A Novel Method for Access Policy Consolidation for Event Processing Systems

Raminaidu Allu [1], K.R.Koteswararaoass[2]

[1]Student, Department of CSE, Gokul Institute of Technology and Sciences, India

[2] Professor, Department of CSE, Gokul Institute of Technology and Sciences, India

Abstract:

Event processing is an approach that can capture and process the data about the events. Complex event processing is the merging the information from multiple origins. Event processing systems has a procedure that continuous event streams will be further applied operations of event streams. In distributed-applications like a large warehouse (where items can be shipped) when we are processing the events. This will be transmitting in between many security authorities. Using the access-policy every incoming event can be se-cured. We can increase the processing of events by calculating the measure of obfuscation values for events. Calculate the threshold for obfuscation as one of the part of access-policy and avoid the access requirements and events will be delivered more reliable. In this way we can deliver the more events. We also perform some experiments to assess engines scalability with respect to number of queries and propose ways for evaluating their ability in adapting to changes in load conditions. Lastly, we show that similar queries have widely different performances on the same or different engines and that no engine dominates the other two in all scenarios.

Keywords: event, access-policy, security, obfusca-tion.

Introduction:

Complex Event Processing (CEP)1 has emerged as a new paradigm to monitor and react to continuously arriving events in (soft-)real time. The wide applicability of event processing has drawn increased attention both from academia and industry, giving rise to many research projects and commercial products.CEP has been used for several purposes, including fraud detection, stock trading, supply-chain monitor-ing, network management, traffic monitoring or intensive care units control. Most scenarios where event en-gines are being deployed are mission-critical situations with demanding performance requirements (e.g., high throughput and/or low latency).

Interestingly, the range of scenarios is very broad and presents very different operational requirements in terms of throughput, response time, type of events, patterns, number of sources, number of sinks, scalability, and more. It is unclear what type of requirements demand more from engines, what happens when parameters are varied, or if performance degrades gracefully. To address the lack of event processing performance information, in this paper we make the fol-lowing contributions.

i.We present a number of micro-benchmarks to stress fundamental operations such as

selection, projection, aggregation, join, pattern detection, and windowing.

ii.We perform an extensive experimental evaluation of three different CEP products (two commercial, one open-source), with varying combinations of window type, size, and expiration mode, join and predicate selectivity, tuple width, incoming throughput, reaction to bursts and query sharing There exists a plethora of ap-proaches for implementing event processing networks and dealing with its intrinsic challenges.

A general problem in this context, though, is to analyse the overall behaviour of an EPN. Yet, there is currently no generally accepted formal model for complex event processing. The potential of utilising coloured Petri nets to this end stems from their capability of specify-ing concurrency in an explicit manner with support for typing of events.

Indeed, this merit has been recognized already for ac-tive database systems which promote rule-based processing in a non-distributed environment. In this paper, we investigate the application of coloured Petri nets for specifying and analysing EPNs.

Our contribution is the mapping of concepts from EPNs to coloured Petri nets with a discussion of design choic-es. Further, we report on the validation of the coloured Petri net obtained for the FFDA with its implementa-tion in ETALIS, an open-source event processing engine. Finally, we demonstrate the merits of analysis and simulation capabili-ties for this domain, thereby contributing to the formal foundations of EPNs and opening this emerging field for Petri net analysis.
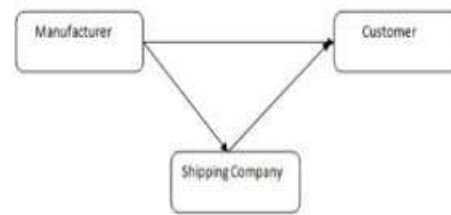

Figure 1. Access Control and Event Dependency

Event Processing Networks:

Event types and events: An event is a happening of interest, an `occurrence within a particular system or domain. Events are typed and an event type is a speci-cation for a set of events with related semantics and structure.

A common model for events is attribute-based, i.e., each event has a set of (required or optional) attributes organised as key-value pairs. For instance, an event of type delivery request' may be characterised by a num-ber of attributes and values, such as `pickup = 24.09.12' and `time = 3 days'.

Event producers and consumers:An EPN, consists of event producers and event consumers. Event produc-ers emit events, eventually event consumers react upon the occurrence of events. Event processing agents act as both event producers and consumers.

Event channels:Event channels link the components of an EPN and forward events without applying any changes to them. They may incorporate routing mech-anisms that limit the set of potential input events for event consumers.

Event processing agents:Components that work on streams of events are called event processing agents
(EPA). We distinguish EPAs that (1) filter events,

(2) transform events, and (3) detect event patterns. A Fillter EPA performs a selection of events, typically based on the event attributes.

