Complex Event Processing over Distributed Uncertain Event Streams

XinLong Zhang, YonghengWang , XiaoMing Zhang
College of Information Science and Engineering
Hunan University
Changsha, China
mysileng@gmail.com

***Abstract*—In the 21st century, as technologies of perceptual recognition develops, devices of information generation begin to accurately sense, measure and monitor the physical world in real time.Complex Event Processing(CEP), which can be used to extract user level information from raw data, becomes the key part of the IoT middleware. Most of the current study of complex event processing has not focus on distributed uncertain event streams. In this paper, a high performance method over distributed uncertain event streams is proposed. This method uses improved matched tree that include node buffer list and probability calculation to detect complex event in single uncertain event stream. Distributed stream processing platform Storm also is brought in to improve the performance and stability. Based on Storm, two level parallel methods are proposed to increase the throughput. The experiment study shows that this method is efficient to process complex events over distributed uncertain event streams.**

Keywords-complex event processing; distributed and parallel; uncertain event streams; storm

I. INTRODUCTION

On account of networking requirements of the physical world and expansion demands of the information world, the Internet of Things as a new type of network emerges. A main problem to be solved of wide apply of The Internet of Things is the information processing part. For a RFID application, original events are formed by original signals from RFID reader's reading. Events in reality are various and simple original events cannot be described directly. For instances, "cars are living home", "cars are running to the supermarket", and so on. Those events cannot be described by simple RFID events, but they are meaningful to users. We define complex events which were original events and were processed later to be significant to users. And we call this processing course Complex Event Processing, CEP [1].

Because RFID readers may misread, dirty read and omit events, RFID is of high level of uncertainty. We call the uncertain original RFID data uncertain events. Uncertain events arrange in a row according to the priority of delivery, thus uncertain event flow with timing relationships. Actually in many situations, atomic events read by RFID are uncertain ones. Uncertain event streams widely exist in reality, which contributes to the significance of complex event processing in the uncertain event streams.

Because the Internet of Things is heterogeneous and owns mass data, upper-layer applications oriented to the Internet of Things are almost distributed systems. Generally, during the data processing procedure of applications of the Internet of Things, the business always requires efficiency and real-time performance. Therefore, distributed and parallel computing becomes the key point. In the early days, in the e Internet of Things, complex event processing employs centralized architectures. However, Information era has come where the Internet of Things should process mass data and centralized architectures cannot effectively complete the processing tasks any more.

This problem can be summarized into two challenges to complex event processing of the distributed uncertain event stream. One is how to efficiently detect the real-time event flows of large number and high speed. The other is how to accurately compute the uncertain results caused by uncertain event source. To solve this problem, this paper proposes a processing method oriented to distributed uncertain event stream, Parallel And Distributed Uncertain event Stream Complex Event Processing (PDUCEP). PDUCEP extends the basic method of matched tree to process complex event in single event stream. In order to support distributed uncertain event streams, PDUCEP comes up with two level parallel method, and brings in distributed stream processing platform Storm. Meanwhile, another method is proposed which computes probability of uncertain events based on markov chain.

II. RELATED WORK

Currently, related basic models of complex event processing include: models based on priority automatons[2,3], models based on Petri net [4,5], models based on directed graphs [6,7], and models based on matching tress. SASE is the earliest prototype system that executes complex event processing for RFID data. And SASE is the first that defines querylanguages of describing RFID complex events. Besides, it proposes a method of RFID complex event processing based on query programming. RCEDA method applies time charts to describe and detect complex events. However, this method considers only single complex event detection and ignores intermediate result sharing among complex events.

Among the works ondistributed uncertain complex event processing, R and Suciu focus on uncertain data problems of many applications on InternetofThings, and conclude great challenges from uncertain data faced by researchers uncertain data[10]. Dalvi theoretically explains the foundation of non-deterministic data management and related challenges [11].Mert proposed the CEP method of distributed resources oriented [12].Tao etc proposed a

distributed CEP method of RFID application oriented, but this method is not for large amount of data and to optimize the performance of sliding window [13].

