I.J. Education and Management Engineering, **2013, 1, 7-13**
Published Online January 2013 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2013.01.02

Available online at http://www.mecs-press.net/ijeme

Event Stream Processing in BRT Environment

Wenfeng Liu

The Department of Computer Science, Beihang University Beijing，China

Abstract

Bus Rapid Transit (BRT) system is a kind of Intelligent Transportation System (ITS), which combines wireless communications, computational technologies, floating car data/floating cellular data technologies, inductive loop detection and so on. The GPS (Global Position System) data and the loop detection data are arriving at the BRT dispatch center massively at the same time so the dispatcher can know the real-time status of buses. The passenger flow data are also captured by devices deployed in the stations and then be passed to center too. How to handle those data at low latency and get some events to be used in bus dispatching is really a problem. In this paper, we propose a framework using ESP (event stream processing) technologies that solves the problem more effectively. At last we demonstrate a prototype system implementation that applies in BRT Line 1 project in Beijing.

Index Terms: BRT; ITS; Event Stream Processing; Public Bus Dispatching

1. INTRODUCTION

In BRT, we want to monitor the buses' position data at real time, and it usually adopts Automatic Vehicle Location (AVL) system to locate the car. It involves GPS, RFID (Radio Frequency Identification Devices) or other technologies. Through these position data, we can analysis the running status of these buses, and detect some abnormal events that will influence the efficiency of this public transportation service. We can also get the passenger flow data by using AFC (Auto Fair Collection) system.

These location data of these buses and the passenger flow data arriving at BRT center on a high-volume level. The meanings lay in these data are very important to dispatcher. For example over time due to the increase of passenger flow or the traffic jam, the distance between buses will keep increasing. The dispatcher of BRT better be notified about these abnormal events, because it will affect the efficiency of BRT running and provide bad passenger experience. Then the dispatcher can do some dynamic scheduling work react to these abnormal events.

In above scenario, we need to deal with high-volume data with low-latency cost. In the next section, we will compare three different ways to process these data, and get a conclusion from these comparisons on which way we need to adopt.

* Corresponding author.
E-mail address: 266LWF@sse.buaa.edu.cn

2. Strem processing Methods

There are at least three different software system technologies that can potentially be applied to solve high-volume low-latency streaming problems. These are custom coding, Database Management Systems, and stream processing engines [1].

A. Traditional custom coding(Messaging)

Traditionally, custom coding has been used to solve high-volume, low-latency streaming processing problems. This approach is universally despised because of its inflexibility, high cost of development and maintenance, and slow response time to new feature requests, all because the process logic was highly coupling with code and hard to change. The communication between different modules is archived by sending messages. The meaning and the format of these messages are defined by programmer, there is no easy way to alter these messages and adapt new feature request. The basic architecture for messaging is shown in Fig. 1.

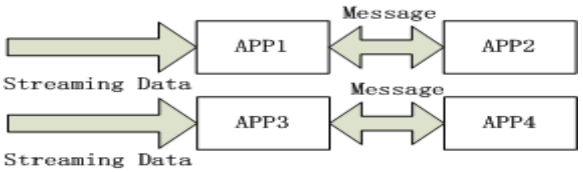

Figure 1. Message Passing Basic Architecture

B. Database Management Systems (DBMSs)

The DBMS are very popular in the last decades. Because they have the ability to reliably store massive data sets and efficiently process human-initiated queries. They use a "Process-after-store" model that means the data that we want to process are first stored, potentially indexed, and then get processed, which causes the latency is not that low. The data are kind of "static" once we throw it in a database, we can retrieve "dynamic" data using different query statements. The performance can be enforced via using Main-memory DBMSs which avoid going to disk for most operations, given sufficient main memory. The basic architecture for DBMSs is shown in Fig. 2.

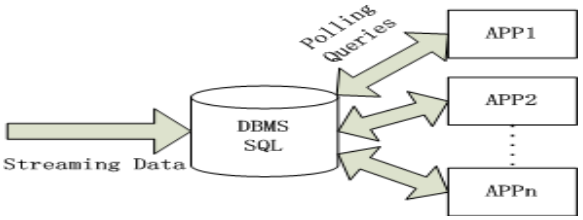

Figure 2. DBMS Basic Architecture

C. Event Stream Processing(ESP)

Event Stream Processing (ESP) systems target applications that produce data continuously and requires (soft) real-time processing. An ESP application consists in a directed acyclic graph of operators that are traversed by the data. Inside each operator, the pieces of data, called events, can be transformed, aggregated, combined, enriched, or filtered out [2]. In ESP engine the query statement is "static" while the data are "dynamic", they

flow into the ESP engine and get processed real-time based on the query engine, and the processed data which normally indicates complex events flow into the event sink afterward. The basic architecture for ESP is shown in Fig. 3.