A transformation EPA takes events as input, processes them, and produces a set of derived events as output using a stateful or stateless data transformation opera-tion. A pattern detection EPA defines a complex detec-tion logic and outputs derived events.

Event contexts: Event processing agents work on events that are considered to be relevant. This rel-evance is determined by the event context, which is defined along different dimensions.

Most prominently, the temporal dimension partitions events based on their occurrence time, e.g., using a sliding window.

Then, event detection concerns only events that oc-curred within the same window. Eventsmay also be partitioned based on space, external state, or seg-ments of attribute values.

For the aforementioned example, one may require joint processing of events of type `delivery request' and `de-livery bid'. Still, only relevant events of both types shall be considered, e.g., events that occur within a window of two hours and match in their attribute values (e.g., the bid refers to a request).

Shipping companies are hosted in different domains where events will exchange include the confidential in-formation i.e. destination name, product name.

In this exchange of event information third party was received the information about the event but it is con-fidential and given to the access rights to only the ship-ping company.

Communication between producer and consumer

In this event information create access-control that he secured data will be processed multiple domain in large complex event processing systems. This will al-low defining the obfuscation threshold as an access policy avoids all the access restrictions and deliver the events. The number of events will be increased application will be react quickly thus the increase of complex event processing systems can be utilized.
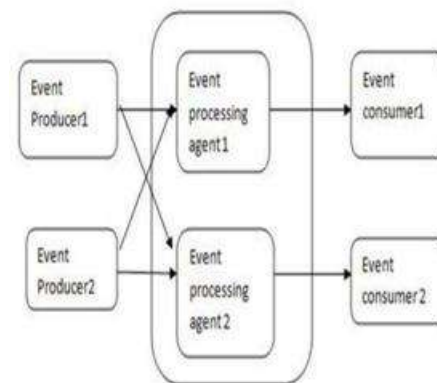
In this proposed system we derive access-requirements by selecting the event attributes for access-policy. This will allows the requirements for a chain of dependent operators in G.

We measure the threshold for obfuscation for attri-butes of events in every Correlation process obfusca-tion of attributes of events are produced is determined by proposed of access-policy consolidation protocol. If threshold for obfuscation is reached for any of the at-tribute that particular attribute Access-Requirement will be avoided.

Access-policies:

Access-control will allows for specified access-rights to set of all event attributes. These Access-Rights are given by the event stream produces and gives the op-erators based upon Access-Requirement i.e. location or role or may be any of the event operators. Requirements are generally indirect properties for all the operators. Generally, we declare the access-rights within the Access-policy and here we declare the Ac-cess-policy as Acp for any operator ω define as set of Attributes and Access-requirements as acr.

Event Processing Overview:

Like Data Stream Management Systems (DSMS) CEP systems are designed to handle real time data that ar-rive constantly in the form of event streams. CEP que-ries are continuous in the sense that they are registered once and then run indefinitely, returning updated re-sults as new events arrive. Due to low-latency require-ments, CEP engines manipulate events

in main memory rather than in secondary storage me-dia. Since it is not possible to keep all events in mem-ory, CEP engines use moving windows to keep only a subset (typically the most recent part) of the event streams in memory. In addition to these features shared with DSMS, CEP engines also provide the abil-ity to define reactive rules that fire upon detection of specific patterns. Ideally, CEP engines should be able to continuously adapt their execution to cope with varia-tions (e.g., in arrival rate or in data distributions) and should be able to scale by sharing computation among similar queries.

Window Policies:

Moving windows are fundamental structures in CEP engines, being used in many types of queries. Windows with different properties produce different results and have radically different performance behaviors. Win-dow policies determine when events are inserted and removed (expired) from moving windows and when to output computations.

Three aspects define a policy:

i.Window type: determines how the window is defined. Physical or timebased windows are defined in terms of time intervals. Logical, count-based, or tuple-based are defined in terms of number of tuples 2.

ii.Expiration mode: determines how the window end-points change and which tuples are expired from the window. In sliding windows endpoints move together and events continuously expire with new events or passing time (e.g., "last 30 seconds"). In jumping or tumbling windows the head endpoint moves continu-ously while the tail endpoint moves (jumps) only spo-

radically (e.g. "current month").

The infrequent jump of the tail endpoint of jumping windows is said to close or reset the window, expiring all tuples at once. In a landmark window one endpoint is moving, the other is fixed, and events do not expire (e.g., "since 01-01-2000").

iii.Update interval (evaluation mode): determines when to output results: every time a new event arrives or expires, only when the window closes (i.e., reaches its maximum capacity/age), or periodically at selected intervals. In general, commercial engines do not sup-port all the combinations above.

All the measures reported represent averages of at least two performance runs after the system reaches a steady state. Event Analysis: the process of analysing suitably prepared events and their payloads and meta-data for useful information.

