JOIV : Int. J. Inform. Visualization, 5(4) - December 2021 438-447

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

An Efficient Approach for Uncertain Event Detection in RFID Complex Event Processing

Siti Salwani Binti Yaacob [a,b,1], Hairulnizam Bin Mahdin [b,2], Mohammed Saeed Jawad [b], Nayef Abdulwahab Mohammed Alduais [b], Akhilesh Kumar Sharma [c], Aldo Erianda [d]

[a] *Jabatan Teknologi Maklumat & Komunikasi, Politeknik Sultan Abdul Halim Mu'adzam Shah, Bandar Darulaman, Jitra, 06000, Malaysia*
[b] *Center of Intelligent and Autonomous Systems, Faculty of Computer Science & Information Technology, Universiti Tun Hussein Onn Malaysia, 86400, Malaysia*
[c] *Department of Information Technology, Manipal University Jaipur, Jaipur, Rajasthan 303007, India*
[d] *Department of Information Technology, Politeknik Negeri Padang, West Sumatera, Indonesia*
Corresponding author: [1] salwani1039@gmail.com; [2] hairuln@uthm.edu.my

***Abstract*— The globalization of manufacturing has increased the risk of counterfeiting as the demand grows, the production flow increases, and the availability expands. The intensifying counterfeit issues causing a worriment to companies and putting lives at risk. Companies have ploughed a large amount of money into defensive measures, but their efforts have not slowed counterfeiters. In such complex manufacturing processes, decision-making and real-time reactions to uncertain situations throughout the production process are one way to exploit the challenges. Detecting uncertain conditions such as counterfeit and missing items in the manufacturing environment requires a specialized set of technologies to deal with a flow of continuously created data. In this paper, we propose an uncertain detection algorithm (UDA), an approach to detect uncertain events such as counterfeit and missing items in the RFID distributed system for a manufacturing environment. The proposed method is based on the hashing and thread pool technique to solve high memory consumption, long processing time and low event throughput in the current detection approaches. The experimental results show that the execution time of the proposed method is averagely reduced 22% in different tests, and our proposed method has better performance in processing time based on RFID event streams.**

***Keywords*— Uncertain event detection; complex event processing; RFID.**

Manuscript received 14 Feb. 2021; revised 22 May 2021; accepted 17 Oct. 2021. Date of publication 31 Dec. 2021.

I. INTRODUCTION

With the increasing risk of fraud, industries based on Radio Frequency Identification (RFID) systems are seeking to manage the entire process more efficiently in real-time. The most common frauds in this sector are the non-cash fraud, such as stealing goods and materials by employees [1], and the misuse of stocks [2] that continues to be the cause of concerns. Furthermore, as data from sensors flows into the organization at ever-faster speeds, conventional data management architectures were not suitable for looking for uncertain patterns such as counterfeit and missing items within a reasonable time frame [3]. Besides high data velocity, these industrial requirements such as shortened life cycles, more significant variability, and customized products, are reasons for complexity in the production system. To process this streaming data according to the manufacturing stages requires appropriate techniques to detect uncertain conditions and changes.

In recent lists of industrial systems trends, complex event processing (CEP) is thriving for good reasons. CEP is a technique used to describe an analytical engine that processes information from single or multiple sources of events by filtering, analysing and matching semantically low-level events relating syntacting structures between individual words such as sensor data, log data, or RFID data [4]. Fig. 1 shows the entire process in CEP, from event producing to event use represented by [5]. The sensor devices generate raw data. This raw data is pre-processed to create clean data and stored in the database. The pre-processing includes filtering

redundancy, detecting anomalies, as well as standardization according to the domain.

After refining raw data into historical or logs, simple events are generated using an interpreter. The interpreter's function is to execute instructions written in a programming or scripting language into a machine language program. This simple event is a time-series where it comprises a sequence of clean raw data written in machine language. Once it is completed, these simple events are ready to be processed in the complex event processing engine. The CEP engine is a cutting-edge technology for extricating information from a stream of events. It is a data processing technique according to predefined rules based on the specific domain or the application requirement. A rule, rules, ruleset, or pattern is nothing but a knowledge (facts) base consisting of one or more business rules (or simply rules). In other words, a rule in the ruleset represents some knowledge. Rules are usually in the form of 'if-then'. The CEP engine comprises knowledge and instruction representation sets and events processing algorithms. It dictates data flows and steps of how data should be processed based on rules and generates complex events as output. For example, if the matching is successful in the CEP engine, a complex event will be generated and sent to the business application.

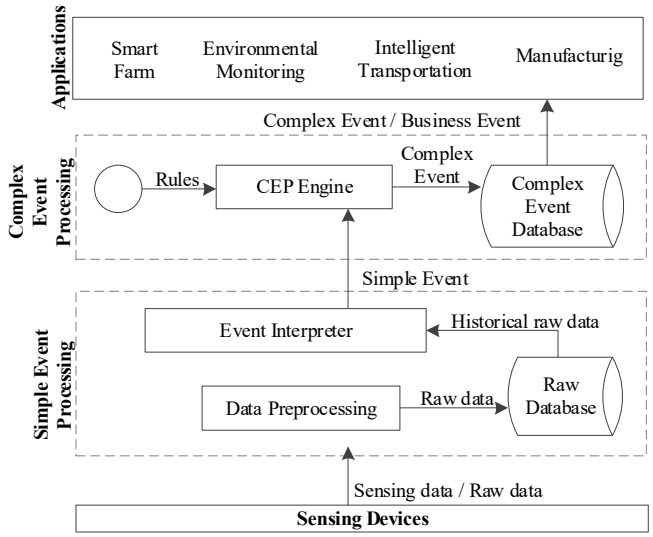

Fig. 1 Complex Event Processing Architecture

Complex events, also known as high-level semantic events, are generally associated with significant business events to indicate that they will be responded to in real-time. An example of business events or complex events is whether the package is delivered or product processing completed. There is another underlying assumption that the event rules or patterns can always be calculated or deterministic [6]. However, in an industry such as product manufacturing, those assumptions do not hold, and those patterns are often uncertain [7].

According to [8], the event source that streams into the CEP system may contain incorrect data that causes uncertainty of events generated by either event stream or event rules. Uncertainty in the event stream is more related to the deformities and unclearness caused by the sensor [9]. While uncertainty rules are the faulty judgment about an event occurring or the errors such as an abnormal situation or behavior detected according to the business situation in a particular domain during the complex event detection process [10]. According to [11], the essential for CEP engines is the engine being able to process a very high volume of data with minimal delay (low latency) even in presence of complex rules and a large number of incoming events. Due to low-latency requirements, CEP over high-speed rate may consume a large amount of CPU and memory resources during the complex event detection process [12]. Since it is not possible to keep all events in memory, if memory management is not adequately handled, intermediate results stored in the main memory may excessively grow [4]. Not only that, the CEP engine may result in long processing time as a result of high latency. Otherwise, some will suffer from high setup costs, requiring hours, if not days of unknown configuration setting [13]. Thus, lightweight processing and minimizing memory consumption play an important role in the robustness of event stream processing. After all, without having CEP, a system may result in false positives. Which means, a wrong indication in reporting or conditions that actually do not present a presence in higher level.

