Heuristic Event Filtering Methodology for Interval based Temporal Semantics

V. Govindasamy
Research Scholar
Department Of Computer Science and Engineering
Pondicherry Engineering College
Pondicherry, India

P. Thambidurai, PhD.
Principal and Professor, Computer Science and Engineering
Perunthalaiver Kamarajar Institute of Technology , Karaikal, India

ABSTRACT
In this paper, a novel heuristic event filtering methodology that exploits the temporal characteristics in the Complex Event query is presented. Complex Event query is processed in Complex Event Processing(CEP). CEP involves inferring complex events from primitive events in real time. Massive amount of primitive events arrive in real time from multiple distributed sources in real time applications like E-business applications, Business Intelligence systems, Stock Monitoring Systems and Hazard Monitoring Systems. Removal of irrelevant events or filtering will result in effective processing. Thus, there is a need for effective filtering mechanism to filter out the irrelevant events. Therefore, Event Filtering (EF) is to be performed ahead of the event processing. A heuristic event filtering methodology over Sliding Window to increase the throughput of the system is proposed. The proposed system has been validated using a prototype.

Keywords
Complex Event Processing, Event Filtering, Publisher/subscriber model and Event Processing.

1. INTRODUCTION
CEP paradigm deals with inferring knowledge from continuously arriving primitive events. The ultimate aim of CEP is to identify meaningful events. These events may be opportunities in some systems like Business Intelligence systems or threats in hazard detection systems. CEP system is to respond to these opportunities or threats with minimal delay.

The CEP engine inspects the continuously arriving primitive events from multiple sources and detects patterns of interest and notifies to the end users. Patterns are detected by amalgamation of primitive events to composite or complex events. A primitive event is a set of attributes with timestamp. A complex event is the inference derived from a set of multiple primitive events. A complex event is an event of interest to the end users or customers. A pattern or a complex event query is represented in a SQL like query language.

The state of art sensor technology, business process using internet and inexpensive storage has induced high prevalence of primitive events. These primitive events are massive, transient and continuous in nature. The characteristics [2] of the primitive events include

- Enormous amount of events arrive in real time.
- Multiple sources for events exist.
- The events are processed in one pass.
- Bounded temporary storage is required for processing.

These primitive events serve as input to CEP systems [1]. The CEP system combines the primitive events together to detect

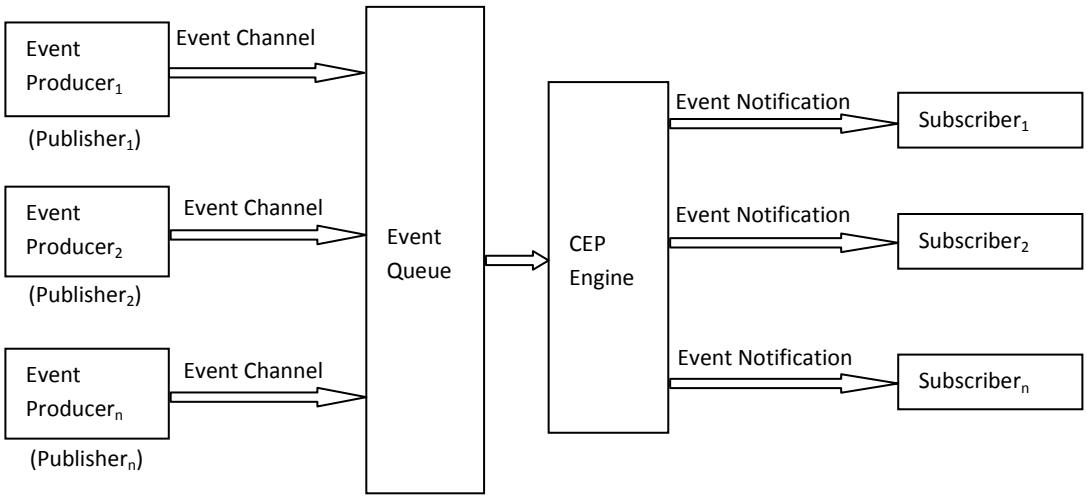

Fig. 1 The Functioning of CEP system

patterns. The Fig. 1 highlights the functioning of the CEP systems. The event producers produce the primitive events. The event consumers consume primitive events from multiple event producers that are distributed across a network. The CEP systems can be modeled as a Publisher/subscriber system. Herein, the publishers are event producers that produce primitive events. The events are accumulated in an event queue for further processing. The CEP engine collates the primitive events to complex events. These complex events are matched with the specific interests registered by the subscribers. If a match occurs, then the corresponding subscriber is notified.

