2008－DBS－146 (45)
2008／9／23

対象情報源の動的変化を考慮した分散ストリーム処理最適化手法の提案

大喜 恒甫[†] 渡辺 陽介[††] 北川 博之[†,†††] 天笠 俊之[†,†††] 川島 英之[†,†††]

† 筑波大学システム情報工学研究科 〒305–8573 茨城県つくば市天王台 1-1-1
†† 東京工業大学 学術国際情報センター 〒152–8550 東京都目黒区大岡山 2-12-1
††† 筑波大学計算科学研究センター
E-mail: †ohki@kde.cs.tsukuba.ac.jp, ††watanabe@de.cs.titech.ac.jp,
†††{kitagawa,amagasa,kawasima}@cs.tsukuba.ac.jp

あらまし 近年、情報源から連続して配信されるストリームデータに対する処理要求が増加しており、これら処理要求を効率的に処理するためのストリーム処理エンジンが注目を集めている。ストリーム処理エンジンでは利用者からの問合せ中に明記された対象情報源からのストリームデータが到着する度に連続して問合せ処理結果を生成する。これまで我々は、同じ問合せ中で利用者の興味のある対象情報源を動的に選択可能な枠組みをストリーム処理エンジンに追加した。さらに多くの利用者の要求に応じるために、本研究では各ストリーム処理エンジンで問合せ処理を分散して行う分散ストリーム処理を考える。利用者の興味のある情報源が変わると、情報源から利用者へのデータ転送経路が変わるため、それに合わせて分散環境中の各ストリーム処理エンジンが行う処理の配置を動的に組み替えていかなければならない。本稿では、対象情報源の変化を考慮した分散ストリーム処理における最適化手法を提案する。最適化手法では利用者の興味に合わせた最適なデータ転送経路および処理の配置を繰り返し計算する。また、最適化処理を行う中央管理システムのアーキテクチャについても述べる。

キーワード データストリーム、連続的問合せ、複数問合せ最適化、情報源の動的選択

An Optimization Method for Distributed Stream Processing with Dynamic Selection of Target Information Sources

Kosuke OHKI[†], Yousuke WATANABE[††], Hiroyuki KITAGAWA[†,†††], Toshiyuki AMAGASA[†,†††], and Hideyuki KAWASHIMA[†,†††]

† Graduate School of Systems and Information Engineering, University of Tsukuba
1-1-1 Tennoudai, Tsukuba-shi, 305-8573, Japan
†† Tokyo Institute of Technology Global Scientific Information and Computing Center
2-12-1 Ookayama, Meguro-ku, 152-8550, Japan
††† Center for Computational Sciences, University of Tsukuba
E-mail: †ohki@kde.cs.tsukuba.ac.jp, ††watanabe@de.cs.titech.ac.jp,
†††{kitagawa,amagasa,kawasima}@cs.tsukuba.ac.jp

Abstract A demand for query processing to stream data which is continuously delivered form information sources has been increasing. Stream processing engines process stream data efficiently. Users can receive processing results from the stream processing system when they register their queries into the system. We have already developed a new framework in which target information sources that users are interested in can be dynamically selected. To process many users' requests, we study a distributed stream processing schema. In distributed environments, we plan to minimize a network usage. However, the network usage dynamically changes. In this paper, we propose an optimization method for distributed stream processing considering users' interests. The optimization method calculates the optimal data flow dynamically. We also introduce a central control system for the optimization method.

Key words data stream, continuous query, multiple query optimization, dynamic source selection

1. まえがき

近年、デバイス技術やネットワーク技術の普及に伴い、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、情報配信サービスや、実世界からのセンサーからの温度データ、映像データ、音声データなどがある。ストリームデータに対する問合せ要求としては、フィルタリングや異なる情報源との統合利用などの連続的ストリーム処理が挙げられる。これらの処理を行うアプリケーションを利用者が一から作ることは困難である。そこで、ストリームデータに対する問合せ処理の基盤技術としてストリーム処理エンジン [1] があり、我々の研究室においても StreamSpinner [4], [10] というストリーム処理エンジンを研究・開発中である。

ストリーム処理エンジンでは、利用者が問合せ記述中に指定した情報源からストリームデータが到着する度に繰り返し演算評価を行う連続的問合せ方式 [3] を用いている。さらに、地理的に分散した情報源やデータベースを効率的に扱うために、複数のストリーム処理エンジンで利用者の問合せを分担して処理をする分散ストリーム処理 [2] もある。

