Supercomputing for Big Data ET4310 (2016)

Assignment 2
Parag Bhosale(4517415)

September 2016

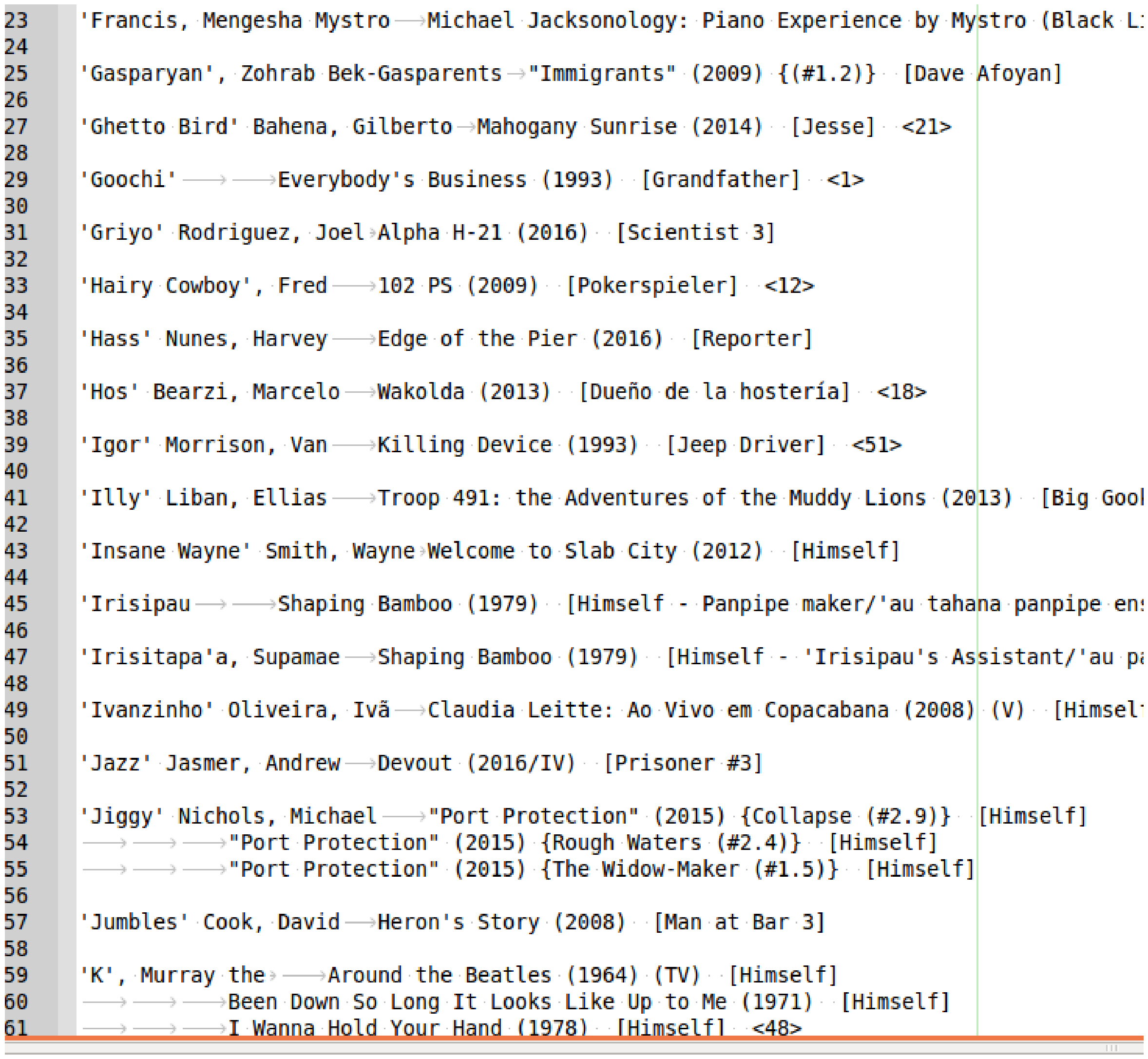
Introduction

The aim of the assignment is to use Apache Spark for an intensive algorithm for semi-structured input data. The algorithm in this assignment is **six degrees of Kevin Bacon** where we need to find out actors which are at up to distance 6. To achieve this, IMDB database is used for this assignment.

Background

This assignment is based on iterative algorithm because it is needed to iterate the algorithm to calculate actors at each distance up to 6. For such algorithm, use of the caching will be effective. This will improve the performance of iterative algorithm significantly. This is because applications can access the data from RAM instead of disk. This can be done by using .cache() only for those RDDs which are accessed repeatedly. In this assignment, SparkListener is used to log data about memory size, the number of partitions to a file. We can log data at the start of the application, at the end of the application or when a stage is completed. This is useful to view memory footprint of the RDDs used especially those are cached.

