پاکستان ہمکنار
Urdu Search Engine

*This document gives detailed description of developments that were made during the **4th** milestone of this project. It provides progress report explaining Urdu Specific issues and solutions for query response System*

High Performance Computing and Networking Lab

Center for Language Engineering

Al-Khawarizmi Institute Of Computer Science,

University Of Engineering and Technology, Lahore

Urdu Specific Issues and Solutions for query Response System

Contents

1. Query Response System:

For query response and indexing system, it was decided to run Apache Solr in cloud mode instead of single mode. It has a lot of benefits over single mode e.g. if duration operation, it is required to reload configuration, then it can be done very easily without interruption of Solr using Zookeeper ensemble that will be discussed later. Similarly, in cloud mode, there should be more than one instances of Solr are running. In this case, if any instance goes down due to some problem, request s could be served very easily on other Solr instances running in the cloud. Configuration details are little bit different in cloud mode as compared to single mode. One major difference is use of Zookeeper ensemble for configuration management. We have already used external Zookeeper ensemble for Hbase configuration. One option was to use that one for Solr management also. But it was decided to run separate zookeeper ensemble for query response system so that this system should not depended on any backend services running on other systems. Configuration details of ensemble is given below section. For query response system, single zookeeper server was used in ensemble [1][2][3].

As for as Solr index storage is concerned, by default it uses local file system to store index structure in form of blocks. But it can be moved to distributed file systems such as Apache Hadoop (HDFS) etc. But at basic level, local file system is enough and it can handle millions of documents very easily. But distributed file system for index has its own benefits for example one can post process index via Hadoop very easily that is very difficult on local file system[4][5].

Architecture diagram of query response system has been show in Figure 1. In current configuration, due to resources problem, we have to use single zookeeper ensemble although in production there should be at least three zookeeper instances running. In figure, “Humkinar Site core” block diagram contains website source code related portion e.g. HTML5, PHP, Apache Webserver and Redis files etc. Configuration details of Solr and Zookeeper has been discussed in next section.

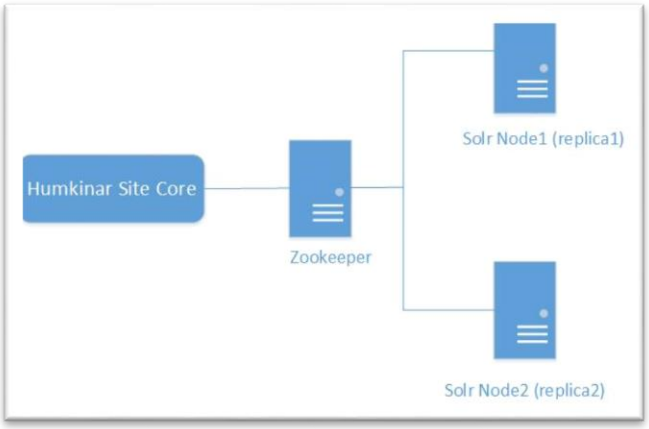

Figure 1: Query Response System Architecture

Solr Cloud Configuration Steps

Following steps were taken to configure Apache Solr in cloud mode.

i. Download Solr version as recommended by Apache Nutch (At time of configuration, it was 4.10.3).

ii. Copy example directory and rename to node1

```
cp –r $SOLR-HOME/example $SOLR_HOME/node1
```

iii. Schema.xml file in node1/solr/collection1/conf has been updated as discussed in 3rd deliverable. It is annexed in Appendix also.

iv. In order to start Apache Sor at first time, configuration files should be uploaded to zookeeper server and following command has been used for this purpose. Solr default port is 8983 and zookeeper default port is 2181. We have changed solr port to 8900. This command upload given configuration files to zookeeper and also starts Solr in cloud mode.

```
Cd $SOLR_HOME/node1

Java java -Djetty.port=8900  -Dbootstrap_confdir=./solr/collection1/conf -
Dcollection.configName=myconf -DnumShards=1 -DzkHost=10.11.21.40:2181 -jar start.jar
```

In order to quit Solr, type ctrl+c from keyboard. It will stop the solr services.

v. Replica of $SOLR_HOME was created and moved to second system (10.11.21.41) where solr second will be configured.

vi. In order to run Solr after first time, following command should be used. (Now we do not need to upload configuration again to Zookeeper as it has been already loaded at first time)

```
Cd $SOLR_HOME/node1

java -Djetty.port=8900 -DzkHost=10.11.21.40:2181 -jar start.jar
```

vii. Similar command should be used on second system to start Solr second replica.

viii. In order to maintain session, Solr should be started via screen utility (or as backend service)

ix. Solr admin can be accessed via 10.11.21.40:8900 URL in browser. Figure 2 show this admin panel which gives cloud information. It shows that Solr node at 10.11.21.40 is working properly and is leader while Solr at 10.11.21.41 is recovering.