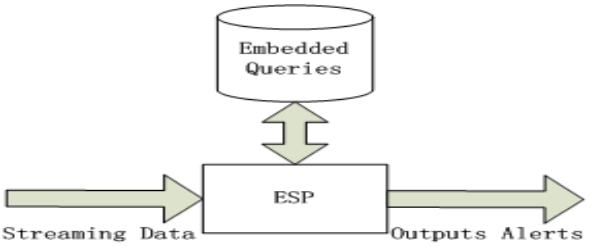

Figure 3. ESP Basic Architecture

D. The Comparison between three methods

The table1 compares above three technologies from query paradigm, latency, data rate, query semantics and degree of coupling.

TABLE I. COMPARISON BETWEEN THREE METHODS

Item	**Stream Process Method**		
	Messaging	***DBMS***	***ESP***
Query Paradigm	Programmer Defined	Ad-hoc queries or requests	Continuous standing queries
Latency	Milliseconds or less	Seconds, hours, days	Milliseconds or less
Data Rate	≥1.000 events/sec	≥ 100 events/sec	≥10.000 events/sec
Query Semantics	Temporal analytics	Declarative relational analytics	Declarative relational and temporal analytics
Degree of Coupling	Tight	Loose	Loose

From this table, we can see the solution of ESP has the advantages of Low-Latency, High Data-Rate and Loose- Coupling. In BRT environment, it is very important that we can monitor all the buses in real-time and react to the abnormal events. The ESP technology meets all the requirements.

3. Overview of event stream processing

A. Preliminary

In the model of event stream processing, we encounter three basic concepts:

Events. An event is defined to be an instantaneous occurrence of interest at a point in time [3]. In BRT, the events normally mean the GPS and RFID position data and the passenger flow data. They all have attributes. Take the GPS data for example: they contain the code of the bus, the longitude and latitude data, the instant speed and direction of the vehicle and the timestamp which records the time when the event is trigged.

Event Streaming. In most event processing scenarios, it is assumed that the input to the ESP engine is a potentially infinite event stream that contains all events that might be of interest [3, 4].

Event Query Statement. Event Query Statement are queries on the sequential event stream [5]. It defines a language that can specify how individual event is filtered and how multiple events are correlated via time-based and value-based constraints. It is usually implemented via SQL-Like syntax. The processed events usually indicate an alert or some situations that we are interested in.

B. The Query Paradigm

The query statements normally contain following operations [6]:

- **Projection.** The ESP engine performs specific computations over individual events as part of their transformation to output events.
- **Filtering.** The filter propagates the event to the output stream only if some specified conditions are satisfied.
- **Joins.** This operation correlate values from different streams. It compares each event from one input stream with each event from a second input stream. If their valid time intervals overlap and the join condition hold, the operation produces one output event.
- **Unions.** It combines events from different steams.
- **Group and Apply.** It partition events into event groups, then aggregations and other operations can be performed on the event groups.
- **Time Windows.** In applications that process real-time events, a common requirement is to perform some computation over subsets of events that fall within some period of time.

C. The Current Status of ESP

A wide range and ever growing numbers of applications nowadays, including network monitoring, e-business, health-care, financial analysis, and security supervision, rely on being able to process queries over data streams that take the form of time ordered series of events [7]. In the commercial domain, all the bigger software vendors (IBM, Oracle, Microsoft, Sybase, Progress Software, Software AG and TIBCO) own products in the event processing areas, getting it to the main stream of enterprise computing [8]. The open source communities are also very active in this filed too [9] [10].

4. ESP applied in BRT Environment

In previous sections we compared the differences between three methods in handling event stream, and gave a brief introduction to ESP. In this section, we will propose a framework using ESP technology to handle the abnormal events that happened in routine BRT running. We adopted Microsoft's StreamInsight platform [6] as our ESP Engine in BRT Line 1 project in Beijing.

A. Basic Framework

In BRT system, we can get the GPS data and RFID data of the running buses and the passenger flow data at real-time. Those data are transferred to the input adapters. The adapters translate and deliver incoming event streams to the ESP Engine. The ESP Engine processes those events based on the query logic that user define, they can be registered or canceled at run time. These queries take a potentially infinite feed of time-sensitive input data, perform some computation on the data, and output the result in an appropriate manner. During this

process, the ESP Engine may need some static data from Database. The output adapters work opposite to input adapters, they receive the events processed by the ESP Engine, translate the events into a format expected by the event sink. The event sink can be a device, database, GUI (Graphical User Interface), or any other programs as long as they can handle those output data with specific form. In our BRT environment, the event sinks are dashboard, an alert notification interface and database. The dashboard show the real-time status of buses and passenger flow, and the alert notification will send an alarm to dispatcher once an abnormal event occurs. The event record can be also sent to database, so we can review the historical event in the future. Fig. 4 demonstrates the basic framework of ESP in BRT.

B. Event Example

In this section, we will show some query logics that indicate some events in BRT system. Before we go further, we'd like to point out that the StreamInsight platform that we adopted in our system uses LINQ (Language Integrated Query) as its standing query statement. LINQ is a mechanism for expressing declarative queries over data sets that is fully integrated into a host language such as C#.

