リンク集約とパタンキャッシュを用いた複合イベント処理の高性能化

西村 直孝† 川島 英之‡

†筑波大学 システム情報工学研究科 コンピュータサイエンス専攻 〒305-0001 茨城県つくば市天王台 1-1-1
‡筑波大学 システム情報系 情報工学域 〒305-0001 茨城県つくば市天王台 1-1-1
E-mail: †m1_nisimura@kde.cs.tsukuba.ac.jp, ‡kawashima@cs.tsukuba.jp

あらまし 複合イベント処理はイベントストリームから価値あるパタンを抽出する重要な技術である．本稿では，複合イベント問合せを高速に実行する手法を提案する．要となるコンセプトは，部分的パタンキャッシングである．この手法は，パタンオカレンス(問合せの結果)生成時にたどったリンクをキャッシュとして保存しておくことで，次のパタンオカレンス生成時にはリンクをたどることなくパタンオカレンスを生成し，高速化を図る．

キーワード 複合イベント処理，AIS，データストリーム

1. はじめに

近年では，多くのストリーム源がイベントモニタリングのために利用されている．例として，ネットワークパケットストリームはマルウェア検出のために解析され，ビデオストリームは犯罪を検出するために解析されており，株価ストリームはアルゴリズムトレーディングのために解析されている．それぞれのセンシングデバイスは物理空間やサイバー空間を監視し，継続的にセンサデータを生成する．この種のデータはストリームデータと呼ばれる．ストリームデータを解析するために，異なる種類の問合せ実行基盤が提案されており，その中の 1 つに複合イベント処理基盤[1，2，3，5，7]がある．

本稿は，ストリームデータに対する複合イベント問合せのスループット改善を目的とする．複合イベント処理の手法は SASE[1，5] や Cayuga[6]，ZStream[2]が提案されている．特に著者らは，高頻度に到着する RFID のイベントストリーム処理を目的とした高性能複合イベント問合せ実行方法である SASE に着目する．SASE では，アプリケーションから投入された問合せは非決定性有限オートマトン(Nondeterministic Finite Automaton, NFA)に変換される．各々の NFA の状態は，入力されるイベントを適合した状態に格納するために AIS(Active Instance Stack)と呼ばれるバッファを持つ．新しいイベントが NFA に到着すると，そのイベントは一致する AIS に格納される．NFA の受理状態に到達するイベントが到着すると，それぞれの AIS 内のイベントは問合せの結果を生成するためにスキャンされる．本稿では，このような問合せを満たす結果を**パタンオカレンス**と呼ぶことにする．加えて，その生成過程はパタンオカレンス生成と呼ぶことにする．最後に，生成されたパタンオカレンスはアプリケーションに送られる．

本稿では，“PATTERN”，“FROM”，“WHERE”，“WITHIN”の 4 つの句で構成される上記のフォーマットにしたがって複合イベントの処理を論じる．

```
PATTERN       <event pattern>
FROM          <input stream>
WHERE         <filtering conditions>
WITHIN        <window>
```

PATTERN 句ではイベントの並びを表現する．FROM 句では入力されるストリームを指定する．SASE[1，5]では複数の情報源から到着する全てのストリームは単一の入力ストリームに統合されているが，本稿では単一の情報源のみを想定する．ゆえに，本稿は[8]で述べられているようなグルーピング演算子を考慮しない．WHERE 句では，PATTERN 句で述べたイベントの選択条件を記述する．WITHIN 句ではウィンドウサイズを定める．

上記の問合せをよりわかりやすくするために，実行例として以下の問合せを用いる．

```
PATTERN  SEQ (A, B)
FROM     PacketStream
WHERE    A.dstport = 80 AND B.dstport != 80
WITHIN   500 events
```

“SEQ(A，B)”はイベント A の後にイベント B が発生することを意味する．イベント A とイベント B は単一の情報源から発生することを想定する．“FROM PacketStream”は問合せの情報源を表す．IP パケットのストリームはネットワークから得られるものとし，各々の IP パケットからヘッダ情報が抽出されるものとする．その後，抽出されたヘッダ情報は“PacketStream”と名付けられたイベントストリームのスキームに適合するために成型される．ゆえに，情報源から到着する全てのイベントは IP パケットのヘッダ情報に関連する同じ属性を持つ．WHERE 句ではパケットの宛先ポートを意味する“dstport”という属性が指定されている．それに基づいて，上記の表現“A.dstport = 80 AND B.dstport != 80”は 1 番目のパケ

