Processamento Distribuído de Eventos Complexos (CEP)

Gustavo Baptista

Departamento de Informática, PUC-Rio

gbaptista@inf.puc-rio.br

Resumo. Complex Event Processing (CEP) é uma tecnologia emergente para o processamento em tempo real de fluxos de informações em sistemas baseados em eventos. Com CEP, pode-se expressar interesse sobre padrões de relacionamentos entre eventos, tais como temporalidade, causalidade, dependência e composição. Recentemente, o interesse na área de CEP tem se voltado para soluções com arquiteturas distribuídas, para permitir a escalabilidade na correlação e detecção de situações para um grande número de fontes (potencialmente dispersas geograficamente), regras de processamento, e de consumidores de eventos, presentes em cenários de sistemas de larga escala. Este trabalho tem o objetivo de investigar soluções CEP distribuídas. Para isto, alguns dos sistemas e trabalhos acadêmicos existentes são investigados, com suas principais características listadas, e principalmente, características de distribuição da arquitetura.

1. Introdução

Nos últimos anos, modelos de comunicação e processamento assíncronos orientados a eventos tem sido amplamente estudados pela comunidade acadêmica, difundidos e aceitos pela indústria. A área de processamento de eventos envolve tanto aspectos de comunicação quanto de processamento assíncronos baseados em eventos, que são combinados para permitir abordagens de processamento de eventos que tem trazido benefícios para diversos domínios de aplicação.

Para a comunicação entre processos, por exemplo, o contraste entre o modelo síncrono e assíncrono pode ser ilustrado pela comparação entre modelos síncronos de comunicação do tipo requisição-resposta (e.g protocolos cliente/servidor, tais como HTTP), em que um cliente envia uma requisição para o servidor e fica bloqueado esperando uma resposta, com um modelo assíncrono do tipo publish/subscribe (ou pub/sub) em que a comunicação entre produtores e consumidores de informações é realizada com desacoplamento temporal (não há bloqueio no envio ou recepção de dados) espacial e anônimo (produtores e consumidores de informação não necessitam ter conhecimento uns dos outros), o que leva a um aumento de desempenho tanto na comunicação quanto nas lógicas de processamentos de aplicações. As capacidades dos sistemas pub/sub evoluíram bastante ao longo dos últimos anos, desde a inicial capacidade dos consumidores e produtores de informações especificarem canais de eventos (Sistemas Baseados em Canais), até a capacidade de especificarem expressões de interesse sobre o conteúdo dos eventos (Sistemas Baseados em Conteúdo) [3]. O modelo publish/subscribe hoje se mostra como um modelo robusto para a comunicação em alto desempenho. Arquiteturas pub/sub antes centralizadas, tiveram evoluções para sua distribuição em redes overlays de brokers distribuídos, trazendo escalabilidade e disponibilidade, e mais recentemente, abordagens peer-to-peer com avançadas especificações de qualidade de serviço de forma padronizada, tal como o padrão Data

Distribution Service da OMG [4]. Diversos sistemas comerciais e acadêmicos permitem a comunicação pub/sub.

Já na área de processamento de informações, podemos comparar aplicações que utilizam bancos de dados tradicionais de uma forma síncrona (os bancos de dados são entidades passivas que recebem requisições síncronas das aplicações) com sistemas de processamento de eventos, que permitem que aplicações interajam com eles de forma assíncrona. Este modelo surgiu a partir dos requisitos de domínios de aplicação que necessitam de um sistema que seja uma entidade ativa que dê suporte às suas atividades, como por exemplo, sistemas de monitoramento de processos de negócio, detecção de fraudes ou falhas, segurança, sistemas inteligentes para computação ubíqua e sensível a contexto, dentre muitos outros. Os sistemas de processamento de eventos recebem continuamente fluxos de dados (e.g fluxos de eventos, em geral recebidos através de um modelo de comunicação pub/sub), processam de forma contínua estes dados e enviam de forma assíncrona notificações para partes interessadas (i.e. eventos representando situações de interesse). De fato, tanto a comunicação quanto o processamento de forma assíncrona são complementares, de forma que em geral os sistemas de processamento assíncrono de eventos utilizam algum sistema de comunicação assíncrona.

A área de processamento de eventos é dividida entre sistemas gerenciadores de fluxos de dados (Data Stream Management Systems, ou DSMS) e sistemas de processamento de eventos complexos (Complex Event Processing Systems, ou CEP) [5]. A primeira, é caracterizada por sistemas que são evoluções dos Bancos de Dados Ativos (Active Databases), que utilizam o conceito de queries ativas, isto é, queries de bancos de dados que permanecem produzindo resultados em tempo real sobre um banco de dados quando este é atualizado. Os DSMSs não definem semântica sobre os dados processados ou gerados, deixando para seus clientes esta atribuição. A segunda categoria, a dos sistemas CEP, é uma evolução direta dos sistemas publish/subscribe, em que eventos são as unidades de informação e processamento, e possuem uma semântica bem definida.

O modelo de CEP estende as capacidades do modelo publish/subscribe baseado em conteúdo, com a capacidade de especificar relacionamentos entre diferentes eventos, como por exemplo, causalidade, temporalidade, seqüências, agregações e composições. Um evento complexo é um evento composto por eventos mais elementares que o compõe, isto é, uma abstração de mais alto nível representando uma situação causada pela ocorrência de outros eventos. Modelos de causalidade são associados aos eventos, permitindo um rastreamento completo das causas de um evento (i.e. a seqüência dos eventos que levaram o evento a ocorrer). Agregações sobre eventos também são possíveis, trazendo a capacidade do cálculo de funções sobre os dados observados sobre conjuntos de eventos, tais como médias, máximos, mínimos, dentre outros, o que permite abstrações ainda maiores. A temporalidade entre eventos permite que janelas de tempo sejam especificadas para o processamento de seqüências de eventos que ocorram dentro das mesmas [6]. Por exemplo, poderia ser definido que um padrão de eventos é correspondido quando uma determinada seqüência de eventos ocorre, levando em consideração a ordenação entre os mesmos, e especificando uma janela de tempo para a ocorrência destes eventos (e.g especificar que somente eventos em determinada faixa de horário em um dia são considerados, ou que cada evento na seqüência ocorre em até 5 minutos após o anterior).