In this research, we focus on detecting the uncertainty in event rules efficiently, especially abnormal behavior such as counterfeit and missing items in RFID CEP distributed systems [14]. To solve the issues of high-memory consumption, long processing time, and high latency, we proposed uncertain event detection using modified hashing technique and thread pool technique to detect an uncertain event in a RFID CEP distributed system. Hashing brings a more flexible and reliable method of data acquisition than any other data structure. Hashing uses hash functions to map the larger key values into the smaller keys. The contributions of this paper can be summarized as follows:

- We propose a novel uncertain event detection algorithm (UDA), that is based on the hashing technique and thread pool technique to detect uncertain patterns in RFID CEP in the distributed system.
- Comprehensive overall performance evaluation of the proposed approach is carried out and in comparison with Instans [15], ESPER [16], NFA-HTS [40] and LCA [43].

II. Materials and Method

In this section, reviews of related literature are discussed. It includes an in-depth analysis of existing approaches including the challenges on the techniques that have been used before.

A. RFID System Architecture

In the RFID distributed system, the devices include RFID reader, RFID tag, and the middleware. RFID has been known for cost-effective technology as it bridged the gap between automation and information flow by improving traceability and offline reliability[17]. Despite the benefits, the system inevitably carries some practical issues such as inaccuracies and imperfections due to the surrounding structures and mediators [18]. The inconsistency of raw RFID data has made it become unreliable. Besides the fact that the data arrives in huge capacity and high volume, RFID data has its implicit

meanings that need to be processed, refined, and filtered to get meaningful data [19].

In RFID CEP distributed systems, data often arrives in a huge capacity and in the form of an event stream. These data that are carrying information to the CEP engine are known as RFID event, event and simple event [20]. While in the RFID CEP engine, it consists of simple event processing and complex event processing [21]. Simple event processing involves pre-processing data such as filtering and aggregating. Complex event processing detects complex events based on sequence of appearances or a set of simple events by non-stop inspecting the event flow [4]. In CEP, the processing includes two types of events; simple event and complex event [22]. Simple events are a set of events produced during the interconnection and interaction between readers and tagged objects [23]. In contrast, a complex event is a combination of simple events by pattern matching based on rules that happen over some time.

B. Simple Event Processing

Simple event processing is a simple process of raw data such as filtering out repeated data and aggregate data to get a set of data before transforming into an event or simple event. Based on Table 1, the RFID data such as electronic product tag (EPC) is used as an example to depict the possible event in the entire process of device interaction. In this process, the data is enriched by adding product name (Tag A). This is known as a simple event. The simple event can also represent a wider range of sensors.

C. Complex Event Processing

Complex event processing is an approach of searching patterns of simple events in real-time. It involves some challenges as the events often happened concurrently [24], [25], and it has their temporal and spatial relationship [26]. For example in Table 1, consider a verification rule needed to be done to verify the existence of Tag A according to the RFID reader in Department A and Department B. Based on the experiment, there is a certain value that sets up the threshold. Considering the threshold is time, Tag A needs to have a threshold time of more than 24 hours to consider it exists and ready to ship. Based on this situation, the events have shown their temporal and spatial relationships. In the CEP system, there is a stream of events generated by the users or devices, and queries for patterns according to its rules. These events were created in a high declarative query language such as Event Processing Language (EPL) [15].

TABLE I
EXAMPLE OF SIMPLE EVENT AND COMPLEX EVENT

	Simple Event	Complex Event
Rules	if **a** then **b**	if (**a** and **b** and **c**) and (not **d**) and (time window < **e** minutes) and (asset model = **x**) then **f**
Type of event	Tag A is epc:001010.0101.1000	Tag A has done a quality check in Department A and Department B from 10.00 am until 10.00 pm and is ready for shipping.

The previous year ahead, a few RFID CEP approaches have been reformed and have defined query languages to express the complex event. A literature survey shows that the scientists have focused on the design and implementation of systems that can manage complex event queries over event streaming data in real-time. Some general complex event detection systems such as SASE [27], CAYUGA [28], [29], CASCADIA[30], and RCEDA[20] provide basic detection functions for complex event detection. Although the strategies offer the greater whole and essential processing capabilities to the complex event, those processing strategies implement repeatable and needless processing, inserting and seek operations while detecting complex events in RFID event streams.

D. Related Works

Previous years, many studies have been conducted in different parts of the engine to analyze a large amount of event stream and detect such patterns in real-time. The proposed approach includes queries modification of event processing language [31], constructing and enhancing rules to detect event pattern [10], and modelling the uncertainty of events using preferred techniques for complex event detection.

The detection of uncertainty in the event processing engine has been extensively studied in many areas . One of the research lines revolves around a method that employs probabilistic graphical models to handle uncertainty [32]. In general, this model was designed based on the proposed networks where nodes serve as random variables and edges conceal probabilistic dependencies. There are two main classes of probabilistic graphical models used in CEP: Bayesian Model [8] and Markov Model [33]. These probabilistic models indicate a mathematical framework to compute uncertainty and appoint probability values using possible semantics computation. The proposed framework shows and supports correlations through conditional probability, probability distribution, and joint probability factors. In the sensor's environment, these models consider a more realistic situation; for example, the RFID reader have a tendency to fail to detect a specific tag even if it is under the reader's detection vicinity [34]. The detection computation may take various non-negative numbers from 0 to 1.

According to [35], in order to solve the uncertainty issues, the authors proposed CEP2U to an existing CEP language, TESLA. This approach uses a Bayesian network to model uncertainty in rules and probability theory to manage uncertainty in events in weather forecasting engines. The authors choose additional event processing strategy, namely, batch-like solutions to lessen latency withinside the processing engine. Similarly, TESLA makes use of consumption policies [36] to mould event contexts with assist for aggregates and sliding window specification.

According to [34], a value-driven uncertainty-aware data processing method was proposed. The authors consider RFID detection reliability, timeliness, and the throughput of an assembly line to determine the potential benefits of RFID implementation in a mixed-model assembly system. There are three components introduced in the proposed method: a CEP system, a Bayesian inference model and a value-driven optimization model. The Bayesian inference model focuses to distinguish the uncertainty propagation with the aid of mathematically decoding the reliability of the complex event that has been processed, filtered and aggregated from the raw data stream.

