I completed the pyspark code & adopted the approach using Scala. Below is the data flow between each stages in pyspark code,

1. Loaded the movies\_title(movie\_titles.txt) data in movie\_ref\_rdd & training set(mv\_000\*.txt) data in movies\_rdd

* movies\_rdd : List(File\_Content) => (“1:\n10,4,2008-10\n..”, “2: :\n10,100,2008-10\n..”,”)
* movie\_ref\_rdd : List(row) => ( (1, 2003, Dinosar\_planet), .. )

1. Created a parser to parse the text file contents from movies\_rdd & store it in parse\_csv\_rdd.

* Used a flatmap to emit each row from the file. Emits data as (movie\_id, user\_id, rating, rating\_dt)
* parse\_csv\_rdd : List( (movie\_id, user\_id, rating, rating\_dt) ) => ( (1, 10, 4, 2008-10), …)

1. Created two rdd’s that gets the input from parse\_csv\_rdd & emit the values by user\_id & movie\_id. The respective map functions are grouped by keys.

* User\_mapper\_func.groupByKey = > (movie\_id, (movie\_id, user\_id, rating, rating\_dt) )
* Movie\_mapper\_func.groupByKey => (user\_id, (movie\_id, user\_id, rating, rating\_dt) )

1. Joined the movie\_mapper\_rdd with the movie\_title\_reference\_rdd. It will get all the related movie information.

* Movie\_mapper\_rdd.join(movie\_reference\_rdd)
* Joined\_rdd = List(movie\_name, List(user\_ratings), List(movie\_facts)). => (1, ((10, 5, 2018-01), …), ((1, “Dinosar planet”, 1992)) )

1. Created the mapper function which reduces the above joined\_rdd to ( movie\_id, movie\_name, movie\_publication, ratings\_count )

* Emit ((1, “Dinosar Planet”, “1992”, 100020), …)

1. Created a custom sort function to sort the movies list by (ratings\_count.desc, publication.desc, name..asc)
2. Got the top M movies which is rated by atleast R users & stored it in a broadcast value.

* M\_movies = Broadcast((“10”,”20303”))

1. Created a user\_reducer\_rdd which takes the input from the input user\_mapper\_rdd & emits the user with it’s attributes. I used the broadcast value to get the average rating given to m\_movies.

* List(row) => (user\_id, highest\_rated\_films: list, m\_movies\_avg\_rating, intersect\_count)

1. Created an utility function to get the top U users, who has given least average rating for top M movies.

Assumptions:

* Top M movies : Get the top movies which has highest number of ratings, which is rated by atleast R users
* Top U users: Who has given least average ratings for M movies
* Highest ranked film by each user is determined by latest publication date and movie name

**Results:**

**M = 5, U = 25, R = 50**

|  |  |  |  |
| --- | --- | --- | --- |
| user\_id | movie | year\_release | rating\_dt |
| 61369 | Hotel Rwanda | 2005 | 2005-06-02 |
| 61884 | A Very Long Engagement | 2004 | 2005-02-24 |
| 428168 | Crash | 2005 | 2005-11-23 |
| 491149 | Family Guy: Freakin' Sweet Collection | 2004 | 2005-06-01 |
| 595870 | Lord of the Rings: The Fellowship of the Ring | 2001 | 2004-11-21 |
| 1134871 | The Dinner Game | 1998 | 2004-04-14 |
| 1511683 | 11:14 | 2005 | 2005-11-30 |
| 2103916 | Batman Begins | 2005 | 2005-10-24 |
| 2632868 | Fahrenheit 9/11 | 2004 | 2004-11-25 |
| 157179 | Agatha Christie's Poirot: Death on the Nile | 2004 | 2005-11-29 |
| 455481 | Before Sunset | 2004 | 2005-02-18 |
| 904816 | Arrested Development: Season 2 | 2004 | 2005-11-18 |
| 1171240 | Ab Tak Chhappan | 2004 | 2005-05-06 |
| 1389855 | Irreversible | 2002 | 2004-10-27 |
| 1481664 | Eternal Sunshine of the Spotless Mind | 2004 | 2005-01-13 |
| 1648411 | Assisted Living | 2005 | 2005-09-28 |
| 1716826 | The Incredibles | 2004 | 2005-09-22 |
| 1805653 | Arrested Development: Season 2 | 2004 | 2005-11-05 |
| 2194694 | A Very Long Engagement | 2004 | 2005-09-06 |
| 2421961 | Kill Bill: Vol. 2 | 2004 | 2004-12-27 |
| 2607573 | Lost in Translation | 2003 | 2005-08-09 |
| 101554 | Eternal Sunshine of the Spotless Mind | 2004 | 2005-02-18 |
| 111085 | The Looney Tunes Golden Collection: Vol. 3 | 2005 | 2005-10-17 |
| 172264 | Curb Your Enthusiasm: Season 4 | 2005 | 2005-11-05 |
| 213486 | Eternal Sunshine of the Spotless Mind | 2004 | 2005-03-18 |