Diversos sistemas e linguagens CEP estão disponíveis, tanto acadêmicos como comerciais, para a especificação de padrões de eventos, que são descritos através de regras que expressam o interesse de um consumidor de eventos sobre a ocorrência dos

relacionamentos acima descritos. Dentre algumas categorias de linguagens disponíveis, podemos citar as do tipo ECA (Event-Condition-Action) que permitem a especificação de um padrão de detecção de eventos, uma condição e uma ação a ser executada caso o padrão e a condição sejam atendidos. Em geral estas linguagens permitem que os padrões de detecção de eventos expressem os relacionamentos acima, e além disto, permitam uma consulta externa a uma base de conhecimento (também chamada de contexto) através da chamada de funções, que permite que regras de negócio ou aspectos externos ao processamento de eventos sejam considerados, como por exemplo, consultas à históricos de eventos, bases de conhecimento, dados estatísticos ou regras de negócio, e estados globais de processamento.

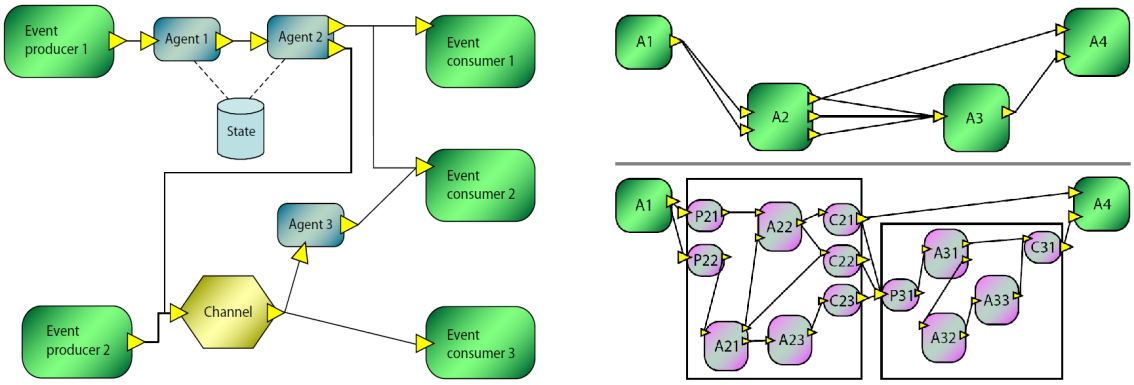
Como já foi mencionado, diversos sistemas CEP acadêmicos e comerciais foram desenvolvidos e disponibilizados nos últimos anos. Entretanto, a área de CEP ainda não foi explorada completamente e ainda está se estabelecendo, com muitos problemas em aberto para a comunidade de pesquisa. Recentemente, o grande aumento na quantidade de fontes e consumidores de dados levaram a comunidade a investigar formas de aumentar a escalabilidade nos sistemas de processamento de eventos. Sistemas CEP centralizados que processam todos os dados originados de uma grande quantidade de produtores de eventos, tem que lidar com um alto custo de processamento, além de uma grande sobrecarga de comunicação. Além disto, alguns cenários são inerentemente distribuídos, com produtores e consumidores de informações distribuídos geograficamente com longas distâncias entre si, o que aumenta ainda mais a sobrecarga e atraso na comunicação. Portanto, a distribuição do sistema de processamento de eventos tem sido um assunto de grande investigação.

Este trabalho tem o objetivo de investigar soluções distribuídas para o processamento de fluxos de informações em sistemas distribuídos que utilizem a técnica de CEP. Para isto, alguns sistemas e trabalhos acadêmicos que implementam CEP são investigados, com suas principais características listadas, e principalmente, características de distribuição da arquitetura para o processamento escalável. A motivação desta pesquisa vem do fato de que o modelo de CEP se adéqua muito bem à área de Computação Móvel Ubíqua e Sensível a Contexto, onde situações de contexto poderiam ser especificadas e expressas através de padrões de eventos complexos, implantados em uma arquitetura distribuída com escalabilidade para um grande número de dispositivos móveis.

2. Event Processing Agents (EPAs) e Event Processing Networks (EPNs)

Uma rede de processamento de eventos (Event Processing Network ou EPN) é uma representação conceitual que permite a representação do comportamento de um sistema de processamento utilizando elementos independentes de plataforma, que descrevem partes da funcionalidade do processamento de eventos de uma forma abstrata. Uma EPN pode ser vista como uma coleção de agentes processadores de eventos, produtores e consumidores de eventos conectados por canais. Os produtores, consumidores, agentes e canais são representados como formas que compões os nós em um grafo.

Estes nós possuem entradas e saídas, mostrados como triângulos. Um agente de processamento de eventos (Event Processing Agent ou EPA) é um módulo de software que processa eventos. Um EPA obtém eventos de entrada, realiza algum processamento, e pode encaminhar ou gerar novos eventos. Dentre os principais tipos de EPAs existentes, podemos citar agentes de Filtragem, Transformação e Detecção de Padrões.



(a) (b)

Figura 1 - Exemplo de Event Processing Network (a) e aninhamento de EPNs (b) [1]

As entidades da EPN podem estar conectadas diretamente ou através de canais de eventos, que representam um canal de comunicação de muitos para muitos, tal como um sistema pub/sub, por exemplo. Além disto, a representação de EPN contém também nós que representam estados globais que podem ser consultados pelos agentes durante o processamento. Representações gráficas de EPNs podem conter aninhamentos de outras EPNs, isto é, o nó de um EPA no diagrama pode representar uma EPN aninhada, ao invés de representar um único agente. Neste caso, as entradas e saídas do agente que contém a EPN aninhada tornam-se os respectivos produtores e consumidores de eventos internos ao agente [1].

Na prática, as diferentes implementações de sistemas e linguagens para processamento de eventos representam estas entidades de diferentes formas como artefatos de software. O modelo de EPN é, portanto, uma abstração que descreve o comportamento funcional de sistemas de processamento de eventos, sem necessariamente representar a realização física das funcionalidades. As EPNs são citadas neste trabalho pois são uma base para entender como o processamento de eventos é organizado logicamente em sistemas CEP. Mesmo que os sistemas não utilizem a representação de EPN, o conhecimento desta notação permite um melhor entendimento da organização dos conceitos que estão sendo utilizados. Outra forma de representação gráfica de mais baixo nível muito utilizada por diversos trabalhos é a representação de regras de padrões de eventos através de autômatos, representando operadores das regras decompostas em estados do autômato e as transições de processamento de eventos entre os mesmos.