一方で、利用者の興味に合わせて対象情報源を動的に選択したいという問合せ要求が高まっている。例えば、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を追跡したいという要求があるとする。このとき、位置情報を用いて特定人物の最寄りのカメラからの映像のみを取得することになるが、人物は移動するので、その位置情報の変化に伴って接続するカメラを連続的に切り替えることが必要となる。そこでこのようなアプリケーションの構築を支援するため、我々は興味のある情報源を動的に選択する仕組みをストリーム処理エンジンに追加した [9]。さらに多くの問合せ要求に応えるために、分散環境での処理が必要となる。

本研究では、利用者の興味に合わせて対象情報源を動的に選択したいという要求を効率的に処理するための分散ストリーム処理の枠組みを提案する。ここでは、必要最小限なネットワーク帯域使用をすることを目指す。ネットワークを有効に利用することで、多くの利用者の問合せ要求に応じることが可能となる。分散環境で問合せ処理を実行する中で、利用者の興味の移り変わりに伴い、対象情報源から利用者へのデータ転送経路が変わり、ネットワーク帯域使用量を最適にするデータ転送経路も変化する。例えば、先の特定人物を追跡したいという要求では、人物の移動により処理対象となるカメラが変われば、ネットワーク上の最も近いストリーム処理エンジンも変化する。このような場合、映像取得や解析処理を行うストリーム処理エンジンを変化に合せて繰り返し変更していけば、最適なネットワーク帯域使用量を維持できると考えられる。

また、複数利用者の問合せ要求の処理においては、利用者同士の興味のある情報源が共通であった場合、情報源への接続を共有することが可能となる。人物追跡の例で考えると、利用者 A が targetA への追跡要求を、利用者 B が targetB への追跡要求を登録しているとする。ある時点において、targetA と targetB が同じ位置にいる時には、同じカメラからの映像取得を共有化することでネットワーク帯域使用の削減が可能となる。

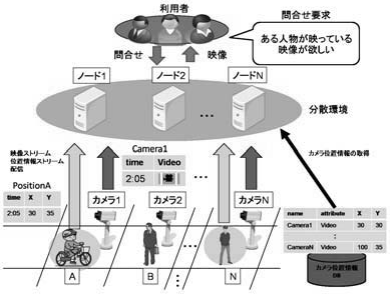

図 1　人物追跡アプリケーション
Fig. 1　Tracking application for moving objects

そこで、上記のような状況を考慮し、ネットワーク帯域使用量を最小にするための最適なデータ転送経路を作り出し、各ストリーム処理エンジンに処理を指示する中央管理システムを構築する。

中央管理システムは問合せ要求に対して、以下の処理手順をとる。

（1） 利用者からの問合せを解析し、分散環境の各ストリーム処理エンジンに処理配置の指示する

（2） 利用者の興味のある情報源についての情報を収集する

（3） 利用者の興味を考慮した最適な処理の配置及びデータ転送経路を計算し、ネットワーク帯域使用量の見積りが分散環境で現在実行処理中のネットワーク帯域使用量よりも少なくなっていれば、処理配置の変更を指示する

本稿の構成は以下の通りである。まず、2. で利用者の興味のある情報源が移り変わるアプリケーションについて説明する。そして、3. では、対象情報源の移り変わりを考慮した分散ストリーム処理を考え、各ストリーム処理エンジンをまとめる中央管理システムのアーキテクチャについて述べる。4. では各ストリーム処理エンジンに連結処理を指示する方法について述べる。5. で関連研究について述べ、6. でまとめと今後の課題について述べる。

2. 人物追跡アプリケーション

利用者の興味のある情報源が移り変わるアプリケーションの一例として、人物追跡アプリケーションを考える 1。大規模なネットワークカメラが整備され、各ネットワークカメラの名前と位置情報等が登録されたカメラ位置データベースが準備されている環境において、位置情報を発するセンサが付けられている特定人物を追跡したいという要求である。

この要求において、利用者の興味のある対象情報源は追跡対象の最寄りのカメラ群である。追跡対象の移動に伴い、興味のあるカメラは移り変わる。この要求を満たすためには、興味のあるカメラに関する情報を探索し、そのカメラに接続を確立し

