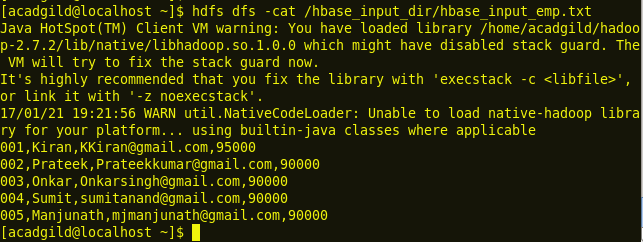
**Data Bulk Loading into HBase Table Using MapReduce**

In this blog, we will be discussing the steps to perform data bulk loading file contents from HDFS path into an HBase table using Java MapReduce API. Before, moving forward you can follow below link blogs to gain more knowledge on HBase and its working.  
[Beginners Guide to Apache HBase](https://acadgild.com/blog/apache-hbase-beginners-guide/)  
[Integrating Hive with HBase](https://acadgild.com/blog/integrating-hive-with-hbase/)  
[Performing CRUD Operations on HBase Using Java API](https://acadgild.com/blog/perform-crud-operations-hbase-using-java-api/)  
[Introduction to HBase Filters](https://acadgild.com/blog/introduction-hbase-filters/)  
[Read and Write Operations in HBase](https://acadgild.com/blog/read-write-operations-hbase/)  
[How to Import Table from MySQL to HBase](https://acadgild.com/blog/how-to-import-table-from-mysql-to-hbase/)  
Apache HBase gives us a random, real-time, read/write access to [Big Data,](https://acadgild.com/big-data/big-data-development-training-certification) but here it is more important that how do we get the data loaded into HBase. As HBase Put API can be used to insert the data into HDFS, but inserting the every record into HBase using the Put API is lot slower than the bulk loading.  
Thus, it is better to load a complete file content as a bulk into the HBase table using Bulk load function.  
Bulk loading in HBase is the process of preparing HFiles and loading it directly into the region servers.   
In our example, we will be using a sample data set hbase\_input\_emp.txt which is saved in our hdfs directory hbase\_input\_dir. You can download this sample data set for practice from the below link.  
**DATASET**  
[HBase\_input\_emp.txt](https://drive.google.com/open?id=0B_P3pWagdIrrUDJrT2pvSzIxbG8)  
Please refer the description for the above data set containing  three columns named as:  
Column 1: Employee Id  
Column 2: Employee name  
Column 3: Employee mail id  
Column 4: Employee salary  
You can follow below steps to perform bulk load data contents from Hdfs to HBase via MapReduce job.  
**Extract the data from the source, and load into HDFS.**  
If data is in Oracle, MySQL you need to fetch it using Sqoop or any such tools which gives mechanism to import data directly from a database into HDFS. If your raw files such as .txt, .pst, .xml are located in any servers then simply pull it and load into HDFS. HBase doesn’t prepare HFiles directly reading data from the source.  
As of our example, our data is already available in our hdfs path. We can use cat command to see the input file hbase\_input\_emp.txt content which is saved in the hbase\_input\_dir folder of hdfs path.  
hdfs dfs -cat /hbase\_input\_dir/hbase\_input\_emp.txt  
  
  
**Transform the data into HFiles via MapReduce job.**   
Here we write a MapReduce job which will process our data and create HFile. There will be only Mapper class and will be no Reducer class. In our code, we configure HFileOutputFormat.configureIncrementalLoad() doing which HBase creates its own Reducer class.

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.client.HTable;

import org.apache.hadoop.hbase.client.Put;

import org.apache.hadoop.hbase.io.ImmutableBytesWritable;

import org.apache.hadoop.hbase.mapreduce.HFileOutputFormat;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.client.Put;

import org.apache.hadoop.hbase.io.ImmutableBytesWritable;

import org.apache.hadoop.hbase.util.Bytes;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class HBaseBulkLoad {

public static class BulkLoadMap extends Mapper<LongWritable, Text, ImmutableBytesWritable, Put> {

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String[]parts=line.split(",");

String rowKey = parts[0];

//The line is splitting the file records into parts wherever it is comma (‘,’) separated, and the first column are considered as rowKey.

ImmutableBytesWritable HKey = new ImmutableBytesWritable(Bytes.toBytes(rowKey));

//Here the row key is first converted to Bytes as Hbase understand its data as Bytes, and also object is created as ImmutableBytesWriteable

Put HPut = new Put(Bytes.toBytes(rowKey));

//This will write the rowKey values into Hbase while creating an object.