According to [8], another approach based on evolving Bayesian networks was proposed in order to predict the traffic flow. This Bayesian model is primarily based on event type and time using the inference method based on a mixture of a finite number of Gaussian distribution (Gaussian mixture model) and Expectation-Maximization (EM) algorithm. Their evaluation includes simulation of real-time data from traffic-monitoring networks, where vehicles travelled to and from the designated location.

Another approach using the Markov model was proposed to estimate the likelihood of hurricanes under an uncertain data stream in CEP [33]. The method focuses on the uncertainty in the event by considering the event detection such as events that may be lost or incorrectly detected. The solution was modeled based on a historical event that depicts a complex event as a sequential composition of simple events.The proposed technique makes use of a fixed of operators primarily based on timestamp and length of durations of the interval events. The computation of the expected probability of a complex event is designed using techniques based on the Markov chain.

According to [37], in order to detect anomaly of manufacturing equipment, the author utilizes the WSO2 Data Analytics Server (DAS) by clustering data generated with the aid of using sensors, and the state transitions among the determined clusters are modelled based on Markov chain model. The implementation of the thread pool as part of their approach resulted in significant performance improvements. The benefit of the thread pool technique is that it can manage and execute a large number of tasks concurrently.

According to [38], in the RFID system, leveraging the Markov model is an efficient method to model uncertainty in the event streams. However, using CEP may result in serious privacy leakage. In CEP, privacy preservation techniques on complex events are significant. To address this issue, an approach was proposed based on the Markov model in RFID for privacy preservation. This approach focuses on the privacy preservation of the complex events with the aid of using Markov chains technique. Based on the features capabilities of Markov correlated events, the authors have described optimization targets and proposed two methods. The first is taking the uncertainty and quantity of query results into account. The latter, the authors discard policy based on Markov steady states to enhance processing efficiency.

Automata theory is a machine that detects patterns within data taken from some character set or alphabet. The activity of finite automata (FA) is to just accept or reject an input subject to the sample described via way of means of the FA happening withinside the flow of the stream or data. In automata theory, there are two types of finite automata; deterministic finite automata (DFA) and non-deterministic finite automata (NFA).

In DFA, we can dictate the state to which the machine will process for each input. Previous approaches have used DFA to determine complex events. According to [39], the authors proposed a pattern recognition approach using DFA and Pattern Markov Chain to detect credit card fraud. Besides detecting pattern occurrences, this approach will estimate the number of future events until the expression is satisfied.

NFA is a machine that used to detect pattern using a string of input symbol $\sum$. For each input symbol, it changes to a new state until all input symbols have been utilized and the machine reaches its final state. In CEP, NFA is a representation to detect regular expression patterns among simple events from the RFID event streams.

According to [40], the authors proposed a complex event detection approach by using NFA and hash table (NFA_HTS). This method was extended by proposing an approach using hash table B+ tree structure (NFA-HTBS) in [41]. The NFA is used to match related simple events from massive RFID event streams, while a hash table is used to detect complex events from large matched results using B+ tree data structure. The advantage of using HTBS lies in the instant classification and storage technologies while reducing the unnecessary operation therefore reducing the time consumed, high memory used and improving the event throughput. Hash table is a flexible data structure for processing instant associative lookups. According to [42], the research focuses on minimizing the operations in the CEP engine based on the previous CEP engine, RCEDA [20]. The proposed approach initiates an event query tree and detects homogeneous and unnecessary operations by propagating the tree and hash table to detect complex events. According to [43], lazy chain automata were proposed to detect complex events. This approach mainly utilizes the NFA and lazy evaluation model to detect the frequency order of the event types in stock exchange.

In recent years, many sophisticated CEP tools have been developed to process and interpret massive data into a meaningful event. The CEP tool is a resource that has much relevance to the overall subject of knowledge management or business intelligence [44]. For example, a real-time intrusion detection system was tested using CEP tool (ESPER) in RFID middleware [45]. ESPER is a Java-based tool and open-source CEP engine that is used to develop the application. Besides storing and running the queries against stored data, the ESPER engine enables applications to store the queries and run through the data. The approach was designed to detect and differentiate the real RFID tag and clone RFID tag in RFID middleware. In general, the frequent value of the incoming RFID tag data will be compared with the predefined threshold value. If the RFID tag is new, the system will compute the frequent value of the same attribute in the database. An alarm will occur only when the value of RFID tag data is below the predefined threshold.

Besides intrusion, another research was conducted to detect theft and counterfeit in the manufacturing ecosystem [15]. The research compares the tool performance between ESPER[16] and INSTANS [15] engine in CEP platform tool in RFID middleware. The internals of ESPER were designed particularly relying on state machine and delta networks. Besides ESPER, INSTANS is an incremental engine for near real-time processing of complex, layered, heterogeneous events. It focuses on the continuous query that is in RDF format of the stream [46]. To process RDF, INSTANS makes use of the Rete algorithm. In this research,the events generated withinside the manufacturing process had been modelled using Extensible Markup Language (XML) and Resource Description Framework (RDF) formats. Both engines compare events from a different location to identify theft and counterfeit situations. While INSTANS perform the detection process in the database, the ESPER engine performs

the detection process in the streaming window before it is sent into the database. The authors concluded that ESPER shows better performance and better platform maturity, however INSTANS is rapid to process classical scenarios used for event stream processing. Batch based solutions are commonly not efficient because of the excessive latency involved. On top of that, they have got the overhead of processing overlapped information among batches a couple of times. Hence, this case requires an incremental approach to resolve the speed dimension.

E. Discussions

Based on the previous section, we assume that it is very challenging to propose an approach that gives high accuracy, low memory usage, high throughput and low detection time. In proposing a solution to detect uncertain events in event streams, having the right strategy is important to attain the desired objective. In this research, we aim at reducing the network communication cost by minimizing the unnecessary process that leads to high memory consumption and gets better results in less time. We especially take proper benefit of the storage technologies and immediate classification by implementing the hashing technique in the proposed approach. Hashing has become a part of the method used, especially in the research that is related to the real-time detection systems [47]. Besides, it can lessen lots of inserting, storing and seeking operations, and it may lessen the average searching cost by using optimum hash function, computed and flexible table size, and internal data structures.

We specifically implement ConcurrentHashMap to manage the incoming event from RFID event streams. ConcurrentHashMap is an enhancement of HashMap [48] that is a thread-safe implementation in which multiple threads can operate on a single object without any complications. While HashMap is a hash table based implementation that implements the maps interface [49]. It is used for storing key and value pairs. In other words, it maintains the mapping of key and value. ConcurrentHashMap hasn't been widely used yet, although it presents numerous advantages [50].

A hash is a function that converts one value to another. Different data structures based on hashing do different internal work. Hash table is a data structure that stores data in an associative manner [51]. In a hash table, data is saved in an array format, where each data value has its unique index value. Hash tables store data with the aid of using a technique called hashing. While in the hashing process, hash code of every element is computed internally, and the value of hash code determines the index at which the element is saved within the hash table.

