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 on local cluster development and crawling system along with code and test case*

High Performance Computing and Networking Lab

Center for Language Engineering

Al-Khawarizmi Institute Of Computer Science,

University Of Engineering and Technology, Lahore

Setting Up Local Cluster and Arbitrary Crawling

1 Contents

1. Setting Up Local Cluster:

As it is already discussed (3rd deliverable report) that Apache Hadoop comprises different daemons e.g. NameNode, JobTracker, DataNode and TaskTracker etc. In order to setup local cluster for crawling, Apache Hadoop should be configured in fully distributed mode. In this mode, it is recommended that NamdeNode and JobTracker daemons should be configured on single separate machine (known as master) and other daemons (DataNode and TaskTracker) should be configured on remaining systems (known as slaves) [1][2].

Similarly Hbase should also be configured in fully distributed mode i.e. HMaster should be on separate node (known as master) and RegionServers and QuorumPeer (Zookeeper) should be on separate machines (known as salves).

Figure 1 shows deployed network diagram used for local cluster development and crawling system. All these systems are Linux based system. Details of each system is given below sections.

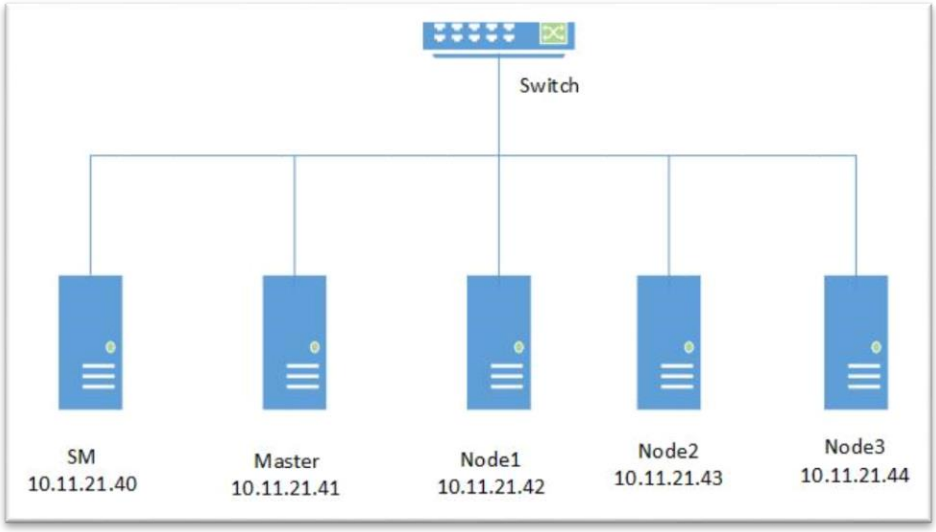

Figure 1: Local Cluster Network Diagram

1.1 Search Management (SM)

Basic indexing system and search management is deployed on this system. Apache Solr is configured here. Humkinar website is also configured on this system via Apache web server. Complete configuration details of different services running on this system is provided in a separate report on query response system.

1.2 Master

This is the master node in Hadoop cluster. On this machine, Apache Hadoop daemons Namenode and Jobtracker are running. Hadoop is configured as multi node cluster (one master and three slave nodes). Apache Hive is also configured at this machine for data analysis. One of three Apache Hbase region server daemons are also running here (slave node). Solr cloud node is also running on this machine as a backup. This node is single point of failure (in case of Hadoop 1.x version) i.e. that if this systems goes down all running jobs will be lost. In latest version of Hadoop (2.x) they have provide high availability of NameNode that reduces that reduces the chance of single point of failure[3].

1.3 Node1

Apache Hbase master daemon (Hmaster) is configured here. This machine is also running as Hadoop slave node (Datanode and Tasktracker daemons running on this machine). Our crawler is also configured here and it is crawling continuously. As our hbase is running in fully distributed mode (one master and three slave nodes). For that we are using external zookeeper ensemble that consists of three instances. One of its instance is running on this machine.

