THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - INDIVIDUAL ASSESSMENT

Unit of Study:	COMP5349 Cloud Computing
Assignment name:	Assignment 1: Simple Data Analysis with MapReduce and Spark
Tutorial time:	Thursday 4:00-6:00pm
Tutor name:	Andrian Yang

DECLARATION

Student ID:	470161133
Student name:	Dongdong Zhang
Signed	[illegible]
Date	26/04/2018

SIT Building, J12
The University of Sydney
NSW 2006 Australia

T +61 2 9351 3423
F +61 2 9351 3838
E sit.info@sydney.edu.au
sydney.edu.au/it

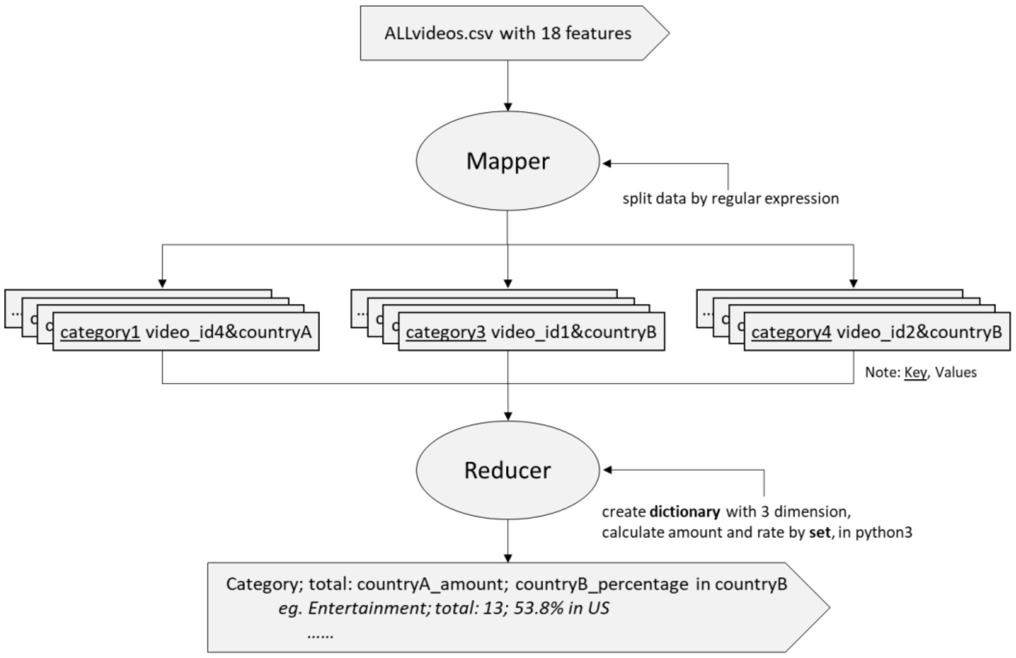
ABN 15 211 513 464
CRICOS 00026A

COMP5349 Assignment 1
Name: Dongdong Zhang
SID: 470161133
Tutor Name: Andrian Yang

Workload	Implementation	Programming Language
Category and Trending Correlation	MapReduce	Python3
Impact of Trending on View Number	Spark	Python3

Workload: Category and Trending Correlation

This workload is for question 1, and is completed under MapReduce with Python3. The sequence of transformations and actions are illustrated in Figure 1.



ALLvideos.csv with 18 features
Mapper
split data by regular expression
category1 video_id4&countryA
category3 video_id1&countryB
category4 video_id2&countryB
Note: Key, Values
Reducer
create dictionary with 3 dimension, calculate amount and rate by set, in python3
Category; total: countryA_amount; countryB_percentage in countryB
eg. Entertainment; total: 13; 53.8% in US
......

Figure 1: MapReduce phase for the workload

One MapReduce job is used for this workload. It consists of two phases: Map then Reduce, the Figure 1 shows the main structure of the system.

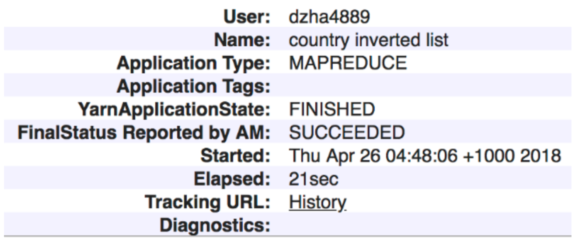
About the mapper, for each input row from ALLvideos.csv with 18 features, the mapper breaks the line data using regular expression, and only selects the lines whose country is either country A or country B, then yields the key is category, and values are video_id and country.

About the reducer, it received output from mapper sorted by category (the key), then using dictionary and set function in Python3 to calculate the number of videos trending in country A and the percentage of them also trending in country B. Finally, it integrates the data and output with regular format.

Parallelization:

The mapper and reducer phase can process the data in parallel. The mapper runs in parallel on different partition of input data that divided by "category" (key). I have set to use 3 reducers, they run in parallel on different partition of the intermediate results. Each partition deal with data corresponding to a subset of categories.