ConcurrentHashMap uses an array to store data. In this research, ConcurrentHashMap is a modification of hashmap for concurrent processes that executes threads that are more than one. By default, the concurrenthashmap concurrently permits sixteen (15) threads to examine and write from the map with none outside synchronization [52]. Unlike hashtable and HashMap, it in no way locks the entire map. Instead, it divides the map into segments and locking is done on those. In ConcurrentHashMap, any range of threads can carry out retrieval operations; however for updates in the object, the thread needs to lock the unique segment in which the thread desires to operate. This form of locking mechanism is referred to as Segment locking. Hence at a time, 16 update operations must be completed by using threads.

Besides ConcurrentHashMap, motivated by [37], we utilize the thread pool technique to overcome the unnecessary process while performing small tasks in massive RFID event streams. Threads are lightweight processes within a process. In other words, the thread is a lightweight subprocess, and the processing unit is known as a thread. It is a separate path of execution. Threads are independent. If there is an exception in one thread, it does not affect the other threads.

The thread pool technique has been used for designing scalable multithreaded and distributed systems [53], [54]. A thread pool is a group of threads that are created at the time the application is started. The thread pool architecture comprises a task queue, worker thread, and a pool of thread that manages the worker threads. The thread pool provides new worker threads or completion threads or tasks that are waiting for the job and reuse many times in the worker process event cycle

The idea of these combination techniques is to protect the thread production and deletion process when processing small tasks asynchronously. In this research, the proposed technique provides better performance as respect to time and memory complexity. The details of the proposed solution will be discussed in the next section.

F. Detections of uncertain Events

In a manufacturing company, especially supply chain, RFID tags were attached to the products so that the visibility of every product can be monitored along the lifecycle of the workflows. As motivated by [15], the RFID tags are semantically designed and formatted according to the electronic product code (EPC) standard. While the RFID reader detects tags using antennas that are mounted at different locations in the motivated areas. The arriving reading that is carrying information from the RFID readers is in the form of event stream [55]. To model and detect the uncertain event in the RFID CEP, rules were introduced according to the standard and acted as a guideline to detect the affected events. As for any proposed approach, the ability to detect an accurate pattern is crucial.

By referring to the characteristics of uncertainties in CEP [4], the uncertainties were classified into two types; uncertainty in the event streaming, and uncertainty in rules. Uncertainty in events streaming may happen due to operational failure of the sensors. For example, even though missing events in the stream happened as a false negative, these missing events were wrongly inserted as false positives in the event streams. While uncertainty in rules due to lack of precision based on comparison operators ($=, <, >, \ldots$) between uncertain values of attributes such as event timestamp and event location, of matched pattern.

Motivated by [15], consider there are two event windows, production and packaging. One, the production line window is when the reader reads the items on the conveyor belts once the process of making the item is completed. Two, the packaging window is for monitoring the correct items assembly before it is packaged and transferred into the warehouse. Fig 2. portrays the process of movement of items with EPC tag during the production process. Time (t) moves

as the items move from the production section to the packaging section.

In this scenario, the manufacturer assigns an Electronic Product Code (EPC) to tag items, packages and pallets. EPC is defined in the EPCglobal Tag Data Standard, a universal identifier that gives a completely unique identification of a physical object everywhere withinside the world. The objects are packed in the package, packages are loaded onto pallets and pallets are shipped.

To show a clear process of the movement of items in a manufacturing company, Fig. 2 and Fig. 3 illustrate the movement of items from the production area to the packaging area. At t = 1 and t = 2, reader_1 detects the first batch of events from the production area. At event t = 3 from the packaging window shows that there is imprecision recorded. This imprecision is known as missing item as that particular item exists in the production window but not in the packaging window. While counterfeit is detected in the packaging window at time t = 5. This is because the specific items recorded in the does not exist in the production window at the t = 3 and t = 4 but detected in the packaging window. Meanwhile t = 5 and t = 6 is an example of the correct event detected in the production window and at t = 7 in the packaging window.

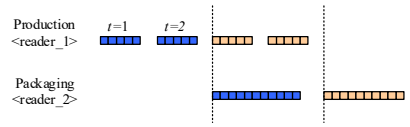

Fig. 2 Event stream model at manufacturing

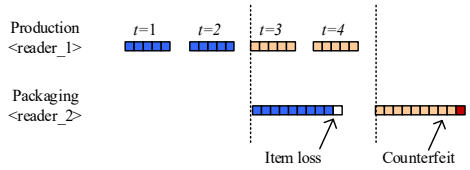

Fig. 3 Missing item and counterfeit in the event stream

Based on Fig. 4, the uncertain event detection process takes place in the RFID CEP engine. The RFID tag readings represent the raw, collected data stream. Tag reading event gives specific tag's information at a given time point. The data from the RFID data streams are usually unreliable and prone to be incomplete and contain noise. Simple event processing is responsible for cleaning the tag reading events. After data cleaning, the data stream is interpreted as a history of simple events. At this stage, as events are divided by the source of the event (i.e. Event from production and event from packaging), this event sourcing strategy can be an excellent solution to deal with the highly concurrent environment.

However, concurrency instigates the requirement for communication among executing threads, this is to synchronize their operations. It is shared memory communication, which guarantees the shared resources are accessed individually and appropriately. While complex events are application-based, that displays the need to be tracked and monitored withinside the application. Complex events are composed via means of the use of predefined patterns or rules. In this research, The rule was defined based on previous research in the real industry scenario [15]. The rule to detect the uncertain events is as follows;

- Missing item: the EPC tags and the associated event ID's exist in production but not in packaging.
- Counterfeit: Item EPC and associated event ID's exist in packaging but not in the production area.

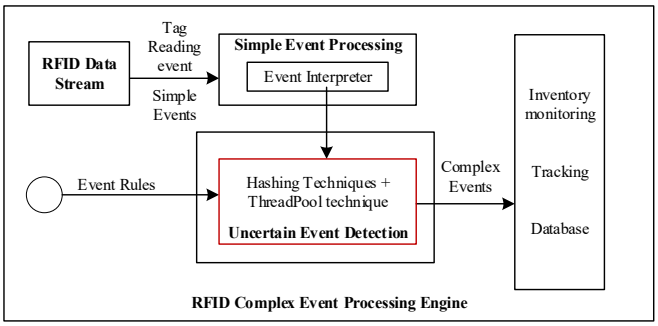

Fig. 4 Framework of RFID CEP engine

In this paper, we focus on the detection of uncertain events, focusing on counterfeit and missing items in the manufacturing company. The proposed algorithm consists of two subprocesses; main process and thread pool process.