ットの宛先ポートが 80 番であり,かつその直後に発生するパケットの宛先ポートが 80 番ではないことを意味する．“WITHIN 500 events”はウィンドウサイズを意味し，[8]で用いられている CQL 表現では“[ROWS 500]”に相当する．

実際のアプリケーションでは，センシングデバイスは高い入力レートでストリームデータを配信する．ゆえに，複合イベント問合せ実行機構は入力レートよりも高速にストリームデータを処理することが要求される．この要求が満たされない場合，複合イベント問合せ実行機構は入力ストリームの一部を取りこぼし，問合せ結果の品質を劣化させる．

SASE は顕著な問題を 2 つ抱えている．それは大きなウィンドウサイズと中間結果の大きさである．この問題に注目するにあたり，SASE はアルファベット文字でイベントを表現することで単純化している．本稿でもこの方針を適用し，スループットを改善することに着目する．当然ながら著者らが対象とするアプリケーションも SASE と類似する．

高スループットな問合せ処理を達成するために，著者らは SASE を拡張することで[24]において，FLA(Forward Link Aggregation)という手法を提案した．しかし，入力されるイベントの並びによっては，FLA ではスループットが改善するとは限らない．これは後述するリンク集約と概念を用いているからであり，リンク集約ができない場合には同じリンクを何度もたどる問題が発生する．FLA の問題については 4.1 節で詳述する．そこで，著者らはキャッシュ構造を利用することによって，この問題を解決する PRC(Partial Result Caching)という手法を提案する．この手法は，パタンオカレンス生成時にたどったリンクをキャッシュとして保存しておくことで，次のパタンオカレンス生成時にはリンクをたどることなくパタンオカレンスを生成し，スループットの改善を図る．

本稿は以下の章によって構成される．2 節では関連研究を述べ，3 節では以前に著者らが提案した手法について述べる．4 節で部分的パタンキャッシングを利用した高スループットな複合イベント問合せ実行手法を提案する．最後に 5 節で本稿の結論と今後の課題を述べる．

2. 関連研究

ストリームデータ処理の研究は多岐にわたる．ストリームデータ処理の初期の概念は，ウィンドウのセマンティックスと連続的問合せをリレーショナルデータモデルに統合することで設立され[4]，STREAM[8], Aurora[11], Borealis[12], NiagaraCQ[10], TelegraphCQ[9]などといったいくつかの研究試作システムが開発された．このようなシステムの全てはリレーショナルデータモデルに基づいており，特定のアプリケーションに適合するような工夫はされていない．

一方で，他のいくつかのシステムは特定のアプリケーションを目的として設計されている．例えば，GigaScope[14]はパケット検査のために AT&T ベル研究所で開発され，XStream[13]はモルモットを含む野生動物を監視するために提案されており，リレーショナル演算子だけでなく信号処理の演算子も提供している．

研究試作システムから大きく発展して，現在ではすでに uCosminexus Stream Data Platform[19], Microsoft SQL Server StreamInsight[15, 18]，Oracle CEP[16]，EsperTech[17], System S[20]などの商用ストリームエンジンが利用可能である．

リレーショナルデータモデルでは，全てのタプルは独立である．言い換えるならば，リレーショナルデータモデルはタプルの非順序集合に基づいている．それに対し，ストリームデータのタプルは連続的に生成されるので固有の順序を持っており，それらは時間によって順序付けされる．

順序問題を解決するために，複合イベント処理と呼ばれる新しい種類のストリーム処理基盤が提案された．様々な複合イベント処理のプロジェクトが立ち上がり，初期の研究は複合イベント問合せの問合せ処理を高速化することを試みている．その代表的なプロジェクトとして SASE[1，5]，Cayuga[6]，ZStream[2]がある．その後複合イベント処理の研究分野は広がっていき，確率付きのストリームデータを処理する[22]，元となるモデルに入れ子構造の問合せモデルを追加した[23]などが提案されている．