Implementation



```
'Francis, Mengesha Mystro	Michael Jacksonology: Piano Experience by Mystro (Black L

'Gasparyan', Zohrab Bek-Gasparents	"Immigrants" (2009) {(#1.2)}  [Dave Afoyan]

'Ghetto Bird' Bahena, Gilberto	Mahogany Sunrise (2014)  [Jesse]  <21>

'Goochi'		Everybody's Business (1993)  [Grandfather]  <1>

'Griyo' Rodriguez, Joel	Alpha H-21 (2016)  [Scientist 3]

'Hairy Cowboy', Fred	102 PS (2009)  [Pokerspieler]  <12>

'Hass' Nunes, Harvey	Edge of the Pier (2016)  [Reporter]

'Hos' Bearzi, Marcelo	Wakolda (2013)  [Dueño de la hostería]  <18>

'Igor' Morrison, Van	Killing Device (1993)  [Jeep Driver]  <51>

'Illy' Liban, Ellias	Troop 491: the Adventures of the Muddy Lions (2013)  [Big Gool

'Insane Wayne' Smith, Wayne	Welcome to Slab City (2012)  [Himself]

'Irisipau		Shaping Bamboo (1979)  [Himself - Panpipe maker/'au tahana panpipe en

'Irisitapa'a, Supamae	Shaping Bamboo (1979)  [Himself - 'Irisipau's Assistant/'au pa

'Ivanzinho' Oliveira, Ivã	Claudia Leitte: Ao Vivo em Copacabana (2008) (V)  [Himsel

'Jazz' Jasmer, Andrew	Devout (2016/IV)  [Prisoner #3]

'Jiggy' Nichols, Michael	"Port Protection" (2015) {Collapse (#2.9)}  [Himself]
			"Port Protection" (2015) {Rough Waters (#2.4)}  [Himself]
			"Port Protection" (2015) {The Widow-Maker (#1.5)}  [Himself]

'Jumbles' Cook, David	Heron's Story (2008)  [Man at Bar 3]

'K', Murray the		Around the Beatles (1964) (TV)  [Himself]
			Been Down So Long It Looks Like Up to Me (1971)  [Himself]
			I Wanna Hold Your Hand (1978)  [Himself]  <48>
```

Figure 1: Input file snapshot using geany editor

Algorithm

For this assignment, IMDB database is used. To process the data, header and trailer are removed from both actors and actresses files. In figure 1, we can conclude that each record is separated by two newline characters , in addition, each actor can be separated by the tab. Observing this two, we can get format (actor,list¡movies¿). After that files are read by using **newAPIHadoopFile** and further transformations are done. as a record delimiter.
Once (actors,list¡movie¿) is obtained, further flattening is done and required filters are applied to get (actor,movie). As there are male and female actors, flag "**::F::**" is added to the name of the females. This has the advantage over adding a new value in the tuple. First, out of the total actors, there are only 35% female actors. This means flag can be added to less number of data. If

we add new value such as (actor,gender,movie), each record will have a new value. But as the self join is required further, this approach will create (movie,(actor,gender),(actor,gender)) tuple in order to get co-actors. This will require more memory than our implemented approach of adding a flag to the female name as this will create only (actor,gender) and addition data is added to only 35% of the total data.
Once (actor,movie) is obtained, we can get co-actors by self-joining the tuple by using movies as the key. This will create (movie,(actor1,actor)) where (actor,actor) are co-actors. Then individual actor can be obtained by taking distinct of first value of the (actor1) and next step is to add distance to this (actor,distance)//

Iteration

I have implemented iteration using two methods. The first method is slightly different than the one specified in the lab manual.The second method is as per stated in the lab manual.

Method one

Consider two rdd (actor1,actor) and (actor,distance). Now flip the values from first to get (actor,actor1) and join it to (actor,(actor1,distance)).The meaning of (actor1,distance) means the distance of actor1 for each coactor. So actors which have Kevin Bacon as co actor will have one distance as 0 and other as 9999 from other actors. So when we reduce (actor1,distance) to get min(distance)+1, we will get actors at the distance 1. Similarly, further iterations are done.
In this implementation, rdd of (actor1,actor) must be cached as it is used in each iteration.

Method Two

Method two is similar to the method one,but in this method, AggregateByKey is applied to get (actor,list¡actors¿)). This rdd is joined with the distance RDD to get (actor,(distance,list¡actors¿)). Hence, once this is done flattening is done on the rdd to get (actor,distance+1). This RDD is reduceByKey to get the minimun distance. This RDD is now contained actors from distance 1 from Kevin Bacon. Similarly, other iterations are done till distance 6.

SparkListener

In this assignment, SparkListener is implemented to log memory size of the cached RDD to a file. I have used onStageCompleted to log cached RDD information to a file. To check if a rdd is cached or not, iscached flag is used. Further memsize and disksize is used to log memory size of the particular RDD. We can set a name for a RDD using **.setmname("name")** and can display name by using **.name** in the listener code.

Results

The assignment was run on kova as well as my laptop with 2 cores.
Specifications for Lenovo

- Lenovo z51 intel core i5 2.20 GHz GB RAM
- Number of $CPU = 4$ and Number of cores per $socket = 2$

Machine	Runtime(min) with compressed	Runtime(min) with out compressed
Kova	23	18
Lenovo	NA	29

Table 1: Average run time for method one

(I could not able to run method with compression on Lenovo. Computer hanged due to poor performance)

Machine	Runtime(min) with compressed	Runtime(min) with out compressed
Kova	30	24-26

Table 2: Average run time for the method two

As we can observe from the table, memory size of cached RDD decreases but at the same time execution time increases for the application. So, there is a tradeoff between memory and execution time which can be chosen by setting compressedRDD flag true or false.

Conclusion

First, semi-structured data is converted successfully into structured data using newAPIHadoopFile and flattening the data so that it can be helpful for further processing. Output files are created as expected but the performance can be improved further for the both methods. The main drawback is the self-join which creates up to 2.5 GB of data and is used in every iteration. Iterations are implemented correctly with the help of variable ListBuffer. Information about the RDDs is successfully logged to the file using SparkListener.This can be helpful in monitoring memory footprint of the RDD. Further, the behavior of the application is observed using compression flag. Execution time is increased if compression is used. This is because additional time is needed for the decompression of the data but memory footprint will be less. This is helpful is the system has memory constrains.