1.4 Node2 & Node3

These nodes are running as slave nodes for Hadoop and hbase i.e Datanode and Tasktracker. Hadoop slave daemons and regionserver which is hbase slave daemon are also running on both of these machines. On each machine, Apache Zookeeper ensemble instance is running. On node2 Hadoop daemon Secondry Namenode is also running.

In order to add new Datanodes (slaves) then its configuration will be same as in node3 hadoop configuration.

Similarly Hbase was also configured on these systems in such a way that HMaster daemon of Hbase was configured on Node1, region servers were configured on Master, Node2 and Node3 and Zookeeper ensemble was configured on Node1, Node2 and Node3. This distributed was deployed by following [4] tutorial. Table 1 gives the details of daemons on each system. Configuration details of Apache Hadoop and Apache Hbase in fully distributed mode is described in below section.

Table 1: Local Crawling System services (Daemons) Distribution

System	Configured Daemons
Master (10.11.21.41)	NameNode, JobTracker, SecondryNameNode, RegionServer, solr (replica)
Node1 (10.11.21.42)	DataNode, TaskTracker, HMaster, zookeeper server
Node2 (10.11.21.43)	DataNode, TaskTracker, Region Server, zookeeper server
Node3 (10.11.21.44)	DataNode, TaskTracker, Region Server, zookeeper server
SM (10.11.21.40)	Apache Solr, Httpd (web server), Redis (In cache memory), Index Storage

/etc/hosts file was updated on all systems with following information. It was required both for Hadoop as well as for Hbase configuration.

```
10.11.21.40 solr
10.11.21.41 master regionserver1
10.11.21.42 node1 hmaster zkserver1
10.11.21.43 node2 regionserver2 zkserver2
10.11.21.44 node3 regionserver3 zkserver3
```

1.5 Apache Hadoop Configuration:

First problem was the Apache Hadoop version selection. It was required that version should be compatible with Hbase and Nutch Crawler. At the time of configuring local cluster, it was found that compatible version of Hadoop and Hbase with Nutch 2.x are 1.2.1 and 0.94.x respectively. Also Hadoop 1.2.1 is most stable release in 1.x series [5].

In order to configure Apache Hadoop in fully distributed mode, following steps were carried out.

i. Java Development KIT (JDK) 8.x was downloaded and installed on each system and corresponding environment variables were also set in .bashrc file e.g. $JAVA_HOME etc.
ii. Apache Hadoop version 1.2.1 was download from nearest mirror and copied to each node.
iii. $HADOOP_HOME/conf/core-site.xml was updated (on Master and all nodes) with following code. ($HADOOP_HOME is the directory where Hadoop binary was extracted)

```
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/hpcnl/crawler/hadoop-1.2.1/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://master:9000</value>
<description>The name of the default file system.</description>
</property>
```

Note: There are many other tags that we can use according to our requirements but not required at this time.

iv. $HADOOP_HOME/conf/mapred-site.xml was updated with following code on all machines (master and slaves)

```
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>master:9001</value>
<description>The host and port of master node</description>
</property>
</configuration>
```

v. $HADOOP_HOME/conf/hdfs-site.xml was updated with following code on all machines (master and slaves). Replication factor should be more than 2 in deployment cluster to avoid data loss.

```
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
<description>Default block replication.</description>
</property>
```

vi. Updated $HADOOP/conf/master on Master system only with following information

```
master
```

This file contains information about where SecondryNamenode daemon will run started. For our case it is master system.

vii. Updated $HADOOP_HOME/conf/slaves on master machine only (Machines where data nodes will be running) like

```
Node1
Node2
Node3
```

Note: On slave nodes, do not edit conf/masters or conf/slaves that contains information of localhost. Only the thing that you can change is localhost to IP of that machine. For example on node1 you can change conf/ (master or slave) to node1 instead of localhost.

