DAT470/DIT065 Assignment 3

Mirco Ghadri
mircog@chalmers.se

Monday 6th May, 2024

The topic of this assignment is the MRJob module in python and how it can be used to solve problems related to cluster computations. The focus will be on using MRJob to investigate twitter followers/followings in a large dataset of twitter accounts [1].

Problem 1

(a) Describe a MapReduce algorithm that determines the Twitter ID of the person who follows the largest amount of users, the number of users this person follows, the average number of users followed, and the number of accounts that do not follow anyone. (2 points)
Answer: The MapReduce algorithm for this will be pretty straightforward.

- **Mapper:** The MRJob algorithm will pass each line in the input file as a value argument to the mapper function. The key will be a random value, such as the line number, which we can ignore. We want to convert the input line in the value argument to a key value pair that we can pass to the reducer. The key should be the user id and the value should be the number of user id's that this user follows. We can extract all this information from the line we read in the input file.
- **Reducer:** The reducer will receive a key which is the user id, and a generator object which contains all the values that were mapped to that key. We then want to take the sum of the values in the generator object which we can do by using the command `sum`. The reducer should yield a new tuple with a key of None and a value of key-value pair that was passed to it as argument, except for the value modified so that it is the aggregate value calculated above using `sum`. The reason we yield a new tuple with key None is so that we can calculate aggregate statistics for the data, such as person with most followings, average number of followings, number of users with no followings etc. This will be done inside our second reducer function.
- **Reducer2:** Here we calculate all the aggregate statistics. The reason this is possible is because we yielded a tuple with key None, so all data will have the same key and will be passed to this function so that we can iterate over it.

- **steps:** Here, we create the pipeline that will do the calculation. We start with the first step which contains the mapper and reducer. The second step should only contain reducer2. We are now done.

(b) The file mrjob_twitter_follows.py *contains the interface for a MRJob implementation that where you will need to fill in the blanks by implementing your algorithm. Implement the algorithm.* (4 points)
Answer: See `assignment3_problem1.py`.

(c) Measure the scalability of your algorithm on 1, 2, 4, ..., 32 cores. Plot the empirical speedup as the function of cores. Use the 10M dataset for scalability experiments. In addition to the plot, report the single-core runtime on the dataset. (3 points)
Answer: The command we used to run the job on SLURM was `/opt/local/bin/run_job.sh -e mrjob -c x -s mrjob_twitter_measure.py -- -w x /data/2024-DAT470-DIT065/twitter-2010_10M.txt` where we replaced x by the number of CPU cores/workers allocated for the job. We did not need to implement multiprocessing ourselves, since the MRJob module automatically supported it. All we had to do was to specify the number of workers to the `mrjob_twitter_measure.py` script and also allocate the correct number of CPU cores to SLURM. To get the single-core runtime, the command was thus: `/opt/local/bin/run_job.sh -e mrjob -c 1 -s mrjob_twitter_measure.py -- -w 1 /data/2024-DAT470-DIT065/twitter-2010_10M.txt`

The running times were as follows:

Table 1: Time elapsed for different numbers of workers

Number of workers	Time (s)
1	235.536
2	145.673
4	116.618
8	104.351
16	98.323
32	90.927

To calculate the empirical speedup as a function of the number of CPU cores, we used the formula

$$S = \frac{t_1}{t_n} \tag{1}$$

where t_1 is equal to the running time with 1 worker and t_n is the running time with n workers. The resulting empirical speedup for n workers is:

(d) Run your implementation (with a large number of cores) on the full dataset. Report the values (ID and number of accounts followed by the account that follows most accounts, average number of accounts followed, and the number of accounts that follow no-one. (2 point)
Answer: To run it on the full dataset, we needed to use as many cores as we could possibly get. Since the default bayes partition only offered 35 CPU cores, we did not use it. Instead we used the partitions cpu-markov/cpu-shannon which offered 80 CPU cores. The command to run on this partition was:

