Workload	Implementation	Programming Language
Category and Trending Correlation	MapReduce	Python
Impact of trending on view numbers	Spark	Python

Workload: Category and Trending Correlation

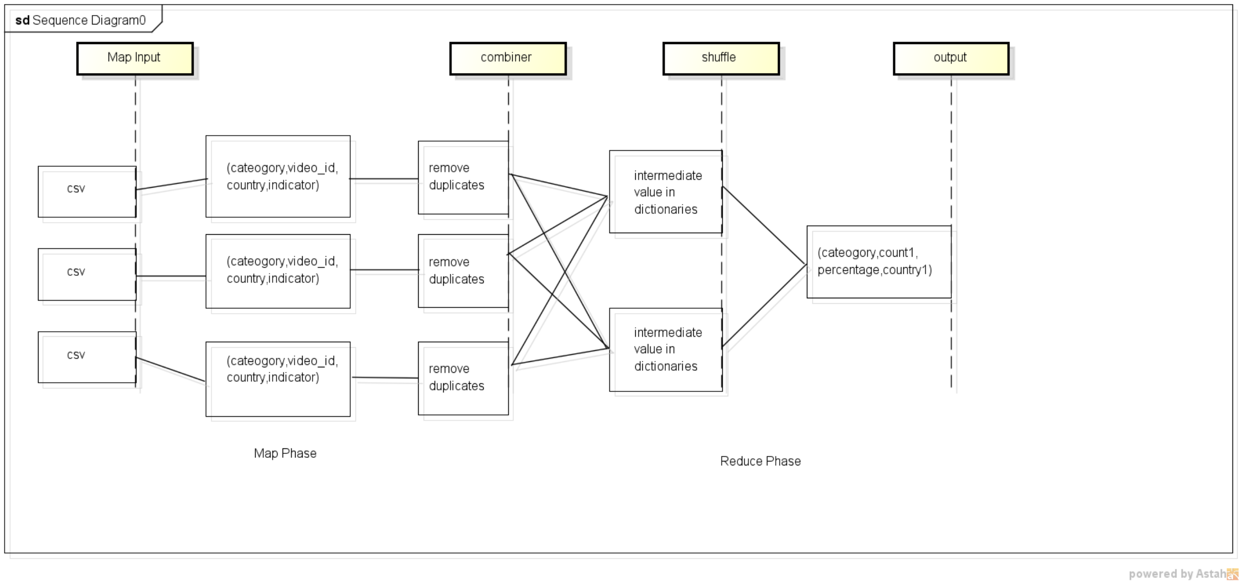
MapReduce is chosen to deal with the questions. In the map phase, when it reads the first row of the file, it will automatically jump to the second row because first row contains header only. Starting from second row, 4 pieces of information is derived from file: category, video_id,country as well as indicator. If the corresponding country value equals c1, which is the argument input for the first country, the indicator is 1. The same rule applies to the second country. The reason being that indicator can be used to sort records into its own dictionaries during the reduce phase in order to facilitate the calculation. The output from the map phase will be formatted as (category, video_id,country as well as indicator).

Combiner is also considered to minimize the input to reduce phase by removing the duplicate records. This is achieved by using country plus video id to search in the list created in the combiner function, if there is no hit, the combiner will output the results, otherwise it will jump to the next record, denoting a detection of duplicate records.

In the reducer phase, two dictionaries are created for each country respectively. If the indicator shows 1, the record will be inserted to dictionary 1 with key value pair of video id and category. The same principle applies to indicators of 2. The next step is to find from dictionary 1, how many records actually overlap with dictionary 2, the result will be stored in the dictionary same with key value pair of category and its corresponding counts. Then for country 1, dictionary total is created to calculate the total number for each category in country 1. The last stage is to find for each category in the overlap dictionary, the corresponding count for dictionary total will be count 1 and the count for dictionary same will be count 2, the answer will simply be count2/count1 and eventually print the output based on the required format.

Parallelization

The map phase can be run concurrently on different partitions to get the output. Each partition will individually perform the mapping process to extract the category, video id, country and indicator as output. Similarly, the combiner function is expected to run in parallel as well when it comes to eliminating duplicates. In the reduce phase, the output from the combiner will be further processed on different partitions as well, this includes inserting processed data into the dictionary and perform the calculations.

