情報処理学会論文誌　データベース　Vol. 2　No. 3　1–17 (Sep. 2009)

対象情報源を動的に選択可能なストリーム処理の実装と評価

大喜恒甫[†1]　渡辺陽介[†2]
北川博之[†1,†3]　川島英之[†1,†3]

近年，ネットワークカメラやセンサなどの多様な情報源から連続して配信されるストリームデータが増加しており，これらに対する高度利用要求が高まっている．そこで，利用者がこれらの要求を容易に実現するための基盤システムであるストリーム処理エンジンの研究・開発がさかんに行われてきた．ストリーム処理エンジンでは，利用者から問合せが登録されると，その問合せを連続して実行する．膨大なストリームデータが配信されてくる環境下において，すべてのストリームデータを処理するのではなく，状況に応じて処理対象の情報源を動的に選択してほしいという要求は多く存在する．しかし，従来のストリーム処理エンジンを用いたストリーム処理は，処理対象とする情報源を問合せ記述中にあらかじめ明記しておく必要があったため，状況に応じて動的に対象情報源を切り替えるような処理要求は実現できなかった．そこで，本研究ではストリーム処理における情報源の動的選択に関する研究を行った．本研究は以下の 2 つの特徴を持つ．(1) 単独のストリーム処理エンジンを利用した情報源の動的選択．我々が研究・開発中のストリーム処理エンジン StreamSpinner に情報源の動的選択のための機能を追加した．(2) 複数のストリーム処理エンジンを利用した情報源の動的選択．本研究では，処理対象の情報源を動的に選択可能な分散ストリーム処理を取り扱う．この分散ストリーム処理環境では，対象情報源の切替わりにともない，情報源から利用者へのネットワーク上のデータ転送経路とネットワーク使用量が変わる．そこで，対象情報源の変化に合わせて分散ストリーム処理環境中の各ストリーム処理エンジンが行う処理を動的に組み替えることで，ネットワーク使用量を最適化する分散ストリーム処理管理システムを構築した．本論文では，(1)，(2) における機能の詳細について述べる．また，それぞれの機能の評価実験について述べる．

Implementation and Evaluation of Stream Processing with Dynamic Selection of Target Information Sources

Kosuke Ohki,[†1] Yousuke Watanabe,[†2]
Hiroyuki Kitagawa[†1,†3] and Hideyuki Kawashima[†1,†3]

The volume of stream data delivered from different information sources is increasing. There are a variety of demands to utilize such stream data for applications. Stream processing engine can continuously process stream data according to user requests. In conventional frameworks, the user must explicitly specify information sources in advance, and the engine cannot change information sources during query processing. However, there are many cases in which target information sources change over time. We therefore need an additional framework to deal with changes of the target information sources in the same query. We propose dynamic source selection in stream processing. This paper includes two contributions. (1) Dynamic source selection in a single node. We have implemented a framework of dynamic source selection in a stream processing engine. (2) Dynamic source selection in multiple nodes. We focuses on distributed stream processing which can dynamically select target information sources during query processing. When target information sources change, data transfer routes and network traffic from the target information sources to users also change. We therefore propose a management system to keep the optimal network traffic for distributed stream processing with dynamic selection of target information sources. The system reconsiders operator allocations according to changes of target information sources. In this paper, we explain our framework in detail, and we present efficiency of our approach.

1. はじめに

近年，デバイス技術やネットワーク技術の普及にともない，実世界から能動的に配信されるストリームデータと呼ばれるデータに対する高度利用要求に注目が集められている．ストリームデータの例としては，実環境におけるセンサデバイスから配信される温度・光・加速

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 東京工業大学学術国際情報センター
Global Scientific Information and Computing Center, Tokyo Institute of Technology
†3 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba

度データ，GPS・位置タグからの位置情報，ネットワークカメラからの映像データなどがある．これらストリームデータに対する処理要求としては，温度センサにおいて閾値以上の温度が検知されたときに通知してほしいというようなフィルタリング要求や，温度センサとカメラのデータを統合して，閾値以上の温度が配信されたときに，その場所の映像を温度とともに見たいというような異なる情報源との統合利用要求などがあげられる．しかし，利用者が一からこれらの処理を行うアプリケーションを作ることは困難である．そこで，ストリームデータ処理のための基盤システムであるストリーム処理エンジン[3),5),7)] に注目が集まっている．ストリーム処理エンジンでは，利用者から問合せが登録されると，問合せ中に指定された情報源から配信されるストリームデータに対して問合せを連続して実行し，処理結果を利用者へ送る．また，実世界情報を配信する情報源などは，地理的に分散して存在していることが多い．広域に配置された情報源に対する有効な処理方法として，分散ストリーム処理[4),6)] の研究が行われている．分散ストリーム処理では，ストリーム処理エンジンが地理的に分散配置され，各エンジンが連携することで分散ストリーム処理環境全体で効率的な処理を行うことを目的としている．

ストリームデータに対する処理要求の一種として，膨大なストリームデータの中から処理してほしいストリームデータを動的に選択してほしいという要求がある．たとえば，大規模なネットワークカメラが整備された環境において，位置情報を発するセンサを身に付けた特定の人物の映っているカメラからの映像が欲しいという要求を一例としてあげる．このとき，利用者が欲しいストリームデータは特定人物の最寄りのネットワークカメラからの映像データのみであるが，人物は移動するので，その位置情報の変化にともなって接続するネットワークカメラを連続的に切り替えることが必要となる．しかし，従来のストリーム処理エンジンで上記の要求を実行するには，処理対象となる可能性のある全ネットワークカメラから配信される映像データを収集し，必要なものを選択しなければならず，非常に高い処理コストを要する．一方，人物の位置情報をもとに最寄りのネットワークカメラを探索し，そのネットワークカメラからの映像のみを処理することが可能であれば，処理コストを大幅に抑えることができると考えられる．そこで，本研究では対象情報を動的に選択可能なストリーム処理の枠組みを提案する．

本研究は，単独ノードのみが利用可能な環境と，複数ノードを利用可能な環境を考慮し，単独ノード上で情報源の動的選択を実現するための研究と，複数ノード上で情報源の動的選択を実現するための研究の 2 点からなる．ここで，ノードはストリーム処理エンジンが動作しているマシンを指す．

単独ノードを利用した情報源の動的選択では，問合せ処理中に処理対象の情報源を動的に探索可能とする機能を我々が研究・開発をしているストリーム処理エンジン StreamSpinner[17)] に追加した．情報源の動的選択機能として，入力されるデータに基づいて対象情報源を選択する新規演算を提案する．また，対象情報源からのストリームデータを処理する必要がある期間だけ，その情報源への接続を動的に確立するための情報源の接続管理機能も提案する．

複数ノードを利用した情報源の動的選択では，情報源の動的選択機能を有したストリーム処理エンジンを複数組み合わせた分散ストリーム処理を取り扱う．この分散ストリーム処理環境では，対象情報源の変化にともない，単位時間あたりのデータ転送量であるネットワーク使用量が変化する．そこで，各ストリーム処理エンジンが扱う演算の配置を動的に組み替えることでネットワーク使用量を最適に保つ分散ストリーム処理管理システムを構築した．提案する演算配置最適化手法では，対象情報源とノード間の遅延とデータレートの変動に着目し，演算配置を決める．

本論文の構成は以下のとおりである．2 章でストリーム処理と情報源の動的選択要求要求について述べる．3 章で単独ノードを利用した情報源の動的選択について述べ，4 章で複数のノードを利用した情報源の動的選択について述べる．5 章において評価実験の結果を示し，6 章で関連研究について述べ，7 章では本論文をまとめる．

2. ストリーム処理と情報源の動的選択要求

本章では，ストリームデータに対する処理基盤であるストリーム処理エンジンのアーキテクチャと，ストリーム処理における情報源の動的選択要求について述べる．

2.1 ストリーム処理エンジン StreamSpinner

本研究で用いるストリーム処理エンジン StreamSpinner のアーキテクチャは **図 1** である．ラッパは，ストリームデータを受け取るモジュールであり，情報源からストリームデータが到着すると，受け取ったデータを表形式に変換して，メディエータに送る役割を持つ．情報源ごとの多様なデータ構造をラッパで表形式に変換することで，ネットワークカメラ，位置情報，センサデータ，データベースなど異なる情報源どうしの統合利用が可能である．利用者からの問合せは SQL ライクな形式で記述される．図 1 の問合せは温度センサの値が 40 度以上のときにカメラからの最新の 1 分間の映像を取得したいという要求である．MASTER 節に記述されている情報源からストリームデータが到着するたびに問合せが繰り返し評価される．FROM 節の [now]，[1 min] はウィンドウ幅と呼ばれるものであり，演算の適用対象となるタプルを選択するための時間幅のことを指す．now が登録されると，MASTER 節に

3　対象情報源を動的に選択可能なストリーム処理の実装と評価

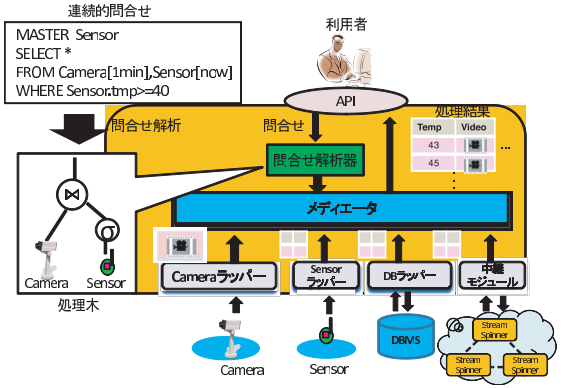


図 1　StreamSpinner のアーキテクチャ
Fig. 1　Architecture of StreamSpinner.

