International Journal of Automation and Computing 16(1), February 2019, 52-64
DOI: 10.1007/s11633-018-1164-5

Optimization RFID-enabled Retail Store Management with Complex Event Processing

Shang-Lian Peng[1] Ci-Jian Liu[2] Jia He[1] Hong-Nian Yu[3,4] Fan Li[1]

[1] College of Computer Science and Technology, Chengdu University of Information Technology, Chengdu 610225, China

[2] The Southwest Jiaotong University (SWJTU)-Leeds Joint School, Southwest Jiaotong University, Chengdu 610041, China

[3] School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China

[4] Faculty of Science and Technology, Bournemouth University, Poole BH125BB, UK

Abstract: Radio frequency identification (RFID) enabled retail store management needs workflow optimization to facilitate real-time decision making. In this paper, complex event processing (CEP) based RFID-enabled retail store management is studied, particularly focusing on automated shelf replenishment decisions. We define different types of event queries to describe retailer store workflow action over the RFID data streams on multiple tagging levels (e.g., item level and container level). Non-deterministic finite automata (NFA) based evaluation models are used to detect event patterns. To manage pattern match results in the process of event detection, optimization algorithm is applied in the event model to share event detection results. A simulated RFID-enabled retail store is used to verify the effectiveness of the method, experiment results show that the algorithm is effective and could optimize retail store management workflow.

Keywords: Complex event processing (CEP), radio frequency identification (RFID), Internet of things, data stream, supply chain, retail store.

1 Introduction

In recent years, supply chain management has been changed rapidly due to the development of technologies and consumer needs. In the retail store scenario, management of stock based on contactless techniques such as radio frequency identification (RFID) would reduce out-of-stock problems for retailers and suppliers. It is reported that 8.3% of customers cannot find what they want while shopping, which would cause a great loss of revenue (about 4%) each year[1].

Out-of-stock (OOS) can be generally defined as the situation where consumers cannot find products they expect to buy on the shelf of a retail store during a shopping trip. In the literature, OOS is categorized into two cases: 1) the item or product is not in the store or in the warehouse; 2) the item or product is not on the corresponding shelf (out-of-shelf)[2, 3], but it is in the store, maybe in another shelf, another place, but the consumer cannot find the product in the right places.

RFID can be used to provide product visibility across the supply chains and logistics with its unique and automatic object identification capabilities[4]. RFID has gained wide applications in retail store vendors such as Wal-Mart and Metro[5].

In a RFID enabled supply chain[6, 7], RFID data can be generated in a stream way which leads to heavy load for the information processing engine to react in nearly real time. Complex event processing (CEP) is a real time computing paradigm which would detect patterns of interest over boundless event streams[8]. RFID CEP would provide real time product status insight over the event streams if the workflow is well managed within the information system.

Many research works of RFID in retail store have investigated the value of RFID to optimize shelf replenishment in a retail store with model selection and workflow reorganization[9–12]. In this paper, we focus on utilizing CEP over RFID event streams to provide real time retail store management decisions. First, we define retail store backroom and the sales floor flows of goods with event patterns over the streams. Then we present the corresponding event query execution model with an optimization method. Finally, simulation experiments are presented to verify the effectiveness of the proposed methods.

This paper is organized as follows. In Section 2, related works are discussed. Motivating example and assumptions are described in Section 3. Patterns are defined in Section 3. In Section 4, we present our CEP execution model and optimization algorithm. The simulation study is introduced in Section 5 and our conclusion is made in

Research Article
Special Issue on Addressing Global Challenges Through Automation and Computing
Manuscript received March 5, 2018; accepted November 14, 2018; published online December 27, 2018
Recommended by Associate Editor Jie Zhang

Section 6.

2 Related works

2.1 Complex event processing

CEP has been extensively studied in active databases and message-based systems in the past[13–19]. Unfortunately, these traditional frameworks cannot satisfy the requirements of RFID systems because their complex event specification does not support value-based constraints and sliding windows. Moreover, their implementations were based on fixed data structures, such as tree[10], directed graph[11], finite automata[16], or Petri net[15], which cannot flexibly accommodate the necessary extensions required by RFID CEP.

The non-deterministic finite automata (NFA) approach for RFID CEP was first proposed by [19], which also presented the optimization technique of evaluating value-based constraints and sliding windows earlier to reduce the size of intermediate results. The work[20] of tailored NFA framework from [12] is to facilitate the processing of RFID complex events defined at the granularity of objects. It also proposed several optimization techniques to reduce memory consumption and improve system throughput. Recently, the more general problem of pattern matching over event streams has also been investigated in [21–23]. Other RFID complex event processors developed include HiFi[24] and Siemens RFID middleware[25]. The HiFi system aims to support efficient aggregation of data generated by individual receptors distributed over a tree-structure network. The Siemens RFID middleware uses an event-based framework to transform raw RFID data into semantic data. Its event processing is implemented by a graph computation model. Neither the HiFi system nor Siemens RFID middleware provides necessary optimization techniques for huge-volume event processing. Ray et al.[26] optimizes nested complex event queries utilizing caching technique. Unfortunately, none of the works mentioned above considers the effect of limited memory on processing efficiency. Recent work of Qi et al.[27] addresses pattern aggregation during event detection. Instead of aggregating event patterns in a post processing step, they construct pattern aggregation online in the detection of event processing in linear time. However, computing aggregation and evaluating CEP queries together would lead to system overhead especially in a memory limited computing context.

Some works[28–34] study CEP in a cloud environment, big data and distributed computing scenarios. However, event processing in distributed environments is complicated to implement and needs to consider both memory usage and communication cost in order to balance the system load.

In [35], event-based control and filtering problems are categorized and discussed for event-based networked systems, which is similar to the model used in this paper.

2.2 RFID data warehousing

There are also some works[36–38] on warehousing and querying massive RFID data. These works assume the presence of a cleansed RFID database which records the durations items stay at different locations. The RFID path pattern query is then evaluated on the database through data warehousing techniques. Note that constructing such database requires nearly all RFID readings (except the redundant ones) to be recorded and the authors did not address its performance issues. In contrast, RFID complex event processing is directly executed over streaming RFID readings. It can effectively filter RFID events and only forwards meaningful complex events to backend databases.

2.3 Stream data processing

Another research area related to RFID CEP is streaming data management[39–41]. The proposed systems were mainly designed for relational data and their operations did not support some critical specifications of RFID complex events, such as sequence constraints and negation event. Cao and Rundensteiner[42] exploit intra-stream and inter-stream correlations to partition the query into an optimal query plan and obtain pruning opportunities to delete intermediate join results. The limited-memory optimization techniques proposed for streaming data processing[43–46] therefore cannot be applied directly in RFID CEP. There are also some works[47–49] on optimizing sequence queries in relational databases. Sequence query is different from CEP in that it targets a consecutive sequence of relational tuples and allows backtracking on the input during processing. Moreover, the proposed optimization techniques for sequence query were tailored for relational database management system (RDBMS) environment, thus not applicable to CEP.

Chow et al.[50] propose a real-time RFID based knowledge system to solve logistics process management problems. Input of the knowledge-based decision system would be some high level of complex events detected from the middleware level over RFID data streams. Zhu and Huang[51] aim to extract rules from data streams. Rules can be considered as high level of condition which can be also used in some Petri net based event detection engines.

