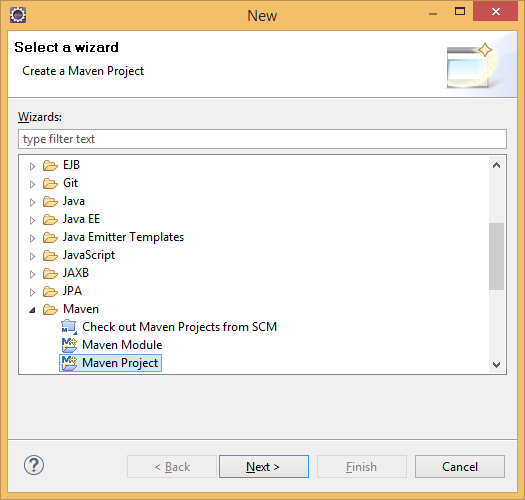
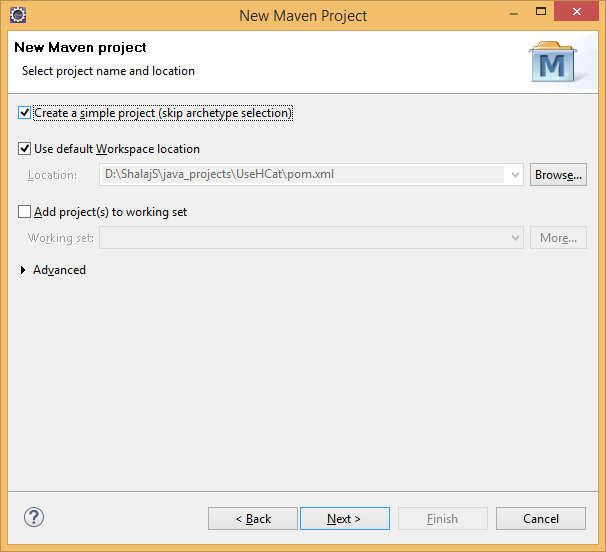
# HCatlog mapreduce example

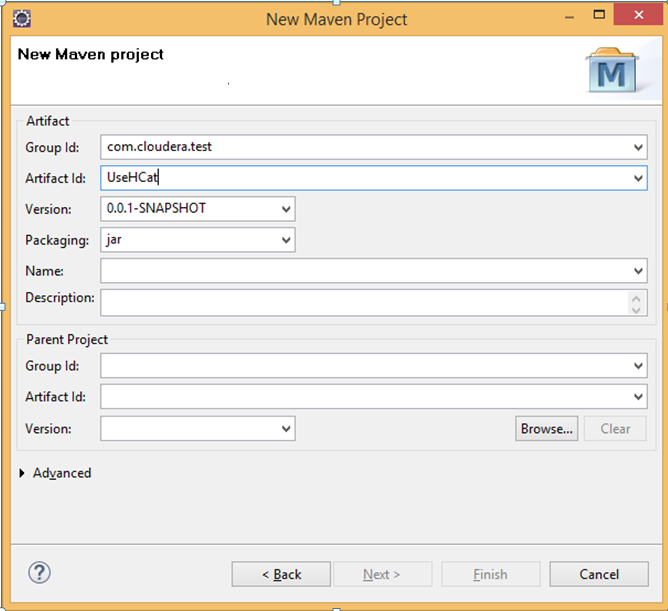
HCatalog is a table storage management tool for Hadoop that exposes the tabular data of Hive metastore to other Hadoop applications. It enables users with different data processing tools (Pig, MapReduce) to easily write data onto a grid

So instead of reading and writing file on hdfs, we can directly use hive schema and tables to read data from hive and write data to hive using Hcatlog

