DEIM Forum 2011 D1-5

高信頼化を考慮した分散ストリーム処理の問合せ最適化方式

塩川　浩昭† 　北川　博之†,†† 　川島　英之†,††

† 筑波大学大学院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
†† 筑波大学計算科学研究センター　〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †shion@kde.cs.tsukuba.ac.jp, ††{kitagawa,kawasima}@cs.tsukuba.ac.jp

あらまし　近年，時々刻々と配信されるストリームデータが実世界に増大し，このようなデータに対する問合せ処理の実現が重要視されている．そして，それらの処理要求を実現する分散ストリーム処理環境に関する研究がこれまで行われてきた．分散ストリーム処理環境の構築は，ストリーム処理において性能向上や負荷分散，地理的に離れた情報源の統合利用などを実現する上で重要な基盤技術である．近年の研究では，分散した計算資源の効率利用に向けた問合せ最適化が注目されてきたが，問合せの単一障害点回避については考慮されてこなかった．そこで本稿では，このような分散ストリーム処理環境に対し，我々の先行研究である高信頼化手法 Semi-Active Standby 方式を応用した分散問合せ最適化方式を提案する．提案方式は，各問合せの単一障害点回避を考慮した分散問合せ最適化を行う．本稿では，提案方式の詳細と，評価実験結果について述べる．

キーワード　ストリーム処理，高信頼化，問合せ最適化

A Query Optimization Scheme for Highly Available Distributed Stream Processing Environments

Hiroaki SHIOKAWA†, Hiroyuki KITAGAWA†,††, and Hideyuki KAWASHIMA†,††

† Graduate School of Systems and Information Engineering, University of Tsukuba　Tennoudai 1–1–1, Tsukuba City, Ibaraki, 305–8573 Japan
†† Center for Computational Sciences, University of Tsukuba　Tennoudai 1–1–1, Tsukuba City, Ibaraki, 305–8573 Japan
E-mail: †shion@kde.cs.tsukuba.ac.jp, ††{kitagawa,kawasima}@cs.tsukuba.ac.jp

1. ま え が き

近年，各種センサデバイスや映像データ，株価情報などに代表される，時々刻々とデータを配信するストリーム型情報源が増大し，それらに対する継続的な問合せ処理要求が高まっている．ストリームデータに対する典型的な問合せ処理要求の例として，センサデバイスが配信するデータのフィルタリングや，GPS の示す位置情報とネットワークカメラから配信される映像データの統合利用などが挙げられる．そして，このような問合せ処理要求を実現するための基盤システムとして，ストリーム処理エンジン (SPE) [1, 8] が研究開発されている．SPE では，利用者が問合せ処理要求を登録すると，内部で処理木と呼ばれる処理プランを生成する．そして，情報源からデータが到着する度に処理プランを計算機のメモリ上で即時実行し，その処理結果を利用者に配信する．

SPE では，地理的に離れた場所に設置された情報源を扱う際や，負荷分散を行う際に，ネットワーク上に分散した複数の計算資源上で動作する分散ストリーム処理環境 [2, 4] を構築する．分散ストリーム処理環境では，複数の SPE が互いに連携し，情報源から利用者までストリームデータを中継することで処理を実現する．このような分散ストリーム処理環境では，複数の SPE がネットワークを介してデータのやり取りをするため，どの処理をどの計算機に割当てるかに依存して問合せ処理の性能が変化することになる．ゆえに，分散ストリーム処理環境における問合せ最適化の研究 [9, 11] は重要な研究テーマの一つとなっている．

また一方で，SPE はメモリ上で処理を実行するため，分散ストリーム処理環境を構成する一部の計算機が故障などにより停止してしまうと，それが単一障害点となりデータが中継されず分散ストリーム処理環境全体が停止してしまうという問題がある．また，システム全体が停止している間もストリームデータは到着し続けるため，大量のストリームデータが失われるこ

とになる．しかし，分散ストリーム処理環境上でサービスを運用する際には，単一障害点の回避や，障害回復にかかる時間をサービス上の制約時間以下に抑えるといった点が重要な課題となる．ゆえに，分散ストリーム処理環境において，高信頼化処理は欠くことのできないものとなっている．

そこで本稿では，分散ストリーム処理環境において，高信頼化を考慮した問合せ最適化方式の提案を行う．本方式では，既存の問合せ最適化方式である Relaxation アルゴリズム [9] に対し，我々の先行研究である Semi-Active Standby 方式 [13] を適用する事により，登録された各問合せのリカバリ時間制約を満たす高信頼化を実現し，問合せのネットワーク使用量を準最小化する．さらに本稿ではシミュレーション実験により，本提案方式の有効性を示す．

本稿の構成は以下のとおりである．2 節にて，関連研究について説明する．3 節にて，本稿における前提と研究課題を示し，4 節で提案方式の詳細ついて述べる．そして，5 節にて評価実験，6 節で本稿のまとめと今後の課題について説明する．

