Bielefeld University
Faculty of Technology

Big Data Analytics

Summer 2023

Number 03, Submission Deadline: May 15th 2023, 11:59 PM

1. **MapReduce fundamentals:** (4 P)
   Describe the MapReduce graphs that model the following problems:

   - The multiplication of a `nxn` matrix by a vector of length `n`.
   - The grouping and aggregation[1] on the relation `R(A,B)`, where `A` is the grouping attribute and `B` is aggregated by the `MAX` operation. Assume `A` and `B` have domains of size `a` and `b`, respectively.

2. **Simulated MapReduce with Snakemake:**
   You will have noticed that the counting process before was pretty suboptimal and took a long time to sequentially walk through every line of the input. We now want to calculate the average grade for each student by simulating MapReduce through three steps, formulated as Snakemake rules: Split, Apply, and Join. The idea behind this is rather simple: To speed up resource intensive tasks the large set of inputs is first split into chunks that are then processed individually (and potentially in parallel, using multiple threads/processes/machines). After all input chunks have been processed the individual results are joined (aggregated) into the final end result. Be sure to familiarize yourself with the concept of intermediate rules and checkpoints[2] in Snakemake for this task. Remember that you can specify more `--cores` when running `snakemake` to speed things up.

   (a) **Split**: Create and implement a Snakemake checkpoint rule "split" that takes in an input file, e.g. `coursesTaken-A.csv` and creates chunks of up to 100000 lines. These chunks should be stored as separate output files in a dataset specific subdirectory `chunks/A/` or `chunks/B`, named for example: (4 P)

[1] This corresponds to $GROUP\_BY$ in SQL

[2] `https://snakemake.readthedocs.io/en/stable/snakefiles/rules.html#data-dependent-conditional-execution`

`chunks/A/1.csv`, `chunks/A/2.csv`, ..., `chunks/A/200.csv`.

Questions: Why can't we use a simple rule for this task? How many chunk files are created in this step?

(b) **Apply**: Implement a script `sum_chunk.py` that takes a single chunk as input and outputs a similarly named file with extension .sum. The content should take the following form by summing up the number of courses and total grade for each student in each chunk: `<student_id>\t<course_count>\t<grade_sum>` (4 P)

(c) **Join**: Calculate the average grade for each student over all `.sum` files created in the previous step. Write the output into a file called `aggregated-A.txt` / `aggregated-B.txt`. Each line should contain a student ID, a tab character, and their average grade. `<student_id>\t<average_grade>` (4 P)

(d) Execute your pipeline for both input datasets using `snakemake aggregated-A.txt` and `snakemake aggregated-B.txt` . (2 P)

Questions : (4 P)

- You only specify the output filename `aggregated-A.txt` of what you eventually want to see. How does this propagate information through the workflow specified by the Snakefile?
- Which parts of this workflow is performing map tasks, which are reducers?

3. **Visualization:**
The previous task left you with output files `aggregated-A.txt` and `aggregated-B.txt` that contain pairs of student IDs and their average grade.

For this task , add a new rule that creates a histogram plot of all average student grades and saves it as `aggregated-A.png` / `aggregated-B.png`. You can create the plot using either the `matplotlib` or the `seaborn` library. (4 P)

4. **Workflow graph and communication cost:**
Create a visual graph of the workflow you created in tasks 2-4. (2 P)

Important:
Please submit your group solution via Moodle. Note that Snake-

make will require you to create files such as Snakefile, please ZIP your solution directory (without including the data files)..