2. RELATED WORK

In this section, a literature appraisal of existing CEP approaches for Event Filtering is presented.

2.1 Filtering

Gianpaolo Cugolaa et al. [1] have implemented an event filtering technique based on static index and counting algorithm. Subscription generalization is an optimization method to decrease the load on the filter component [3]. It is based on selectivity's of subscription. By generalization, the complexity of subscription is reduced and this results in less number of predicates. The two methods of generalizations proposed are generalization by pruning and generalization by replacement. Fusheng Wang et al. [4] have proposed two methods for RFID data filtering. They are i) Low level data filtering ii) Semantic data filtering. Ehab Al-Shaer et al. [5] presents a framework for generic event filtering based on object oriented method. The system captures the common components that may appear in the filtering system. This framework promotes design pattern reuse and code reuse. Ming Li et al. [6] has advocated for a cascade purge that relies on window constraints in CEP based on time interval.

Sven Bittner [7] [10] has designed a Boolean based filtering algorithm. Subscriptions are represented as tree with predicates as leaf nodes and operators as inner nodes. The subscriptions are analyzed and rewritten. Common predicates among subscriptions are identified and indexed. Sven Bitter et al. [8] have proposed a classification scheme for distributed event filtering algorithms - i) Event Forwarding (EF): For higher number of matching events, EF is efficient. ii) Profile Forwarding(PF): For low proportion of matching events, PF is efficient .Kostas Kontogiannis et al. [9] has introduced a new technique - the Event Dependency Graph (EDG), formed by a collection of relations, aims to denote structural and behavioral associations between events in one or more log files. Eight kinds of dependencies have been defined: Coincidental Dependency, Logical Dependency, Topological Dependency, Temporal Dependency, Procedural Dependency, Transactional Dependency, Communicational Dependency and Correlation Dependency.

2.2 Windowing and Sampling

Sven Bittner et al. [11] investigate the memory requirements for the filtering algorithms. There are two classes of filtering algorithms in Publisher/subscriber system - i) Filtering using Boolean subscriptions ii) filtering in canonical forms. It has been inferred that non-canonical approaches show better scalability. If disjunctions are involved, then non-canonical filtering is better. Graham Cormode et al. [12] have generalized the classical reservoir sampling algorithms to continuously maintain a random sample over multiple distributed streams. A new communication efficient protocol for sampling full streams in the Sliding Window has been proposed. Kun-ta Chuang et al. [13] has introduced a new sampling method called feature preserved sampling that generates high quality sample over Sliding Windows. The sampling quality refers to the degree of consistency between the sample proportion and population proportion. Rainer Gemulla et al. [14] has analyzed the various sampling schemes that sample a Sliding Window over recent time and a new method - bounded priority sampling has been introduced in this paper.

The Literature Survey reveals that Uniform Sampling over fixed Sliding Window is an approximation method in data streaming systems. The applicability of which are more relevant. The Sliding Window size is dependent on the time interval mentioned in the query. A further optimization has been proposed in this system by performing sampling of the event records in the Sliding Window. This kind of optimization is crucial because events arrive faster than it can be processed. The events that match the pattern are also much less in comparison to the number of incoming events. The filtering of events can be done by sampling the event records within a window. Thus the query is evaluated in samples of the event stream rather on the stream itself. The efficient filtering will automatically result in the improvement of the systems scalability. Uniform Sampling over Sliding Window in CEP has not been explored.

3. PROBLEM STATEMENT

An efficient scheme to reduce the number of events forwarded to the Complex Event Processing Engine is necessary to increase the throughput of the system. The performance of the existing CEP approaches degrades with the increase in number of primitive events. Thus, there is a need to provide scalability for the CEP system. This is to be achieved by a heuristic method based on the time context specified in the complex query.

3.1 Heuristic Event Filtering for Interval based Temporal Semantics

Subscriptions or queries in CEP systems are continuous in nature. Continuous queries can be evaluated by sliding a window over the recent events. Imposing Sliding Windows on events is a natural method of approximation. Properties in Sliding Window approximation include well definiteness, easily understandable and clear semantics. Filtering techniques based on Sliding Window provides a graceful solution for CEP systems.