III. PARALLEL AND DISTRIBUTED USCEP

A. Event Model And Probability Calculation

Definition1 simple probabilistic event : simple probabilistic event represented as:<RID,A, T, Pro> where RID is RFID tag and A is the area of RFID reader location, T is time scale of the incident. Pro is the value of the probability of occurrence, it represents the probability of event that RID in area A is Pro at time T.

Definition 2 complex probabilistic event : Complex event is a combination of primitive events or complex events by some rules. A complex probabilistic event is represented as <E, R, Ts, Pr> where E represents the elements that compose the complex event, R represents the rule of the combination, Ts represents the time span of the complex event and Pr is the probability.

In the following we define some operators that can be used query language:

Definition 3 ¬operator : In the sequence of events, matched pattern event is not detected in the specified area. ¬(A)≡A(t). It means that RID is not detected in the area A when time scale is t. Pro(¬(A_1))=1-Pro(A_1).

Definition 4 ∨operator : In the sequence of events, one matching pattern event is detected in the specified area among some. ∨(A,B)≡∃ X(t) X∈{A,B}. It means that RID is detected in the area A when time scale is t. If A_1 and B_1 are independent of each other, Pro(∨(A_1,B_1))=1-Pro(∧(¬(A_1),¬(B_1))).

Definition 5 SEQ operator : In the sequence of events, matching pattern events are detected in the specified area by correct sequence. SEQ(A, B, C)≡$t_1<t_2<t_3$, ∃ A(t_1)∧B(t_2)∧C(t_3). It means that RID is detected in the area A , and then in the area B, in the area C finally.

Definition 6 TSEQ operator : SEQ operator with time span limit. TEQ(A, B; 1,3) ≡$t_1<t_2$, $1 \leqq t_2-t_1 \leqq 3$, ∃ A(t_1)∧B(t_2). It means that RID is detected in the area A when time scale is t1, then after detection of at least one or up to three time scales, again in area B.

In fact, calculation of the complex probabilistic events can be divided into two sets: Set I represents a collection of independent simple probabilistic events; Set D represents a collection of dependent simple probabilistic events. Set I can be calculated similar to definition 3、4 simply; Set D contains one or more Markov chain.It can be calculated by the following formula：

$$Pro(E)=\prod_{e\in I} Pro(e)\bullet\prod_{di\subseteq D}(Pro(e_1)\bullet\prod_{n=1}^{|di|-1} Pro(e_{n+1}\mid e_n)) \quad (1)$$

Where d_i is one Markov chain of dependent set D, e1 represents the first event in the Markov chain, Pro(e_{n+1}|e_n) represents continuous events are related，it is calculated as follows:

$$Pro(e_n,e_{n+1})=Pro(e_n)\bullet Pro(e_{n+1}\mid e_n) \quad (2)$$

In fact, we get the probability of Pro(en+1|en) by the conditional probabilistic table. Shown in table I, it is the conditional probabilistic lookup table of a specific RID. In the beginning, the initial value set manually through the historical experience. With the operation of the system, simple probabilisticstreams continue to flow into the system, the system according to the input stream for simple learning and changing.

TABLE I. CONDITIONAL PROBABILISTICLOOKUP TABLE

RID	
Conditional Event	Probability
B | A	0.9
C | B	0.8
...	...

B. *Improved Uncertain CEP Over Single Event Stream*

Uncertain event streams detection is the main part of the CEP engine. The basic event stream detection method extends matched tree and we add a buffer list for each tree node to buffer the successful matching records. Tree leaf nodes are simple event node, intermediate node is the operator node that contains complex event rule. Each operator node consists of several child node, and each child node may be a leaf node or another operator node. They express the semantics by the logical rules of operator. When an event that associated with the node is detected, the probability value is calculated, and then successful record will be stored in buffer list. In addition, query event window limits the size of buffer list, so old record will be discarded.