Table 2: Empirical speedup for different numbers of workers

Number of workers (n)	Speedup (S)
1	1
2	1.616
4	2.019
8	2.256
16	2.395
32	2.589

`/opt/local/bin/run_job.sh -e mrjob -p cpu-markov -c 80 -s mrjob_twitter_measure.py -- -w 80 /bayes_data/2024-DAT470-DIT065/twitter-2010_full.txt`.
The result was the following:

Table 3: Summary Statistics

Metric	**Value**
Most followed id	813286
Most followed	770155 people
Average followed	35.253 people
Count follows no-one	5963082 users
Number of workers	80
Time elapsed	171.608 s

(e) The account who follows most other accounts still exists and is active. Determine whose account it is. (1 point)
Answer: We used this website https://tweeterid.com/ to determine that the user id 813286 belonged to the user @barackobama.

Problem 2

(a) Describe a MapReduce algorithm determines the Twitter ID of the person who has the largest amount of followers, the number of followers this person has, the average number of followers, and the number of accounts that do not have any followers. (2 points)
Answer: This algorithm is a bit more difficult than the first algorithm since it is not as straight forward. However, we can start defining the functions we will need:

- **mapper:** As before, we are interested in the value argument fed to the mapper. It contains a line of the format id_0 : id_1 id_2 ... id_n. For each id to the right of the colon(:), we want to yield a value of (id,1). The reason is that it indicates that the user with that specified id has 1 follower. However, we also want to do (id_0,0), that is, yield the id to the left of the colon. This makes sure that we also keep a count for all users, not only the users who have followers, but also the users who have potentially zero followers.

- **reducer:** Here, we receive a key value pair of id and a generator of ones/zeroes from the mapper. We want to take the sum of the values using `sum`. We then yield a new tuple with key None and value of the tuple that was passed to it but with `sum` function applied to its value. So it would look like this: `None, (id,sum(ones))`. The reason we create a tuple with key None is so that the second reducer function will have a reference to all values so that it can calculate aggregate statistics.

- **reducer2:** This function will calculate the maxmimum number of followers, the average number of followers, the users with no followers etc. It can do so because it has a reference to all data since they shared the same key None.

- **step:** This creates the pipeline. We create the first step consisting of mapper and reducer. The second step consists of only reducer2.

(b) The file mrjob_twitter_followers.py contains the interface for a MRJob implementation that where you will need to fill in the blanks by implementing your algorithm. Implement the algorithm. (4 points)
Answer: See `assignment3_problem2.py`.

(c) Measure the scalability of your algorithm on 1, 2, 4, ..., 32 cores. Plot the empirical speedup as the function of cores. Use the 10M dataset for scalability experiments. In addition to the plot, report the single-core runtime on the dataset. (3 points)
Answer:

Table 4: Running time for different numbers of workers

Number of workers	Time (s)
1	733.855
2	547.444
4	383.075
8	306.671
16	292.654
32	281.521

Table 5: Empirical speedup for different numbers of workers

Number of workers (n)	Speedup (S)
1	1
2	1.341
4	1.917
8	2.392
16	2.507
32	2.608

(d) Run your implementation (with a large number of cores) on the full dataset. Report the values (ID and number of followers by the account that has most followers, average number of followers, and the number of accounts that have no followers. (2 point)
Answer: We got the following table of results. It is worth noting that this took much longer time to run, even though we ran with 80 CPU cores here also.

Table 6: Summary Statistics

Metric	**Value**
Most followers id	19058681
Most followers	2997469 people
Average followers	35.253 people
Count no followers	1548949 users
Number of workers	80
Time elapsed	2131.355 s

(e) The account that the most followers, unfortunately, is defunct. Still, it should be relatively easy to find out who that was. Determine whose account it used to be. (1 point)
Answer: Using the tool https://tweeterid.com/ unfortunately did not reveal who the id 19058681 used to belong to. However, after searching for the id on google, it seemed like it belonged to the account @aplusk owned by the actor Ashton Kutcher. This could further be confirmed by pasting the url: https://twitter.com/intent/user?user_id=19058681 into the search bar and hitting enter. It would lead to the twitter page of that same account.

References

[1] Haewoon Kwak et al. "What is Twitter, a Social Network or a News Media?" In: *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. 2010. DOI: 10.1145/1772690.1772751.