1. Create Maven project







Click Finish

pom.xml file

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.cloudera.test</groupId>  <artifactId>UseHCat</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>hcatalog-examples</name>  <url>http://maven.apache.org</url>  <properties>  <hadoop.version>2.5.0-cdh5.2.0</hadoop.version>  <hcatalog.version>0.13.1-cdh5.2.0</hcatalog.version>  </properties>  <build>    <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-shade-plugin</artifactId>  <configuration>  <createDependencyReducedPom>true</createDependencyReducedPom>  </configuration>  <executions>  <execution>  <phase>package</phase>  <goals>  <goal>shade</goal>  </goals>  <configuration>  <transformers>  <transformer  implementation=*"org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"* />  <transformer  implementation=*"org.apache.maven.plugins.shade.resource.ManifestResourceTransformer"*>  </transformer>  </transformers>  </configuration>  </execution>  </executions>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>2.3.2</version>  <configuration>  <source>1.6</source>  <target>1.6</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-dependency-plugin</artifactId>  <version>2.4</version>  <executions>  <execution>  <id>copy-dependencies</id>  <phase>package</phase>  <goals>  <goal>copy-dependencies</goal>  </goals>  <configuration>  <outputDirectory>${project.build.directory}/lib</outputDirectory>  <overWriteReleases>false</overWriteReleases>  <overWriteSnapshots>false</overWriteSnapshots>  <overWriteIfNewer>true</overWriteIfNewer>  </configuration>  </execution>  </executions>  </plugin>  </plugins>    </build>  <repositories>  <repository>  <id>cloudera-releases</id>  <url>https://repository.cloudera.com/artifactory/cloudera-repos</url>  <releases>  <enabled>true</enabled>  </releases>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.8.1</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-client</artifactId>  <version>${hadoop.version}</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.apache.hive.hcatalog</groupId>  <artifactId>hive-hcatalog-core</artifactId>  <version>${hcatalog.version}</version>  <scope>compile</scope>  <exclusions>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-annotations</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-auth</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-archives</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-common</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-hdfs</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-app</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-common</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-core</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-hs</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-jobclient</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-shuffle</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-client</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-common</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-api</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-server-common</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-server-nodemanager</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-server-resourcemanager</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-server-web-proxy</artifactId>  </exclusion>  <exclusion>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-yarn-server-tests</artifactId>  </exclusion>  </exclusions>  </dependency>  </dependencies>  </project> |