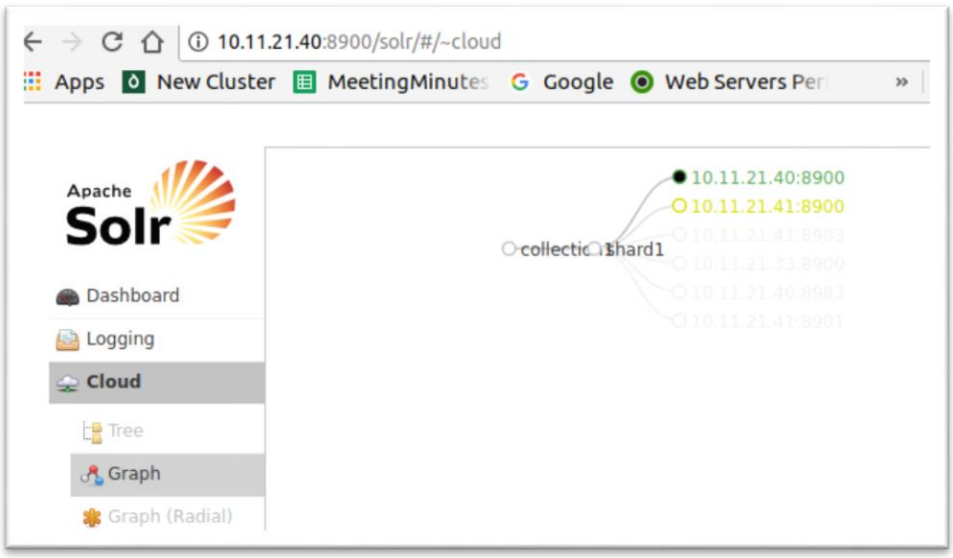

Figure 2: Solr Cloud Admin Web Interface of Humkinar

Configuration management via zkcli.sh

It is quite possible that during the operation, configuration needs to be updated. Instead of restarting Solr, Zookeeper provides option to upload latest copy of configuration files without the interruption of any other service. When new configuration is upload, then old configuration is removed. In order to upload or download latest copy of Solr Cloud configuration, following steps should be taken.

i. To download fresh copy of current configuration of Solr cloud, execute following commands in terminal on 10.11.21.41 or 10.11.21.40.

```
cd $SOLR_HOME

mkdir output

sh node1/scripts/cloud-scripts/zkcli.sh -zkhost 10.11.21.40:2181 -collection collection1 -confname myconf -solrhome node1/solr -cmd downconfig -confname myconf -confdir output
```

ii. Similarly, in order to upload latest configuration (after some update e.g. schema.xml changed etc.) following command should be used.

```
Cd $SOLR_HOME

sh node1/scripts/cloud-scripts/zkcli.sh –zkhost 10.11.21.40:2181 -collection collection1 -cmd upconfig -confname my_new_config -confdir node1/solr/conf
```

Note: new_config is updated configuration directory that is being uploaded.

iii. Many other properties can also be updated or changed using zkcli.sh scripts. Some examples can also be found in [6].

2. Zookeeper configuration for query Response System

In order to configure Zookeeper configuration for Solr Cloud, following steps were taken.

i. Download zookeeper that is compatible with Solr. We have used 3.4.x version.
ii. Extract binary and create a new configuration file in $ZOOKEEPER directory.

```
cd $ZOOKEEPER_HOME/
cp conf/zoo_sample.cfg conf/zoo1.cfg
```

iii. Update zoo1.cg with following code

```
tickTime=2000
dataDir=/home/hpcnl/USE/zookeeper-3.4.6/zookeeperData
clientPort=2181
```

iv. Create a directory zookeeperData in $ZOOKEEPER_HOME. It will keep zookeeper data etc.
v. To start/stop zookeeper, run following commands

```
bin/zkServer.sh start conf/zoo1.cfg

bin/zkServer.sh stop conf/zoo1.cfg
```

vi. Use “jps” command to know Zookeeper daemon is running or not and “nc” to know the health of server (as explained in Hbase section).

3. Urdu Specific Issues

While implementing query response system for Humkinar, we have to face many problems specific for Urdu language. We have also deployed a prototype of Humkinar website (written in PHP) to test our search results. While implementing this website, many type of problems occurred from query composition to rendering. All such problems has been discussed one by one.

Urdu Font Problem

Among several problems those we encountered at the frontend (Humkinar website), a major problem was the selection of Urdu font that would have been compatible with our theme style and as well light and readable. Default browser font could not meet out criteria.