At the main process, this stage manages the arrivals of events. We will store the resources in the event; the occurrence time of an event in the temporary string variable and the objects' tag ID (elements) in the temporary concurrentHashMap. When the process reaches the end of the elements, we check if the event is the production event or packaging event. If the event is the production event, in a separate thread, we pass the temporary event's occurrence time and temporary concurrentHashMap into the thread. If the event is the packaging event, in a separate thread, we pass the temporary event's occurrence time and temporary concurrentHashMap into the new concurrenthashMap in the thread. After passing the resource, we clear the temporary, concurrentHashMap and start reading the next event.

For the thread pooling process, is where the uncertain detections work. During the detection in the thread pool, if there are matches between production events and packaging events during the time period, the matches get pushed to match items listed in the main process. If there are no matches within that period, if the tag ID is found in the production event but not in the packaging event, the tag ID gets pushed as a missing item. If the tag ID is found in the packaging event but not in the production event, the tag ID gets pushed as a counterfeit item.

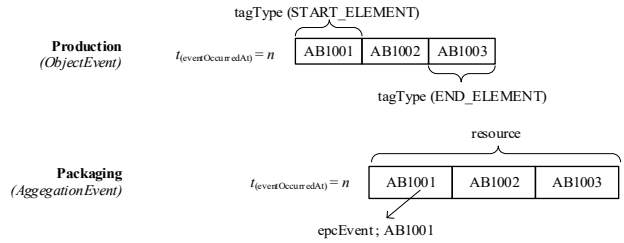

Fig. 5 Terms used in the proposed algorithm

G. Uncertain Detection Algorithm (UDA)

In this section, the proposed algorithm is described. The algorithm works based on the reference scenario, process

model at manufacturing [15], that is universal to be implemented to other industries as well. The terms used in the algorithm are defined based on Fig. 5. The explanation of Fig. 5 is interrelated with Fig. 2 and Fig. 3 in the previous subsection.

Algorithm 1 shows the proposed approach to detect counterfeit and missing items in the RFID event stream. We only focus on the operation to detect the uncertain event between the production events and the packaging events in the event stream. The algorithm is as follows;

ALGORITHM 1: Uncertain Detection Algorithm (UDA)

```
INPUT: xmlEvent
1.  BEGIN
    //Read XML as stream
2.  WHILE(xmlStreamReader ← hasNext(dalam ni apa))
3.      IF (tagType==xmlEvent ← START_ELEMENT)
4.          SWITCH (xmlStreamReader)
5.              CASE "eventOccuredAt":
6.                  update temporaryEventOccuredAt =
                    eventOccuredAt
7.                  set internal_clock_time = eventOccuredAt
8.                  BREAK
9.              CASE "element":
10.                 IF (epcEvent == xmlEvent ←
                    START_ELEMENT)
11.                     Push resource →
                        temporaryEventOccuredAt in
                        temporaryConcurrentHashMap as key →
                        value
12.                     BREAK
13.                 END IF
14.         END SWITCH
15.     ENDIF
16.     ELSE IF (tagType==xmlEvent → END_ELEMENT)
            IF (tagType == ObjectEvent)
            //Append temporaryConcurrentHashMap in new
            thread
17.         IF (PRODUCTION_TID ←
            concurrentHashMap(eventOccuredAt)
18.             FOR(entry ← PRODUCTION_TID)
19.                 IF(entry(eventOccuredAt))
20.                     PRODUCTION_TID++
21.                 END IF
22.             END FOR
23.         END IF
24.         LOST_ITEM++
25.         REMOVE ALL LIST OF PRODUCTION_TID after
            t-winc
26.     ENDIF
27.     CLEAR temporaryConcurrentHashMap
28.     CLEAR temporaryEventOccuredAt
            ELSE IF tagType==AggreagtionEvent
            //Append temporaryConcurrentHashMap in new
            thread
29.         IF (PACKAGING_TID ←
            concurrentHashMap(eventOccuredTime)
30.             FOR(entry ← PACKAGING_TID)
31.                 IF(entry == eventOccuredAt)
32.                   keyset ← entry
33.                 END IF
34.             END FOR
35.         END IF
36.         COUNTERFEIT_ITEM++
37.         REMOVE ALL LIST OF PACKAGING_TID after t-
            winf
38.     END IF
39.     CLEAR temporaryConcurrentHashMap
40.     CLEAR temporaryEventOccuredAt
41.     END IF
42. END WHILE
43. END
```

In this algorithm, the case 'eventOccuredAt' is to update the internal clock. While case 'element' reads the epc and adds them into the temporaryConcurrentHashmap.

Based in Algorithm 1, at line 1 to line 3, read the incoming event as streaming data. At line 3, if it is START_ELEMENT, store the occurrence time of an event in a temporary string variable (line 5 to line 8) and the elements of events in the temporaryConcurrentHashMap (line 9 to line 12). At line 16, if the tag is END_ELEMENT?, and if the event is from the production area (ObjectEvent), in a separate thread, we pass the temporary event's occurrence time and temporaryConcurrentHashMap into the thread (line 17 to line 26). After passing the resource, the clear eventOccurredAt, temporaryConcurrentHashMap and started reading the next event (line 28 and line 29). In line 30, if the event is from packaging area (AggregationEvent), in a separate thread, we pass the temporary event's occurrence time, and temporaryConcurrentHashMap into the thread (line 31 to line 40). After passing the resource, the clear eventOccurredAt, temporaryConcurrentHashMap and started reading the next event (line 41 and line 42).

This is the section where we create a thread pool with a fixed number of threads. A separate task is created for every event in the event stream and is submitted it to the thread pool.

In line 18, if the event just now was a production event, in a separate thread, merge the PRODUCTION_IDs to concurrentHashMap. In the thread pool, start the processing function to auto-remove and push missing items (line 19 to line 26). If there is a temporary entry from production, for which event occurred time in addition with active window detection where length is less than or equal to time extracted from the current event, report the PACKAGING_TID as missing item detected in production.

At line 31 to line 39, if the event just now was a packaging event, in a separate thread, merge the PACKAGING_TIDs into concurrentHashMap (line 31). In thread pool line 32 to line 33, if a pair of resource or element is found between production's thread and packaging's thread, it is added to the hashmap of matched items. If there is a temporary entry from packaging, for which event occurred time in addition with active window detection where length is less than or equal to time extracted from the current event, report the PACKAGING_TID as counterfeit detected in packaging. At the end of the process, the main thread checks if there are any threads still running, the process will hold a bit until all the process terminates itself.

III. Results and Discussion

In order to prove the efficiency of the proposed approach, performance analysis was conducted and compared with the existing approaches and tools. For the computer environment, Intel i3 processor and eight-gigabyte memory were used, and the operating system is Windows 7. The modules for CEP were implemented in Java. The dataset used in the experiment is presented as a time-varying stream of events that represents an observation of the actual counterfeiting and theft detection scenarios from the pharmaceutical company recorded by the sensor [15].

