分散ストリーム処理における対象情報源の動的変化を考慮した問合せ最適化手法の評価

大喜　恒甫† 　渡辺　陽介†† 　北川　博之†,††† 　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,kawasima}@cs.tsukuba.ac.jp

あらまし　近年，広域に配置されたネットワークカメラやセンサなどの各種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，これらに対する効率的な処理をする分散ストリーム処理に関する研究が盛んに行われてきた．本研究では，我々の研究グループで研究・開発をしているストリーム処理エンジン StreamSpinner を利用し，問合せ処理中に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，対象情報源の切り替わりに伴い，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．本稿では，分散ストリーム処理管理システムの最適化の有効性を検証する．

キーワード　データストリーム，連続的問合せ，分散データストリーム処理，情報源の動的選択

Evaluation of a Query Optimization Method for Distributed Stream Processing with Dynamic Selection of Target Information Sources

Kosuke OHKI†, Yousuke WATANABE††, Hiroyuki KITAGAWA†,†††, and Hideyuki KAWASHIMA†,†††

† Graduate School of Systems and Information Engineering, University of Tsukuba
1-1-1 Tennoudai, Tsukuba-shi, 305-8573, Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology
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,kawasima}@cs.tsukuba.ac.jp

Abstract　Stream data which is continuously delivered from information sources in wide area has been increasing. Researches of distributed stream processing to stream data has become important. Our research group focuses on distributed stream processing which can dynamically select target information sources during query processing. When target information sources change, a data transfer route and network traffic from the target information sources to users also change. We therefore proposed a management system to keep optimal network traffic in distributed stream processing with dynamic selection of target information sources. To keep traffic low, the system reconsiders operator allocations according to changes of target information sources. In this paper, we evaluate the efficiency of our distributed management system by experiment.

Key words　data stream, continuous query, distributed processing, 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，温度センサからの温度データ，位置タグ・GPS 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，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．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，ストリームデータに対する処理の基盤技術であるストリーム処理エンジン [1], [4], [6] に注目が集まっている．我々の研究グループにおいても，StreamSpinner [15] というストリーム処理エンジンを研究・開発中である．

StreamSpinner では利用者が SQL ライクな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 [13] を用いている．また，利用者の処理して欲しい処理対象の情報源を問合せ処理中に StreamSpinner が動的に選択して処理をすることも可能である．例えば，地理的に分散したネットワークカメラ，各ネットワークカメラの配置場所等を登録したカメラ位置データベース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人物の映像を監視したいという要求があるとする．この時に，人物の位置情報とカメラ位置データベースを用いて人物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が理想的な処理であるが，人物の位置情報の変化に伴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必要がある．StreamSpinner ではこのような対象情報源が変化する要求に対して，問合せ処理中に対象情報源を動的に選択する機能を有している [11].

また，広域に配置した情報源からのデータを効率的に処理するために，複数のストリーム処理エンジンが利用者からの問合せ要求を分担して処理する分散ストリーム処理に関する研究 [2], [5] が盛んに行われている．本研究では，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利用して問合せ処理中に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，対象情報源の切り替わりに伴い，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最適なネットワーク使用量を保つ分散ストリーム処理管理システムを既に論文 [12] で提案した．しかし，提案だけで分散ストリーム処理管理システムの実装とその評価がされていなかった．

本稿では，実装を完了した分散ストリーム処理管理システムの有用性を評価実験を通して検証する．評価実験では，対象情報源が変化する分散ストリーム処理環境のネットワーク使用量に影響を与える様々な状況をシミュレートし，各状況における分散ストリーム処理管理システムが最適化したネットワーク使用量の評価と，実環境上での評価実験を行う．また，広域分散環境において分散ストリーム処理監視システムの実運用も可能であることも示す．

本稿の構成は以下である．2. で本研究と関連研究の違いを説明する．そして，3. で対象情報源を動的に選択可能な分散ストリーム処理と分散ストリーム処理管理システムについて説明する．その後，4. で分散ストリーム処理システムの評価実験について述べ，5. で広域分散環境での運用を示し，6. で本稿をまとめる．

2. 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 [1]，STREAM [6]，TelegraphCQ [4] 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 [2]，Medusa [5] がある．利用者からの問合せ要求はリレーショナル代数の標準演算のリンクとしてあらわされ，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欲しいストリームデータ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関わらず，ストリーム処理エンジンが処理対象となる情報源を問合せ処理中に動的に選択する機能を有していない．一方，我々の研究グループで研究・開発中の StreamSpinner [15] では，問合せ処理中に対象情報源を動的に選択可能である [11].

