Complex Event Processing Module in IOT Resource Access and Intelligent Processing Platform

Yang LIU and Chuan-chang LIU

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Keywords: Configurable CEP, IOT, Drools, Real time analysis.

Abstract. The enormous number of sensors and smart objects being conveyed in the IOT (Internet of Things) represent the potential for IOT platform to identify and respond to live-circumstances. For utilizing this hidden potential, CEP (Complex Event Processing) intends to effectively distinguish event patterns (Complex Event) in the real time sensor stream and thusly help in understanding a ‘distributed intelligence’ in the Internet of Things. IOT Resource Access and Intelligent Processing Platform aims at providing a general platform to access different sensors and smart objects, distribute messages around heterogeneous systems and transform low-level data streams to high-level ones, which complex event processing module is responsible for. Therefore, apart from the basic demand to process real time event stream with low latency and high throughput, this module is also designed to access and create event streams dynamically, manage event streams and processing logistic at runtime and trigger business rule or work flow.

Introduction

As the rapid development of Internet of Things, its applications are sprung up in more and more fields. There have been diverse systems in the field of sensor networks, including IOT monitoring systems, intelligent home system, and so on. And they are all providing monitoring and intelligent services via physical sensing equipment, but varies from each other by the specific business logistics. Therefore, the IOT Resource Access and Intelligent Processing Platform, which provide general features and configurable business logistics for all the IOT applications comes into being.

Complex event processing module in IOT platform benefits its users to make decision based on real-time information and historical data. Observations from sensors are not only useful for recognizing the patterns on ongoing events, but also for learning routines and creating a specific profile detailing daily activities that can be used to analyze history log and deepen the understanding of current business scenario. This challenge can be summarized by the following questions: how do we take raw data from sensors, filter it, aggregate it and then transform that into relevant information that can be associated with business logistic.

Complex event processing (CEP) is a technological method for tracking and analyzing streams of data to derive a conclusion from the data, which is used to predict high-level events likely to result from specific sets of low-level factors [1]. CEP has already been found in many applications, mostly in the fields of stock market data analysis and real time databases. Its event-based nature makes it a fitting paradigm for sensor processing work.

Firstly, this paper will recall how the complex event processing module works and cooperates with other modules. Then this paper will provide some backgrounds on the whole system and CEP module particularly, and show how to get all parts work. Finally, this paper will report on the use of an existing event stream processor to successfully implement the complete chain, from low-level sensor data up to a sequence of discrete and high-level actions.
System Architecture

There is diverse equipment in IOT Resource Access and Intelligent Processing Platform, including temperature and humidity sensors, electromagnetic spectrum, electron cameras and others even after the system has been put into use. These physical devices will feed their detected data into system continuously. Then system will derive dangerous or target events out of massive real time data by user-defined rule, and transport target events to upper applications. As a result, this platform can realize full range monitoring in the target area. Moreover, it can detect dangerous events in real time and dynamically.

This platform’s key functional requirements are as follows:

- Collect sundry smart objects’ information, mainly including RS232 serial port equipment, TCP / UDP gateway equipment.
- Realize real time monitoring on the business scenario, and derive high-level events out of massive low-level events. In addition, it ought to be flexible to fit a variety of complex business scenarios, and provide a convenient, automated dynamic event management interface.
- Supports distributed data communication mechanism.

Based on different functions, this platform is divided in 3 individual parts: resource access module, pub-sub module and complex event processing module. Its System architecture diagram is shown below:

![System Architecture Diagram](image)

- Resource Access Module is responsible for resolving physical device according to its own protocols, structuring raw data and pub the structured data to pub-sub module.
- Pub-Sub Module is based on distributed event dispatch mechanism, and is responsible for the real time data exchange between each module.
- Complex Event Processing Module, which this paper focuses on, is responsible for deriving high-level event from high-speed real-time event stream and reporting target event stream to pub-sub module as well.
Following two sub parts will describe the main techniques used in Pub-Sub Module and Complex Event Processing Module in detail.

**Pub Sub Module**

In software architecture, publish–subscribe is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead characterize published messages into classes without knowledge of which subscribers, if any, there may be. Similarly, subscribers express interest in one or more classes and only receive messages that are of interest, without knowledge of which publishers, if any, there are.[2]

In the publish–subscribe model, subscribers typically receive only a subset of the total messages published. The process of selecting messages for reception and processing is called filtering. There are two common forms of filtering: topic-based and content-based.

- In a topic-based system, messages are published to "topics" or named logical channels. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe, and all subscribers to a topic will receive the same messages.
- In a content-based system, messages are only delivered to a subscriber if the attributes or content of those messages match constraints defined by the subscriber. The subscriber is responsible for classifying the messages.

In this IOT platform, publishers, like sensors, publish its data and same kind of sensors usually has the same form of data. And subscribers, like Complex Event Processing Module, focus on the data of some certain sensors so that they can be processed further. Therefore, topic-based pattern is more suitable for the IOT platform.

**Complex Event Processing Module**

The challenge is to modify the data so that it can be used as a basis for decision-making. The traditional way where information is added to a database and later accessed as reports, does not meet the business challenge today. Real-time solutions are needed. Instead of first storing data and then put queries to it, a query is created and data is then run through each query. With CEP, queries can be stored, other than the data. CEP can process each event in real time, and emit results when query criteria are met, without actually needing to store data. Next sub parts will define some key concepts and techniques used in CEP.