viii. Made all Hadoop systems passwordless i.e. they can SSH each other without password and user name should be same on all nodes. In our case, user name is "*hpcnl".* First we generate key pairs

via “ssh-keygen” command on all machines and then copied these keys to other nodes via “"ssh-copy-id” command.

ix. Before first time start, we format NameNode via following command on master machine.

```
$HADOOP_HOME/ bin/hadoop namenode -format
```

Note: Warning: Do not format a running cluster because this will erase all existing data in the HDFS file system.

x. In order to start Hadoop cluster, following command was executed on master machine.

```
$HADOOP_HOME/ bin/start-all.sh
```

It will start all services. But it is recommended that these services should be started separately via start-dfs.sh and start-mapred.sh scripts.

xi. In order to know different services status on any machine, run “jps” command. It will show all java services information with PID. On master system, three daemons namely NameNode, JobTracker and SecondryNameNode while on slave’s data Node and task tracker daemons were running.

xii. To stop complete cluster at once, followed script was executed

```
$HADOOP_HOME/ bin/stop-all.sh
```

Note: In case of any problem in any node, view corresponding log file in that node.

1.6 Apache Hbase Configuration:

After Apache Hadoop configuration in fully distributed mode, next we have configured Hadoop database known as Apache Hbase in fully distributed mode. Following steps were carried out for this purpose.

i. Downloaded and extracted a fresh copy of Apache Hbase version 0.94.14 on all four nodes

ii. Updated $HBASE_HOME/conf/hbase-env.sh on all nodes. In this step we have given information about JAVA_HOME and also informed Hbase that you should not manage Zookeeper i.e. external Zookeeper ensemble will be used.

```
export JAVA_HOME=/usr/java/jdk1.8.0_25/
export HBASE_MANAGES_ZK=false
```

iii. Updated $HBASE_HOME/conf/hbase-site.xml on all machines with following code.

```
<configuration>
<property>
<name>hbase.master</name>
<value>Hmaster:12000</value>
</property>
<property>
<name>hbase.master.port</name>
<value>12000</value>
</property>
<property>
<name>hbase.rootdir</name>
<value>hdfs://master:9000/USE_hbase</value>
</property>
<property>
<name>hbase.cluster.distributed</name>
<value>true</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/hpcnl/crawler/zookeeper-3.4.5/zookeeperData/1</value>
</property>
<property>
<name>hbase.zookeeper.quorum</name>
<value>zkserver1,zkserver2,zkserver3</value>
</property>
<property>
<name>hbase.zookeeper.property.clientPort</name>
<value>2181</value>
</property>
</configuration>
```

iv. Updated the $HBASE_HOME/conf/regionservers file on all the hbase cluster nodes. Added hostnames of all the region like.

```
regionserver1
regionserver2
regionserver3
```

v. In order to start/stop and test Hbase, first Hadoop services should be running properly. Following available scripts were used to start/stop Hbase cluster

```
$HBASE_HOME/bin/start-hbase.sh
$HBASE_HOME/bin/stop-hbase.sh
```

vi. In order to check java services details “jps” utility was used.

1.7 Apache Zookeeper Configuration

Apache ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. It was decided to run external zookeeper ensemble so that it should not be dependent on Hbase i.e. if Hbase goes down due to some reason even then zookeeper servers should be up. It is recommended that number of zookeeper servers should be odd rather than even so that in case of failure of one or more servers, there should not be a tie condition to continue of stop ensemble for leader [6]. For our cluster development, it was decided to use three zookeeper servers in ensemble.