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 [7]，ストリーム処理エンジンが動作している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 [9]，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 [10] などがある．我々の演算配置最適化手法は手法 [7] 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．また，他の演算配置最適化手法に関する研究においても，対象情報源が動的に切り替わる分散ストリーム処理を取り扱った研究はされていない．

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 [8] 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，対象となる情報源数が増加するとスキーマ変換のための処理コストが膨大となってしまい，ストリーム処理を想定した研究となっていない．

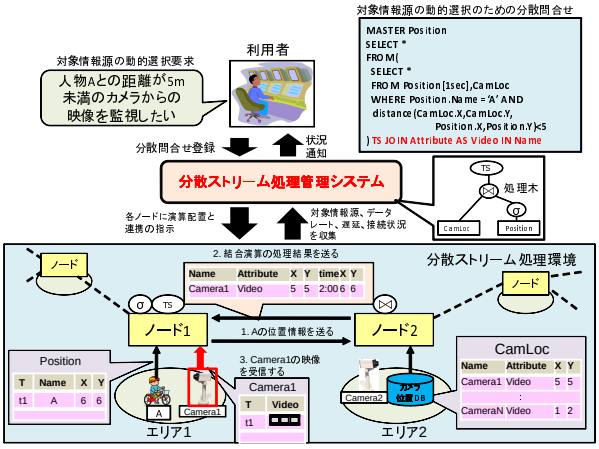

図 1　対象情報源の動的変化を考慮した分散ストリーム処理

Fig. 1　Distributed stream processing with dynamic selection of target information sources

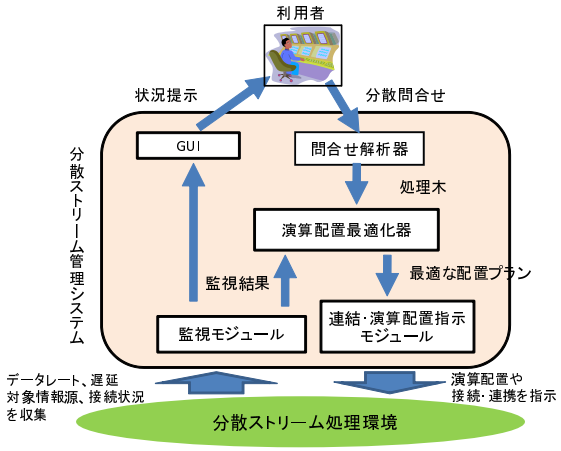

図 2　分散ストリーム処理管理システムのアーキテクチャ

Fig. 2　Architecture of distributed management system

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，分散ストリーム処理管理システムについて述べる．図 1 は概要であり，図中のノードと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．対象情報源が変化する問合せ要求として特定人物の監視要求を例に挙げている．これは，広域に配置したネットワークカメラ，各ネットワークカメラの配置場所等を登録したカメラ位置データベース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人物 A の映像を監視したいという要求である．図 1 では，ノード 1 が人物 A との距離が 5m 未満のネットワークカメラ 1 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 は動的にネットワークカメラ 2 に切り替り変えて映像受信を続けて行う．この時に，ノード 2 とネットワークカメラ 2 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 とネットワークカメラ 2 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 で行っているネットワークカメラ 2 からの映像受信をノード 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，分散ストリーム処理管理システムは，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する．

分散ストリーム処理管理システムに対象情報源の動的選択のための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，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もりを計算する．この見積もり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．

本研究で扱う対象情報源の動的選択の仕組みについては 3.1 で説明する．3.2 で分散ストリーム管理システムのアーキテクチャについて説明し，ネットワーク使用量を最小とする演算配置最適化手法に関しては，3.3 で説明する．

3.1 対象情報源の動的選択機能

問合せ処理中に対象情報源を動的に選択するために，Target Selective Join 演算を既に提案している．以降，TS-Join 演算と省略する．この演算は論文 [11] で述べられている ASSIGN 演算と本質的に同じ演算である．TS-Join 演算は，入力タプル中に記載された情報源名と属性名をもとに，対応する情報源のデータを取得するためのものである．この TS-Join 演算は以下のように定義されている．