**Event.** Event is an activity that has happened in real life or in an information system. An event can also occur as a result of other events [3]. An event has three aspects:

- Form: the form of an event is an object
- Significance: an event signifies an activity
- Relativity: an activity is related to other activities by time, causality and aggregation,

A complex event is an aggregation of other events, which are called members of events. An aggregation is the relationship between a complex event and its members. In an event processing applications, complex events are events at high level than the levels of its members. When going to higher level, events are filtered, constrained and aggregated. The number of event get smaller and events, likewise, become more abstract. [3]

**EPL.** The Event Processing Language (EPL) is a generic term for a programming language that enables the detection of complex event in a system. Most EPLs are similar to the Structured Query Language (SQL) with clauses like SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY. EPL statements are used to derive and aggregate information from streams of events, and join or merge event streams.

Apparently, different business logistics can be described by different EPLs, and this is the main disparity among diverse information systems. So this CEP Module will provide a user interface for users to create any suitable EPLS according to the business scenario, and rest work will be triggered automatically. In the end, a business-irrelevant, configurable platform in finished.
Esper. Esper is a component for complex event processing (CEP) and event series analysis. It and EPL provide a highly scalable, memory-efficient, in-memory computing, SQL-standard, minimal latency, real-time streaming-capable Big Data processing engine for any-velocity online and real-time arriving data and high-variety data, as well as for historical event analysis. [4]

Its Architecture is as below:

![Esper Architecture Diagram](image)

Input data of this CEP Module is high-speed real time event stream. Input adapter will convert each input event to java bean and send it to the Esper Core Engine. Esper Core Engine will be responsible for registering to-be-listened event, monitoring and process these events. Only when some rules are met, event listener will be triggered immediately and call Output Adapter to package and send the output event afterwards.

**Design and Implementation**

As shown in figure 1, Complex Event Processing Module contains 4 sub parts, which are:

- Data Source part provides interfaces to maintain (download/create) data template from Pub-Sub Module and registers event to the CEP engine.
- Rule-Configure part provides interfaces for users to create specific EPL rules according to its business logistic, and generate event listener automatically based on the binding relationship of input event stream and triggered output stream.
- Multi-User Management part provides event management, EPL rule management and real time monitoring for users at runtime.
- Business Rule Engine part enable users to trigger web service or workflow as soon as an output event is generated.

Following parts will describe the design and implementation of each sub modules in detail.

**Data Source part.** Firstly, this part should clarify the event topics, which users are interested in, and then download these from Pub-Sub Module. The system will know how to parse different forms of event stream. Actually, there is no explicit boundary between input and output event. CEP Module can accept both real event stream, like what physical devices emit, and virtual event stream, which is generated as an output stream by other real or virtual ones.

To enable the automagical parse, this part converts schema file, which is an abstraction of some specific physical equipment data and is downloaded from Pub-Sub Module, to corresponding Java class, finish the compiling work at runtime and register the event to Esper engine. As a result, the system can understand each event and manage events anytime, no matter what the data format is.

**Rule-Configure part.** This part provides GUI to define and manage EPL rules, which support complex and different business scenario. Based on the pre-defined output event, this part will bind input and output event stream by EPL rule and generate corresponding listener automatically.
When an EPL rule is started, this part will subscribe event topics from Pub-Sub Module, convert received event to engine-understanding java bean and send the java beans to the core engine. In aspect of data format, the format of data subscribed from Pub-Sub Module is JSON or XML, as mentioned before. While Esper supports 4 kinds of data format, that is POJO, Java.Util.Map, Object Array and XML. Due to the worse performance of XML according to official document, POJO is selected as the data format used in this part. In aspect of performance, there is no double that high-speed real time event streams will lead to heavy burden of converting work. Any delay in converting work may result in the disorder when processing event streams. Therefore, multi-thread technology is adopted to guarantee the sequential order and interval of events.

When a rule is met, event’s corresponding listener will be triggered, and the generated output event will be parsed to schema file and published to Pub-Sub Module so that it can be subscribed by others.

Multi-User Management part. CEP Module is designed as C/S pattern to ensure CRUD operations and system’s regular service simultaneously. Esper Core Engine, Data Source and Rule-Configure Module are set in server side to guarantee that system service won’t be interrupted. Client side is responsible for:

- Event Management includes event CRUD, event start/stop and event destroy functions.
- EPL Configuration is responsible for define EPL rule based on templates and delete.
- Running Status Monitoring include monitoring based on event, EPL rule and CPU usage.

To simplify communication between server and client, Apache Mina is chosen. Apache MINA is a network application framework which helps users develop high performance and high scalability network applications easily. It provides an abstract event-driven asynchronous API over various transports such as TCP/IP and UDP/IP via Java NIO. [5]

Business Rule Engine. A business rule system enables these company policies and other operational decisions to be defined, tested, executed and maintained separately from application code. CEP Module use Drools (a rule engine) to define and trigger workflow after an output event is generated. The communication is through Pub-Sub Module, too. Main steps are described below:

- Add topic’s session information in configuration file.
- Configure specific rules in .drl file for each topic. Operations triggered by each rule include but not limited to remote web service and workflow.
- Subscribe target topics from Pub-Sub and parse data to corresponding object in Drools.

Summary

CEP framework can support IOT applications in identifying circumstance in the around scene. This CEP Module in IOT Resource Access and Intelligent Processing Platform realize the core task of deriving high-level data from high-speed real time low-level data, support configurable EPL rules and management work for users to create applications flexible to any business scenario. In the future work, CEP Module should provide more transparent EPL rule configuration GUI to lower down the difficult for normal customers, and improve performance in low latency and high parallel computing.

References