For the configuration of zookeeper, following steps were carried out.

i. Downloaded zookeeper that is compatible with Hbase

ii. created a configuration for zookeeper

```
cd $ZOOKEEPER_HOME

cp conf/zoo_sample.cfg conf/zoo1.cfg
```

iii. Updated ZOOKEEPER_HOME/conf/zoo1.cfg with following code

```
tickTime=2000
dataDir=/home/hpcnl/USE/zookeeper-3.4.6/zookeeperData/2
clientPort=2181
initLimit=5
syncLimit=2
server.1=zkserver1:2888:3888
server.2=zkserver2:2889:3889
server.3=zkserver3:2890:3890
```

Note: For each server, same configuration file was be used other than datadir.

iv. Logging option was also updated via $ZOOKEEPER_HOME/conf/log4j.properties

```
log4j.rootLogger=DEBUG, CONSOLE, ROLLINGFILE
```

v. Commands used to start/stop zookeeper ensemble

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

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

vi. g properly or not, following solution was deployed.

```
echo ruok | nc zkserver1 2181
```

If it returns “imok” then server if running properly otherwise there is some problem.

vii. In order to run three Zookeeper servers, same steps should be carried out on each system.

2 Arbitrary Crawling:

As discussed in 3rd deliverable that after Apache Nutch build job completion, two directories (binaries) are created in runtime directory that are local and deploy. In order to run Nutch crawling job in distributed mode, deploy binary should be used. It run on MapReduce job on Hadoop workers. We have configured Apache Nutch on Master Node for distributed crawling. (Although it can be configured on any system or even on separate system also).

Apache Nutch Configuration

Apache Nutch configuration was updated with following information.

i. Replaced $NUTCH_HOME/conf/hbase-site.xml with Hbase respective file (hbase-site.xml) as discussed in Hbase configuration section.

ii. $NUTCH_HOME/conf/nutch-site.xml was updated with following code

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
   <name>http.agent.name</name>
   <value>USE-crawler</value>
</property>
<property>
   <name>storage.data.store.class</name>
   <value>org.apache.gora.hbase.store.HBaseStore</value>
   <description>Default class for storing data</description>
</property>
<property>
  <name>plugin.includes</name>
 <value>protocol-httpclient|protocol-http|indexer-solr|urlfilter-
regex|parse-(html|tika)|index-(basic|more|urdu)|urlnormalizer-
(pass|regex|basic)|scoring-opic|myPlugin</value>
</property>
<property>
<name>parser.character.encoding.default</name>
<value>utf-8</value>
</property>
<property>
  <name>http.robots.403.allow</name>
  <value>true</value></property>
<property>
  <name>db.max.outlinks.per.page</name>
  <value>-1</value>
  </property>
<property>
  <name>http.robots.agents</name>
  <value>USE-crawler,*</value>
  </property>

</configuration>
```

iii. Other important configuration files are given in Appendix A.

iv. After some manual effort for seed to find Urdu websites, about 1000 URLs were collected.

v. In order to run Distributed job on Hadoop cluster, it is required that seed file should be placed on HDFS rather than local file system. So we placed seed file in HDFS.

vi. Nutch provides two scripts “nutch.sh” and “crawl.sh” in bin directory. Second script is properly scheduled nutch steps in sequence. It is recommended to use this instead of “nutch.sh” at starting point.

vii. Following command was used to run crawler on Hadoop cluster for about 200 iterations. This command is similar to what we did at the time of single node cluster. But now this job is distributed to three workers nodes instead on single system.

```
bin/crawl urls/urdu-seed.txt USE_CrawlData http://10.11.21.40:8900/solr 200
```

viii. After the job completion, raw content was stored on HDFS while index was stored on search management (SM) system.

ix. Figure 2 shows HDFS states when job was finished. Where *USE_hbase* is Apache Hbase base directory that was set at time of configuration. Next is the table name that we gave when started the job. All crawled content information are kept is this directory. This directory contains many sub directories that contains raw content and some parsed content such as out links, in links, meta data of page. Figure 3 shows complete list of all such directories. It has been obtained via Hadoop web interface that can be accessed via port 10.11.21.41:50070 for our case.

```
[hpcnl@master ~]$ hadoop fs -du /USE_hbase/USE_CrawlData_webpage
Warning: $HADOOP_HOME is deprecated.