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

R はリレーションを指す．TS-Join 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 $r.A_i$ の値に一致する名前のリレーション ($rel(r.A_i)$) を取得し，属性 $r.A_1,\ldots,r.A_j$ 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．問合せ記述中に TS-Join 演算を利用するために，TS JOIN 句を以下のように記す．

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

$A_1,\ldots,A_j$, A_i, および $N_1,\ldots,N_j$ は TS-Join 演算の定義式と同じである．TS JOIN 句を用いた問合せ記述例は図 1 に記述されている．

具体的に図 1 を用いて処理手順を説明する．人物 A の最寄りのネットワークカメラの映像を監視するには図 1 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，分散ストリーム処理環境で問合せ処理が行われておらず，対象情報源となるネットワークカメラは不明である．問合せ処理を実行すると人物 A の位置情報がノード 1 からノード 2 に送られ，位置情報とカメラ位置データベースとの結合処理により人物 A との距

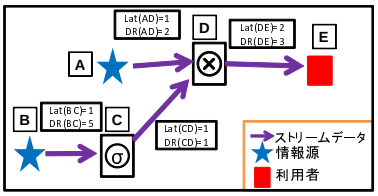

図 3　分散ストリーム処理の例

Fig. 3　An example of distributed processing

離が 5m 未満のネットワークカメラ 1 が探索される．探索結果がノード 2 からノード 1 に送られると，TS-Join 演算では結合演算の処理結果から，利用者の問合せに記述された TS JOIN 句のパラメータを元に Name 属性の値であるネットワークカメラ 1 に接続を行い，Attribute 属性の値であるネットワークカメラ 1 の Video 属性の値を取得して新しい属性 Video として，入力タプルと直積を取り利用者に配信する．この処理過程は人物 A の位置情報を受信する度に繰り返し行われ，対象情報源となるネットワークカメラはその度に動的に選択される．

3.2　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 2 である．各モジュールの説明を下に列挙する．

- 問合せ解析器：利用者からの分散問合せを解析し，標準演算と TS-Join 演算からなる処理木を作成する．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なものに保つ．演算配置最適化手法は，初めに分散ストリーム処理環境の演算配置を算出する初期演算配置最適化手法と，定期的に現在の最適な演算配置を算出する再演算配置最適化手法からなる．
- 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る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同士の連結・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．

演算配置最適化手法は次の 3.3 で述べる．

3.3　演算配置最適化手法

演算配置最適化手法について説明する．詳細は論文 [12] を参照して頂きたい．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以下の式を用いて算出する．

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

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 3 を用いて具体的に説明する．図では，利用者の問合せを

```
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 "TS-Join")
05.       V_i ← V_i ∪ {I_dummy}
06.       E_i ← E_i ∪ {I_dummy ← TS-Join}
07.     END IF
08.   END FOR
09. END FOR
10. OptimalPlan := Calculate(G, N, L, D)
11. AllocOperators(OptimalPlan)
```

図 4　初期演算配置最適化手法のアルゴリズム

Fig. 4　Algorithm for initial optimal operator allocation

```
Input   :
  G     : a set of query trees { 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
Targets : a set of target information sources {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) ∈ G)
03.   FOR(Operator ∈ V_i)
04.     IF( Operator equals "TS-Join")
05.       FOR(I ∈ (I_Targets))
06.         V_i ← V_i ∪ I
07.         E_i ← E_i ∪ {I ← TS-Join}
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 size
```

図 5　再配置演算配置最適化のアルゴリズム

Fig. 5　Algorithm for optimal operator reallocation

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式 (2) を用いて次のように計算される．

$DR(AD)Lat(AD)+DR(BC)Lat(BC)+DR(CD)Lat(CD)+DR(DE)Lat(DE) = 1\times2+1\times5+1\times1+2\times3 = 14(Byte)$

ネットワーク使用量 $u(q)$ 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．

3.3.1　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 [7] 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，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 [7] 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図 4 である．10 行目の Calculate 関数は，既存手法 [7] であり，11 行目の AllocOperators 関数により，算出された演算配置を分散環境の各ノードに指示する．