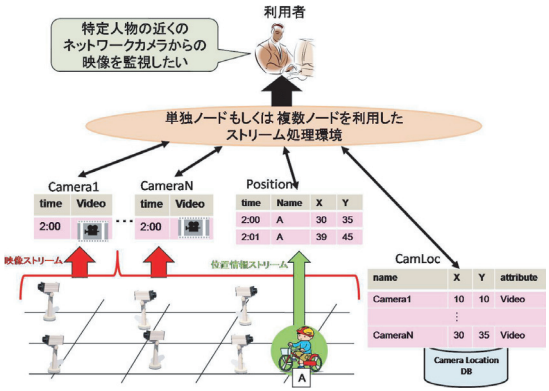


図 2　人物映像監視アプリケーション
Fig. 2　Application of tracking moving objects.

記された情報源からのデータの到着時刻と同じタイムスタンプを持つタプルが処理対象になり，1 min が登録されると，最近 1 分間に到着したデータに対して問合せが評価される．問合せ解析器では，問合せから各リレーショナル代数演算子の実行順序を木構造で表現した処理木を生成し，メディエータに送る．メディエータでは，処理木に基づいてラッパから受け取ったデータに対して連続的に問合せ処理を行う．

また，StreamSpinner では，他の StreamSpinner と中継モジュールを介して連結することで，分散ストリーム処理をすることが可能である．中継モジュールには連結先の StreamSpinner の IP アドレス，登録する問合せ要求，その問合せの処理結果を受け取るときの型に関する情報が必要となる．連結先の StreamSpinner からの処理結果は受信側の StreamSpinner では，入力ストリームデータとしてメディエータで処理される．

2.2　ストリーム処理における情報源の動的選択要求

膨大なストリームデータの中から，動的に必要な情報源を選択してほしいという要求は多くある．たとえば，**図 2** に示すように，地理的に分散したネットワークカメラが N 台設置されており（Camera1, . . . ,CameraN），各ネットワークカメラの配置場所などを登録したカメラ位置データベース（CamLoc）が整備された環境において，位置情報を発するセンサ（Position）を身に付けた人物 A の映像を監視したいというような要求があるとする．このような要求の具体的なアプリケーションとしては，児童の登下校見守りシステム[20]などがあげられる．このとき，通常のリレーショナル代数演算のみで問合せを記述すると

```
MASTER Position
SELECT *
FROM(
    SELECT Camera1.Video
    FROM Position[1msec], Camera1[1min], CameLoc
    WHERE Position.Name='A'
      AND CamLoc.Name = 'Camera1'
      AND distance(CamLoc.X, CamLoc.Y, Position.X, Position.Y)<50
    UNION
    .
    .
    .
    UNION
    SELECT CameraN.Video
    FROM Position[1msec], CameraN[1min], CamLoc
    WHERE Position.Name='A'
      AND CamLoc.Name='CameraN'
      AND distance(CamLoc.X, CamLoc.Y, Position.X, Position.Y)<50
    )AS AllCamera
```

図 3　Union 句を用いた問合せ記述
Fig. 3　Query contained union clause.

図 3 のように書くことができる．図 3 の副問合せでは，まずネットワークカメラごとに位置情報および座標データベースとの結合を行っている．関数 distance は，人物 A とカメラの距離を計算するユーザ定義関数で，距離が 50 メートル未満であれば条件を満たす．Camera1, . . . ,CameraN に対して同様の結合・選択処理を行った結果に UNION を適用し

たものが副問合せの結果である．この問合せを用いれば，目的とする処理結果を得ることはできる．しかし，問合せ記述の肥大化と，全ネットワークカメラへの常時アクセスのための高負荷という問題点が生じる．従来のストリーム処理エンジンは，図 3 のような SQL をベースとした問合せ言語[2] を採用しているため，使用する情報源を FROM 節などに明示的かつ固定的に記述せざるをえなかった．

一方で，位置情報とカメラ位置データベースを用いて人物 A の最寄りのネットワークカメラを探索し，そのネットワークカメラからの映像のみを取得することで，システム内で使用される計算機資源を少なく抑えられる可能性があるが，人物 A の位置情報の変化にともなって接続するネットワークカメラを連続的に切り替える機能が必要となる．上記のような，問合せ処理中にストリーム処理エンジンが対象情報源からのストリームデータを動的に選択をする機能は従来のストリーム処理の枠組みには存在していない．

そこで，本研究において情報源の動的選択のための枠組みを新たに提案する．

3. 単独ノードを利用した情報源の動的選択

3.1 情報源の動的選択の概要

本研究では，利用者がある時点において処理をしてほしいデータを提供する情報源の集合を対象情報源と呼ぶ．この対象情報源を動的に選択する処理の概要を**図 4** を用いて説明する．情報源の動的選択は，問合せ処理中に対象情報源を探索する処理と，探索結果である対象情報源を動的に選択する処理の 2 つからなる．図 4 右側が探索処理であり，図 4 左側が対象情報源に関する情報を受け取り，動的にその対象情報源を選択する処理である．標準のリレーショナル代数演算には，入力されるデータに基づいて情報源を動的に選択する演算が存在しないため，新しく Target Selective Join 演算を提案する．

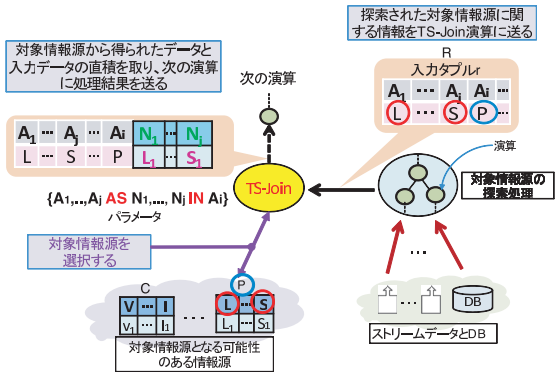


図 4　情報源の動的選択の概要
Fig. 4　Overview of dynamic source selection.

以下，情報源の動的選択の仕組みについて述べる．

3.2 TS-Join 演算

本研究では，動的な情報源選択のために新たに Target Selective Join 演算を提案する．以降 TS-Join 演算と省略する．TS-Join 演算は，入力タプルに含まれる複数の属性値を，リレーション名，属性名と見なして，対応するリレーションの属性と入力タプルとの直積を出力する．TS-Join 演算を以下のように定義する．TS-Join 演算への入力はリレーション $R(A_1, \ldots, A_n)$ とする．

$$TS\text{-}Join_{A_i,\{A_1,\ldots,A_j\},\{N_1,\ldots,N_j\}}(R)$$
$$= \bigcup_{r \in R} \{r \times t | t \in \delta_{N_1,\ldots,N_j} \pi_{attr(r.A_1,\ldots,r.A_j)}(relation(r.A_i))$$

ここで，r は R におけるタプルを指す．$relation(r.A_i)$ はタプル r における A_i の値に対応するリレーションであり，$attr(r.A_1, \ldots, r.A_j)$ はタプル r の属性 $A_1, \ldots, A_j$ の値に対応する属性名の集合を表す．これらに対応するリレーションや属性がなかった場合は，タプル [r, ε] を出力するものとする．$\delta_{N_1,\ldots,N_j}$ はリネーム演算であり，属性名を $N_1, \ldots, N_j$ に変更する．TS-Join 演算中の 3 つのパラメータは以下である．

- A_i は対象情報源の名前を値に持つ属性．
- $\{A_1, \ldots, A_j\}$ は対象情報源中の取得したい属性集合を値として持つ R の属性集合．
- $\{N_1, \ldots, N_j\}$ は対象情報源中から取得した値を保持する属性集合．

TS-Join 演算の動作を図 4 を用いて説明する．TS-Join 演算は図中に記されているパラメータを持ち，図 4 の対象情報源の探索処理によってリレーション R が得られると，まず，$r.A_i$ の値から対象情報源の名前を得る．ここでは，$r.A_i$ の値である P と C が対象情報源となる．そして，その対象情報源の中から $r.A_1, \ldots, r.A_j$ の値に該当する $P.L, \ldots, P.S$ と $C.V, \ldots, C.I$ の属性値を取得し，対象情報源の探索処理から送られた入力データとの直積をとる．最後に，対象情報源から新たに取得した属性を $N_1, \ldots, N_j$ にリネームし，次の演算へ処理結果を送る．

3.3 情報源の動的選択を用いた応用例

問合せ記述中で TS-Join 演算を用いるには FROM 節の中に以下のように記述する．

$$TS\ JOIN\ \ A_1[, \ldots, A_j]\ \ AS\ \ N_1[, \ldots, N_j]\ \ IN\ \ A_i$$

```
MASTER Position
SELECT *
FROM(
    SELECT *
    FROM Position[1sec], CamLoc
    WHERE Position.Name ='A' AND
    distance(CamLoc.X, CamLoc.Y, Position.X, Position.Y)< 5
) TS JOIN Attribute AS Video IN Name
```

図 5　情報源の動的選択のための問合せ
Fig. 5　Query for dynamic source selection.

