ISSN 2349-4506
Impact Factor: 2.785

Global Journal of Engineering Science and Research Management

A DISTRIBUTED COMPLEX EVENT PROCESSING METHOD BASED ON EVENT STREAM PARTITION

Xuanling Chen, Yongheng Wang*
* College of Information Science and Engineering, Hunan University, Changsha 410082, China

DOI: 10.5281/zenodo.401006

KEYWORDS: complex event processing; operational distribution; partition.

ABSTRACT
The CEP technology not only can extract valuable information, but also support real-time analysis and decision-making. And simple CEP technology is more to deal with a single complex event or a small amount of complex events, so we propose a distributed complex event processing method. In this paper, we propose a distributed complex event processing framework and an event partitioning strategy based on query event type and an operational distributed method for partitioning complex queries. And the experimental results show that our method is more effective.

INTRODUCTION
Nowadays, the data is closely related to us. Stock trading, RFID-based retail and inventory management, health-care, financial services, and transport networks, etc, generating enormous data all the time. These data are fast, disorganized, various and in high volume, have to say that we live in a world full of data. On the one hand we are fortunate that the great value of this information, on the other hand we are distressed in the value of the limitations, that is, the value of density is too low. Advances of Complex Event Processing(CEP) technology provides significant new power for humans to sense, understand and manage the world. Driven by massive data, CEP technology not only can extract valuable data and events, but also support real-time analysis and decision-making; In addition, compared to static data records[1], it is more inclined to deal with data streams to extract information of interest. To facilitate the processing, information is often transformed into event sets, including the state of the event, and the relationship, hierarchy and action between events. By mining these events, we can filter unrelated events, abstract low-level events to high-level events and correlate simple events to generate complex events. Ultimately implement matching, judgment, decision making, and forecasting with the support of a variety of detection technologies.

Existing complex event processing systems, such as Esper[2], SASE[3], and Cayuga[4], do not support distributed complex event handling. Before this, people have gradually realized the limitations that centralized processing presented in the face of the large data: due to network congestion and single points of failure and the lack of robustness and scalability, network communication cost is high, and low processing efficiency, and adopt distributed processing. Until now, distributed complex event processing technology is gradually mature. The literature[5] focuses on the problem of best dealing with a large number of event streams in a distributed manner, where the cost model calculation is introduced and the cost of each query is considered. Each query and node are taken into account the model to calculate the cost. The literature[6] proposed the realization of partition, pipeline and distributed operation methods, through a single point of operation and distributed operation to verify the performance of the system. The most important of the distributed approach is the distribution of load between nodes. There are usually two kinds of allocation processing methods: operation distribution[7] and query distribution[8]. The operation assignment is a method of dividing a query into different subquery sequences to handle complex queries. Each step is assigned to a node in the system. Query assignment refers to the allocation of a set of queries between nodes of a distributed system. This paper focuses on the method of operation distribution.

The nodes in distributed complex event processing are independent of each other, so the main problems we face are: the division of data and complex event query sentences. In this paper, we propose a distributed complex event processing framework and an event partitioning strategy based on query event type and an operational distributed method for partitioning complex queries. In this paper, we propose a distributed framework of complex event

processing and introduce an event partitioning strategy based on event types of query, and an operation distributed method for partitioning complex queries. The overall structure of this paper is as follows: the chapter 2 describes the related work in recent years in this direction. The chapter 3 is about the related concept and technology. The chapter 4 is the core of this article: distributed complex event processing. Chapter 5 is the experimental analysis mentioned in Chapter 4. The last chapter is the summary and the work plan in the future.

RELATED WORK

At present, the research of CEP has become more and more mature, and put forward a variety of systems and methods, but there are still many places need to be improved, especially in distributed complex event processing.