Para a implementação ou gerenciamento de um sistema de processamento de eventos, deve-se lidar com as particularidades de uma determinada plataforma, e quais artefatos de software são suportados por ela e por suas linguagens. A implementação de um modelo EPN em artefatos de execução pode ser feita de diversas formas. Estas formas de implementação estão relacionadas o modelo de distribuição e paralelismo utilizado, que é tratado na Seção a seguir.

3. Principais Características

Esta Seção apresenta os aspectos a serem investigados nos sistemas estudados. Dentre diversas características possíveis para estudo, duas principais foram escolhidas pois são as características que definem como um sistema CEP é distribuído e como lida com o conceito de temporalidade e causalidade dos eventos processados.

3.1. Arquitetura de Distribuição e Paralelismo

Diversas aplicações que utilizam processamento de eventos incluem um grande número de fontes e consumidores de eventos, possivelmente dispersas geograficamente, produzindo e consumindo grandes quantidades de informações que o sistema CEP precisa processar de forma eficiente. Então, um aspecto importante é considerar a arquitetura de implantação do sistema, i.e., como os componentes que implementam a arquitetura podem ser distribuídos através de diversos nós para alcançar a escalabilidade.

Em uma arquitetura centralizada de processamento de eventos, o processamento de fluxos de informações originados nas fontes é realizado por um único nó na rede. A EPN é implementada por um único artefato centralizado, que contém a funcionalidade de todos os EPAs e realiza internamente o roteamento de eventos entre eles.

Já em uma arquitetura distribuída, os fluxos de informações são processados por um conjunto de máquinas de processamento, cada uma executando em um nó diferente de uma rede de computadores, que colaboram entre si para realizar o processamento. A EPN é implementada com artefatos de execução separados para cada um dos EPAs e um sistema de notificações responsável por distribuir os eventos entre eles. Esta abordagem provê o máximo de modularidade pois cada artefato pode ser estabelecido em um diferente nó em uma rede de computadores, permitindo uma distribuição e paralelismo na execução do processamento dos agentes.

A distribuição de agentes poderia também ser feita seguindo alguns critérios, por exemplo, artefatos que implementam agentes poderiam ser distribuídos por segmentos relacionados às informações a serem processadas (e.g. perfis de usuários, faixas etárias de clientes, etc), dividindo o processamento em artefatos de execução distintos, cada um responsável pelo processamento de determinado segmento. Outra opção seria uma divisão por partes funcionais de um sistema, por exemplo, ter artefatos responsáveis pelo processamento de requisições de determinada informação, outros por filtragem, agregação e etc. Por fim, a distribuição poderia ser determinada por localizações geográficas em que cada artefato é responsável pelo processamento de determinadas regiões [1].

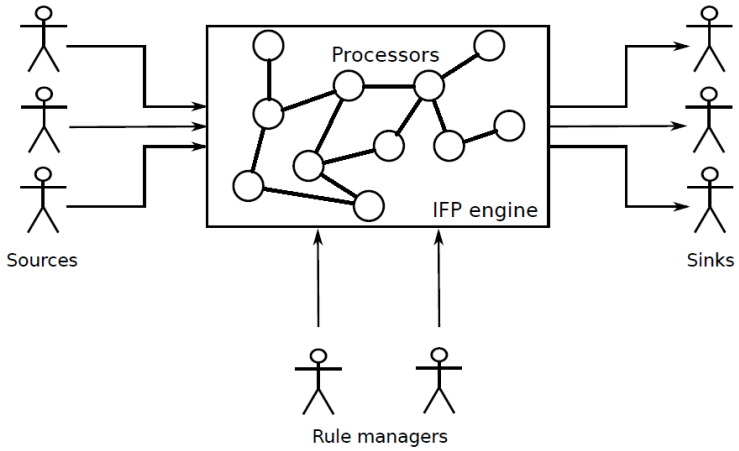


Figura 2 - Organização de uma arquitetura distribuída [5]

Os tipos de arquitetura distribuídas podem ser classificados em duas categorias, sistemas em cluster e sistemas em rede. Em arquiteturas em cluster, a escalabilidade é alcançada com a distribuição do processamento dos fluxos de informações através de um cluster de máquinas fortemente conectadas, que em geral fazem parte de uma mesma rede local. Nestes tipos de arquiteturas, os enlaces que conectam as máquinas de processamento entre si tem desempenho muito maior do que os enlaces que conectam

os produtores e consumidores de fluxos de dados com o cluster. Além disto, os nós processadores ficam em um mesmo domínio administrativo.

Por outro lado, arquiteturas em rede têm o foco em minimizar a utilização de largura de banda das redes, dispersando os nós processadores através de uma WAN, com o objetivo de processar os fluxos de informações o mais perto possível de suas fontes. Como resultado, em uma arquitetura em rede os enlaces que conectam os nós processadores entre si são similares aos que conectam os produtores e consumidores de informações ao sistema. Neste caso, os nós processadores ficam em diferentes domínios administrativos [5].

3.2. Modelo de Tempo

Modelo de tempo é como um sistema relaciona os eventos que entram na arquitetura de processamento e com a passagem do tempo. Esta questão é muito importante, pois este aspecto tem impacto na capacidade de definir ordenação entre os eventos, o que influencia diretamente os operadores oferecidos pela linguagem de regras utilizada e portanto em sua expressividade. Por exemplo, sem uma ordenação total entre eventos não é possível definir seqüências e todos os operadores baseados nelas.

