

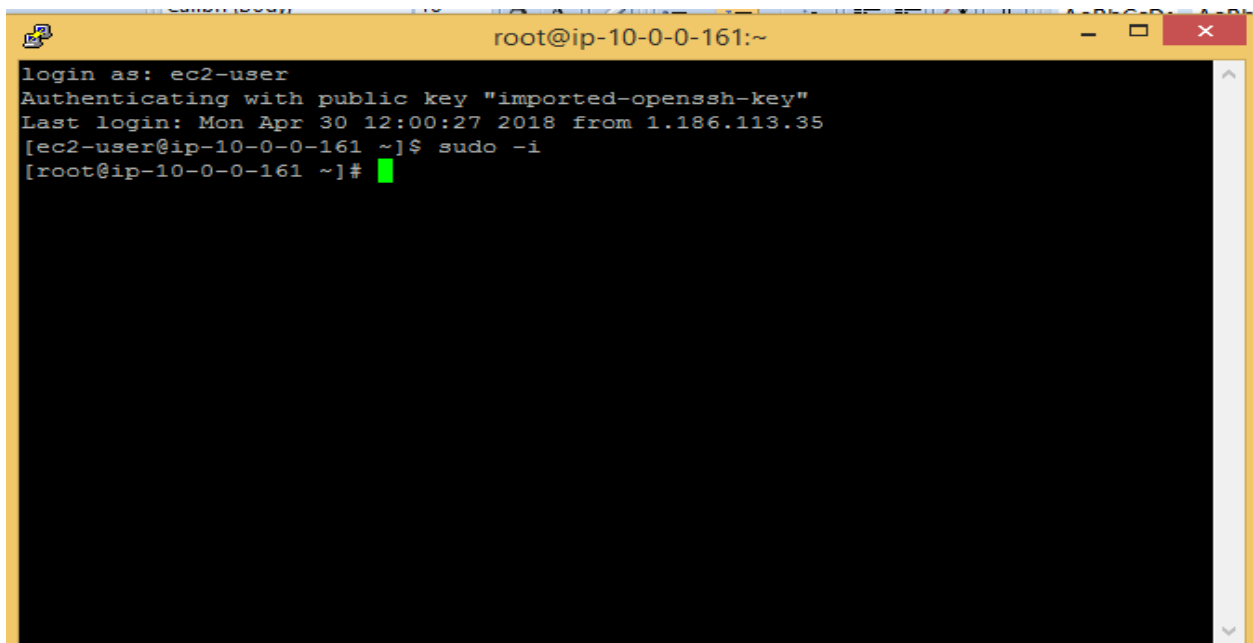
Saavn MapReduce Project

Files included in the project.

- Record.java
- RecordArrayWritable.java
- Utils.java
- saavnDriver.java
- saavnMapper.java
- saavnCombiner.java
- saavnPartitioner.java
- saavnreducer.java

Steps involved in arriving at trending songs are:

1. Exported project in jar file named saavnFinal.jar
2. Place saavnFinal.jar in S3 bucket under the folder saavn23042018.
3. Login into ec2 instance using root user as shown in the figure.



```
root@ip-10-0-0-161:~  
login as: ec2-user  
Authenticating with public key "imported-openssh-key"  
Last login: Mon Apr 30 12:00:27 2018 from 1.186.113.35  
[ec2-user@ip-10-0-0-161 ~]$ sudo -i  
[root@ip-10-0-0-161 ~]#
```

4. Check whether AWSCLI is properly installed and path is set. If path is not set, please set it as shown below.

```
[root@ip-10-0-0-161 ~]# aws
-bash: aws: command not found
[root@ip-10-0-0-161 ~]# alias aws='/usr/local/bin/aws'
[root@ip-10-0-0-161 ~]# aws s3 ls
2018-04-23 12:14:30 saavn23042018
[root@ip-10-0-0-161 ~]#
```

5. Then copy jar i.e. saavnFinal.jar into ec2 instance's root folder.

command: **aws s3 cp s3://saavn23042018/saavnFinal.jar /root**

6. Once jar file is copied, run below command.

command: **hadoop jar saavnFinal.jar com.trend.saavnDriver
s3a://mapreduce-bde/part-00000 s3a://saavn23042018/output**

File wise Program Logic

- Record.java

This file contains a Java object. Entity included in this is day of the month (day), weight of the song (weight), hour of the day at which song was played (hour).

- RecordArrayWritable.java

Created a custom ArrayWritable for Record Object. It is used to store array of Record object in distributed fashion. It contains getRecordArrayWritable() method to fetch the array of record Object.

- Utils.java

It contains a method sortByValues(Map) which takes map(songId as key and Total Count of song) as input. It sorts the map based on values in decending order. It is used to arrive at top 100 songs for each day.