[23]の 2 節でも述べられているように,SASE は 2006 年に提案されてはいるが最新の複合イベント処理の手法の 1 つである．他の最新の手法としては，NFA の代わりに木構造に基づいた問合せ処理手法である ZStream[2]があるが，本稿は NFA に基づいた手法に着目しているため，著者らは提案手法と SASE を比較することを目的とする．

3. FLA (Forward Link Aggregation)

本節では，著者らが[24]で提案した複合イベント処理手法である FLA(Forward Link Aggregation)の処理手順について述べる．FLA はリンクを集約することで SASE[1]よりも高いスループットを実現するが, 1 節で述べたような問題を抱えている．本稿で述べる提案手法はその問題を解決するために FLA を拡張しているので，4 節の前に FLA について述べる．

3.1. 準備

以降の説明では，ウィンドウサイズを 9，問合せパタンを“SEQ(A，B，D)”，以下の a1 から d9 までのイベントが入力されるものとする．

a1, c2, b3, a4, d5, b6, d7, a8, d9

入力されるイベントのうち，1 文字目がイベントタイプを表しており，2 文字目は到着時刻を表す．例えば，“c2”というイベントはイベントタイプが“C”で，そのタイムスタンプは“2”である．

本稿では[1, 5]で述べられている“skip till next match”の考え方を採用する．例えば，上記のイベントから解の 1 つとして，<a1，b3，d5>が得られる．このパタンオカレンスにおいて，“c2”は読み飛ばされる．以降では，この種のイベントを“スキップイベント”と呼び，他のイベントを“ターゲットイベント”と呼ぶ．

この節の残りは，上記の状況を例に挙げて FLA の手続きを述べる．

3.2. FLA の手続き

3.2 節では，FLA において入力されたイベントがどのように処理され，いつパタンオカレンスが生成されるか，3.1 節で述べた問合せを例示して 4 ステップに分けて述べる．

STEP1：例として挙げられた問合せ“SEQ(A，B，D)”は図 1 のような NFA に変換される．図 1 において，“*”は[1]で述べられているように自己ループ遷移を表す．自己ループ遷移は NFA にスキップイベントが到着したときに発生する．もし到着したイベントがスキップイベントではない場合，そのイベントは NFA の状態を次の状態に遷移させる．

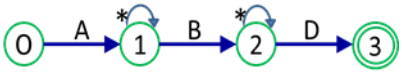

図 1: 指定されたパタンを受理する NFA

STEP2：NFA の各々の状態について，AIS(Active Instance Stack)が作成される．それぞれの AIS はその状態に対応したイベントを格納する．NFA にターゲットイベントが到着した場合，そのターゲットイベントは対応する AIS に格納される．FLA では，格納されたターゲットイベントに対してリンクを与える．そのリンクは新しいターゲットイベントと 1 つ先の AIS に格納されたイベントの中で最も小さいタイムスタンプを持つイベントとの間に張られる．本稿ではリンク先のイベントで最もタイムスタンプの小さいものを，RIS(the most Recent Instance in the Subsequence stack)と表記する．FLA では指定されたパタンの最初のイベントが発生したとき，ウィンドウサイズの大きさだけ AIS にイベントを格納する．図 2 は“a1”から“d9”までのイベントが到着したときの状況を示している．図 2 において，“a1”は“b3”へのリンクを持っており，“b3”は“d5”へのリンクを持っている．

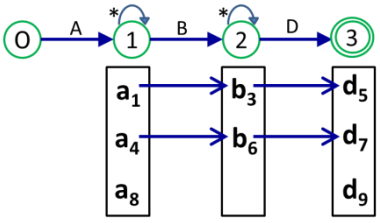

図 2: RIS が与えられて AIS 内に格納されたイベント

STEP3：FLA は同じリンク先を持つイベントを集約する．集約されたイベントは 1 つのクラスタとして扱われる．図 2 の状況では，集約されるイベントはない．

STEP4：リンクをたどることでパタンオカレンスを生成する．図 2 において，リンク先となるイベントは，RIS となるイベントとその下に位置するイベント全てである．そのため，“a1”から“b6”へ向かう矢印はなく“b6”から“d9”へ向かう矢印はない．このステップでは，<a1，b3，d5>，<a1，b3，d7>，<a1，b6，d7>，<a4，b6，d7>，<a1，b3，d9>，<a1，b6，d9>，<a4，b6，d9>の 7 つのパタンオカレンスが生成される．