A maioria dos sistemas CEP utiliza um modelo de tempo absoluto, que associa cada item com um timestamp que representa um tempo absoluto, geralmente interpretado como o momento da ocorrência do evento relacionado. Então, estes timestamps definem uma ordenação total entre os itens. Nestes sistemas, os timestamps são totalmente expostos à linguagem de regras, que geralmente provê recursos avançados baseados neles, como seqüências. Estes timestamps podem ser atribuídos pelos produtores de informações quando o evento é gerado, ou pela máquina de processamento quando recebe o evento. Outros sistemas, utilizam um modelo de intervalos de tempo em que dois timestamps obtidos de um ou mais relógios globais representam o momento em que o evento em questão iniciou e o tempo em que terminou. Outro caso, são os sistemas com modelo causal, que associam a cada item algum tipo de marcação que, embora não permita representar um instante absoluto no tempo, pode definir uma ordenação parcial entre itens, geralmente refletindo algum tipo de relacionamento causal, i.e., o fato que a ocorrência de um evento foi causada pela ocorrência de outro evento. O mais importante é que estas marcações e a ordenação que elas impõe estejam disponíveis para a linguagem de regras [5].

4. Sistemas CEP com Distribuição

4.1. NEXT-CEP (Distribuição em Cluster)

O NextCEP é um sistema de processamento de eventos complexos com uma arquitetura distribuída em clusters de nós fortemente acoplados. Ele utiliza uma linguagem de regras que inclui operadores tradicionais SQL, tais como operadores de filtragem e projeção, junto com operadores de detecção de padrões, incluindo seqüências e iterações. O foco principal do projeto é a otimização de regras de padrões de eventos. Para isto, as detecções são realizadas através da tradução das regras para autômatos não determinísticos, que são então processados por uma máquina de otimização de regras, utilizando um modelo de custo baseado nas taxas de eventos de entrada e saída de eventos sobre cada operador presente na regra. A máquina de otimização de regras realiza a reescrita das regras de forma a redefinir a ordem de processamento de seus operadores para se obter um plano de avaliação da regra, implantando os operadores da regra em diferentes nós do cluster, com o objetivo de minimizar a utilização de recursos de CPU e tempo de processamento.

A arquitetura do sistema possui uma entidade centralizada de gerenciamento, chamada de Central Manager, responsável por receber, processar e otimizar queries, as instanciar e implantar seus operadores de regras em nós trabalhadores (Operator Nodes). Um gerenciador de nós trabalhadores dentro do Central Manager é responsável por monitorar quais Operator Nodes estão disponíveis para a implantação de operadores de regras. Cada Operator Node possui proxies para origens e destinos de eventos (Source Proxy e Sink Proxy) que podem ser proxies para produtores ou consumidores de eventos ou outros Operator Nodes, e também instâncias de operadores de regras. Um gerenciador de clientes aceita conexões de clientes e cria um tratador de cliente que recebe queries, as envia para a máquina (Engine) e retorna respostas da Engine para os clientes [7].

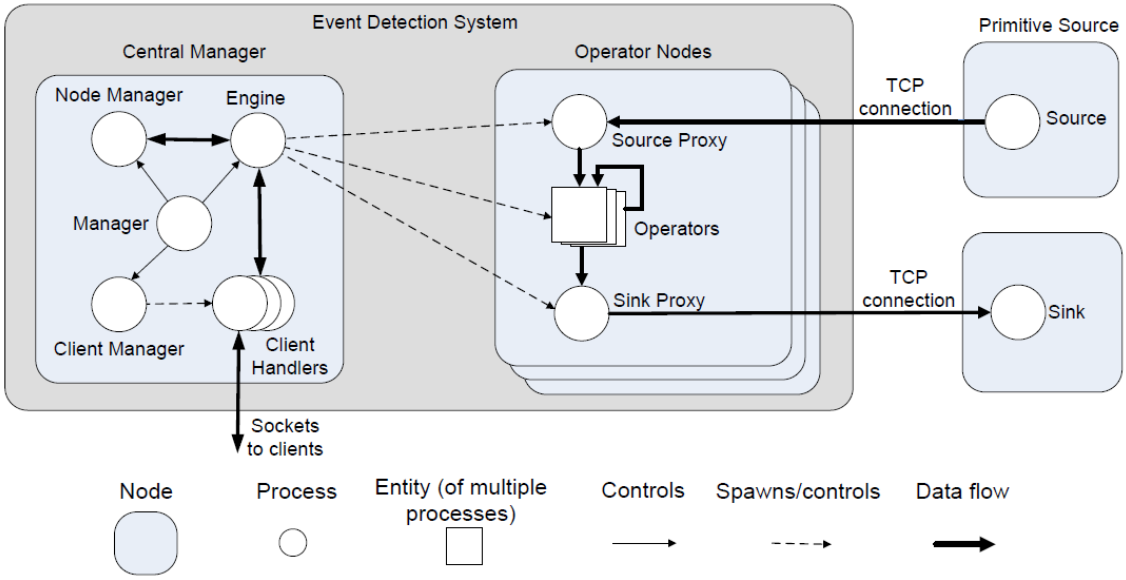


Figura 3 - Visão geral da arquitetura do Next CEP [7]

A Engine realiza o parsing das queries, registra fontes e consumidores, e otimiza as queries recebidas dos clientes criando os operadores e proxies de Sources e Sinks nos nós disponíveis de acordo com o plano de implantação da query. A Engine também mantém um rastreamento de queries que estão em execução e estatísticas utilizadas para otimização de queries. Como o Central Manager tem conhecimento de todos os operadores e suas implantações, ele é responsável pela reutilização de operadores existentes quando novas queries são submetidas.

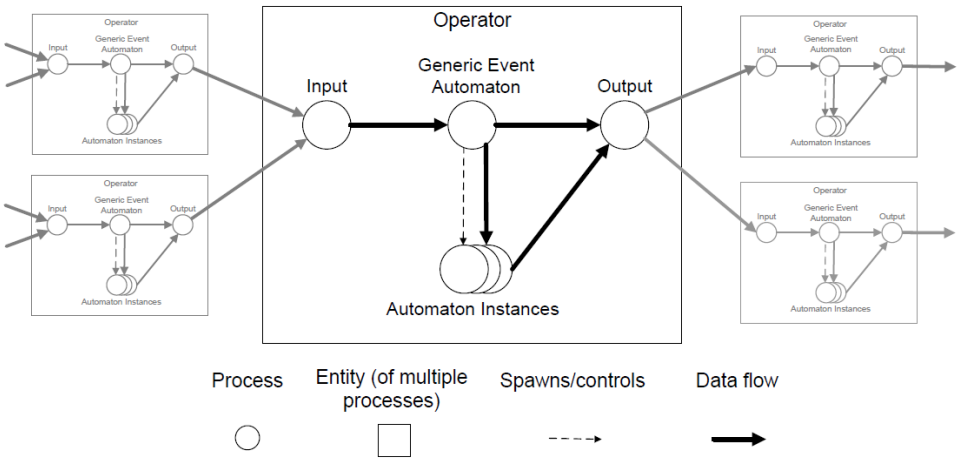


Figura 4 - Visão dos componentes e processos de um Operator Node [7]

