Processamento de Eventos Complexos: principais motores e cenários de implementação

Rodrigo Otávio Silva de Azevedo[1], **Diogo Menezes Ferrazani Mattos**[1]

[1]LabGen/MídiaCom – TET/PPGEET/UFF
Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

Abstract. *Even in a large-scale development era of technologies such as Internet of Things (IoT) and Big Data, much of the information produced ends being lost or forgotten, due to the lack of an attribution of a sense for its existence. Thus, the complex event processing arises with a bias of allowing the assignment of meaning to information, from the correlations between several more primitive events, so that it is possible to react in real time to formed patterns. Thus, this article proposes a theoretical review on the main concepts about the processing of complex events, in addition to highlighting the most commonly used CEP engines and application scenarios. Furthermore, the differences between the different approaches for each CEP engine are highlighted in view of their main operators and structural characteristics on event processing, as well as their adaptability.*

Resumo. *Mesmo em uma era de desenvolvimento em larga escala de tecnologias como Internet das Coisas (IoT) e Big Data, grande parte da informação produzida termina sendo perdida ou esquecida, devido à falta de atribuição de um sentido a sua existência. Assim, o processamento de eventos complexos surge com um viés de permitir a atribuição de sentido a informações, a partir de correlações entre diversos fluxos de eventos primitivos, para que seja possível reagir em tempo real a padrões de ocorrência de eventos identificados. Assim, este trabalho propõe uma revisão teórica sobre os principais conceitos sobre o processamento de eventos complexos, além de destacar os motores de processamento de eventos complexos (Complex Event Processing - CEP) mais utilizados e cenários de aplicação. Outrossim, destacam-se as diferenças entre as diferentes abordagens para cada motor CEP em vistas dos seus principais operadores e características estruturais sobre o processamento de eventos, bem como sua capacidade de adaptação a diferentes cenários.*

1. Introdução

Aplicações que demandam grandes fluxos de informação e capacidade preditiva estão se tornando cada vez mais comuns. Como exemplo, pode-se citar o mercado financeiro, quando se quer prever a alta ou a baixa de um ativo, ou até mesmo para fins médicos, com o propósito de identificar doenças, ao analisar a evolução de critérios clínicos.

Adicionalmente, tais aplicações, em geral, dependem de dados provenientes de múltiplos sensores e apresentam necessidade de processamento e identificação em tempo real, tornando técnicas do tipo armazenar e processar inviáveis neste contexto. Neste sentido, o processamento de eventos complexo (*Complex Event Processing* - CEP) tem

tomado mais força, sendo uma alternativa para viabilizar a predição de acontecimentos e detecção de anomalias em vista de fluxos de eventos atuais e anteriores.

A tecnologia de CEP apresentam como premissa adquirir fluxos de dados brutos, recebidos por uma fonte de informação, e.g. sensores, dispositivos móveis, e transformá-los em dados complexos, com o enriquecimento de dados e análise de relação condicional e temporal entre eventos. Ademais, há no estado da arte diversos sistemas CEP com diferentes premissas voltados para diferentes cenários de implementação.

Dessa forma, este artigo propõe uma revisão teórica sobre as principais características de sistemas de processamento de eventos complexos bem como a evolução e adaptação deste modelo de processamento de acordo com o cenário de implementação, bem como é feita alusão aos principais motores CEP e suas vantagens e desvantagens, bem como são destacados alguns dos principais cenários de utilização de CEP na literatura.

O restante do artigo esta organizado da seguinte forma. Uma introdução conceitual sobre os fundamentos do processamento de eventos complexos, bem como otimizações para sistemas CEP é feita na Seção 2. A Seção 3 apresenta os principais motores de processamento de eventos do estado-da-arte. A Seção 4 destaca alguns dos cenários de aplicação com exemplos. Na Seção 6 são feitas comparações de características dos principais motores CEP. Por fim, a Seção 6 faz a conclusão do artigo.

2. Processamento de eventos

Nesta seção, serão apresentados os principais conceitos de sistemas de processamento de eventos, com foco em processamento de eventos complexos, bem como diferentes modificações feitas sobre sistemas que implementam CEP para melhorar as capacidades de processamento, bem como para prover ciência ao contexto em alto nível.

2.1. Conceitos iniciais de processamento de eventos

O conceito de evento se relaciona intrinsecamente a alguma ocorrência no ambiente sob análise, capturado por um sistema. Tais eventos são provenientes dos produtores de eventos, que podem ser de vários tipos, tais como sensores, logs de aplicativos, fluxos de dados de vídeo, entre outros. Adicionalmente, mediante a combinação de uma série de eventos simples, por meio de construtores de eventos, derivações, relações temporais e agregações, se torna possível identificar novos padrões de interesse, a serem definidos de acordo com o conjunto de regras e de padrões de eventos, o que caracteriza um evento complexo.