3.3.2　再演算配置最適化手法

再演算配置最適化手法では，分散ストリーム処理環境から得られる対象情報源を元に処理木上の対象情報源と TS-Join 演算

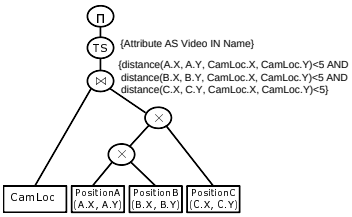

図 6　対象情報源の探索が複雑化した問合せ
Fig. 6　Complicated target search query

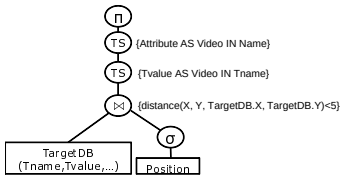

図 7　複数の TS-Join を用いた問合せ
Fig. 7　Query with multiple TS-Join

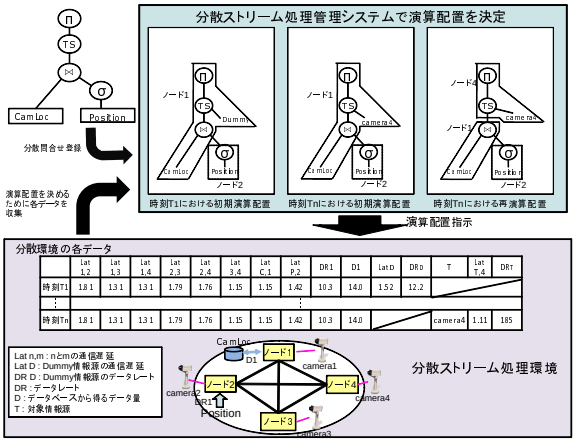

図 8　演算配置最適化手法により導出された各配置プラン
Fig. 8　Operator allocation plan generated by method of optimal operator allocation

をリンクさせる．そして，再構成した処理木を従来手法を用いてその時点の最適な演算配置を算出する．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小さければ，演算の再配置を指示する．

再演算配置最適化手法のアルゴリズムは図 5 である．Targets は対象情報源の集合を表し，I_{on} は，n 番目の TS-Join 演算の入力となる対象情報源を表す．例えば，I_{o1} では，TS-Join 演算 $o1$ にデータを入力する情報源を意味する．

4. 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．

4.1 シミュレーション実験

4.1.1 人物の映像監視要求を用いた実験

3 種類の問合せをデータレート，遅延を人工的に生成したシミュレーション環境で処理した時の評価実験に関して述べる．本実験で用いる問合せは，図 1 中の問合せ要求例として挙げた人物の映像監視要求と，図 6 の対象情報源の探索が複雑化した問合せと，図 7 の複数の TS-Join 演算を用いた情報源の動的選択の問合せの 3 つの問合せである．

図 6 の問合せは，各移動人物が身に付けている位置タグから位置情報が配信される環境において，移動人物 3 名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 7 の問合せでは，位置情報配信サービスから位置情報が配信され，TargetDB には，図 1 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 TS-Join 演算のそれぞれの選択率は 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み合わ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一台のネットワークカメラを対象情報源として選択