2.4 RFID supply chain and retail store management

Retail store management utilizes automatic data collection techniques to accelerate process tracking. Different levels of tagging mean different kinds of management complexity and user experience. Choi et al.[52] use item-level RFID tagging methods to enhance customer shopping experience in retail stores. Customer shopping behavior is collected using RFID and fuzzy screening algorithms are utilized to analyze customers′ preferences. Condea et al.[53] consider RFID for retail store automatic shelf replenishment decisions based on RFID case-level tagging. Metzger et al.[54] compare the performance of different RFID-based shelf replenishment policies, they find that different levels of tagging leads to different decision making. For example, false negative readings exist prevalently in RFID-enabled applications, Sorensen et al.[55] investigate the impact of false-negative RFID reads on the performance of shelf inventory control policies. He et al.[56] investigate supply chain management based on artificial intelligence method–rough set, which is different from the stream-based method in this paper. Yang and Zhao[57] study supply chain network equilibrium with revenue sharing contract to adjust the contract parameters to achieve the new coordinated state through bargaining. This work is a predictive model to detect sudden demand increase in a supply chain while in this paper, we use a stream model to detect long-running events over a period of time. Deng et al.[58] propose control transfer to solve the sharing problem of tickets among different servers of different departments, however, the model is not a stream model as used in this paper. Peng et al.[59] investigate some optimization methods in a RFID based system, which can be considered as baseline work of this paper.

Generally, retail store practitioners tend to utilize business database software to manage the whole supply chain, which would be expensive, complex to deploy, hard to extend and manage. However, as retail store data collection techniques develop quickly, high speed data volume and varying user demands would make the traditional system overloaded, it is reasonable to manage the information over different semantic levels. Streaming-based methods (e.g., complex event processing) would change the retail store data management with lightweight and easy-to-use algorithms. In this paper, we consider using event processing methods (stream-based) to optimize the retail store management. Especially, we focus on algorithm optimization in a middleware level which can be easily integrated into an existing retail store management system.

3 Motivation example and event model

3.1 Motivation example

In the supply chain and retail management scenarios, RFID aims to provide item, case and pallet level of product visibility at every monitoring point along the lifecycle of the control workflows.

Generally, a RFID application system is composed of RFID transponder (tag), RFID reader, RFID antenna and software system. In the supply chain, product suppliers attach tags to different level of logistic objects (items, cases, pallets, shipments and cargos). In this paper, we assume RFID tags are formatted by the electronic product code (EPC) standard. The RFID reader reads tags EPC code through antennas mounted on different points.

We consider the RFID-enabled retail store scenario that item level, case level of products are tagged. RFID readers are mounted on the backroom entrance, on the backroom to sales floor entrance, on the sales floor shelves, on the checkout and on the exit door. The RFID-enabled retail logic used in this paper is shown in Fig. 1. In this scenario, we suppose each product in the sales floor is tagged. In the back room, incoming deliveries are monitored through the dock door RFID reader. Products are packed into cases. Both products and cases are tagged.

3.2 RFID-enabled retail store shelf replenishment process

When products are delivered to the retail store through the dock door, the dock door RFID reader captures the event and integrates each case tag (Here we assume that products arrive at the back door in cases). The RFID system filters RFID readings and updates the stock information in the decision system. When products are moved into the sales floor from the backroom, the RFID reader at the backroom to sales floor entrance captures RFID readings of each case.

The decision support system will keep the information of the shelf replenishment and update related information lists. The RFID tagged cases would be returned back to the back door.

From Fig. 1 and the above replenishment process, we can see that objects and cases movement and status need to be monitored in a continuous way. In the event processing domain, these movements are defined as patterns with event operators. In Section 3.3, we will use a declarative pattern definition language to define common patterns used in retail store replenishment monitoring.

3.3 Event model and pattern definition language

A complete RFID system consists of tags, interrogators, middleware, backend databases and applications. RFID tags are usually attached to physical objects and have memory storing the information about their corresponding objects. RFID interrogators remotely read the information contained in tags through radio signals. They are usually installed at critical business locations to monitor target objects. RFID middleware sits between hardware and software. It is responsible for filtering raw readings and transforming them into data suitable for end applications. CEP has been specifically proposed to address the issue of how to transform primitive events into semantically meaningful complex events. Its results can be directly forwarded to end applications or stored in

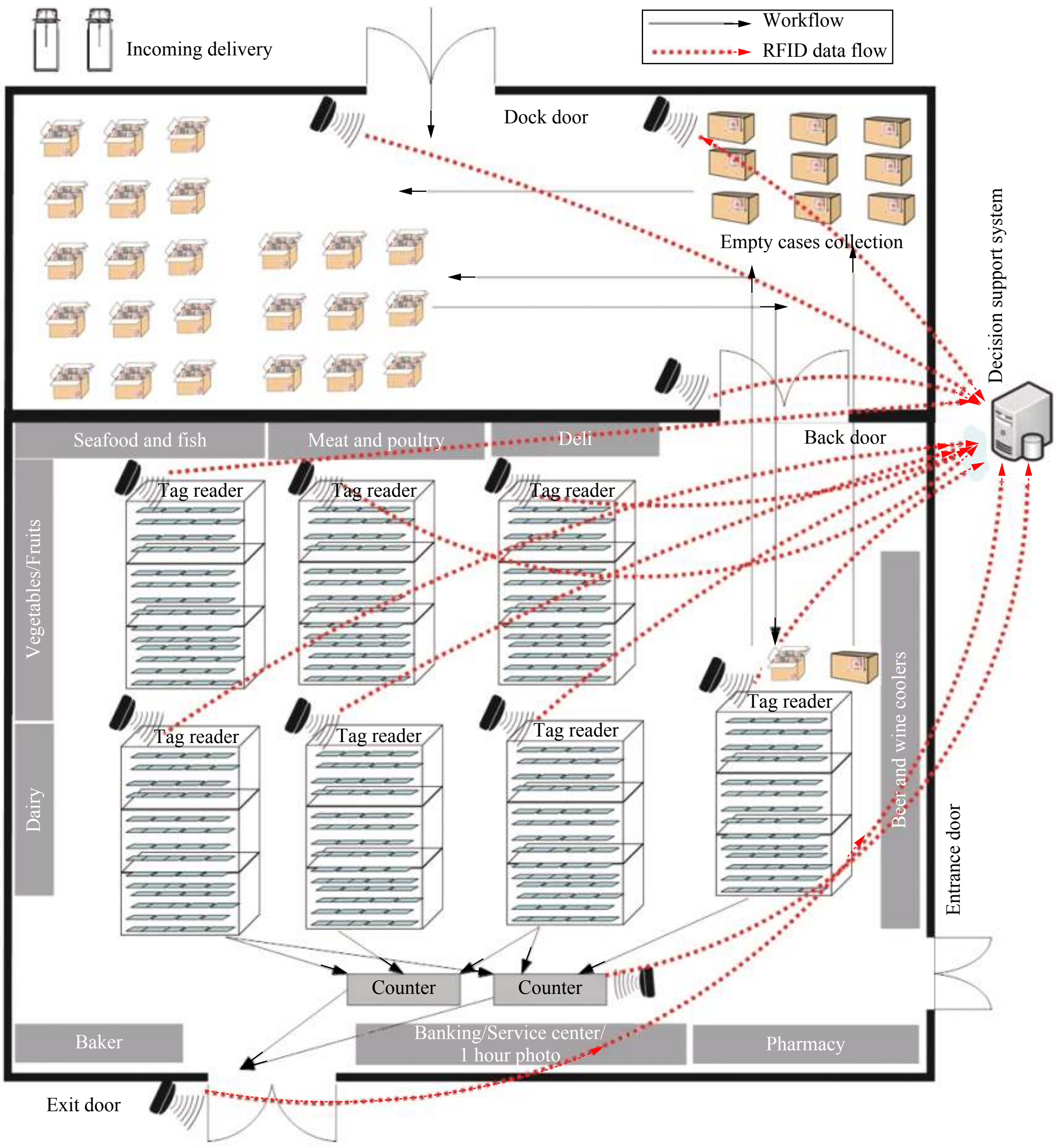

Fig. 1 Motivating example of a RFID-enabled retail store

backend databases for future use.

