V

Universität Göttingen
Department of Computer Science

Julian Kunkel

Exercise Week 8
HPDA / 2021
240 Minutes Total
Discussion in Week 8: 2021-12-20

1. The tasks described in this worksheet are part of the formative assessment. They serve the purpose to prepare you for the examination. We will discuss the solutions during the next **interactive session** after they are handed out – while they fit to the lecture of the week they are handed out, they might be discussed in two weeks time due to the bi-weekly exercise schedule.
2. Make sure to plan your time for the whole sheet carefully. The complete exercise should represent approximately three hours of independent study. The time limit indicates how much time you should spend on each task, and not how much time you may actually need; it is important that you engage with the material and not that you complete all tasks perfectly. Feel free to collaborate and team up.
3. The exercises are designed to challenge you and train you further as guided self-study. The time limit might be too ambitious for you; you may team up with colleagues. It is not an issue as long as you manage to at least partially resolve each task within the time budget. If you (and team) are struggling, reach out for help in Teams! You may also share your thoughts on the channel.
4. We recommend that you create a (private) GIT repository where you store your findings and outcomes while processing the exercises. This portfolio of work could be useful in the future.

Contents

Task 1: Streaming Concepts (90 min)

Perform the following tasks on the slightly modified data flow example (original and definition of the operations in the exercise of Week 5). It borrows some syntactical elements from Spark streaming.

```python
# Slightly modified operator: Create a stream at TCP port 9999
d = dataflow.stream("localhost", 9999)

flat = d.map(lambda t: (t[0], eval(t[3])))
bd = flat.filter(lambda t: "HPDA" in t[1])
bd.write("out.csv") # stores contiguous stream of tuples

fm = flat.flatmap(lambda t: [[t[0], x] for x in t[1]])
# Similar to previous group operator, but now we take a window of 30s by which we group
z = fm.group_by_window_and_key(30, lambda t: t[1])
r = z.reduce(lambda t: len(t))

# Offer data at port 9998 as a contiguous stream of information
r.tcp_sink(9998)

d.start_streaming()
```

Assume we want to encode this program as a continuous streaming workflow.

- Sketch the processing DAG for the dataflow given in the Python Code. How could this be encoded as a Storm topology?
- Create a mapping of the parallel execution of the individual operation on a cluster with 2 compute nodes with 10 CPUs each.
- Discuss the inaccuracies that can be caused when using at-least-once processing semantics under failures of one of the nodes.

Portfolio (directory: `8/stream`)

`8/stream/dag.pdf`	The processing DAG

Task 2: Streaming Data Model for Crime Data (Theory) (150 min)

In this task, we build a data model for streaming crime data. Consider we are responsible to manage the police forces in a country and must delegate our limited (resources) to crime places. For such a decision, we may want to consider the severity of the crime, the location of our forces and the crime location, the number of required officers and their equipment. We receive all this information in real-time from various streaming sources, e.g., a report by a normal person, a report by the police force, and even data from social media.

Officers in the management should be provided with the refined information to delegate efficiently. In this task, we have to deal with input that has low quality, e.g., a crime reported by several people is more likely to be real and, thus, important than one only reported by a single person. Albeit the concepts and model you develop are applicable for other streaming engines, we focus on Storm in this task.

We assume to have at least the following input sources available (you may extend the list):

- Officer GPS, this source reports: time, officer ID, location (GPS coordinates)
- Officer supply, reports: time, officer ID, location, equipment (list)
- Social media source, we automatically try to filter relevant messages that are indicators for a potential crime and tread it like a reported crime. It reports: time, text, reporter, location
- Reported crime, this is a single unconfirmed crime reported by phone, it reports: time, location, type, text

Create a topology for this task and document the transformations on the individual bolts. Describe a simple heuristics to identify relevant crimes and how the topology could suggest an operator useful candidates to assign police force.

Output of the topology could be a tuple of (crime, location, severity, list of officers to send, time until destination). Note that officers could always move from a less important crime to a more severe crime.

You may want to use Bolts that access/manipulate persistent information (such as the HBase*Bolts). How could the different nodes be parallelized? What kind of grouping would be necessary?

Could we use DRPCs in this scenario for something useful (for what?)?

Portfolio (directory: `8/stream-crime`)

`8/stream-crime/streaming-model.pdf`	The data model with description and answers to the questions.