Os Operator Nodes executam os operadores e os proxies. As fontes e destinos de fato de eventos não são parte do sistema e são conectadas através de conexões TCP para garantir a ordenação dos pacotes e de acordo com o modelo de tempo do sistema. Os operadores em um Operator Node podem enviar eventos uns para os outros e também para operadores em outros nós na rede. Cada Operator Node possui quatro tipos de processos. Eventos de entrada são recebidos por um processo de entrada e são então processados por um Autômato Genérico de Eventos. Este autômato é instanciado com a especificação de um tipo de operador particular para ser executado. O Autômato Genérico de Eventos cria instâncias de Autômatos de acordo com a semântica do operador. Um processo de saída envia eventos complexos detectados para outros operadores ou Sink Proxies e também permite que operadores sejam reutilizados.

Não há a utilização de um tempo global na arquitetura, mas além das fontes de eventos serem sincronizadas utilizando o protocolo de sincronização de tempo em rede (Network Time Protocol ou NTP), é utilizada uma política de garantia de detecção [7, 8]. Em uma política de garantia de detecção, um evento é consumido a partir de um fluxo de eventos de entrada apenas quando o mesmo se torna estável. Sabe-se que um evento é estável depois que outro evento com um timestamp posterior ao mesmo originado a partir da mesma fonte é recebido pelo mesmo fluxo de eventos (as fontes de eventos utilizam TCP para a transmissão ordenada de pacotes). O consumo de eventos estáveis garante que falsos positivos de eventos complexos não sejam detectados. Portanto, a detecção correta não depende de sincronização de relógios dos nós, que são sincronizados apenas para que haja uma precisão mínima entre os mesmos.

4.2. PADRES (Distribuição em Rede)

PADRES é um sistema pub/sub distribuído baseado em conteúdo que provê processamento de eventos complexos (CEP). Sua linguagem para definição de subscrições é baseada no modelo tradicional de subscrição baseada em conteúdo, onde a subscrição é um conjunto de regras expressas como predicados. As regras expressas pode envolver mais de um item de informação, permitindo composições, seqüências e iterações. As composições são definidas através de subscrições compostas, que são formadas por diversas partes menores, chamadas de subscrições componente, relacionadas entre si por operadores lógicos relacionais do tipo AND e OR. Se uma subscrição composta é correspondida, isto significa que um padrão de eventos, chamado de evento complexo foi detectado, que representa uma abstração de mais alto nível. Padres utiliza um modelo de tempo absoluto, que pode ser referenciado nas regras para a escrita de padrões complexos com restrições temporais, com a utilização de janelas de tempo [9].

O sistema é formado por um conjunto de brokers conectados através de uma rede overlay. A rede overlay é um conjunto de conexões que formam a base para o roteamento de subscrições, advertisements e publicações. O roteamento de mensagens no PADRES é baseado no modelo de publish/subscribe/advertise estabelecido pelo projeto SIENA [10]. Os dados de roteamento overlay são armazenados em tabelas de roteamento overlay em cada broker. Especificamente, cada broker conhece seus vizinhos a partir de suas tabelas de roteamento.

Cada broker possui duas tabelas de roteamento, uma para subscrições e outra para publicações. Estas tabelas de roteamento permitem a propagação de subscrições e publicações apenas para as partes interessadas do sistema. Quando um publicador deseja realizar uma publicação na rede, ele antes envia um advertisement que é transmitido para todos os brokers na rede. Os brokers recebem este advertisement, e armazenam na tabela de roteamento de subscrições (Subscription Routing Table ou SRT). Desta forma,

sempre que uma subscrição é recebida por um broker, esta será encaminhada apenas para brokers vizinhos para os quais exista um advertisement correspondente. Além disto, as subscrições também são armazenadas em uma tabela de roteamento de publicações (Publication Routing Table ou PRT), para que quando uma publicação for recebida, esta será encaminhada apenas para os brokers vizinhos que contenham subscrições correspondentes. A Figura 5 mostra um exemplo de advertisement, subscrições e publicações, os números indicam a ordem de execução [9].