O processamento de eventos é dividido em duas áreas: processamento de fluxo e processamento de eventos complexos (CEP). Neste sentido, o processamento de fluxo está mais focado em análises contínuas sobre dados, segundo operadores tais como filtragem, agregação, enriquecimento e classificação. Dentre as plataformas para processamento de fluxos, pode-se destacar Apache Kafka [Garg 2013], Apache Spark [Zaharia et al. 2016] e Apache Flink [Friedman and Tzoumas 2016], sendo esta última capaz de implementar algumas funcionalidades de motores CEP.

O CEP, entretanto, usa padrões em vez de sequências de eventos simples para detectar eventos compostos e tomar decisões. Cabe ressaltar que os limites entre o CEP e o

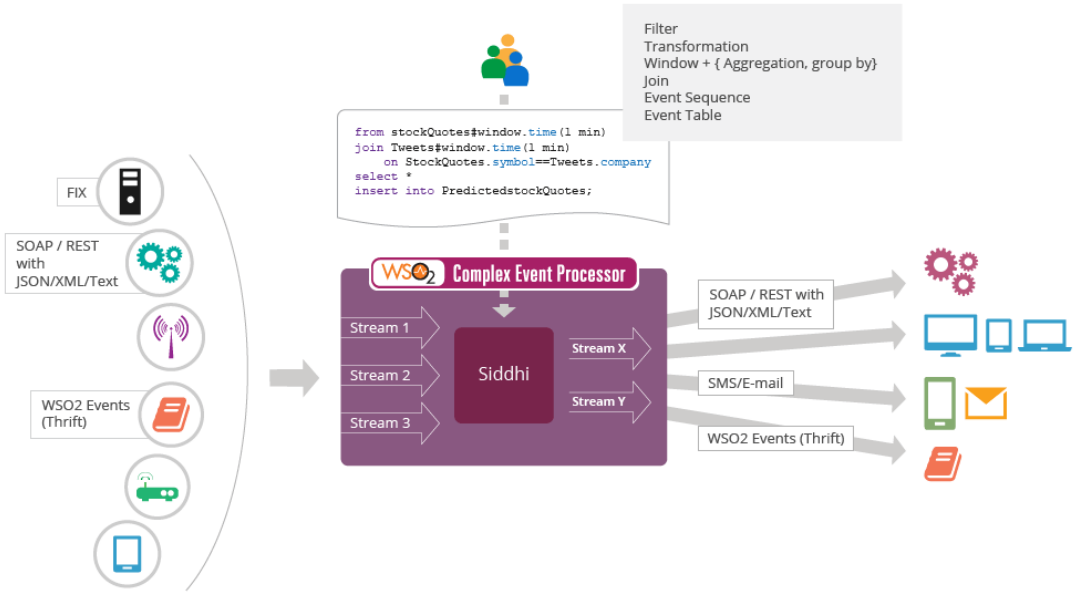

Figura 1. Arquitetura genérica do sistema *WSO2 Complex Event Processor* usando motor Siddhi [1].

processamento de fluxo nem sempre são claros. Segundo [Dayarathna and Perera 2018], em certos trabalhos, o CEP foi implementado como um operador em um sistema de processamento de fluxo de uso geral, no entanto, os motores CEP implementaram suporte para a realização de processamento de eventos de uso geral. Assim, o CEP pode ser considerado uma forma especial de processamento de eventos e, portanto, as teorias e aplicações elaboradas para processamento de eventos são igualmente aplicáveis ao CEP.

2.2. Arquitetura base de um Sistema CEP

Como ilustrado na Figura 1, pode-se dizer que sistemas CEP, em alto nível, são constituídos por elementos básicos, sendo estes: nós fonte, pré-processadores, motor CEP e nós de escoamento. Neste contexto, a função dos nós fonte é realizar a coleta de dados brutos provenientes de sensores e/ou outras fontes de fluxo de informação. Já os pré-processadores são utilizados para fazer a conversão desses dados brutos para linguagem de eventos, de acordo com o motor CEP utilizado. Os motores CEP, por sua vez, aplicam as regras sobre o fluxo de eventos recebido, identificando padrões de interesse. Mediante a detecção de uma situação de interesse, o motor CEP notifica os nós de escoamento, que podem corresponder a diversos tipos de serviço como envio de e-mail, mudança em parâmetros do sistema, entre outras possibilidades.

Na Figura 1, pode-se notar que além dos serviços reativos para onde o motor CEP escoa resultados do seu processamento, há um outro fluxo responsável por armazenar os referidos resultados em uma base de dados histórica, para futura reavaliação das regras e do modelo CEP implementado.

[1]Figura extraída de `https://wso2.com/products/complex-event-processor/`.

2.3. Fundamentos do Processamento de Eventos Complexos

Existem alguns conceitos que podem ser ditos como fundamentais para o processamento de eventos complexos. Assim, nesta subseção serão abordados os principais conceitos voltados para o processamento de eventos.

