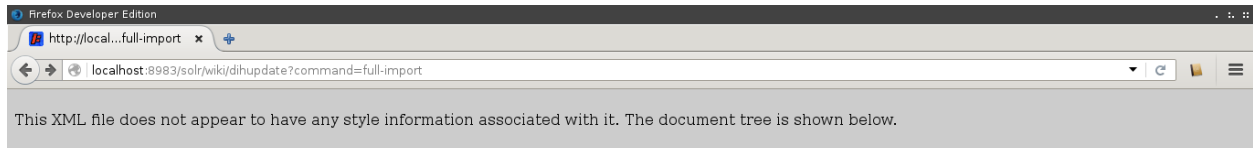


Figure 3.5.4 – Solr Index Job Response



```
-<response>
- <lst name="responseHeader">
  <int name="status">0</int>
  <int name="QTime">8</int>
</lst>
- <lst name="initArgs">
  - <lst name="defaults">
    <str name="config">data-config.xml</str>
  </lst>
  <str name="command">full-import</str>
  <str name="status">idle</str>
  <str name="importResponse"/>
- <lst name="statusMessages">
  <str name="Total Requests made to DataSource">0</str>
  <str name="Total Rows Fetched">28</str>
  <str name="Total Documents Processed">21</str>
  <str name="Total Documents Skipped">7</str>
  <str name="Full Dump Started">2016-04-26 20:24:39</str>
- <str name="">
  Indexing completed. Added/Updated: 21 documents. Deleted 0 documents.
</str>
  <str name="Committed">2016-04-26 20:24:41</str>
  <str name="Time taken">0:0:2.105</str>
</lst>
</response>
```

3.6 Refinement

Refining the project involves executing the instructions above on the cluster with the full English Wikipedia. So far, the steps that have been taken have been to build the external utilities p7zip and xml-avro that are needed from source on the cluster.

The full English Wikipedia dataset was downloaded onto the cluster. This was done by compiling a text document with the link to each of the archives in the dataset, and running wget to automate the download. The size of the entire compressed dataset is about 107GB. Uncompressed, the dataset is estimated to be around 10TB.

There were many lessons learned in this project, and future refinements to include. These cover areas of the prototype, such as solving the problems with converting XML to Avro, specifically the memory constraints and creating a compatible XML Schema Definition. Using Avro would give a minor improvement for flexibility of the data to be used with other programs that might not natively accept XML, as well as smaller file sizes.

Another area of refinement is to configure Solr to access data directly from HDFS, rather than from a local filesystem. Research has been done about this method, and there are solutions offered by Apache or Cloudera that allow Solr to directly interface and read from HDFS. Apache offers tools such as “morphlines” that extend Solr’s capabilities to read from any type of data and any data source. Cloudera is an enterprise platform that

packages Apache Hadoop and Apache Solr together, as well as several external utilities, that allow easier configuration and use.

Finally, all of the steps listed for the prototype need to be completed on the cluster, with the full dataset. This should be simple and straightforward, by following the instructions, however it will be a very lengthy process due to the size of the data and the processing time.

3.7 Testing

The prototype was tested in stages, as each step was completed. The first step, downloading the smallest 7z archive from the Wikipedia dataset was tested by generating an md5 checksum from the downloaded information and comparing it to a value provided by the Wikimedia Foundation, for verifying integrity of the download.

After extracting the XML file from the 7z archive, the file was opened and examined to make sure that the contents were correct and expected. This involved checking the XML entries, fields, and the namespace used to match the items specified in Wikipedia's DTD.

The majority of the trials involving converting the XML to Avro could not be tested, because of the challenges and errors faced. When converting the XML to Avro using a generic schema, however, this file was checked to make sure that the contents were as expected and correct Avro.

When the data was imported into HDFS, this was verified to work, using the Hadoop Web Dashboard to check that the used memory of HDFS had increased. The contents of the HDFS were also listed, to display that the file was in fact imported into HDFS and had the correct file size.

Configuring Solr was a great learning process, and involved testing each step of the installation and configuration to verify that it was being done correctly. This included viewing results from the Solr Web Interface to make sure that the Solr core that would index the data had been created successfully. Whenever there was an error in the configuration, the web dashboard was very helpful, displaying the error and information to track it down.