Sliding Windows are the standards in interval queries where the context is time. It is used to processes only recent data which are more relevant. The Sliding Window size is dependent on the time interval mentioned in the query. A further optimization has been proposed in this system by performing sampling of the event records in the Sliding Window. This kind of optimization is crucial because events arrive faster than it can be processed. The events that match the pattern are also much less in comparison to the number of incoming events. The filtering of events can be done by sampling the event records within a window. Thus the query is evaluated in samples of the event stream rather on the stream itself. The efficient filtering will automatically result in the improvement of the systems scalability.

The Fig. 2 illustrates the high level conceptual framework of the proposed system. The incoming events are stored in an event queue of bounded size .The events are fed into the filter component. The Sliding Window is used to process the most recent events. The size of the Sliding Window depends on the time interval. The sampler component in the filter partitions the Sliding Window. An event record from each partition is processed. The intermediate event records are kept in an Instance Stack. The sampled event records are processed. If they fail to meet the conjunction predicate, then the events in the Instance Stack may be purged, or else they have to be processed.

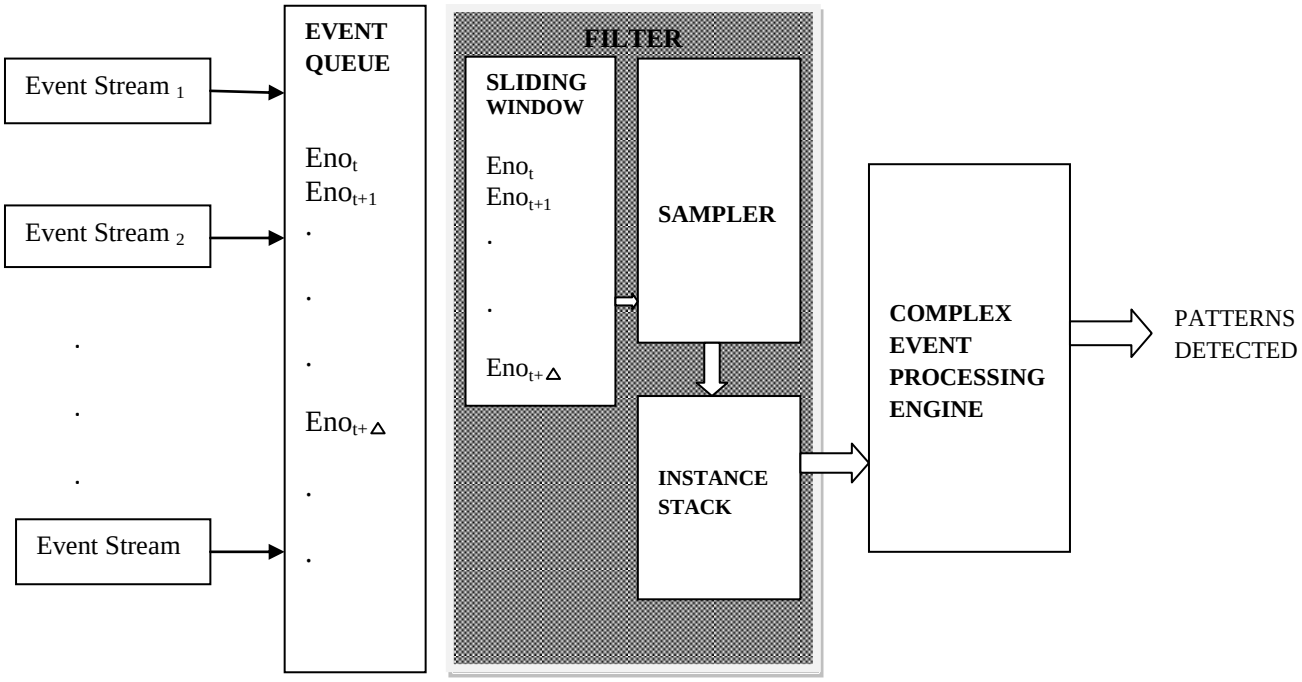

Fig. 2: High Level Conceptual Framework for Heuristic Event Filtering for Interval based Temporal Semantics

3.2 Algorithm for Heuristic Event Filtering for Interval based on Temporal Semantics

To implement this algorithm the infinite event stream E is of the form E= (e_1, e_2,). Each item e_i is of the form (ts_i, $data_i$) where ts_i denotes timestamp and $data_i$ denotes the data. The Sliding Window is represented by Win_Δ (t) = E (t) \ E (t +Δ). Window length is the time span covered by the window. The time interval in the query is given by T. The Instance Stack is represented by I.