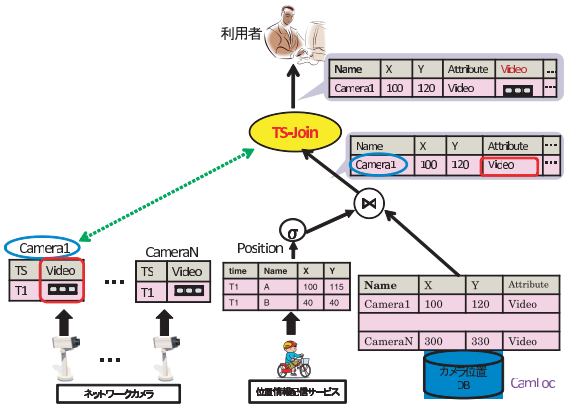


図 6　問合せ処理
Fig. 6　query processing.

$A_1, \ldots, A_j$, A_i, $N_1, \ldots, N_j$ は 3.2 節と対応している．

情報源の動的選択の問合せ例として，2 章において紹介した人物の映像監視要求を用いる．**図 5** の問合せは CamLoc に登録されているネットワークカメラの中から人物 A の位置とネットワークカメラの距離が 5 未満のネットワークカメラを探索し，TS-Join 演算によって動的にネットワークカメラを選択して，その映像ストリームを得ることを目的とした問合せとなっている．

図 5 の問合せ処理の流れは**図 6** である．位置情報配信サービスから配信される各人物の位置情報の中から人物 A を選択し，人物 A の位置情報から距離が 5 未満であるネットワークカメラをカメラ位置 DB に登録されているすべてのネットワークカメラから探し出し，結合演算の処理を行う．この例では，Camera1 が 5 m 未満のネットワークカメラとして探索されている．次に結合演算で得られたタプルの Name 属性の値である Camera1 を基に TS-Join 演算で Camera1 の選択を行う．TS-Join 演算は選択されたネットワークカメラの Video 属性に含まれる映像データを新しい属性名 Video の値として入力タプルに付加し，利用者へ配信する．位置情報が配信されるたびに，同様の処理が繰り返し行われ最寄りのネットワークカメラからの映像のみが利用者へ配信される．

3.4　TS-Join 演算を含む処理木の構築

ここでは TS-Join 演算を含んだ問合せ記述に対応する処理木の構築について述べる．まず，問合せ要求に対して，TS-Join 演算および各リレーショナル代数演算の実行順序を木構造で表現した処理木の候補を生成する．このときに，依存関係をもたない演算どうしが複数含まれる場合は，それらの実行順序について処理木の候補が複数得られる．交換可能なリレーショナル代数演算どうしの実行順序は従来の問合せ最適化方式[1)] に基づいて決定する．TS-Join 演算の場合は，TS-Join が関与する属性（$A_1, \ldots, A_j$，A_i，$N_1, \ldots, N_j$）と関係ないリレーショナル代数演算とは実行順序の依存関係がないので交換可能である．しかし，TS-Join 演算の入力となる対象情報源は探索処理の結果によって変動するので，TS-Join 演算が出力するデータ量もそのたびに変動して正確に見積もることが困難であり，TS-Join 演算以降の処理プランに影響を与えてしまう．よって，TS-Join が全体の処理コストの見積りに与える影響を少なくするように，TS-Join 演算の実行順序が他の交換可能な演算よりも後にくるような処理木の候補を選択する．

3.5　接続管理機能

情報源に常時接続を行い，そのストリームデータを蓄積し続けることは，システムにとって大きな負荷となる．TS-Join 演算により対象情報源からのストリームデータを処理する前に，情報源への接続の確立を行い，その接続が不要になったときに動的に接続を解除する機能があると，システム負荷を減らすことが可能となる．そこで，情報源への接続を動的に管理する接続管理機能をストリーム処理エンジンのメディエータに導入した．

利用者が接続管理機能を利用するには，TS-Join 演算が必要とする情報源の集合を包含するような条件指定をした問合せを与えることになる．接続管理機能では，問合せ処理中に探索して得られた情報源に対応するラッパを動的に稼働させ，ストリームデータの受信を行う．また，TS-Join 演算で処理する必要がなくなった情報源は，接続管理機能がその情報源に対応するラッパを無効化し，接続を解除する．接続管理機能における動的接続・解除を行うための問合せについては，3.5.1 項と 3.5.2 項で説明をする．

```
MASTER Position
ACTIVATE CamLoc.Name
FROM Position[now], CamLoc
WHERE Position.Name = 'A' AND
distance(CamLoc.X, CamLoc.Y, Position.X, Position.Y)< 10
```

図 7　情報源への動的な接続のための問合せ
Fig. 7　Query to create connections.

```
MASTER Position
DEACTIVATE CamLoc.Name
FROM Position[now], CamLoc
WHERE Position.Name = 'A' AND
distance(CamLoc.X, CamLoc.Y, Position.X, Position.Y)≥ 10
```

図 8　情報源への動的な接続解除のための問合せ
Fig. 8　Query to release connections.

3.5.1　情報源への接続確立

動的接続・解除の条件指定も SQL ベースの言語で行う．接続条件を利用者自身が書けるようにしたことで，用途に応じた柔軟な接続管理が可能である．**図 7** は，人物 A とカメラの距離が 10 未満のネットワークカメラへの接続を動的に確立するための問合せである．この問合せの条件指定は図 5 の対象情報源への条件を包含したものとなっている．ACTIVATE 節に指定された属性に含まれる情報源に対して，接続管理機能がその情報源への接続を確立する．

3.5.2　情報源への接続解除

図 8 は，人物 A とカメラの距離が 10 以上離れたときに，そのネットワークカメラへの接続を動的に解除する問合せである．DEACTIVATE 節に指定された属性に含まれる情報源に対して，接続管理機能がその情報源への接続を解除する．このように接続する情報源を必要最低限に抑えることで，ストリーム処理エンジンへの負荷を抑えることができる．しかし，動的接続管理のための記述は，柔軟な対応ができるという利点がある反面，利用者の負担が増えてしまうことも懸念される．その対策として，TS-Join の問合せ記述から接続・解除の条件をシステムが自動で導出するというアプローチが考えられる．それについては今後の課題である．

3.6　実　　装

情報源の動的選択機能を有したストリーム処理エンジン StreamSpinner のプロトタイプシステムの開発について説明する．StreamSpinner は JavaSDK によって実装されている．

図 9　動的選択機能を有した StreamSpinner の動作画面
Fig. 9　Screenshot of streamspinner with dynamic source selection.