The performance evaluation of three processing approaches, proposed approach, Instans [50], ESPER [16], NFA-HTS [40] and LCA [43]. A slight modification has been

made to these algorithms to solve the problems. We measured mainly execution time, memory consumption and detection accuracy according to the event stream scale with 8% uncertain events. These measurements are defined as follows;

- Execution time: the time required to detect uncertain events according to the event stream scale.
- Memory consumption: memory consumed when detecting uncertain events in the event stream according to the event stream scale.
- Detection Accuracy: measures the accuracy of uncertain events according to the event stream scale.

The system execution time or throughput is the time it takes to load and analyse an event stream. It also includes the time it takes to load the output into a consistent format and parse it, and time it takes to write the results to disk. While memory consumption is the amount of memory required to load and analyse while processing an event stream. Detection accuracy is the total accuracy detected over the 8% uncertain event in the event stream. For robustness, we performed ten (10) independent runs for each experiment, and we report the median values. The number of event streams scale is varied from 1000 to 5000 with an increment of times 1000 for each scale.

A. Analysis Based on Detection Accuracy

In this section, extensive experiments were performed to find the detection accuracy of the proposed approach, Instans and ESPER were studied under varying event stream scale. Fig. 6 illustrates the performance of the proposed approach, Instans and ESPER with 8% uncertain events under varying event stream scale. Fig. 6 shows that, UDA and NFA-HTS gets 100% accurate. Instans and ESPER also maintain to get accurate detection at every event stream scale. For Instans, Rete implementations are non-distributed implementation, yet efficient in term of detection accuracy.

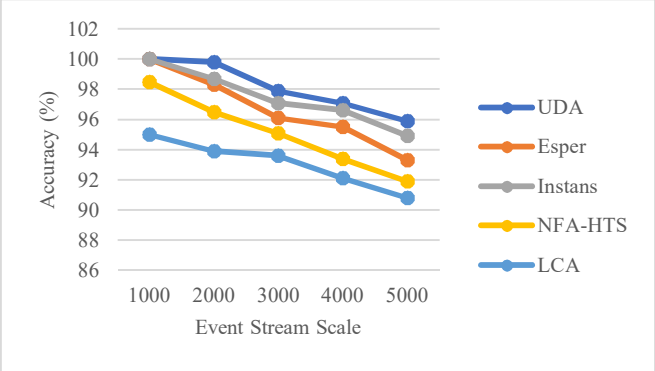

Fig. 6 Detection accuracy percentage under different event stream scale

ESPER is an open source Java-based event stream processing engine that analyzes a set of events and draws conclusions from the event stream. It provides a language called Event Processing Language (EPL) that implements and extends the SQL standard. In Instans, the evaluation of SPARQL (events) subscriptions is sequential. While in the proposed approach (UDA), the evaluation is multithreaded.

B. Analysis Based on Execution Time

Extensive experiments have been conducted in this section to see the results of reduced execution times. Fig. 7 below shows the execution time of the proposed approach using different event stream scale sizes. The execution times of the three approaches to the event streams range 1000 to 5000 events. The UDA took the least time, followed by NFA-HTS. While Instans took the longest time to process the event stream followed by LCA. This is because Instans has to focus on continuously processing RDF event streams while employing the Rete algorithm and propagating data through a matching network. By using Rete, the event matches are produced as soon as all the conditions of SPARQL graph patterns are matched. While the LCA algorithm uses a simple NFA structure to determine potential matches. In other words, events will be buffered until they find a match.

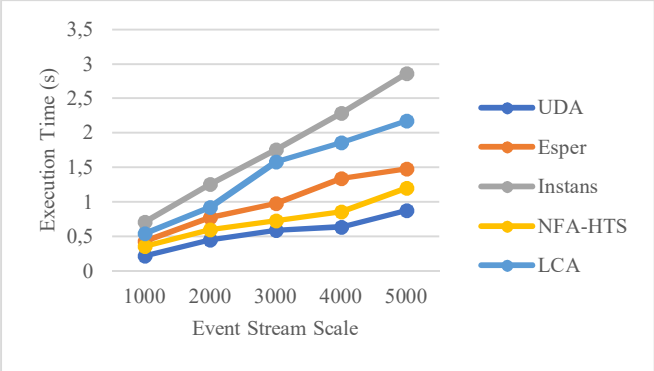

Fig. 7 The execution time of the proposed approach under different event stream scale

While using the UDA algorithm, the event has to match the pattern as soon as the event is merged in the thread pool. At the end of the process, in this approach, the main aim is that if there are any threads still running, the process will hold a bit until all the process terminates automatically. The graph also indirectly shows that by using ConcurrentHashMap and thread pool technique, it requires less time to process events compared to NFA-HTS. NFA-HTS also shares some advantages by using hash table structure. It enables easy retrieval and reducing computation operations based on storage technologies of hash table structure. ESPER is an engine primarily relying on state machines. For ESPER, we do not have a very detailed architectural diagram because the community does not release such materials to the public domain. The code is open source and the API's to use the engine is sufficiently documented.

C. Analysis Based on Memory Consumption

In this section, the proposed approach is compared with Instans, ESPER, NFA-HTS and LCA in terms of memory consumption under different event stream scales. As depicted in Fig. 8, UDA shows the least consumption and slightly better than NFA-HTS compared to the Instans, ESPER and LCA. While Instans and ESPER show higher memory consumption, especially when the event stream scale is massive. This is because, by using the thread pool, we do not have to create, manage, schedule and terminate the thread. Thread pool help mitigate performance issues by reducing the number of threads required and managing their lifecycle.

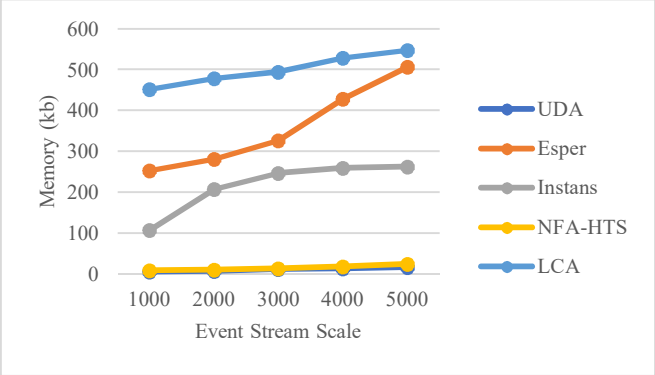

Fig. 8 Memory consumption under different event stream scale

Essentially, thread pool reuses its threads to perform the work pattern detection until it is completed. In non-technical explanation, the threads continue to be in the pool till they may be required, after which they execute the task and return the pool to be reused later so that it can pick up the next available task. This mechanism is beneficial in systems or engines that execute a large number of small tasks or processes. Besides creating threads, destroying a thread has set up and torn down costs (it takes CPU time). While in LCA, the engine continuously buffers until they find a match. Same as Instans, the engine is in charge of continuously evaluating the incoming data and storing intermediate results. At the same time, ESPER exploits in-memory processing to address the requirements of applications that analyze a high volume of event stream and promptly react to events by applying complex computations.