- saavnDriver.java

Here we have mentioned configuration of the map reduce program.

```
job.setJarByClass(saavnDriver.class);  
job.setOutputKeyClass(Text.class);  
job.setOutputValueClass(Text.class);  
job.setMapOutputValueClass(RecordArrayWritable.class);  
job.setMapperClass(saavnMapper.class);  
job.setPartitionerClass(saavnPartioner.class);  
job.setCombinerClass(saavnCombiner.class);  
job.setReducerClass(saavnreducer.class);  
job.setNumReduceTasks(31);
```

- saavnMapper.java

Here Song's weight is calculated based on time window at which the song was played.

Classification of time window

```
window1 = 00 to 05  
window2 = 06 to 11  
window3 = 12 to 17  
window4 = 18 to 23
```

Based on hour at which song is played we calculate current and previous window.

calculation of current and previous window for each song

If song(say puUglc0M with hr 01) is played in 00-05 hr of 1st december. current window for that song is : puUglc0M01window1 (song+day+timeWindow in which song is played) previous window for that song is set to 'previousWindowNotAvailable' because previous day of 1st december is not available with us.

If song(puUglc0M) is played in 00-05 hr other than 1st December eg 2nd december current window for that song is : puUglc0M02window1 (song+day+timeWindow in which song is played) previous window for that song is : puUglc0M01window4(previous window for day 2 window1 will be day 1 window4)

Code :

```
if (0 <= hour && hour <= 05) {  
  
    window = song + "" + day + "window1";  
  
    if (day == 01) {  
  
        previousWindow = "previousWindowNotAvailable";  
  
    } else {  
  
        previousWindow = song + "" + daysList.get(previousDayIndex) +  
            "window4";  
  
    }  
  
}  
  
} else if (06 <= hour && hour <= 11) {  
  
    window = song + "" + day + "window2";  
  
    previousWindow = song + "" + day + "window1";  
  
} else if (12 <= hour && hour <= 17) {
```

```

    window = song + "" + day + "window3";

    previousWindow = song + "" + day + "window2";

} else if (18 <= hour && hour <= 23) {

    window = song + "" + day + "window4";

    previousWindow = song + "" + day + "window3";

}

```

Eg:

- (9yCMCTNq,baa758b4644c3ac51daa2a2384f4703d,1512143128,0
1,2013-12-01)
currentWindow : 9yCMCTNq01Window1(song+day+timeWindow)
previousWindow : previousWindowNotAvailablev
- (DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,0
1,2013-12-02)
currentWindow : DT8dMd7702Window1
previousWindow : DT8dMd7701Window4
- (DT8dMd77,baa758b4644c3ac51daa2a2384f4703d,1512143128,0
8,2013-12-02)
currentWindow : DT8dMd7702window2
previousWindow : DT8dMd7702window1
- (DT8dMd77,4408a264f37275e538bfe9fc30daae5c,1512143128,1
9,2013-12-01)
currentWindow : DT8dMd7701window4
previousWindow : DT8dMd7701window3

To achieve trendness of song we are manipulating weight for songs.

- 1) If a song(puUgIc0M) is present in previous window , song weight = previous window weight for a song(puUgIc0M) + 1 .
- 2) If a song is not present in previous window, song weight = 1.

This way we can calculate trending ie if song is present in previous window we increment weight for that song by 1.

If song is not present, song might not be trending or may be sudden spike but it may also be trending song but to analyse that we need to consider it next window.

Note: We are checking if song is present in previous window or not by using bloom Filters. We are only considering previous window.

Code:

```
        if (daysList.contains(day) &&
            !previousWindow.equalsIgnoreCase("previousWindowNot
Available")) &&
            bloomFilter.membershipTest(new
Key(previousWindow.getBytes())) {
    /**
    * In else weight is assigned to 1 because if false positive
    * occurs hashmap maynot contain value for that window.
    */
    if (map.containsKey(previousWindow))
        weight = map.get(previousWindow) + 1;
    else
```

```
weight = 1;
```

```
    if (!bloomFilter.membershipTest(new  
Key(window.getBytes())))  
        bloomFilter.add(new Key(window.getBytes()));  
        map.put(window, weight);  
        records[0] = new Record(day, weight, hour);  
        context.write(new Text(song), new  
RecordArrayWritable(records));  
} else if (daysList.contains(day) &&  
!bloomFilter.membershipTest(new  
Key(previousWindow.getBytes())) {  
    weight = 1;  
    map.put(window, weight);  
    records[0] = new Record(day, weight, hour);  
    bloomFilter.add(new Key(window.getBytes()));  
    context.write(new Text(song), new  
RecordArrayWritable(records));  
}
```