2.3.1. Tipos de Eventos

Um evento se trata de um objeto que representa ou armazena temporariamente um atividade que aconteceu, está acontecendo ou parece estar acontecendo [Luckham 2008]. Neste sentido, um evento simples pode ser escrito como a tupla $E_s =< id, tempo, tipo, atributo >$, de forma que id se refere a um identificador do evento, tempo se refere a um intervalo $< t_o, t_1 >$, tipo denota o tipo do evento e atributo apresenta um conjunto de atributos que este evento pode armazenar. Os eventos complexos podem ser escritos como a tupla $E_c =< id, tempo, tipo, atributo, membros >$, de tal forma que a tupla $membros =< E_{s1}, ..., E_{sn} >$ se refere ao conjunto de eventos que formam o evento complexo.

2.3.2. Construtores de Eventos

Os construtores de eventos (também conhecidos como operadores de eventos) se tratam de operadores lógicos que estabelecem o relacionamento entre eventos. Deve-se ressaltar que a Linguagem de Processamento de Eventos não apresenta nenhum padrão como SQL para bancos de dados relacionais, porém elas tendem a seguir premissas bem semelhantes à logica proposicional/declarativa, como em Swi-Prolog e YAP.

Dentre os principais construtores de eventos suportados por diferentes CEP comerciais, vale destacar [Alevizos et al. 2015]:

- Seleção (S) - Seleciona eventos que satisfaçam determinadas exigências que outros construtores indicam.
- Disjunção (V) - Representa a ocorrência de um dos dois eventos - Exemplo: $E_c = e_1 V e_2$.
- Conjunção (AND) - Representa a ocorrência de dois eventos simultaneamente - Exemplo: $E_c = e_1 AND e_2$.
- Negação (¬) - Para satisfazer este operador, o evento não deve ocorrer - Exemplo: $E_c = \neg(e_1)$.
- Sequência (;) - Os eventos devem ocorrer um após o outro - Exemplo: $E_c = e_1; e_2$.
- Iteração (*) - Ocorrência do mesmo evento muitas vezes em sequência - Exemplo: $E_c = e_n(*)$.
- Janela (W) - Restringir a ocorrência de um evento a uma determinada janela de tempo, podendo esta ser deslizante ou em lotes.

2.3.3. Regras de Eventos

As regras de abstração de eventos são geradas pela combinação de diversos operadores de modo a identificar padrões de eventos de interesse. Tais regras e padrões são personaliza-

dos pelos gerenciadores de regra, modo a melhor adequar o sistema de processamento de eventos complexos a situações de interesse.

2.3.4. Tipos de Janela

Não se espera que os sistemas CEP detectem eventos complexos considerando em cada ponto de tempo, todos os eventos simples que ocorreram no passado. Para limitar seu espaço de busca, que pode rapidamente se tornar incontrolável, especialmente quando são utilizadas políticas de seleção e de consumo relaxadas, eles normalmente incorporam um operador especial, o de janelamento (W) [Niblett and Etzion 2010].

Uma distinção importante entre os tipos de janela é aquela entre janelas baseadas em tempo e baseadas em contagem (também chamadas baseadas em tupla). As janelas baseadas no tempo impõem uma restrição ao tamanho do intervalo de tempo no qual uma correspondência pode se estender. Com as janelas reais, os eventos pertencentes a uma janela são armazenados em *buffer* e seu processamento começa assim que o tempo da janela expira, para janelas baseadas em tempo, ou o limite de contagem é atingido para aquelas baseadas em contagem. Quando a janela lógica é usada, os eventos simples não são armazenados em buffer, mas processados à medida que chegam [Giatrakos et al. 2020].

2.3.5. Políticas de Consumo de Eventos

As políticas de consumo tratam da questão da seleção de determinada ocorrência, que será utilizada na detecção de um evento complexo. ETALIS [Anicic et al. 2011] propõe três tipos de políticas de consumo de eventos: recente, cronológica e irrestrita. Supondo a ocorrência de um fluxo de eventos A(1), A(2), A(3), B(4), B(5) e C(6), em que no domínio do CEP há um dado padrão que relaciona os eventos A, B e C por $Ec => (A; B; C)$, ou seja, a ocorrência de A, seguida por B e depois por C.

Para política de consumo recente, adota-se o modelo de último a entrar, primeiro a sair (*Last In First Out*, LIFO). Tendo em vista que os eventos no cenário suposto ocorreram de maneira cronológica, o padrão de evento complexo formado pode ser dado como $E_c => A(3); B(5); C(6)$, sendo os demais eventos descartados.

A política de consumo cronológica, também nomeado de pular até a próxima correspondência (*skip-till-next match*, STNM), utiliza o modelo de primeiro a entrar, primeiro a sair (*First In First Out*, FIFO). Tendo em vista que os eventos no cenário suposto ocorreram de maneira cronológica, o padrão de evento complexo formado pode ser dado como $E_c => A(1); B(4); C(6)$, sendo os demais eventos descartados. Além disso, outra maneira de selecionar eventos é a correspondência em estrita contiguidade (*strict-contiguity*, SC), onde ao encontrar a primeira correspondência, todas as demais são descartadas.

