Extending Apache Flink stream processing with Apache Samoa machine learning methods

12th September 2017

Piotr Wawrzyniak

Orange Polska
R&D Centre

Agenda

1. Motivation and goals
2. Apache SAMOA- an overview
3. Recent extensions to SAMOA project
4. Integration of SAMOA ML methods with Flink- idea
5. Implementation
6. Live demo
7. Summary

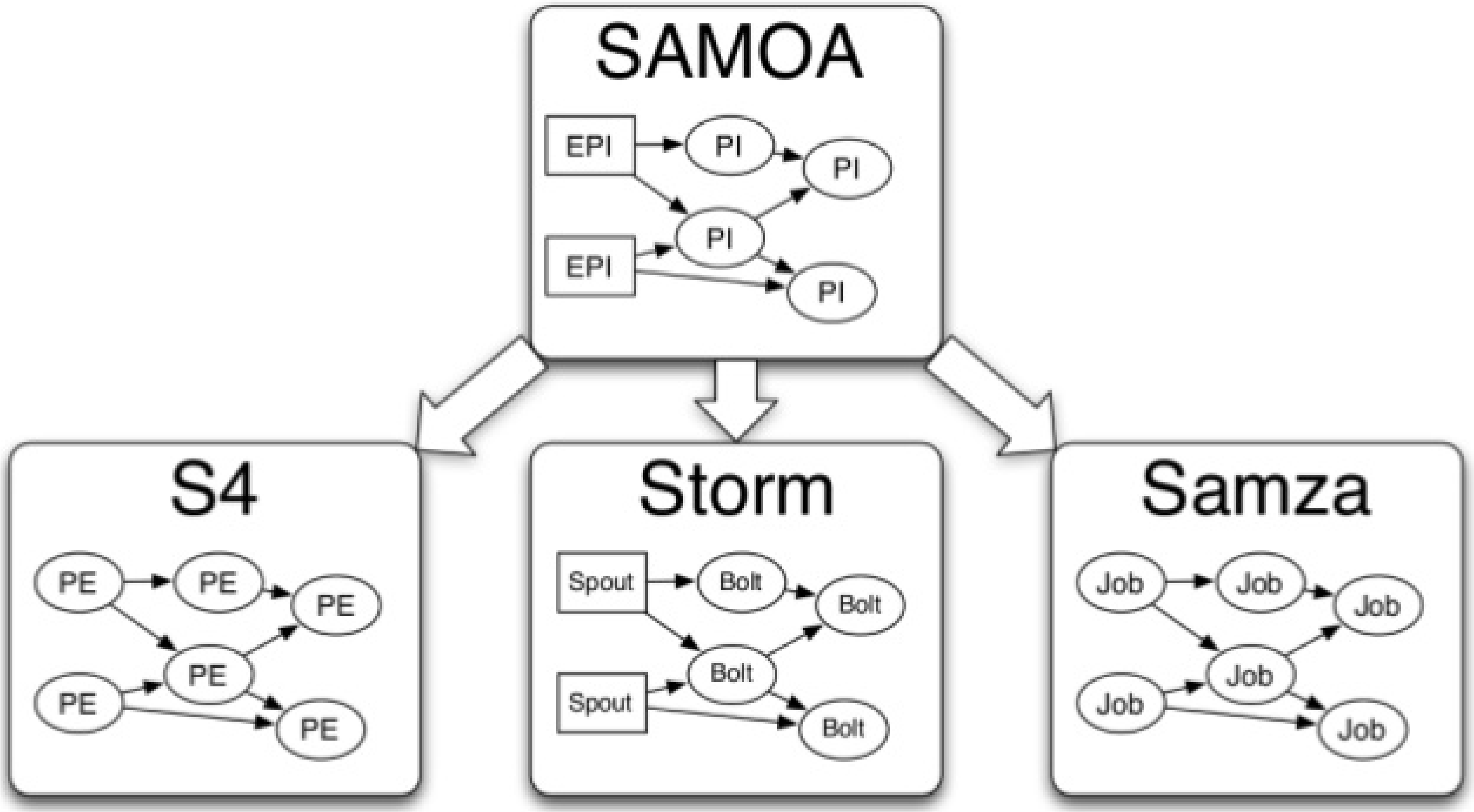
Motivation

1. Growing volume of continuously produced data requires effective means to process them
2. Some of the data streams (in particular in telco) have to be processed in true stream manner, without any data storage being allowed
3. Lack of stream mining methods in FlinkML, at the same Apache SAMOA provides a variety of such methods