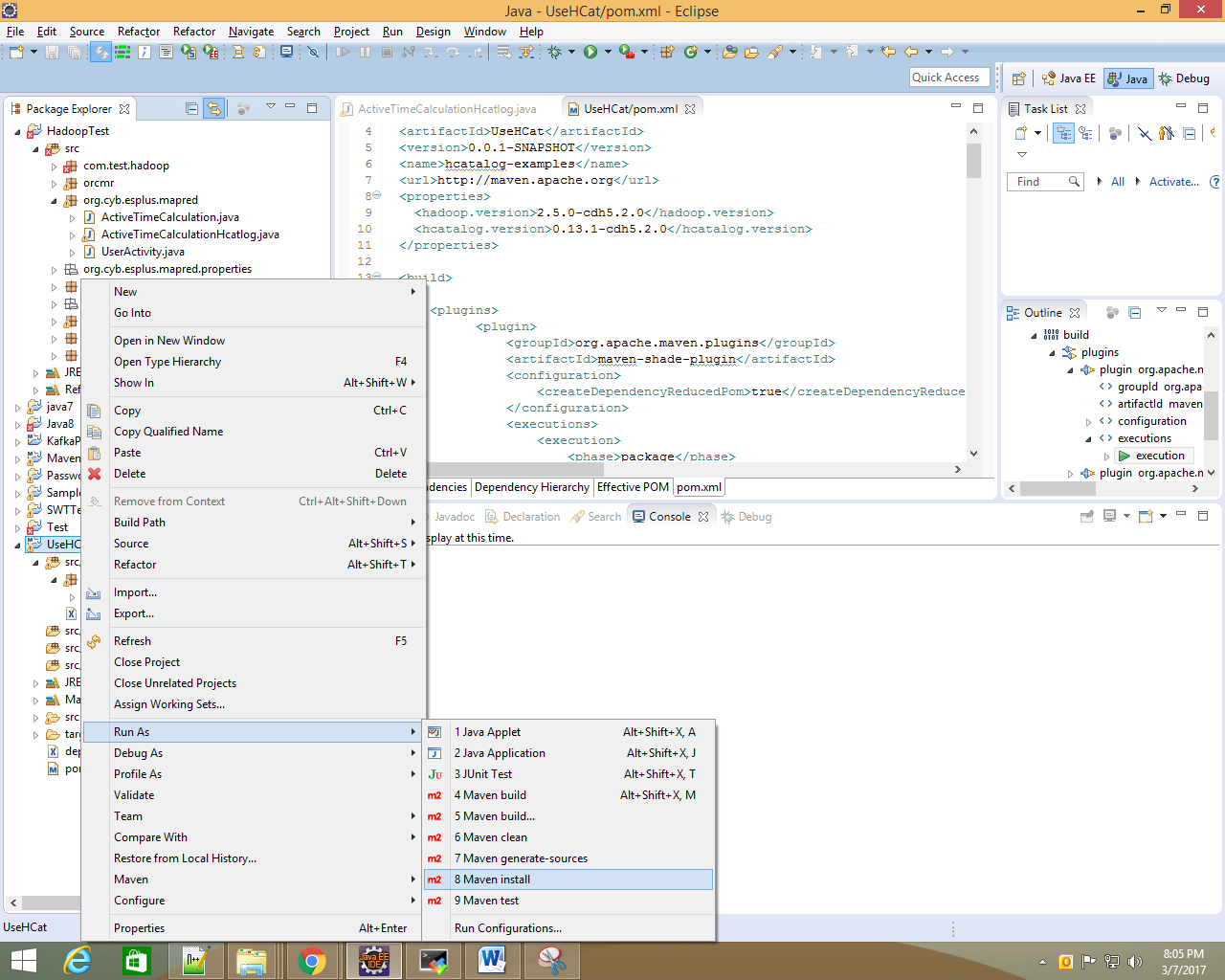
Sample program , it is just reading one table and writing same data on other table

