Duc Anh Phi
Michael Tabachnik
Edgar Brotzmann

Distributed Systems I (IVS1), WS18/19

Solutions for Problem Set 4

Exercise 2

a) *What does broadcast provide?*
   - The broadcast()-functions allows to bind variables and data to each cluster and not only to the every executing task on each cluster. In the given example, the pages.txt-file needs to be sent once to every cluster, where it is accessible for multiple tasks.

   Which other mechanism does it improve and how?
   - It improves the scalability of clusters by reducing the communication costs.

   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.
   - The variable will only be transmitted to a machine, when it wants to access it the first time. Once it is transmitted, the variable will be cached on the node and gets available for all running tasks on it. In conclusion, if there are 100 tasks running on 10 nodes, the variable will be transmitted 10 times over the network, even if the whole cluster consists of 20 nodes.

b) *Describe the function of an accumulator.*
   - An accumulator is like an advanced counter for the whole cluster and not only for a single node. It can be declared as an integer- or double-accumulator and can be increased by any numeric value. Furthermore, there are custom accumulators.

   What is the alternative implementation without an accumulator and why is an accumulator a preferred option?
   - An alternative might be the usage of local variables, which are summed up after all tasks are finished. However, accumulators have the advantage, that you can collect statistics in real time, because updates for the accumulator will be send to the master node, where the value will be updated and send back to the worker nodes.

   Which example application for an accumulator is discussed in the video?
   - In the video, the example application was a filter, which counts how many bad records were collected and how big the records were. At the end, it prints a short summary of the statistics with the total amount of bad records and the average size of those.

   What has to be implemented in order to define a custom accumulator?
   - For any type T, there must exist a zero-element and an addInPlace method (“+=”). The zero-element is necessary to declare an initial value and the addInPlace method is needed to merge the values over the whole cluster.

   Compare the accumulator mechanism to the reduce()-function.
   - The reduce()-function collects data from all RDDs and sums them up. Usually, these operations are executed after all Tasks are finished.
     In contrast to the reduce-function, the master-node manages the value of the accumulator while the worker-nodes send only updates.

c) *Give three examples of RDD operators that result in RDDs with partitioning.*
   - pages.join(visists).reduceByKey(…)
   - pages.join(visists).map(…).reduceByKey(…)
   - pages.join(visists).mapValues(…).reduceByKey(…)

   Explain the connection between partitioning and network traffic.
   - By partitioning the data, the master-node knows which worker nodes which data need. As a result, only snippets with relevant data are sent the nodes and not the whole dataset.

   How does the modification on the PageRank example use partitioning to make the code more efficient?
   - The modification creates ranks and groups them by the given URL. Afterwards, the data are partitioned and can be evaluated faster.

   How does Spark exploit the knowledge about the partitioning to save time in task execution?
   - By reducing the amount of data, which needs to be sent over the network, the tasks doesn't need to wait for the data that long. Furthermore, if the RDDs are using map- or reduce-functions, the data will still be partitioned and the master-node can send the previous data snippet again.

   How can you create a custom Partitioner?
   - You can create a custom Partitioner by implementing a class which extends the class "Partitioner". Furthermore, this class must contain the following three functions:
     - numPartitions
     - getPartition
     - equals

Exercise 4

1. Describe each step of Spark execution model.
   - Create DAG of RDDs to represent computation
   - Create logical execution plan for DAG:
     - Pipeline as much as possible
     - Split into "stages" based on need to reorganize data
   - Schedule and execute individual tasks
     - Split each stage into tasks
     - A task is data + computation
     - Execute all tasks within a stage before moving on
2. In the execution phase, Spark tries to pipeline operations as much as possible. How does pipelining affect performance? Give examples of operations that can be pipelined.
   - A high level of pipelining combined with multiple threads can result in high performance, because the threads can run independent and without the necessity of waiting for other data. In addition, thread-scheduling is much easier: If one task is finished, the processor can easily run the next enqueued task. In the

presentation, the operations “map” and the operations “groupBy”, “mapValues” and “collect” can be pipelined.

3. List the four most common issues described by Aaron. What is the recommended setting and guidelines to deal with the problems described in the talk?
   3.1. Ensure enough partitions for concurrency
   3.2. Minimize memory consumption (especially of sorting and large keys in groupBy’s)
   3.3. Minimize amount of data shuffled
   3.4. Know the standard library
   - 3.1. and 3.2. are about number of partitions: The recommended lower bound is about 2 x number of cores in the cluster and the upper bound is depending on the execution time for each task (at least 100 ms).
   - To 3.2.: Increase spark.executor.memory, increase number of partitions or restructure the program.