1) Detecting the over-speed behavior of the driver: The speed is very important to passengers' safety and comfort. A GPS event contains the bus's instant speed, so we can apply the Filter operation on those events and get the bus code that is running too fast. Here is the LINQ statement:

```
Var OverSpeed =
From gpsEvent In GPSStream
Where gpsEvent.Speed >SpeedThreshold
Select new {BusCode = gpsEvent.BusCode}
```

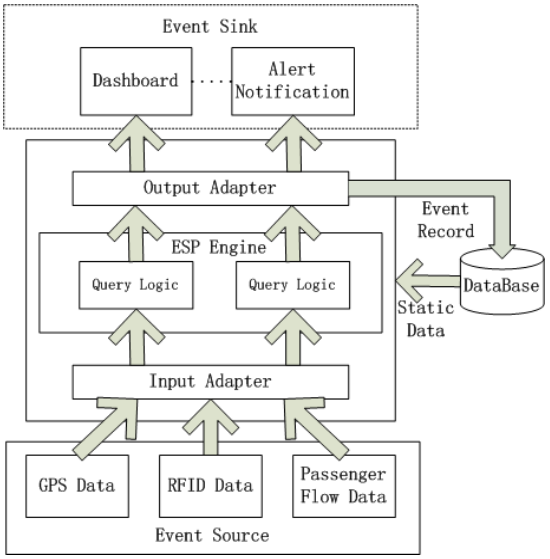

Figure 4. The Basic Framework of ESP in BRT

This query outputs the code of buses running too fast, we can monitor those buses and give some warning messages to the drivers once we detect the behavior.

2) Detecting the GPS working status: Normally the GPS device will send its position data every 10 seconds, but due to weather, bridge or other reasons, we may lose some GPS signals. The query we are about to show is outputting all the buses had sent their GPS data during the last 10 seconds.

3) Detecting Passenger Flow Abnormal: During some big celebrating moments or other occasions, the passenger number may increase rapidly. The dispatcher should be notified about this situation, and sent more buses to handle it. The query below shows how many people arrived at which station in last 1 minute with 1 second refresh frequency. We can pass the query result to another query which set a threshold value for person count.

```
Var GPSGoodBuses =
From gpsEvent In GPSStream.AlterEventDuration (e =>
                    TimeSpan.FromSeconds(10))
Group gpsEvent By  gpsEvent.BusCode Into oneGroup
From eventWindow  In
   oneGroup.SnapshotWindow(SnapshotWindowOutputP
   olicy.Clip)
Select new {BusCode = oneGroup.Key}
```

There are lots of other query logics in our BRT system. We only demonstrate these three queries for example to let you know how we write a query.

```
Var PFStatus =
From PFEvent In PFStream
Group PFEvent By  PFEvent.StationCode Into oneGroup
From window  In oneGroup.HoppingWindow(
        TimeSpan.FromMinutes(1),
        TimeSpan.FromSeconds(1),
        HoppingWindowOutputPolicy.ClipToWindowEnd)
Select new {PersonCount = window.Count(),
                StationCode = oneGroup.Key}
```

C. *Prototype Demonstration*

We already implemented a prototype system in BRT Line 1 project, Beijing. Fig. 5 shows the main form of this prototype.

Figure 5. The Main Form of BRT ESP System

This prototype we implemented has the ability to discover the abnormal events rapidly in this BRT environment.

5. conclusions

In this paper, we have proposed a new framework that using the ESP technology to process data in BRT environment. By using ESP technology, we could detect the over-speed behavior of the driver and the passenger flow abnormal, etc. Based on the framework, we have implemented a prototype that can detect some abnormal event of BRT. The result shows that it makes the BRT run more efficiently.

Acknowledgment

The author is thankful to his research advisor, Associate Prof. Guixi Xiong, for his kind guidance throughout this work. He is also grateful for all the helps from Yun Ma, Yanqin Xie and all other co-workers on this project.

References

[1] Stonebraker M,Cetintemel U,Zdonik S.The 8 requirements of real-time stream processing[J].ACM SIGMOD Record, 2005,34 (4):42-47.
[2] A.Brito, C.Fetzer, and P.Felber. Multithreading-enabled active replication for event stream processing operators. In S. Upadhyaya, editor, Proceedings of th 28th IEEE International Symposium on Reliable Distributed Systems, 2009. SRDS '09, pages 22-31. IEEE Computer Society, Sept. 2009.
[3] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In VLDB, pages 606–617, 1994.
[4] A. Adi and O. Etzion. Amit - the situation manager. VLDB J., 13(2):177–203, 2004..
[5] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In SIGMOD, pages 407–418, 2006..
[6] Microsoft StreamInsight Help Document, version 1.1.0. Microsoft Inc, 2010
[7] Ming Li , Mo Liu , Luping Ding , Elke A. Rundensteiner , Murali Mani, Event Stream Processing with Out-of-Order Data Arrival, Proceedings of the 27th International Conference on Distributed Computing Systems Workshops, p.67, June 22-29, 2007.
[8] Opher Etzion. Event Processing – past, present and future. The 36th International Conference on Very Large Data Bases, Septermber 13-17, 2010
[9] Thomas Bernhardt, Alexandre Vasseur, Esper: Event Stream Processing and Correlation,2007.03
[10] The Open Source Phasor Data Concentrator. http://openpdc.codeplex.com/