SASE[9] is an event management engine based on the NFA, and defines the query language SASE+ for describing the complex event, and which filters and associates the event streams in real time. While it uses the non-deterministic finite automaton with matching buffer pool (NFAb)[10] to match the specific event pattern. It describes a query plan-based approach to effectively implement complex event processing. The literature[11] is optimized on the basis of the above, but the movement of the expired data back and forth, resulting in a large time overhead. Document[12] based on Petri-net for complex event processing, but only for a single complex event, it does not apply to multiple complex event modes. In paper [13], a query-based reuse technique is used to convert the query sequence into equivalent and uniform paradigms according to certain rules, and then extract the same sub-paradigm as reuse. Each of paradigms that has a unified form can share them. In paper [14], a distributed query planning method based on directed acyclic graphs of complex event processing is proposed: The complex tasks are divided into multiple sub-tasks, and then distributed to each node for processing, so as to realize distributed processing. The works in [15] is similar to [16] : Pushing the part of CEP processing logic down to sensor nodes. Processing nodes with embedded CEP capabilities and each node has an integrated CEP engine for information processing, event response and management of complex event processing. But that doesn't get rid of the limitation of the centralized processing, it is still passing the data to the central processor for unified treatment, resulting in network congestion, latency and low processing efficiency. The literature [17] proposed two nodes : processing node and data node. Various requests and data transmission of frequent interaction between the two, can deal with a small amount of the request, but the huge number of complex events is still very limited. The literature of [1] is tried to import distributed plans to network nodes for complex event processing, ultimately reach the distributed event detection. Communicating with each other between nodes and waiting until certain requirements are met before advance to the next process.

All in all, the above work provides a reference and study for us to further research, and we will detail the contents of this paper.

CONCEPT MODEL AND RELATED INTRODUCTION

Basic event: A basic event is a target object that occurs at a time and has a certain semantics. Usually, basic event has correlation with scene[18], in different scenes, the semantics of the events are not the same. Complex event: A complex event is an event that occurs at a certain time period and is formed by an atomic event or a complex event that is compounded with one or more operators[13].

The relationship between the events described by the operator[24]. We have defined four common operators. In which E1 and E2 as basic event or complex event.

1) SEQ(;): Sequence operator---the events occur in sequence. There are constraints between these events, such as (E1;E2), represents that event E2 occurs after E1.
2) OR(∨): Disjunctive form of events, any of the two can occur. (E1∨E2) represents event E1 occurs or event E2 occurs or both.
3) AND(∧): Conjunctive form of events, and both events must occur. (E1∧E2) means that the event E1 and E2 must occur, regardless of the order in which events occur.
4) NGE(¬): address the non-occurrence of events, ¬ E represents no event is detected.

In a distributed query, we need to define a query sequence that refers to a series of sequence of events consisting of query operators and events. The definitions involved are:

1) Sub-event: A sub-event is an event defined in a complex event query sequence that is used to synthesize complex events.
2) Sub-sequence: A sub-sequence refers to a sub-event query that decomposes from a whole query sequence to a complex event.
3) Simple sub-sequence: refers to only the event operator and the corresponding operand (basic event or complex event) of the sub-sequence.

In the process of all the complex event processing, there are rules and test criteria of the model. The event model processing rules used in this paper are SASE+[19]. The behavior rules of events are detected by automatic machine model.

DISTRIBUTED COMPLEX EVENT PROCESSING

The framework of distributed complex event processing

As shown in Figure 1, our distributed complex event processing framework includes the following sections:

1) Query Manager: Manages the entire query requests, sends the required event type to the input adapter for event flow division.
2) Input adapter: formats the data. Receives the data sent by different data sources and formats them as a unified event, and sends the processed event stream to the event processing engine. Embeds a miniature event divider - the event flow is divided according to the type of event being queried.
3) Output adapter: receives the results of the complex event processing engine and processes to form the final result stream to output.

