5M-2

ストリーム処理エンジンと主記憶 DBMS の統合利用方式

小山田昌史† 川島英之‡ 北川博之‡

†筑波大学情報学群情報科学類 ‡筑波大学大学院システム情報工学研究科

1 はじめに

アクセス履歴，監視カメラからの映像，Twitter などの継続的に発信され続ける情報が増加してきた．こうした情報は**ストリームデータ**と呼ばれ，**ストリーム処理エンジン (SPE)** によりリアルタイムに処理されることが一般的となっている．

ストリーム処理の一つに**イベント検出**がある．これは，ストリームデータを監視し，特定の条件に当てはまるデータをイベントとして検出する処理である．検出されたイベントは関連する詳細情報と結合されて用いられることが多い．それゆえ，詳細情報の管理と結合処理方法が問題になる．

詳細情報の格納先に**データベース管理システム (DBMS)** を用いる場合，SPE 内に DBMS を組み込む方式と外部 DBMS を用いる方式の二つが存在する．前者はストリーム処理用に最適化可能であるため性能面で有利であるが導入コストが高い．一方，後者は既存の DBMS をそのまま利用可能なため導入コストは低いが，高性能を引き出すためには工夫が必要となる．

本研究では後者の外部 DBMS を利用する方式において生じる**連続的トランザクション呼び出し**を効率的に行う方法を提案し，実験によりその性能を検証する．また，外部 DBMS として**主記憶** *DBMS* を用いることによりシステム全体としての性能向上を試みる．

2 関連研究

リレーショナルデータとストリームデータとの結合に関して，SPE 内部に DBMS を組み込む方式については，ディスクアクセスを減らし効率的に結合を行う手法がいくつか考案されている [1, 2]．また，外部 DBMS を利用する方式については Botan ら [3] による試みがあげられる．彼らは SPE や DBMS を統合して扱うという枠組みの中で，ストリームデータとリレーショナルデータの両方を DBMS のテーブルに格納することにより，それらの結合処理を実現した．しかしながら，この方式ではタプルが到着するたびに DBMS への書き込み処理が生じるため，二次記憶 DBMS を格納先に用いる場合に十分な性能が発揮されないことがある．

3 連続的トランザクション呼び出し

詳細情報の格納先として外部 DBMS を使用する場合，ストリームデータとして到着したタプルは外部 DBMS の中に格納されている関連情報と結合されたのち再び出力される．このとき，タプルが到着するたびに外部 DBMS に対するトランザクション呼び出しが行われることとなる．そこで連続的トランザクション呼び出しの効率化が課題となる．これに対して本研究では三つの呼び出し方式を提案する．

3.1 同期呼び出し

同期呼び出しは，システムが外部 DBMS にトランザクションを呼び出した後，その結果が返るまで処理をブロックする方式である．処理結果が返るとそれを出力し，次のタプルの処理に移る．この方式では到着したタプルの順番とシステムから出力されるタプルの順番が一致することが保証される．ただし，通信時に処理がブロックされるために連続的な呼び出しを行う場合は非効率的となることがある．

3.2 非同期呼び出し

非同期呼び出しにおいては，システムは外部 DBMS に対してトランザクションを呼び出した後，処理のブロックはせずに次のタプルの処理へと移る．処理結果は別スレッドにより待ち受けておき，結果が返ってきたらそれを出力する．このとき，DBMS 側からは複数のトランザクションが次々と到着するように見え，それらのトランザクションの実行にかかる時間には差があるために，到着した順番とは異なる順番で処理結果を返すこともある．このため，この方式においては到着したタプルの順番とシステムから出力されるタプルの順番は一致しないことがある．一方，処理はブロックされないため，この方式は同期呼び出しと比べて効率的になりうる．

3.3 出力の順序を保証する非同期呼び出し

この方式では到着するタプルへ一意で連続な**タプル識別子**を付与する．そして非同期呼び出しを行うが，返ってきた処理結果はすぐには出力せずにキューへ格納する．このとき，処理結果はキューの中身がタプル番号の昇順に並ぶような位置へ追加される．システムは次に出

Integration of SPE and In-memory DBMS

Masafumi Oyamada†(masa@kde.cs.tsukuba.ac.jp),
Hideyuki Kawashima‡(kawasima@cs.tsukuba.ac.jp),
Hiroyuki Kitagawa‡(kitagawa@cs.tsukuba.ac.jp)
†College of Information Sciences, University of Tsukuba
‡Graduate School of Systems and Information Engineering, University of Tsukuba

力したいタプルの番号を保持しておき，タプルがキューに追加されるたびにキューの先頭にあるタプルを確認し，そのタプルが持つタプル番号と次に出力したいタプル番号が一致する間，タプルの出力と次に出力したいタプル番号のインクリメント操作を繰り返す．これにより，システムから出力されるタプルの順番が到着したタプルの順番と一致することが保証される．