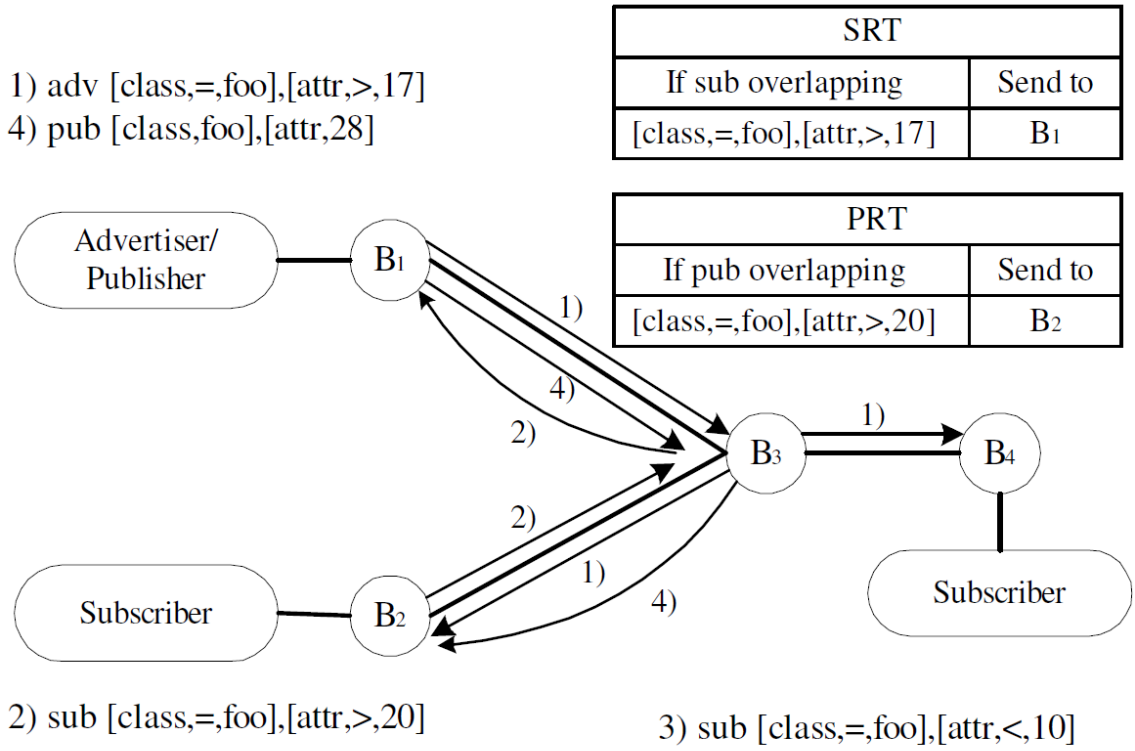


Figura 5 - Exemplo de advertisement e roteamento de subscrição e publicação [9]

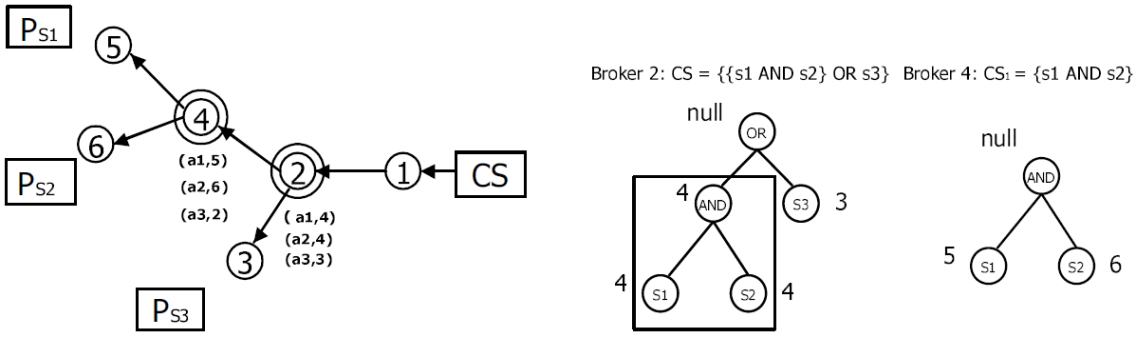
Uma subscrição composta é representada por uma árvore de subscrição (vide Figura 6), onde os nós internos são operadores lógicos e os nós folha são subscrições primitivas. Uma subscrição composta é mapeada para uma regra composta na máquina de regras interna a um broker e inserida na tabela de roteamento de publicações, e cada subscrição componente é parte desta regra. Quando uma publicação correspondendo a uma das subscrições componentes é recebida, a subscrição componente é armazenada em um estado parcialmente correspondido. Quando todos os componentes são correspondidos, a subscrição composta é considerada correspondida.

A detecção de eventos complexos é realizada de forma distribuída. Com o conhecimento dos advertisements, cada subscrição possui seu destino em um Broker. Se todas as subscrições componente tiverem o mesmo destino, um broker deve encaminhar a subscrição composta como um todo para o destino, senão, a subscrição composta deve ser dividida em partes de acordo com os diferentes destinos, e cada parte encaminhada para seu destino separadamente.

O esquema de roteamento é utilizado para detectar o padrão de eventos correspondente a uma subscrição composta em uma localização que seja o mais próxima possível às fontes de dados. Uma subscrição composta é mapeada para uma única regra na máquina de regras interna a um Broker, a qual atua tanto como um roteador de mensagens como um detector de eventos.

Utilizando uma detecção distribuída de eventos complexos, redundâncias de detecção são eliminadas através do compartilhamento de resultados de detecção entre subscritores. Para expressões sobrepostas de subscrições compostas enviadas por clientes, a detecção é executada uma só vez, e subscritores perto uns dos outros podem reutilizar os resultados de detecções. A detecção distribuída também reduz o tráfego na rede.

Uma subscrição composta é encaminhada o mais longe possível através da rede antes que seja dividida, assim um crescimento elevado no número de subscrições na é evitado. Além disto, eventos compostos são detectados perto das fontes de dados na rede e apenas uma única notificação é enviada após uma correspondência, ao invés de um conjunto de publicações, reduzindo o número de publicações roteadas.



(a) (b)

Figura 6 - Exemplo de Árvore de Subscrição (B) e Árvore de Roteamento (a) para uma subscrição composta simples [9]

A Figura 6 mostra um exemplo de subscrição composta, com sua árvore de Subscrição (b) e sua Árvore de Roteamento (a). Na Árvore de Subscrição (b), os números representam os brokers destino de cada parte da subscrição correspondentes à Árvore de Roteamento. Na Árvore de Roteamento, os círculos representam brokers, CS é a subscrição composta. Uma tupla do tipo (an, n) representa um advertisement que corresponde a uma subscrição sn e o número do broker destino para uma subscrição sn, e Psn é um produtor de um evento que corresponde a uma subscrição sn. Neste exemplo, a primeira parte da subscrição composta (s1 AND s2) é correspondida no broker 4 e a subscrição composta completa é correspondida no broker 2.

4.3. DHEP (Distribuição Heterogênea)

Como já foi mencionado anteriormente, diferentes benefícios são obtidos através de arquiteturas distribuídas em cluster ou em rede. Se em uma arquitetura distribuída em cluster existe um alto acoplamento entre as máquinas CEP em uma rede local possibilitando uma baixa latência de comunicação e processamento, com uma arquitetura distribuída em rede, as máquinas de processamento podem ser estabelecidas próximo às fontes produtoras e consumidoras de dados reduzindo o tráfego de rede. Uma combinação destes benefícios talvez seria obtido em uma distribuição híbrida, em que diferentes clusters de máquinas fortemente acopladas em um mesmo domínio de rede existissem em diferentes localizações amplamente distribuídas. O Distributed Heterogeneous Event Processing (DHEP) é um framework que suporta a integração de vários sistemas CEP centralizados em um ambiente distribuído. Entretanto, o DHEP vai além desta questão, permitindo também a utilização de tecnologias heterogêneas de máquinas processadoras de eventos.