て映像を処理する必要がある。興味のあるカメラに関する探索は、追跡対象の位置情報とカメラ位置データベース内の各カメラの位置情報から距離を算出することで実現される。そして、興味のあるカメラの情報を基に、そのカメラに接続を行い、映像データを処理をする。このようにして、人物追跡の要求を満たすことが出来る。

人物追跡アプリケーションのような利用者の興味のある情報源が移り変わる問合せ要求を処理するために、我々の研究室で開発中のストリーム処理エンジンである StreamSpinner が利用できる。利用者の問合せ要求を効率的に処理するために各 StreamSpinner を連携させて問合せ要求を満たす分散ストリーム処理を考え、ネットワーク帯域使用量を最適にするデータ転送経路の利用を目指す。ここで、StreamSpinner が動作しているマシンをノードと呼ぶ。分散環境において人物追跡アプリけーションを処理する際に、二点について考慮する必要がある。

まず、最適なデータ転送経路が人物の移動に伴い変化する点である。例えば、人物 A の追跡要求が登録されているときに、A の近くにあるカメラ 1 にネットワーク上で近いノード 1 で接続を行い、利用者に配信するデータ転送経路が最適である。しかし、A がカメラ N の付近に移動した時に、ノード 1 でカメラ N に接続を行い映像データを取得するよりも、ノード N でカメラ N に接続をして映像データを取得する方が、ネットワーク帯域利用量において最適なデータ転送経路となる。そのため、人物の移動に伴い映像取得や解析処理を行うノードを変化して、最適なデータ転送経路を維持する必要がある。

次に、カメラへの接続を共有することによるネットワーク帯域使用量の削減が可能となる点である。例えば、ある利用者が A の追跡を、別の利用者が B の追跡を問合せ要求で登録しているとする。この時、A がカメラ 2 の付近に移動をし、B が未だカメラ 2 の付近に留まっていたならば、カメラ 2 への接続を同じノードで共有し、映像データを共有することで、ネットワーク帯域使用量を削減することが出来る。

上記、二点を考慮し、最適なデータ転送経路を算出する枠組みについて次の節で紹介する。

3. 対象情報源の動的変化を考慮した分散ストリーム処理

本節では、利用者が注目する情報源の変化に対応して、最適なネットワーク帯域使用量を維持するための中央管理システムのアーキテクチャについて述べる。中央管理システムでは、現在の利用者の注目する対象情報源を考慮した最適なデータ転送経路計算し、分散環境のストリーム処理エンジンに配置する処理を決定する。まず、本研究が前提とするストリーム処理エンジンである StreamSpinner について説明し、次に [9] で提案した単一ノード上での対象情報源の切り替えのための仕組みを説明する。そして、分散環境上での対象情報源の変化に合わせた動的な処理配置最適化のための指標とするネットワーク帯域使用量について話し、各 StreamSpinner に接続・処理要求を出す中央管理システムのアーキテクチャを紹介する。

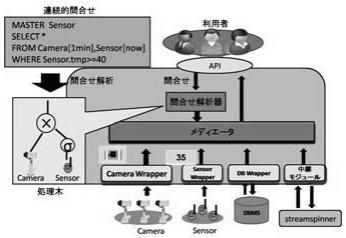

図 2　StreamSpinner のアーキテクチャ
Fig. 2　Architecture of StreamSpinner

3.1 StreamSpinner

StreamSpinner のアーキテクチャは図 2 である。ラッパーは、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内部形式に変換して、メディエータに送る役割を持つ。利用者からの問合せは SQL ライクな形式で記述される。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度に問合せ処理が繰り返し行われる。FROM 節の [now],[1min] はウィンドウ幅と呼ばれるものであり、処理を行う期間を表している。問合せ解析器では、問合せが処理木と呼ばれる処理の流れを表すグラフ構造に置き換えられ、メディエータに送られる。処理木は対象情報源と演算から構成される。メディエータでは処理木に基づいて、ラッパーから受け取ったデータに対して、問合せ処理を連続的に行う。

図 2 の問合せは温度センサの値が 40 度以上の時にカメラからの最新の一分間の映像を取得したいという異種情報源統合利用の要求である。

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 アドレスと処理結果の型に関する情報と問合せ要求が必要となる。連結先の StreamSpinner からの処理結果はストリームデータとしてメディエータで処理される。