Definition 1 (**RFID reading**). Typically, a RFID reading contains data of the form (Oid, Rid, Rtime) as well as other information (attributes), where Oid uniquely identifies an object, Rid refers to an interrogator, and Rtime records the reading's occurrence time.

Definition 2 (**Event type**). Event type is defined as a specification or class label of objects that have the same semantic meaning.

Definition 3 (**Primitive event/Atom event**). Primitive event/Atom event is an event which cannot be divided into smaller events.

Every event is an instance of an event type. For example, in a RFID-enabled super market, RFID reading (10 000, Shelf4, 2011-07-11 20:23:34) denotes good with ID 10 000 on Shelf4 is read at time 2011-07-11 20:23:34, here Shelf4 is event type of this primitive event. In this paper, for the purpose of illustration, we use uppercase letters A, B and C, etc. to denote event types, and lowercase letters a, b and c, etc. to denote event instances of a specific event type.

Definition 4 (**Complex event**). Complex event is an event which is a combination of primitive events and/or complex events connected by event operators.

Primitive events describes When and Where information about an object, while complex events tell us How and with Whom these events happen. So the context and semantic meaning of complex events are useful for decision making. For example, in a RFID-enabled super market, RFID readings of a good at the shelf and the exit point without checkout point would generate complex event – Some good is likely to have been stolen.

Definition 5 (**Event operator**). Event operator is defined as a function that transforms the input events (primitive or complex events) into another event.

Commonly used event operators in complex event specification languages are AND operator, OR operator, NOT (negation) operator, SEQ (sequential) operator, etc. AND operator requires two events to have the same Rtime (timestamp). For OR operator, one of the sub-events occurring would trigger a complex event. NOT (negation, " ! ") operator is a unary operator, it judges an event′s negative occurrence at a given time.

RFID complex event specification usually involves attribute value comparison and large sliding window. A shoplifting monitor in a retail store, for instance, requires to the detection of a sequence of occurrence and non-occurrence events with the same Oid. Note that many languages[12, 13, 48, 49] have been proposed to specify complex event over sensor data streams, among which stream-based and shared event processing (SASE)[12] is a typical one. SASE extends traditional complex event languages developed for active databases to accommodate special requirements of RFID applications. It is declarative and has a flexible evaluation framework based on NFA. It has the overall structure of :

EVENT <event pattern>
[WHERE <qualification>]
[WITHIN <window>]

Note that the EVENT clause contains a sequence construct in particular order, whose components are the occurrences or non-occurrences of primitive events. The WHERE clause filters events through predicate constraints which involve attribute value comparison. The WITHIN clause specifies the sliding window during which the whole sequence of events should occur. We use SASE+ to define patterns in a RFID enabled retail store.

The commonly used patterns (queries) in retail store management are shown as follows:

1) Misplaced items in a store.

Query 1:
PATTERNSEQ(PredefinedShelfReading ps, OtherShelfReading os, ! (ANY(CounterReading, PredefinedShelfReading) cp))
WHERE ps.shelf_id ≠ os.shelf_id AND ps.shelf_id = cp.shelf_id
WITHIN 40 minutes

Query 1 specifies a misplacement of an item by first checking the RFID reading of an item with predefined shelf location ps (this information would be fetched from a data base query or in a cached data structure in memory), followed by a RFID reading of the same item ID at another shelf os, which is not followed by any reading of the item at a checkout counter or back at shelf ps[8]. The predicate "ps.shelf_id ≠ os.shelf_id" guarantees that the PredefinedShelfReading and OtherShelfReading events refer to different shelves. SEQ is an event operator which is used to define events occurred in a sequential time order. The " ! " operator defines a negative component of SEQ. Query 1 means, if an item with tagID1 is predefined to be located at shelf 1, then with 40 minutes, there′s no checkout RFID reading and back to shelf 1 reading of tagID1, but there exists a reading of tagID1 with another shelf ID, this means that item with tagID 1 is misplaced, maybe a stuff member of the retail store is needed to put the item back.

2) Shopping behavior detection.

Query 2:
PATTERNSEQ(ShelfReading x, CounterReading y, ExitReading z)
WHERE x.id = y.id AND x.id = z.id
WITHIN 8 hours

In Query 2, if an item is read at the shelf (not necessarily the predefined shelf), the read at the checkout reader (counter) followed by the exit reader (in order to detect shoplift behavior), then this item is in a normal shopping style.

In order to do replenishment, the decision support system should keep the normal shopping pattern count numbers and update inventory number to remind the manager if some item is going under certain predefined threshold.

3) Inventory shrinkage detection.

Query 3:
PATTERNSEQ(ShelfReading x, CounterReading y, ExitReading z) as P1
WHERE x.Tagid = y. Tagid AND x.Tagid = z.Tagid AND x.Tagid. ItemType = ItemList[i].ItemType
AND count(P1) ≥ ItemList[i].ItemType.threshold
WINTHIN 8 hours

In Query 3, suppose each item on the shelf has an Itemtype attribute which we can query from the local information system and cache into the memory for a real time event detection reference. For each item with ItemType[i], we have kept the corresponding replenishment threshold in a list and load it into the memory for verification. Function count is used to aggregate the normal shopping pattern numbers in order to make replenishment alerts for a specific item.

4 Event evaluation model

4.1 NFA based evaluation

To evaluate the queries as defined in the above section, we should parse these queries into a query plan and utilize an evaluation model to run these queries over RFID event streams. We follow the NFA based model proposed in SASE[14] and SASE+[8] with some minor improvement of data structures. To illustrate the evaluation model we take Query 3 for example.

An event query is parsed into a NFA which is implemented with stack data structure as shown in Fig. 2. Each stack corresponds to an event type which is the RFID reader type, for example counter reader, exit door

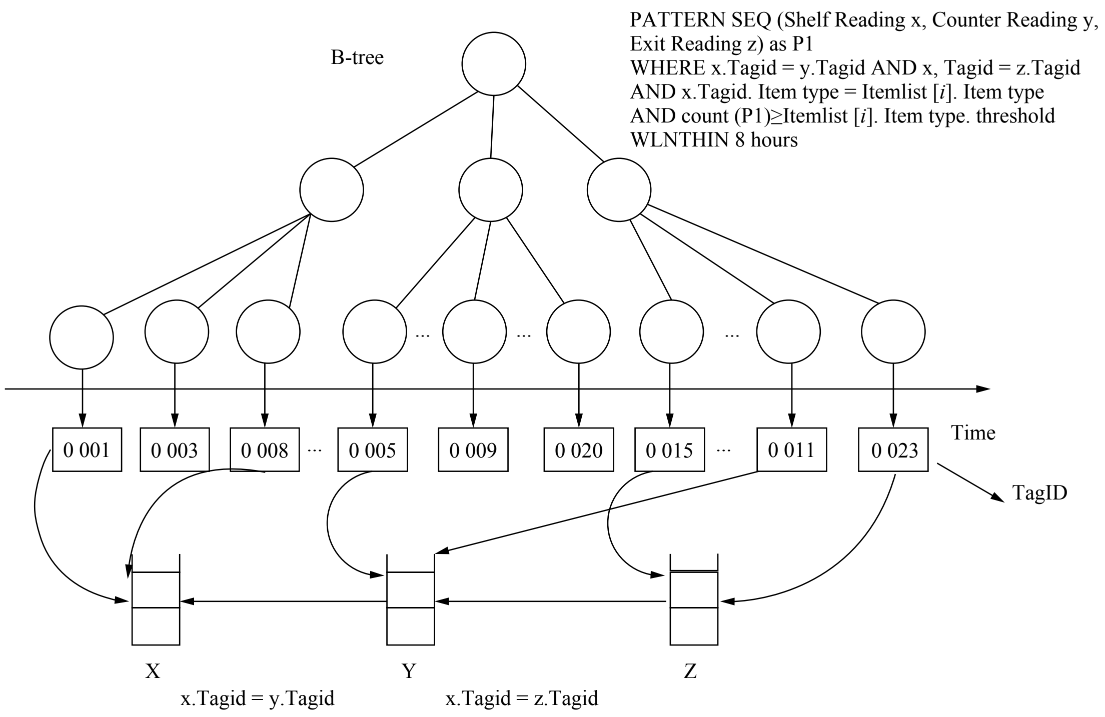
PATTERN SEQ (Shelf Reading x, Counter Reading y,
Exit Reading z) as P1
WHERE x.Tagid = y.Tagid AND x, Tagid = z.Tagid
AND x.Tagid. Item type = Itemlist [i]. Item type
AND count (P1)≥Itemlist [i]. Item type. threshold
WLNTHIN 8 hours
B-tree
0 001
0 003
0 008
0 005
0 009
0 020
0 015
0 011
0 023
Time
TagID
X
Y
Z
x.Tagid = y.Tagid
x.Tagid = z.Tagid

