flink 自定义触发器实现带超时时间的 Count Window

作者：LyZane
原文链接：https://ld246.com/article/1556522275597
来源网站：链滴

flink 的 window 有两个基本款，TimeWindow 和 CountWindow。

TimeWindow 是到时间就触发窗口，CountWindow 是到数量就触发。

如果我需要到时间就触发，并且到时间之前如果已经积累了足够数量的数据；或者在限定时间内没有累足够数量的数据，我依然希望触发窗口业务，那么就需要自定义触发器。

```
import org.apache.flink.api.common.functions.ReduceFunction;
import org.apache.flink.api.common.state.ReducingState;
import org.apache.flink.api.common.state.ReducingStateDescriptor;
import org.apache.flink.api.common.typeutils.base.LongSerializer;
import org.apache.flink.streaming.api.TimeCharacteristic;
import org.apache.flink.streaming.api.windowing.triggers.Trigger;
import org.apache.flink.streaming.api.windowing.triggers.TriggerResult;
import org.apache.flink.streaming.api.windowing.windows.TimeWindow;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * 带超时的计数窗口触发器
 *
 * @author Zane
 * @date 2019-04-26 15:57
 */
public class CountTriggerWithTimeout<T> extends Trigger<T, TimeWindow> {
    private static Logger LOG = LoggerFactory.getLogger(CountTriggerWithTimeout.class);

    /**
     * 窗口最大数据量
     */
    private int maxCount;
    /**
```

```
     * event time / process time
     */
    private TimeCharacteristic timeType;
    /**
     * 用于储存窗口当前数据量的状态对象
     */
    private ReducingStateDescriptor<Long> countStateDescriptor =
            new ReducingStateDescriptor("counter", new Sum(), LongSerializer.INSTANCE);


    public CountTriggerWithTimeout(int maxCount, TimeCharacteristic timeType) {

        this.maxCount = maxCount;
        this.timeType = timeType;
    }


    private TriggerResult fireAndPurge(TimeWindow window, TriggerContext ctx) throws Excep
ion {
        clear(window, ctx);
        return TriggerResult.FIRE_AND_PURGE;
    }


    @Override
    public TriggerResult onElement(T element, long timestamp, TimeWindow window, Trigger
ontext ctx) throws Exception {
        ReducingState<Long> countState = ctx.getPartitionedState(countStateDescriptor);
        countState.add(1L);

        if (countState.get() >= maxCount) {
            LOG.info("fire with count: " + countState.get());
            return fireAndPurge(window, ctx);
        }
        if (timestamp >= window.getEnd()) {
            LOG.info("fire with tiem: " + timestamp);
            return fireAndPurge(window, ctx);
        } else {
            return TriggerResult.CONTINUE;
        }
    }

    @Override
    public TriggerResult onProcessingTime(long time, TimeWindow window, TriggerContext ctx
 throws Exception {
        if (timeType != TimeCharacteristic.ProcessingTime) {
            return TriggerResult.CONTINUE;
        }

        if (time >= window.getEnd()) {
            return TriggerResult.CONTINUE;
        } else {
            LOG.info("fire with process tiem: " + time);
            return fireAndPurge(window, ctx);
```

```
        }
    }

    @Override
    public TriggerResult onEventTime(long time, TimeWindow window, TriggerContext ctx) thr
ws Exception {
        if (timeType != TimeCharacteristic.EventTime) {
            return TriggerResult.CONTINUE;
        }

        if (time >= window.getEnd()) {
            return TriggerResult.CONTINUE;
        } else {
            LOG.info("fire with event tiem: " + time);
            return fireAndPurge(window, ctx);
        }
    }

    @Override
    public void clear(TimeWindow window, TriggerContext ctx) throws Exception {
        ReducingState<Long> countState = ctx.getPartitionedState(countStateDescriptor);
        countState.clear();
    }

    /**
     * 计数方法
     */
    class Sum implements ReduceFunction<Long> {

        @Override
        public Long reduce(Long value1, Long value2) throws Exception {
            return value1 + value2;
        }
    }
}
```


使用示例（超时时间 10 秒，数据量上限 1000）：

```
stream
        .timeWindowAll(Time.seconds(10))
        .trigger(
                new CountTriggerWithTimeout(1000, TimeCharacteristic.ProcessingTime)
        )
        .process(new XxxxWindowProcessFunction())
        .addSink(new XxxSinkFunction())
        .name("Xxx");
```


关于 window 的介绍顺手贴两篇文章：

- simple-flink window
- Flink 原理与实现：Window 机制