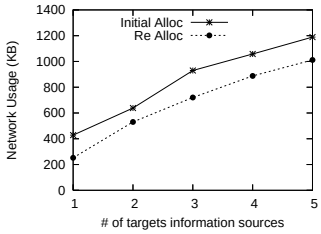

図 9　対象情報源数
Fig. 9　# of target sources

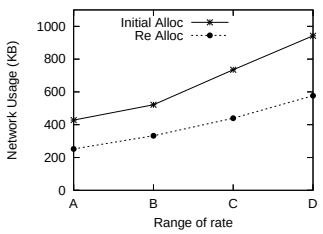

図 10　データレートの変動
Fig. 10　Range of data rate

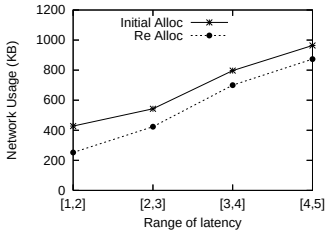

図 11　通信遅延の変動
Fig. 11　Range of latency

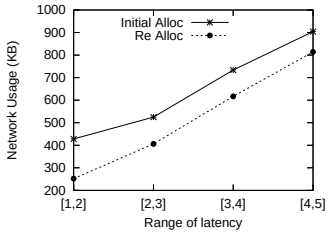

図 12　対象情報源のみ遅延変動
Fig. 12　Range of targets latency

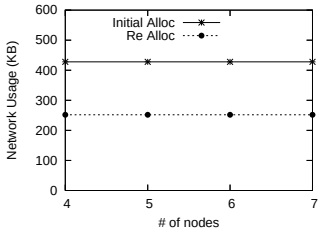

図 13　ノ ー ド 数
Fig. 13　A number of nodes

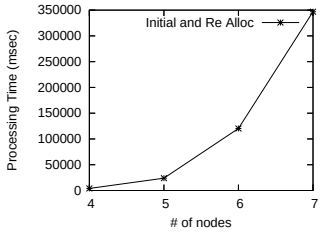

図 14　演算配置最適化に要した時間
Fig. 14　calculating time

- 通信遅延: [1，2] の間でランダムに値を発生
- データレート: [100，200] の間でランダムに値を発生させたネットワークカメラからのデータレートと，[10，20] の間でランダム発生をする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
- ノード数: 4 ノード

上記の基本となる組み合わせから，他を全て同じ条件にして各項目の内いずれか一つのみを変動させた時に分散ストリーム処理管理システムが算出した下記の二点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） 全演算を再評価する再演算配置 (Re Alloc)

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 回ずつ起こした時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．

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み合わ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 8 であり，その時のネットワーク使用量は図 9 から図 13 の X 軸の第一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時刻 T_1 と T_n 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時の最適な演算配置を算出している．基本となる組み合わせと同様に，他の組合せに関しても分散ストリーム処理環境のシュミ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．次に各実験の考察を述べる．

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

実験結果は図 9 である．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であるネットワークカメラの数が増える．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入力として受け取ると，出力されるデータレートが著しく高まる．よって，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

実験結果は図 10 である．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．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た．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 の時に，再演算配置 (Re Alloc) では初期演算配置 (Initial Alloc) に比べて 200KB 程のネットワーク使用量を抑えていることが分かる．

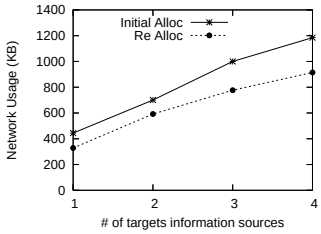

図 15 対象情報源数

Fig. 15 # of target sources

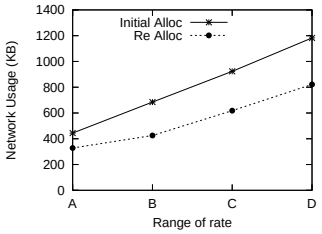

図 16 データレートの変動

Fig. 16 Range of data rate

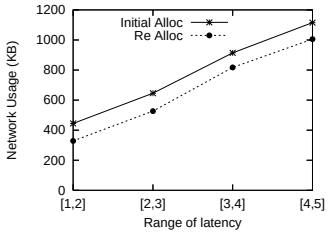

図 17 通信遅延の変動

Fig. 17 Range of latency

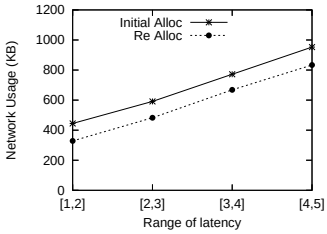

図 18 対象情報源のみ遅延変動

Fig. 18 Range of targets latency

実験 3． 通信遅延を変動させた比較実験

実験結果は図 11 である．本実験では，遅延を最大 5 秒までランダムに発生させた．遅延が大きくなるとネットワーク使用量が増えるが，再演算配置が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

実験結果は図 12 である．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 3 とほぼ同様の結果が得られた．