Detection is performed by recursively bottom-up. Each event query expression will be converted into the corresponding event matched tree. Whenever simple event that expressed by leaf node is detected, the leat node will update the event record in its history buffer, and notify its parent node that the new record is generated by sending message. Accord to parent node's logic semantics, it will use the new record from this child node and the history records from other child nodes to generate as many new event records as possible. If the parent node gets a new record, and then it will notify its parent node again. This process will continue iteration, and stop in the situation that complex event back to the root node or all node can not produce the new record. When each new record that meets logical rule is generated, it will be calculated new probability value based on child node's probability and semantics (method described in section IIIA).

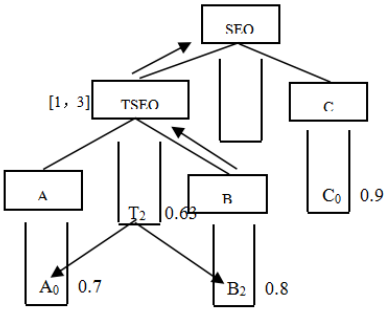

Figure 1. Matched Tree

Now, our query expression is SEQ(TSEQ(A, B; 1, 3), C), so matched tree is showed as figure 1. After simple events (C0, A0, B2) arrive, TSEQ node find that A0 and B2 ment semantics, so it calculate the probability: pro(T2) = pro(A0) * pro(B|A) = 0.63, and generate new record T2 to buffer in history list.

C. Storm And System Architecture

In order to improve system performance and stability of the entire system, we bring in distributed stream processing platform Storm. It is a free, open-source, high fault tolerance, real-time, distributed computing system. Storm is often used in real-time analysis, online machine learning, continuous computing, distributed remote call , ETL and other fields. Figure 2 shows a logical calculation diagram of Storm Topology.

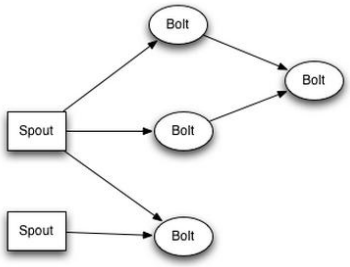

Figure 2. Storm Topology

Storm's core concepts include spout, bolt, topology. Sprout is a data source component, which produces data stream in a topology. Typically spout will read data from an external data source, and then convert them to the internal available data. Bolt handle data, which receive data in a topology and then perform logical tasks. Bolt can receive any number of input stream for processing, and some bolt may also launch new stream to next bolt. The network that made up of spout and bolt, calls topology. Topology is the highest level abstraction in storm. You can submit a topology to storm cluster to run task. Event streams will flow between some spout and bolt.

System architecture shown in Figure 3. Data generated from RFID Reader, and then through data distributer into local cep. All local cep's execution plan are assigned by the global cep. Global cep and N local CEP inter connected by Storm underlying mechanisms (The underlying mechanisms of storm are not belong discussed content of this paper). User submits a query request by cep client, after that the plan scheduling engine generate distributed query plan based on the two level parallel method. Once the plan coms, cep management will call the slave node to execute query plan.

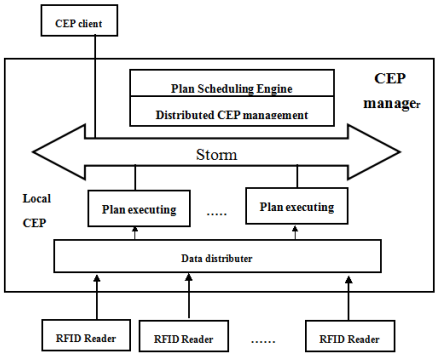

Figure 3. System Architecture

D. Global Parallel

The basic idea of the global parallel design as follows:

(1) After the global CEP engine generate user query plan, we divide each operator node into a single logical computing unit by the structure of query matched tree. We assign N local CEP to run the N logical unit, then all the local CEP is called as sub-task.

(2) Local CEP's responsibility is the task of original operator, so it only two layer typically, very simple operation. The input event streams of bolt node also are the first layer's input. The second layer's output event streams are the complex event that satisfying logic rule. Bolt does not buffer any result(except buffer list), all result move into next node directly.

