Comparison Tests of Siddhi and Esper

Qunzhi

May 24, 2011

1. Query Language

Siddhi does not implement its query language yet, Esper uses EPL whose syntax is similar to SQL. In the following examples all queries are written in EPL and are applied over a single space temperature set point event stream.

SpaceTempSetpointEvent = <roomed, roomType, setpoint, reading, time>

2. Event Algebra

2.1 Selection Operator

The current version of Siddhi supports simple filter specifications using condition operators like “greater than”, “equal” and so on (Query 1). In Esper, selection filters can be defined as arbitrary propositional formulas such as mathematic and logic expressions (Query 2).

```
//selection with a simple condition
String query1 = "select roomId, roomType, setpoint, reading, time " +
                "from SpaceTempSetpointEvent " +
                "where roomType.name() = 'Office'";

//selection with more expressive propositional formulas
String query2 = "select roomId, roomType, setpoint, time " +
                "from SpaceTempSetpointEvent " +
                "where setpoint - reading > 10";
```

2.2 Aggregation Operator

Siddhi provides a set of aggregation functions such as *average, count* but does not support conditions on the aggregation functions (Query 3).

```
//constraints on aggregation functions
String query3 = "select roomType, avg(setpoint) " +
                "from SpaceTempSetpointEvent.win:time(5 seconds) " +
                "group by roomType " +
                "having avg(setpoint) > 60";
```

2.3 Renaming and Projection Operators

These two operators can be used to create new event types from source events. Esper supports these two operators and enables to insert new events into input event streams (Query 4, 5). I think in Siddhi, we can implement the *outputstreamhandler* to realize similar functions.

```
//renaming and projection
String query4 = "insert into OfficeSpaceTempSetpointStream " +
                "select roomId, reading AS measurement, time " +
                "from SpaceTempSetpointEvent " +
                "where roomType.name() = 'Office'";

String query5 = "select roomId, measurement, time " +
                "from OfficeSpaceTempSetpointStream " +
                "where measurement < 50";
```

2.4 Derived Operators

As I tested, the state machine based Sequential pattern (SEQ) does not work in Siddhi (release-0.2).

The current version of Siddhi only supports sliding window specifications, but not tumbling windows (Query 8). Siddhi also does not support event quantifications in sequential patterns (Query 9).

```
//sequence with tumbling time window
String query8 = "select * from SpaceTempSetpointEvent.win:time_batch(5 seconds) "+
                "match_recognize ( " +
                "partition by roomId "+
                "measures A.roomId as roomId, A.reading as reading1, B.reading as
reading2 "+
                "pattern (A B) "+
                "define "+
                "B as B.reading >= A.reading)";

//sequence with kleenee closure quantification
String query9 = "select * from SpaceTempSetpointEvent.win:time_batch(5 seconds) "+
                "match_recognize ( " +
                "partition by roomId "+
                "measures A.roomId as roomId, A.reading as reading1,
first(B.reading) as reading2, last(B.reading) as reading3, C.reading as reading4 "+
                "pattern (A B+ C) "+
                "define "+
                "B as B.reading >= A.reading, "+
                "C as C.reading < 75)";
```

2.5 Selection and consumption policies

Selection and consumption policies for eliminating redundant or duplicate events are not supported in Siddhi and not mentioned in the design document. Esper supports some types of selection policies such as *first/last* (Query 6, 7)

```
//most recent, first selection policies
String query6 = "select * " +
    "from SpaceTempSetpointEvent.win:time(5 seconds).std:unique(roomId)";

String query7 = "select * " +
    "from SpaceTempSetpointEvent.win:time(5 seconds).std:firstunique(roomId)";
```