//Here the fields of tables inside Hbase is are stated to be written

HPut.add(Bytes.toBytes("id"), Bytes.toBytes("name"), Bytes.toBytes(parts[1]));

HPut.add(Bytes.toBytes("id"), Bytes.toBytes("mail\_id"), Bytes.toBytes(parts[2]));

HPut.add(Bytes.toBytes("id"), Bytes.toBytes("sal"), Bytes.toBytes(parts[3]));

context.write(HKey,HPut);

//first we are creating instance PUT with 1st field as row key,

}

}

public static void main(String[] args) throws Exception {

Configuration conf = HBaseConfiguration.create();

String inputPath = args[0];

HTable table=new HTable(conf,args[2]);

conf.set("hbase.mapred.outputtable", args[2]);

Job job = new Job(conf,"HBase\_Bulk\_loader");

job.setMapOutputKeyClass(ImmutableBytesWritable.class);

job.setMapOutputValueClass(Put.class);

job.setSpeculativeExecution(false);

job.setReduceSpeculativeExecution(false);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(HFileOutputFormat.class);

job.setJarByClass(HBaseBulkLoad.class);

job.setMapperClass(HBaseBulkLoad.BulkLoadMap.class);

FileInputFormat.setInputPaths(job, inputPath);

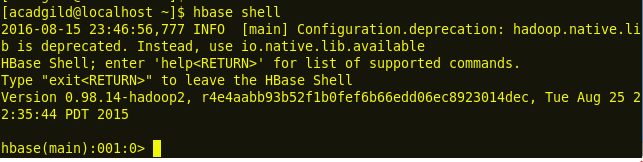
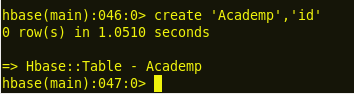
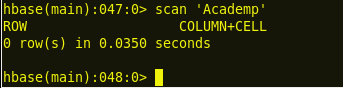
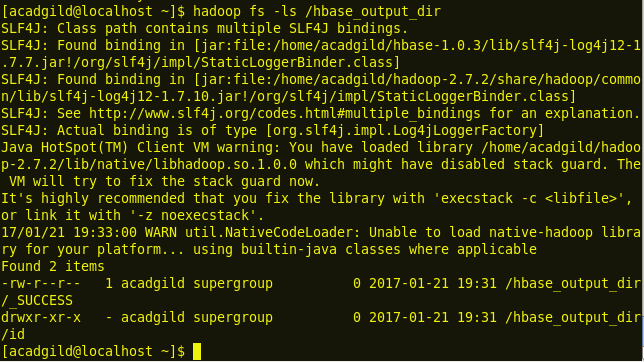
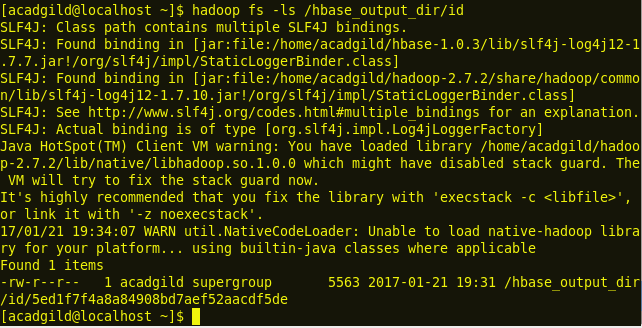
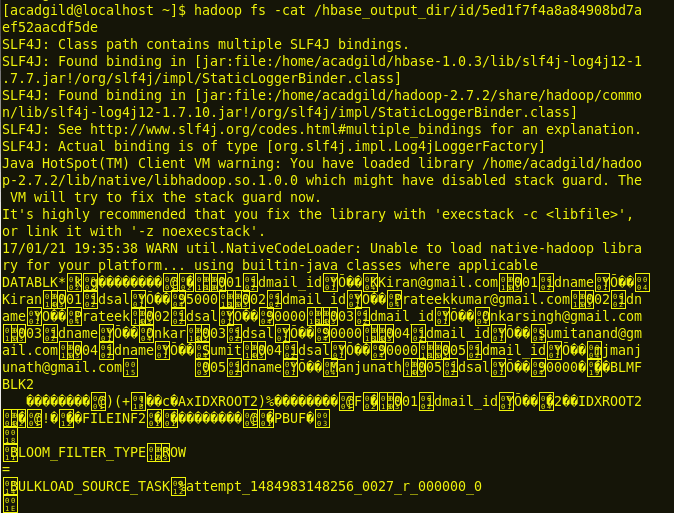
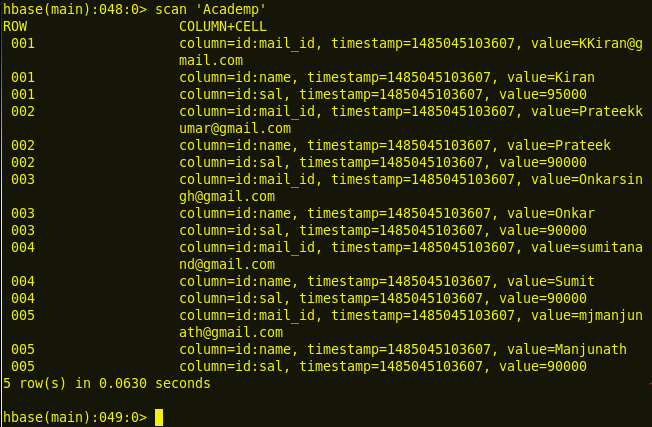
TextOutputFormat.setOutputPath(job, new Path(args[1]));