(3) Each local CEP node is run in parallel. If input event comes, they matching immediately. If output event generates, they also move immediately. All local cep node are not connected unorganized, global cep engine has responsibility to build the logical relationship. The generation of logical relationship is based on matched tree logic operators at all levels. Low-level operator nodes are always treated as an stream source to provide event streams for the high-level operator.

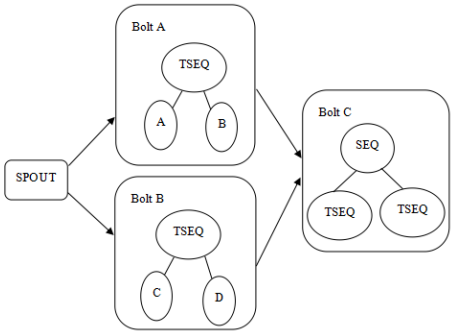

Figure 4. Global Parallel

So all level operation of local cep node look likes the parallel pipeline described in computer architecture. Bolt is the logical execution component in Storm, it can be freely

assembled to any Topology that meet the needs of our own. In PDUCEP, each local cep node can be just the one class of bolt to perform. Because the needs from different levels of computing node coincided with the style of free combination from bolt. Figure 4 shows the example of global parallel.

Now, we assume that there is an query plan: SEQ(TSEQ(A, B; 0, 1), TSEQ(C，D; 1, 2)), and a fragment of input event stream: (a1,b1,c1,d2,b2,d3). Then, let's look at the specific processing through table II.

TABLE II. PROCESSING OF GLOBAL PARALLEL

Time Stamp	Event Input	TSEQ (A，B)	TSEQ (C，D)	SEQ
1	a1,b1,c1			
2	d2,b2	(a1,b1)		
3	d3	(a1,b1) (a1,b2)	(c1,d2)	
4		(a1,b1) (a1,b2)	(c1,d2) (c1,d3)	(a1,b1,c1,d2)
5				(a1,b1,c1,d2) (a1,b1,c1,d3) (a1,b2,c1,d3)

Parallel processing are sequentially performed according to the time slice. The current each operator cell's contents are generated by the previous time slice results and new input event streams. As we can see from the table, after the a1,b1,c1 reach, (a1,b1) is generated in TSEQ(A, B)'s buffer at time slice 2. After d2, b2 reach, TSEQ(A,B) and TSEQ(C,D) get result (a1,b2)、(c1,d2) in parallel at time slice 3. When time slice arrives 4, TSEQ(C, D) generate (c1,d3). Meanwhile,SEQ node get the intermediate result from time slice 3, it generate a complex event result in parallel (a1,b1,c1,d2). According to this iteration, we can get the rest complex events combination: (a1,b1,c1,d3) 、(a1,b2,c1,d3). Since then, we can clearly see that the whole process is based on parallel lines.

E. LocalParallel

Bolt is the logic processing unit in Storm, and it carries the core work of the whole system. In the last section, we make each operator as a class of bolt node. It gets a pipelined parallel and hierarchical method. But we need to know that the instances of one bolt can get more than one in entire storm system. In other words, a class of bolt can generate multiple instances. Each instance of bolt actually is a thread, we call it as Task. A class of Tasks may appear in same server, but most of the situation is that they are not on the same server.

Now, we assume that there is an uncertain input stream fragment: (A1B2A3B4.....A199B200), meanwhile a user's query is TSEQ(A, B; 0, 1). According to the single instance method, we only assign a bolt instance for TSEQ operator, and it need to match 100 times by serial scheme. But if we uses multiple instances plan on this bolt node, and let more instances handleuncertain simple event fragment, then the processing performance will be a larger increase.