Tests Specification and Results:

In this section we discuss the results obtained after running the micro-benchmarks on three CEP engines. We emphasize that it is not our intention to provide an in-depth comparison of existing CEP engines, but rath-er to give a first insight into the performance of current products as a way to identify bottlenecks and opportu-nities for improvement. We focus on analyzing general behavior and performance trends of the engines (e.g. variations with respect to window size, tuple width, or selectivity).

We ran our queries on three CEP engines, two of which are developer's editions of commercial products and the other is the open-source Esper. Due to licensing re-strictions, we are not allowed to reveal the names of the commercial products, and will call engines hence-forth as "X", "Y", and "Z". We tried multiple combina-tions of configuration parameters to tune each engine to its maximum performance (e.g., enabling buffering at client side, or using different event formats and SDK versions.

Methodology:

Tests consisted in running a single continuous query at the CEP engine. They began with an initial 1 minute warm-up phase, during which the loadinjection rate increased linearly from 1 event per second to a pre-de-termined maximum throughput4. After warm-up, the tests proceeded for at

•Event Analytics: the use of statistical methods to de-rive additional information about an event or set of events.

•Event Transforms: processes carried out on event payloads or data, either related to event preparation, analysis or processing. least 10 minutes in steady state with the load generation and injection rate fixed at the maximum throughput. Tests requiring more time to achieve steady state (e.g. using long time-based win-dows) had a greater duration.
•Event Tracking: where events related to some entity are used to identify state changes in Event Analysis Analytics, Transforms, Tracking, Scoring, Rating, Clas-sification that entity.

•Event Rating: where events are compared to others to associate some importance or other, possibly rela-tive, measurement to the event.

•Event Classification: where events are associated with some classification scheme for use in downstream processing. benchmarks measure only steady state performance for a fixed number of queries, and do not consider issues such as adaptability and query plan sharing. SPECjms2007 is a benchmark produced and main-tained by the Standard Performance Evaluation Cor-poration (SPEC) aimed at evaluating the performance and scalability of JMS-based messaging middlewares. SPECjms2007 thus focus on the communication side of event-driven systems rather than on query processing, which distinguishes it from our work.

Multiple Queries (Plan Sharing):

The objective of this micro-benchmark is to analyze how the CEP engines scale with

respect to the number of simultaneous similar queries. The query used in this experiment is a window-to-window join similar to Q3.

Test 1: Identical queries. In this test we focus on com-putation sharing and the main metric is hence through-put. Window size is fixed in 1000 rows. To keep output rate fixed (1 output per input event), all queries have a predicate whose selectivity increases as we add more queries.

Test 2: Similar queries with different window sizes. In this test we focus on memory sharing, so windows are large enough to observe differences when we in-crease the number of queries (in the range [400k-500k events]) and the injection rate is low so that CPU does not become a bottleneck.
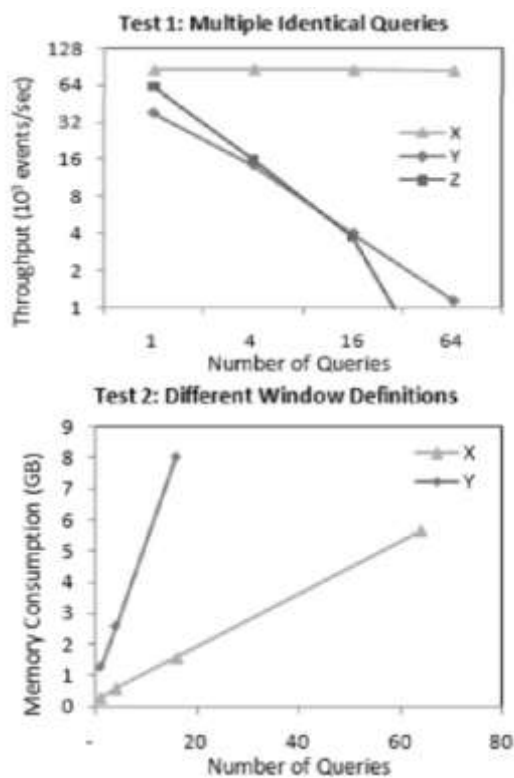
Conclusions and Future Work:

Applications that are realised as EPNs typically process large amounts of events showing a complex interplay. We argued that this calls for appropriate formalisms and tools to analyse the behaviour and performance
of EPNs .Our work addresses this demand by leverag-ing CPNs as a well-established formalism. Still, an au-tomation of a net-based formalisation of EPNs is hin-dered by several factors.

First, despite the advancements on the formal defini-tion of event languages, there is a lack of a generally accepted specification language for EPNs. Second, ex-pressiveness of the languages used to define event contexts, evaluation policies, and the event detection logic is varying.