Fig. 2 Evaluation model of Query 1 in a RFID-enabled retail store

reader, shelf reader, etc. In this paper, we organize all the objects with a B+ tree data structure as the object is first placed into the retail store information system in Fig. 2. There is a pointer from leaf nodes of the B+ tree to the stack data structure which means the evaluation status of that TagID. For example, in Fig. 2, an item with TagID "0001" has an arrow with stack x, this means the evaluation model detects a prefix pattern "x" of SEQ(ShelfReading x, CounterReading y, ExitReading z) in Query 2 with the first event type. For an item with TagID "0005", it points to the stack y, which means the evaluation model detects a prefix pattern "xy" of SEQ(ShelfReading x, CounterReading y, ExitReading z). For TagID "0023", there exists a pointer to stack z which means the evaluation model detects a full pattern of SEQ(ShelfReading x, CounterReading y, ExitReading z), the complex event can be output.

For different event queries, the event model would initiate a similar data structure as shown in Fig. 2 to keep the intermediate pattern match results until the end of the sliding window. As we can see from Query 1 and Query 3 in the above section, they have common sub-expressions which we can use to optimize the evaluation process as shown in Fig. 3.

To share intermediate pattern matches, we take advantage of pointers to keep similar sub patterns relationships between different queries as shown in Fig. 3. As for Query 2, event type "ps" has the same semantic meaning with event type "x" in Query 3, so they can share the same pointer. Negation event type "Y" has the same semantic meanings as Query 3. So we utilize pointers to keep the relationship to evaluate two queries with same sub-expressions. This kind of mechanism saves us a copy of the original event streams and gives us a fast response to multiple event queries in parallel.

4.2 Window join-based evaluation

Window-based stream join[16] is used as counterpart. We briefly describe the workflow of algorithm as shown in Fig. 4.

In window-based stream join event detection algorithms, the sub events are detected two each time. For example, if the complex pattern is SEQ(A, B, C), the algorithm would firstly do SEQ(A, B) and then SEQ((A, B), C). So the input of stream join algorithm is two streams, with two different sliding windows applied over each stream as shown in Fig. 4. The join algorithm is described in the following steps with pattern SEQ(A, B):

1) Check there exists an input tuple s in one of the input streams A or B.

2) For every partially processed event in the hash partition of A or B, do the following 4 steps:

3) Match in memory events of one stream against the corresponding disk-resident events of the other stream that lie within the sliding window.

4) Flush the largest hash partition of A or B.

5) Update the join clock.

6) Release output from the output buffer that has timestamp.

For stream join, the algorithm first copies the stream in the window into a temporal data structure and joins the stream with the original stream in timestamp order. This mechanism will result in multiple stream scanning.