Apache SAMOA- an overview

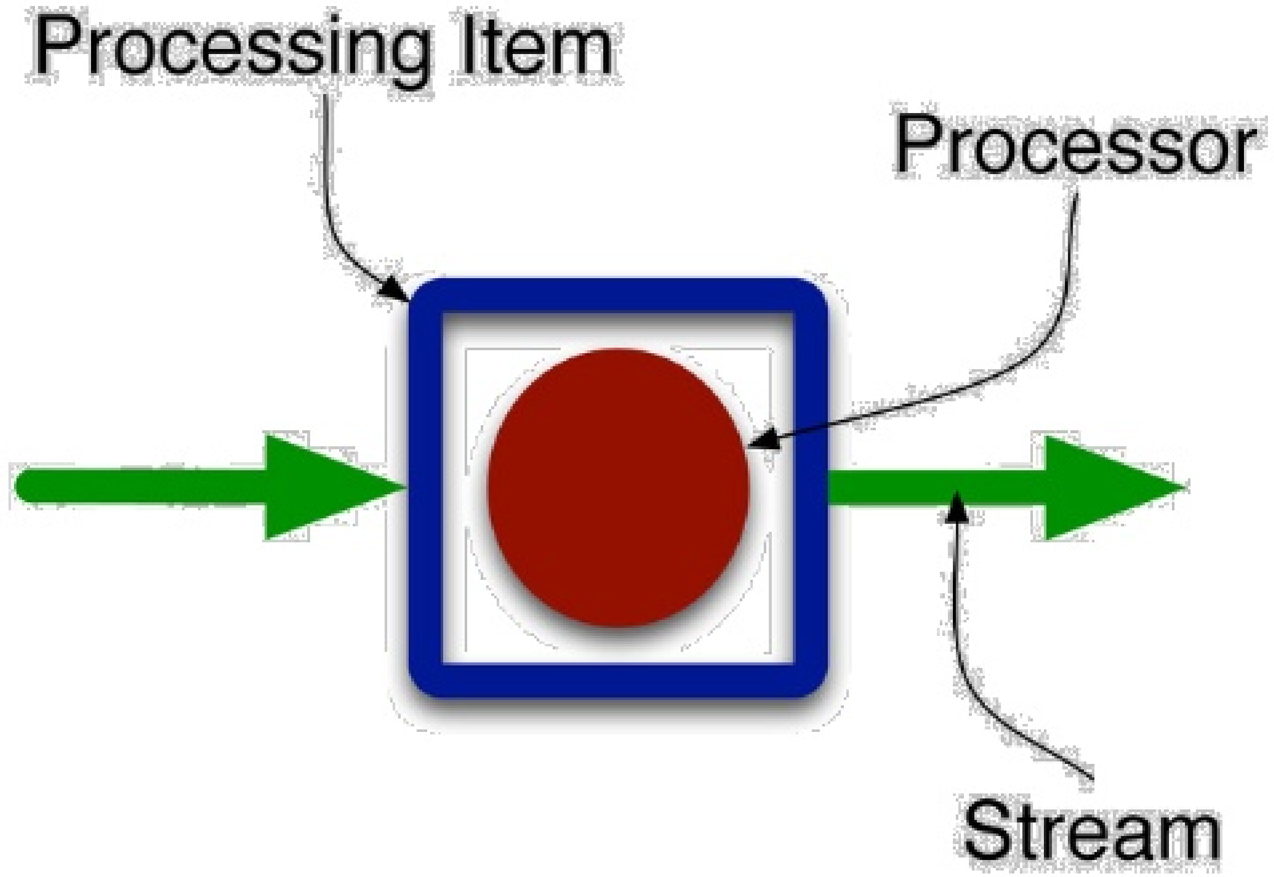
- Apache SAMOA is a platform for mining on big data streams.
- It is a distributed streaming machine learning (ML) framework that contains a programing abstraction for distributed streaming ML algorithms.
- Apache SAMOA enables development of new ML algorithms without dealing with the complexity of underlying streaming processing engines
- Apache SAMOA provides extensibility in integrating new SPEs into the framework. These features allow Apache SAMOA users to develop distributed streaming ML algorithms once and to execute the algorithms in multiple SPEs, i.e., code the algorithms once and execute them in multiple SPEs.
- Includes support for Apache Flink as DSPE