2. 関 連 研 究

分散ストリーム処理環境における問合せ最適化に関する研究として，分散環境中のネットワーク使用量を最適化する研究 [9] や，分散環境を構成している各計算機の CPU 使用率をバランスさせる研究 [11] などが挙げられる．本稿で提案する問合せ最適化方式は，方式 [9] と同様に，分散ストリーム処理環境におけるネットワーク使用量の最適化を目標としているが，本稿はこれに加えて高信頼化を考慮するという点で大きく異なっている．

分散ストリーム処理の高信頼化に関する研究として，平常時にストリーム処理を行っている計算機（プライマリ）に対して，そのバックアップを担当する計算機（セカンダリ）を用意し，単一障害点に備える高信頼化手法の研究 [3, 6] が挙げられる．そのなかでも特に代表的な Hwang らによる研究 [6] について述べる．この研究では，特性の異なる 3 つの高信頼化手法の提案を行なっている．しかしながらこれらの方式は，ネットワーク使用量もしくはリカバリ時間において，システム利用者の要求する制約を満たせなくなる場合があることが，我々の先行研究 [10, 13] により指摘されている．これに対し我々の先行研究 [10, 13] では，システムの実行状況に応じネットワーク使用量およびリカバリ時間の関係を調整可能な高信頼化手法 Semi-Active Standby 方式を提案している．本稿で提案する問合せ最適化方式では，ネットワーク使用量を抑えリカバリ時間の制約を満たす必要があるため，本高信頼化方式を採用している．

3. 前提と研究課題

3.1 システムモデル

本稿では，分散ストリーム処理環境に対する高信頼化の既存研究 [6] と同様に，図 1 に示すようなシステムモデルを有する高信頼化処理が動作するものとする．図 1 は 3 つのノードに分散した場合におけるシステムモデルの例である．ノード N_u，N，N_d は平常時処理を行うノードであり，プライマリと呼ぶ．

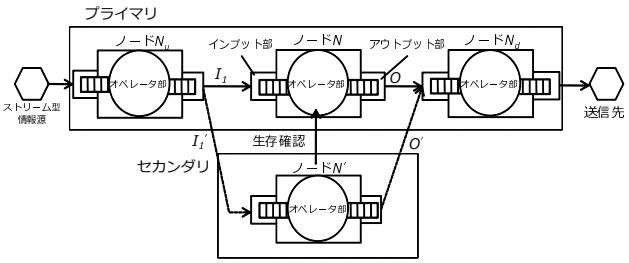

図 1　システムモデル（3 ノード）

またノード N が高信頼化対象ノードであるとき，ノード N' は，ノード N で単一障害点が発生した際に障害回復処理を担当するノードであり，セカンダリと呼ぶ．セカンダリは担当するプライマリに対し，定期的にパケットを送信することで生存確認を行い，プライマリの障害を検知した場合には，セカンダリが処理の引き継ぎを行う（N が停止した場合，I_1，O をそれぞれ I_1'，O' に切り替える）．

このシステムモデルに基づく高信頼化手法はいくつか存在する [6]．本稿では，リカバリ時間を小さく抑えることが可能な高信頼化手法である Active Standby 方式 [6] や Semi-Active Standby 方式 [13]，適応型 Semi-Active Standby 方式 [10] を利用を想定する．

3.2 前　　提

本稿で扱う分散ストリーム処理環境およびシステムモデルに関して，以下の内容を前提とする．

- 任意の計算機における単一障害点の回避を対象とする．
- ビザンチン障害は対象としない．
- Rollback Recovery [6] を対象とする．
- 通信によるデータの順序の入替りと欠損が生じない．

さらに本稿では，次に挙げる項目に関する情報が事前に取得可能であると仮定する．

- 全ての情報源のデータレート
- 全てのオペレータの選択率
- 1 タプル当りの処理時間

3.3 研 究 課 題

前述したシステムモデルと前提に基づき，本稿における研究課題を定義する．

分散ストリーム処理環境における問合せ最適化の既存研究では，ネットワーク使用量の最小化や負荷分散が主な課題とされてきた．しかし，分散ストリーム処理環境上でストリーム型情報源を用いたサービスを運用する際には，単一障害点を回避し，その障害回復時間をサービス上の制約時間以下に抑える必要がある．そこで本稿では，分散ストリーム処理環境に登録される全ての問合せに対して，各問合せに設定されたリカバリ時間制約を充足し，そのネットワーク使用量を最小化することを研究課題と設定する．

本稿では，先行研究 [6, 9] に基づき，障害回復に要するリカバリ時間 R_{time} と，問合せ q におけるネットワーク使用量 $u(q)$ を次のように定義する．

