Weekly Report 04

Practical Course Informatics

14.06.2023

PETLoader

Implementation

- Simplify code, structure
- Documentation https://github.com/Lukasye/pis_project_ss2023_group2
- Changed PETLoad mechanism
  - Load all PETs when initialisation
  - Only access external library when PET pointer out of bound
- Test on different machine

Datageneratorator

Implementation

For the generating of the data introduced by KITTI dataset. We constructed a DataGenerator, a Java Programm which publish the user specified data to a Kafka Server. And the downstream piprline will subscribe to each topic. So that we utilize the upstream infastructure to make sure no bias in the evaluation and also simplify the implementation. We use totally three topics: `test-data`, `test-image` and `user` to accept String input of GPS data, Bytearray of camera perception and also the user configuration change. The Generator contains a Texture User Interface(TUI) to acept user input, the command suported by the program are listed as follow:

Nr.	Command	Parameter	Description
1	run	--number X	Publish X message(including GPS and Camera data) to the topic
2	delayrun	--number X --delay Y	Publish X message(including GPS and Camera data) to the topic, with a delay between each message Y seconds
3	write	--msg X	Write message to user topic to specify new user configuration
4	reset		Reset the pointer for data to 0
5	script	-name X	Read the script in script folder with name X and execute the commands in ti
5	exit		Exit the Generator programm

```
# Script example
Run 10
Delayrun 10 0.5
Write change,1,1,1
Write hello
Run 50
# Activate ImagePET
Write situation,camera
Delayrun 50 1
```

GUI

Implementation

- Change Timestamp to readable text.
- Refresh Issue
- Location reformat
- (Mini map maybe)