情報源の動的選択のために提案した TS-Join 演算を StreamSpinner のメディエータ部に追加した．情報源の接続管理機能もメディエータ部に追加した．

カメラからの映像データの取扱いのために，映像データ変換などに JMF [15]（Java Media Frameworks）のクラスを使用している．映像を取得するためのラッパには QuickTime for Java [18] を使用して，プログラム上で QuickTime を起動し，再生された映像を表形式に変換している．このカメララッパを複数生成することで，数台のカメラ映像を同時に受け取ることができる．

人物の映像監視要求を実行している実行画面は**図 9** である．図の左側はラッパの一覧である．特定人物から一定距離内にあるカメラに対応するラッパが動的に作られている．図の中央下に TS-Join 演算で選択された 2 台のカメラからの映像ストリームが表示されている．このように，単独ノードを利用した情報源の動的選択機能は実装を行った．本章の提案内容についての評価は 5.1 節で示す．

4.　複数ノードを利用した情報源の動的選択

本章では，3 章での提案内容を発展させ，複数ノードを利用した情報源の動的選択について述べる．本研究では，対象情報源が変化する分散ストリーム処理環境上のネットワーク使用量を最適化する分散ストリーム処理管理システムを提案する．

4.1　対象情報源が変化する分散ストリーム処理

対象情報源の動的変化を考慮した分散ストリーム処理について述べる．**図 10** は分散ストリーム処理の例である．2.2 節の要求と同じで，図 5 の問合せを用いる．ここでは，ノード

7　対象情報源を動的に選択可能なストリーム処理の実装と評価

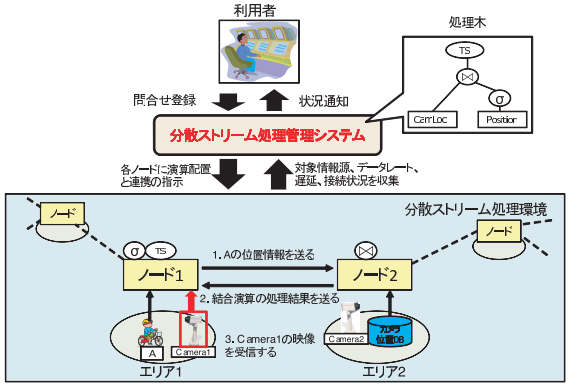


図 10　対象情報源の動的変化を考慮した分散ストリーム処理
Fig. 10　Distributed stream processing with dynamic selection of target information sources.

1 とノード 2 からなる分散ストリーム処理環境を用いて説明する．図 10 では，ノード 1 が人物 A から 5 m 未満にある Camera1 の映像を受信している．そして，人物 A がエリア 2 に移動するとノード 1 は動的に Camera2 に切り替えて映像処理を続けて行う．このときに，ノード 2 と Camera2 との間で行われる通信のネットワーク使用量が，ノード 1 と Camera2 の間でのネットワーク使用量よりも少ない場合には，ノード 1 で行っている Camera2 からの映像の処理をノード 2 に移すことで，ネットワーク使用量を最小に保つことができる．そこで，本研究では対象情報源と各データ転送経路のネットワーク使用量を集計し，最適なネットワーク使用量を保つように管理をする分散ストリーム処理管理システムを構築した．

分散ストリーム処理管理システムに問合せが登録されると，ネットワーク使用量を算出するためにデータレートと通信遅延を分散ストリーム処理環境から収集し，4.3 節で述べる初期演算配置最適化手法により最適な演算配置を計算し，各ノードへ配置・連結要求を出す．そして，定期的に分散ストリーム処理環境から対象情報源，データレート，遅延を収集し，再演算配置最適化手法によりその時刻の最適な演算配置とネットワーク使用量の見積り値を計算する．この見積り値が，現在の分散ストリーム処理環境におけるネットワーク使用量よりも改善されていたら，演算の再配置を各ノードに指示する．次節で分散ストリーム処理管理システムのアーキテクチャを説明する．

4.2　分散ストリーム処理管理システム

分散ストリーム処理管理システムのアーキテクチャは**図 11** である．各モジュールの説明

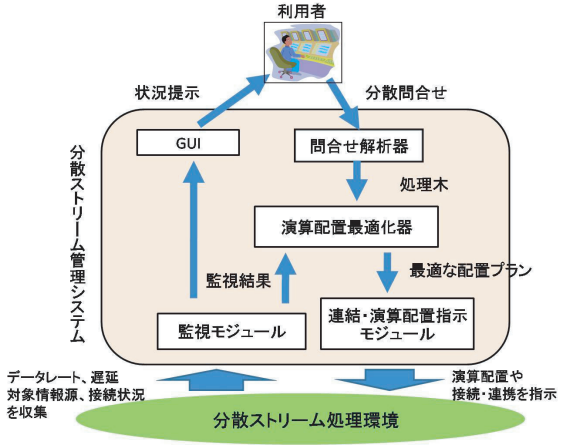


図 11　分散ストリーム処理管理システムのアーキテクチャ
Fig. 11　Architecture of distributed management system.

を下に列挙する．

- 問合せ解析器：利用者からの分散問合せを解析し，処理木を生成する．処理木を生成する際には，3.4 節で述べた内容と同様の方針を用いる．生成された処理木は演算配置最適化器に送られる．
- 演算配置最適化器：演算配置最適化器は演算配置最適化手法を用いて分散ストリーム処理環境のネットワーク上の転送量を最小に保つ．演算配置最適化は監視モジュールから送られてくる分散ストリーム処理環境の情報をもとに行われ，算出した配置プランは連結・演算配置指示モジュールへと送られる．
- 連結・演算配置指示モジュール：演算配置最適化器から受け取った演算の最適配置を各ノードに指示をする．
- 監視モジュール：監視モジュールは分散ストリーム処理環境から対象情報源に関する情報と，データレート・遅延の収集管理を行う．収集した情報は GUI 表示と演算配置最適化器に送られる．
- GUI：GUI は分散ストリーム処理環境の各ノードどうしの連結・接続状況，演算の配置状況，現在の対象情報源を視覚的に利用者に伝える．

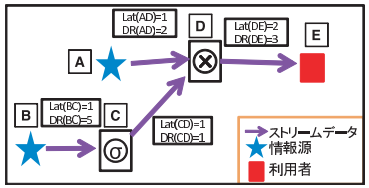


図 12　分散ストリーム処理の例
Fig. 12　An example of distributed processing.

4.3　演算配置最適化手法

本節では，演算配置最適化手法について説明する．最適な演算配置を決定する指標となるネットワーク使用量は Pietzuch らの手法[8] で用いられた以下の式を用いて算出する．

$u(q) = \Sigma_{e \in E} DR(e)Lat(e)$

ここで，q は問合せを表し，$u(q)$ はその問合せに要するネットワーク使用量である．E はノード間のリンクの集合，$DR(e)$ はリンク e 上を流れるデータレート，$Lat(e)$ はリンク e における遅延を表す．なお，通信遅延の単位は秒であり，入力データレートの単位はバイト毎秒である．ネットワーク使用量について**図 12** を用いて具体的に説明する．

図では，利用者の問合せを分散ストリーム処理している．この問合せに要するネットワーク使用量は上記の式を用いて次のように計算される．

$DR(AD)Lat(AD) + DR(BC)Lat(BC) + DR(CD)Lat(CD) + DR(DE)Lat(DE) = 1 \times 2 + 1 \times 5 + 1 \times 1 + 2 \times 3 = 14(Byte)$

ネットワーク使用量を最小にする初期演算配置最適化手法，再演算配置最適化手法のアルゴリズムを次に説明する．

4.3.1　初期演算配置最適化手法

初期配置最適化手法では，問合せ処理を実行する前段階であるので，対象情報源が不明である．そこで，TS-Join 演算を含む処理木を既存の最適化手法[8] で扱えるようにするために，動的に変わりうる部分をダミーの情報源に接続するものとして扱う．ダミー情報源のデータレートはすべての情報源からのデータレートの平均値とし，ダミーの情報源とノード間の通信遅延はすべてのノード間の通信遅延の平均値とする．このように仮の処理木を作ることで，既存手法を利用可能とした．

初期演算配置最適化手法のアルゴリズムは以下である．

(1) 分散ストリーム処理環境からデータレート・遅延を収集．
(2) 既存手法を利用するために，TS-Join 演算にダミーの情報源を付加して処理木を構築．
(3) 全配置候補を列挙し，最もネットワーク使用量が小さい演算配置を算出．

4.3.2　再演算配置最適化手法

再演算配置最適化は定期的に分散ストリーム処理環境の最適な演算配置を計算し，ネットワーク使用量を最適に保つ．

再演算配置最適化手法のアルゴリズムは以下である．

(1) 分散ストリーム処理環境から対象情報源・データレート・遅延を収集．
(2) TS-Join 演算と対象情報源とをリンクさせ処理木を再構成し，最適な演算配置を算出．
(3) 算出された演算配置におけるネットワーク使用量の見積もり値が実行中の演算配置のネットワーク使用量よりも少なければ，演算の再配置を指示．

演算の再配置の具体的な流れは，以下である．

(a) 各ノードに配置された演算のうち，再配置の必要なものを削除する．このときに，削除する演算の入出力が別のノード上の演算から与えられている場合は，データ送受信のためのノード間の接続を解除する．
(b) 算出した最適な演算配置プランに基づき，各ノードに必要な演算を配置する．
(c) TS-Join 演算の動作しているノードで対象情報源への接続を確立する．

演算の再配置を実行した際にかかるオーバヘッドは，上記 (a) から (c) までの処理時間となる．これらは再配置する演算数や接続中の情報源数などによって変化するが，ストリーム処理では 1 つの問合せの演算数が数十個にのぼることはきわめて稀であり，また同時接続する情報源数も動的接続管理機能への要求記述によって調整可能であるため，十分実用的な時間で演算再配置が実行可能であると考えられる．