［定義 1］　リカバリ時間 R_{time}

Algorithm 1 高信頼化を考慮した問合せ最適化方式

1: 仮想座標系を構築（4.2 節）
2: **while** 未配置の問合せ q が存在 **do**
3: 問合せ q の解析と処理木 q_{tree} の生成（4.3 節）
4: q_{tree} 中のプライマリを配置（4.4 節）
5: q_{tree} 中のセカンダリを配置（4.5 節）
6: q_{tree} において高信頼化処理を実行（4.6 節）
7: **end while**

上流側のプライマリ N_{u_i} とセカンダリ N' 間における通信遅延時間を K_i，プライマリ N_{u_i} の保持する再処理が必要なタプル数を Q_i，1 タプル当りの処理時間を p とするとき，R_{time} は以下のように定義される．

$$R_{time} = max(K_i) + \sum_i Q_i p \tag{1}$$

［定義 2］ ネットワーク使用量 $u(q)$
問合せ q に含まれるオペレータ間のリンク集合を L，リンク l に流れるデータレートを $DR(l)$，通信遅延時間を $Lat(l)$ とするとき，ネットワーク使用量 $u(q)$ は以下のように定義される．

$$u(q) = \sum_{l \in L} DR(l) Lat(l) \tag{2}$$

ここで各問合せにおけるリカバリ時間制約を R'_{time} とする．このとき定義 1，2 より本研究課題は，問合せ q に含まれる全ての高信頼化対象ノードについて $R_{time} < R'_{time}$ を満たし，$u(q)$ を最小化することであると定義される．

4. 提 案 方 式

4.1 概　　要

本節では提案方式である高信頼化を考慮した問合せ最適化方式の詳細について述べる．本方式における処理の流れをアルゴリズム 1 に示す．

本方式では，まず Vivaldi アルゴリズム [5] により仮想座標系を構築する．ここで構築される仮想座標系は，分散ストリーム環境を構成する任意の計算機間の通信遅延を，仮想座標系中におけるユークリッド距離で表現したものである．本方式では，この仮想座標系を利用して分散ストリーム環境における問合せ最適化を行う．

仮想座標系構築後，未配置の問合せ q から高信頼化を考慮した処理木 q_{tree} を生成し，q_{tree} の配置を順次行う．まず，q_{tree} 中のプライマリのみに対し，既存方式 Relaxation アルゴリズム [9] を用いて，ネットワーク使用量を最小化する配置を決定する．続いて，先に得られたプライマリの配置に基づき，リカバリ時間制約を満たすセカンダリの配置を探索する．q_{tree} 中に含まれる全処理の配置が完了した後，高信頼化処理を実行する．以後，未配置の問合せが存在する限り同様の処理を継続する．

本節では，以降 4.2 節から 4.6 節にて，アルゴリズム 1 に示した各処理の詳細について述べる．

4.2 仮想座標系の構築

提案方式では，既存研究 [9] と同様に n 次元の仮想座標系の構築を行う．仮想座標系とは，ネットワーク上に分散配置された計算機間の通信遅延を，仮想座標系中におけるユークリッド距離を用いて表現したものである．仮想座標系の例を図 2 に示す．図 2 は，後述するネットワークトポロジシミュレータである GT–ITM を用いて作成した 100 台の計算機からなる分散ストリーム処理環境である．この仮想座標系中における各点は，分散ストリーム処理環境中に存在する計算機を表している．仮想座標系中に含まれる各点間のユークリッド距離は，実際の計算機間における通信遅延を表している．ゆえに，座標系において近い点は実際のネットワーク上でも通信遅延の少ない計算機となり，遠い点は通信遅延の大きな計算機となる．

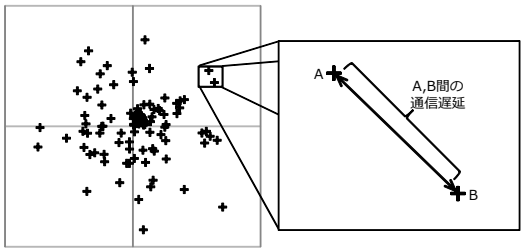

図 2　GT–ITM により生成した 100 ノードの仮想座標系

Algorithm 2 Vivaldi アルゴリズム：入力 (rtt,$\vec{x_i}$,$\vec{x_j}$)

1: **repeat**
2: 実測値と仮想座標系による推定値の誤差を計算
$e = rtt - \|\vec{x_i} - \vec{x_j}\|$
3: 誤差の生じている座標方向への単位ベクトルを生成
$\vec{dir} = u(\vec{x_i} - \vec{x_j})$
4: 誤差修正のためのベクトル f を生成
$\vec{f} = \vec{dir} \times e$
5: 自身の座標をベクトル f 方向へ移動させる
$\vec{x_i} = \vec{x_i} + \delta \times \vec{f}$
6: **until** $e < e_t$