3.2 対象情報源の動的選択機能

本研究では、利用者が処理対象にしたい情報源が時間とともに変化する状況を扱う。ここでは、ある時刻 t における利用者が注目している対象情報源の名前の集合 $S_1, \ldots, S_n$ を時刻 t での利用者の興味と呼称する。[9] では、利用者の興味のある対象情報源の動的な選択のために 2 点の機能を StreamSpinner に追加した。

- 利用者の興味の情報を入力データとして，対象情報源を選択的に切り替える ASSIGN 演算
- ASSIGN 演算が必要とする情報源にのみ接続を行う接続管理機能

利用者の興味のある対象情報源が移り変わるような問合せ要求に対して、次のような処理手順で要求に応える。具体例として 1 の例を用いる。

（1） 利用者の興味のある情報源を探索する。追跡人物の最

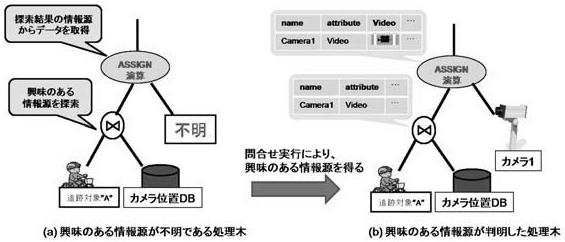

図 3　利用者の興味により変化する処理木
Fig. 3　Variable DAG according to user's interest

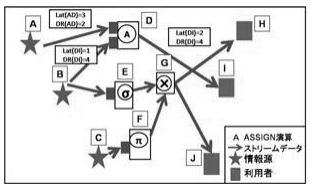

図 4　多ノードを利用した複数の利用者による問合せ
Fig. 4　Multiple queries in multiple nodes

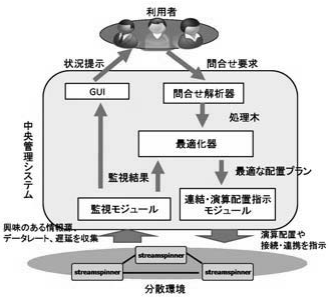

図 5　中央管理システムのアーキテクチャ
Fig. 5　Architecture of the central control system

寄りのカメラの探索にあたる。

（2） 接続管理機能が探索から得られた興味のある情報源に対する接続を確立する。追跡人物の最寄のカメラの接続確立にあたる。

（3） ASSIGN 演算でその情報源からのデータを処理する。カメラからの映像データを取得し、利用者への配信にあたる。

（4） 接続管理機能が興味の対象外となった情報源への接続を解除する。追跡人物がカメラから遠ざかった時に、そのカメラへの接続を解除する処理にあたる。

以下でそれぞれの詳細について説明をする。

3.2.1　ASSIGN 演算

ASSIGN 演算は以下のように定義されている。

$$ASSIGN_{R.A_i,\{R.A_1,\ldots,R.A_j\}}(R) = \bigcup_{r\in R}\{r \times t | t \in \pi_{attr(r.A_1,\ldots,r.A_j)}(rel(r.A_i))\} \quad (1)$$

ASSIGN 演算は、利用者興味を格納した各入力タプル r に対して、属性 r.Ai の値に一致する名前のリレーション（relation(r.Ai)）を取得し、属性 r.A1,…r.Aj の値に一致する属性名だけを残すように射影した後に、入力タプル r との直積を出力する。一致するリレーション名や属性名が存在しなかった場合は、空集合を返すものとする。

具体的に人物追跡アプリケーションを用いて説明する。A を追跡するには図 3 の処理木に表わされる処理の流れとなる。問合せ登録時には、最寄のカメラは不明である。しかし、問合せを実行すると A の位置情報とカメラ位置 DB の各カメラの位置情報の結合演算より最寄りのカメラ 1 を探索できる。ASSIGN 演算では結合演算の処理結果を入力として受け取り、カメラ 1 に接続を行い、属性 Video を取得して入力データに負荷して利用者に配信する。

3.2.2　接 続 管 理

ASSIGN 演算により興味のある情報源からのストリームデータを取得する前に、情報源への接続が確立されている必要がある。そこで、情報源への接続を動的に管理する接続管理機能が導入された。接続管理機能には、ASSIGN 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がされた問合せを与える。接続管理機能はその処理結果から情報源への接続を確立する。これにより、ASSIGN 演算が処理をする前に、情報源への接続が確立できる。ASSIGN 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への接続を解除する。