Our selection of initial fonts, initial selection of fonts had several problems. For example, Urdu font that we found compatible with our website, had problems of split words. For example, the word “ کائنات “ and many similar words like these especially containing the Urdu letter “Hamza” in it was more prone to this problem. The word ﷲ also had the same problem, these sequences of Urdu alphabets were not supported by some fonts.

Moreover, some fonts had missing alphabets and rendering problems. For example, Urdu alignment is from right. However, the standard fonts we used were aligned to the left. The browser treated them as English and were not displayed correctly.

Solution:

First font that met our initial criteria was “Noori Nastalique”. However, this font was larger than 10 MBs and was not suitable for frontend development from the perspective of site performance. We switched to “Nafees Nastalique” that was small in size (less than 3mb) and also supported all Urdu alphabets efficiently.

Stop Words Removal Problem

Another problem that we were facing was the scoring of documents based on stop words. For example, the query اعتدال پسند شہریار خان کی دوسری اننگز کا آج اختتام gave results, those were matched based on trivial stop words as shown below.

آرمی چیف سے رچرڈ اولسن ملاقات، دہشتگردی کی خلاف پاکستان کی کوششوں کو سراہا نیوزی لینڈ میں شاہد آفرید
بچوں کے ورثا کو آج تک انصاف نہ مل س کا کراچی نے تین روزہ انسداد پولیو مہم کا افتتاح کر دیا سرگودھا: وین اور موٹرسائیکل کے درمیان تصادم، 3 افراد
جاں بحق کراچی 24 گھنٹوں کے دوران موسم سرد اور ...

This created unwanted results on the search page.

Solution:

The standard solution was implemented at Solr end which takes a list of stop words in a flat file. The Solr instance can be configured to include the stop words file in its schema.xml. The list of stop words was provided by Center of Language Engineering (CLE, UET Lahore).

Urdu Keyboard Problem

One of the major tool that had to be included at the web app was an Urdu keyboard for users in case they were not adept to standard keyboard for Urdu use. The keyboard had to be designed from the scratch.

Solution:

The keyboard was designed and made functional by Center of Language Engineering (CLE, UET, Lahore) and installed in the web app. Along with the standard keyboard, a user can use on screen Urdu keyboard for better understanding for Urdu characters’ distribution on a physical keyboard. A screenshot of the keyboard is shown below.

This keyboard has standard Urdu alphabets distribution and English keyboard option also.

4. Appendix A

This section includes those Apache Solr configuration files that were updated while configuration it in cloud mode for Humkinar project.