Apache SAMOA- an overview



Apache SAMOA- an overview

- Elastic API where developers can build topology
  - Computational elements- Processors
  - Connectors between them- Streams



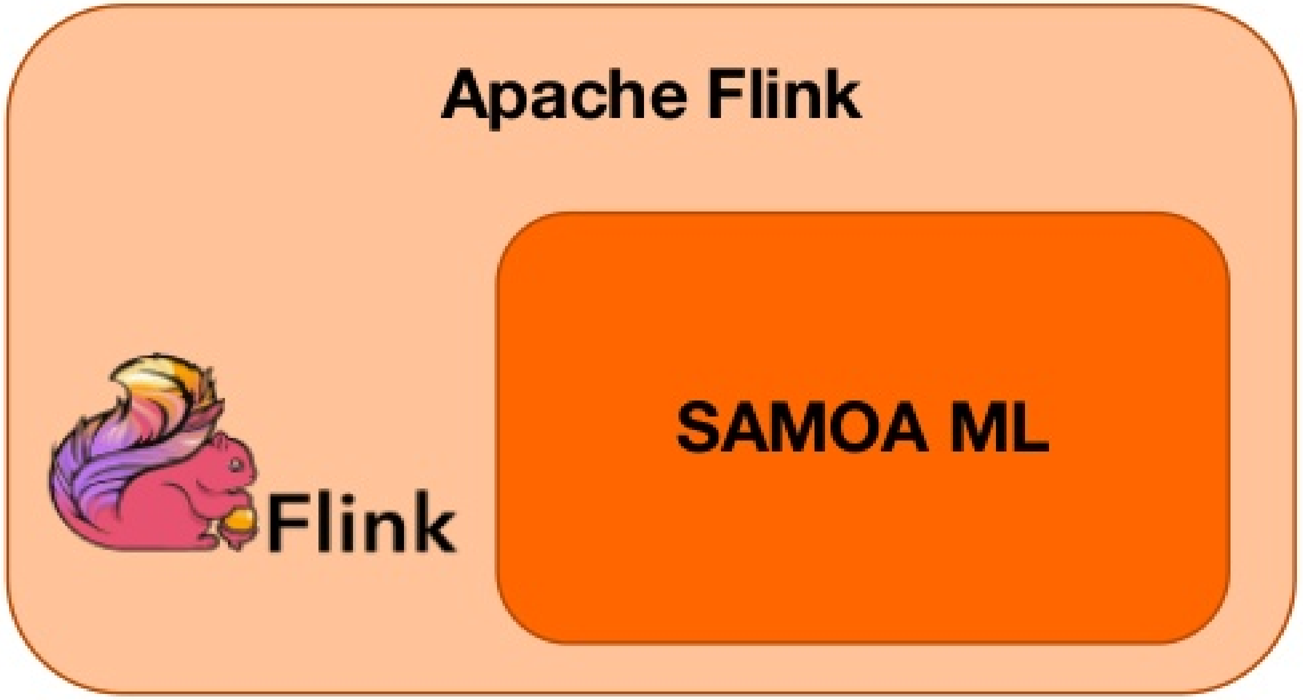
Recent extensions to Apache SAMOA

- **Orange team contributes to SAMOA project** to extend its interoperability within Apache ecosystem:
    - Kafka connectors (components to fetch data from Apache Kafka into SAMOA and to send output from SAMOA to Kafka)
    - Extensions allowing to save individual predictions to file
    - *Implementation of JSON and AVRO mappers for SAMOA objects (ongoing)*
    - *Implementation of task-based Kafka components (ongoing)*
    - *Modifications in ML algorithms (ongoing)*

Integration of SAMOA and Flink

The integration of SAMOA ML methods within Flink can be done in two ways:

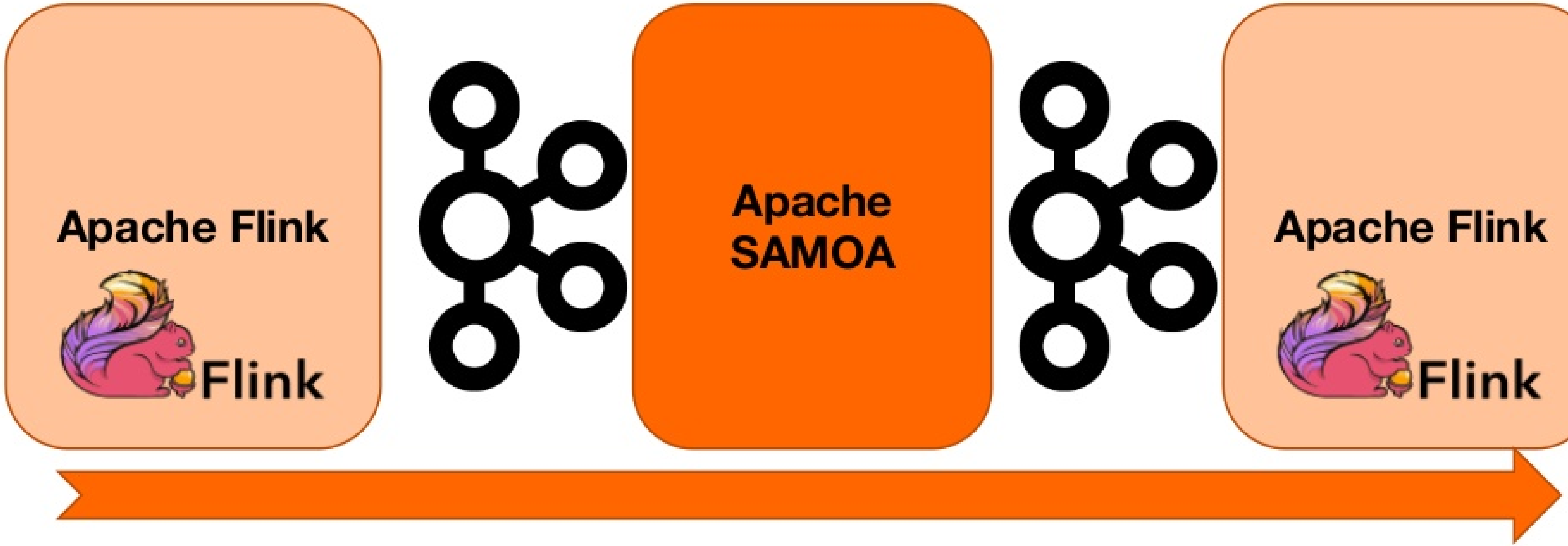
By integrating SAMOA code directly in Flink



Integration of SAMOA and Flink

The integration of SAMOA ML methods within Flink can be done in two ways:

By running SAMOA task and connecting it with Flink (e.g. via Kafka)



Integration of SAMOA and Flink

Code-based integration:

- Entire ML done within Flink execution environment
- Need to rewrite or ‚pack' SAMOA code within Flink patterns
- Some methods cannot be easily adopted (parallelism/scalability issues)

Kafka-based integration:

- Separation of ML and data pre and post processing
- No need to rewrite SAMOA code, it is enough to create SAMOA objects in Flink
- Communication cost (Kafka), but additional buffer at the same time
- SAMOA can run with different parallelism that Flink, thus allowing the use of non-parallel ML methods

Implementation

```
 env.addSource(new ArffSourceFunction(fileName))
              .map((InstanceContentEvent t) -> {
                GsonBuilder builder = new GsonBuilder();
                builder.serializeSpecialFloatingPointValues();
                Gson gson = builder.create();
                return gson.toJson(t);
              })
              .addSink(new FlinkKafkaProducer010<>(brokerList, topic, new
SimpleStringSchema() ));
```