I did run the work at SIT cluster (soit-hdp-pro-1) used 21s:

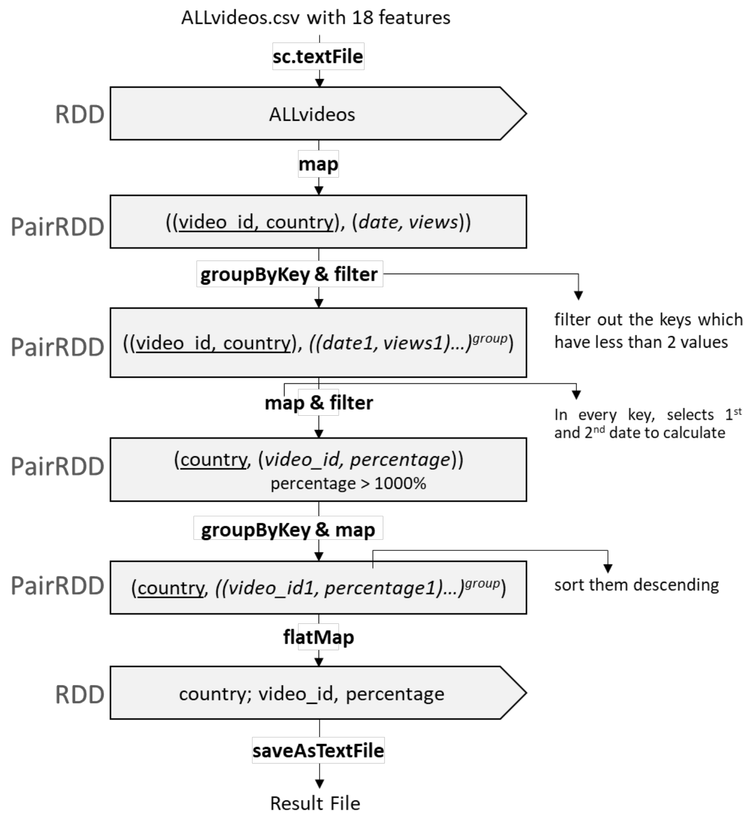


User:	dzha4889
Name:	country inverted list
Application Type:	MAPREDUCE
Application Tags:	
YarnApplicationState:	FINISHED
FinalStatus Reported by AM:	SUCCEEDED
Started:	Thu Apr 26 04:48:06 +1000 2018
Elapsed:	21sec
Tracking URL:	History
Diagnostics:	

COMP5349 Assignment 1
Name: Dongdong Zhang
SID: 470161133
Tutor Name: Andrian Yang

Workload: Impact of Trending on View Number

This workload is for question 2 to find all the videos in every country which have more than 1000% increase in first two trending view. The job is completed under Spark with Python3. The sequence of transformation and actions are illustrated in Figure 2:



ALLvideos.csv with 18 features
sc.textFile
RDD
ALLvideos
map
PairRDD
((video_id, country), (date, views))
groupByKey & filter
filter out the keys which have less than 2 values
PairRDD
((video_id, country), ((date1, views1)…)group)
map & filter
In every key, selects 1st and 2nd date to calculate
PairRDD
(country, (video_id, percentage))
percentage > 1000%
groupByKey & map
PairRDD
(country, ((video_id1, percentage1)…)group)
sort them descending
flatMap
RDD
country; video_id, percentage
saveAsTextFile
Result File

Figure 2: Spark phase for the workload

ALLvideos.csv is read in and mapped to create ((video_id, country), (date, views)) RDD pair. It groups the date (transforms it can be calculated) and views in the same key (video_id, country), then filters out the keys which have less than 2 values to create ((video_id, country), values^n)[1] RDD pair.

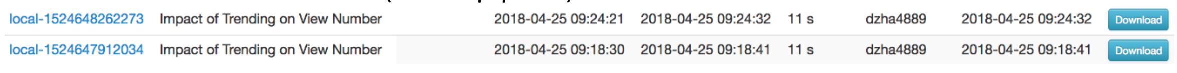
After groupByKey and filter, it uses map to select first two date of values in every keys (video_id, country) then calculate the percentage (2nd / 1st day views) and filter out less than 1000% percentage's values. Based on the intermediate results, the key is changed to country, values changed to video_id and percentage.

Finally, using the keys group values to sort the percentage descending, and flatmap the group data to output format.

Parallelization

The built-in SparkContent parallelize method is utilized to read the original data. All the operations can automatically run in parallel on different partitions of the contents except sortBy transformation. Because the sortBy need compare all data together. Thus I didn't use it, but I implement the group the key for parallelization and map to sort the values in the same keys, so that can save much time. Therefore, I used operations can be parallelized.

I did run the work at SIT cluster (soit-hdp-pro-1) used 11s:



local-1524648262273	Impact of Trending on View Number	2018-04-25 09:24:21	2018-04-25 09:24:32	11 s	dzha4889	2018-04-25 09:24:32	Download
local-1524647912034	Impact of Trending on View Number	2018-04-25 09:18:30	2018-04-25 09:18:41	11 s	dzha4889	2018-04-25 09:18:41	Download

[1] values^n = ((date1, views1), (date2, views2), …, (date_n, views_n))