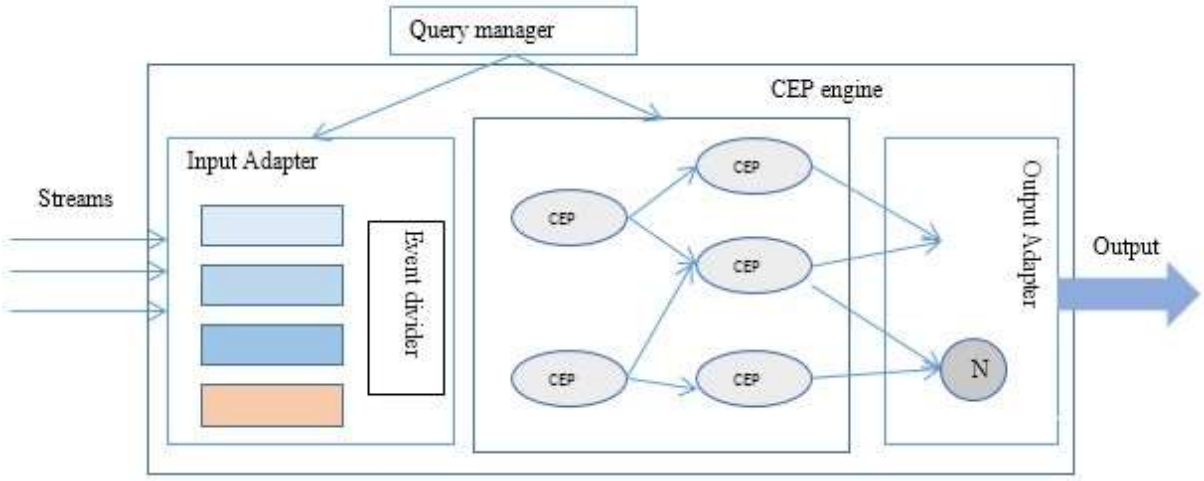

Fig. 1 The framework of distributed complex event processing

4) Distributed complex event processing engine: the core of the entire framework. The main task is to carry out distributed processing (detailed algorithm will be introduced later) based on the division of event flow and complex event query sequence that query manager is submitted. The processing result is sent to the output adapter.

In the distributed complex event processing, there are points we must be clear: how to deploy the event flow and the division strategy of operation in query.

The division strategy of event flow

According to the query sequence pattern sent in the query manager, determine the event type, and then achieve the event division. As shown in Figure 2, our query patterns has SEQ (A, B) and SEQ (B, C, D), thus we will allocate two nodes according to the determined event type (AB) and (BCD). The input adapter simply sends the event stream to the corresponding node for processing.

ISSN 2349-4506
Impact Factor: 2.785

Global Journal of Engineering Science and Research Management

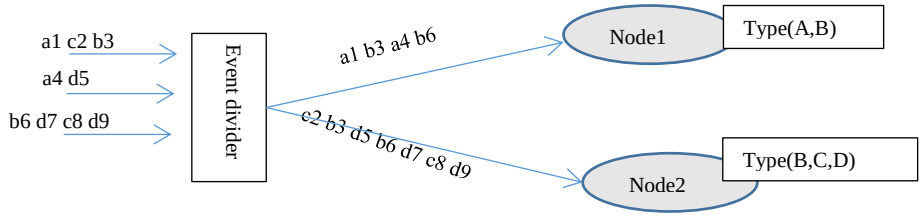

Fig. 2 Event flow division

The distributed method of operation

For the independence of nodes in distributed complex event processing, we divide the complex query sequence statements into different sub-query sequences according to the different operations. Each operation and the corresponding division of sequence can be regarded as a step which is assigned to one node in the system. The nodes are executed independently to achieve distributed processing. The distributed execution steps of operations are as follows:

1. Analyze the complex event query statement, transform it into the suffix expression stack;
2. Take the top element out of the stack and determine whether the element is an event type or an operation type: if the event, then perform step 3; if the operator, then perform step 4;
3. Add the event to EventBuffer[];
4. Combine the operator with EventBuffer[] to generate the operation node and generate its own event cache for that node, and copy EventBuffer[] to the event cache. Then clear the EventBuffer[];
5. Determine whether the type of operation node is a sequence operator. If not, execute the next step. If yes, add the time limit to ensure the order of execution of the left and right operands, and then perform the next step;
6. Determine whether the operation node has been generated, if not, then generate a new node and its event cache and then perform step 7; If yes, update the cache, clear EventBuffer[] after add the events of EventBuffer[], then execute step 7
7. Determine whether the stack is empty, if not empty, then press operation node into the stack; else exit.