実験 5． ノード数を変動させた実験

実験結果は図 13 である．この実験では，ノード数はネットワーク使用量に影響を与えないことが分かる．しかし，ノード数の変動は最適な演算配置を決める時の計算量に影響を与える．ノード数を変化させた時に要する各演算配置手法の合計の時間は図 14 である．ノード数が増加するにつれて，計算時間が著しく増大していることが分かる．今回の評価実験に用いた実装システムでは，$O(n^m)$ の計算コストがかかる．n はノード数であり，m は演算数である．最適な演算配置の計算に時間がかかると，その間に分散ストリーム処理環境の各データが変動し，計算された演算配置が既に最適でないものになっている可能性が生じる．このことから，ネットワーク使用量を最適化するには，演算配置最適化の計算コストも重要な要素であることが分かる．

4.1.2 情報源の探索が複雑化した問合せ要求を用いた実験

図 6 の情報源の探索が複雑化した問合せに対して，分散ストリーム処理管理システムが，同時に接続する対象情報源数を変動させた比較実験，データレートを変動させた比較実験，通信遅延を変動させた比較実験，対象情報源とノード間の通信遅延のみを変動させたそれぞれの状態において算出した初期演算配

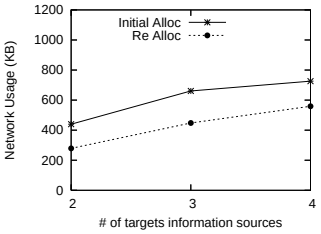

図 19　対象情報源数

Fig. 19　# of target sources

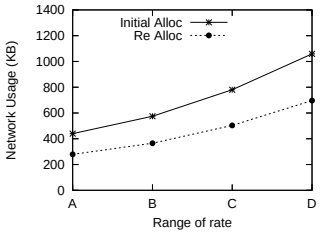

図 20　データレートの変動

Fig. 20　Range of data rate

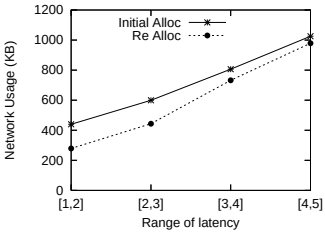

図 21　通信遅延の変動

Fig. 21　Range of latency

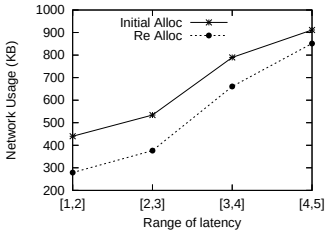

図 22　対象情報源のみ遅延変動

Fig. 22　Range of target latency

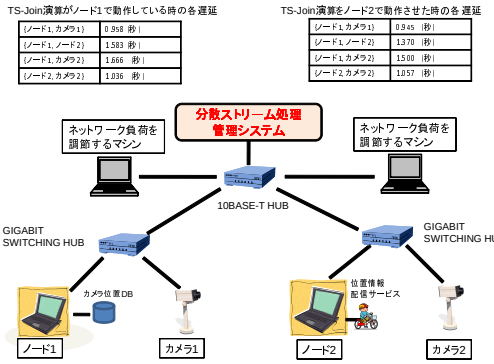

図 23　ネットワーク構成

Fig. 23　Network topology

置と再演算配置のネットワーク使用量は図 15，16，17，18 である．ノード数を変動させた実験は省略する．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検証された．

4.1.3　複数の TS-Join 演算を含む問合せ要求を用いた実験

図 7 の複数の TS-Join 演算を含む問合せに対して，分散ストリーム処理管理システムが，同時に接続する対象情報源数を変動させた比較実験，データレートを変動させた比較実験，通信遅延を変動させた比較実験，対象情報源とノード間の通信遅延のみを変動させたそれぞれの状態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 19，20，21，22 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よりネットワーク使用量を改善していることが検証された．

4.2　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．

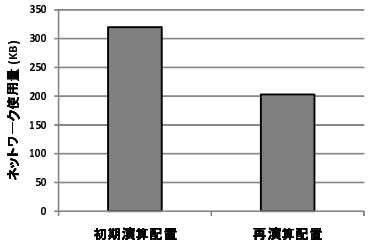

図 24　実環境上での実験結果

Fig. 24　Experimental result in real data

4.2.1　実 験 環 境

外的要因による影響を避けるために，LAN 内に図 23 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．

4.2.2　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 に切り替わった時に，分散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 で動作している時とノード 2 で動作した時の各遅延は図 23 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 24 である．実験結果より最適化を行うことで約 120KB 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，4.1.1 節のシミュレーション実験の基本となる組み合わ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．