|  |
| --- |
| **package** org.cyb.esplus.mapred;  //------------------------------------------------------------------------------  **import** java.io.IOException;  **import** java.util.Map;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.conf.Configured;  **import** org.apache.hadoop.io.NullWritable;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.io.WritableComparable;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.util.Tool;  **import** org.apache.hadoop.util.ToolRunner;  **import** org.apache.hive.hcatalog.data.DefaultHCatRecord;  **import** org.apache.hive.hcatalog.data.HCatRecord;  **import** org.apache.hive.hcatalog.data.schema.HCatSchema;  **import** org.apache.hive.hcatalog.mapreduce.HCatBaseInputFormat;  **import** org.apache.hive.hcatalog.mapreduce.HCatInputFormat;  **import** org.apache.hive.hcatalog.mapreduce.HCatOutputFormat;  **import** org.apache.hive.hcatalog.mapreduce.OutputJobInfo;  //------------------------------------------------------------------------------  /\*\*  \*  \*  \* hadoop jar SampleHcatlogMapRed.jar org.cyb.esplus.mapred.ActiveTimeCalculationHcatlog dummy\_input dummy\_output  \*  \*/  **public** **class** SampleHcatlogMapRed **extends** Configured **implements** Tool{  //---------------------------------------------------------------------------------------  /\*\*  \* Mapper class for Calculating Active Time  \*/  **public** **static** **class** SampleMapper **extends** Mapper<WritableComparable<?>,HCatRecord,Text,HCatRecord> {    /\*  \* (non-Javadoc)  \* @see org.apache.hadoop.mapreduce.Mapper#map(KEYIN, VALUEIN, org.apache.hadoop.mapreduce.Mapper.Context)  \*/  @Override  **protected** **void** map(WritableComparable<?> key, HCatRecord value, org.apache.hadoop.mapreduce.Mapper<WritableComparable<?>, HCatRecord,  Text, HCatRecord>.Context context) **throws** IOException, InterruptedException {    HCatSchema userActivitySchema = HCatBaseInputFormat.*getTableSchema*(context.getConfiguration());      String userName = value.getString("username", userActivitySchema);  /\*Date date = value.getDate("date", userActivitySchema);  Timestamp dateTime = value.getTimestamp("datetime", userActivitySchema);\*/    context.write(**new** Text (userName),value);      }  }  //------------------------------------------------------------------------------  //  // End of Mapper Class  //  //------------------------------------------------------------------------------  /\*\*  \* Reducer class for writing hcatrecord  \*/  **public** **static** **class** SampleReducer **extends** Reducer<Text, HCatRecord, NullWritable, HCatRecord>{  /\*  \* (non-Javadoc)  \* @see org.apache.hadoop.mapreduce.Reducer#reduce(KEYIN, java.lang.Iterable, org.apache.hadoop.mapreduce.Reducer.Context)  \*/  @Override  **protected** **void** reduce(Text key, Iterable<HCatRecord> values,org.apache.hadoop.mapreduce.Reducer<Text, HCatRecord,NullWritable,HCatRecord>.Context context) **throws** IOException, InterruptedException {    **for**(HCatRecord value : values) {  /\* by default values are tab separated  \* if you want to set some specifc columns ----  \* HCatSchema outputSchema = HCatBaseOutputFormat.getTableSchema(context.getConfiguration());  \* HCatSchema inputschema = HCatBaseInputFormat.getTableSchema(context.getConfiguration());  \* HCatRecord record = new DefaultHCatRecord(2);  \* record.setString("username", outputSchema, value.getString("username", inputschema));  \* record.setString("machinename", outputSchema, value.getString("machinename", inputschema));  \* context.write(NullWritable.get(), record);  \*/  context.write(NullWritable.*get*(), value);  }  }  }  //-----------------------------------------------------------------------------------------------------------------    //------------------------------------------------------------------------------  //  // End of Reducer Class  //  //------------------------------------------------------------------------------  /\* Run method for configuring all required properties and run the map reduce job  \* @param arg argument passed as command line arguments  \* @return 0 in case of success else 1  \* @throws Exception in case of any IO Exception  \*/  @Override  **public** **int** run(String[] arg) **throws** Exception {  Configuration conf = getConf();    Job job = Job.*getInstance*();  job.setJobName("ActiveTimeHcatolg");  job.setJarByClass(SampleHcatlogMapRed.**class**);    String inputTableName = arg[0];  // need to create output table in advance the default field terminator is tab (\t)  String outputTableName = arg[1];  // if dbName is null it consider default schema  String dbName = "user\_activity\_tracking";  String filter = **null**;    /\*  \* String filter ="ds <= \"20110925\" and ds >= \"20110924\"";  \* we can define filter and pass it to setInput method  \*/  HCatInputFormat.*setInput*(job, dbName, inputTableName,filter);    job.setInputFormatClass(HCatInputFormat.**class**);    job.setMapperClass(SampleMapper.**class**);  job.setMapOutputKeyClass(Text.**class**);  job.setMapOutputValueClass(DefaultHCatRecord.**class**);      job.setReducerClass(SampleReducer.**class**);  job.setOutputKeyClass(NullWritable.**class**);  job.setOutputValueClass(DefaultHCatRecord.**class**);    Map<String,String> partitions = **null**;    /\*  \* we can add partition value in map and pass it to setoutput method  \* Map partitions = new HashMap<String, String>(1);  \* partitions.put("ds", "20110924");  \*/    HCatOutputFormat.*setOutput*(job, OutputJobInfo.*create*(dbName, outputTableName,partitions));  // when we use below method it give us issue HCatOutputFormat not initialized  //HCatSchema schema = HCatOutputFormat.getTableSchema(conf);  // deprecated mehod work fine  HCatSchema schema = HCatOutputFormat.~~getTableSchema~~(job);  HCatOutputFormat.*setSchema*(job, schema);  // if we don't set the output format class ,  //job put output files under scratch folder which doesn't pick by table  job.setOutputFormatClass(HCatOutputFormat.**class**);    System.*exit*(job.waitForCompletion(**true**)? 0 : 1);  **return** 0;  }  //-----------------------------------------------------------------------------------------------------------------  /\*\*  \* main method for this class  \* **@param** args command line argument  \* **@throws** Exception  \*/  **public** **static** **void** main(String args[]) **throws** Exception {  ToolRunner.*run*(**new** Configuration(), **new** SampleHcatlogMapRed(),args);  System.*exit*(0);  }  //-----------------------------------------------------------------------------------------------------------------  }  //------------------------------------------------------------------------------  //  // End of file  //  //------------------------------------------------------------------------------ |