Once the core was properly configured, and the job was run to index the XML file from the local filesystem, this was verified to be completed successfully by querying all of the documents in the index, and seeing that the index had been populated. Several text queries were made, to retrieve documents that contained known text strings. An example was the query "text:darling", which returned documents from the data titled "Alistair Darling".

3.8 Future Work

The remaining work on this project would be to test further ways to convert the remaining XML files into Avro. One such way is by using a Map/Reduce job that makes use of `XmlInputFormat` and `AvroKeyValueOutputFormat`. However, this would require moving the XML files onto HDFS on the cluster. Once the XML has successfully been converted to Avro, all of this Avro data must be stored in HDFS. The next steps are just to create a Solr core with a schema suitable for indexing the Avro data, and then to index this data. After Solr has indexed the data, the project will be completed and queries for Wikipedia articles can be made through Solr.

4 Acknowledgements

Acknowledgements for this project go to the client, Shivam Maharshi, for presenting the project and explaining details about it. Mr. Maharshi was able to point the design and implementation in the right direction, so work could quickly be started. He also laid out the steps for creating the prototype for this system.

Acknowledgments also go to Sunshin Lee, who provided access to the remote machine and answered questions about execution on the machine.

5 Appendix

5.1 Wikipedia Document Type Definition

```
<!ELEMENT mediawiki (siteinfo,page*)>
<!-- version contains the version number of the format (currently 0.3) -->
<!ATTLIST mediawiki
  version CDATA #REQUIRED
  xmlns CDATA #FIXED "http://www.mediawiki.org/xml/export-0.3/"
  xmlns:xsi CDATA #FIXED "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation CDATA #FIXED
    "http://www.mediawiki.org/xml/export-0.3/
http://www.mediawiki.org/xml/export-0.3.xsd"
  xml:lang CDATA #IMPLIED
>
<!ELEMENT siteinfo (sitename,base,generator,case,namespace)>
<!ELEMENT sitename (#PCDATA)>      <!-- name of the wiki -->
<!ELEMENT base (#PCDATA)>          <!-- url of the main page -->
<!ELEMENT generator (#PCDATA)>     <!-- MediaWiki version string -->
<!ELEMENT case (#PCDATA)>          <!-- how cases in page names are handled
-->
  <!-- possible values: 'first-letter' | 'case-sensitive'
                           'case-insensitive' option is reserved for future -
->
<!ELEMENT namespace (#PCDATA)>     <!-- contains namespace prefix -->
<!ATTLIST namespace key CDATA #REQUIRED> <!-- internal namespace number -
->
<!ELEMENT page (title,id?,restrictions?,(revision|upload)*)>
  <!ELEMENT title (#PCDATA)>        <!-- Title with namespace prefix -->
  <!ELEMENT id (#PCDATA)>
  <!ELEMENT restrictions (#PCDATA)> <!-- optional page restrictions -->
<!ELEMENT revision (id?,timestamp,contributor,minor?,comment?,text)>
  <!ELEMENT timestamp (#PCDATA)>    <!-- according to ISO8601 -->
  <!ELEMENT minor EMPTY>            <!-- minor flag -->
  <!ELEMENT comment (#PCDATA)>
  <!ELEMENT text (#PCDATA)>         <!-- Wikisyntax -->
  <!ATTLIST text xml:space CDATA #FIXED "preserve">
<!ELEMENT contributor ((username,id) | ip)>
  <!ELEMENT username (#PCDATA)>
  <!ELEMENT ip (#PCDATA)>
<!ELEMENT upload (timestamp,contributor,comment?,filename,src,size)>
  <!ELEMENT filename (#PCDATA)>
  <!ELEMENT src (#PCDATA)>
<!ELEMENT size (#PCDATA)>
```