Já a política irrestrita, também nomeado de pular até qualquer correspondência (*skip-till-any-match*, STNM) estabelece o máximo de combinações de eventos possíveis entre os eventos presentes no fluxo de eventos, o que na maioria das vezes não é muito prático para modelagem de eventos ($E_c => A(1); B(4); C(6), E_c => A(2); B(4); C(6), E_c => A(3); B(4); C(6), ..., E_c => A(3); B(5); C(6)$).

2.3.6. Modelos de representação de eventos

Muitos mecanismos CEP fornecem aos usuários uma linguagem de padrão que é posteriormente compilada em alguma forma de autômato [Ré et al. 2008]. O modelo de autômato é geralmente usado para fornecer a semântica da linguagem e / ou como uma estrutura de execução para correspondência de padrões. Além dos autômatos, alguns sistemas CEP empregam modelos baseados em árvore. Neste sentido, modelos baseados em árvore são usados para modelagem e reconhecimento, ou seja, eles podem descrever os padrões de eventos complexos a serem reconhecidos, bem como o algoritmo de reconhecimento aplicado. Recentemente, abordagens baseadas em lógica para CEP têm atraído considerável atenção, uma vez que exibem uma semântica formal e declarativa, e ao mesmo tempo têm se mostrado eficientes o suficiente para aplicações de Big Data [Skarlatidis et al. 2015].

2.4. Otimizações para os Sistemas CEP

As formas de conhecimento em fluxos de eventos tendem a se modificar com o tempo e, dessa maneira, estabelecer regras estáticas pode tornar o sistema CEP defasado, quando este dado com variabilidade temporal não é considerado na construção de um modelo de processamento de eventos. Em [Alakari et al. 2020], de modo a compensar influências de fatores temporais sobre o fluxo de eventos, considera-se um fator de peso relativo para redistribuir e atualizar o conjunto de regras, conforme o fluxo de eventos evolui.

Neste sentido, a estratégia de incorporar o conhecimento de domínio de fontes externas, além dos fluxos de eventos, revelam muitos ganhos no enriquecimento do processamento por parte do CEP, gerando alto nível de noção situacional em tempo real. [Binnewies and Stantic 2011] propõe a utilização de ontologia para modelar o conhecimento sobre um dado domínio de modo a permitir raciocínio automático sobre um dado domínio do conhecimento, o que chamou de processamento de eventos complexo melhorado por ontologia (*ontology-enhanced complex event processing*, OECEP).

Ademais, abordagens como iCEP [Margara et al. 2014] e autoCEP [Mousheimish et al. 2016] uso do histórico de eventos em um dado domínio para modificação automática do conjunto de regras e da seleção de padrões de eventos. iCEP [Margara et al. 2014] é testado em um cenário de tráfego veicular, onde fatores como congestionamentos e acidentes são levados em consideração no processo de tomada de decisão e construção de regras dos motores CEP. Entretanto, ao contrário de iCEP, autoCEP [Mousheimish et al. 2016] se limita a uma única fonte de eventos.

O processo de geração automático de regras é também considerado pelo modelo de estimação de densidade de ocorrência de eventos condicional (*Conditional Event Occurrence Density Estimation*, CEODE) [Christ et al. 2016] e para [Akbar et al. 2015b] a partir de Regressão de Janela Móvel Adaptativa (*Adaptive Moving Window Regression*, AMWR) para dados dinâmicos de Internet das Coisas (Internet of Things, IoT),

Uma outra categoria de melhoria voltada para motores CEP está em ferramentas gráficas para visualização e modelagem de características de motores CEP, a partir de situações de interesse. Em [Singh et al. 2012], propõe-se um modelo de validação, refinamento e reavaliação de modelos de situação, chamado de *EventShop*, que fornece uma GUI amigável que permite aos usuários finais selecionar diferentes fluxos e configurar fil-

tros de situação. Com base nas informações de fontes de dados registradas, o EventShop ingere continuamente fluxos de dados espaço-temporais-temáticos e os converte em fluxos de eventos modificados. Esta funcionalidade também é apresentada em StreamLoader [Mesiti et al. 2016], que é também uma interface gráfica interativa capaz de formar regras para sensores heterogêneos a serem aplicados durante a aquisição de fluxo de dados e para visualização de várias características associadas aos eventos para descobrir situações de interesse.

O MEdit4CEP [Boubeta-Puig et al. 2019] é uma solução de interface gráfica com base em redes de petri, que visa fornecer suporte para tomada de decisão em tempo real. Seu principal objetivo é facilitar aos especialistas do domínio a definição de situações de interesse a serem detectadas, bem como no envio de alertas de notificação em tempo real. A principal vantagem de usar o MEdit4CEP para a definição de padrões de eventos é que ele oculta completamente os detalhes de implementação aos especialistas no domínio, sendo este um acessório mais amigável ao usuário.