Algorithm HEF
Begin
Initialize Win_Δ (t) = E(t) \ E(t + Δ)
Sample e_t from Win_Δ (t)
If *constraint satisfies then*
Begin
While *(t<=(T/2)-1)*
Store event e_{t+1} from E in Instance Stack I
Sample event e_t where t=t+T/2
If *constraint satisfies then*
Examine events from Instance Stack I
Else
Purge the events from Instance Stack I
End
Else
Sample event e_t where t=t+T/2
End

4. PERFORMANCE EVALUATION

4.1 Experimental Setup

The generic application methodology is structured as a message oriented middleware with the set of software components to perform an efficient event processing. These software components communicate through a messaging system called as Java Message Service (JMS) which takes input as event instances from continuously arriving incoming events. The experiment is executed on Windows XP PC with 3.2 GHz processor, 2 GB of RAM and 512 MB cache with the maximum JAVA heap size of 800 Mbytes. It is implemented in open source Java Enterprise Edition/Netbean/Glassfish/JMS environment.

The proposed prototype is to detect the presence of fire hazard using multiple sensors simulating a manufacturing unit.. The prototype uses three different sensors. These sensors act as event sources of input streams. The event sources are temperature sensor, smoke sensor and water sensors. Each event is associated with a timestamp. There are four rules for detection of presence of fire. All the rules have in them a time context. The events are generated using different rates of inputs such as 10 event/second, 20 events/second, 30 events per second etc. The performance of the proposed prototype is compared with a windowing filter using processing time and throughput as parameters.

The experiment is carried out to compare the performance of the CEP approach with and without filtering. The proposed approach achieves high efficiency and scalability compared to the CEP without filtering. The proposed approach filters out the irrelevant events before the event detection starts according to the domain expert specified rules. The performance of the proposed approach is evaluated based on the throughput (number of events executed per second) and

the average processing time (time taken to process the number of events per second).

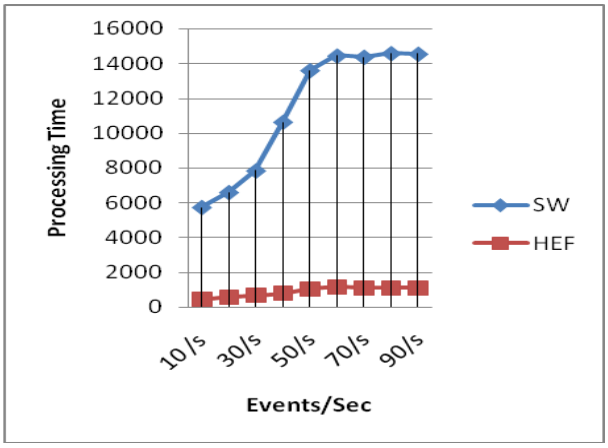

Fig. 3: Processing Time

The Fig. 3 shows the effect of Processing Time with respect to the number of events. From the figure, it can be inferred that for more number of events generated to the rate of 90 events per second, the Heuristic Event Filtering (HEF) performs better when compared to Sliding Window (SW) Filtering. For example, for 90 events per second, the processing time of HEF is less than 2000 milliseconds, whereas the processing time of SW is more than 14000 milliseconds.