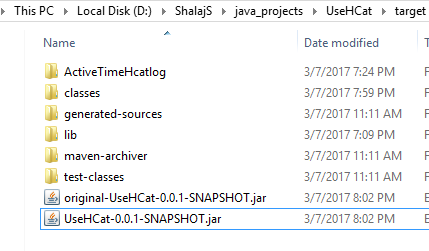
Note: here output table is tab separated

|  |
| --- |
| CREATE TABLE dummy\_output(  username string,  machinename string,  machineid string,  ipaddress string,  domainname string,  pcmlogeventid string,  modulename string,  filename string,  productname string,  companyname string,  description string,  text string,  activetime double,  datetime timestamp,  timeslot int,  date date)  Row Format delimited  fields terminated by '\t' |

Run mvn install to create jar file



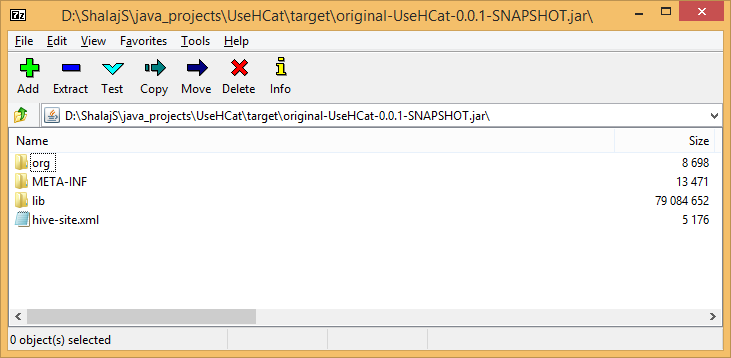
Check target folder, you will find jar created with dependencies



But when we use this jar to run mapreduce job, we encountered below issue

|  |
| --- |
| Caused by: org.datanucleus.exceptions.NucleusUserException: Persistence process has been specified to use a ClassLoaderResolver of name "datanucleus" yet this has not been found by the DataNucleus plugin mechanism. Please check your CLASSPATH and plugin specification. |

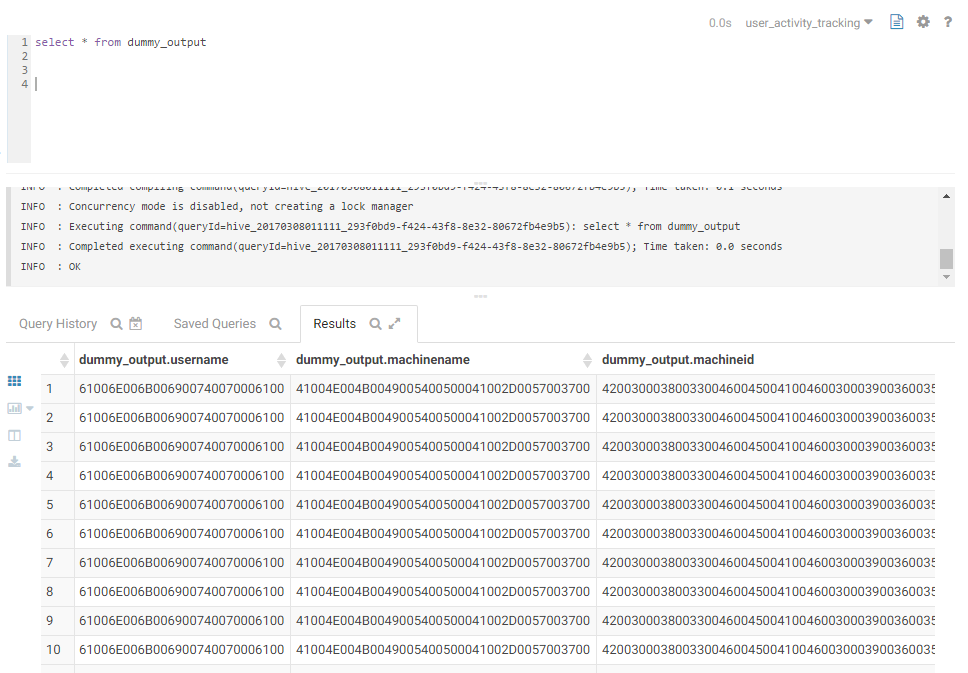
To resolve this issue we need to add all required library file(already created by maven under lib folder) under **original-UseHCat-0.0.1-SNAPSHOT.jar** file



