Methods of Cloud Computing

Winter Term 2020/2021

Practical Assignment No. 5

Due: 31.01.2021 23:55

The primary goal of this assignment is to get familiar with data-flow engines with the example of Apache Flink. You will setup and evaluate a small test application locally and on a distributed Flink installation.

1. Prepare and Test a Local Flink Setup

Download and set up a local installation of Apache Flink. Flink requires a working installation of Java 8. Test your Flink setup by running the `WordCount` example.

Adapt the Word Count example in the following two ways:

- Ignore the case of all words, as well as any words containing non-alphabetical characters.
- Sort the results starting with the word with the most occurrences to the words with the least occurrences.

Process the entire input as a stream using the DataStream API. The sorting will require to use a window over the entire dataset. The input and output files' names are passed via command line arguments. You are free to solve this task in Java or Scala. The output must be a CSV (comma separated values) file as shown below.

```
word,occurrences
lorem,8
ipsum,5
dolor,3
...
```

Test your program with the dataset provided on ISIS (`tolstoy-war-and-peace.txt`). Record the results in the file `WordCountResults.txt`.

Outputs:

- `WordCount.[java|scala]`
- `WordCountResults.txt`

2. Use a Distributed Flink Installation

Execute your word count program on a distributed Flink installation. Choose one of the below options for this.

1. **Flink in Kubernetes**
   - Use the Kubernetes from the previous assignment to deploy Flink and Hadoop on at least 3 worker nodes
   - Use the Flink *session cluster* installation (NOT Flink job cluster).
2. **Flink in VMs**
   - As an alternative to your Kubernetes cluster, you can manually install Flink and Hadoop on three VMs and configure the installation to work together as a cluster.

   Regardless of which option you choose, document all commands that you used to prepare your distributed Flink installation and to prepare and execute the job. Comment each command that you use.

Hints:

- For your Flink in Kubernetes/VMs installation, use HDFS as data input/output. Change the URLs for your files to something like: `hdfs:///path/to/data.csv`

Outputs:

- `cluster.txt`
  - Containing all commands you used for preparing your Flink cluster, including comments explaining what the commands do

3. Experiments & Discussion

Run your word count program with different job configurations (such as initial placements or level of parallelism) to get a feeling on the impact on the runtime.

Answer the following questions:

- Which steps in your program require communication and synchronization between your workers?
- What resources is the job bound by? Memory? CPU? Network? Disk?
- Could you improve the partitioning of your data to yield better run-time?

Outputs:

- `discussion.txt (max. 200 words)`

4. Submission Deliverables

Submit your solution on the ISIS platform as individual files. Please submit ONLY the necessary files and use exactly the given file names!

Expected submission files:

- `WordCount.[java|scala]`
  - **6 points**
- `WordCountResults.txt`
  - **3 points**
- `cluster.txt`
  - **6 points**
- `discussion.txt`
  - **6 points**

Total points: 21