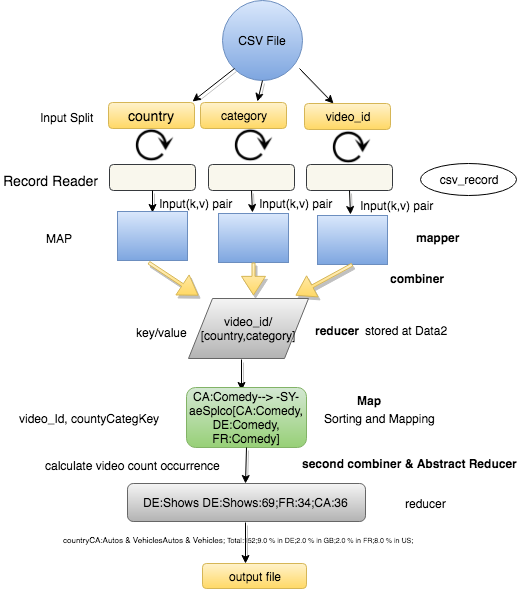
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Workload | Implementation | Programming Language |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Category and Trending Correlation | Map Reduce | Java |  |
|  |  |  |  |  |
|  | Impact of Trending on View Number | Spark | Java |  |
|  |  |  |  |  |

# Workload: Category and Trending Correlation



**The flow of transformation and action are illustrated in Figure1**

In this workload the Allvideos.csv is read and and filtered to redirect to first mapper class and mapped to generate a key value pair (video\_id ,category and country) depending on the number of key’s value in CSV file “ALLvideos.csv” where key is generated as video\_id and countyCategkey

Further combiner is implemented to extract unique value of Category list with respect to particular country key and in further processing the at Reducer video\_id is consider as key and respective occurrence of that video\_id with rescpect to all the country with respective value category is saved as value and output file generated as Data2 in cluster for cycle1 Job in this form -0NhqVYR4UY [DE~News & Politics, CA~News & Politics]”.

Further, the cycle1 output file in Data1 is consider as input for mapping in cycle2 job where the first splitting and filtering done on “\t” and then in next step between the value of Country Category list array.

In next iteration the output of map2 is saved as in form “CA~ Entertainment--> 4CGBSGEkbK [CA~Entertainment,

US~Entertainment, DE~Entertainment] “array list value form.

In cycle 2 Mapper matches maps each category to the full record of that video id occurrences in different countries, then reducer then finds for a given category in country how many videos occurred in other countries and finds the final result.

After second mapping processor, Abstract Reducer which is used to execute the count result with respect the count of total unique occurrences in one particular category in one country with respect to the other country and then the result is forwarded to reducer as a combined summary count of distinct trending video in each of that particular country look like this “CA: Education:384; DE:35; GB:11; FR:42; US:66;”.

The output can be displayed in terminal and can be forwarded to Hadoop cluster with Assignment1.jar and script which is generated from maven installed execution.

Output is saved with final name as part-r-00000 text file.

**Ex :- CA:Autos & Vehicles Autos & Vehicles;Total:152; 9.0 % in DE; 2.0 % in GB; 2.0 % in FR; 8.0 % in US;**

**Summary of Code implementation if above explanation confusing: -**

1- Program is divided in two cycles

2- CYCLE ONE:

Map: reads input and connects videoId country & category together.

Reduce: brings all country & category information together for a video\_id .

3- CYCLE TWO:

Mapper: maps each category to the full record of that video id occurrences in different countries.

Reducer: then finds for a given category in country how many videos occurred in other countries and finds the final result.

4. cycle1 input=files from hdfs output=input of cycle 2 , cycle 2 in=output of cycle 1 , cycle 2 output to hdfs

## **Parallelization**

Initially the input split is done parallel when considering all the key value to create a mapped record after processing from Allvideos.csv. In sequence to that the all key value pair operation performed in mapper and reducer is performed in parallel. As each cycle job is dependent on the output of first where combiner has input from mapper and similarly reducer have input from combiner, so interdependency reduces the scope of parallelism. But, Map and Reduce does perform their own internal parallelism with data and operation when task in executed in yarn cluster.