PET Activation

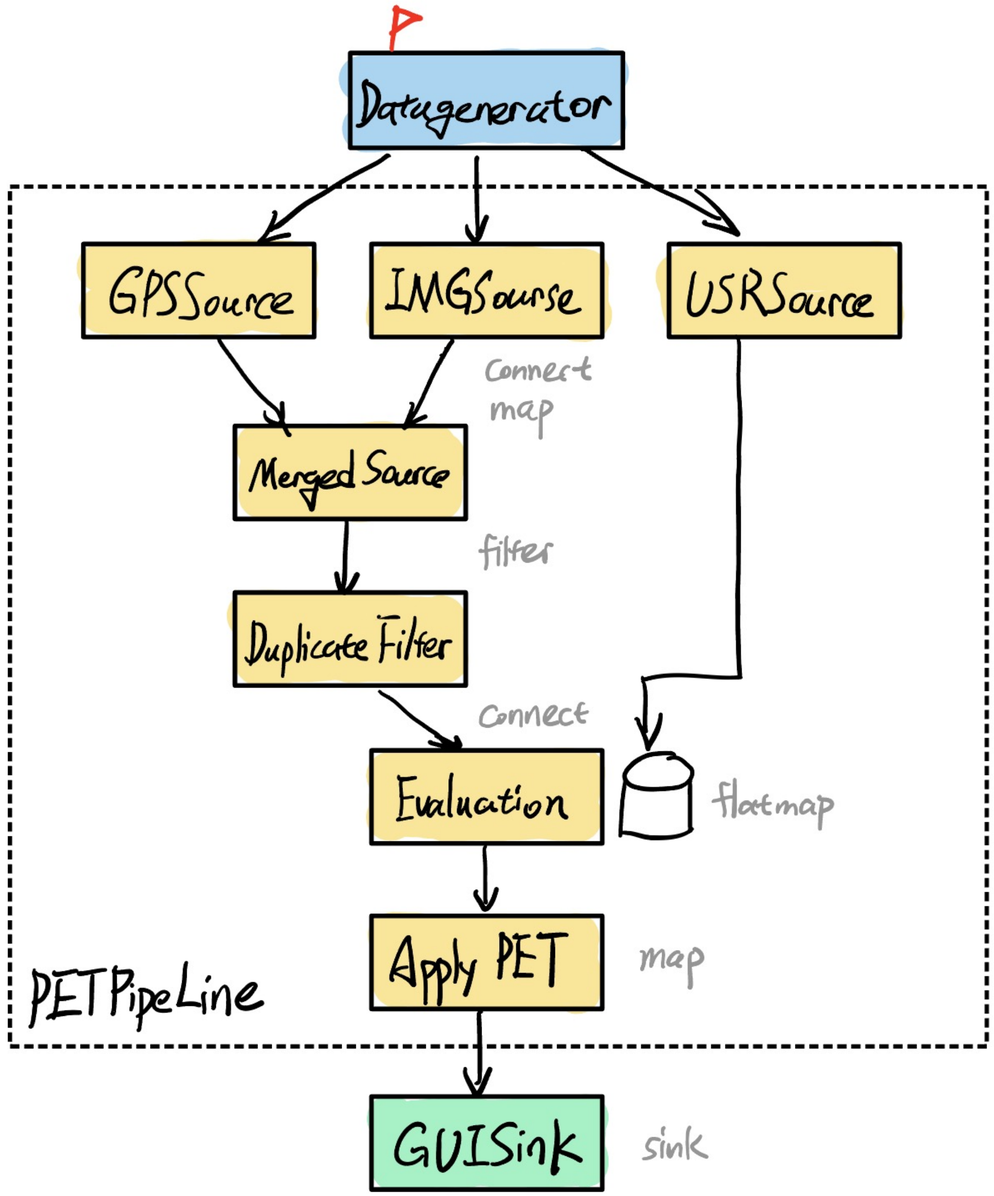
Some Idea

- Location Situation
  - Automatically activate through Euclidean distance
- Speed Situation
  - Manually activate through test-user topic
- Camera Situation
  - Manually activate through test-user topic

Whole Pipeline

Implementation

```
public static void main(String[] args) throws Exception {
    Lukasye *
    new PETPipeLine( confPath: "D:\\Projects\\pis_project_ss2023_group2-main\\carPET\\config\\Pipeconfig.json") {
        2 usages  Lukasye *
        @Override
        void buildPipeline() {
            env.setParallelism(1);
            // Read Image from kafka topic
            FlinkKafkaConsumer011<byte[]> kafkaSource = new FlinkKafkaConsumer011<>(
                    topic: "test-image", new PETUtils.ReadByteAsStream(), kafkaPropertyImg);
            FlinkKafkaConsumer011<String> sensorDataConsumer =
                    createStringConsumerForTopic( topic: "test-data", BOOTSTRAPSERVER, GROUPID);
            FlinkKafkaConsumer011<String> userDataConsumer =
                    createStringConsumerForTopic( topic: "test-user-input", BOOTSTRAPSERVER, GROUPID);
            DataStreamSource<byte[]> imageSource = env.addSource(kafkaSource);
            DataStreamSource<String> dataSource = env.addSource(sensorDataConsumer);
            DataStreamSource<String> userSource = env.addSource(userDataConsumer);

            // Merge two Stream
            ConnectedStreams<byte[], String> connectedDataStream = imageSource.connect(dataSource);
            SingleOutputStreamOperator<SensorReading> SensorReadingStream =
                    connectedDataStream.flatMap(new PETUtils.assembleSensorReading());

            // Duplicate filter
            SingleOutputStreamOperator<SensorReading> filteredStream =
                    SensorReadingStream.filter(new PETUtils.duplicateCheck());

            // Evaluation
            SingleOutputStreamOperator<SensorReading> evaluatedStream =
                    filteredStream.connect(userSource).flatMap(new PETUtils.evaluateSensorReading());

            //Apply PET
            SingleOutputStreamOperator<SensorReading> resultStream =
                    evaluatedStream.map(new PETUtils.applyPET<Double>(PETconfpath,  Type: "SPEED"))
                    .map(new PETUtils.applyPET<Tuple2<Double, Double>>(PETconfpath,  Type: "LOCATION"))
                    .map(new PETUtils.applyPET<byte[]>(PETconfpath,  Type: "IMAGE"));

            // Sink
            resultStream.addSink(new PETUtils.showInGUI(GUI));

        }
    };
}
```



Plan for next phase

- NiFi
- Debug and refinement of the code
- Mechanism for benchmark
- Complete the scenario and its script