IV. Conclusion

In this research, the issue of uncertain event detection has been studied, and a brand new approach based on hashing and thread pool technique was proposed. The overall performance of the proposed approach is compared with the existing approaches. The results show that the performance of the proposed approach was better than the opposite approaches in terms of detection time and memory usage. Moreover, this research additionally demonstrated that, with the aid of using thread pool technique, the proposed approach performs quicker than the opposites in terms of the efficiency of the experiments. Despite the convenience of use, the thread pool has the subsequent limitations when compared to manually handling the threads. With thread pool, we do not have control over the state and priority of the thread. We additionally cannot provide a solid identification to the thread and maintain monitoring it. The technique may be inefficient whilst there is a excessice demand for the thread pool.

Synchronization overhead is one of the reasons that lead to inaccurate detections. Even though the detection accuracy deteriorates when the event stream scale increases, the proposed approach processes at the optimum rate when the event stream is at a moderate scale. For future work, we plan to expand our research on the relationship between pool size and the number of threads can be set dynamically.

Acknowledgement

This research is sponsored by Universiti Tun Hussein Onn Malaysia under Fundamental Research Grant Scheme Vot 1611 and the Ministry of Higher Education through MyBrain scholarship.

References

[1] Guthrie, John, Sarah Todd, and Jeffrey Alstete. "Inside advice on educating managers for preventing employee theft." International Journal of Retail & Distribution Management (2006).

[2] Singh, Mohan, Smriti Sachan, Akansha Singh, and Krishna Kant Singh. "Internet of Things in pharma industry: possibilities and challenges." In Emergence of Pharmaceutical Industry Growth with Industrial IoT Approach, pp. 195-216. Academic Press, 2020.

[3] Siddiqa, A., Hashem, I.A.T., Yaqoob, I., Marjani, M., Shamshirband, S., Gani, A. and Nasaruddin, F., 2016. A survey of big data management: Taxonomy and state-of-the-art. Journal of Network and Computer Applications, 71, pp.151-166.

[4] Flouris, I., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Kamp, M. and Mock, M., 2017. Issues in complex event processing: Status and prospects in the big data era. Journal of Systems and Software, 127, pp.217-236.

[5] Lan, L., Shi, R., Wang, B., Zhang, L. and Jiang, N., 2019. A universal complex event processing mechanism based on edge computing for internet of things real-time monitoring. IEEE Access, 7, pp.101865-101878.

[6] Correcher, J.F., Alonso, M.T., Parreño, F. and Alvarez-Valdés, R., 2017. Solving a large multicontainer loading problem in the car manufacturing industry. Computers & Operations Research, 82, pp.139-152.

[7] Bhat, S. and Krishnamurthy, A., 2016. Interactive effects of seasonal-demand characteristics on manufacturing systems. International Journal of Production Research, 54(10), pp.2951-2964.

[8] Wang, Y., Gao, H. and Chen, G., 2018. Predictive complex event processing based on evolving Bayesian networks. Pattern Recognition Letters, 105, pp.207-216.

[9] Muzammal, M., Gohar, M., Rahman, A.U., Qu, Q., Ahmad, A. and Jeon, G., 2017. Trajectory mining using uncertain sensor data. IEEE Access, 6, pp.4895-4903.

[10] Lee, O.J. and Jung, J.E., 2017. Sequence clustering-based automated rule generation for adaptive complex event processing. Future Generation Computer Systems, 66, pp.100-109.

[11] Cugola, G. and Margara, A., 2012. Low latency complex event processing on parallel hardware. Journal of Parallel and Distributed Computing, 72(2), pp.205-218.

[12] Hewa Raga Munige, T., 2016. Real time stream processing for Internet of Things and sensing environments (Doctoral dissertation, Colorado State University).

[13] Hallé, S., 2017. From complex event processing to simple event processing. arXiv preprint arXiv:1702.08051.

[14] Rinne, M., Solanki, M. and Nuutila, E., 2016, June. RFID-based logistics monitoring with semantics-driven event processing. In Proceedings of the 10th ACM international conference on distributed and event-based systems (pp. 238-245).

[15] Rinne, M., Nuutila, E. and Törmä, S., 2012, November. INSTANS: High-performance event processing with standard RDF and SPARQL. In 11th International Semantic Web Conference ISWC (Vol. 914, pp. 101-104).

[16] "EsperTech." http://www.espertech.com/ (accessed May 29, 2019).

[17] Sarac, A., Absi, N. and Dauzere-Peres, S., 2015. Impacts of RFID technologies on supply chains: a simulation study of a three-level supply chain subject to shrinkage and delivery errors. European Journal of Industrial Engineering, 9(1), pp.27-52.

[18] Fan, T., Tao, F., Deng, S. and Li, S., 2015. Impact of RFID Technology on Supply Chain Decisions with Inventory Inaccuracies. International Journal of Production Economics, 159, pp.117-125.

[19] Yao, X., Zhang, J., Li, Y. and Zhang, C., 2018. Towards flexible RFID event-driven integrated manufacturing for make-to-order production. International Journal of Computer Integrated Manufacturing, 31(3), pp.228-242.

[20] Wang, F., Liu, S. and Liu, P., 2009. Complex RFID event processing. The VLDB Journal, 18(4), pp.913-931.