仮想座標系の構築には分散型のアルゴリズムである Vivaldi アルゴリズム [5] を利用する．Vivaldi アルゴリズムでは，各計算機が他の計算機との通信時に測定された通信遅延から自律的に仮想座標系中における自己の座標を決定していく．Vivaldi アルゴリズムの詳細をアルゴリズム 2 に示す．Vivaldi アルゴリズムでは入力として，仮想座標系における自身の座標 $\vec{x_i}$ と通信相手の座標 $\vec{x_j}$，座標 $\vec{x_j}$ の計算機と通信を行った際に測定された通信遅延時間 rtt を受け取る．そしてこれらの値から，通信遅延時間 rtt と仮想座標系中のユークリッド距離 $\|\vec{x_i} - \vec{x_j}\|$ の差をとり，現在の仮想座標系における誤差 e を算出する．続いて，座標 $\vec{x_i}$，$\vec{x_j}$ から誤差修正方向を示す単位ベクトル $\vec{dir}$ を取得し，e と $\vec{dir}$ から誤差修正ベクトル $\vec{f}$ を生成する．最後に，$\vec{f}$ を $\delta(0 \leqq \delta \leqq 1)$ 倍したベクトル分，座標 $\vec{x_i}$ を移動させることにより実測値と仮想座標系の誤差を修正していく．この処理は誤差 e が閾値 e_t より小さくなるまで継続される．Vivaldi アルゴリズムでは，これらの処理を各計算機において，他の計算機と通信が発生する毎に実行する事により仮想座標系の修正を行う．

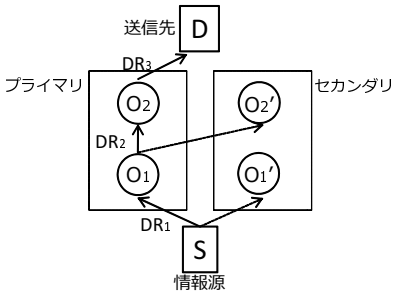

図 3　高信頼化を考慮した処理木 q_{tree} の例

さらに本研究では，各計算機において CPU 使用量をモニタリングすることで，仮想座標系に対して各計算機の CPU 使用量を考慮した次元軸を追加する．CPU 使用量を表現する次元軸では，利用者により事前に定義されるスケーリング関数を用いることにより，CPU 使用量を通信遅延時間として表現する．この次元軸を導入することにより，通信遅延が近い複数の計算機が存在した場合，CPU 使用量が小さい計算機の方がより近く，CPU 使用量が大きい計算機の方がより遠く仮想座標系中において表現されることになる．

4.3　処理木の生成と解析

問合せ q から高信頼化を考慮した処理木 q_{tree} を生成する．図 3 に生成される処理木の例を示す．まず問合せから処理木 (S, O_1, O_2, D) を生成し，プライマリ (O_1, O_2) に対してそれぞれ 1 つずつセカンダリ (O_1', O_2') を用意する．続いて，q_{tree} のプライマリ (O_1, O_2) に対して Relaxation アルゴリズム [9] を適用するために，事前に与えられた各情報源のデータレートと各オペレータの選択率を基にして，プライマリ間のデータレート (DR_1, DR_2, DR_3) を計算する．

4.4　プライマリの配置

前節で生成した q_{tree} のうち，プライマリのみ配置を行う．本方式は高信頼化におけるリカバリ時間制約を充足し，ネットワーク使用量の最小化する問合せ最適化を目標とする．そこで，Relaxation アルゴリズム [9] を利用し，q_{tree} 中のプライマリによるネットワーク使用量を最小とする配置を決定する．

4.4.1　Relaxation アルゴリズムの特性

Relaxation アルゴリズムは，仮想座標系において処理木のネットワーク使用量が最小となるような配置を決定するアルゴリズムである．このアルゴリズムでは，分散ストリーム処理環境に配置される処理木を，バネ（オペレータ間のリンク）により接続された，質量を持たない物体（オペレータ）の集合としてモデル化している．Relaxation アルゴリズムの目標は，このようにバネモデルとしてモデル化された処理木に対して，バネの弾性エネルギーを最小化するようなオペレータの位置を求めることである．

バネモデルにおいて，物体がバネ i により受ける力を $\vec{F_i}$，バネ定数を k_i，バネの伸びを $\vec{s}$ とすると，$\vec{F_i} = \frac{1}{2} k_i \vec{s}$ となる．これに基づき，仮想座標系で動作する Relaxation アルゴリズムでは，オペレータ間のリンク l におけるバネ定数をリンク l におけるデータレート $DR(l)$，バネの伸びをリンク l における通信遅延時間 $Lat(l)$ と設定する．これにより，オペレータがリンク l により受ける力 $\vec{F_l}$ は以下のようになる．

$$\vec{F_l} = \frac{1}{2} DR(l) Lat(l) \tag{3}$$

Relaxation アルゴリズムではバネの弾性エネルギーを最小化するために，q_{tree} 中における $\vec{F_l}$ の総和を最小化することから，

$$\min \sum_{l \in L} \frac{1}{2} DR(l) Lat(l) < \min \sum_{l \in L} DR(l) Lat(l) \tag{4}$$