Now use this jar to run Hadoop jar command

|  |
| --- |
| [root@mac127 esplus]# hadoop jar original-UseHCat-0.0.1-SNAPSHOT.jar org.cyb.esplus.mapred.ActiveTimeCalculationHcatlog dummy\_input1 dummy\_output  17/03/08 01:10:59 INFO hive.metastore: Trying to connect to metastore with URI thrift://mac127:9083  17/03/08 01:11:00 INFO hive.metastore: Connected to metastore.  17/03/08 01:11:00 INFO Configuration.deprecation: mapred.output.dir is deprecated. Instead, use mapreduce.output.fileoutputformat.outputdir  17/03/08 01:11:01 INFO client.RMProxy: Connecting to ResourceManager at mac127/172.27.155.127:8032  17/03/08 01:11:02 INFO Configuration.deprecation: io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  17/03/08 01:11:02 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.  17/03/08 01:11:03 INFO Configuration.deprecation: mapred.input.dir is deprecated. Instead, use mapreduce.input.fileinputformat.inputdir  17/03/08 01:11:03 INFO log.PerfLogger: <PERFLOG method=OrcGetSplits from=org.apache.hadoop.hive.ql.io.orc.ReaderImpl>  17/03/08 01:11:03 INFO orc.OrcInputFormat: FooterCacheHitRatio: 0/1  17/03/08 01:11:03 INFO log.PerfLogger: </PERFLOG method=OrcGetSplits start=1488915663024 end=1488915663058 duration=34 from=org.apache.hadoop.hive.ql.io.orc.ReaderImpl>  17/03/08 01:11:03 INFO mapreduce.JobSubmitter: number of splits:1  17/03/08 01:11:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1484320332413\_0279  17/03/08 01:11:03 INFO impl.YarnClientImpl: Submitted application application\_1484320332413\_0279  17/03/08 01:11:03 INFO mapreduce.Job: The url to track the job: http://mac127:8088/proxy/application\_1484320332413\_0279/  17/03/08 01:11:03 INFO mapreduce.Job: Running job: job\_1484320332413\_0279  17/03/08 01:11:11 INFO mapreduce.Job: Job job\_1484320332413\_0279 running in uber mode : false  17/03/08 01:11:11 INFO mapreduce.Job: map 0% reduce 0%  17/03/08 01:11:22 INFO mapreduce.Job: map 100% reduce 0%  17/03/08 01:11:31 INFO mapreduce.Job: map 100% reduce 50%  17/03/08 01:11:32 INFO mapreduce.Job: map 100% reduce 100%  17/03/08 01:11:33 INFO mapreduce.Job: Job job\_1484320332413\_0279 completed successfully  17/03/08 01:11:33 INFO mapreduce.Job: Counters: 49  File System Counters  FILE: Number of bytes read=142812  FILE: Number of bytes written=1020635  FILE: Number of read operations=0  FILE: Number of large read operations=0  FILE: Number of write operations=0  HDFS: Number of bytes read=61814  HDFS: Number of bytes written=1488618  HDFS: Number of read operations=12  HDFS: Number of large read operations=0  HDFS: Number of write operations=4  Job Counters  Launched map tasks=1  Launched reduce tasks=2  Data-local map tasks=1  Total time spent by all maps in occupied slots (ms)=8215  Total time spent by all reduces in occupied slots (ms)=12724  Total time spent by all map tasks (ms)=8215  Total time spent by all reduce tasks (ms)=12724  Total vcore-seconds taken by all map tasks=8215  Total vcore-seconds taken by all reduce tasks=12724  Total megabyte-seconds taken by all map tasks=8412160  Total megabyte-seconds taken by all reduce tasks=13029376  Map-Reduce Framework  Map input records=2679  Map output records=2679  Map output bytes=1655262  Map output materialized bytes=142804  Input split bytes=16706  Combine input records=0  Combine output records=0  Reduce input groups=1  Reduce shuffle bytes=142804  Reduce input records=2679  Reduce output records=2679  Spilled Records=5358  Shuffled Maps =2  Failed Shuffles=0  Merged Map outputs=2  GC time elapsed (ms)=136  CPU time spent (ms)=8730  Physical memory (bytes) snapshot=974127104  Virtual memory (bytes) snapshot=4806758400  Total committed heap usage (bytes)=1272971264  Shuffle Errors  BAD\_ID=0  CONNECTION=0  IO\_ERROR=0  WRONG\_LENGTH=0  WRONG\_MAP=0  WRONG\_REDUCE=0  File Input Format Counters  Bytes Read=0  File Output Format Counters  Bytes Written=0 |

