|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Workload | Implementation | Programming Language |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Category and Trending Correlation | Spark | Python |  |
|  |  |  |  |  |
|  | Impact of Trending on View Number | Map Reduce | Java |  |
|  |  |  |  |  |

# Workload: Category and Trending Correlation

The sequence of transformations and actions are illustrated in Figure [1.](#page1)

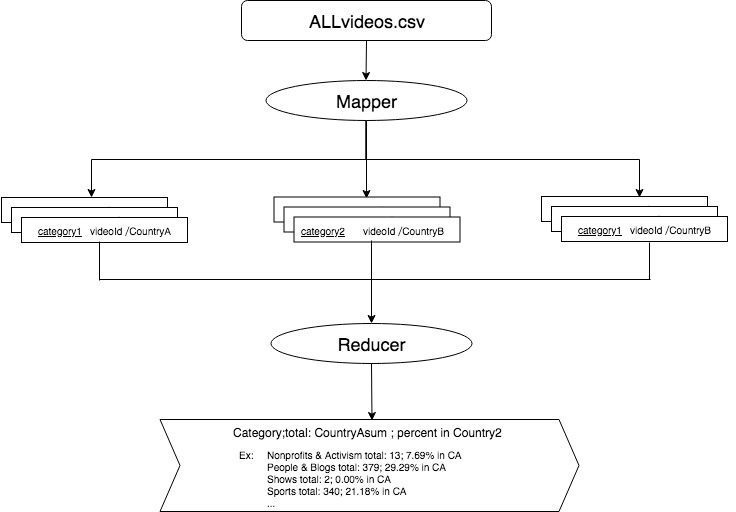


Figure 1: MapReduce phase for the workload.

**In map phase**, the ALLvideos.csv was read in and split by regular expression to and filtered with the country contains CountryA and CountryB which were given in the configuration. I took each pair with category as key and country/videoId as value. So the pair is formatted as (category,country/videoed) which can be processed by reducer later.

**In reducer phase**,I created two ArrayList to store the country data. The Hashset is used to remove the duplicated country record so that we can get the true number of the times that the country appeared in the record. Then I counted the record containing the times category with first country also appeared in second country. Then do the division method to get the percent. The last two step is combing the counted times and percentage in second country as value. Then set category as key to get the output result.

## Parallelization

VideoMapper and VideoReducer can process data in parallel. VideoMappers run in parallel on partition of the data and **set “category” as key**. I have **set 3 reducers** to run the job. They run in parallel on **aggregation of the intermediate results with same key** and **count the percent comparing the country**. Each partition handles data related with a **subset of categories**

# Workload: Impact of Trending on View Number

The sequence of transformations and actions are illustrated in Figure [2.](#page1)

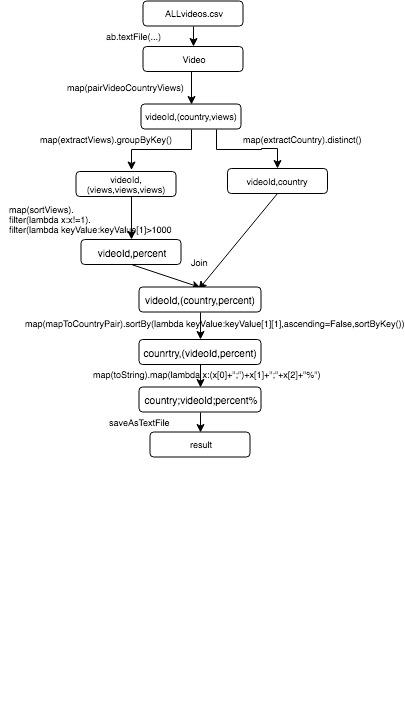


Figure 2: Spark for the workload

The **core** idea of this solution is extract the views from **(videoId,(country, views)**pair to **calculate the percentage** and then join the **(videoId, country)** pair and then transform the **(videoId,(country, percent))** into **(country,(videoId, percent)**.

First step is reading in ALLvideos.csv to create Video RDD then using map to convert Video RDD into (videoId, (country ,views)) pairRDD. “SortViews” function is used to extract the first two views number to do the division method and multiple 100 to get the percentage. Filter the (percent >1000) to fit the condition of the assignment. The distinct() function is used to remove the duplicated (videoId, country) pairRDD. The final step is sort the percent and format output. Saved result as text file.

## Parallelization

The ALLvideos.csv is read in then parallelize using the built-in SparkContent parallelize method. The **map** and **maptoPair** operations can run in parallel on different **partitions of the (videoId,(country, view))and (videoId, view)**. The **join** operations can run in parallel to **get (videoId,(country,view))pair**. Shuffling is required during the join operation. The **groupByKey** transformation can run in parallel and pipeline with the **group of the key:videoId.**