となるため，定義 2 に示したネットワーク使用量を最小化することになる．

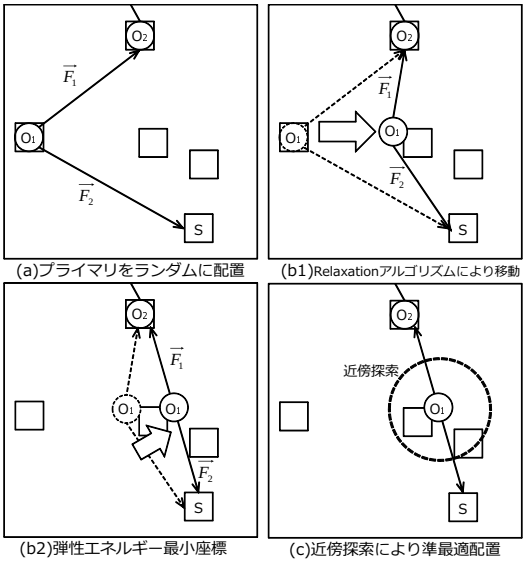

図 4　Relaxation アルゴリズムの動作例

4.4.2　Relaxation アルゴリズムによる処理の流れ

Relaxation アルゴリズムでは最初に，入力されたプライマリを分散ストリーム処理環境中の計算機に対して図 4(a) の様にランダムに配置する．続いて，Relaxation アルゴリズムを用いて，図 4(b1,b2) の様に仮想座標系中における弾性エネルギーが最も小さくなる座標を求める．

Relaxation アルゴリズムの詳細をアルゴリズム 3 に示す．Relaxation アルゴリズムは入力として，プライマリ O_i を受け取る．まず，O_i と隣接するプライマリ O_j を取得し，O_i，O_j 間におけるバネの力，すなわちネットワーク使用量を表すベクトル $\vec{F}$ を求める．この処理を全ての隣接プライマリに対して実行し，最終的に O_i にかかるネットワーク使用量を表すベクトルの総和 $\vec{F}$ を求める．最後に $\vec{F}$ を $\delta (0 \leq \delta \leq 1)$ 倍したベクトル分，O_i の座標 $\vec{O_i}$ を移動させる．この処理は各プライマリにおいて $\|\vec{F}\|$ が閾値 F_t より小さくなるまで繰り返される．

アルゴリズム 3 により，仮想座標系における各プライマリの最適な座標が求められた後，各プライマリを実際の計算機へと配置する．配置する計算機の決定には近傍探索を用いる．アルゴリズム 3 により得られた各プライマリの座標を中心とし，最近傍探索を行い，最初に見つかった計算機へプライマリを配置する．(図 4(c))

4.5　セカンダリの配置

プライマリの配置完了後，セカンダリの配置を行う．本研究

Algorithm 3 Relaxation アルゴリズム：入力 (O_i)

1: **repeat**
2: O_i にかかるネットワーク使用量ベクトル $\vec{F}$ の初期化
$\vec{F} \leftarrow \vec{0}$
3: **for all** O_j in $Connected(O_i)$ **do**
4: O_i と隣接するプライマリによるネットワーク使用量ベクトル $\vec{F}$ の総和を求める
$\vec{F} \leftarrow \vec{F} + (\vec{O_i} - \vec{O_j}) \times DR(O_i, O_j)$
5: **end for**
6: ネットワーク使用量ベクトル $\vec{F}$ を基にプライマリ O_i の座標を移動させる
$\vec{O_i} \leftarrow \vec{O_i} + \vec{F} \times \delta$
7: **until** $\|\vec{F}\| < F_t$

では既存研究 [9, 11] とは異なり高信頼化を考慮した問合せ最適化を行うため，セカンダリの配置を行う点において従来方式とは大きく異なっている．以下に，セカンダリ配置の方針と流れを示す．

4.5.1 セカンダリ配置の方針

本方式では，ネットワーク使用量を最小化し，高信頼化におけるリカバリ時間制約を満たす問合せ最適化を行う．本稿におけるリカバリ時間 R_{time} は定義 1 により，通信遅延時間 K と再処理時間 Qp の和である $R_{time} = K + Qp$ を採用する．この定義により，リカバリ時間を最も小さく抑えることが可能な Active Standby 方式による高信頼化を仮定した場合でも，高信頼化対象の上流側にあるプライマリとセカンダリとの間の通信遅延時間に等しい時間がリカバリ時間として必要となる．そこで，本研究ではセカンダリ配置の方針として，まず，仮想座標系上においてネットワーク使用量を最小化する起点座標を求める．そして，起点座標を中心に，リカバリ時間制約を満たす計算機を貪欲法により探索する．

