COMP5349 Assignment 1
Name: Yuanqi Pang
SID: ypan9830
Tutor Name: Andrain Yang

Workload	Implementation	Programming Language
Category and Trending Correlation	Hadoop	Java
Impact of Trending on View Number	Spark	Java

Workload: Category and Trending Correlation

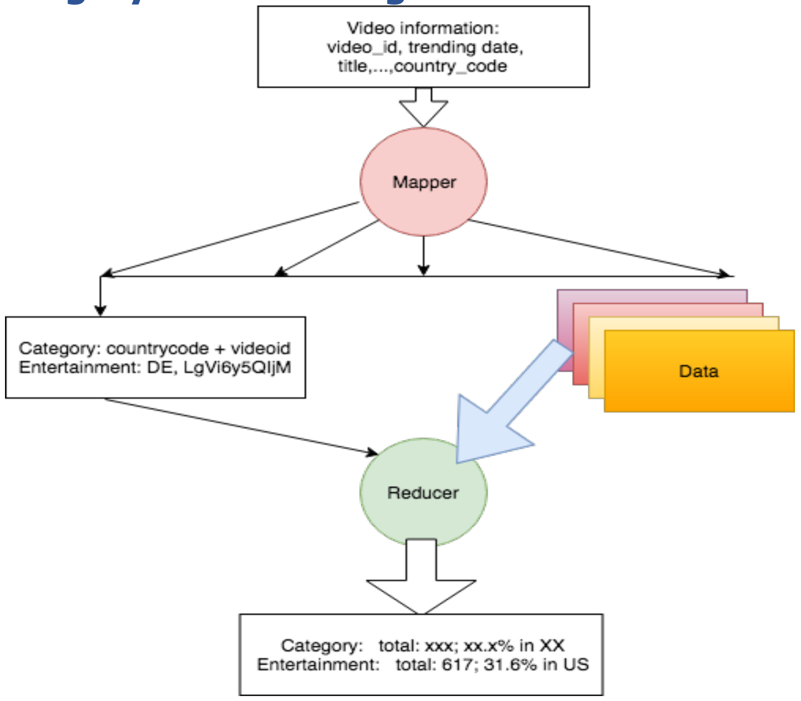

Video information:
video_id, trending date,
title,...,country_code
Mapper
Category: countrycode + videoid
Entertainment: DE, LgVi6y5QIjM
Data
Reducer
Category: total: xxx; xx.x% in XX
Entertainment: total: 617; 31.6% in US

Figure 1: Map Reduce framework.

The sequence of Mapping and Reducing are illustrated in Figure 1. The first part is done by using Hadoop mapreduce, and the process is controlled by the driver class. First the data file is read line by line via the mapper, only the data within the 2 input countries is chosen by the mapper and send to the reducer class. The detailed input and output format is shown in the figure 1.
In the reduce part, data of 2 countries are separated and stored in 2 HashMap. The Reducer is key-oriented, so the class is to handle every single category. The data contains repeated values of different trending date, which will affect the counting of category size, the repeated data is removed by using HashMap containsKey method to judge. Thus, the size of the first HashMap is the total number of video of country A in this category. The percentage of the videos in another country can be calculated by checking and comparing each element in 2 HashMap. The total number and the percentage is formatted and sent to the output.

Parallelization:

Both the mapper and reducer can be parallelized, and if done in parallel. Data can be divided and mapped in parallel. The number of reducer can be set in driver class, the reducer will run in parallel on different categories and the result will be shown in multiple files.

COMP5349 Assignment 1
Name: Yuanqi Pang
SID: ypan9830
Tutor Name: Andrain Yang

Workload: Impact of Trending on View Number

The algorithm design is indicated in figure 2, which including all the specified input and output format through all function. All functions used in this project is the spark build-in function. After the group by key function, trending views that has the same video id and country are grouped together. The data used in this project is ordered by trending date, so it is unnecessary to consider the sorting of trending date, the quantity of first and second trending views is selected to calculate the percentage. The increase percentage is swapped to be the key and the data was sorted by the percentage. After another swap, the key value pair was remapped, the country code is the new key and the value is the video id and percentage. The dataset is sorted again by country to make the data of same country together.
Different kind of algorithm is tried to improve efficiency, when combining some functions together in one function, the project will run faster in local Spark while slower using Spark cluster. For example, if convert the pairRDD into ArrayList after the calculation of increment percentage, then sort the ArrayList by customized comparator, the procedure cannot be parallelized, thus improve the efficiency in local and lose efficiency on cluster. The algorithm is designed to have best parallelization and efficiency on cluster.