Now check the dummy\_output table

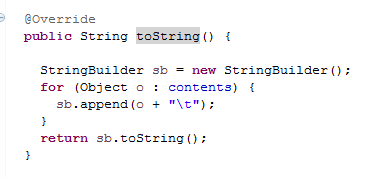


We can see data has been loaded, every time we run Hadoop job it will insert new data

We need not to worry about input format this program will work for all formats

# Hcatlog example - secondary Sort and delimiter

DefaultHcatRecord class use tab as delimiter to separate different column’s values



If we need to use some different delimiter other than tab we need to extends DefaultHcatRecord file and override toString method

|  |
| --- |
| package org.cyb.esplus.mapred;  //------------------------------------------------------------------------------  import java.util.List;  import org.apache.hive.hcatalog.data.DefaultHCatRecord;  //------------------------------------------------------------------------------  /\*\*  \*  \* Custom Hcat record which uses delimiter as a column separator  \*  \*/  public class CustomHcatRecord extends DefaultHCatRecord{    private List<Object> contents;    private String delimiter;  /\*\*  \* Constructor  \* @param size size of columns  \* @param delimiter column separator  \*/  public CustomHcatRecord(int size,String delimiter) {  super(size);  this.delimiter = delimiter;  }  //------------------------------------------------------------------------------  @Override  public String toString() {  contents = getAll();    StringBuilder sb = new StringBuilder();  for (Object o : contents) {  sb.append(o + delimiter);  }  return sb.toString();  }  }  //------------------------------------------------------------------------------  //  // End of file  //  //------------------------------------------------------------------------------ |

# Use Case

Input table structure

|  |
| --- |
| CREATE TABLE dummy\_input (  username string,  machinename string,  machineid string,  ipaddress string,  domainname string,  pcmlogeventid string,  modulename string,  filename string,  productname string,  companyname string,  description string,  text string,  activetime double,  datetime timestamp,  timeslot int,  date date) |

Output table structure

|  |
| --- |
| CREATE TABLE dummy\_output (  username string,  machinename string,  machineid string,  ipaddress string,  domainname string,  pcmlogeventid string,  modulename string,  filename string,  productname string,  companyname string,  description string,  text string,  starttime timestamp,  endtime timestamp,  activetime double,  timeslot int,  date date) |