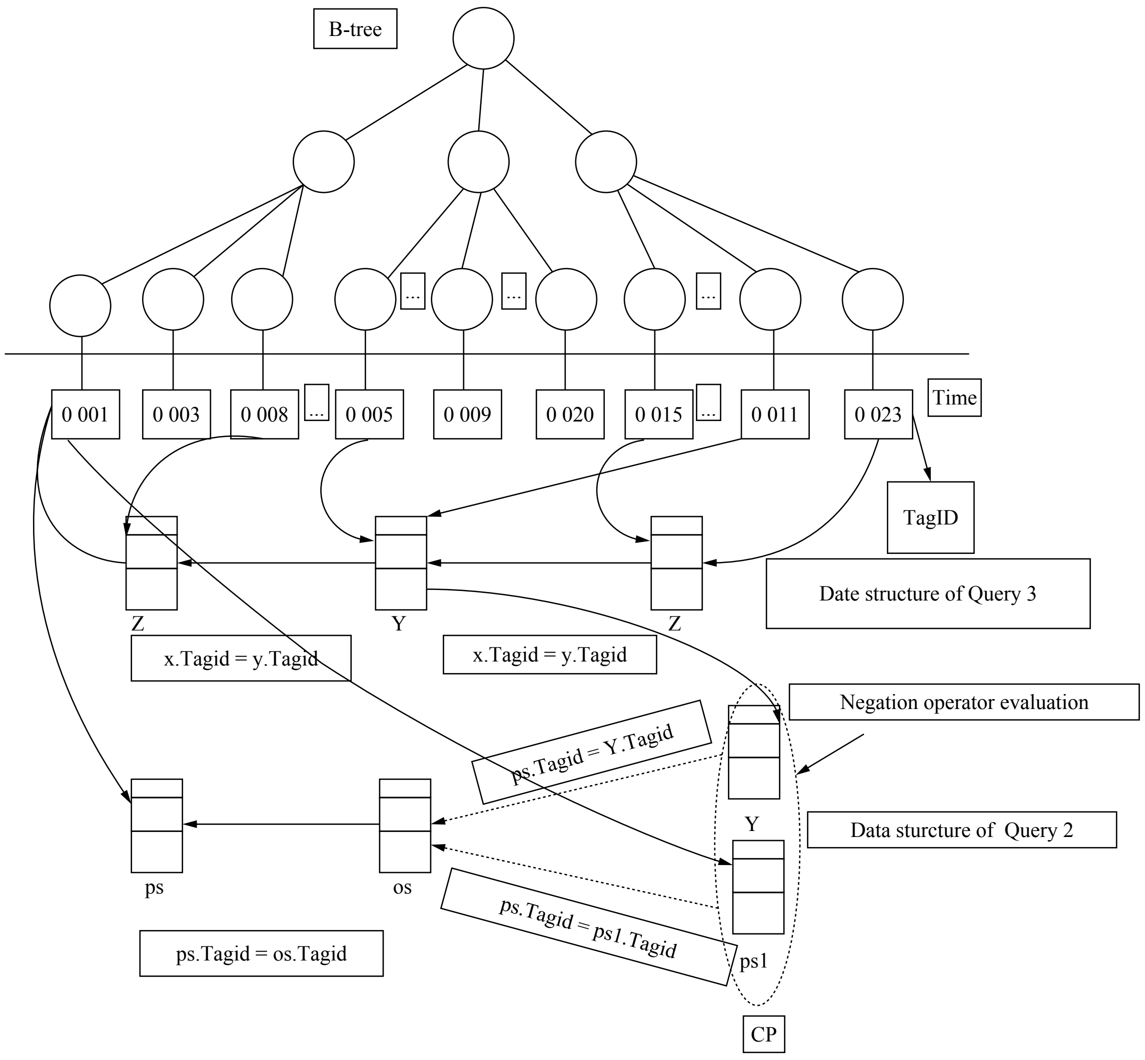

Fig. 3 Optimization of query with common sub patterns

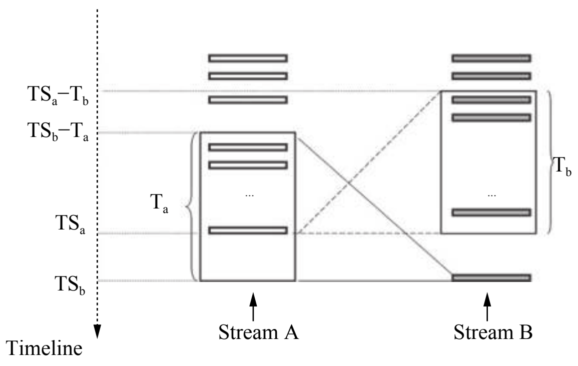

Fig. 4 Window based stream join for event detection

5 Experimental study

5.1 Experiment setup

Although many retail stores have applied RFID in the automatic management system, they usually take advantage of integrating the RFID solution with the current information system. Queries are always described with SQL or T-SQL. SQL and T-SQL can also be used to express complex events described in previous section, however, the structure of these queries may become complex and the execution engine may need to be optimized to run these long queries over unbounded event streams. As we utilize a declarative language to express complex patterns and run these queries over event streams with sliding windows, we need to set up a middleware layer before the primitive events are sent to the decision systems according to [15]. The structure of the simulated system is described in Fig. 5.

We simulate the RFID-enabled retail store with the structure as shown in Fig. 5. In a RFID scenario, raw RFID data cannot be pushed directly into the information system due to large volume, fast speed and low-level semantics[15]. So we constructed a middleware layer to buffer and preprocess the raw data. Context information of the retail system is stored in the middleware layer, especially in the event detection engine to optimize the detection process. Partial matches (intermediate matches)

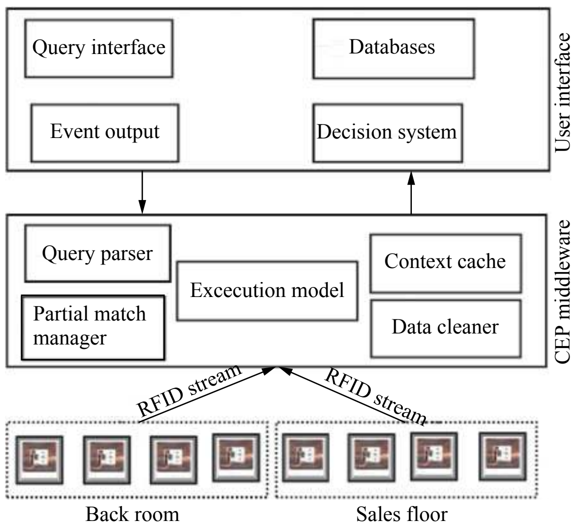

Fig. 5 Simulated RFID-nabled store with a middle ware layer

are managed especially due to sharing requirements. The simulated scenario has 12 event types (readers or reading logic points) and about 100 – 500 kinds of products. The logic units used in the simulation are in the form of case level and item level in the back room. The simulated RFID-enabled retail store is implemented with C# on a laptop with 4 GB memory and two 2.6 GHz processors.

We use stream join[16] and basic NFA evaluation models as our counterpart. We tested response time on different scales of event streams and sliding windows. Memory usages of the algorithms are also tested. To test the replenishment effect with event processing in retail stores, we set different shelves' replenishment thresholds for different products. Querys 1 and 3 are used as the replenishment complex patterns.

5.2 Experimental results on response time

We use an NFA evaluation model similar to SASE[8, 14] and an improved one with an auxiliary data structure B+ tree, and we compare the stream join algorithm with the two algorithms.

The first experiment is shown in Fig. 6. Response time is tested over different sizes of streams in a window. The window size is set as 8 hours as we think generally a retail store would at least serve for 8 hours a day. From Fig. 6, we can see that stream join consumes more time as stream size increases and NFA with a B+ tree auxiliary data structure outperforms the other two methods due to the sharing between different event instances and the deletion of partial matches with the help of an ordered B+ tree.

To test the scalability of the proposed algorithm on different lengths of the shared sub-expressions, we fix the stream size and vary the pattern length. The result is shown in Fig. 7. Two queries of length 5 are tested with different length of shared sub-expression. The stream size is set to 50 000 events. As we can see from Fig. 7, when two queries share no sub-xpressions, NFA models spends less time than the B+ tree based NFA method. This is because the B+ tree based method needs some time to build the tree and maintain the tree. As sub expression sharing increases, the B+ tree based NFA outperforms the NFA method as the sharing data structure reduces time to create and maintain new instances. But, when share length becomes the same as the pattern itself, we need to keep the relationship between different event types, which will need more time to maintain the states of each event instance.

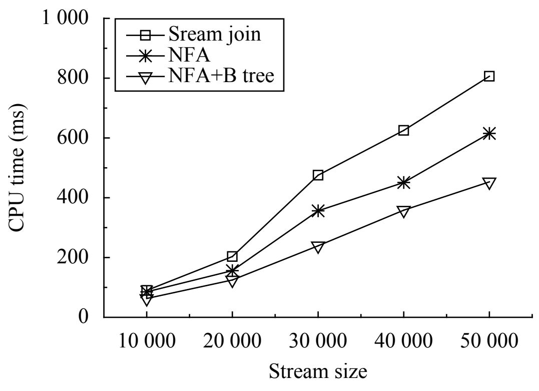

Fig. 6 Response time comparison, sliding window: 8 hours, pattern: SEQ, length: 5

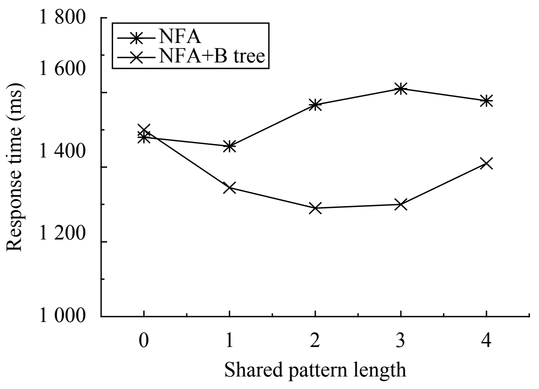

Fig. 7 Respose time for scalability test, sliding window: 8 hours, pattern: SEQ, length: 5, share subexpresion length 1–4

5.3 Experiment on replenishment decision support

For replenishment tests, we use a SQL-based trigger to define a replenishment query, then we compare the SQL queries with the event pattern queries Querys 1 and 3 over average response time. The replenishment threshold of each product is loaded into memory, both SQL and NFA based algorithm can read the data. In this experiment, we set the stream data size as 50 000 events and vary the product numbers from 200 to 600, and we use $P = 0.2$ and 0.5 to denote the probability a product is likely to be replenished.

From Fig. 8, it is noted that the SQL based trigger responses are slower than the stream-based method because the inner optimization of SQL is limited for those kinds of complex queries and NFA based method is easy to define complex queries and has a more efficient data structure to manage the event detection process.

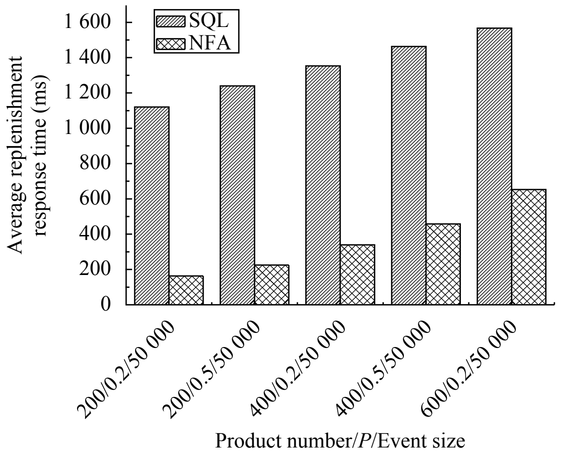

Fig. 8 Average reposne time for replenishment

5.4 Experiment on shopping behavior detection

For in-shop behavior detection, we focus on shoplifting as described in Query 2 in the previous section. The shoplifting query needs streams from shelf, counter and exit door. The stream from the shelf exhibits greater amounts than the other two points. So we need to cache events of shelf and remove outdated events when the sliding window slides forward. We set the sliding window length as 1 hour. The algorithm simulates 10 hours of shopping activity. Shoplifting events account to 0.1% of the total transaction.