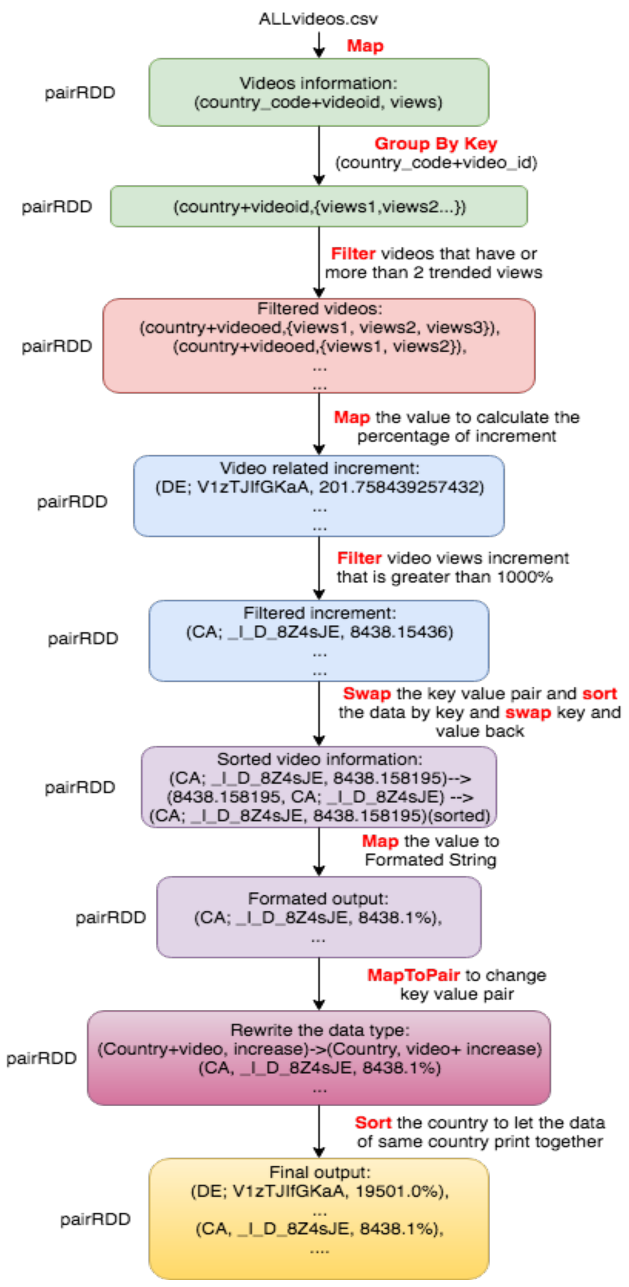
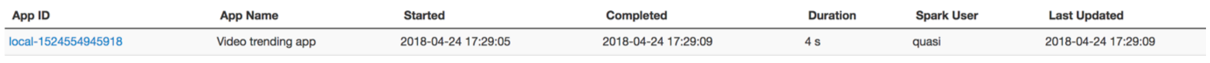

ALLvideos.csv
Map
pairRDD
Videos information: (country_code+videoid, views)
Group By Key (country_code+video_id)
pairRDD
(country+videoid,{views1,views2...})
Filter videos that have or more than 2 trended views
pairRDD
Filtered videos: (country+videoed,{views1, views2, views3}), (country+videoed,{views1, views2}), ...
Map the value to calculate the percentage of increment
pairRDD
Video related increment: (DE; V1zTJIfGKaA, 201.758439257432) ... ...
Filter video views increment that is greater than 1000%
pairRDD
Filtered increment: (CA; _I_D_8Z4sJE, 8438.15436) ... ...
Swap the key value pair and sort the data by key and swap key and value back
pairRDD
Sorted video information: (CA; _I_D_8Z4sJE, 8438.158195)--> (8438.158195, CA; _I_D_8Z4sJE) --> (CA; _I_D_8Z4sJE, 8438.158195)(sorted)
Map the value to Formated String
pairRDD
Formated output: (CA; _I_D_8Z4sJE, 8438.1%), ...
MapToPair to change key value pair
pairRDD
Rewrite the data type: (Country+video, increase)->(Country, video+ increase) (CA, _I_D_8Z4sJE, 8438.1%) ...
Sort the country to let the data of same country print together
pairRDD
Final output: (DE; V1zTJIfGKaA, 19501.0%), ... (CA, _I_D_8Z4sJE, 8438.1%), ...

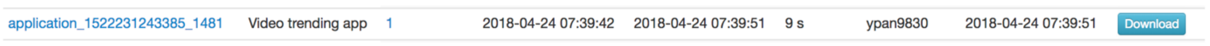
Figure 2. Spark framework Design

Parallelization

All the spark build-in function can be run in parallel. Thus all the map function can be executed in parallel. The sort function cannot be parallelized as all the data need to be compared together, the dataset at sort phase is not big so it will not affect the running time much. All other functions can be parallelized and data can be partitioned.



App ID	App Name	Started	Completed	Duration	Spark User	Last Updated
local-1524554945918	Video trending app	2018-04-24 17:29:05	2018-04-24 17:29:09	4 s	quasi	2018-04-24 17:29:09



application_1522231243385_1481	Video trending app	1	2018-04-24 07:39:42	2018-04-24 07:39:51	9 s	ypan9830	2018-04-24 07:39:51	Download