Implementation

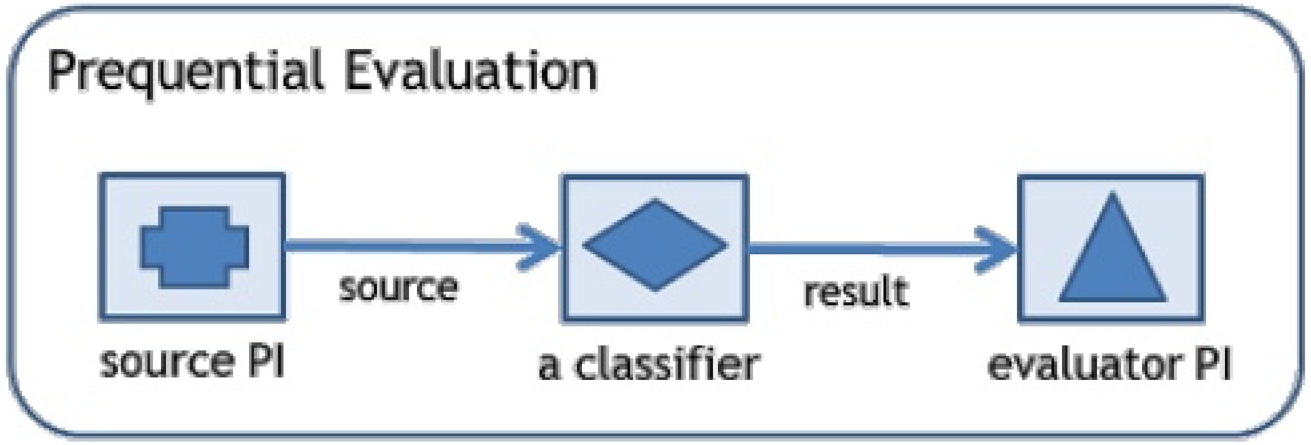
```
DataStream<String> samoaResults = env.addSource(new FlinkKafkaConsumer010<>(results,
new SimpleStringSchema(), props));


samoaResults.writeAsText(outFile, FileSystem.WriteMode.OVERWRITE);


env.execute("FlinkForward.Berlin");
```

Implementation

In SAMOA we will use Prequential Evaluation task or interleaved-test-then-train evolution. The idea is very simple: we use each instance first to test the model, and then to train the model. The Prequential Evaluation task evaluates the performance of online classifiers doing this.



Implementation

New parameters for Prequential Evaluation task:

- -l: classifier to train
- -s: stream to learn from
- -e: classification performance evaluation method
- -i: maximum number of instances to test/train on (-1 = no limit)
- -f: number of instances between samples of the learning performance
- -n: evaluation name (default: PrequentialEvaluation_TimeStamp)
- -d: file to append intermediate csv results to
- -o: broker for output Kafka topic
- -p: broker for output Kafka topic (port)
- -k: output Kafka topic (to send classification results to)

Implementation

New stream implemented: Kafka Stream

- Fetch data from Apache Kafka using KafkaEntranceProcessor
- Provides conformity with StreamSource pattern used in number of predefined tasks in SAMOA
- Parameters are: kafka broker, topic and timeout time.
- By default uses new JSON serializer/deserializer

Live demo

- Demo is based on Airlines dataset from MOA (Massive Online Analysis) project. The task is to predict whether a given flight will be delayed, given the information of the scheduled departure.
- The dataset (in arff file) is read by custom source utilizing components from SAMOA API.
- We will use Vertical Hoeffding Tree for delay prediction task. Vertical Hoeffding Tree (VHT) classifier is a distributed classifier that utilizes vertical parallelism on top of the Very Fast Decision Tree (VFDT) or Hoeffding Tree classifier
- The modified Prequential Evaluation task of SAMOA will be used, the modification includes in particular the use of Kafka as the data source and data sink.
- The results of computation will be received in Apache Flink and stored in a text file.

Live demo

orange™

Summary

- The integration of SAMOA ML and Flink can be done via Kafka, as SAMOA has appropriate components to handle input and output data streams
- The advantage is the possibility to independently tune or change ML method without modification of Flink code and lack of direct code integration and maintenance
- By separating ML in the processing chain, ML itself can run with different parallelism, it is crucial for some ML methods, especially those from MOA project
- The main disadvantage is communication overhead and possible buffering delay. The installation of Apache SAMOA is required for the ML execution

Thanks

Orange team involved in cooperation and contribution to SAMOA project:

- Giyyarpuram Madhusudan
- Piotr Wawrzyniak
- Maciej Grzenda
- Jerzy Jegier
- Jakub Jankowski

Thank you ☺

You can reach me at
piotr.wawrzyniak@orange.com

orange™