# Workload: Impact of Trending on View Number

In this workload the CSV loaded with Spark Context in Spark Driver. After the job is launched Allvidoes.csv file read by spark and further process is initiated. Spark Configuration is defined to define the app name and set master which can relate to spark-submit command properties attacking to the configuration flag.

In next step Spark Session in defined spark session builder to define conf and with enableHiveSupport to enable Hive support, including connectivity to a persistent Hive metastore, support for Hive serdes, and Hive user-defined functions. In next step, User-Defined Functions (aka UDF) is a feature of Spark SQL to define new Column-based functions that extend the vocabulary of Spark SQL’s DSL for transforming Datasets. This function is used to implement or calculate percentage increase in two trending dates.

In Filtering step Spark read and format function used to read Allvioe.csv file from input path and splitting on the basis of delimiter (“,”) and further select function is used to iterate and filter the relevant field from the input dataset. A Dataset is a strongly typed collection of domain-specific objects that can be transformed in parallel using functional or relational operations and this is the reason the Spark Dataset is used for second workload to achieve parallelism. The Key rows such as “video\_id", "country", "views", "trending date which are required for the fulfilment of workload 2.

Then, trending date column is reformatted using cast function to create new column data set. It then partitions the dataset using video Id for partitioning a formula was used to multiply available executor\*cores \* 20 to achieve parallelism. Then the spark SQL command is implemented. Then a windows function rank was called on trending date over video id & country this would order all trending dates and assign the row rank.

Then we created 2 views from the above dataset namely t1 & t2. In the next step, t1 & t2 were inner joined over video\_id & country new table/dataset looks like t1 (video ID, country, first day views, second day views) condition is represented below: -

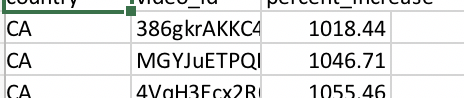
**Representation of PSEUDO query look like this: -**

SELECT t1. video\_id, t1.country, t1.views as first day views, t2.views as second day views from t1 JOIN t2 ON t1.video\_id=t2.video\_id AND t1.country =t2.country WHERE t1.rank=1 AND t2.rank=2.

In next step, a user defined function called calculate Percent Increase in SQL to calculate percentage increase in first two days this function takes first\_day\_view & second\_day\_views as input this would give the dataset with columns video\_id, country, percentage increase.

CalculatePercentIncrease.java class is integrated in work set to evaluate percentage calculation.

Finally, simple filter to select & display for rows with more than 1000 percent increase also it is saved in csv.



## **Parallelization**

Spark partitioning is used to achieve parallelism, so we can divide the load of CSV column with respect to video ID in the cluster into different nodes and cores, so parallelism and efficiency can be achieved. Number of partition is equal to number of cores in the cluster by this all partition will process parallel and resources are also used equally*.*

**Implementation for Both worload**

In both Spark and Map Reduce the unnecessary data has been removed at mapping phase itself. In design variation python is used to implement as requirement of workload for work load with Hadoop 3.0.0. and Spark x2.3.0 with to initiate spark module. Both framework have different design and implementation and chosen as per workload requirements in project.

**Filtering: -** In both the workload with map reduce and spark implementation irrelevant data has been filtered out in mapping and selecting process during the first stage of processing itself the data is filtered.

**Design Variation:**

**Map Reduce:** In this particular designing phase of Map Reduce the data has been filtered in during the first cycle/job mapper stage. The shuffling is reduced as less data is been processed in the next stages of map reduce.

**Spark:** In this a particular design phase DAG data flow is more concentrated towards the parallel process of partitioning where different nodes are allotted to process data in parallel to reduce a lot shuffling I/O read write and increasing the efficiency of algorithm.

**Reference:**

* <https://coe4bd.github.io/HadoopHowTo/multipleJobsSingle/multipleJobsSingle.html>
* <https://spark.apache.org/docs/2.1.0/api/java/>
* https://stackoverflow.com/