If we don’t specify any field terminator while creating table, it is terminated by ‘\001’ (ctrl+A) separated

Find attached dataset for dummy\_input table



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UserName** | **Application** | **DateTime** | **Date** | **TimeSlot** |
| U1 | A1 | 2017-01-27 07:33:37.381 | 2017-01-27 | 7 |
| U1 | A2 | 2017-01-27 08:12:37.381 | 2017-01-27 | 8 |
| U1 | A3 | 2017-01-27 11:12:37.381 | 2017-01-27 | 11 |
| U1 | StopWindow | 2017-01-27 11:18:37.381 | 2017-01-27 | 11 |

Here we want to add dummy row where time slot changed and also calculate active time in output table

So the output table looks like as below

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **UserName** | **Application** | **startTime** | **endTime** | **ActiveTime** | **Date** | **TimeSlot** |
| U1 | A1 | 2017-01-27 07:33:37.381 | 2017-01-27 08:00:00.000 | 1582.619 | 2017-01-27 | 7 |
| U1 | A1 | 2017-01-27 08:00:00.000 | 2017-01-27 08:12:37.381 | 757.381 | 2017-01-27 | 8 |
| U1 | A2 | 2017-01-27 08:12:37.381 | 2017-01-27 09:00:00.000 | 2842.619 | 2017-01-27 | 8 |
| U1 | A2 | 2017-01-27 09:00:00.000 | 2017-01-27 10:00:00.000 | 3600.0 | 2017-01-27 | 9 |
| U1 | A2 | 2017-01-27 10:00:00.000 | 2017-01-27 11:00:00.000 | 3600.0 | 2017-01-27 | 10 |
| U1 | A2 | 2017-01-27 11:00:00.000 | 2017-01-27 11:12:37.381 | 757.381 | 2017-01-27 | 11 |
| U1 | A3 | 2017-01-27 11:12:37.381 | 2017-01-27 11:18:37.381 | 360.0 | 2017-01-27 | 11 |
| U1 | StopWindow | 2017-01-27 11:18:37.381 | 2017-01-27 11:18:37.381 | 0.0 | 2017-01-27 | 11 |

Here we need to use secondary sort, as we need to push data to reducer in descending order of dateTime

So we used **Sort Comparator Class**, **Grouping Comparator Class** and **Partitioner Class** to achieve this

Please follow below link to understand this concept

<http://www.bigdataspeak.com/2013/02/hadoop-how-to-do-secondary-sort-on_25.html>

In our logic we create dummy row if time slot change and get the end time from previous row , here all highlighted rows are dummy rows created by program

Please find attached below the actual program



We need to use below jars to run this program on cluster

* hive-exec-0.13.1-cdh5.2.0.jar
* hive-hcatalog-core-0.13.1-cdh5.2.0.jar
* hive-metastore-0.13.1-cdh5.2.0.jar
* libfb303-0.9.0.jar

To delete output table we need below jars ( we first truncate output table before inserting new data)

* hive-jdbc-1.1.0-cdh5.9.0.jar
* hive-service-0.13.1-cdh5.2.0.jar

Run below command after creating jar file

|  |
| --- |
| hadoop jar ActiveTimeHcatlog.jar org.cyb.esplus.mapred.ActiveTimeCalculationHcatlog dummy\_input dummy\_output |

# Secondary sort without using HcatLog

If we don’t want to use hive tables we can directly use the hdfs file to read from file and write to file in hdfs

Pleas find attached files for the same



|  |
| --- |
| hadoop jar ActiveTime.jar org.cyb.esplus.mapred.ActiveTimeCalculation /user/hive/warehouse/user\_activity\_tracking.db/dummy\_input /user/test/output |

And later we can store output file into hive table

|  |
| --- |
| #hive>load data inpath '/user/test/output overwrite into TABLE user\_activity\_tracking.dummy\_output; |

Please follow below link for more understanding

<http://riccomini.name/posts/hadoop/2009-11-13-sort-reducer-input-value-hadoop/>