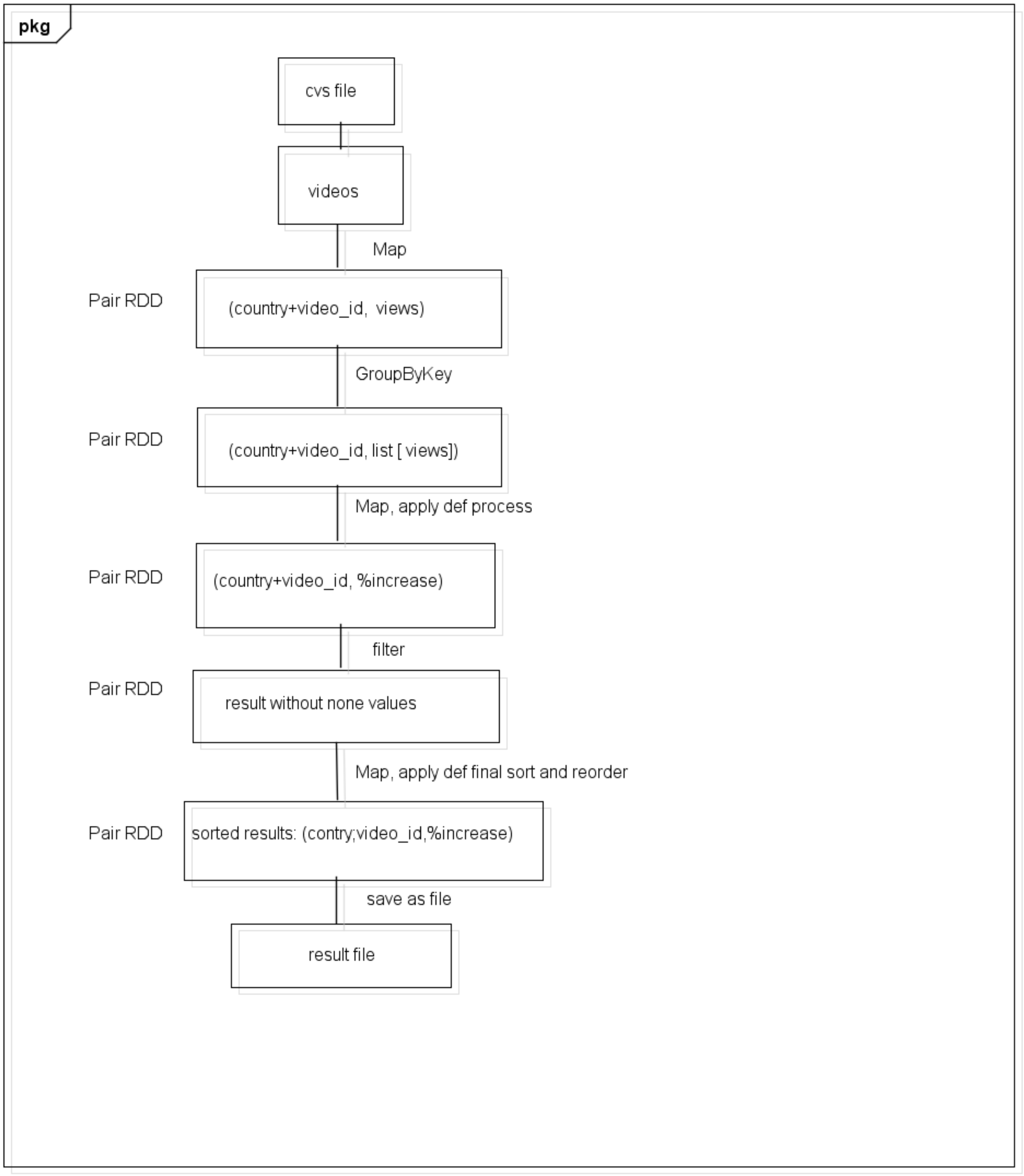

sd Sequence Diagram0
Map Input
combiner
shuffle
output
csv
csv
csv
(cateogory,video_id, country,indicator)
(cateogory,video_id, country,indicator)
(cateogory,video_id, country,indicator)
remove duplicates
remove duplicates
remove duplicates
intermediate value in dictionaries
intermediate value in dictionaries
(cateogory,count1, percentage,country1)
Map Phase
Reduce Phase
powered by Astah

Workload: Impact of Trending on view numbers

Spark is used to address the issue. First, the input file will be mapped as RDD pairs (country + video_id, views). The reason why trending date is not considered is because based on my observations, the views provided in the original file is accumulated views, which means when country+video_id is equal, the more views, the later trending date will be. Then the RDD pairs will be grouped by the key (country+video_id), and the result will be like key plus a list of views under that particular key. The process function is used to sort the views in each list and only return the output when the percentage increase between the smallest views (first trending date) and second smallest views (second trending date) is greater than 1000%. The result will be further filtered in order to remove none values. The filtered result will be sorted based on country plus views and grouped by country, sorting in decreasing order based on views for each country. This is achieved by applying final sort, sortByKey as well as reorder functions. Finally, the results will be saved in output file in given output path.

Parallelization

Many crucial steps can be done in parallel, for example, in the first step, when constructing the RDD pairs (country + video_id, views), the work is done on different partitions. Similarly, when reformatting the RDD pairs, conducting the sorting and performing calculations, as well as regrouping and sorting the result, the jobs are all done concurrently. As long as map is used, the work is considered to be done in parallel.


pkg
cvs file
videos
Map
Pair RDD
(country+video_id, views)
GroupByKey
Pair RDD
(country+video_id, list [views])
Map, apply def process
Pair RDD
(country+video_id, %increase)
filter
Pair RDD
result without none values
Map, apply def final sort and reorder
Pair RDD
sorted results: (contry;video_id,%increase)
save as file
result file