STEP5：FLA は NFA の受理状態に対応する AIS を初期化する．この例では，“d5”と“d7”と“d9”が削除される．図 3 はこれらのイベントが削除されたあとの状況を表す．

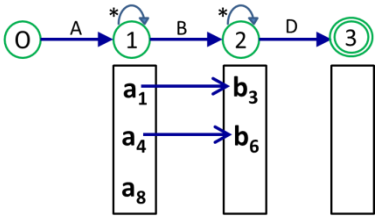

図 3: 受理状態に対応する AIS の初期化

STEP6：“a1”から始まるパタンオカレンスは全て生成したので，“a1”を削除する．次は“a4”から始まるパタンオカレンスを生成する．“b3”が RIS となるイベントはないため，“b3”も削除する．図 4 はこれらのイベントが削除されたあとの状況を表す．

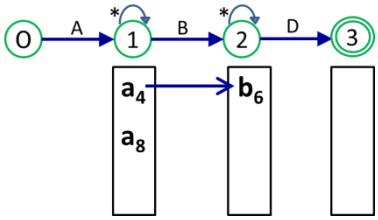

図 4: “a1” と “b3” の削除

FLA は STEP6 が終わったあとは STEP2 に戻る．“a4” が全ての AIS の中で最もタイムスタンプの小さいイベントなので，ウィンドウサイズを考慮して次は “d12” までのイベントを AIS に格納する．図 5 は AIS が“d12” までのイベントを格納した状況を表す．

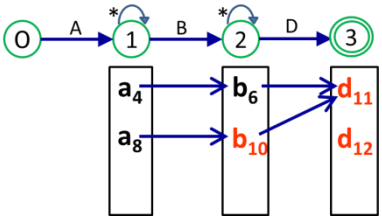

図 5:“d12” までのイベントの格納

4. 提案手法

4.1. FLA の問題点

3 節で述べた FLA は高スループット化の実現を阻害する問題が 2 つある．

1 つ目は，1 回目のパタンオカレンス生成と 2 回目のパタンオカレンス生成の結果には最終イベント以外同じ部分的パタンオカレンスが含まれていることである．例えば，3.2 節の処理の続きとして “d11” が発生したことを考えると，2 回目の STEP3 では<a4，b6，d11>というパタンオカレンスが生成されるが，<a4，b6，d7>と<a4, b6, d11>では，部分的パタンオカレンス<a4, b6>はどちらにも含まれている．しかし，これらのパタンオカレンスは異なるステップのパタンオカレンス生成の結果なので，同じリンク(a4→b6)を再びたどらなければこの結果は生成されない．

2 つ目の問題は，FLA はリンクを集約することによって高スループット化を実現しているので，リンクが集約できないようなイベント列が入力された場合は性能が発揮されず，同じリンクを複数回たどるような上述の問題が発生する．言い換えるならば，FLA のスループットは入力イベント列のイベントの分布に依存する．

4.2. 部分的パタンオカレンスのキャッシュ構築

4.1 節の問題を解決するために，著者らは PRC(Partial Result Caching)という手法を提案する．この手法は時系列順に部分的パタンオカレンスのキャッシュ構造を構築し，FLA のようにリンクトラバーサルだけでパタンオカレンスを生成するのではなく，このキャッシュ構造内の部分的パタンオカレンスも利用してパタンオカレンス生成を行うことで高スループット化を図る．以降では，キャッシュ構造内に格納される部分的パタンオカレンスを**パタンキャッシュ**とする．PRC のパタンキャッシュは，1 度リンクトラバーサルでたどったものであり，以降のパタンオカレンス生成ではこのパタンキャッシュを利用することで，FLA では解決できないような同じリンクを複数回たどる問題を解決する．

4.3. PRC の処理手順

本節では，問合せを “SEQ(A，B，D)”，ウィンドウサイズを 9，以下のイベント列が入力されるという条件のもと，PRC の処理手続きについて述べる．

a1, c2, b3, a4, d5, b6, d7, a8, d9, b10, d11, d12, b13

STEP1：指定されたパタンの最初のイベントが発生したとき，ウィンドウサイズの大きさだけ AIS にイベントを格納する．ターゲットイベントが到着した場合には PRC は対応する AIS にそのイベントを格納し，そのイベントが他のイベントの RIS になる場合には RIS として設定する．RIS となるイベントはリンク元のイベントよりも大きいタイムスタンプを持ち，かつ 1 つ先の AIS 内で最も小さいタイムスタンプを持つ．“a1” から “d9” までのイベントが NFA に到着した状況は図 2 と同じである．