We have verified the CPU time and memory used with different size of stream in the window. For CPU time, the stream size varies from 2 to 20 ($\times$ 10^4) while for the memory usage test, we set the stream size as 10 – 100 ($\times$ 10^4). From Fig. 9, we could see that, the NFA based method performs faster shoplift response than the SQL based and stream-based methods. But as we could see from Fig. 10, the NFA based method utilizes much more memory than other methods since the NFA model needs to cache the intermediate results until the sliding window forwards to the next circle even if we used some data structure to store and sort the intermediate results in order to accelerate the event detection process.

5.5 Experimental on pattern aggregation

For aggregation pattern analysis, we utilize the pattern that the pallet is read at reader A followed by Reader B and finally read at a loading dock. We assume that there are many objects in the retail store that could pass through A and B, but only part of them pass through the loading dock. This kind of scenario is common in supply chain management systems so we could utilize the optimization algorithm to delete intermediate pattern matches in order to accelerate event detection. The event query is described in Query 4 as follows.

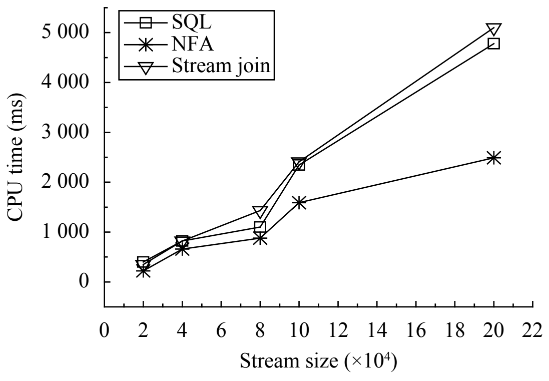

Fig. 9 Average reposne time for shoplifting detection

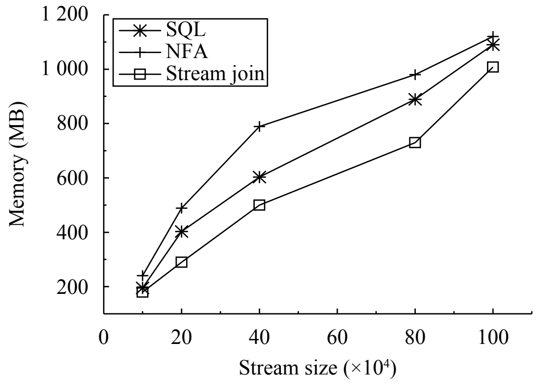

Fig. 10 Memory usage for shoplifting detection

Query 4:

PATTERN Count(SQE(A as a, B as b, loading Dock as ld)

WHERE a.TagID = b.TagID = ld.TagID

AND a.packaging_level = “pallet”

AND

{PATTERN

(StartLoading s, RfidReading + pp[ ], EndLoading c)

WHEREld

AND s.session_id = c.session_id

AND pp[i].packaging_level = “pallet”

}

In Query 4, pallets that go through readers A, B and the loading dock are counted. This pattern includes a nested pattern that counts pallets at the loading dock door and an aggregation count over the overall pattern.

In this experiment, we set the event stream as 200 000, and vary the selective possibility of the stream ratio that one object passes through A, B and the docking door to simulate different scenarios. As shown in Fig. 11, the NFA-based method outperforms the other methods while the SQL-based is the most time-consuming one due to the current SQL database system lacking special optimization for this kind of complex query. The stream join also works better than the SQL-based method. With the optimization in the NFA-based model, the event detection process can be optimized without waiting for the complete detection of complex events, thus the response time is lowered. The memory consumption of three methods are similar to the above experiments, so we omit the figure here.

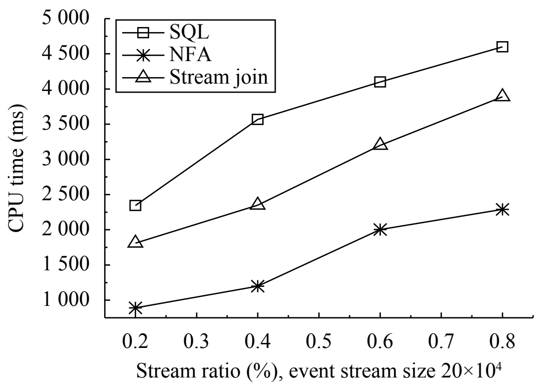

Fig. 11 Average reposne time for aggregation pattern detection

6 Conclusions

In this paper, we investigate the problem of event detection-based RFID retail store management. We use a declarative event definition language to define a control flow of the retail store and propose an evaluation model and optimization algorithm. Experiments on a simulated retail store verify that our method is effective and efficient. In the future, we would generalize the method in a distributed supply chain management system and take account of a greater level of RFID tagging and application.

Acknowledgements

This work was supported by National Social Science Fund (No. 16CTQ013), the Application Fundamental Research Foundation of Sichuan Province, China (No. 2017JY0011), and the Key Project of Sichuan Provincial Department of Education, China (No. 2017GZ0333).

References

[1] D. Corsten, T. Gruen. Desperately seeking shelf availability: An examination of the extent, the causes, and the efforts to address retail out-of-stocks. *International Journal of Retail & Distribution Management*, vol. 31, no. 12, pp. 605–617, 2003. DOI: 10.1108/09590550310507731.

[2] K. Pramatari, P. Miliotis. The impact of collaborative store ordering on shelf availability. *Supply Chain Management: An International Journal*, vol. 13, no. 1, pp. 49–61, 2008. DOI: 10.1108/13598540810850319.

[3] K. Pramatari. Collaborative supply chain practices and evolving technological approaches. *Supply Chain Management: An International Journal*, vol. 12, no. 3, pp. 210–220, 2007. DOI: 10.1108/13598540710742527.

[4] T. J. Fan, F. Tao, S. Deng, S. X. Li. Impact of RFID technology on supply chain decisions with inventory inaccuracies. *International Journal of Production Economics*, vol. 159, pp. 117–125, 2015. DOI: 10.1016/j.ijpe.2014.10.004.

[5] S. Shin, B. Eksioglu. An empirical study of RFID productivity in the U.S. retail supply chain. *International Journal of Production Economics*, vol. 163, pp. 89–96, 2015. DOI: 10.1016/j.ijpe.2015.02.016.

[6] J. Liu, B. Xiao, K. Bu, L. J. Chen. Efficient distributed query processing in large RFID-enabled supply chains. In *Proceedings IEEE Conference on Computer Communications*, Toronto, Canada, pp. 163–171, 2014. DOI: 10.1109/INFOCOM.2014.6847936.

[7] S. Y. Qi, Y. Q. Zheng, M. Li, Y. H. Liu, J. L. Qiu. Scalable industry data access control in RFID-enabled supply chain. *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3551–3564, 2016. DOI: 10.1109/TNET.2016.2536626.

[8] H. P. Zhang, Y. L. Diao, N. Immerman. On complexity and optimization of expensive queries in complex event processing. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, A Utah, Snowbird, USA, pp. 217–228, 2014. DOI: 10.1145/2588555.2593671.