The detailed algorithm is as follows:The line 1-4 is the first step; the line 6-9 includes step 2 and 3; the line 10-13 is step 4; the line 14-16 is step 5; the line 17-20 is step 6. The fifth line is the global judgment condition of step 7, that is, whether the stack is empty.Up to now, our distributed operation has been established, and each node performs complex event processing independently and get the results, that will be sent to the output adapter to get the final result. And finally achieve distributed complex event processing.

Algorithm operation of distributed algorithm

Input: Query q
Output: Operation-Node node[],Event-Buffer nodeEventBuffer[]
Method:

```
1.  length = q.length();
2.  InitStack(postexpStack[length]);
3.  q convert to postexpStack[length]
4.  top == length;
5.  while postexpStackIsEmpty()  is true  //stack is not empty
6.     temp = postexpStack[top];
7.     top--;
8.     if temp is the event type;
9.   EventBuffer[i] == temp;
10.      else temp is operator
11.        generate-node node[i] == temp;
12.        generate-eventbuffer nodeEventBuffer[i] == EventBuffer[i];
13.        clear EventBuffer[i];
14.        if  node[i] == ';' ;
```

```
15.          node[i-1].ts<node[i].ts<node[i+1].ts;
16.        else next  stept
17.        if  node[i] is exit;
18.          update nodeEventBuffer[i]
19.          clear EventBuffer[i];
20.        else return to step 11;
21.      end
22.    end
```

RESULTS AND DISCUSSION

Based on the complex event processing system SASE, we can realize a distributed approach to achieve event flow division and operational partitioning. The experimental environment is Intel 2.13GHz, Windows7 with 6GB memory, and the algorithm uses Java programming language.

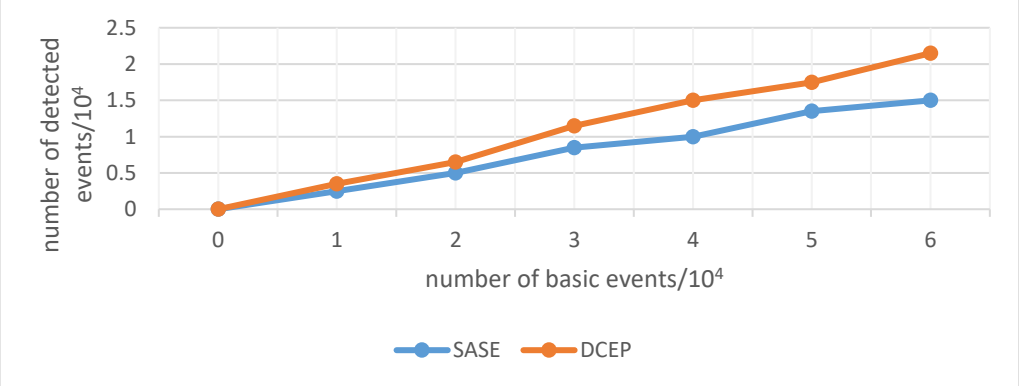

Fig. 3 Experimental results

We compare the SASE and the distributed complex event processing (DCEP) method proposed in this paper. As can be seen from Figure 3, as the number of basic events increases, the number of complex events that can be handled by both is also rising. But it is clear that the method proposed in this paper is more effective because SASE not only uses the active instance stack for intermediate result storage and backtracking, but also has large sliding window for data restriction; While the DCEP method not only divides the event flow, but also extracts the events of the corresponding types of events, and avoids the filtering of other unrelated events. It proposes the way of distributing the operations and divides the complex queries into one small operation query, which will be assigned to the independent node for processing. Overall, our approach is more advantageous.