Example of songs present in current and previous window

	Current window	Previous window	
DT8dMd77,4408a264f37275e538bfe9fc30daae5c,1512143128,19,2013-12-01	DT8dMd7701window4	DT8dMd7701window3	1(since song is not present in previous window (ie DT8dMd7701window3)weight is 1)
DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,01,2013-12-02	DT8dMd7702window1	DT8dMd7701window4	1+1 (since song is present in previous window weight of song = weight of song in previous window(ie DT8dMd7701window4)+1)
DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,01,2013-12-02	DT8dMd7702window1	DT8dMd7701window4	1+1 (since song is present in previous window weight of song = weight of song in previous window(ie DT8dMd7701window4)+1)
DT8dMd77,baa758b4644c3ac51daa2a2384f4703d,1512143128,08,2013-12-02	DT8dMd7702window2	DT8dMd7702window1	2+1(since song is present in previous window weight of song = weight of song in previous window(ie DT8dMd7702window1)+1)
DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,08,2013-12-02	DT8dMd7702window2	DT8dMd7702window1	2+1(since song is present in previous window weight of song = weight of song in previous window(ie DT8dMd7702window1)

Example of songs not present in current and previous window

	Current window	Previous window	
V-HvGNCT,477552b6e41394619569fbfb9a590d5b,1512143128,19,2013-12-01	V-HvGNCT01window4	V-HvGNCT01window3	1(since song is not present in previous window (ie V-HvGNCT01window3)weight is 1)
V-HvGNCT,15657672f66cf3d80a8bf948fd094107,1512143128,08,2013-12-02	V-HvGNCT02window2	V-HvGNCT02window1	1(since song is not present in previous window (ie V-HvGNCT02window1)weight is 1)
V-HvGNCT,477552b6e41394619569fbfb9a590d5b,1512143128,08,2013-12-02	V-HvGNCT02window2	V-HvGNCT02window1	1(since song is not present in previous window (ie V-HvGNCT02window1)weight is 1)
V-HvGNCT,4408a264f37275e538bfe9fc30daae5c,1512143128,08,2013-12-02	V-HvGNCT02window2	V-HvGNCT02window1	1(since song is not present in previous window (ie V-HvGNCT02window1)weight is 1)

- **saavnCombiner** : Combiner is used to combine all the records based on songID.

structure of map: {songID,Record(day,weight,hour)}

1. V-HvGNct,477552b6e41394619569fbfb9a590d5b,1512143128,08,2013-12-02
2. V-HvGNct,15657672f66cf3d80a8bf948fd094107,1512143128,08,2013-12-02
- 3.V-HvGNct,4408a264f37275e538bfe9fc30daae5c,1512143128,08,2013-12-02

After Mapper

```
{ V-HvGNct,Record(02,1,08)}
{ V-HvGNct,Record(02,1,08)}
{ V-HvGNct,Record(02,1,08)}
```

After Combiner

```
[ V-HvGNct,{Record(02,1,08), Record(02,1,08), Record(02,1,08)}]
```

- **saavnPartioner.java** : It is used to decide in which Reducer task song will go to.

1. V-HvGNct,477552b6e41394619569fbfb9a590d5b,1512143128,08,2013-12-02
2. V-HvGNct,15657672f66cf3d80a8bf948fd094107,1512143128,08,2013-12-03

After Mapper

{ V-HvGNCT,Record(02,1,08)}

{ V-HvGNCT,Record(03,1,08)}

After Partitioner

{ V-HvGNCT,Record(02,1,08)} will be reduced in partitioner 1

{ V-HvGNCT,Record(03,1,08)} will be reduced in partitioner 2

- **saavnReducer.java**

It is used to calculate total count of a song at a particular day.

code:

```
String song = key.toString();
```

```
for (RecordArrayWritable val: values) {
```

```
    Writable[] records = val.getRecordArrayWritable();
```

```
    Record re = (Record)records[0];
```

```
    sum = sum + re.getWeight();
```

```
}
```

```
    map.put(song, sum);
```

eg : [DT8dMd77,{(02,2,01),(02,3,08),(02,3,08)}]

sum= 8

(DT8dMd77,8) is put into map for calculating trending songs. Utils class is to calculate trending songs.

Test Cases

1.

