Flink CEP – Complex Event Processing with Flink

1. Objective – Flink CEP

So, in this tutorial on Complex Event Processing with **Apache Flink** will help you in understanding Flink CEP library, how Flink CEP programs are written using Pattern API. Moreover, we will see various Flink CEP pattern operations with syntax, Pattern detection in CEP and advantages of CEP operations in Flink. Also, we will learn Flink Complex Event Processing use cases and examples to get in-depth knowledge of Complex Event Processing for Flink.

Flink CEP – Complex Event Processing with Flink

2. What is Complex Event Processing with Apache Flink

With the increasing size of data and smart devices continuously collecting more and more data, there is a challenge to analyze this growing stream of data in near real-time for reacting quickly to changing trends or for delivering up to date business intelligence which can decide company's success or failure. Detection of event patterns in data streams is a key problem in real-time processing.

Flink handles this problem through Complex event processing (CEP) library that addresses this problem of matching the incoming events against a pattern to produce complex events which are derived from the input events. CEP executes relevant data on a stored query unlike traditional RDBMSs and discards irrelevant data. This enables CEP queries to be applied on a potentially infinite stream of data and also enables inputs to be processed immediately. This aspect effectively leads to CEP's real-time analytics capability. This gives the opportunity to quickly get hold of what's really important in data. In this manner, Flink CEP is 1 of the key component of **Apache Flink ecosystem**.

Complex Event Processing with Apache Flink

Apache Flink is a natural fit for CEP workloads due to its true streaming nature and its capabilities for low latency as well as high throughput stream processing. Consequently, CEP has found application in a wide variety of use cases as described below and has provided several **features of Apache Flink** that have created a huge **difference between Apache Flink, Hadoop** and **Apache Spark**.

3. Pattern API in Flink

Flink CEP program can be written using the pattern API that allows defining complex event patterns. So, each pattern consists of multiple stages or states. The pattern needs to start with initial state and then to go from one state to the next, the user can specify conditions. Now, each state must have a unique name to identify the matched events later on. So, we can append states to detect complex patterns.

Flink CEP – Pattern API in Apache Flink

Do you know different pattern operations? Let us see some of the most commonly used CEP pattern operations:

a. Begin

It defines pattern starting state and is written as below:

```
Pattern<Event, ?> start = Pattern.<Event>begin("start");
```

b. Next

It appends a new pattern state and matching event need to succeed the previous matching pattern as below:

```
Pattern<Event, ?> next = start.next("next");
```

c. FollowedBy

It appends a new pattern state but here other events can occur between 2 matching events as below:

```
Pattern<Event, ?> followedBy = start.followedBy("next");
```

d. Where

It defines a filter condition for current pattern state and if the event passes the filter, it can match the state as below:

```
patternState.where(new FilterFunction <Event>() {
@Override
public boolean filter(Event value) throws Exception {
return ... // some condition
}
});
```

e. Or

It adds a new filter condition which is ORed with an existing filter condition. Only if an event passes the filter condition, it can match the state.

f. Within

It defines the maximum time interval for an event sequence to match the pattern post which it discards. We can write it as:

```
patternState.within(Time.seconds(10));
```

4. Pattern Detection

So, we need to create pattern stream to run stream of events as below using input stream and a pattern:

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

5. Use cases for Flink CEP

Apache Flink CEP is used for the large number of applications like for financial applications such as stock market trend, credit card fraud detection and RFID-based tracking and monitoring. It also finds its usage in detecting network intrusion by specifying patterns of suspicious user behavior.
Refer Flink use case tutorial to get real-time use cases of Apache Flink and how industries are using Flink for their various purposes.

6. Conclusion – Flink CEP

Flink CEP includes various challenges like:

- It has the ability to achieve high throughput and low latency processing
- Ability to produce the results as soon as the input event stream is available
- Ability to provide aggregation over time, timeout between two events of interest and other computations
- To provide real-time alerts & notifications on detection of complex event patterns

So, this was all in Flink Complex Event Processing. Still, if you face any issue in learning Flink CEP, you can freely ask us through comments.

Reference for Flink