3. Motores CEP

O processamento de eventos complexos leva em conta fluxos de dados de aplicações em tempo real para inferir sobre padrões e realizar associações com bases de conhecimento previamente estabelecidos. Neste sentido, os motores CEP são entidades capazes analisar e correlacionar eventos à medida que são recebidos, de modo a evitar o armazenamento desnecessário de dados e reagir no caso da da detecção de padrões de interesse.

Dessa maneira, são utilizados motores CEP para realizar a tarefa de processar os eventos recebidos e atribuir regras, conforme estabelecido pelo gerenciador de regras do domínio. Os motores CEP podem ser embarcados em sistemas de processamento de eventos tais como plataformas de de computação de dados distribuídos (*Distributed Stream Computing Platforms*, DSCP), que visam oferecer suporte à distribuição da computação em múltiplos nós em um *cluster*, e plataformas de processamento de eventos (*Event Processing Platform*, EPP), que fornecem funções de filtragem de eventos, correlação e abstração.

Dentre os principais motores CEP do estado da arte, pode-se destacar Esper, Siddhi, Cayuga, RuleCore, C-SPARQL e ETALIS.

3.1. RuleCore

O RuleCore é um motor CEP baseado no conceito de componentes orientados a eventos acoplados, que se comunicam uns com os outros indiretamente usando um modelo de publicação/assinatura. Os dados são armazenados internamente em um banco de dados relacional. Isso permite uma análise, visualização, simulação e relatório offline dos dados coletados usando ferramentas tradicionais de banco de dados [Astrova et al. 2019]. O motor Rulecore utiliza a linguagem XML para representar eventos.

3.2. Cayuga

Cayuga suporta detecção online de um grande número de padrões complexos em fluxos de eventos. Os padrões de eventos são escritos em uma linguagem de consulta baseada em operadores que possuem uma semântica formal bem definida. Isso permite que o

Cayuga execute otimizações de reescrita de consulta [Brenna et al. 2007]. Todos os operadores são combináveis, permitindo que Cayuga construa padrões complexos a partir de subpadrões mais simples. Este motor implementa uma linguagem de consulta exclusiva, Cayuga Event Language (CEL), que usa um formato XML. Adicionalmente, os esquemas de eventos em Cayuga são fixos e tratados como tuplas [Robins 2010]

3.3. C-SPARQL

C-SPARQL é uma extensão do SPARQL com suporte consultas contínuas em fluxos de dados RDF, usando janelas e cláusulas de agregação para suportar o processamento de fluxo RDF. No entanto, C-SPARQL é limitado em sua capacidade de expressar relacionamentos temporais entre padrões RDF, dado que faltam operadores temporais mais expressivos, o que pode dificultar seu escopo de aplicação. Este motor não fornece consultas verdadeiramente contínuas; no entanto, as consultas podem ser avaliadas periodicamente. Ademais, tal motor é incapaz de lidar com janelas de tempo curtas e fluxos de dados de alta frequência [Sheriff and Geetha 2014].

3.4. ETALIS

ETALIS é um sistema dedutivo baseado em regras que atua como um mecanismo de execução unificado para correspondência de padrões temporais e raciocínio semântico. Ele implementa duas linguagens separadas para detecção de padrões e raciocínio semântico: Linguagem ETALIS para Eventos (ELE) e EP-SPARQL para raciocínio de fluxo. Ambos são transformados em regras Prolog e executados por um motor de inferência Prolog. Além disso, o motor ETALIS apresenta recursos como avaliação de eventos fora de ordem, cálculo de funções de agregação, inserção e remoção dinâmicas de padrões e várias políticas e estratégias de consumo de eventos em tempo real.

3.5. Siddhi

Siddhi combina *multithreading*, filas e uso de *pipelining*, consultas aninhadas e fluxos de *threading*, otimização de consulta e eliminação de subconsulta comum de modo a tentar melhorar o desempenho frente a abordagem apresentada por Cayuga. Adicionalmente, vale ressaltar que o motor Siddhi apresenta linguagem de consulta similar a SQL, conexão com banco de dados, apresenta janelas do tipo deslizante, temporal, por tupla e em lotes, além de propiciar agregação de eventos e alta taxa de eventos por segundo [Buddhika et al. 2014].

O motor CEP Siddhi tem sido muito usado embarcado em soluções de código aberto, tais como Apache Flink, Apache Eagle e também em produtos pagos, como WSO2 Complex Event Processor e Uber para análise de fraudes em seus produtos. Além disso, este motor recebe atualizações constantes, sendo capaz de utilizar variados tipos de operadores, tendo um amplo espectro de aplicação. Ademais, cabe destacar que Siddhi apresenta distribuições em Java e Python.

3.6. Esper

Esper utiliza uma linguagem de processamento de eventos chamada Event Processing Language (EPL) que implementa e estende o padrão SQL e permite que sejam utilizadas expressões enriquecidas com ontologias e relações temporais. Adicionalmente, um recurso a se destacar de Esper é capacidade de se conectar ao sistema CEP sob demanda.