4.5.2 セカンダリ配置の流れ

q_{tree} より，既に配置されている高信頼化対象の上流側プライマリの座標を取得する．そして，取得した座標からセカンダリ配置場所探索の起点となる起点座標を算出する．起点座標の計算は，セカンダリの種類により異なる．以下にそれぞれの計算方法を示す．

- **セカンダリの入力が 1 つの場合**

セカンダリ O' が選択演算や射影演算に代表されるような 1 入力となるオペレータである場合，リカバリ処理は高信頼化対象 O の上流側にあるプライマリ O_u からのみ行われることになる．したがって，O' に対して，ネットワーク使用量を最小化するセカンダリ配置座標は O_u の持つ座標である．ゆえに本方式では，セカンダリ O' の入力が 1 つの場合，上流側のプライマリ O_u の存在する座標から，CPU 使用量を除いた次元に射影した座標を起点座標とする．(図 5)

- **セカンダリの入力が 2 つの時**

セカンダリ O' が結合演算や和演算に代表されるような 2 入力となるオペレータである場合，リカバリ処理は高信頼化対象 O の上流側にある 2 つのプライマリ O_{u1}, O_{u2} から行われる．ゆえに，O_{u1}, O_{u2} からの入力に対してネットワーク使用量を

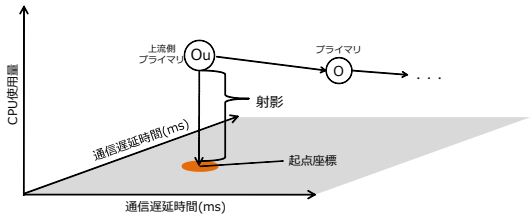

図 5　起点座標の射影

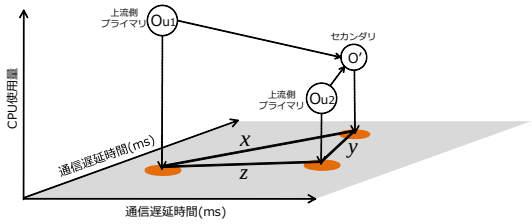

図 6　上流側プライマリとセカンダリの配置関係

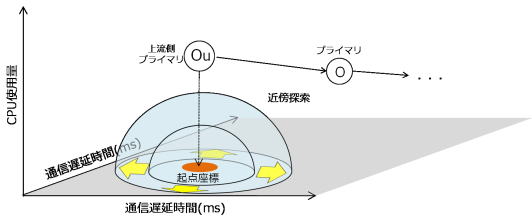

図 7　貪欲法による探索

最小化するような起点座標を選択する必要がある．ここで，上流側のプライマリにおけるデータレートをそれぞれ $DR(O_{u1})$, $DR(O_{u2})$（ただし，$DR(O_{u1}) \leqq DR(O_{u2})$）とし，仮想座標系上で図 6 に示すような位置関係でセカンダリ O' の配置が与えられた場合を考える．この時，O_{u1}, O_{u2} と O' の間に発生するネットワーク使用量 NW は以下の式となる．

$$NW = DR(O_{u1}) \times x + DR(O_{u2}) \times y \tag{5}$$

各プライマリにおけるデータレートの関係が $DR(O_{u1}) \leqq DR(O_{u2})$ であるため，ネットワーク使用量は以下の様な関係となる．

$$NW \geqq DR(O_{u1}) \times (x + y) \tag{6}$$

さらに，三角不等式により，$x + y \geqq z$ となるため，

$$NW \geqq DR(O_{u1}) \times (x + y) \geqq DR(O_{u1}) \times z \tag{7}$$

となる．ゆえに，セカンダリ O' が 2 入力となる場合における，ネットワーク使用量を最小化する起点座標は，データレートの大きい上流側のプライマリ O_{u2} の持つ座標となる．したがって本方式では，CPU 使用量のバランスを考慮し，O_{u2} の持つ座標を図 5 の様に CPU 使用量を除いた次元に射影した座標を起点座標とする．

起点座標決定後，以下の配置条件に従い起点座標を中心とした貪欲法による近傍探索を行い，該当する計算機が見つかった場合にセカンダリを配置する．(図 7)

配置条件

（1） 高信頼化対象のプライマリとは異なる計算機である
（2） セカンダリと上流にあるプライマリ間の通信遅延時間がリカバリ時間制約以下である

以上の処理により，セカンダリの配置を決定する．本方式では，上流側のプライマリの座標を起点座標として近傍探索を行うため，上流側のプライマリとセカンダリ間の通信遅延時間がリカバリ時間制約以下となる条件の下，ネットワーク使用量が可能な限り小さくなる配置を選択する．