[9] F. Thiesse, T. Buckel. A comparison of RFID-based shelf replenishment policies in retail stores under suboptimal read rates. *International Journal of Production Economics*, vol. 159, pp. 126–136, 2015. DOI: 10.1016/j.ijpe.2014.09.002.

[10] I. P. Vlachos. A hierarchical model of the impact of RFID practices on retail supply chain performance. *Expert Systems with Applications*, vol. 41, no. 1, pp. 5–15, 2014. DOI: 10.1016/j.eswa.2013.07.006.

[11] J. Fernie, L. Sparks. *Logistics and Retail Management: Emerging Issues and New Challenges in the Retail Supply Chain*, London, UK: Kogan Page, 2014.

[12] T. J. Fan, X. Y. Chang, C. H. Gu, J. J. Yi, S. Deng. Benefits of RFID technology for reducing inventory shrinkage. *International Journal of Production Economics*, vol. 147, pp. 659–665, 2014. DOI: 10.1016/j.ijpe.2013.05.007.

[13] S. Chakravarthy, V. Krishnaprasad, E. Anwar, S. K. Kim. Composite events for active databases: Semantics, contexts and detection. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, pp. 606–617, 1994.

[14] S. Gatziu, K. R. Dittrich. Events in an active object-oriented database system. In *Proceedings of the 1st International Workshop on Rules in Database Systems*, Edinburgh, UK, pp. 23–39, 1994. DOI: 10.1007/978-1-4471-3225-7_2.

[15] N. H. Gehani, H. V. Jagadish, O. Shmueli. Composite event specification in active databases: Model & implementation. In *Proceedings of the 18th International Conference on Very Large Data Bases*, San Francisco, USA, pp. 327–338, 1992.

[16] R. Meo, G. Psaila, S. Ceri. Composite events in Chimera. In *Proceedings of the 5th International Conference on Ex-*

tending Database Technology, Avignon, France, pp. 56–76, 1996. DOI: 10.1007/BFb0014143.

[17] D. Zimmer, R. Unland. On the semantics of complex events in active database management systems. In *Proceedings of the 15th International Conference on Data Engineering*, IEEE, Sydney, Australia, pp. 392–399, 1999. DOI: 10.1109/ICDE.1999.754955.

[18] R. E. Gruber, B. Krishnamurthy, E. Panagos. CORBA Notification Service: Design challenges and scalable solutions. In *Proceedings of the 17th International Conference on Data Engineering*, IEEE, Heidelberg, Germany, pp. 13–20, 2001. DOI: 10.1109/ICDE.2001.914809.

[19] E. Wu, Y. L. Diao, S. Rizvi. High-performance complex event processing over streams. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Chicago, USA, pp. 407–418, 2006. DOI: 10.1145/1142473.1142520.

[20] Q. Chen, Z. H. Li, H. L. Liu. Optimizing complex event processing over RFID data streams. In *Proceedings of the 24th IEEE International Conference on Data Engineering*, Cancun, Mexico, pp. 1442–1444, 2008. DOI: 10.1109/ICDE.2008.4497583.

[21] J. Agrawal, Y. L. Diao, D. Gyllstrom, N. Immerman. Efficient pattern matching over event streams. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Vancouver, Canada, pp. 147–160, 2008. DOI: 10.1145/1376616.1376634.

[22] H. P. Zhang, Y. L. Diao, N. Immerman. Recognizing patterns in streams with imprecise timestamps. *Information Systems*, vol. 38, no. 8, pp. 1187–1211, 2013. DOI: 10.1016/j.is.2012.01.002.

[23] H. Zhang, Y. Diao, N. Immerman. Recognizing patterns in streams with imprecise timestamps. *Information Systems*, vol. 38, no. 8, pp. 1187–1211, 2013. DOI: 10.1016/j.is.2012.01.002.

[24] O. Cooper, A. Edakkunni, M. J. Franklin, W. Hong, S. R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. Wu. HiFi: A unified architecture for high fan-in systems. In *Proceedings of the 30th International Conference on Very Large Data Bases*, ACM, Toronto, Canada, pp. 1357–1360, 2004.

[25] F. S. Wang, P. Y. Liu. Temporal management of RFID data. In *Proceedings of the 31st International Conference on Very Large Data Bases*, ACM, Trondheim, Norway, pp. 1128–1139, 2005.

[26] M. Ray, E. A. Rundensteiner, M. Liu, C. Gupta, S. Wang, I. Ari. High-performance complex event processing using continuous sliding views. In *Proceedings of the 16th International Conference on Extending Database Technology*, ACM, Genoa, Italy, pp. 525–536, 2013. DOI: 10.1145/2452376.2452437.

[27] Y. M. Qi, L. Cao, M. Ray, E. A. Rundensteiner. Complex event analytics: Online aggregation of stream sequence patterns. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Snowbird, USA, pp. 229–240, 2014. DOI: 10.1145/2588555.2593684.

[28] W. A. Higashino, M. A. M. Capretz, L. F. Bittencourt. CEPSim: Modelling and simulation of complex event processing systems in cloud environments. *Future Generation Computer Systems*, vol. 65, pp. 122–139, 2016. DOI: 10.1016/j.future.2015.10.023.

[29] N. Dziengel, M. Seiffert, M. Ziegert, S. Adler, S. Pfeiffer, J. Schiller. Deployment and evaluation of a fully applicable distributed event detection system in wireless sensor networks. *Ad Hoc Networks*, vol. 37, pp. 160–182, 2016. DOI: 10.1016/j.adhoc.2015.08.017.

[30] Y. H. Wang, K. Cao, X. M. Zhang. Complex event processing over distributed probabilistic event streams. *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1808–1821, 2013. DOI: 10.1016/j.camwa.2013.06.032.

[31] S. Jayasekara, S. Kannangara, T. Dahanayakage, I. Ranawaka, S. Perera, V. Nanayakkara. Wihidum: Distributed complex event processing. *Journal of Parallel and Distributed Computing*, vol. 79–80, pp. 42–51, 2015. DOI: 10.1016/j.jpdc.2015.03.002.

[32] N. Giatrakos, A. Artikis, A. Deligiannakis, M. Garofalakis. Complex event recognition in the big data era. *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1996–1999, 2017. DOI: 10.14778/3137765.3137829.

[33] M. Dayarathna, S. Perera. Recent advancements in event processing. *ACM Computing Surveys* (*CSUR*), vol. 51, no. 2, pp. 1–36, 2018. DOI: 10.1145/3170432.

[34] S. Hagedorn, T. Räth. Efficient spatio-temporal event processing with STARK. In *Proceedings of the 20th International Conference on Extending Database Technology*, OpenProceedings.org, Venice, Italy, pp. 570–573, 2017.

[35] L. Zou, Z. D. Wang, D. H. Zhou. Event-based control and filtering of networked systems: A survey. *International Journal of Automation and Computing*, vol. 14, no. 3, pp. 239–253, 2017. DOI: 10.1007/s11633-017-1077-8.

[36] H. Gonzalez, J. W. Han, X. L. Li, D. Klabjan. Warehousing and analyzing massive RFID data sets. In *Proceedings of the 22nd International Conference on Data Engineering*, IEEE, Atlanta, USA, 2006. DOI: 10.1109/ICDE.2006.171.

[37] H. Gonzalez, J. W. Han, X. L. Li. Flowcube: Constructing RFID flowcubes for multi-dimensional analysis of commodity flows. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, ACM, Seoul, Korea, pp. 834–845, 2006.

[38] C. H. Lee, C. W. Chung. Efficient storage scheme and query processing for supply chain management using RFID. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Vancouver, Canada, pp. 291–302, 2008. DOI: 10.1145/1376616.1376648.

[39] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, S. Zdonik. Monitoring streams: A new class of data management applications. In *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, China, pp. 215–226, 2002.