1	V-HvGNcT,477552b6e41394619569fbfb9a590d5b,1512143128,13,2013-12-01	↓
2	puUgIc0M,15657672f66cf3d80a8bf948fd094107,1512143128,13,2013-12-01	↓
3	9yCMCTNq,baa758b4644c3ac51daa2a2384f4703d,1512143128,13,2013-12-01	↓
4	DT8dMd77,4408a264f37275e538bfe9fc30daae5c,1512143128,13,2013-12-01	↓
5	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,13,2013-12-01	↓
6	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,19,2013-12-01	↓
7	I8E7jszF,8b5f6437ebfcea428a3db24e09d13ee0,1512143128,19,2013-12-01	↓
8	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,19,2013-12-01	↓
9	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,19,2013-12-01	↓
10	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,19,2013-12-01	↓
11	V-HvGNcT,477552b6e41394619569fbfb9a590d5b,1512143128,01,2013-12-02	↓
12	V-HvGNcT,15657672f66cf3d80a8bf948fd094107,1512143128,01,2013-12-02	↓
13	DT8dMd77,baa758b4644c3ac51daa2a2384f4703d,1512143128,01,2013-12-02	↓
14	V-HvGNcT,4408a264f37275e538bfe9fc30daae5c,1512143128,01,2013-12-02	↓
15	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,01,2013-12-02	↓
16	↓	
17	Day 1 output	↓
18	yZSd2-yD	5 1↓
19	DT8dMd77	5 2↓
20	V-HvGNcT	1 3↓
21	9yCMCTNq	1 4↓
22	puUgIc0M	1 5↓
23	I8E7jszF	1 6↓
24	↓	
25	Day 2 output	↓
26	DT8dMd77	6 1↓
27	V-HvGNcT	3 2←

2.

1	V-HvGNct,477552b6e41394619569fbfb9a590d5b,1512143128,01,2013-12-01↓
2	puUgIc0M,15657672f66cf3d80a8bf948fd094107,1512143128,01,2013-12-01↓
3	9yCMCTNq,baa758b4644c3ac51daa2a2384f4703d,1512143128,01,2013-12-01↓
4	DT8dMd77,4408a264f37275e538bfe9fc30daae5c,1512143128,01,2013-12-01↓
5	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,01,2013-12-01↓
6	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,07,2013-12-01↓
7	I8E7jszF,8b5f6437ebfcea428a3db24e09d13ee0,1512143128,07,2013-12-01↓
8	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,07,2013-12-01↓
9	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,07,2013-12-01↓
10	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,07,2013-12-01↓
11	V-HvGNct,477552b6e41394619569fbfb9a590d5b,1512143128,13,2013-12-01↓
12	V-HvGNct,15657672f66cf3d80a8bf948fd094107,1512143128,13,2013-12-01↓
13	DT8dMd77,baa758b4644c3ac51daa2a2384f4703d,1512143128,13,2013-12-01↓
14	V-HvGNct,4408a264f37275e538bfe9fc30daae5c,1512143128,13,2013-12-01↓
15	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,13,2013-12-01↓
16	↓
17	Day 1 output↓
18	↓
19	DT8dMd77 11 1↓
20	yZSd2-yD 5 2↓
21	V-HvGNct 4 3↓
22	9yCMCTNq 1 4↓
23	puUgIc0M 1 5↓
24	I8E7jszF 1 6←

3.

1	V-HvGNct,477552b6e41394619569fbfb9a590d5b,1512143128,19,2013-12-01↓
2	puUgIc0M,15657672f66cf3d80a8bf948fd094107,1512143128,19,2013-12-01↓
3	9yCMCTNq,baa758b4644c3ac51daa2a2384f4703d,1512143128,19,2013-12-01↓
4	DT8dMd77,4408a264f37275e538bfe9fc30daae5c,1512143128,19,2013-12-01↓
5	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,19,2013-12-01↓
6	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,01,2013-12-02↓
7	I8E7jszF,8b5f6437ebfcea428a3db24e09d13ee0,1512143128,01,2013-12-02↓
8	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,01,2013-12-02↓
9	yZSd2-yD,e68fbc2d1137d2041b69c255b4c2ebf3,1512143128,01,2013-12-02↓
10	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,01,2013-12-02↓
11	V-HvGNct,477552b6e41394619569fbfb9a590d5b,1512143128,08,2013-12-02↓
12	V-HvGNct,15657672f66cf3d80a8bf948fd094107,1512143128,08,2013-12-02↓
13	DT8dMd77,baa758b4644c3ac51daa2a2384f4703d,1512143128,08,2013-12-02↓
14	V-HvGNct,4408a264f37275e538bfe9fc30daae5c,1512143128,08,2013-12-02↓
15	DT8dMd77,2c237ae19cab0d7503a863d733f446f5,1512143128,08,2013-12-02↓
16	↓
17	day 1 output↓
18	yZSd2-yD 1 1↓
19	DT8dMd77 1 2↓
20	V-HvGNct 1 3↓
21	9yCMCTNq 1 4↓
22	puUgIc0M 1 5↓
23	↓
24	day 2 output↓
25	DT8dMd77 10 1↓
26	yZSd2-yD 4 2↓
27	V-HvGNct 3 3↓
28	I8E7jszF 1 4←