4.4　広域分散環境 InTrigger 上での運用

分散ストリーム処理管理システムの動作確認のため，広域環境での管理・運用を行った．分散ストリーム処理環境は，情報爆発プロジェクト[19] の InTrigger[13] を使用した．InTrigger は，現在，日本国内で 11 拠点にまたがり数百ノードを持つ，分散処理の研究などのために構築されたプラットフォームである．**図 13** のように，分散ストリーム処理環境管理システムを東大に配置し，分散ストリーム処理環境は 5 ノードで構成されている．ここでは，人物の映像監視要求に対して，広島大のノードに割り当てられた TS-Join 演算により，監視対象の人物付近にあるネットワークカメラからの映像を受信している．ただし，今回の InTrigger 上での実験では実際のネットワークカメラの映像データ代わりに疑似データを使用した．これは，InTrigger の各ノードの OS が Linux であり，QuickTimePlayer が Linux 上ではサ

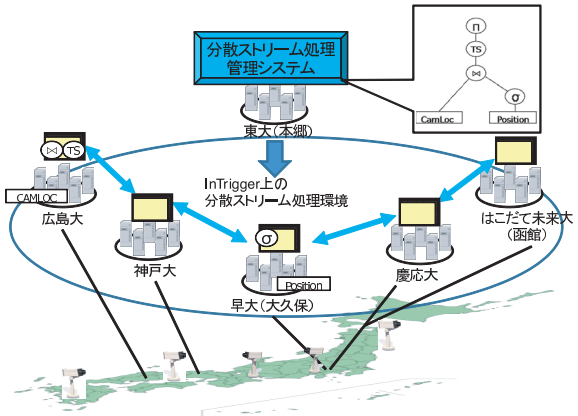


図 13　広域分散環境 InTrigger 上の動作状況
Fig. 13　Performance in InTrigger.

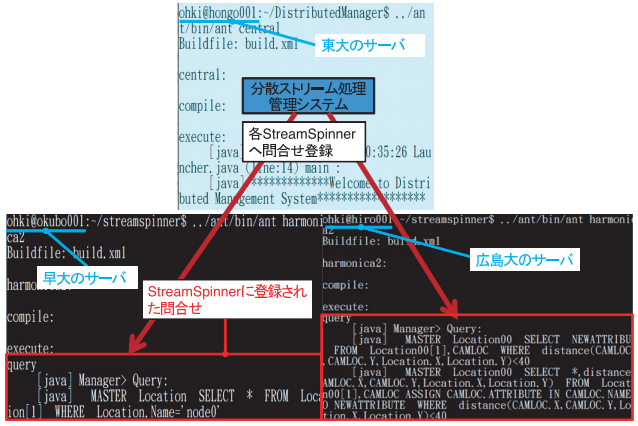


図 14　InTrigger 上での実行画面
Fig. 14　Screenshot of prototype system in InTrigger.

ポートをされていないことから，人物の映像監視のための QuickTimePlayer による映像の取扱いができないことが理由としてあげられる．そこで，今回は映像データの代わりに疑似データで処理を行った．実際に分散ストリーム処理管理システムが各ノードに連結指示をしているスクリーンショットは**図 14** となっている．このスクリーンショット上では，演算配置の要求を受けていないノードの画面を省略している．

以上述べたように，分散ストリーム処理管理システムを実装し，広域分散環境上において運用管理が可能であることを確認した．システムの評価については，5.2 節で述べる．

5. 評 価 実 験

本章では，単独ノードと複数ノードを用いた情報源の動的選択に対する評価実験について述べる．

5.1　単独ノードを用いた情報源の動的選択の評価実験

単独ノードを利用した情報源の動的選択機能の有効性を検証するための評価実験を行った．情報源の動的選択要求には，図 5 と同様の人物の映像監視要求を用いた．ストリーム処理エンジン StreamSpinner が動作しているマシンは OS: windows vista，CPU: AMD Athlon 64 × 2 Dual Core Processor 3800 + 2.00 GHz，Memory: 2 GB である．評価実験で測定するシステムの負荷は，java 測定ツールである jconsole [14] を用いて，CPU 占有率と Memory 使用量を測定した．必要な台数分ネットワークカメラを用意することはコストがかかるため，本節の実験ではあらかじめ撮影した複数種類の映像データを，映像配信ソフト Darwing Streaming Server [12] によりストリーム配信することで，実際のネットワークカメラの代用とした．また，位置情報データは移動オブジェクトの行動シミュレータである Mobireal [16] を使用して生成した．

実験は以下の 2 点に関して行った．

(1) 人物の映像監視要求を処理する際に，情報源の動的選択機能を用いた手法と，すべてのカメラに常時接続する方法におけるシステム負荷の比較．
(2) カメラ 100 台と移動人物 1,000 人から構成される大規模なストリーム配信環境において，提案手法を用いたときのシステム負荷の測定．

5.1.1　実　　験

提案手法と従来手法の比較実験

人物の映像監視要求を処理する際に，全カメラに接続する手法（Naive Method）と，提案した情報源の動的選択手法（Proposed Method）を用いた場合の StreamSpinner の負荷の比較実験を行った．10 台のカメラと，位置情報を発する人物 1 人を配置しており，1 秒ごとに位置情報を取得している．実験に用いた問合せは以下である．

(1) 全カメラに接続を行う問合せ（従来手法）

全カメラに接続を行う問合せは図 3 中の CameraN を Camera10 までとした問合せである．全カメラに接続を行う問合せではすべてのカメラへの接続を確立したうえで，追跡人物から距離が 50 のカメラからの映像を取得することになる．

(2) 情報源の動的選択機能を用いた問合せ（提案手法）

情報源の動的選択機能を利用するために，3 つの問合せを登録した．TS-Join 演算を用いた問合せは図 5 と同様のものであり，接続管理のための問合せは，図 7 と図 8 と同様のものである．ただし，選択条件のパラメータはそれぞれ 50，60，60 と変更している．

2 つの手法におけるシステムの CPU 占有率とメモリ使用量を測定した．全カメラに接続を行う問合せと，情報源の動的選択機能を用いた問合せの比較実験の結果は**図 15** である．ここで，横軸は計測時間（秒），縦軸は CPU 占有率とメモリ使用量である．提案手法が従来手法に比べて優れていることが分かる．[80, 200]，[270, 330]，[380, 460] の区間でカメラへの接続が確立されるたびに，CPU 占有率とメモリ使用量が増える．しかし，システムが接続を解除すると CPU 占有率は減少した．この結果から，提案手法ではハードウェア資源を低く抑えることが可能であることが検証された．

大規模ストリームデータ配信環境下における提案手法の検証実験

大規模な映像ストリーム環境において，限られた資源の中でも提案手法が有用であることを確認する実験を行った．位置情報を発する人物 1,000 人，カメラ 100 台からなるストリーム環境において，1 人の人物を追跡する．単独マシンで接続できる最大カメラ台数は限られており，大規模ストリームデータ配信環境下では，常時接続の手法は利用することができない．提案手法では，図 5，図 7，図 8 のパラメータをそれぞれ 5，10，10 とした問合せを用いて，システムの CPU 占有率とメモリ使用量を測定した．実験結果は**図 16** である．ここで，横軸は計測時間（秒），縦軸は CPU 占有率とメモリ使用量である．提案手法を用いた場合，計測した 500 秒の間に追跡人物がカメラの近くに接近した回数は 7 回となった．追跡人物の移動にともない，最寄りのカメラへの接続・解除が行われ，そのたびに CPU 占有率，メモリ使用量ともに増加する．しかし，CPU 占有率は一定の値を示しており，大規模映像ストリーム環境に対して単独のマシン上でストリーム処理エンジンの動作が可能であることを示しており，提案手法の有用性を確認できる結果となっている．

ここで，実験結果からメモリリークが発生し，メモリ使用量が増加していることが分かる．その原因として，カメラへの接続解除の不備があげられる．ラッパの接続の解除指示が正常に反映されず，映像の受信を続けることが原因となっている．対策としては，依存ライブラリやラッパ実装方式の変更などにより改善可能であると考えている．

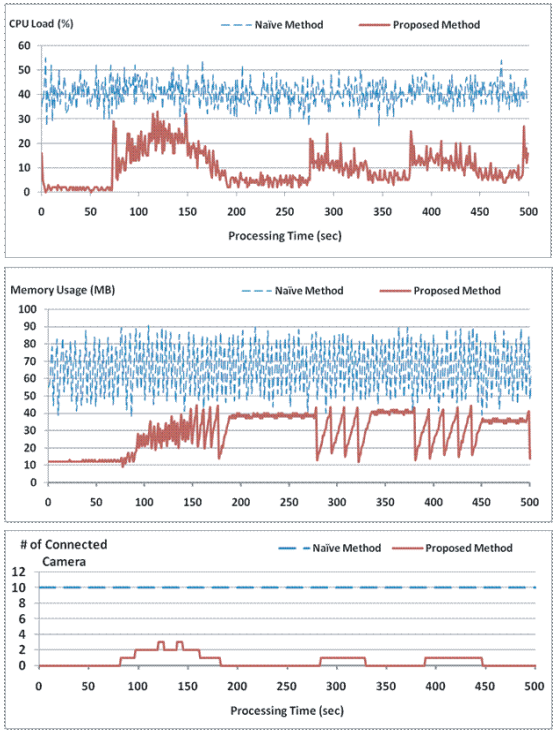


図 15　提案手法と従来手法を利用したときの CPU 占有率（上），メモリ使用量（中），接続したカメラ台数（下）の比較

Fig. 15　CPU Load and Memory Usage and number of connected cameras in comparison of naive and proposed approaches.