4 分析モデル

前節で，同期呼び出しに対し非同期呼び出しが効率的となる場合があると述べた．ここでは，いかなる条件下でどの程度の差が生まれるかを検証するために，それぞれのモデル化を行なう．

外部 DBMS においてトランザクション一つ辺りの実行に要する時間を t，システムと外部 DBMS 間の通信におけるラウンドトリップタイムを d，システムに対するタプルの到着間隔を i とする．このとき，同期処理と非同期処理において N タプルの処理に要する時間 T_{sync} と T_{async} は以下のようになる．

$$T_{sync} = \begin{cases} N(t+d) & (i \leq t+d) \\ (N-1)i + (t+d) & (i > t+d) \end{cases}$$

$$T_{async} = (N-1)i + (t+d)$$

このモデルに従うと同期呼び出しと非同期呼び出しの実行時間差は $i < t+d$ のとき生じると言え，その差は以下のように表せる．

$$T_{sync} - T_{async} = (N-1)(t+d-i)$$

ここから，タプルの到着間隔がトランザクションの実行に要する時間よりも短いほど，非同期呼び出しが同期呼び出しに比べて良い性能を示すことが分かる．また，タプルの到着間隔がトランザクションの実行に要する時間よりも長い場合は，非同期呼び出しと同期呼び出しの間に差は生じないということも言える．

5 評価実験

各呼び出し方式の性能差を検証するため，100 タプルからなるデータを一定レートで 1 タプルずつシステムに送信し，システムが外部 DBMS 中の詳細情報を取得しタプルとの結合処理を終えるまでの時間を計測した．外部 DBMS としては分散主記憶 RDBMS である VoltDB[4] を用い，クラスタを構成するノードは 3 つ，各ノードのスレッド数は 3 に設定した．システムは結合クエリを受け取ると，それを選択クエリに変換して VoltDB へと渡す．このとき選択クエリ中の ? には `stream.id` の値が入れられる．これにより取得した詳細情報をシステム側でタプルと共に出力することにより，システム全体としての結合処理を実現する．

```
SELECT *
FROM   stream, info
WHERE  stream.id = info.id
```
システムへのクエリ

```
SELECT *
FROM   info
WHERE  info.id = ?
```
VoltDB へのクエリ

ストリームデータの到着間隔を 0 msec から 50 msec まで変化させて結合を行なった時の処理結果を図 1 に示す．横軸は到着間隔を，縦軸は 100 タプルの結合に要した時間を表している．ここから，ストリームの到着間隔が短い，すなわち到着レートが高いときに非同期呼び出しが同期呼び出しよりも効率的となっているということが分かる．これは前節で示したモデルにより得られた知見と一致する．また，非同期呼び出しにおいて出力の順序を保証する方式が，単純な非同期呼び出しと比べ遜色無い性能を示していることも観察される．

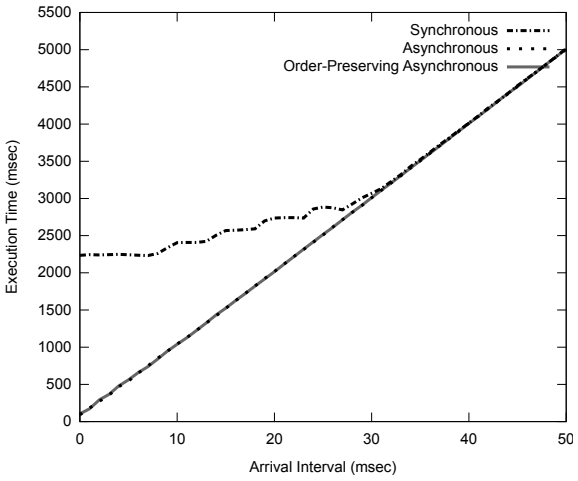

図 1　100 タプルの連続的な結合に要した時間

6 まとめ

本研究ではストリームデータとその詳細情報を結合する処理において，外部 DBMS を用いる場合に生じる連続的なトランザクション呼び出しを効率的に行う方式を検討した．ストリームの到着レートが高いときに非同期呼び出しが効率的となることをモデル化と実験により示し，出力結果の順番が保証されないという非同期呼び出しの欠点を補う手法を提案した．

謝辞

本研究の一部は科学研究費補助金 (# 21013004，# 22700090) による．

参考文献

[1] N. Polyzotis *et al.* Supporting Streaming Updates in an Active Data Warehouse. *ICDE*, 2007
[2] A. Chakraborty *et al.* A Partition-based Approach to Support Streaming Updates over Persistent Data in an Active Data Warehouse. *IPDPS*, 2009
[3] I. Botan *et al.* Design and Implementation of the MaxStream Federated Stream Processing Architecture. *Technical Report TR-632*, 2009
[4] http://voltdb.com/