Found 109 items
4849         hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/.tableinfo.0
0            hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/.tmp
4297163428   hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/02bfce56e16c
8586393296   hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/05a114561ad5
13979076890  hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/05f7c9205301
8447235182   hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/097f4f6b0511
10281122265  hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/0af729688908
8375653067   hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/0dd32fd6a20c
2222574289   hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/0e7f51d4f649
6046442239   hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/12bf8e10ada5
8634540113   hdfs://master:9000/USE_hbase/USE_CrawlData_webpage/1566c087bc8b
```

Figure 2: HDFS State after job completion

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner
.regioninfo	file	0.72 KB	2	64 MB	2017-07-08 16:21	rw-r--r--	hpcnl
.tmp	dir				2017-07-19 17:32	rwxr-xr-x	hpcnl
f	dir				2017-07-19 17:32	rwxr-xr-x	hpcnl
h	dir				2017-07-08 16:21	rwxr-xr-x	hpcnl
il	dir				2017-07-08 16:21	rwxr-xr-x	hpcnl
mk	dir				2017-07-19 14:34	rwxr-xr-x	hpcnl
mtdt	dir				2017-07-19 14:34	rwxr-xr-x	hpcnl
ol	dir				2017-07-08 16:25	rwxr-xr-x	hpcnl
p	dir				2017-07-08 16:21	rwxr-xr-x	hpcnl
recovered.edits	dir				2017-07-19 10:12	rwxr-xr-x	hpcnl
s	dir				2017-07-19 14:34	rwxr-xr-x	hpcnl

Figure 3: Crawled content fileds of a document

x. Apache Nutch also created index of crawled document. That can be searched via Solr admin or through REST API. Figure 4 shows the statistics of Apache Solr current index. There are about 4 million documents in index at this time.

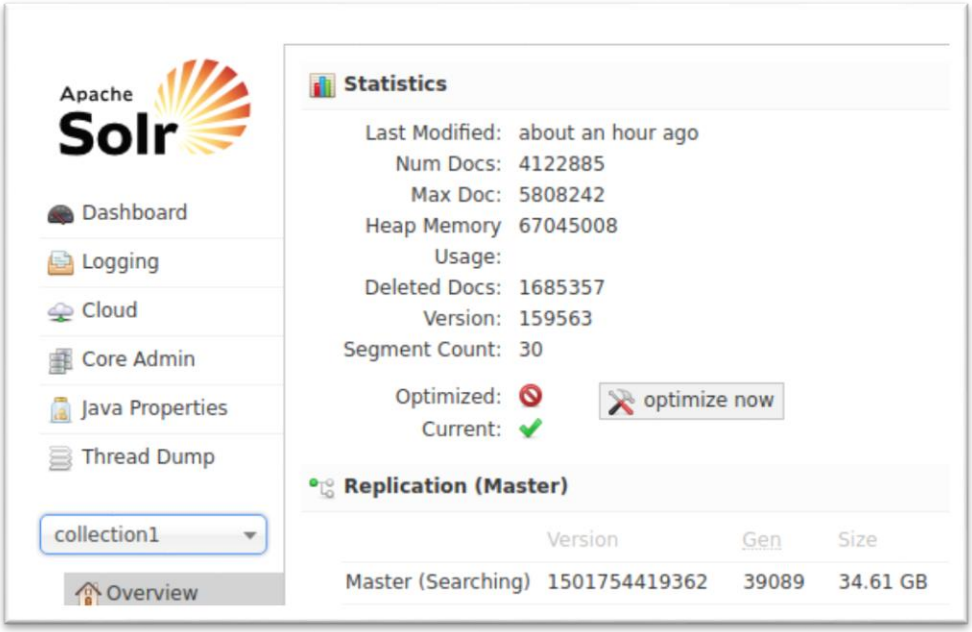

Figure 4: Solr Index Statistics

xi. Solr data directory of collection1 core contains index in raw form. Figure 5 shows such sample index in data directory in Solr core.

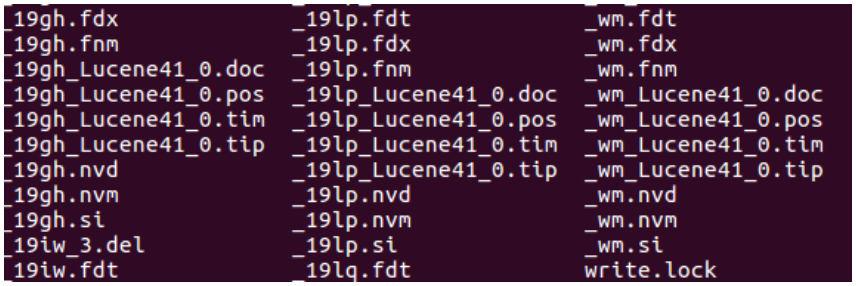

Figure 5: Solr Indexed files list of crawled data

3 Appendix A

This section contains some auxiliary files of Apache Nutch configuration that were used during crawling.

Solrindex-mapping.xml

This files contains mapping of Nutch fields to corresponding fields in Apache Solr. In case of any new field addition in Nutch, this field should be updated.