HFileOutputFormat.configureIncrementalLoad(job, table);

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

The above step finishes the MapReduce programming part.  
Now, we need to create a new table in Hbase to import table contents from hdfs input directory. So, follow the below steps to import the contents from hdfs path to Hbase table.  
**Enter HBase shell:**  
Before entering to HBase shell user should start the start HBase service. Use below command to start HBase services.  
**start-hbase.sh**  
https://i0.wp.com/s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/HBase+MapReduce+Bulk+Load/2.1_start_hbase.sh.PNG?zoom=0.9024999886751175&resize=332%2C23&ssl=1  
After starting the hmaster service use below command to enter HBase shell.  
**HBase shell**  
  
**Create table:**  
We can use create command to create a table in HBase.  
**Create ‘Academp’,’id’**  
  
**Scan table:**  
We can use **scan** command to see a table contents in Hbase.  
**Scan ‘Academp’**  
  
We can observe from the above image no contents are available in the table **Academp**  
**Export  Hadoop\_classpath:**  
In the next step, we need to load the HBase library files into the Hadoop classpath this enables the Hadoop client to connect to HBase and get the number of splits.   
export HADOOP\_CLASSPATH=$HBASE\_HOME/lib/\*   
https://i1.wp.com/s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/HBase+MapReduce+Bulk+Load/1.2_export_HADOOP_CLASSPATH_HBASE_HOME_lib.PNG?zoom=0.9024999886751175&resize=600%2C43&ssl=1  
**Mapreduce jar execution:**  
Now, run the MapReduce job by following below command to generate the HFiles.  
hadoop jar /home/acadgild/Desktop/BKLoad.jar /hbase\_input\_dir/hbase\_input\_emp.txt /hbase\_output\_dir Academp   
https://i0.wp.com/s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/HBase+MapReduce+Bulk+Load/1.3_hadoop_jar_BKLoad_inputfile_outputdir_hbasetablname.PNG?zoom=0.9024999886751175&resize=642%2C39&ssl=1  
Here, the first parameter is the input the input directory where our input file is saved, the second parameter is the output directory where we will be saving the HFiles, and the third parameter is the HBase table name.  
Now, let us use list command to list the HFiles which are stored in our output directory ‘hbase\_output\_dir’  
hadoop fs -ls /hbase\_output\_dir    
  
hadoop fs -ls /hbase\_output\_dir/id  
  
We can use below command to see the output HFile content which is saved in the sub-directory ‘id’  
hadoop fs -cat /hbase\_output/dir/id/5ed1f7…..  
  
After the data has been prepared using HFileOutputFormat, it is loaded into the cluster using completebulkload. This command line tool iterates through the prepared data files, and for each one determines the region the file belongs to. It then contacts the appropriate Region Server which adopts the HFile, moving it into its storage directory and making the data available to clients.  
Now, load the files into HBase by telling the RegionServers where to find them.   
**HBase hadoop jar execution:**  
Once the HFiles are created in HDFS directory, we can use below command to store the HFiles contents into HBase table.   
hadoop jar /home/acadgild/Downloads/hbase-server-0.98.14-hadoop2.jar completebulkload /hbase\_output\_dir/ Academp   
https://i2.wp.com/s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/HBase+MapReduce+Bulk+Load/1.7_hadoop_jar_hbase_server_hadoop2_jar_hbase_output_dir_hbasetablename.PNG?zoom=0.9024999886751175&resize=597%2C40&ssl=1  
**Scan Academp table:**  
Now, we can use scan command on the table Academp to see the contents which are exported from HDFS path.  
scan ‘Academp’  
  
Thus, from the above steps, we can observe that we have successfully imported bulk data into an HBase table using Java API.   
We hope this post has been helpful in understanding importing bulk data into HBase table. In case of any queries, feel free to comment below and we will get back to you at the earliest.