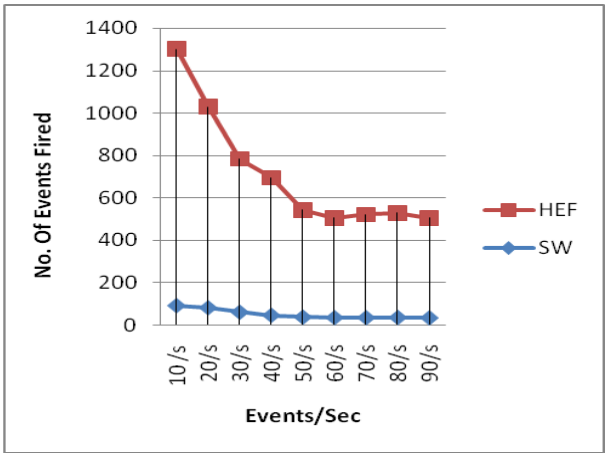

Fig. 4: Throughput

The Fig. 4 compares the Throughput of the Heuristic Event Filtering (HEF) and Sliding Window (SW) Filtering. Throughput is defined as the number of complex events fired per second. The graph records the complex events fired at various arrival rates of events. The graph shows that the number of complex events fired is more for HEF when compared with SW Filtering. For example, when the arrival rate of events is 70 events per second, the number of complex events fired is 20 complex events for SW Filtering, But, HEF achieves around 560 complex events.

5. CONCLUSION

In this paper, a high level conceptual framework for heuristic event filtering for interval based temporal semantics is presented. The system is designed to improve scalability of the Complex Event Processing system. This sampling of events within a Sliding Window is necessary to improve the performance of the filter. The future work will be to complement this filter with other canonical event filters and to study the performance improvement.

6. REFERENCES

[1] Gianpaolo Cugola , Alessandro Margara, "Complex event processing with T-REX", Software, Volume, August 2012,

[2] Georges HEBRAIL, "Data stream management and mining", Mining Massive Data Sets for Security, IOS Press, 2008

[3] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, JenniferWidom, "Models and Issues in Data Stream Systems", ACM PODS, pp.1-16, 2002

[4] Fusheng Wang, Shaorong Liu, Peiya Liu, "Complex RFID event processing", The VLDB Journal, pp.913–931, 2009

[5] Ehab Al-Shaer, Mohamed Fayad, Hussein Abdel-Wahab, Kurt Maly, "Adaptive Object-Oriented Filtering Framework for Event Management Applications", ACM Computing Surveys, Volume 32 Issue 1, March 2000

[6] Ming Li, Murali Mani, Elke A. Rundensteiner, Tao Lin, "Complex event pattern detection over streams with interval-based temporal semantics", DEBS '11 Proceedings of the 5th ACM international conference on Distributed event-based system, Pages 291-302 , 2011

[7] Sven Bittner, Annika Hinze, "Pruning Subscriptions In Distributed Publish/Subscribe Systems", Proceedings of the 29th Australasian Computer Science Conference - Volume 48, Pp. 197-206, 2006

[8] Sven Bittner, Annika Hinze, "Classification and Analysis of Distributed Event Filtering Algorithms", Lecture Notes in Computer Science, Volume 3290, pp 301-318, 2004

[9] Kostas Kontogiannis, Ahmed Wasfy, Serge Mankovskii, "Event clustering for log reduction and run time system understanding", Proceedings of the 2011 ACM Symposium on Applied Computing, Pages 191-192 , 2011

[10] Sven Bittner, "Supporting arbitrary Boolean subscriptions in distributed publish/subscribe systems", Proceedings of the 3rd international Middleware doctoral symposium, 2006

[11] Sven Bittner, Annika Hinze, "A Detailed Investigation of Memory Requirements for Publish/Subscribe Filtering Algorithms", LNCS 3760, pp. 148–165, Springer-Verlag Berlin Heidelberg 2005

[12] Graham Cormode, S. Muthukrishnan, Ke Yi," Continuous Sampling from Distributed Streams", Journal of the ACM, Vol. 59, No. 2, Article 10, April 2012

[13] Sven Bittner, Annika Hinze, "The Arbitrary Boolean Publish/Subscribe Model: Making the Case_The arbitrary Boolean publish/subscribe model: making the case", Proceedings of the 2007 inaugural international conference on Distributed event-based systems, pp. 226 - 237, 2007

[14] Kun-Ta Chuang, Hung-Leng Chen,Ming-Syan, "Chen Feature-Preserved Sampling over Streaming Data" , ACM Transactions on Knowledge Discovery from Data, Vol. 2, No. 4, Article 15, January 2009

[15] Rainer Gemulla, Wolfgang Lehner, "Sampling Time-Based Sliding Windows in Bounded Space Sampling time-based Sliding Windows in bounded space", Proceedings of the 2008 ACM SIGMOD international conference on Management of data,pp.379-392 ,2008.