100 台のカメラへ同時に接続して映像ストリームを処理するには膨大なリソースが必要となってしまうが，本実験により，提案手法を用いて情報源を動的に選択して接続するのであれば，一般的な PC 程度のリソースでも十分動作可能なことが示された．

5.2　複数ノードを用いた情報源の動的選択の評価実験

本節では，分散ストリーム処理管理システムが分散ストリーム処理環境のネットワーク使用量を様々な状況下においても最適化することを検証するためにシミュレーション実験と実環境上での実験を行った．

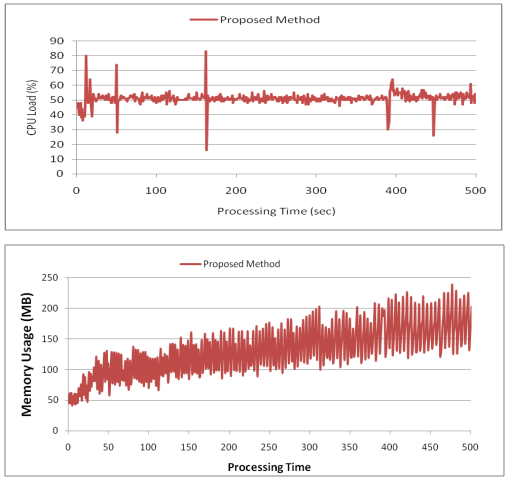


図 16　大規模ストリームデータ環境上の提案手法を利用したときの CPU 占有率（上），メモリ使用量（下）
Fig. 16　CPU Load and Memory Usage in 100 cameras environment.

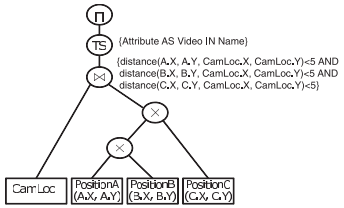


図 17　対象情報源の探索が複雑化した問合せ
Fig. 17　Query with complicated target search process.

5.2.1　シミュレーション実験

人物の映像監視要求を用いた実験

データレート，遅延を人工的に生成したシミュレーション環境で 3 種類の問合せを処理したときの評価実験に関して述べる．本実験で用いる問合せは，図 5 中の問合せ要求例としてあげた人物の映像監視要求と，**図 17** の対象情報源の探索処理が複雑化した問合せと，**図 18** の複数の TS-Join 演算を含む問合せの 3 つの問合せである．

図 17 の問合せは，3 人の追跡対象との距離がそれぞれ 5 未満であるネットワークカメラの映像が欲しいという要求である．

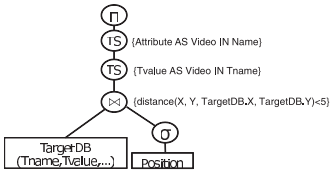


図 18　複数の TS-Join を用いた問合せ
Fig. 18　Query with multiple TS-Join.

図 18 の問合せでは，TargetDB には，図 10 の CamLoc に蓄積されている情報に加えて，Tname と Tvalue が属性として追加されている．Tname と Tvalue の値には，最初の TS-Join 演算で選択したい情報源名とその取得する属性を蓄積しておく．

各問合せにおける選択演算と結合演算と TS-Join 演算のそれぞれの選択率は 0.1，0.1，1.0 としている．処理の構成単位である各演算は分散ストリーム処理環境の各ノードに割り当てられる．

本実験では，分散ストリーム処理環境中のデータの基本となる組合せを以下のように決めた．

- 対象情報源数：4 台のネットワークカメラから 1 台のネットワークカメラを対象情報源として選択．
- 通信遅延：[1，2]（秒）の間でランダムに値を生成．
- データレート：カメラからのデータレートは [100，200]（KB/s）の間でランダムに値を発生させ，位置情報のデータレートとデータベースから取得するデータ量は [10，20]（B/s）の間でランダムに値を生成．
- ノード数：4 ノード．

データレートの設定をランダムにした理由は，実環境においては，カメラの機種によりデータレートが異なり，また，カメラから送信されるデータ量も一定値ではない状況があり，シミュレーション実験において実環境に合わせた実験を行うためである．

上記の基本となる組合せから，他をすべて同じ条件にして各項目のうちいずれか 1 つのみを変動させる．このときに，分散ストリーム処理管理システムが算出した下記の 2 種類の演算配置におけるネットワーク使用量の比較を行った．

（1）変動発生後も初期演算配置のままで固定した演算配置（Initial Alloc）．
（2）変動発生後に演算の配置場所を再評価した演算配置（Re Alloc）．

人物の映像監視要求に対して，分散ストリーム処理管理システムが基本となる組合せにおいて初期演算配置と再演算配置を算出した結果は**図 19** であり，そのときのネットワーク使

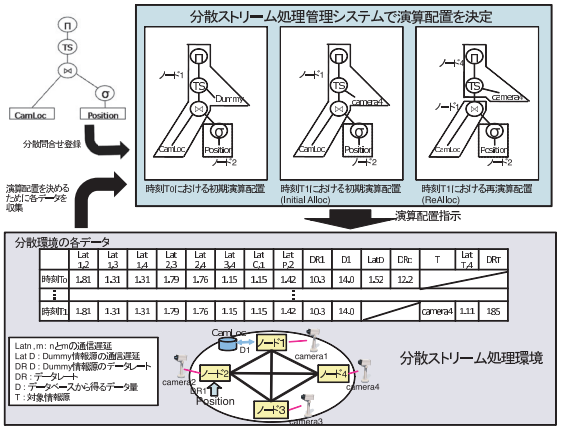


図 19　演算配置最適化手法により導出された各配置プラン
Fig. 19　Operator allocation plan generated by the proposed method.

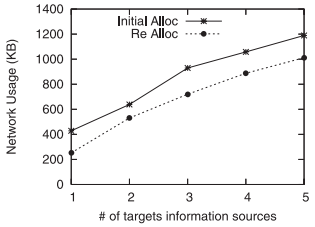


図 20　対象情報源数
Fig. 20　# of target sources.

用量は**図 20**，**図 21**，**図 22**，**図 23** の X 軸の第 1 項目のネットワーク使用量となる．分散ストリーム処理環境から初期配置を行った時刻 T_0 と対象情報源が具体的に分かって変化に合わせた再配置をした後の時刻 T_1 において各データを収集し，分散ストリーム処理管理システムがそのときの最適な演算配置を算出している．基本となる組合せと同様に，他の組合せに関しても分散ストリーム処理環境のシミュレートされた各データから演算配置とネットワーク使用量を算出している．ここで，実験結果で示すネットワーク使用量は，それぞれの変動を 5 回ずつ起こしたときにシステムが算出したネットワーク使用量の平均値である．

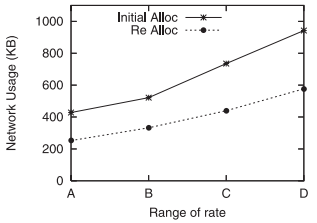


図 21　データレートの変動
Fig. 21　Changes of data rate.

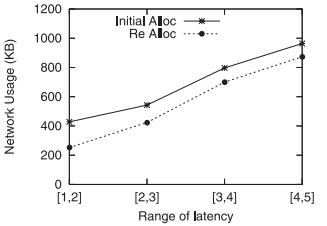


図 22　通信遅延の変動
Fig. 22　Changes of latency.

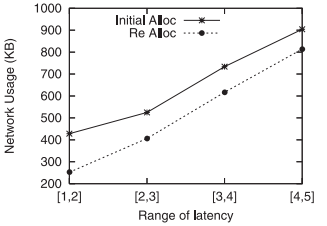


図 23　対象情報源とノード間の遅延のみ変動
Fig. 23　Changes of latency between targets and nodes.