STEP2：パタンキャッシュを用いてパタンオカレンスを生成する．この時点では，パタンキャッシュが生成されていないので生成されるパタンオカレンスもない．

STEP3：NFA の初期状態に対する AIS について，リンク先が同じイベントを集約してクラスタ化する．図 2 の状況では，リンク先が同じイベントはないので，集約されるイベントはない．

STEP4-1：リンクをたどることでパタンオカレンスを生成する．リンク先となるイベントは RIS と，RIS と同じスタック内に存在する RIS よりもタイムスタンプの大きいイベントである．このステップで生成されるパタンオカレンスは 3 節の STEP4 と同じである．

STEP4-2：パタンオカレンス生成中にたどってできる

部分的パタンオカレンスは，リンク先が NFA の受理状態に対応するイベント以外であれば，それをキャッシュ構造に格納する．図 6 にリンクをたどることでパタンキャッシュが追加されたあとのキャッシュ構造を示す．図 6 のように，キャッシュ構造は A イベントの時系列順になるように並べる．パタンオカレンスが生成し終わるまで STEP4-1 と STEP4-2 が繰り返し実行される．

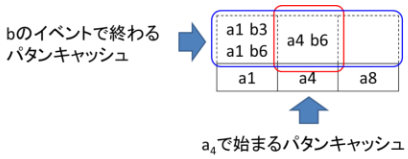

図 6: リンクをたどることでキャッシュ構造に追加されるパタンキャッシュ

STEP5 : NFA の初期状態以外に対応する AIS を初期化する．これに伴い，NFA の初期状態に対応する AIS 内のイベントの RIS も初期化する．図 2 の状況から“b3”，“b6”，“d5”，“d7”，“d9”が削除される．また，“a1”と“a4”の RIS も初期化される．このステップ実行後の状況を図 7 に示す．

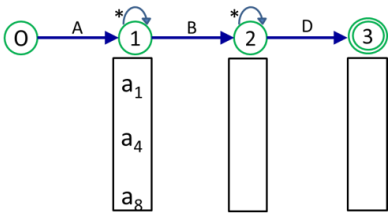

図 7: NFA の初期状態以外に対応する AIS の初期化

STEP6 : NFA の初期状態に対応する AIS の最初のイベント(“a1”)を削除する．同様に，キャッシュ構造に格納された“a1”から始まるパタンキャッシュ<a1, b3>, <a1, b6>も削除する．“a1”を削除した後の AIS を図 8 に示し，その時点でのキャッシュ構造を図 9 に示す．

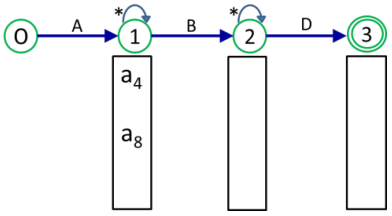

図 8: “a1”削除後の AIS

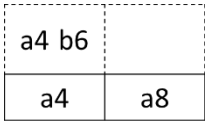

図 9: “a1”削除後のキャッシュ構造

STEP6 実行後は STEP1 に戻る．ただし，“a4”が全ての AIS の中で最もタイムスタンプの小さいイベントなので，ウィンドウサイズを考慮して次は“d12”までのイベントを AIS に格納する．図 10 に“d12”までのイベントを格納した AIS を示す．異なるステップとして区別するために，以降のステップは STEP2’のようにプライム符号を付けて表記する．

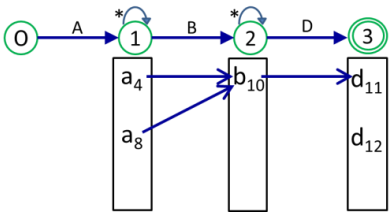

図 10: “d12”までのイベントの格納

STEP2’ : パタンキャッシュを用いてパタンオカレンスを生成する．3 番目以降の AIS 内のイベントのタイムスタンプが，パタンキャッシュの末尾のイベントよりも大きければ，それらを結合することで新しいパタンオカレンスを生成する．図 10 の“d11”と“d12”のタイムスタンプはパタンキャッシュの末尾のイベント“b6”よりも大きいため，このステップでは 2 つのパタンオカレンス<a4, b6, d11 >, <a4, b6, d12 >が生成される．これらのパタンオカレンスはパタンキャッシュと AIS 内のイベントを組み合わせるだけで生成できるため，リンクトラバーサルによって 1 つずつイベントをたどっていくよりも高速にパタンオカレンスを生成できる．

