Heidelberg University
Winter semester 2017/2018

Group Parallel and Distributed Systems (PVS)
Artur Andrzejak, Diego Costa

Problem Set 1 for lecture Mining Massive Datasets

Due October 30, 2017, 11:59 pm

Exercise 1 (3 points)

Consider a cluster of n machines. Each has a probability p of failing in a given period of time T.

a) What is the probability of at least one machine failure during this period of time? Hint: Consider the complementary event that no machine fails during that time.

b) For $0 \leq k \leq n$, what is the probability p_k (in terms of k, n and p), of exactly k machines failing during T? Give an explanation.

c) Show, that the probabilites from **b)** satisfy: $p_1 + p_2 + \ldots + p_n = 1 - (1-p)^n$

Exercise 2 (6 points)

Familiarize yourself with Spark. You can use either Python or Java.

a) Describe each of the following transformations: `join()`, `sort()`, `groupby()`. In particular, what is in each case the type of input and type of output? Give for each one an example ("on paper") with a simple input and output.

b) Give three own programming examples of your choice for transformations (but not for `map()` or `filter()`) and three examples for actions (again, of your choice). Write executable code and test its correctness (either single program or several ones). To generate initial RDDs you can use code from lecture two or from Spark documentation. Submit as solution the source code and results of program runs.

Exercise 3 (4 points)

Recall the example implementation of k-means from lecture two. The implementation uses a for-loop which executes several identical operations for each of the k clusters. Improve the code by using Spark group operations instead of the for-loop.

a) Analyse the k-means code in `/examples/src/main/python/kmeans.py`. Describe the iteration loop of this code, especially how `reduceByKey()` is used.

b) Rewrite the above code, using `groupByKey()` (and possibly other operations) to achieve the same result. Test the correctness. What is the scalability of this solution? Submit your code and explanations of it as a solution.

Exercise 4 (3 points)

Watch the presentation about *Advanced Spark Features* by Matei Zaharia[1] and answer the following questions:

[1] *Advanced Spark Features* by Matei Zaharia: Slides, Video

a) What does broadcast provide? Which other mechanism does it improve and how? Which features of the distributed program determine the number of times the variable will be actually transmitted over the network? Explain the role of tasks and nodes in this.

b) Describe the function of an accumulator. What is the alternative implementation without an accumulator and why is an accumulator a preferred option? Which example application for an acculumator is discussed in the video? What has to be implemented in order to define a custom accumulator? Compare the accumulator mechanism to the `reduce()`-function.

c) Give three examples of RDD operators that result in RDDs with partitioning. Explain the connection between partitioning and network traffic. How does the modification on the PageRank example use partitioning to make the code more efficient? How does Spark exploit the knowledge about the partitioning to save time in task execution? How can you create a custom Partitioner?

Exercise 5 (Bonus, 2 points)

Consider the code of the k-means implementation in lecture two[2]. In lines 130 and 157, a copy of a Python array is created. As explained in lecture two, the suffix `[:]` creates a copy of the whole array (this copy is then referenced by the left-hand side variable).

Consider the versions of the same code, where the suffix `[:]` is sometimes removed from each of the two lines. There are three cases, where this suffix is missing from either one or both lines. Describe the behavior of the code in each of these three cases. Explain your answer.

[2] IMMD Public Code