We showed, however, that these aspects have a large influence on the formalisation. Further work is needed to consolidate the different approaches to EPN model-ling, so that boundaries of any net-based formalisation of EPNs can be made explicit.In this paper we present-ed a performance study of event processing systems.

We proposed a series of queries to exercise factors such as window size and policy, selectivity, and event dimensionality and then carried out experimental eval-uations on three CEP engines. The tests confirmed that very high throughputs can be achieved

by CEP engines when performing simple operations such as filtering. In these cases the communication channel – in our tests, the client API– tends to be the bottleneck.

We also observed that window expiration mode had a significant impact on the cost of queries. In fact, for one of the tested engines the difference in performance between jumping and sliding windows in one test was about 4 orders of magnitude. With respect to joins, tests revealed that accessing data stored in databases can significantly lower the throughput of a system.

Pre-loading static data into CEP engine offers good performance and may thus solve this issue, but this ap-proach is feasible only when data do not change often and fit in main memory. The tested engines had dispa-rate adaptability characteristics.

We observed that the approach used to receive events from clients – either blocking or non-blocking – plays a fundamental role on that aspect, although further investigation is still required to fully understand this topic (e.g., testing bursts of variable amplitudes and durations or having changes in other parameters such as data distributions).

Finally, the tests with multiple queries showed that plan sharing happened only in one CEP engine and only for identical queries (we still plan to broaden the investigation of this topic by incorporating tests with other. Another direction for future work is the deriva-tion of model con_gurations for simulation once a CPN model has been created and validated for an EPN. That relates in particular to the application of data mining techniques to infer event distributions and dependen-cies between event producing components.

REFERENCES:

[1] Björn Schilling, Boris Koldehofe, Kurt Rothermel and
Umakishore Ramachandran. Access Policy Consolida-tion for Complex Event Processing. In Proceedings of the IEEE Conference on Networked Systems (NetSys), pp. 92-101. IEEE.

[2]A. Buchmann and B. Koldehofe, "Complex event processing,"it - Information Technology, vol. 51:5, pp. 241–242, 2009.

[3]A. Hinze, K. Sachs, and A. Buchmann, "Event-based applications and enabling technologies," in Proceed-ings of the Third ACM International Conference on Distributed Event-Based Systems, ser. DEBS '09. New York, NY, USA:ACM, 2009, pp. 1:1–1:15.

[4]P. Pietzuch, "Hermes: A scalable event-based middleware,"Ph.D. dissertation, University of Cam-bridge, 2004.

[5]G. Li and H.-A. Jacobsen, "Composite subscriptions in content-based publish/subscribe systems," in Proc of the 6th Int. Middleware Conf., 2005, pp. 249–269.

[6]G. G. Koch, B. Koldehofe, and K. Rothermel,
"Cordies: expressive event correlation in distributed systems," in Proc. of the 4th ACM International Con-ference on Distributed Event-Based Systems (DEBS), 2010, pp. 26–37.

[7]B. Koldehofe, B. Ottenwälder, K. Rothermel, and U.
Ramachandran, "Moving range queries in distributed complex event processing," in

Proc. of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS),2012, pp. 201–212.

[78B. Schilling, B. Koldehofe, U. Pletat, and K. Rother-mel, "Distributed heterogeneous event processing: Enhancing scalability and interoperability of CEP in an industrial context," in Proc. of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS), 2010, pp. 150–159.

[9]B. Schilling, B. Koldehofe, and K. Rothermel, "Effi-cient and distributed rule placement in heavy constraint-driven event systems," in Proc. of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC), 2011, pp. 355–364.

[10]M. A. Tariq, B. Koldehofe, A. Altaweel, and K. Ro-thermel, "Providing basic security mechanisms in bro-ker-less publish/ subscribe systems," in Proceedings of the 4th ACM Int.Conf. on Distributed Event-Based Sys-tems (DEBS), 2010, pp.38–49.

[11]L. I. W. Pesonen, D. M. Eyers, and J. Bacon,"Encryption-enforced access control in dynamic multidomain pub-lish/subscribe networks," in Proc. of the 2007ACM International Conference on Distributed Event-Based Systems (DEBS), 2007, pp. 104–115.

[12]J. Bacon, D. M. Eyers, J. Singh, and P. R. Pietzuch, "Access control in publish/subscribe systems," in Proc. of the 2nd ACM International Conference on Distribut-ed Event-Based Systems (DEBS), 2008, pp. 23–34.

[13]M. A. Tariq, B. Koldehofe, G. G. Koch, I. Khan, and K. Rothermel, "Meeting subscriber-defined QoS con-straints in publish/subscribe systems," Concurrency and Computation: Practice and Experience, vol. 23, no. 17, pp. 2140–2153,2011.