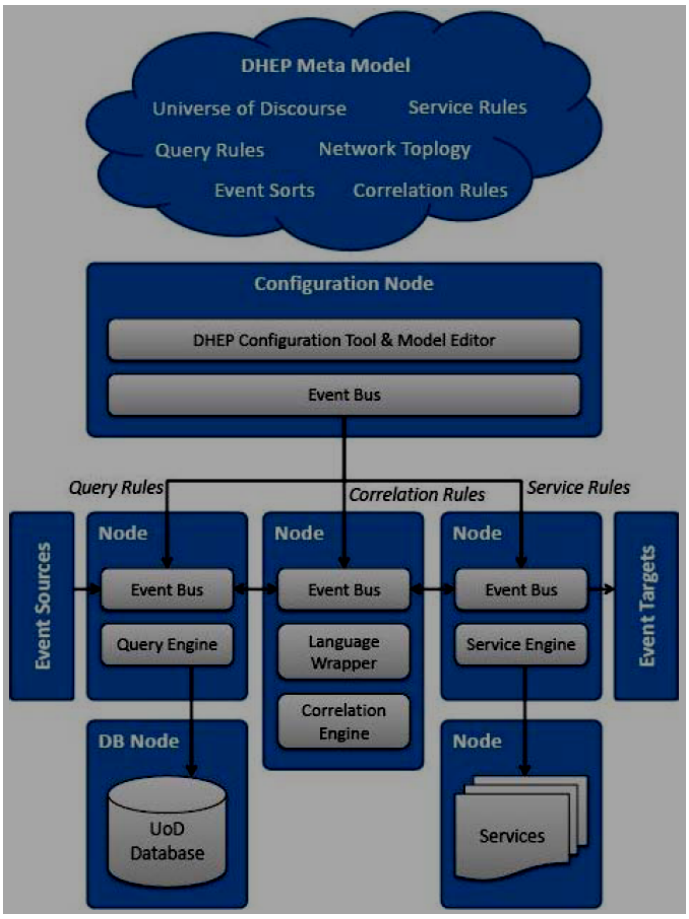


Figura 7 - Visão geral do sistema

Seu objetivo é interconectar máquinas heterogêneas CEP centralizadas existentes em uma rede. Para isto, estas máquinas centralizadas são acopladas em um framework que cuida de todas as características necessárias para a construção de um sistema de processamento distribuído, tal como o gerenciamento da comunicação entre os nós e implantação de regras distribuídas. Além disto, como máquinas heterogêneas não são capazes de interagir sem algum tipo de tradução de eventos, uma meta-linguagem é utilizada para definir todas as entidades do sistema. Esta linguagem permite a utilização de ontologias para projetar e gerenciar eventos, regras e informações de contexto dentro de um sistema distribuído. O DHEP provê também uma ferramenta de configuração para o projeto do sistema de processamento de eventos, com um editor gráfico, onde o usuário pode modelar e gerenciar o sistema, além de gerenciar a atribuição e implantação de regras em nós distribuídos. A Figura 7 mostra uma visão geral do sistema.

O sistema DHEP é baseado em uma meta linguagem para a definição de elementos do sistema. A linguagem permite a definição de: um modelo de contexto orientado a objetos, tipos de eventos que são transmitidos dentro do sistema, regras de processamento de eventos para correlação de padrões de eventos, acesso à informações externas de contexto, nós de processamento de eventos com suas respectivas máquinas CEP, e descrições de implantação de regras, com atribuição de regras à nós da rede e máquinas de processamento.

Ambiente de Execução. Cada nó na rede executa o ambiente de execução do sistema DHEP (DHEP Runtime Environment ou RE) que abstrai as funcionalidades realiza todas as tarefas necessárias para permitir um processamento distribuído de eventos no sistema.

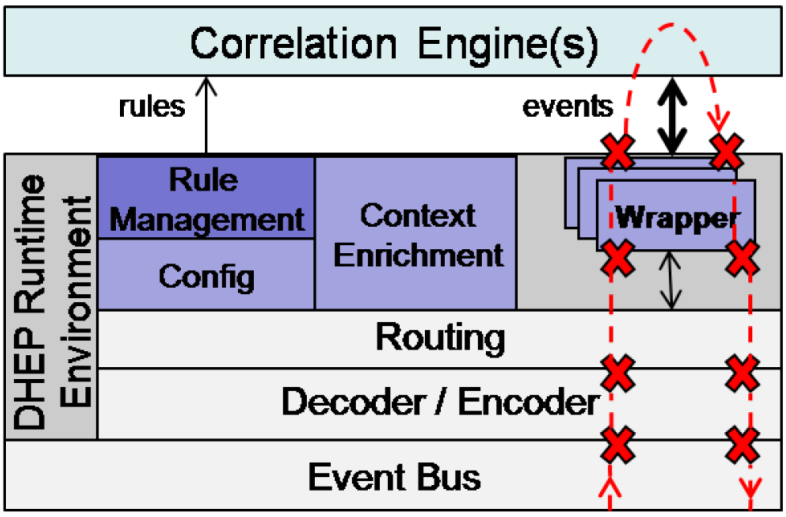


Figura 8 - Principais componentes do Runtime Environment [2]

Barramento de Eventos. O Barramento de Eventos (Event Bus) realiza a distribuição de eventos. A interface de comunicação do RE é projetada com módulos substituíveis de diferentes tecnologias de comunicação, permitindo a integração de múltiplos sistemas heterogêneos. Os módulos de comunicação já implementados incluem sockets, Java Message Service (JMS), e filas de mensagens.

Decoder/Encoder. O Decoder/Encoder realiza a tradução entre eventos que trafegam pela rede para representação semântica dos mesmos, consultando o meta-modelo, deserializando eventos que chegam no nó e encapsulando-os dentro de objetos que representam os eventos semanticamente e no sentido inverso serializando representações semânticas de eventos para eventos de saída.

Routing. O componente de roteamento recebe objetos de eventos de entrada do decoder e os encaminha de acordo com uma tabela de roteamento. A tabela contém informações sobre o próximo destinatário de cada tipo de evento na rede, mas também informações sobre as máquinas locais que devem processar o evento naquele nó. Então, o barramento de eventos distribui cada evento de entrada, enviando-o respectivamente para uma ou mais máquinas locais ou outros nós na rede.