実験 1．同時に接続する対象情報源数を変動させた比較実験

人物の映像監視要求では，利用者が同時に複数人を追跡したいという要求や，追跡人物の周辺にネットワークカメラが密集している場合が考えられる．このような状況では，TS-Join 演算が同時に処理をする対象情報源の数が増える．そこで，同時に接続する対象情報源数を

変動させ，提案手法の評価を行った．実験結果は図 20 である．TS-Join 演算が複数のネットワークカメラのような高いデータレートを持つ情報源からのデータを入力として受け取ると，処理結果のデータ量が著しく高まっていることが分かる．そして，TS-Join 演算を含めた以降の各演算が配置されたノード間の遅延がネットワーク使用量に影響を与えている．実験結果から，再演算配置により遅延の少ない演算配置をすることで，ネットワーク使用量を抑えることが可能となっている．

実験 2．データレートを変動させた比較実験

実環境において，情報源の種類によりデータレートは異なる．実環境を想定した実験を行うために，人工データにおいてもネットワークカメラと他の情報源との間にデータレートの差をつけて実験を行った．実験結果は図 21 である．横軸の A は情報源のデータレートを 10 から 20 の間でランダムに発生させ，対象情報源の候補であるネットワークカメラのデータレートを 100 から 200 でランダムに発生させたものを指している．B，C，D はそれぞれ情報源のデータレートとネットワークカメラのデータレートを {[20，30]，[200，300]}，{[30，40]，[300，400]}，{[40，50]，[400，500]} の間でランダムに発生させた状態を指す．実験結果から，D のときに，再演算配置（Re Alloc）では初期演算配置（Initial Alloc）に比べて 200 KB ほどのネットワーク使用量を抑えていることが分かる．

実験 3．通信遅延を変動させた比較実験

実環境では，ノード間のネットワーク負荷や各ノードの処理能力によって遅延が変動する．そこで，遅延を最大 5 秒までランダムに発生させて実験を行った．実験結果は図 22 である．ネットワーク使用量は 4.3 節で定義した式により算出されるので，遅延が大きくなるにつれ，ネットワーク使用量が増える．しかし，再演算配置を行うことで，より少ないネットワーク使用量を保っていることが分かる．

実験 4．対象情報源とノード間の通信遅延のみを変動させた比較実験

本実験では，ノード間の遅延は変動させずに，対象情報源とノード間の遅延のみをランダムに 5 秒まで発生させた．実験結果は図 23 である．実験 3 とほぼ同様の結果が得られた．

情報源の探索が複雑化した問合せ要求を用いた実験

図 17 の情報源の探索が複雑化した問合せに対して，実験 1，2，3，4 と同様の状況において算出した初期演算配置と再演算配置のネットワーク使用量は**図 24**，**図 25**，**図 26**，**図 27** である．探索が複雑化することで，全体的のネットワーク使用量が増えていることが分かる．しかし，いずれの状況においても，再演算配置によりネットワーク使用量を改善していることが確認された．

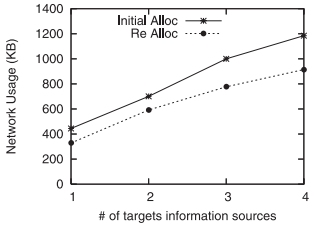


図 24　対象情報源数
Fig. 24　# of target sources.

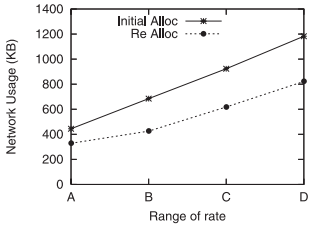


図 25　データレートの変動
Fig. 25　Changes of data rate.

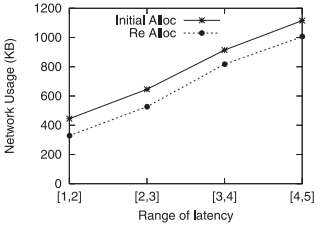


図 26　通信遅延の変動
Fig. 26　Changes of latency.

複数の TS-Join 演算を含む問合せ要求を用いた実験

図 18 の複数の TS-Join 演算を含む問合せに対して，実験 1，2，3，4 と同様の状況において算出した初期演算配置と再演算配置のネットワーク使用量は**図 28**，**図 29**，**図 30**，**図 31** である．それぞれの実験結果より，TS-Join 演算が複数ある問合せにおいても再演算配置に

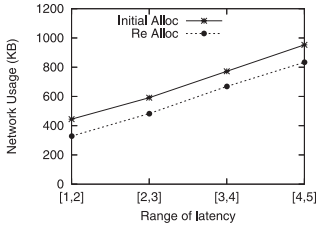


図 27　対象情報源のみ遅延変動
Fig. 27　Changes of latency between targets and nodes.

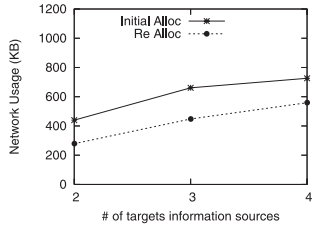


図 28　対象情報源数
Fig. 28　# of target sources.

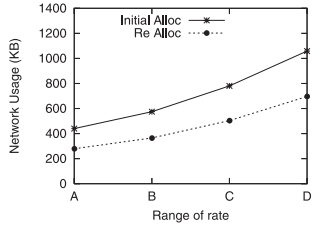


図 29　データレートの変動
Fig. 29　Changes of data rate.

よりネットワーク使用量を改善していることが確認された．

5.2.2　実環境上での実験

次に，分散ストリーム処理管理システムが実環境上において実際にネットワーク使用量を削減することを検証する．

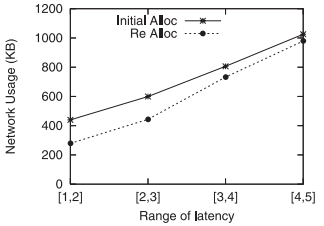


図 30　通信遅延の変動
Fig. 30　Changes of latency.

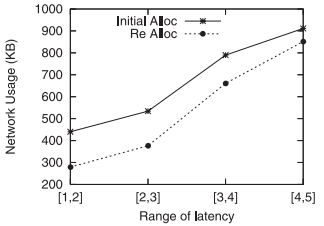


図 31　対象情報源のみ遅延変動
Fig. 31　Changes of latency between targets and nodes.

実 験 環 境

外的要因による影響を避けるために，LAN 内に**図 32** の実験環境を構築した．ノード間にネットワーク遅延の違いを生じさせるために，ネットワーク負荷を調節するマシンが，指定したサイズのネットワークパケットを送信する．通信遅延は，あるノードからネットワーク内の別の IP アドレスに対するパケットの送信完了までの時間であり，分散管理ストリーム処理システムがそれぞれのリンク間の通信遅延を収集する．

実　　験

実験の流れと実験結果について述べる．実験は，ネットワーク負荷を高めた状態において分散ストリーム処理管理システムが行うネットワーク使用量の最適化の検証を行った．ネットワークカメラのデータレートは 194（KB/s），位置情報のデータレートは 18（B/s）であり，カメラ位置 DB には 604 バイトのデータが蓄積されている．実験では，実験開始から 10 秒後においてノード 1 に TS-Join 演算が配布され，ネットワークカメラ 1 の映像の処理を行った．それから，対象情報源がネットワークカメラ 2 に切り替わったときに，分散

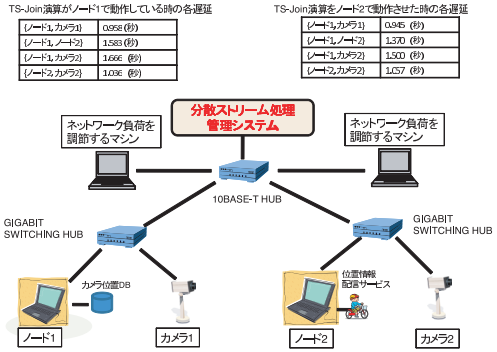


図 32　ネットワーク構成
Fig. 32　Network topology.

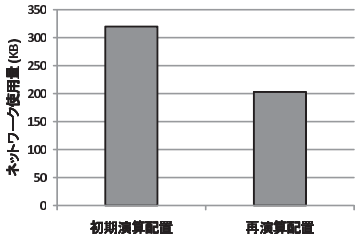


図 33　実環境上での実験結果
Fig. 33　Experimental result in real data.