3.3　ネットワーク帯域使用量

次に、分散環境における最適化の指標とするネットワーク帯域使用量に関して説明する。本稿では問合せ q に対するネットワーク帯域使用量 u(q) を、[5] で用いられた式に基づいて算出する

$$u(q) = \Sigma_{e\in E} DR(e)Lat(e) \quad (2)$$

ここで、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、ネットワーク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。

ネットワーク帯域使用量について図 4 を用いて具体的に説明する。図 4 には地理的に分散された複数のノードが配置されている。左側の A,B,C が情報源を表している。中間の D,E,F,G のノードには入力データを処理するための演算が配置されている。右側の H,I,J が問合せを行う利用者を表している。ストリームデータの流れを矢印の方向で表し、ASSIGN 演算は便宜上 A と表記している。ここで、利用者 I の問合せは A,B,D,I の処理木として表される。そのため、ネットワーク帯域使用量は DR(AD)Lat(AD)+DR(BD)Lat(BD)+DR(DI)Lat(DI)= 3*2+1*4+2*3=18(byte) で表される。

複数の問合せのネットワーク帯域使用量の合計を求める場合はすべての問合せのリンクに対して、同様の手順を用いることで算出される。

3.4　システムアーキテクチャ

中央管理システムは、利用者の興味、データレート、遅延を収集して、各 StreamSpinner に連携・演算配置を指示する機能

を有する。中央管理システムのアーキテクチャは図 5 である。特に重要な役割を持つのが最適化器である。最適化器は、ネットワーク帯域使用量を最小化する演算の初期配置、再配置を行う。詳細な処理の流れについては次の節で説明する。各機能の説明を下に列挙する。

- 問合せ解析器: 利用者からの問合せを解析し、処理木を作成する。処理木を作成する時に、ASSIGN 演算への入力を減らす処理を可能な限り先に行う配置を考える。なぜなら、ASSIGN 演算で興味のある情報源を選択する前に、対象となる情報源を絞りこむことで取り扱うストリームデータを減らすことでシステム負荷を減らすことができるからである。
- 最適化器: 最適化器はネットワーク帯域使用量を最小化するような最適な演算配置のプランを算出する。演算配置指摘化処理は、問合せを分散環境に登録する初期配置最適化と、問合せを実行し、定期的に利用者の興味を収集して現在の最適配置プランを再計算する再配置最適化からなる。
- 連結・演算配置指示モジュール: 最適化器から受け取る演算の最適配置プランを基に、各ストリーム処理エンジンの連結や演算配置を指示する。
- 監視モジュール: 監視モジュールは分散環境の各ストリーム処理エンジンから、実行中の問合せ情報、利用者が注目する対象情報源に関する情報、データレート、遅延の収集管理を行う。収集した情報は GUI 表示や演算配置最適化のために利用される。
- GUI: GUI は分散環境の各ストリーム処理エンジン同士の連結や演算の配置状況を視覚的に利用者に伝える。

4. 最適な演算配置法

分散環境において、各演算をネットワーク帯域使用量が最小になるような演算配置プランを算出する手法 [5] が既に提案されている。しかし、利用者の興味に合わせて対象情報源が移り変わる状況は考慮されていない。そこで、既存手法を基に対象情報源の変化を考慮した最適な演算配置を行う手法を提案する。これは、演算の初期配置最適化手法と再配置最適化処理から成る。初期配置最適化処理では、問合せ処理を実行する前段階であるので、利用者の興味のある情報源が不明である。そこで、ASSIGN 演算を含む処理木を既存の最適化手法で扱えるようにするために、動的に変わりうる部分に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、ネットワーク遅延はすべてのノード間のネットワーク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、既存手法を利用することが出来る。

再配置最適化処理では、実行中のそれぞれの問合せに対して、現在の興味の情報に基づいて、処理木上で対象情報源と ASSIGN 演算をリンクさせる。再構成した処理木を従来手法を用いて最適なプランを算出する。この時に、利用者同士の興味の一致により、同一のノードから共通の情報源への接続を行う場合、そのリンクは共有化できるとみなしてネットワーク使用帯域使用量を算出する。よって、接続の共有が出来るプランが有利となる。算出された最適な演算配置プランにおけるネットワーク帯域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帯域使用量よりも小さければ、実際に演算を再配置する。以下、詳細なアルゴリズムを載せる。