STEP3’ : NFA の初期状態に対する AIS について，リンク先が同じイベントを集約してクラスタ化する．図 10 では“a4”の RIS と“a8”の RIS はどちらも同じなので，これらのリンクを集約して 1 つのクラスタとして扱う．“a4”と“a8”をクラスタ化したあとの状況を図 11 に示す．

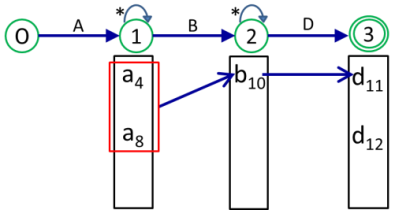

図 11: “a4”と“a8”のリンク集約

STEP4-1’：リンクをたどることでパタンオカレンスを生成する．このステップでは，4 つのパタンオカレンス< a8，b10，d11 >，< a8，b10，d12 >，< a4，b10，d11 >，< a4，b10，d12 >が生成される．

STEP4-2’：パタンオカレンス生成中にたどってできる部分的パタンオカレンスは，リンクの宛先が NFA の受理状態に対応するイベント以外であれば，それをキャッシュ構造に格納する．リンクをたどることで追加できるパタンキャッシュが，追加されたあとのキャッシュ構造を図 12 に示す．パタンオカレンスが生成し終わるまで STEP4-1’と STEP4-2’が繰り返し実行される．

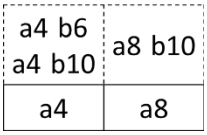

図 12: 新たなパタンキャッシュの追加

STEP5’：NFA の初期状態以外に対応する AIS を初期化する．“b10”と“d11”と“d12”が削除され，“a4”と“a8”の RIS は初期化される．

STEP6’：NFA の初期状態に対応する AIS の最初のイベント(“a4”)を削除する．同様に，キャッシュ構造に格納された“a4”から始まるパタンキャッシュ<a4, b6>，<a4，b10>も削除する．

PRC は STEP1 から STEP6 までを繰り返すことでパタンオカレンスを生成する．FLA を用いて STEP2’以降を実行した場合，11 回のリンクトラバーサルを必要とする．しかし，PRC ではリンクトラバーサルの回数は 5 回であり，FLA よりも少ない．PRC はキャッシュ構造も利用してパタンオカレンスを生成しているので，生成されるパタンオカレンス数は同じである．

5. まとめ

著者らは，複合イベント処理においてスループットが極めて重要な指標であることから，本稿で高スループット化を実現する複合イベント処理手法を提案した．今後の課題は，SEQ オペレータ以外も使用できるような問合せの表現能力の拡張，実際のデータセットを用いた評価などがあげられる．

参 考 文 献

[1] Eugene Wu, Yanlei Diao, Shariq Rizvi, “High-Performance Complex Event Processing over Streams”, Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, Illinois, USA, pp.407-418, June. 2006.

[2] Yuan Mei, Samuel Madden, “ZStream: A Cost-based Query Processor for Adaptively Detecting Composite Event”, Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, Rhode Island, USA, pp.193-206, June-July. 2009.

[3] Xin Li, Hideyuki Kawashima, Hiroyuki Kitagawa, “Complex Event Processing over Uncertain Data Streams”, The 70th National Convention of IPSJ, Ibaraki, Japan, March. 2010.

[4] H. Garcia-Molina, J.D. Ullman, and J. Widom, “Database Systems - The Complete Book Second edition”, Prentice Hall Press Upper Saddle River, USA, June. 2008.

[5] Yanlei Diao, Neil Immerman, Daniel Gyllstorm, “SASE+: An Agile Language for Kleene Closure over Event Streams”, UMass Technical report 07-03, University of Massachusetts, Amherst, March. 2007.