Schema.xml

This file defines schema of a document that is being index with some other useful properties such as store or index etc. If store is true then it means that corresponding field value should be stored and index means value should be indexed. There are some default search fields and unique key related configuration in this file also. Next section gives complete details of schema.xml file fields. Extra information such as classes' definitions for each field along other configuration has been skipped in below details.

```
<fields>
<field name="id" type="string" stored="true" indexed="true" required="true"/>
<!--  core fields  -->
<field name="batchId" type="string" stored="true" indexed="false"/>
<field name="digest" type="string" stored="true" indexed="false"/>
<field name="boost" type="float" stored="true" indexed="false"/>
<!--  fields for index-basic plugin  -->
<field name="host" type="url" stored="false" indexed="true"/>
<field name="url" type="url" stored="true" indexed="true"/>
<field name="orig" type="url" stored="true" indexed="true"/>
<!--
 stored=true for highlighting, use term vectors  and positions for fast
highlighting
-->
<field name="content" type="text_general" stored="true" indexed="true"/>
<field name="title" type="text_general" stored="true" indexed="true"/>
<field name="cache" type="string" stored="true" indexed="false"/>
<field name="tstamp" type="date" stored="true" indexed="false" default="NOW"/>
<!--  catch-all field  -->
```

```
<field name="text" type="text_general" stored="false" indexed="true"
multiValued="true"/>
<!--  fields for index-anchor plugin  -->
<field name="anchor" type="text_general" stored="true" indexed="true"
multiValued="true"/>
<!--  fields for index-more plugin  -->
<field name="type" type="string" stored="true" indexed="true" multiValued="true"/>
<field name="contentLength" type="string" stored="true" indexed="false"/>
<field name="lastModified" type="date" stored="true" indexed="false"/>
<field name="date" type="tdate" stored="true" indexed="true"/>
<!--  fields for languageidentifier plugin  -->
<field name="lang" type="string" stored="true" indexed="true"/>
<!--  fields for subcollection plugin  -->
<field name="subcollection" type="string" stored="true" indexed="true"
multiValued="true"/>
<field name="author" type="string" stored="true" indexed="true"/>
<field name="tag" type="string" stored="true" indexed="true" multiValued="true"/>
<field name="feed" type="string" stored="true" indexed="true"/>
<field name="publishedDate" type="date" stored="true" indexed="true"/>
<field name="updatedDate" type="date" stored="true" indexed="true"/>
<!--
 Custom fields created for indexing our OCR processed books
-->
<field name="publisher" type="string" stored="true" indexed="false"/>
<field name="publisherURL" type="string" stored="true" indexed="false"/>
<field name="domain" type="string" stored="true" indexed="false"/>
<field name="group" type="string" stored="true" indexed="true"/>
<!--  Custom fields created for images and business data -->
<field name="catagory" type="string" indexed="true"/>
<field name="address" type="text_general" indexed="true" stored="true"/>
<field name="city" type="string" indexed="true" stored="true"/>
<field name="country" type="string" indexed="true" stored="true"/>
<field name="rep" type="string" stored="true" indexed="false"/>
<field name="rep_title" type="string" stored="true" indexed="false"/>
<field name="webid" type="url" stored="true" indexed="false"/>
<field name="phone" type="string" indexed="false" stored="true"/>
<field name="fax" type="string" indexed="false" stored="true"/>
<field name="mobile" type="string" indexed="false" stored="true"/>
<field name="email" type="string" indexed="false" stored="true"/>
<!--  fields for Ad server  -->
<field name="keywords" type="text_general" stored="true" indexed="true"/>
<field name="adpath" type="url" stored="true" indexed="false"/>
<field name="aratio" type="text_en_splitting" stored="true" indexed="true"/>
<field name="width" type="string" stored="true" indexed="false"/>
<field name="higth" type="string" stored="true" indexed="false"/>
```

```
<field name="content_urdu" type="text_general" stored="true" indexed="true"/>
<field name="title_urdu" type="text_general" stored="true" indexed="true"/>
<!--  Field for poetry books name  -->
<field name="book_name" type="string" indexed="true" stored="true"/>
<!--  Field for Arabic text  -->
<field name="text_arabic" type="string" indexed="false" stored="true"/>
<!--  Field for defination of dictinary words.  -->
<field name="dict_word" type="exactstring" indexed="true" stored="true"/>
<field name="def_dict" type="string" indexed="false" stored="true"/>
<!--
 Ad Server variable ( Not used in production currently)
-->
<field name="bid_v" type="float" stored="true" indexed="true" default="0"/>
<!--  fields for creativecommons plugin  -->
<field name="cc" type="string" stored="true" indexed="true" multiValued="true"/>
<!--  fields for tld plugin  -->
<field name="tld" type="string" stored="false" indexed="false"/>
<field name="_version_" type="long" indexed="true" stored="true"/>
</fields>
<uniqueKey>id</uniqueKey>
<defaultSearchField>content_urdu</defaultSearchField>
<solrQueryParser defaultOperator="OR"/>
```

Jetty.xml Update

By default, Apache Solr listen on all interface. In order to limit this excess, $SOLR_HOME/node1/etc/jetty.xml file was updated with following code. All other configuration were left as it is (default).

```
 <Call name="addConnector">
      <Arg>
          <New class="org.eclipse.jetty.server.bio.SocketConnector">
            <Set name="host"><SystemProperty name="jetty.host"
default="127.0.0.1"/></Set>
            <Set name="port"><SystemProperty name="jetty.port"
default="8983"/></Set>
            <Set name="maxIdleTime">50000</Set>
            <Set name="lowResourceMaxIdleTime">1500</Set>
            <Set name="statsOn">false</Set>
          </New>
      </Arg>
    </Call>

<Call name="addConnector">
      <Arg>
```

```
          <New class="org.eclipse.jetty.server.bio.SocketConnector">
            <Set name="host"><SystemProperty name="jetty.host"
default="10.11.21.40"/></Set>
            <Set name="port"><SystemProperty name="jetty.port"
default="8983"/></Set>
            <Set name="maxIdleTime">50000</Set>
            <Set name="lowResourceMaxIdleTime">1500</Set>
            <Set name="statsOn">false</Set>
          </New>
      </Arg>
</Call>
```

Solr was restricted to listen on localhost and 10.11.21.40 interface. It was required due to the reason that it was decided to use another interface in search management system.

5. References

1. http://www.chrissulham.com/sitecore-on-solr-cloud-part-1/
2. https://lucene.apache.org/solr/4_10_3/
3. http://zookeeper.apache.org/doc/r3.4.10/zookeeperStarted.html
4. https://www.tutorialspoint.com/apache_solr/apache_solr_on_hadoop.html
5. https://hortonworks.com/hadoop-tutorial/searching-data-solr/
6. https://cwiki.apache.org/confluence/display/solr/Command+Line+Utilities