Input :
- G : a set of query graphs $\{ Q_1(V_1, E_1), ..., Q_i(V_i, E_i)\}$
- V_i : a set of operators (vertices of graph) in query Q_i
- E_i : a set of edges in query Q_i
- N : a set of available nodes

```
01. INITIAL_PLACEMENT(G, N, L,D )
02. FOR(Q_i(V_i, E_i) ∈ G)
03.   FOR(Operator ∈ V_i)
04.     IF (Operator equals "ASSIGN")
05.       V_i ← V_i ∪ {I_dummy}
06.       E_i ← E_i ∪ {I_dummy ← ASSIGN}
07.     END IF
08.   END FOR
09. END FOR
10. OptimalPlan := Calculate( G, N, L, D )
11. AllocOperators( OptimalPlan)
```

図 6 初期配置のアルゴリズム
Fig. 6 Algorithm for initial location

4.1 初 期 配 置

初期配置のアルゴリズムは図 6 である。複数の処理木 (G)、分散環境中のストリーム処理エンジンが動作しているマシン (Nodes)、一定期間サンプリングしていたデータレート・遅延の平均 (D,L) を利用して、最適な演算配置プランを作成するアルゴリズムである。2-9 行目で複数の処理木の中から ASSIGN 演算を探し、ASSIGN 演算にダミーの情報源を接続する。10 行目の Calculate 関数では、ネットワーク帯域使用量を最小化する既存手法 [5] を用いて最適な演算配置を決める。11 行目の AllocOperators 関数により、求められた演算の最適な配置を基に、分散環境の各ストリーム処理エンジンに配置する。

4.2 再 配 置

再配置のアルゴリズムは図 7 である。初期配置と同様の情報に加えて利用者の興味（Interests）を基に、一定時間ごとに演算配置の再評価を行い、ネットワーク帯域使用量が改善されていれば、再配置を行うアルゴリズムである。I_{on} は、n 番目の ASSIGN 演算の入力となる興味の情報を表す。例えば、I_{o1} では、演算 o1 にデータを入力する情報源を意味する。2-11 行目では、複数の処理木の中の ASSIGN 演算と対象情報源の間にリンクを追加する処理を行っている。これにより、現在の利用者の興味から処理木が再構成される。再構成された処理木は 12 行目で Caluculate 関数により、既存手法をもとに最適な演算配置を計算される。13 行目では算出された現時点の最適なプランにおけるネットワーク帯域使用量と既に配置済みのプランによるネットワーク帯域使用量の見積もり値 (PreviousNetworkUsage) を比較し、算出したプランが優れていれば、14 行目で AllocOperators 関数によって再配置が実行される。

Input :
・G : a set of query graphs { $Q_1(V_1, E_1), ..., Q_i(V_i, E_i)$}
・V_i : a set of operators (vertices of graph) in query Q_i
・E_i : a set of edges in query Q_i
・N : a set of available nodes
・Interests : a set of user interests $\{Io_1, ..., I_{on}\}$
・P : a value of networkusage in current query processing

01. REALLOCATION(G, N, L, D, Interests, P)
02. $FOR(Q_i(V_i, E_i) \in G)$
03. $FOR(Operator \in V_i)$
04. IF(Operator equals "ASSIGN")
05. $FOR(I \in (I_{operator}))$
06. $V_i \leftarrow V_i \cup I$
07. $E_i \leftarrow E_i \cup I \leftarrow ASSIGN$
08. END FOR
09. END IF
10. END FOR
11. END FOR
12. OptimalPlan := Calculate(G, N, L, D)
13. $IF(OptimalPlan.NetworkUsage < P)$
14. AllocOperators(OptimalPlan)
15. END IF