Wrapper. O componente wrapper é responsável pela integração de diferentes máquinas CEP. Ele age primariamente como um adaptador entre o componente de roteamento e máquina de processamento no mesmo nó. Um wrapper é requerido para cada máquina CEP diferente, pois sua principal tarefa é fazer a tradução entre a meta-linguagem e a máquina CEP alvo. O conceito de wrapper é utilizado também para a configuração do sistema. Portanto um wrapper de configuração é implementado, que recebe todas as informações relevantes do sistema através de eventos de configuração que são distribuídos através do barramento de eventos. Os eventos de configuração podem conter, por exemplo, informações de roteamento e estabelecimento de regras. O wrapper de configuração cuida das configurações apropriadas para a tabela de roteamento e move regras para as máquinas corretas de processamento de eventos (de acordo com um algoritmo de propagação de regras).

Rule Management. O gerenciador de regras (Rule Management) é responsável pela distribuição, migração e estabelecimento de regras no sistema. Para isto ele utiliza eventos de configuração recebidos para gerenciar a implantação de regras. O Rule Management oferece interfaces para mover regras para outros nós, assim como para implantá-las em uma máquina local de processamento de eventos que está conectada através de um dos wrappers. Atualmente a geração de eventos de configuração se dá

pelo usuário através da ferramenta de configuração, mas mecanismos mais sofisticados estão sendo desenvolvidos para que o componente faça atribuições de forma autônoma e distribuída, lidando com restrições e critérios de otimização dos ambientes heterogêneos. Tradutores de regras são anexados ao Rule Management, pois as descrições de regras utilizando a meta-linguagem do DHEP não são entendidas pelas várias máquinas CEP diretamente.

Event Context Enrichment. Diferentemente de abordagens centralizadas, sistemas distribuídos de CEP ainda não permitem o acesso a informações externas. O sistema DHEP entretanto, oferece esta característica, com uma máquina de consultas que é capaz de enriquecer eventos com contexto recuperado de fontes externas, tais como um banco de dados. A configuração da máquina de consulta é feita através de regras de consultas que também são especificadas na linguagem DHEP. O transporte de contexto é garantido já que todos os objetos especificados podem ser serializados e deserializados com o encoder/decoder. Para reduzir o custo esperado produzido pelo acesso ao banco de dados, é implementado um mecanismo de cache configurável.

Configuration Tool. A ferramenta de configuração é o ponto em que o usuário modela, implanta e interage com o sistema. A ferramenta é um editor gráfico de modelagem, mas sua principal funcionalidade está nos mecanismos de implantação iniciados e coordenados pela ferramenta através do envio de eventos de configuração para os componentes Rule Management do framework.

5. Conclusão

Este trabalho apresentou uma visão geral dos aspectos relacionados à arquiteturas distribuídas para processamento de eventos complexos. Foram apresenta dos três trabalhos, cada um com uma categoria diferente de distribuição de arquitetura. O primeiro com uma distribuição de arquitetura em cluster, que possui nós fortemente acoplados em uma rede local de computadores para aumentar a capacidade de processamento. O segundo, com uma arquitetura distribuída em rede, que possui nós amplamente distribuídos através de uma rede WAN com o objetivo de estabelecer nós processadores de eventos o mais próximo possível dos produtores e consumidores de eventos, reduzindo assim o tráfego de rede. E o terceiro, uma abordagem heterogênea, que consiste de um framework para interligação de diferentes máquinas de processamento centralizadas, e que utiliza uma meta-linguagem e ontologias para definição dos elementos do sistema, e tradutores e wrappers para a interoperabilidade entre diferentes tecnologias.

Embora nos últimos anos diversos trabalhos tenham atacado o problema de CEP distribuído, a maioria das soluções comerciais distribuídas utilizadas comercialmente adotam uma distribuição em cluster, e poucas distribuições em rede apenas recentemente começaram a ser utilizadas. Isto pode ser devido ao fato de que CEP está fortemente acoplado aos processos de negócio e por isto necessita de acesso eficiente aos dados contextuais dos processos de negócio, que em geral executam dentro de centros de processamento de dados, e que em geral podem se beneficiar dos clusters de computadores presentes nestes centros.

Duas implementações CEP comerciais se destacam, o Esper [11], que é uma implementação Java possui uma distribuição em cluster, e o TIBCO Business Events [12] que possui uma distribuição em rede. Entretanto, a descrição destes sistemas não foi possível dentro do escopo deste trabalho pois suas documentações ou não estão bem consolidadas ao ponto de descrever suas estruturas de forma concisa, ou até mesmo (no caso do segundo sistema) não há documentação disponível sem um contrato comercial.

6. Referências Bibliográficas

1. O. Etzion and P. Niblett, *Event processing in action*, Manning Publications Co., 2010.
2. B. Schilling, et al., "Distributed heterogeneous event processing: Enhancing scalability and interoperability of cep in an industrial context," ACM, 2010, pp. 150-159.
3. P. EUGSTER, et al., "The Many Faces of Publish/Subscribe," *ACM Computing Surveys*, vol. 35, no. 2, 2003, pp. 114-131.
4. OMG, "Data-Distribution Service for Real-Time Systems (DDS)," *Book Data-Distribution Service for Real-Time Systems (DDS)*, Series Data-Distribution Service for Real-Time Systems (DDS), ed., Editor ed.^eds., 2006, pp.
5. G. Cugola and A. Margara, *Processing flows of information: From data stream to complex event processing*, Technical report, Politecnico di Milano, 2010. Submitted for Publication.
6. D.C. Luckham, *The power of events: an introduction to complex event processing in distributed enterprise systems*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2001.
7. N.P. Schultz-Møller, et al., "Distributed complex event processing with query rewriting," ACM, 2009, pp. 1-12.
8. P. Pietzuch, et al., "A framework for event composition in distributed systems," Springer, 2003, pp. 997-997.
9. E. Fidler, et al., "The PADRES Distributed Publish/Subscribe System," 2005.
10. A. Carzaniga, et al., "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, 2001, pp. 332-383.
11. Esper, "Esper," http://esper.codehaus.org/.
12. TIBCO, "TIBCO Business Events," http://www.tibco.com/products/business-optimization/complex-event-processing/businessevents/default.jsp.