Design Document for DTESP

(General Integration Data Turbine and Esper)

Heejin Choi

1. Purpose of subsystems
   1. Introduction

Task of DTESP consists of

1. Reading configuration file🡪LoadXml(), Item classes (SourceItem, SinkItem,… etc)
2. Data Turbine initialization and connecting🡪Init\_DT()
3. Esper initialization🡪Init\_Esper()
4. Reading from Data Turbine and sending it Esper  
   🡪Fetch(), class ChannelIndexSortedByTime, class ReceivedDataFromChannel
5. Listens to result of Esper queries and send to Data Turbine🡪 EsperEventListener class

Codes for these task are divided into methods involving some inner classes, for instance, Reading configuration file is done at LoadXml() involving inner classes such as SourceItem, SinkItem, and etc.

1. Reading configuration file  
   At method LoadXml(), it reads xml file and parse data into corresponding data type, and add to list. The data will be needed for future initialization of Esper or Data Turbine. For instance, <Source> node will create SourceItem class and read the xml file and set the fields and added to list.
2. Data Turbine initialization and connecting  
   At method Init\_DT(), it iterates through data classes created by parsing xml file and setups Data Turbine connection. For instance, it iterates through all SourceItem from the list which is created by II and creates Source class and connect to Data Turbine server.
3. Esper initialization  
   At method Init\_Esper(), it iterates through data classes created by parsing xml file and setups Esper. For instance, it iterates through all QueryItem from the list which is created by II and creates Esper Queries.
4. Reading from Data Turbine and sending it Esper  
   At method Fetch(), it reads the sink channels and data from each channel is stored as one inner class ReceivedDataFromChannel. To send data to Esper in ascending time order, every earliest data of ReceivedDataFromChannel is inserted to inner class ChannelIndexSortedByTime which behaves as a sorted list. It always sends the first data in ChannelIndexSortedByTime so that data is sent in ascending time order over all channels.
5. Listens to result of Esper queries and send to Data Turbine

EsperEventListener class is responsible for the task. When creating query at Ⅳ, if the result of the query need to be sent to Data Tubine, EsperEventListener is attached for each query for this task.

1. Entity diagram

ReceivedDataFromChannel, and ChannelIndexSortedByTime class is used in Fetch() method to send data in ascending time order.

EsperEventListener class is created in Init\_Esper() method to receive result from esper and send to Data Turbine.

SourceItem, SinkItem, SourceChannelItem, SinkChannelItem, QueryItem, and EventItem represents xml configuration.

ChannelIndexSortedByTime

void Clear()

void Add(double time, int cindex)

int PopFirstChannelIndex()

boolean IsEmpty()

ReceivedDataFromChannel

boolean bIsempty()

double GetData()

double GetTime()

void Next()

void Clear()

SinkChannelItem

SinkItem

EventItem

SourceChannelItem

QueryItem

SourceItem

EsperEventListener

Update()

DTESP

void LoadXml()

void Init\_DT()

void Init\_Esper()

void Fetch()

1. Interaction

[Example of <Source> node]

Add to list

Parse XML and create

SourceItem

dtesper.LoadXml()

EsperEventListener

QueryItem

dtesper.Init\_Esper()

[Example QueryItem and EsperEventListener in Init\_Esper method]

Attach event listener

Create Esper query

Put next data of the ReceivedDataFromChannel which the popped data was from

Send to Esper

Pop first data of ChannelIndexSortedByTime which is the earliest data of overall channel

Put every earliest data and index of ReceivedDataFromChannel pair

to ChannelIndexSortedByTime

Read data from each channel and data from each channel is save at ReceivedDataFromChannel

dtesper.Fetch()

ChannelIndexSortedByTime

ReceivedDataFromChannel

[Example ReceivedDataFromChannel and ChannelIndexSortedByTime used in Fetch() method]