図 7 再配置のアルゴリズム
Fig. 7 Algorithm for reallocation

4.3 具体的な処理の流れ

2. で説明した人物追跡アプリケーションを用いて、具体的な最適演算配置の流れを説明する。人物 A の映っている映像を得たいという問合せを考える。初期配置の段階では、追跡対象の最寄のカメラに関する情報を得ることが出来ないので、図 3 の (a) の処理木のように、ASSIGN 演算には dummy の情報源を接続して仮の処理木を作る。仮の処理木、利用可能なノード数、データレート、遅延から既存手法を用いて最適なプランを作成し、分散環境の各ノードに演算を配置する。そして、問合せ実行を開始することにより、現在の位置情報とカメラの位置を基に利用者の興味のある情報源が中央管理システムに逐次通知される。例えば、ある時刻における対象情報源がカメラ 1 である場合，再配置最適化処理において、カメラ 1 が処理木の ASSIGN 演算の子ノードとして追加される（図 3(b)）。再構成した処理木を同様に既存手法を用いて最適なプランを作成する。この時に、作成したプランにおけるネットワーク帯域使用量 (b) と既に実行中のプランにおけるネットワーク帯域使用量 (a) を比較し、(b) が優れていた場合は (b) の最適なプランを基に演算の再配置をする。

5. 関 連 研 究

ストリームデータを効率的に取り扱うための基盤技術として Aurora [1] がある。これらは、一台のマシン上で動作する集中型のストリーム処理エンジンである。ストリーム処理エンジンは、明記された情報源からのストリームデータのみを連続して処理をする。しかし、我々のストリーム処理エンジンでは問合せ中の対象情報源を動的に探索可能なモデルとなっている。

分散型のストリーム処理エンジンには、Aurora を拡張した Borealis [2] がある。利用者からの問合せを最小処理単位である演算に置き換えて、各演算を複数のストリーム処理エンジンで動作させることで、分散ストリーム処理を行うシステムである。

また、分散ストリーム環境において、情報源やストリーム処理エンジン間のデータレートや遅延を基に、ネットワーク帯域使用量を最小に抑えるために、各演算を適切なノードに配置するための手法 [5] やアプリケーションは通常固定して動作する点に着目をし、他の演算の配置を最適化する手法 [8] や演算の移動を考えずに、各ノードの負荷を平均化することで、問合せ処理を可能とする手法 [7] などが提案されている。しかし、対象情報源の切り替わりを考慮した演算配置最適化は考慮されていない。

6. む す び

本研究では、各ストリーム処理エンジンから利用者の興味のある情報源に関する情報を収集し、情報源への接続と演算配置の指示を行う中央管理システムを紹介した。利用者の興味に伴いネットワーク帯域使用量が変わるので、最適となるデータ転送経路を維持するために最適な演算配置方法を提案した。

今後は、複数のストリーム処理エンジンをまとめるシステムの実装を行い、性能評価をする予定である。また、各ストリーム処理エンジンのマシン負荷も考慮した最適な演算配置のための機能も追加したい。

謝辞 本研究の一部は科学研究費補助金特定領域研究（# 19024006），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.

文 献

[1] D. Abadi, et al. "Aurora: A New Model and Architecture For Data Stream Management", VLDB Journal Vol.12,No2,pp.120-139,2003.
[2] D. Abadi, et al. "The Design Of The Borealis Stream Processing Engine", In Proc. of the Conference on Innovative Data Systems Research, 2005.
[3] A. Arasu, et al. "The Cql Continuous Query Language: SeMantic Foundations and Query Execution". VLDB Journal, Vol. 15, No. 2, pp.121-142, 2006.
[4] http://www.streamspinner.org/
[5] P.Pietzuch, et al."Network-Aware Operator Placement for StreamProcessing Systems", Proc. ICDE, p.49, 2006.
[6] C. Wyss, et al. "Extending Relational Query Optimization to Dynamic Schemas for Information Integration In Multidatabases",SIGMOD' 07, 2007.
[7] Y. Xing, et al. " Dynamic Load Distribution in the Borealis StreamProcessor", Proc. ICDE, pp. 791-802, 2005.
[8] 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", 電子情報通信学会技術研究報告 Vol. 107, No. 131, pp. 345-350.
[9] 大喜恒甫, 渡辺陽介, 北川博之, 天笠俊之, 川島英之. "ストリーム処理における情報源の動的選択機能の評価", 電子情報通信学会第 19 回データ工学ワークショップ (DEWS2008)
[10] 渡辺陽介, 北川博之. "連続的問合せに対する複数問合せ最適化手法", 電子情報通信学会論文誌, Vol.J87-D-I, No.10, pp.873-886, 2004 年 10 月.