```
<mapping>
        <!-- Simple mapping of fields created by Nutch IndexingFilters
             to fields defined (and expected) in Solr schema.xml.

             Any fields in NutchDocument that match a name defined
             in field/@source will be renamed to the corresponding
             field/@dest.
             Additionally, if a field name (before mapping) matches
             a copyField/@source then its values will be copied to
             the corresponding copyField/@dest.

             uniqueKey has the same meaning as in Solr schema.xml
             and defaults to "id" if not defined.
         -->
        <fields>
                <field dest="content" source="content"/>
                <field dest="title" source="title"/>
                <field dest="host" source="host"/>
                <field dest="batchId" source="batchId"/>
                <field dest="boost" source="boost"/>
                <field dest="digest" source="digest"/>
                <field dest="tstamp" source="tstamp"/>
                <field dest="pageLength" source="pageLength"/>
                <field dest="ur_pcnt" source="ur_pcnt"/>
                <!-- field dest="pf_score" source="pf_score"/ -->
        </fields>
        <uniqueKey>id</uniqueKey>
</mapping>
```

Regex-urlfilter.txt

This file controls what type of MIME documents are allowed to crawl or what should be stopped. By default, it only allows text and block images, compressed files, JavaScript and CSS files etc.

```
# The default url filter.

# Better for whole-internet crawling.

# Each non-comment, non-blank line contains a regular expression
# prefixed by '+' or '-'.  The first matching pattern in the file
# determines whether a URL is included or ignored.  If no pattern
# matches, the URL is ignored.

# skip file: ftp: and mailto: urls
-^(file|ftp|mailto):

# skip image and other suffixes we can't yet parse
# for a more extensive coverage use the urlfilter-suffix plugin
-
\.(gif|GIF|jpg|JPG|png|PNG|ico|ICO|css|CSS|sit|SIT|eps|EPS|wmf|WMF|zip|ZIP|ppt|PPT
|mpg|MPG|xls|XLS|gz|GZ|rpm|RPM|tgz|TGZ|mov|MOV|exe|EXE|JPEG|bmp|BMP|js|JS)$

# skip URLs containing certain characters as probable queries, etc.
-[?*!@=]

# skip URLs with slash-delimited segment that repeats 3+ times, to break loops
-.*(/[^/]+)/[^/]+\1/[^/]+\1/

# accept anything else
+.
```

4 References:

1. https://www.quora.com/What-is-the-difference-between-Namenode-+-Datanode-and-Jobtracker-+-Tasktracker-and-Combiners-Shufflers-and-Mappers+Reducers-in-the-following-ways
2. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/
3. https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithNFS.html
4. http://jayatiatblogs.blogspot.com/2013/01/hbase-installation-fully-distributed.html
5. http://nutch.apache.org
6. https://www.quora.com/HBase-Why-we-run-zookeeper-with-odd-number-of-instance