Flink CEP: Introduction and comparison to current pure-flink implementation

Leon Schäffer

Table of contents

Introduction of CEP

- Complex Event Pattern
- Define Pattern like RegEx
  - Define simple conditions
  - Combine multiple Simple conditions to a single complex condition
- Classes have to have the:

```
@Override
public int hashCode() {}

@Override
public boolean equals(Object t) {}
```

Defining a Pattern

Always starts with:

- begin(String name)
- begin(pattern_sequence)

Followed by:

where(condition)

```
start.where(SimpleCondition.of(value -> value.getName().startsWith("foo")));
```

start = pattern variable name

Pattern repetition Options

- times(int exact number)
- times(int from, int to)
- optional()
- greedy() (Single Pattern only)
- oneOrMore()
- timesOrMore(int number)
- consecutive()
- allowCombinations()

Combining Patterns

- or(condition)
- until(condition)
- subtype(Class)
- next(pattern)
- followedBy(pattern)
- followedByAny(pattern)
- notNext(pattern)
- notFollowedBy(pattern)
- within(time)

Combination of Pattern Example

```
DataStream<Event> input = ...;

Pattern<Event, ?> pattern = Pattern.<Event>begin("start")
    .where(SimpleCondition.of(event -> event.getId() == 42))
    .next("middle")
    .subtype(SubEvent.class)
    .where(SimpleCondition.of(subEvent -> subEvent.getVolume() >= 10.0))
    .followedBy("end")
    .where(SimpleCondition.of(event -> event.getName().equals("end")));

PatternStream<Event> patternStream = CEP.pattern(input, pattern);
```

Cep Conclusion

- Can detect Patterns for Objects like RegEx for Strings
- Can output found matching Patterns

Current Implementation

```
streamOperator
    .map(AlertJob::log) SingleOutputStreamOperator<String>
    .map(Contribution::createContribution) SingleOutputStreamOperator<Contribution>
    .filter(contrib -> contrib.filterBoundingBoxAndPattern(boundingBox, pattern))
    .map(log -> 1) SingleOutputStreamOperator<Integer>
    .windowAll(TumblingProcessingTimeWindows.of(seconds(windowSeconds))) AllWindowedStream<Integer, TimeWindow>
    .reduce(Integer::sum) SingleOutputStreamOperator<Integer>
    .addSink(mailSink) DataStreamSink<Integer>
    .uid(sinkName)
    .name(sinkName);
```

https://github.com/GIScience/osmalert/blob/main/flinkjobjar/src/main/java/org/heigit/osmalert/flinkjobjar/AlertJob.java

Contribution.java

https://github.com/GIScience/osmalert/blob/main/flinkjobjar/src/main/java/org/heigit/osmalert/flinkjobjar/model/Contribution.java

Comparison

Flink CEP	Current Implementation
- New Technology	+ Using existing possibilities
O Necessary to define pattern for filtering	O Necessary to write functions for filtering
- Multiple Outputstreams can be necessary	+ Can work on a single output stream

Sources

1. https://nightlies.apache.org/flink/flink-docs-master/docs/libs/cep/
2. https://github.com/GIScience/osmalert