[40] J. J. Chen, D. J. DeWitt, F. Tian, Y. Wang. NiagaraCQ: A scalable continuous query system for internet databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Dallas, USA, pp. 379–390, 2000. DOI: 10.1145/342009.335432.

[41] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, M. A. Shah. TelegraphCQ: Continuous dataflow processing. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, San Diego, USA, pp. 668, 2003. DOI: 10.1145/872757.872857.

[42] L. Cao, E. A. Rundensteiner. High performance stream query processing with correlation-aware partitioning. *Proceedings of the VLDB Endowment*, vol. 7, no. 4, pp. 265–276, 2013. DOI: 10.14778/2732240.2732245.

[43] U. Srivastava, J. Widom. Memory-limited execution of

windowed stream joins. In *Proceedings of the 30th International Conference on Very Large Data Bases*, ACM, Toronto, Canada, pp. 324–335, 2004.

[44] Y. C. Tu, S. Liu, S. Prabhakar, B. Yao. Load shedding in stream databases: A control-based approach. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, ACM, Seoul, Korea, pp. 787–798, 2006.

[45] N. Alon, P. B. Gibbons, Y. Matias, M. Szegedy. Tracking join and self-join sizes in limited storage. In *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ACM, Philadelphia, Pennsylvania, USA, pp. 10–20, 1999. DOI: 10.1145/303976.303978.

[46] J. Wu, K. L. Tan, Y. L. Zhou. Window-oblivious join: A data-driven memory management scheme for stream join. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, IEEE, Banff, Canada, pp. 21, 2007. DOI: 10.1109/SSDBM.2007.43.

[47] P. Seshadri, M. Livny, R. Ramakrishnan. The design and implementation of a sequence database system. In *Proceedings of the 22th International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc., San Francisco, USA, pp. 99–110, 1996.

[48] R. Sadri, C. Zaniolo, A. Zarkesh, J. Adibi. Expressing and optimizing sequence queries in database systems. *ACM Transactions on Database Systems*, vol. 29, no. 2, pp. 282–318, 2004. DOI: 10.1145/1005566.1005568.

[49] A. Lerner, D. Shasha. AQuery: Query language for ordered data, optimization techniques, and experiments. In *Proceedings of the 29th International Conference on Very Large Data Bases*, ACM, Berlin, Germany, pp. 345–356, 2003.

[50] H. K. H. Chow, K. L. Choy, W. B. Lee. A dynamic logistics process knowledge-based system–An RFID multi-agent approach. *Knowledge-based Systems*, vol. 20, no. 4, pp. 357–372, 2007. DOI: 10.1016/j.knosys.2006.08.004.

[51] X. D. Zhu, Z. Q. Huang. Conceptual modeling rules extracting for data streams. *Knowledge-based Systems*, vol. 21, no. 8, pp. 934–940, 2008. DOI: 10.1016/j.knosys.2008.04.003.

[52] S. H. Choi, Y. X. Yang, B. Yang, H. H. Cheung. Item-level RFID for enhancement of customer shopping experience in apparel retail. *Computers in Industry*, vol. 71, pp. 10–23, 2005. DOI: 10.1016/j.compind.2015.03.003.

[53] C. Condea, F. Thiesse, E. Fleisch. RFID-enabled shelf replenishment with backroom monitoring in retail stores. *Decision Support Systems*, vol. 52, no. 4, pp. 839–849, 2012. DOI: 10.1016/j.dss.2011.11.018.

[54] C. Metzger, F. Thiesse, S. Gershwin, E. Fleisch. The impact of false-negative reads on the performance of RFID-based shelf inventory control policies. *Computers & Operations Research*, vol. 40, no. 7, pp. 1864–1873, 2013. DOI: 10.1016/j.cor.2013.02.001.

[55] H. Sorensen, S. Bogomolova, K. Anderson, G. Trinh, A. Sharp, R. Kennedy, B. Page, M. Wright. Fundamental patterns of in-store shopper behavior. *Journal of Retailing and Consumer Services*, vol. 37, pp. 182–194, 2017. DOI: 10.1016/j.jretconser.2017.02.003.

[56] Y. He, X. D. Liang, F. M. Deng, Z. Li. Emergency supply chain management based on rough set – house of quality. *International Journal of Automation and Computing*, published online. DOI: 10.1007/s11633-018-1133-z.

[57] A. T. Yang, L. D. Zhao. Supply chain network equilibrium with revenue sharing contract under demand disruptions. *International Journal of Automation and Computing*, vol. 8, no. 2, pp. 177–184, 2011. DOI: 10.1007/s11633-011-0571-7.

[58] H. F. Deng, W. Deng, H. Li, H. J. Yang. Authentication and access control in RFID based logistics-customs clearance service platform. *International Journal of Automation and Computing*, vol. 7, no. 2, pp. 180–189, 2010. DOI: 10.1007/s11633-010-0180-x.

[59] S. L. Peng, J. He, H. N. Yu, S. Cang. Complex event processing for RFID-enabled retail store. In *Proceedings of the 23rd International Conference on Automation and Computing*, IEEE, Huddersfield, UK, 2017. DOI: 10.23919/IConAC.2017.8081977.

Shang-Lian Peng received the B. Sc. degree in information and computing science from China West Normal University, Chine in 2004, the M. Sc. degrees in computer science from Wuyi University, China in 2007, and the Ph. D. degree in computer science from Northwestern Polytechnic University, China in 2012. Since 2012, he is a faculty member at Chengdu University of Information Technology, China. He has published about 15 refereed journal and conference papers. He is a member of China Computer Federation (CCF), Association for Computing Machinery (ACM) and IEEE.

His research interests include data management, database, RFID, internet of things, and cloud computing.

E-mail: psl@cuit.edu.cn (Corresponding author)

ORCID iD: 0000-0003-0696-5682

Ci-Jian Liu is a undergraduate in computer science at the SWJTU-Leeds Joint School, Southwest Jiaotong University, China. He has contributed in the design and implementation of the event processing system.

His research interests including clouding computing and event detection.

E-mail: 19632611@qq.com

Jia He received the B. Sc. degree in computer science from Southwest University, China in 1989, and the Ph. D. degree in computer science from University of Electronic Science and Technology of China, China in 2012. She is a professor in Chengdu University of Information Technology, China. She is a member of CCF, ACM and IEEE.

Her research interests include cloud computing, intelligent computing and artificial intelligence.

E-mail: hejia@cuit.edu.cn

Hong-Nian Yu received the B. Eng. degree in electrical and electronic engineering from Harbin Institute of Technology, China in 1982, the the M. Sc. degree in control engineering from Northeast Heavy Machinery Institute, China in 1984, and the Ph. D. degree in robotics in King's College London, UK in 1994. He is a professor in Bournemouth University, UK. He has held academic positions at the Universities of Sussex, Liverpool

John Moor, Exeter, Bradford, Staffordshire and Bournemouth in UK. He is currently a professor in computing at Bournemouth University, UK. He has extensive research experience in mobile computing, modelling, scheduling, planning, and simulations of large discrete event dynamic systems with applications to manufacturing systems, supply chains, transportation networks, computer networks and RFID applications, modelling and control of robots and mechatronics, and neural networks. He has published over 200 journal and conference research papers. He is a member of the Engineering and Physical Sciences Research Council (EPSRC) Peer Review College. He is senior member of IEEE.

His research interests include mobile computing, modelling, scheduling, planning and simulations of large discrete event dynamic systems.

E-mail: yu61150@ieee.org

Fan Li received the B. Sc. and M. Sc. degrees in computer science from University of Electronic Science and Technology of China (UESTC), China in 2003 and 2006, respectively. He is a lecturer in Chengdu University of Information Technology, China. He is a member of CCF. He has extensive research experience in cloud computing, virtualization, modelling.

His research interests include cloud computing and distributed computing.

E-mail: 24023793@qq.com