Sequence File Format:-

(a) Sequence file format is one of the Hadoop specific file format and stores the serialized data hence the transferring of the file becomes very easy if stored in the sequence file format.

(b) Hadoop does not work very well with a lot of small files hence in order to solve this the sequence files are used as the container to store the small files and convert them into one large file.

(c) Sequence files are flat files containing key, value pairs.

(d) Every Sequence file has got a header which describes the about the file. The header contains information such as key/value class names, version, file format, metadata about the file and sync marker to denote the end of the header.

(e) Sequence file can be compressed so as to take up less space. This compression codec is also mentioned in the header.

(f) Sequence file format can be compressed in three ways

Uncompressed

Record compressed

Block compressed

.

Nline Input Format

It which splits N lines of input as one split.

job. set Input Format Class(N Line Input Format .class);

N Line Input Format. Add Input Path(job, input Path);

job.getConfiguration().setInt("mapreduce.input.lineinputformat.linespermap", 1);

Above lines can be used to set the how many lines are considered for one split . Here it is set to one. We can modify to any number one wants.

Hence fixed number of lines are sent to each mapper.

N Line Input Format is used in applications that take a small amount of input data and run an extensive (that is, CPU-intensive) computation for it, then emit their output.

Another example of n line input is seeding of the data from different sources.

We can create a “seed” input file that lists the data sources, one per line. Then each mapper is allocated a single data source, and it loads the data from that source into HDFS.

Default value is always set to one.

DB Input and output Format:-

Database input format allows Hadoop to input to output data to any relational database systems. Allowing relational data to be more easily incorporated into your data processing pipeline.

DB Input Format uses the JDBC to connect to the sources as it is widely used.

DB Input Format can work with MySQL, PostgreSQL, and several other database systems.

To start using DB Input Format to connect to your database, you’ll need to download the appropriate database.

These are the changes that are to be made in driver to use db input format

DB Configuration. configure DB(<Configuration>,<Driver>,<Connection String>,<User Name>,<Password>);

job. Set Input Format Class(DB Input Format. class);

DB Output Format. set Output(<Job>,<Output Table>,<List of table columns>);

Changes in mapper code:-

public class DB Input Format Map extends Mapper<Long Writable, DB Input Writable, Text, Null Writable>

protected void map(

Long Writable id,

DB Input Writable value, Context context)

Implementation of the Db input format:-

public void

read Fields(Result Set rs) throws SQL Exception

{

id = rs. get Int(1);

name = rs. get String(2);

location = rs. get String(3);

}

Similar modifications have to be done in Db output format:-

Driver code:-

DB Configuration. Configure DB(<Configuration>,<Driver>,<Connection String>,<User Name>,<Password>);

job. Set Input Format Class(DB Input Format. class);

DB Output Format. Set Output(<Job>,<Output Table>,<List of table columns>);

Mapper code:-

public class DB Output Format Map

extends Mapper<Long Writable, Text, DB Output Writable, Null Writable>protected void map(Long Writable id, Text value, Context context)

context. write(new DB Output Writable(Integer. parseInt (line Array[1]), line Array[2], line Array[3]), Null Writable. get())

Implementation of Db output format:-

public void write(Prepared Statement) throws SQL Exception

{

ps. set Int(1, id);

ps.setString(2, name);

ps.setString(3, location);

}