[21] Q. J. Lei, L. S. Bo, and C. J. Kun, "Online Monitoring of Manufacturing Process Based on autoCEP," International Journal of Online Engineering (iJOE), vol. 13, no. 06, pp. 22–34, Jun. 2017.
[22] Vlahakis, G., Apostolou, D. and Kopanaki, E., 2018. Enabling situation awareness with supply chain event management. Expert Systems with Applications, 93, pp.86-103.
[23] Jia, X., Wenming, Y. and Dong, W., 2009, November. Complex event processing model for distributed RFID network. In Proceedings of the 2nd international Conference on interaction Sciences: information Technology, Culture and Human (pp. 1219-1222).
[24] Brunelli, D., Gallo, G. and Benini, L., 2016, September. Sensormind: virtual sensing and complex event detection for Internet of Things. In International Conference on Applications in Electronics Pervading Industry, Environment and Society (pp. 75-83). Springer, Cham.
[25] Zhang, Y. and Sheng, V.S., 2018. Fog-enabled Event Processing Based on IoT Resource Models. IEEE Transactions on Knowledge and Data Engineering, 31(9), pp.1707-1721.
[26] Wang, Y., Zheng, L., Hu, Y. and Fan, W., 2018, December. Multi-source heterogeneous data collection and fusion for manufacturing workshop based on complex event processing. In Proceedings of the 48th International Conference on Computers & Industrial Engineering (CIE), Auckland, New Zealand (pp. 2-5).
[27] Agrawal, J., Diao, Y., Gyllstrom, D. and Immerman, N., 2008, June. Efficient pattern matching over event streams. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 147-160).
[28] Demers, A.J., Gehrke, J., Panda, B., Riedewald, M., Sharma, V. and White, W.M., 2007, January. Cayuga: A General Purpose Event Monitoring System. In Cidr (Vol. 7, pp. 412-422).
[29] Brenna, L., Demers, A., Gehrke, J., Hong, M., Ossher, J., Panda, B., Riedewald, M., Thatte, M. and White, W., 2007, June. Cayuga: a high-performance event processing engine. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (pp. 1100-1102).
[30] Welbourne, E., Khoussainova, N., Letchner, J., Li, Y., Balazinska, M., Borriello, G. and Suciu, D., 2008, June. Cascadia: a system for specifying, detecting, and managing RFID events. In Proceedings of the 6th international conference on Mobile systems, applications, and services (pp. 281-294).
[31] Gillani, S., Zimmermann, A., Picard, G. and Laforest, F., 2019. A query language for semantic complex event processing: Syntax, semantics and implementation. Semantic Web, 10(1), pp.53-93.
[32] Akila, V., Govindasamy, V. and Sandosh, S., 2016, April. Complex event processing over uncertain events: Techniques, challenges, and future directions. In 2016 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC) (pp. 204-221). IEEE.
[33] Rincé, R., Kervarc, R. and Leray, P., 2018, September. Complex event processing under uncertainty using Markov chains, constraints, and sampling. In International Joint Conference on Rules and Reasoning (pp. 147-163). Springer, Cham.
[34] Tang, L., Cao, H., Zheng, L. and Huang, N., 2015. Value-driven uncertainty-aware data processing for an RFID-enabled mixed-model assembly line. International Journal of Production Economics, 165, pp.273-281.
[35] Cugola, G., Margara, A., Matteucci, M. and Tamburrelli, G., 2015. Introducing uncertainty in complex event processing: model, implementation, and validation. Computing, 97(2), pp.103-144.
[36] Cugola, G. and Margara, A., 2012. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys (CSUR), 44(3), pp.1-62.
[37] Akram, N., Siriwardene, S., Jayasinghe, M., Dayarathna, M., Perera, I., Fernando, S., Perera, S., Bandara, U. and Suhothayan, S., 2017, June. Anomaly detection of manufacturing equipment via high performance rdf data stream processing: Grand challenge. In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (pp. 280-285).
[38] Li, F., Wang, N., Gu, Y. and Chen, Z., 2016, September. Effective Privacy Preservation over Composite Events with Markov Correlations. In 2016 13th Web Information Systems and Applications Conference (WISA) (pp. 215-220). IEEE.
[39] Alevizos, E., Artikis, A., Katzouris, N., Michelioudakis, E., Paliouras, G. and Paliouras, G., 2018. The Complex Event Recognition Group. ACM SIGMOD Record, 47(2), pp.61-66.
[40] Wang, J., Liu, J., Lan, Y. and Cheng, L., 2018. An Efficient Complex Event Detection Algorithm based on NFA_HTS for Massive RFID Event Stream. Journal of Electrical Engineering and Technology, 13(2), pp.989-997.
[41] Wang, J., Lu, S., Lan, Y. and Cheng, L., 2018. An Efficient Complex Event Processing Algorithm Based on NFA-HTBTS for Massive RFID Event Stream. International Journal of Information Technologies and Systems Approach (IJITSA), 11(2), pp.18-30.
[42] Bok, K., Kim, D. and Yoo, J., 2018. Complex event processing for sensor stream data. Sensors, 18(9), p.3084.
[43] Kolchinsky, I. and Schuster, A., 2018. Efficient adaptive detection of complex event patterns. arXiv preprint arXiv:1801.08588.
[44] K. Tawsif, J. Hossen, J. Emerson Raja, M. Z. H. Jesmeen, and E. M. H. Arif, "A Review on Complex Event Processing Systems for Big Data," in Proceedings - 2018 4th International Conference on Information Retrieval and Knowledge Management: Diving into Data Sciences, CAMP 2018, Mar. 2018, pp. 2–7.
[45] Yin, S.N., Kang, H.S., Chen, Z.G. and Kim, S.R., 2016, October. Intrusion detection system based on complex event processing in RFID middleware. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems (pp. 125-129).
[46] Rinne, M. and Nuutila, E., 2014, October. Constructing event processing systems of layered and heterogeneous events with SPARQL. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 682-699). Springer, Berlin, Heidelberg.
[47] Arbuzin, D., 2017. Real-time detection of moving crowds using spatio-temporal data streams.
[48] Stusek, M., Masek, P., Zeman, K., Kovac, D., Cika, P., Pokorny, J. and Kröpfl, F., 2016. A Novel Application of CWMP: An Operator-grade Management Platform for IoT. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, 5(3), pp.171-177.
[49] Jantkal, B.A. and Deshpande, S.L., 2017, August. Hybridization of B-Tree and HashMap for optimized search engine indexing. In 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon) (pp. 401-404). IEEE.
[50] Pinto, G., Torres, W., Fernandes, B., Castor, F. and Barros, R.S., 2015. A large-scale study on the usage of Java's concurrent programming constructs. Journal of Systems and Software, 106, pp.59-81.
[51] Debnath, B., Haghdoost, A., Kadav, A., Khatib, M.G. and Ungureanu, C., 2016. Revisiting hash table design for phase change memory. ACM SIGOPS Operating Systems Review, 49(2), pp.18-26.
[52] Zheng, Yajie, Dongwen Zhang, Yang Zhang, Song Guo, Yanan Liang, Mengmeng Wei, and Xin Yu. "Comparison and reconfiguration of Java hash mechanisms on parallel environment." Hebei Journal of Industrial Science and Technology (2017): 06.
[53] Wei, L., Wan, S., Guo, J. and Wong, K.K., 2017. A novel hierarchical selective ensemble classifier with bioinformatics application. Artificial intelligence in medicine, 83, pp.82-90.
[54] Du, P., Ren, J., Pan, J. and Luo, A., 2014. New cross-matching algorithm in large-scale catalogs with ThreadPool technique. Science China Physics, Mechanics and Astronomy, 57(3), pp.577-583.
[55] Bhargavi, R., 2016. Complex Event Processing Framework for Big Data Applications. In Data Science and Big Data Computing (pp. 41-56). Springer, Cham.