ストリーム処理管理システムが演算配置最適化を行い，TS-Join 演算をノード 2 に移した．TS-Join 演算がノード 1 で動作しているときとノード 2 で動作したときの各遅延は図 32 であり，初期演算配置と再演算配置のネットワーク使用量を比較した結果は**図 33** である．実験結果より最適化を行うことで約 120 KB のネットワーク使用量を少なくしており，分散管理システムが実環境においてもネットワーク使用量の削減に貢献できることが確認された．

また，ネットワーク負荷を高めた状態は，5.2.1 項のシミュレーション実験の基本となる組合せと似た状況であり，お互いの実験結果も近似している．このことから，実環境においてもシミュレーション実験で示したような様々な状況下において分散ストリーム処理管理システムがネットワーク使用量を最適化することが見込まれる．

5.2.1 項のシミュレーション環境により，対象情報源の変化する分散ストリーム処理環境上の様々な状況下において，定期的に演算の最適な配置を計算することで分散ストリーム処理環境のネットワーク使用量を最適化することが可能であることを明らかにした．また，5.2.2 項の実環境の実験により，実環境の実運用においてもネットワーク使用量を最適化することが可能であることを検証した．これらより，ネットワーク使用量の面において分散ストリーム処理管理システムの有効性を示した．

6. 関 連 研 究

単独マシン上でストリームデータを連続的に処理するストリーム処理エンジンには Aurora[3]，TelegraphCQ[5]，STREAM[7] がある．これらのストリーム処理エンジンでは，あらかじめ指定された情報源からのストリームデータのみを連続的に処理をする．分散ストリーム処理の研究には，Borealis[4]，Medusa[6] がある．利用者からの問合せ要求はリレーショナル代数演算のリンクとして表され，各演算が任意のストリーム処理エンジン上に配置され互いに連結しながら問合せ処理が行われる．これらストリーム処理エンジンに関する研究では，利用者が処理をしてほしい情報源が状況の変化により移り変わるような問合せ要求が多々存在するにもかかわらず，本研究で扱ったような対象情報源を動的に選択する機能を有していない．

分散ストリーム処理環境の各ストリーム処理エンジンへ割り当てる演算の最適な配置決定手法には，通信遅延とデータレートから算出されるネットワーク使用量を最小化するための演算配置を算出する手法[8]，ストリーム処理エンジンが動作している各マシン間の負荷のばらつきを最小化することを目的とした手法[10]，標準演算に加えて，アプリケーションも考慮した最適な演算配置を決定する手法[11] などがある．我々が提案した演算配置最適化手法は文献 8) の手法と同じく，分散ストリーム処理環境のネットワーク使用量を最小化することを目的としているが，対象情報源が動的に切り替わる分散ストリーム処理環境上での演算の配置最適化を行う点が大きく異なる．

問合せ処理中に対象情報源を動的に選択する技術に関連する研究としては，FISQL[9] と呼ばれる問合せ言語と処理系がある．これは，データベース中のスキーマ情報を問合せ処理の中で参照できるようにすることで，異なるスキーマで保管された複数データベースを同一スキーマに変換して処理を行うことができる．しかし，ストリーム処理を想定した研究となっていない．

7. 結　　論

本研究では，対象情報源を動的に探索可能なストリーム処理に関する研究を行った．本論文では，単独ノードを利用した情報源の動的選択に関する研究と，複数ノードを利用した情報源の動的選択に関する研究について述べた．

単独ノードを利用した情報源の動的選択に関する研究では，ストリーム処理エンジン上で情報源の動的選択のための機能を実現した．そして，情報源の動的選択機能を有したストリーム処理エンジンが，大規模なストリームデータ配信環境においても必要なストリームデータを動的に選択することで，システムの負荷を抑えることが可能であると評価実験により確認した．

複数ノードを利用した情報源の動的選択に関する研究では，分散ストリーム処理において，対象情報源の変化に合わせてネットワーク使用量を最適化する分散ストリーム処理管理システムを構築した．様々な分散ストリーム処理環境の状況下において，分散ストリーム処理管理システムがネットワーク使用量を最適化していることを評価実験によって確認した．また，分散ストリーム処理管理システムを利用した広域分散環境上での疑似データを用いた動作確認も行った．

今後の課題は，ネットワーク使用量の計測の高精度化，アプリケーションに応じた演算配置処理の動的な実行タイミングの調整，分散ストリーム処理環境の状況を視覚的に表すインタフェースの作成，そして，演算配置最適化手法の計算量を減らすための高速化があげられる．

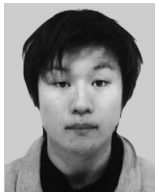
謝辞　本研究の一部は科学研究費補助金基盤研究（A）（# 18200005），科学技術振興機構 CREST「自律連合型基盤システムの構築」による．

参 考 文 献

1) Ayad, A. and Naughton, J.F.: Static Optimization of Conjunctive Queries with Sliding Windows Over Infinite Streams, *SIGMOD Conference 2004*, pp.419–430 (2004).
2) Arasu, A., Babu, S. and Widom, J.: The CQL Continuous Query Language: Semantic Foundations and Query Execution, *VLDB Journal*, Vol.15, No.2, pp.121–142 (2006).
3) Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N. and Zdonik, S.: Aurora: A new model and architecture for data stream management, *VLDB Journal*, Vol.12, No.2, pp.120–139 (2003).
4) Abadi, D., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J., Lindner, W., Maskey, A.S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y. and Zdonik, S.: The Design of the Borealis Stream Processing Engine, *Proc. CIDR* (2005).
5) Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F. and Shah, M.: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World, *Proc. CIDR* (2003).
6) Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y. and Zdonik, S.: Scalable Distributed Stream Processing *Proc. CIDR* (2003).
7) Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G., Olston, C., Rosenstein, J. and Varma, R.: Query Processing, Resource Management, and Approximation in a Data Stream Management System, *Proc. CIDR* (2003).
8) Pietzuch, P., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M. and Seltzer, M.: Network-Aware Operator Placement for StreamProcessing Systems, *Proc. ICDE*, p.49 (2006).
9) Wyss, C.M. and Wyss, F.I.: Extending Relational Query Optimization to Dynamic Schemas for Information Integration In Multidatabases, *SIGMOD' 07* (2007).
10) Xing, Y., et al.: Dynamic Load Distribution in the Borealis StreamProcessor, *Proc. ICDE*, pp.791–802 (2005).
11) 稲守孝之，渡辺陽介，北川博之，天笠俊之，川島英之：分散ストリーム処理環境におけるアプリケーション配置最適化手法，夏のデータベースワークショップ DBWS2007，2007 年電子情報通信学会技術研究報告，Vol.107, No.131, pp.345–350 (2007).
12) Darwin Streaming Server. http://www.apple.com/jp/quicktime/streamingserver
13) InTrigger プラットフォーム．https://www.logos.ic.i.u-tokyo.ac.jp/intrigger/registration/
14) jconsole. http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/management/jconsole.html
15) JMF. http://java.sun.com/products/java-media/jmf/
16) mobireal. http://www.mobireal.net/
17) StreamSpinner. http://www.streamspinner.org
18) QuickTime Player. http://www.apple.com/jp/quicktime/
19) 情報爆発プロジェクト．http://www.infoplosion.nii.ac.jp/info-plosion/
20) ドコモ・システムズ株式会社：登下校情報連絡サービス Kids in Feel. http://www.docomo-sys.co.jp/products/kidsinfeel/

(平成 21 年 3 月 20 日受付)
(平成 21 年 6 月 8 日採録)

(担当編集委員　安形 輝)



大喜　恒甫（正会員）

2007 年早稲田大学人間科学部卒業．2009 年筑波大学大学院システム情報工学研究科修了．StreamSpinner プロジェクトにおける研究・開発に従事．修士（工学）．現在は日本電気（株）に勤務．日本データベース学会学生会員．

渡辺　陽介（正会員）

2001 年筑波大学第三学群情報学類卒業．2006 年同大学大学院博士課程システム情報工学研究科修了．博士（工学）．科学技術振興機構 CREST 研究員の後，2008 年より東京工業大学学術国際情報センター助教として，ストリームデータ処理，デスクトップ検索等の研究活動に従事．日本データベース学会，ACM 各会員．

北川　博之（フェロー）

1978 年東京大学理学部物理学科卒業．1980 年同大学大学院理学系研究科修士課程修了．日本電気（株）勤務の後，1988 年筑波大学電子・情報工学系講師．同助教授を経て，現在，筑波大学大学院システム情報工学研究科教授，ならびに計算科学研究センター教授．理学博士（東京大学）．異種情報源統合，XML とデータベース，データマイニング，センサデータベース，WWW データ管理等の研究に従事．著書『データベースシステム』（昭晃堂），『The Unnormalized Relational Data Model』（共著，Springer-Verlag）等．日本データベース学会理事，情報処理学会フェロー，電子情報通信学会フェロー，ACM，IEEE-CS，日本ソフトウェア科学会各会員．

川島　英之（正会員）

2005 年慶應義塾大学大学院理工学研究科開放環境科学専攻後期博士課程修了．同年慶應義塾大学理工学部助手．2007 年筑波大学大学院システム情報工学研究科講師，ならびに計算科学研究センター講師．博士（工学）．センサデータ管理に関する研究に従事．ACM 会員．