Esper é um mecanismo CEP de código aberto altamente eficiente, pois pode processar mais de 500.000 eventos por segundo [2]. Esper EPL é uma linguagem de processamento de alto nível, pois fornece uma grande variedade de operadores temporais e de padrão para a definição de situações de interesse.

4. Aplicações do Processamento de Eventos Complexos

Estabelecer relações entre trabalhos que utilizam CEP, suas métricas de avaliação e motores utilizados, de modo a traçar comparações e destacar pontos positivos e negativos de cada abordagem, quando possível.

4.1. Internet das Coisas

Em [Akbar et al. 2015a] propõe-se uma abordagem distribuída baseada em CEP chamada Micro Complex Event Processing (μCEP), que foi desenvolvida para rodar em dispositivos embarcados com poder de processamento limitado sendo ainda capaz de atualizar as regras em tempo real. Neste sentido, visa-se uma abordagem para encontrar parâmetros otimizados para regras de CEP usando métodos de aprendizado de máquina, clusterização adaptativa, para adaptar o espaço de conhecimento de acordo com o contexto sentido pelos dispositivos. A arquitetura proposta é capaz de inferir eventos complexos a partir de fluxos de dados brutos de uma maneira distribuída, rodando em placas de baixa capacidade de processamento.

Em [Rahmani et al. 2021], um sistema CEP é implementado em cenário hospitalar, de modo a identificação de padrões relativos a saúde de um paciente a partir da análise de dados provenientes de fluxos de dados brutos de sensores de IoT. Neste trabalho, os autores abordam o conceito de rede de área do corpo sem fio (*Wireless Body Area Network*, WBAN), onde dispositivos como oxímetro, eletrocardiograma e termômetros se conectam a um *smartphone*, responsável por ser o intermediário entre a borda e o motor CEP, Esper, que se encontra na nuvem.

O trabalho de [Corral-Plaza et al. 2020] propõe uma arquitetura que permite homogeneizar, processar, serializar e analisar dados heterogêneos e não estruturados para detectar situações de interesse em tempo real. Os dados que esta arquitetura é capaz de processar são obtidos de sensores IoT observando o contexto e fornecendo dados estruturados (JSON / XML) ou não estruturados (CSV) sobre ele, ou seja, sensores de qualidade do ar, gerenciamento de água, estacionamento inteligente, etc. Neste sentido, a arquitetura proposta utiliza o motor Esper devido a sua capacidade de adaptar suas regras em tempo de execução, o que se torna vantajoso em cenários dinâmicos como os de Internet das Coisas. Os dados dos sensores são homogeneizados pelo Apache Kafka e em seguida serializados pelo Apache Avro, sendo então enviados para o motor CEP.

4.2. Detecção de Intrusão

Outra área de bastante interesse para implementação do processamento de eventos complexos é a detecção de intrusão em redes de computadores. Em [Jayan and Rajan 2014] utiliza-se motores CEP baseados em Esper para realizar a detecção de intrusão em redes

[2]Informação retirada do website do Esper: https://www.espertech.com/esper/

domésticas, a partir do pré-processamento de fluxos de dados provenientes de roteadores, além do envio de parte desses dados para um banco de dados, onde são armazenadas situações históricas de interesse, sendo processadas por modelos de aprendizado de máquina, para futuramente alimentar o motor CEP.

Já em [Jun and Chi 2014] realiza-se o processamento de eventos recebido do filtrador de eventos, que é responsável por correlacionar os fluxos de entrada com eventos de IoT e alimentar o motor de eventos complexos, onde se é realizada a consulta com o banco de dados de eventos com padrões para detecção de intrusão. Em seguida, tais resultados são enviados para um motor de ação, que realiza medidas proativas para prever e interromper uma possível intrusão. Entretanto, os dados históricos de eventos não são aproveitados para atualização de regras no motor Esper.

Na mesma linha da abordagem anterior, [Roldán et al. 2020] proporciona um sistema de detecção de intrusão para ataques em dispositivos IoT que propicia a detecção de anomalias não previamente conhecidas pelo motor CEP. Neste caso, os padrões de eventos e de situações de interesse são definidos a partir do GUI do Medit4CEP, em que o modelo gráfico gerado é posteriormente traduzido para linguagem de Esper.

4.3. Demais aplicações

O artigo de [Jin et al. 2019] propõe a utilização de sistemas CEP para aplicações de sistemas educacionais adaptativos (AES), visando a personalização da experiência do usuário final de acordo com uma séria de fluxos de eventos de entrada provenientes do processamento de dados brutos de voz, interação com mouse e teclado, bem como dados fixos do usuário. Neste artigo, os autores utilizam uma estratégia de distribuição do CEP em três camadas (eventos de operação, eventos de atividade e eventos de conhecimento de aprendizado) a partir de múltiplos agentes de processamento de eventos, onde diferentes tipos de dados são correlacionados para geração do espaço de aprendizado do usuário. Vale destacar que o sistema Esper foi utilizado, devido a sua melhor capacidade de lidar com variados tipos de operadores semânticos.