[6] Alan Demers et al, “Cayuga: A General Purpose Event Monitoring System”, Proceedings of Conference on Innovative Data Systems Research 2007, Asilomar, CA, pp.412-422, January. 2007.

[7] Zhitao Shen, Hideyuki Kawashima and Hiroyuki Kitagawa, “Efficient Probabilistic Event Stream Processing with Lineage and Kleene-plus”, International Journal of Communication Networks and Distributed Systems, Vol. 2, No.4 pp.355-374, June. 2009.

[8] Arasu, A., Babu, S., and Widom, J. 2004. “The CQL Continuous Query Language: Semantic Foundations and Query Execution”, The VLDB Journal, vol.15, no.2, pp.121-142, June. 2006.

[9] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S. R., Reiss, F., and Shah, M. A. 2003. “TelegraphCQ: Continuous Dataflow Processing”, Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, CA, pp.668-668, June. 2003.

[10] Chen, J., DeWitt, D. J., Tian, F., and Wang, Y. “NiagaraCQ: a Scalable Continuous Query System for Internet Databases”, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, TX, pp.379-390, June. 2000.

[11] Balakrishnan, H., Balazinska, M., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Galvez, E., Salz, J., Stonebraker, M., Tatbul, N., Tibbetts, R., and Zdonik, S. 2004. “Retrospective on Aurora”. The VLDB Journal, vol.13, no.4, pp.370-383, December. 2004.

[12] Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag S. Maskey, Alexander Rasin, Esther Ryvkina, Nesime Tatbul,

Ying Xing, and Stan Zdonik, “The Design of the Borealis Stream Processing Engine”, Proceedings of Conference on Innovative Data Systems Research 2005, Asilomar, CA, pp.277-289, January. 2005.

[13] Girod, L., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H., and Madden, S. “XStream: a Signal-Oriented Data Stream Management System”, Proceedings of 24th IEEE International Conference on Data Engineering, Cancun, Mexico, pp.1180-1189, April. 2008.

[14] Cranor, C., Johnson, T., Spataschek, O., and Shkapenyuk, V. “Gigascope: a Stream Database for Network Applications”, Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, CA, pp.647-651, June. 2003.

[15] Microsoft, “Microsoft SQL Server StreamInSight”, http://www.microsoft.com/sqlserver/2008/en/us/r2-complex-event.aspx, accessed August. 2013.

[16] Oracle, “Oracle CEP”, http://www.oracle.com/us/technologies/soa/service-oriented-architecture-066455.html, accessed August. 2013.

[17] EsperTech Inc., “EsperTech”, http://www.espertech.com/, accessed August. 2013.

[18] Chandramouli, B., Goldstein, J., and Maier, D. “On-the-fly Progress Detection in Iterative Stream Queries”, Proceedings of VLDB Endowment, vol.2, no.1, pp.241-252, August. 2009.

[19] Hitachi,Ltd., “Real-Time Log Analysis Using Hitachi uCosminexus Stream Data Platform”, http://www.hitachi.com/products/it/software/prod/cosminexus/index.html, accessed August. 2013.

[20] Gedik, B., Andrade, H., Wu, K., Yu, P. S., and Doo, M. 2008. “SPADE: The System S Declarative Stream Processing Engine”, Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Vancouver, BC, pp.1123-1134, June. 2008.

[21] Barzan Mozafari, Kai Zeng, and Carlo Zaniolo, “High-performance Complex Event Processing over XML streams”, Proceedings of 2012 ACM SIGMOD International Conference on Management of Data, Arizona, USA, pp.253-264, May. 2012.

[22] Thanh T. L. Tran, Liping Peng, Yanlei Diao, Andrew McGregor, Anna Liu, “CLARO: Modeling and Processing Uncertain Data Streams”, The VLDB Journal, vol. 21, no. 5, pp. 651-676, October. 2012.

[23] Medhabi Ray , Elke A. Rundensteiner , Mo Liu , Chetan Gupta , Song Wang , Ismail Ari, “High-performance Complex Event Processing using Continuous Sliding Views”, Proceedings of 16th International Conference on Extending Database Technology, Genoa, Italy, pp. 525-536. March. 2013.

[24] Naotaka Nishimura, Hideyuki Kawashima, “Accelerating CEP with Link Aggregation and Bulk Evaluation”, IPSJ Transaction on Database.(Conditional Acceptance)