CONCLUSION

As more and more application of the emergence and development continuously, the generation of data is more rapid and messy. The processing of data stream becomes more and more important, and complex event processing technology has been widely applied to all walks of life. Although in the face of such predecessors have tried many methods, but still need further research. We propose a framework of distributed complex event processing has a good interpretation of our process: The division strategy of event flow based on the type of event. On this basis, the complex query will be divided into sub-query operation, and each operation corresponds to a processing node of operation and event buffer. Experiments show that our method is effective.

ACKNOWLEDGEMENTS

This paper is based on the National Natural Science Foundation of China, "Research on Active Complex Event Processing Technology for Large-scale Internet of Things" (No.61371116).

ISSN 2349-4506
Impact Factor: 2.785

REFERENCES

1. Saleh O, Sattler K U. "Distributed Complex Event Processing in Sensor Networks," IEEE, International Conference on Mobile Data Management. IEEE Computer Society, 2013, pp. 23-26.
2. Esper Tech [Online]. Available: http://www.espertech.com.
3. D. Gyllstrom, E Wu, HJ Chae, Y Diao. "SASE: Complex Event Processing over Streams". Rro of CIDR, pp.407-411, Dec 2007.
4. L. Brerma, A. Demers and J. Gehrke. "Cayuga: a high-performance event Processing engine", in proeeedings of the 2007 ACM SIGMOD international conference on Management of data. pp.1100-1102, June 2007.
5. Kumarasinghe, M., Tharanga, G., Weerasinghe, L., Wickramarathna, U., and Ranathunga, S., "VISIRI - Distributed Complex Event Processing System for Handling Large Number of Queries," Coordination, 2015.
6. Jayasekara, S., Kannangara, S., Dahanayakage, T., Ranawaka, I., Perera, S., & Nanayakkara, V., "Wihidum: Distributed complex event processing," Journal of Parallel & Distributed Computing, 2015, pp. 42-51.
7. Cugola, G., Margara, A., "Deployment strategies for distributed complex event processing," Computing 95(2), 2013, pp. 129–156.
8. Isoyama, K., Kobayashi, Y., Sato, T., Kida, K., Yoshida, M., and Tagato, H., "A scalable complex event processing system and evaluations of its performance," In Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, ACM, 2012, pp. 123–126.
9. Wu E, Diao Y, and Rizvi S, "High-performance complex event processing over streams," ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, Usa, June. DBLP, 2006, pp. 407-418.
10. Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman, "Efficient pattern matching over event streams," 2008.
11. Y. H. Wang, S. H. YANG, "High-Performance complex event processing for large-scale rfid applications," Proceedings of 2010 the 2nd international conference on signal processing systems (ICSPS). China: IEEE, pp. 127-131, July 2010.
12. F. S. Wang, S. R. Liu, P. Y. Liu. "Complex RFID event processing,"The Very Large Databases Journal, vol. 18, no. 4, pp. 913-931' August 2009.
13. Dongdong XU, Lingyun YUAN, "Effective event handling method of Internet of things based on event sharing mechanism," Journal of Computer Applications, 2015, pp. 326-331.
14. Yuan, L., Xu, D., Ge, G., and Zhu, M., "Study on distributed complex event processing in Internet of Things based on query plan," IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, IEEE, 2015, pp. 666-670.
15. JunLi, "Distributed complex event processing based on finite state automata," Journal of Huazhong University of Science and Technology, 2012.
16. Zappia I., Ciofi L., Paganelli F., Iadanza E., Gherardelli M., and Giuli D., "A distributed approach to Complex Event Processing in RFID-enabled hospitals," Euro Med Telco Conference, IEEE, 2014, pp. 1-6.
17. Zhuangzhuang Kang, Qun CHEN, and Linchao SUN, "Study on distributed RFID complex event processing technology," Computer Engineering and Science, 2011, pp. 136-142.
18. Wang z, "Real-time query processing and optimising strategy for RFID complex events," Wuhan: Huazhong University of Science and Technology, 2011.
19. Diao Y, Immerman N, Gyllstrom D., "Sase+: An agile language for kleene closure over event streams," 2007.