Nesta outra abordagem [Ma et al. 2019], Se trata de um método de identificação de risco baseado em CEP para sistemas de compras online, que inclui principalmente o pré-processamento dos fluxos de dados de compras dos usuários e a formulação dos padrões de eventos de acordo com os comportamentos de risco dos usuários. Adicionalmente, o Esper identifica os comportamentos de compra dos usuários por meio das linguagens de padrão de evento. Por fim, os eventos de risco são enviados ao modelo de processamento de resposta para a operação de aviso prévio. Vale ressaltar que o sistema CEP proposto também conta com uma base de dados de conhecimento para enriquecimento do processamento pelo CEP, bem como realizar atualizações nos padrões de interesse, quando necessário.

5. Comparações de Motores CEP do Estado da Arte

Nesta seção serão consideradas características fundamentais para sistemas CEP, sendo estas: a presença de operadores, tipos de janela, presença de modelo temporal, políticas de seleção e de consumo de eventos.

Na Tabela 1, pode ser vista a relação entre os operadores de acada um dos motores CEP. Percebe-se que abordagens mais recentes e que recebem constantes atualizações

	S	P	AND	V	¬	;	*
Esper	V	V	V	-	V	V	V
Flink	V	V	V	-	V	V	V
ETALIS	V	-	V	V	V	V	V
C-SPARQL	V	-	V	V	-	-	-
Cayuga	V	V	V	-	-	V	V
Siddhi	V	V	V	V	V	V	V

Tabela 1. Motores de processamento de eventos complexos e seus principais operadores

	W	PS	PC	M	T
Esper	Todos	Configurável	Reuso/Consumo	Híbrido	Pontos
Flink	Lógica	Todos	Reuso/Consumo	Automata	Pontos
ETALIS	Lógica	Configurável	Reuso/Consumo	Automata	Pontos e Intervalos
C-SPARQL	-	-	Reuso	Automata	-
Cayuga	-	STAM	Reuso	Automata	Intervalos
Siddhi	Lógica	SC/STAM	Reuso/Consumo	Híbrido	Pontos

Tabela 2. Motores de processamento de eventos complexos e suas características estruturais.

como Siddhi, Esper e Flink apresentam a maioria dos principais operadores.

Já na Tabela 2, encontram-se os mesmos motores CEP, conforme características estruturais, tais como tipo de janela (W), políticas de seleção (PS), políticas de consumo (PC) e modelo de representação (M) e modelo temporal (T). Percebe-se que, tal como para os operadores, abordagens com atualizações mais recentes como Esper e Siddhi podem ser destacadas devido ao alto grau de configurabilidade de seus motores e arquiteturas.

6. Conclusão

O artigo forneceu uma visão abrangente dos principais conceitos de processamento de eventos complexos, na tentativa de detalhar seus principais componentes bem como algumas de suas melhorias descritas no estado da arte. Ademais, alguns dos principais cenários de implementação de sistemas que envolvem o processamento de eventos complexos, dando destaque a internet das coisas e detecção de intrusões. Adicionalmente, foram apresentados os principais motores CEP, com maior prevalência de uso, onde notou-se predominância de Esper sobre as demais soluções. Finalmente, fez-se um comparação entre os motores CEP mencionados no presente trabalho em relação aos operadores e características estruturais, cabendo destacar a capacidade adaptativa de motores CEP mais recentes como Esper e Siddhi, o que justifica sua predominância na literatura contemplada.

Agradecimentos

Este trabalho foi realizado com recursos do CNPq, FAPERJ, RNP, CAPES, CGI/FAPESP (2018/23062-5) e Prefeitura de Niterói/FEC/UFF (Edital PDPA 2020).

Referências

Akbar, A., Carrez, F., Moessner, K., Sancho, J., and Rico, J. (2015a). Context-aware stream processing for distributed iot applications. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 663–668.

Akbar, A., Carrez, F., Moessner, K., and Zoha, A. (2015b). Predicting complex events for pro-active iot applications.

Alakari, A., Li, K. F., and Gebali, F. (2020). A situation refinement model for complex event processing. *Knowledge-Based Systems*, 198:105881.

Alevizos, E., Skarlatidis, A., Artikis, A., and Paliouras, G. (2015). Complex event recognition under uncertainty: A short survey. *CEUR Workshop Proceedings*, 1330:97–103.

Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., and Studer, R. (2011). *ETALIS: Rule-Based Reasoning in Event Processing*, pages 99–124. Springer Berlin Heidelberg, Berlin, Heidelberg.

Astrova, I., Koschel, A., Kobert, S., Naumann, J., Ruhe, T., and Starodubtsev, O. (2019). Evaluating rulecore as event processing network model. Proceedings of the 15th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST, pages 297 – 300.