5.2 data-config.xml

```
<dataConfig>
  <dataSource type="FileDataSource" encoding="UTF-8" />
  <document>
    <entity name="page"
      processor="XPathEntityProcessor"
      stream="true"
      forEach="/mediawiki/page/"
      url"path/to/xml/file/to/index.xml"
      transformer="RegexTransformer,DateFormatTransformer"
    >
      <field column="id"          xpath="/mediawiki/page/id" />
      <field column="title"       xpath="/mediawiki/page/title" />
      <field column="revision"    xpath="/mediawiki/page/revision/id" />
      <field column="user"
        xpath="/mediawiki/page/revision/contributor/username" />
      <field column="userId"
        xpath="/mediawiki/page/revision/contributor/id" />
      <field column="text"        xpath="/mediawiki/page/revision/text"
      />
      <field column="timestamp"
        xpath="/mediawiki/page/revision/timestamp" dateTimeFormat="yyyy-MM-
        dd'T'hh:mm:ss'Z'" />
      <field column="$skipDoc"   regex="^#REDIRECT .*"
        replaceWith="true" sourceColName="text"/>
    </entity>
  </document>
</dataConfig>
```

5.3 Avro Schema File

```
{
  "namespace": "ly.stealth.xmlavro",
  "protocol": "xml",

  "types": [
    {
      "name": "Attribute",
      "type": "record",
      "fields": [
        {"name": "name", "type": "string"},
        {"name": "value", "type": "string" }
      ]
    },

    {
      "name": "Element",
      "type": "record",
      "fields": [
        {"name": "name", "type": "string"},
        {"name": "attributes", "type": {"type": "array", "items":
"Attribute"}},
        {"name": "children", "type": {"type": "array", "items": ["Element",
"string"]} }
      ]
    }
  ]
}
```


5.4 Extract and Import Script

```
#!/bin/bash

prefix="enwiki/"
suffix=".7z"

for compressed_file in enwiki/*.7z
do
    p7zip_15.14.1/bin/7za x -oXML/ $compressed_file
    xml_file=${compressed_file#$prefix}
    xml_file=${xml_file%$suffix}
    xml_file="XML/$xml_file"
    hdfs dfs -put $xml_file /user/cs4624s16_wiki/XML/.
    mv $compressed_file finished/.
    rm $xml_file
done
```

6 References

- [1] en.wikipedia.org, “Wikipedia, the free encyclopedia”, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Main_Page. [Accessed: 28- April- 2016].
- [2] meta.wikimedia.org, “Data Dumps – Meta”, 2016. [Online]. Available: https://meta.wikimedia.org/wiki/Data_dumps. [Accessed: 28- April- 2016].
- [3] dumps.wikimedia.org, "Wikimedia Downloads", 2016. [Online]. Available: <https://dumps.wikimedia.org/backup-index.html>. [Accessed: 19- Feb- 2016].
- [4] mediawiki.org, "Help:Export - MediaWiki", 2016. [Online]. Available: <https://www.mediawiki.org/wiki/Help:Export#DTD>. [Accessed: 19- Feb- 2016].
- [5] avro.apache.org, "Apache Avro™ 1.8.0 Documentation", 2016. [Online]. Available: <https://avro.apache.org/docs/current/>. [Accessed: 19- Feb- 2016].
- [6] github.com, “elodina/xml-avro”, 2016. [Online]. Available: <https://github.com/elodina/xml-avro>. [Accessed: 28- April- 2016].
- [7] Y. Koike, “A Conversion Tool from DTD to XML Schema”, 2001. [Online]. Available: https://www.w3.org/2000/04/schema_hack/. [Accessed: 28- April- 2016].
- [8] T. Moyer, "Hadoop on the Wikipedia", *Tpmoyer-gallery.appspot.com*, 2013. [Online]. Available: <https://tpmoyer-gallery.appspot.com/hadoopWikipedia>. [Accessed: 19- Feb- 2016].
- [9] hadoop.apache.org, "Welcome to Apache™ Hadoop@!", 2016. [Online]. Available: <http://hadoop.apache.org/>. [Accessed: 19- Feb- 2016].
- [10] wiki.apache.org, “DataImportHandlr – Solr Wiki”, 2015. [Online]. Available: https://wiki.apache.org/solr/DataImportHandler#Example:_Indexing_wikipedia. [Accessed: 28- April- 2016].
- [11] lucene.apache.org, "Apache Solr - Features", 2016. [Online]. Available: <http://lucene.apache.org/solr/features.html>. [Accessed: 20- Feb- 2016].
- [12] appsintheopen.com, “Map Reduce with XML Input and Multiple Avro Outputs”, 2015. [Online]. Available: <http://appsintheopen.com/posts/48-map-reduce-with-xml-input-and-multiple-avro-outputs>. [Accessed: 1- May- 2016].