Data Stream Processing and Analytics **Prof. Vasiliki Kalavri**
Spring Semester 2020 vkalavri@bu.edu

Assignment 2

~~**Submission deadline:** February 26 2020, 11:59 pm~~
Submission deadline: March 2 2020, 11:59 pm

The rest of this assignment assumes a UNIX-based setup. If you are a Windows user, you are advised to use Windows subsystem for Linux (WSL), Cygwin, or a Linux virtual machine to run Flink in a UNIX environment.

1. Download and build the *updated* `clusterdata-analysis-2` project

The updated `clusterdata-analysis-2` project contains utility classes, template applications, tests, examples, and bug fixes. **Download the new project** from Blackboard, extract it, and build it:

```
cd clusterdata-analysis-2
mvn clean package
```

Open IntelliJ and import the new project following the same steps as in Assignment #1. You can use the same dataset, parts 0, 1, and 2 of the Job and Task Events tables.

You will need to set the input path variables again: Open the `clusterdata.utils.AppBase` class and point the variables `pathToJobEventData` and `pathToTaskEventData` to the appropriate locations.

2. Tasks

The assignment consists of the following four tasks. Each task's contribution to the assignment grade is indicated inside parentheses.

a. Implement GlobalTaskStatistics (10/100)

Write a Flink program that computes the number of (i) successful, (ii) failed, and (iii) killed tasks per minute (in event time). Tasks that complete successfully are identified by a FINISH event. The program must output events of type `Tuple4<Long, Integer, Integer, Integer>`, where the first field is the start timestamp of the event-time window, and the next three fields are the number of successful, failed, and killed tasks inside the window, respectively.

b. Implement PerMachineStatistics (10/100)

Write a Flink program that computes the number of (i) successful, (ii) failed, and (iii) killed tasks per machine per minute (in event time). Tasks that complete successfully are identified by a FINISH event. The program must output events of type `Tuple5<Long, Long, Integer, Integer, Integer>`, where ~~the first field is the machine id, the second field is the start timestamp of the event-time window~~ the first field is the start timestamp of the event-time window, the second field is the machine id,, and the next three fields are the number of successful, failed, and killed tasks inside the window, respectively.

c. Identify Busy Machines (30/100)

Write a Flink program that identifies every 5 minutes the busiest machines in the cluster during the last 15 minutes (in event time). The program must output the number of busy machines during the last 15 minutes. A machine is considered busy if more than a configurable threshold number of tasks have been scheduled on it during the specified window period. The program must output events of type

`Tuple2<Long, Integer>`, where the first field is the end timestamp of the event-time window and the second field is the number of busy machines during that window.

Hint #1: The operator responsible for counting busy machines needs a way to know when it has received results from all previously triggered windows, i.e., it needs to know when it has seen all records up to a certain timestamp.

Hint #2: If you use the `getCurrentKey()` method of the `OnTimerContext` or try to retrieve the record key from the `apply()` method of a `WindowFunction`, you might need to apply type casting.

d. Identify job stages (50/100)

Write a Flink program that identifies job stages as *sessions*, where a stage is defined as a period of SUBMIT events occurring close to each other (in event time). We assume that a stage has finished after an inactivity period of 10 minutes. The program must output the longest session per job, once the job has finished. The output events must be of type `Tuple2<Long, Integer>`, where the first field is the jobId and the second field is the number of tasks in the longest stage.

Hint #1: You cannot get the information about when a job has finished from the task events stream. Instead, you will need to connect the job and task streams and use timers or watermarks to decide when the results can be safely emitted to the output.

Hint #2: The dataset contains timestamps in microseconds, while Flink event time operates at millisecond scale. After timestamps and watermarks are assigned at the sources, timestamps are internally converted to milliseconds. All downstream Flink operators use the converted timestamps and not the original timestamps in the input data. If you need to set a timer based on the timestamp of a job event or task event, make sure to access the timestamp via the `Context`/`OnTimerContext` object and not directly via the `JobEvent`/`TaskEvent` object.

3. Tests

A testing infrastructure is already in place in the template project provided to you. If you want to write more tests, you can use the `JobEventCountTest` class as an example. For the test base classes to work properly, you will need to use the `printOrTest()` method as the dataflow sink.

4. Deliverables

To submit your solutions to this assignment, **upload the `src` folder of your clusterdata-analysis project to Blackboard**. If you have edited the `pom.xml` file in any way, make sure to provide the updated file as well. Note that **well-documented code is always easier to understand and grade**, so please make sure your code is clean, readable, and has comments.