Binnewies, S. and Stantic, B. (2011). Introducing knowledge-enrichment techniques for complex event processing. In Abd Manaf, A., Sahibuddin, S., Ahmad, R., Mohd Daud, S., and El-Qawasmeh, E., editors, *Informatics Engineering and Information Science*, pages 228–242, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boubeta-Puig, J., Díaz, G., Macià, H., Valero, V., and Ortiz, G. (2019). Medit4cep-cpn: An approach for complex event processing modeling by prioritized colored petri nets. *Information Systems*, 81:267–289.

Brenna, L., Demers, A., Gehrke, J., Hong, M., Ossher, J., Panda, B., Riedewald, M., Thatte, M., and White, W. (2007). Cayuga: A high-performance event processing engine.

Buddhika, T., Ray, I., Linderman, M., and Jayasumana, A. (2014). Secure complex event processing in a heterogeneous and dynamic network. In Kolodny, M. A., editor, *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR V*, volume 9079, pages 61 – 73. International Society for Optics and Photonics, SPIE.

Christ, M., Krumeich, J., and Kempa-Liehr, A. W. (2016). Integrating predictive analytics into complex event processing by using conditional density estimations. In *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 1–8.

Corral-Plaza, D., Medina-Bulo, I., Ortiz, G., and Boubeta-Puig, J. (2020). A stream processing architecture for heterogeneous data sources in the internet of things. *Computer Standards & Interfaces*, 70:103426.

Dayarathna, M. and Perera, S. (2018). Recent advancements in event processing. *ACM Comput. Surv.*, 51(2).

Friedman, E. and Tzoumas, K. (2016). *Introduction to Apache Flink: Stream Processing for Real Time and Beyond*. O'Reilly Media, Inc., 1st edition.

Garg, N. (2013). *Apache Kafka*. Packt Publishing.

Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., and Garofalakis, M. (2020). Complex event recognition in the big data era: a survey. *The VLDB Journal*, 29.

Jayan, K. and Rajan, A. K. (2014). Preprocessor for complex event processing system in network security. In *2014 Fourth International Conference on Advances in Computing and Communications*, pages 187–189.

Jin, D., Shi, S., Zhang, Y., Abbas, H., and Goh, T.-T. (2019). A complex event processing framework for an adaptive language learning system. *Future Generation Computer Systems*, 92:857–867.

Jun, C. and Chi, C. (2014). Design of complex event-processing ids in internet of things. In *2014 Sixth International Conference on Measuring Technology and Mechatronics Automation*, pages 226–229.

Luckham, D. (2008). The power of events: An introduction to complex event processing in distributed enterprise systems. In Bassiliades, N., Governatori, G., and Paschke, A., editors, *Rule Representation, Interchange and Reasoning on the Web*, pages 3–3, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ma, Z., Yu, W., Zhai, X., and Jia, M. (2019). A complex event processing-based online shopping user risk identification system. *IEEE Access*, 7:172088–172096.

Margara, A., Cugola, G., and Tamburrelli, G. (2014). Learning from the past: Automated rule generation for complex event processing.

Mesiti, M., Ferrari, L., Valtolina, S., Licari, G., Galliani, G., Dao, M.-S., and Zettsu, K. (2016). streamloader: An event-driven etl system for the on-line processing of heterogeneous sensor data. In *EDBT*.

Mousheimish, R., Taher, Y., and Zeitouni, K. (2016). autoCEP: Automatic Learning of Predictive Rules for Complex Event Processing. In *International Conference on Service-Oriented Computing*, Banff, Canada.

Niblett, P. and Etzion, O. (2010). *Event Processing in Action*. Simon and Schuster.

Rahmani, A. M., Babaei, Z., and Souri, A. (2021). Event-driven iot architecture for data analysis of reliable healthcare application using complex event processing. *Cluster Computing*, 24(2):1347–1360.

Ré, C., Letchner, J., Balazinksa, M., and Suciu, D. (2008). Event queries on correlated probabilistic streams. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 715–728.

Robins, D. (2010). Complex event processing. In *Second International Workshop on Education Technology and Computer Science. Wuhan*, pages 1–10. Citeseer.

Roldán, J., Boubeta-Puig, J., Luis Martínez, J., and Ortiz, G. (2020). Integrating complex event processing and machine learning: An intelligent architecture for detecting iot security attacks. *Expert Systems with Applications*, 149:113251.

Sheriff, I. and Geetha, A. (2014). A survey of research dimensions in complex event processing. *International Journal of Applied Engineering Research*, 9:7769–7780.

Singh, V. K., Gao, M., and Jain, R. (2012). Situation recognition: An evolving problem for heterogeneous dynamic big multimedia data. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, page 1209–1218, New York, NY, USA. Association for Computing Machinery.

Skarlatidis, A., Paliouras, G., Artikis, A., and Vouros, G. A. (2015). Probabilistic event calculus for event recognition. *ACM Transactions on Computational Logic (TOCL)*, 16(2):1–37.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016). Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65.