4.6 Semi-Active Standby 方式による局所最適化

全てのオペレータの配置が完了した q_{tree} に対して高信頼化処理を実行する．3.1 節で述べた通り，本稿では高信頼化手法として，Active Standby 方式 [6] や Semi-Active Standby 方式 [13]，適応型 Semi-Active Standby 方式 [10] を想定している．

Active Standby 方式は，各プライマリと並列してセカンダリでもストリームデータの処理を行う高信頼化手法である．3.1 節で示したシステムモデル上において，ネットワーク使用量を犠牲にする代わりにリカバリ時間を最も小さくすることが可能である．

我々の先行研究である Semi-Active Standby 方式，適応型 Semi-Active Standby 方式ではこの Active Standby 方式を拡張し，高信頼化にかかるネットワーク使用量とリカバリ時間の間に存在するトレードオフを調整可能にしたものである．これらの方式はバッチサイズと呼ばれるパラメータを調整することにより，バックアップデータの通信量と所在を制御する．これにより，バッチサイズを大きくすることで，リカバリ時間は大きくなるが，ネットワーク使用量を小さく抑えることができる．これに対し，バッチサイズを小さくすることで，ネットワーク使用量は大きくなるがリカバリ時間を小さく抑える．前述した Active Standby 方式は，このバッチサイズを最も小さくした場合に当たる．そこで本稿では，これらの特性を利用して，リカバリ時間制約を充足する範囲内でネットワーク使用量を削減する局所的な最適化を行う．

5. 評価実験

5.1 実験概要

ネットワークトポロジジェネレータである Georgia Tech Internetwork Topoloy Generator(GT–ITM) [12] を用いて分散ストリーム処理環境のシミュレータを実装し，ネットワーク使用量，リカバリ時間，CPU 使用量の測定と比較を行う．本実験では，図 8 に示したパラメタを用いて GT-ITM により 100 ノードからなる分散環境を構築し，100 個の問合せを登録する．本実験で用いた問合せは図 9 のような 4 つの情報源と選択演算子，結合演算子から構成される有向木とした．問合せ中の情報源と送信先は，登録時にランダムに選択される．各情報源から配信されるストリームデータのデータレートは 2KB/s とした．また，本実験では同じ処理を行うオペレータの統合は考慮しないものとする．加えて，各通信路の最大帯域幅は無限であるとし，特定の通信路に負荷が集中しても通信速度や通信遅延時間の劣化はないものとした．

本実験では，処理木中の各オペレータをランダムで配置する**ランダム方式**，CPU 使用量が最も少ない計算機にオペレータを優先して配置する**ラウンドロビン方式**，プライマリを Relaxation アルゴリズムで配置し，セカンダリを全て起点座標に配置する**上流方式**を比較手法とした．

設定項目	設定値
method keyword	Geo
number of graph	1
initial seed	なし
n	100
scale	100
edge method	3
alpha	0.33

図 8 GT-ITM パラメータ

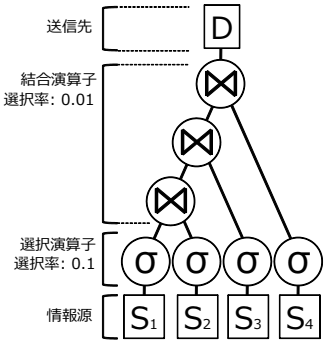

図 9 実験で用いる処理木

5.2 実験 1:ネットワーク使用量の比較

本実験では，各方式における分散ストリーム処理環境全体のネットワーク使用量の推移を比較する．本実験では，リカバリ時間制約を 100ms，高信頼化における Ack 送信間隔を 200ms と設定した．さらに，提案方式と上流方式については，Active Standby 方式により高信頼化した提案方式 (AS)，上流方式 (AS) と，適応型 Semi-Active Standby 方式により高信頼化した提案方式 (A-SAS)，上流方式 (A-SAS) についても比較を行った．ランダム方式とラウンドロビン方式については，適応型 Semi-Active Standby 方式による効果が期待できないとし，Active Standby 方式により高信頼化した場合のみ比較の対象とした．

5.2.1 実験結果

実験結果を図 10，11 に示す．図 10，11 は，問合せを 1 個から 100 個まで追加した場合のネットワーク使用量累積値の推移である．x 軸は登録した問合せの割合，y 軸はネットワーク使用量累積値を表す．図 10 は Active Standby 方式で高信頼化を行った場合の比較，図 11 は，提案方式，上流方式に置いて Active Standby 方式，適応型 Semi-Active Standby 方式を適用した場合の比較を示している．

- **Active Standby 方式を用いた時の各方式の比較**

図 10 の実験結果より，提案方式 (AS)，上流方式 (AS) は，ランダム方式，ラウンドロビン方式に対してネットワーク使用量を約 60%程度に抑えることがわかる．ランダム方式とラウンドロビン方式では，各オペレータ間に生じる通信遅延時間を考慮しないでオペレータの配置を行う．そのため，仮想座標系中において遠い計算機間で通信を行う様な配置が発生してしまい，ネットワーク使用量が増大してしまっている．