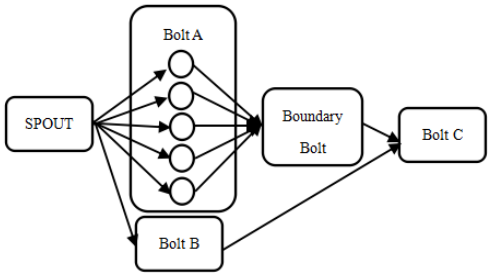

Figure 5. Local parallel

As we can see from figure 5. We expand bolt A into five instances, and the input streams are divided into five equal fragment at the same time. So, the first instance will get the sub-fragment (A1B2A3B4.....A39B40) to run. After that, local result will move into next bolt node. If we have N instances now, then we can divide input streams into N parts by the order in accordance with time slice. Afterward, each fragment can move into each instance P1、P2.....PN。

However, there is a boundary problem that it may miss complex event. At the junction of event fragment that processed by Pi and Pi+1, if the last part of events in Pi and the beginning of events in Pi+1 meet the conditions of complex event, then they may can not run matching algorithm. Because the different instance may at different server, and they can not shared buffer list. In order to solve this problem, we use a boundary bolt to merge boundary data at the end of split flow. Boundary bolt will not check all input stream, on the contrary it only check events within a range of time slice between upper and lower boundary. Time limit is determined by the minimus time window in user's query. Such as, TSEQ(A, B; 0, 1) shows the time limit is 1 time slice. If input events are (A1B2A3B4), (A5B6A7B8), then only B4 and A5 need to be moved into boundary bolt.

IV. EXPERIMENT

In the experimental evaluations we compare the PDUCEP algorithm with the two other methods: Centered USCEP method (we call it CUSCEP) and distributed USCEP method (we call it DUSCEP),USCEP is the single uncertain event streams method that described in section III B.In the distributed USCEP method, we execute USCEP on all slave nodes parallel to get the result and move needed events to the master node but the two parallel method are not used. We use four computers with Intel Xeon E3-1200 processor and 16GB memory. The operating system is CentOS 5.5. A car-based SUMO network emulator is used to automatically generate a variety of RFID and sensor events by different device configurations. The emulator can simulate a variety of car networking scenarios that from simple to complex.

In the first experiment we studied the performance of the three uncertain vent processing methods for different buffer List Size and result shown in figure 6. As we can see, when the buffer List Size is large, throughput of all method are dropped at the same time, because buffer increases and the time that used in buffer operation (addition/delete/find/update) also augment. Furthermore, PDUCEP has better performance than the other two methods. The reason must be the two parallel method achieved the goal.

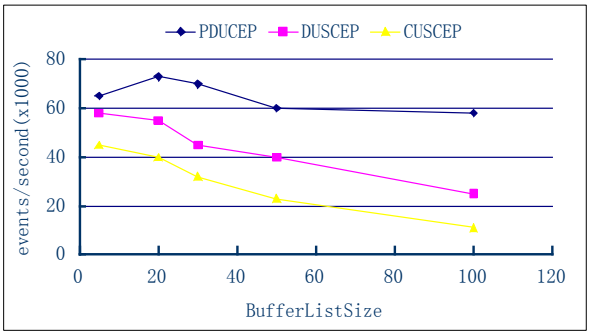

Figure 6. Performance For Different Buffer List Size

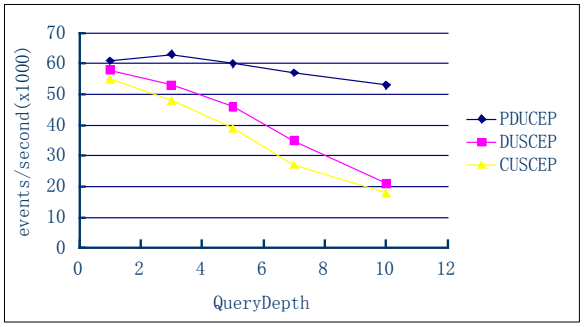

Figure 7. Performance For Different Query Depth

In the next experiment, the buffer List size is fixed at 20. The performance for different QueryDepth is shown in figure7. QueryDepth is the depth of query matched tree. As we can see when the QueryDepth is increased, the performance for all methods is reduced. The reason is that the augment of tree depth makes basic detection processing complex. But the throughput reduction of PDUCEP is less that that of the other methods. Because pipeline layered design, the deeper matched tree becomes, the longer pipeline gets. So the performance reduction for PDUCEP is partly eliminated by the parallel method.