シミュレーション環境と実環境の実験結果から，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り，分散ストリーム処理管理システムの有効性を示した．

5.　広域分散環境での運用

分散ストリーム処理管理システムの実運用を目指し，広域環

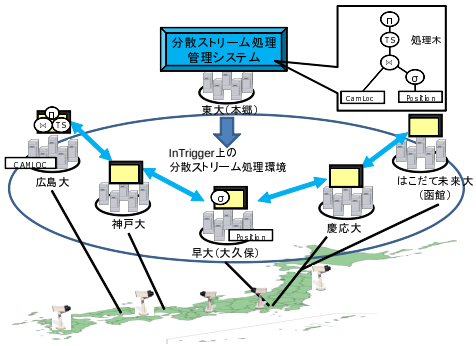

図 25　InTrigger 上の分散ストリーム処理環境を管理する分散ストリーム処理管理システム

Fig. 25　Distributed management system which manages distributed environment on InTrigger

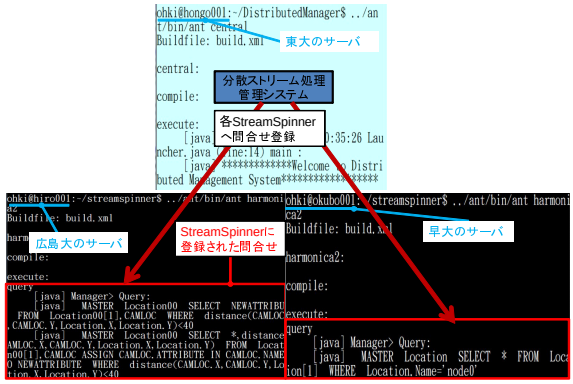

図 26　分散ストリーム処理管理システムの実行画面

Fig. 26　A screenshot of distributed management system

境で対象情報源が変化する分散ストリーム処理環境の管理・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 [16] の InTrigger [14] を使用し，StreamSpinner により構築した．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 25 は，分散ストリーム処理環境管理システムを東大に配置し，5 ノードで図 1 の人物の映像監視処理している様子を表している．ここでは，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，InTrigger 上ではネットワークカメラからの映像データを取り扱えないので，擬似データを配信している．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 26 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．

上記に述べたように，広域分散環境上において運用管理が可能であることを示した．

6. 結　　論

本論文では，対象情報源が動的に変化する分散ストリーム処理環境における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の評価を行った．評価実験の結果から定期的に再配置を行うことで，ネットワーク使用量を最適に保つことが可能であることを検証した．また，広域分散環境 InTrigger を用いた実際の運用管理も可能であることを示した．

今後の課題は演算配置最適化手法の計算の改善と，分散ストリーム処理環境の状況を示すインターフェースの作成を行う予定である．

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．

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] S. Chandrasekaran, et al., “TelegraphCQ: Continuous Dataflow Processing for an Uncertain World”, Proc. CIDR, 2003.

[5] D.Abadi, et al, “Scalable Distributed Stream Processing”, Proc. CIDR, 2003.

[6] Rajeev Motwani, et al. “Query Processing, Resource Management, and Approximation in a Data Stream Management System”. Proc. CIDR, 2003.

[7] P.Pietzuch, et al.“Network-Aware Operator Placement for Stream Processing Systems”, Proc. ICDE, p.49, 2006.

[8] C. Wyss, et al. “Extending Relational Query Optimization to Dynamic Schemas for Information Integration In Multidatabases”, Proc.SIGMOD ’07, pp.473-484, 2007.

[9] Y. Xing, et al. “ Dynamic Load Distribution in the Borealis Stream Processor”, Proc. ICDE, pp. 791-802, 2005.

[10] 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.

[11] 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”, データ工学ワークショップ DEWS2008, 2008 年

[12] 大喜恒甫，渡辺陽介，秋山亮，北川博之，天笠俊之，川島英之. “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”, iDB フォーラム 2008 年

[13] 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.

[14] InTrigger, https://www.logos.ic.i.u-tokyo.ac.jp/intrigger/

[15] StreamSpinner, http://www.streamspinner.org/

[16] 情報爆発プロジェクト, http://www.infoplosion.nii.ac.jp/infoplosion/