【Flink】（十三）Flink CEP Library 使用案例分析

写在前面：我是「云祁」，一枚热爱技术、会写诗的大数据开发猿。昵称来源于王安石诗中一句 [云之祁祁，或雨于渊]，甚是喜欢。

写博客一方面是对自己学习的一点点总结及记录，另一方面则是希望能够帮助更多对大数据感兴趣的朋友。如果你也对 数据中台、数据建模、数据分析以及 Flink/Spark/Hadoop/数仓开发 感兴趣，可以关注我 https://blog.csdn.net/BeiisBei ，让我们一起挖掘数据的价值~

文章目录

一、前言

根据Flink CEP library来监控数据中心中每个机柜的温度。当在一定的时间内，如果有2个连续的Event中的温度超过设置的阈值时，就产生一条警告；一条警告也许还不是很坏的结果，但是如果我们在同一个机柜上连续看到2条这种警告，这种情况比较严重了。所以根据第一个警告流的输出，通过定义另一个Pattern，以上一步的输出作为第二个pattern的输入，来定义一个“严重”的问题。

二、CEPTest

```
package cep;


import org.apache.flink.cep.CEP;
import org.apache.flink.cep.PatternSelectFunction;
import org.apache.flink.cep.pattern.Pattern;
import org.apache.flink.cep.pattern.conditions.SimpleCondition;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.api.windowing.time.Time;

import java.util.List;
import java.util.Map;

public class CEPTest {

    public static void main(String[] args) throws Exception {
        StreamExecutionEnvironment env =
```

```
StreamExecutionEnvironment.getExecutionEnvironment();
        env.setParallelism(1);

        // DataStream : source
        DataStream<TemperatureEvent> inputEventStream = env.fromElements(new TemperatureEvent("xyz", 22.0),
                new TemperatureEvent("xyz", 20.1), new TemperatureEvent("xyz", 21.1),
                new TemperatureEvent("xyz", 22.2), new TemperatureEvent("xyz", 22.1),
                new TemperatureEvent("xyz", 22.3), new TemperatureEvent("xyz", 22.1),
                new TemperatureEvent("xyz", 22.4), new TemperatureEvent("xyz", 22.7),
                new TemperatureEvent("xyz", 27.0), new TemperatureEvent("xyz", 30.0));

        // 定义Pattern，检查10秒钟内温度是否高于26度
        Pattern<TemperatureEvent, ?> warningPattern = Pattern.<TemperatureEvent>begin("start")
                .subtype(TemperatureEvent.class)
                .where(new SimpleCondition<TemperatureEvent>() {

                    public boolean filter(TemperatureEvent subEvent) {
                        if (subEvent.getTemperature() >= 26.0) {
                            return true;
                        }
                        return false;
                    }
                })
                .within(Time.seconds(10));

        //匹配pattern并select事件,符合条件的发生警告，即输出
        DataStream<Alert> patternStream = CEP.pattern(inputEventStream, warningPattern)
                .select(
                        new PatternSelectFunction<TemperatureEvent, Alert>() {
                            @Override
                            public Alert select(Map<String, List<TemperatureEvent>> event) throws Exception {
                                return new Alert("Temperature Rise Detected: " + event.get("start") + " on machine name: " + event.get("start"));

                            }
                        });

        patternStream.print();

        env.execute("CEP on Temperature Sensor");

    }
}


```

三、Alert

```
package cep;

public class Alert {

```

```
    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public Alert(String message) {
        this.message = message;
    }

    @Override
    public String toString() {
        return "Alert [message=" + message + "]";
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((message == null) ? 0 : message.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if(this == obj) return true;
        if(obj == null) return false;
        if(getClass() != obj.getClass()) return false;

        Alert other = (Alert) obj;

        if(message == null) {
            if(other.message != null) {
                return false;
            }else if(!message.equals(other.message)) {
                return false;
            }
        }
        return true;
    }
}

```

四、MonitoringEvent

```
package cep;

public abstract class MonitoringEvent {
    private String machineName;

    public String getMachineName() {
```

```
        return machineName;
    }

    public void setMachineName(String machineName) {
        this.machineName = machineName;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((machineName == null) ? 0 : machineName.hashCode());

        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if(this == obj) return true;
        if(obj == null) return false;
        if(getClass() != obj.getClass()) return false;
        MonitoringEvent other = (MonitoringEvent) obj;
        if(machineName == null) {
            if(other.machineName != null) {
                return false;
            }else if(!machineName.equals(other.machineName)) {
                return false;
            }
        }
        return true;
    }

    public MonitoringEvent(String machineName) {
        super();
        this.machineName = machineName;
    }
}

```

五、TemperatureEvent

```
package cep;

public class TemperatureEvent extends MonitoringEvent{

    public TemperatureEvent(String machineName) {
        super(machineName);
    }

    private double temperature;

    public double getTemperature() {
        return temperature;
    }

```

```
15      public void setTemperature(double temperature) {
16          this.temperature = temperature;
17      }
18
19      @Override
20      public int hashCode() {
21          final int prime = 31;
22          int result = super.hashCode();
23          long temp;
24          temp = Double.doubleToLongBits(temperature);
25          result = (int) (prime * result +(temp ^ (temp >>> 32)));
26
27          return result;
28      }
29
30      @Override
31      public boolean equals(Object obj) {
32          if(this == obj) return true;
33          if(!super.equals(obj)) return false;
34          if(getClass() != obj.getClass()) return false;
35
36          TemperatureEvent other = (TemperatureEvent) obj;
37          if(Double.doubleToLongBits(temperature) !=
    Double.doubleToLongBits(other.temperature)) return false;
38          return true;
39      }
40
41      @Override
42      public String toString() {
43          return "TemperatureEvent [getTemperature()=" + getTemperature() + ",
    getMachineName=" + getTemperature() + "]";
44      }
45
46      public TemperatureEvent(String machineName, double temperature) {
47          super(machineName);
48          this.temperature = temperature;
49      }
50
51  }
52
```