From all the experiments we can see PDUCEP gets better performance and scalability than common methods for large buffer List Size and complex query.

V. CONCLUSIONS

In this paper we propose a method PDUCEP to detect distributed uncertain event Streams. Based on matched tree, it transform the different simple events to complex events with the combination of matched tree and node's buffer list. At the same time, it would calculate the new probability through markov chain. In addition, this method uses Storm to build system architecture. Base on Storm, we design two parallel scheme to improve performance : global and local parallel. The experiments show that PDUCEP is effective when process large uncertain event streams and it can be used for large-scale RFID applications.

ACKNOWLEDGMENT

This project is sponsored by the "Context-aware and proactive complex event processing for large scale internet of things(13JJ3046)" project of Hunan Province Natural Science Fund and the "complex event processing in large scale internet of things (K120326-11)" project of Changsha technological plan.

REFERENCES

[1] David C. Luckham, The power of events: an introduction to complexevent processing in distributed enterprise systems [M].Boston: AddisonWesley, 2002.

[2] E. Wu, Y. Diao , S. Rizvi. High-performance complex event processing overstreams.Proceedings of the 2006 ACM SIGMOD international conference onManagement of data,June 27-29, 2006, Chicago, IL, USA.

[3] J. Agrawal, Y. Diao, D. Gyllstrom, et al. Efficient pattern matching overevent streams.SIGMOD Conference 2008: 147-160.

[4] W. Hu, W. Ye, Y. Huang, et al. Complex Event Processing in RFID Middleware:A ThreeLayer Perspective. Proceedings of the Third InternationalConference on In Convergenceand Hybrid Information Technology, 2008 Vol.1: 1121-1125.

[5] X. Jin, X. Lee, N. Kong, et al. Efficient Complex Event Processing overRFID Data Stream.Proceedings of the Seventh IEEE/ACIS InternationalConference on (2008): 75-81.

[6] F. Wang, S. Liu, P. Liu, et al. Bridging Physical and Virtual Worlds:Complex EventProcessing for RFID Data Streams. Proceedings of the 10thInternational Conference onExtending Database Technology (EDBT'2006),2006: 588-607.

[7] C. Zang, Y. Fan. Complex event processing in enterprise informationsystems based onRFID. Enterprise Information Systems, v.1 n.1, 2007 :3-23.

[8] H. Liu, S. Goto, J. Li. The Study and Application of Tree-based RFIDComplex EventDetection Algorithm. Proceedings of the 2nd InternationalSymposium on ElectronicCommerce and Security(ISECS'2009), NanChang,China, 2009 : 527-531.

[9] Y. Li, J. Wang, L. Feng, et al. Accelerating Sequence Event DetectionThrough CondensedComposition. Proceedings of the 5th InternationalConference on Ubiquitous InformationTechnologies & Applications, Sanya,China, December, 2010.

[10] Rizvi S, Jeffrey S, Krishnamurthy S, et al. Events on the edge.In:Proceedings of the 2005 ACM SIGMOD International Conference onManagement of data. New York: ACM, 2005, 885-887

[11] Dalvi N, Suciu D. Management of probabilistic data foundations andchallenges. In:Proceedings of the 26th ACM SIGMOD-SIGACT-SIGARTSymposium on Principles of Database Systems. New York:ACM, 2007, 1-12

[12] M. Akdere, U. Çetintemel, N. Tatbul. Plan-based complex event detectionacross distributedsources. Proceedings of the VLDB Endowment, v.1 n.1,August 2008.

[13] T. Ku, Y. Zhu, K. Hu, et al. A Novel Distributed Complex Event Processingfor RFIDApplication. Proceedings of the Third International Conferenceon Convergence and HybridInformation Technology, vol. 1, 2008:pp.1113-1117.