提案方式 (AS) と上流方式 (AS) を比較すると，全ての問合せを登録した時点で上流方式 (AS) の方が約 4.4%程度ネットワーク使用量を小さく抑えていることがわかる．上流方式 (AS) ではセカンダリを起点座標に配置するため，Relaxation アルゴリズムを用いた場合において，ネットワーク使用量を最小化する．これに対し提案方式 (AS) では起点座標を中心にリカバリ

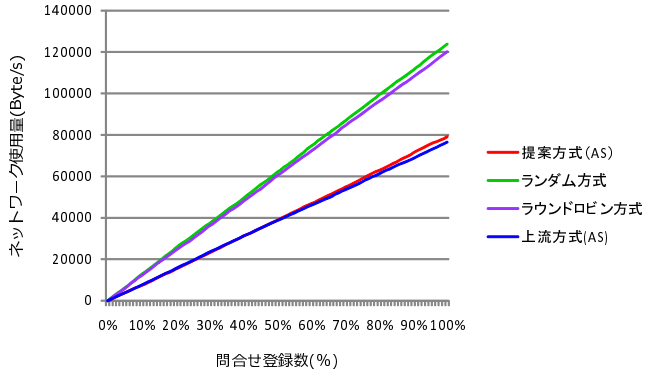

図 10　Active Standby 方式における各方式のネットワーク使用量比較

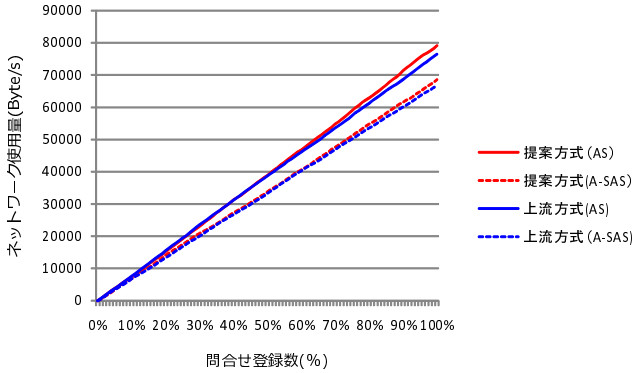

図 11　高信頼化方式の違いによる各方式のネットワーク使用量比較

時間制約を充足するセカンダリの配置を探索する．ゆえに，リカバリ時間制約を充足する配置を優先して選択するため，ネットワーク使用量を上流方式に対して数%多く消費するような配置を選択することになる．

- **異なる高信頼化方式における各方式の比較**

図 11 の実験結果により，提案方式 (AS)，提案方式 (A-SAS)，上流方式 (AS)，上流方式 (A-SAS) について比較を行う．

全ての問合せを登録した時点で上流方式 (A-SAS)，提案方式 (A-SAS)，上流方式 (AS)，提案方式 (AS) の順にネットワーク使用量を小さく抑えていることがわかる．同一の方式において比較を行うと，適応型 Semi-Active Standby 方式を用いた場合の方が，Active Standby 方式を用いた場合に比べて，提案方式では 14%程度，上流方式では 12%程度ネットワーク使用量を小さく抑えられていることがわかる．提案方式 (AS) や上流方式 (AS) では，各方式により配置の決定したセカンダリに対して，高信頼化対象となるプライマリと並列してバックアップデータを送信する．これに対し，提案方式 (A-SAS) や上流方式 (A-SAS) では，バッチサイズを大きくすることで，リカバリ時間制約を守れる範囲内で，セカンダリへの通信量を減らしている．したがって，各方式において，リカバリ時間制約を充足するようなセカンダリが存在する場合には，高信頼化方式として適応型 Semi-Active Standby 方式を用いることにより，ネットワーク使用量を削減することが可能であると考えられる．一方で，リカバリ時間制約が充足されていないセカンダリに対して適応型 Semi-Active Standby 方式を適用してもネットワーク使用量削減効果は得られない．ゆえに，適応型 Semi-Active Standby 方式を用いる場合には，分散ストリーム環境に登録された問合せのうち，リカバリ時間制約を充足するような問合せがより多く存在できる場合に対して有利に働くと考えられる．

5.3　実験 2:リカバリ時間の比較

本実験では，最大リカバリ時間，平均リカバリ時間，リカバリ時間制約充足率の比較を行う．リカバリ時間制約を 100ms，高信頼化における Ack 送信間隔を 200ms と設定し，提案方式 (AS)，提案方式 (A-SAS)，ランダム方式，ラウンドロビン方式，上流方式においてリカバリ時間の違いを比較した．

表 1　リカバリ時間の比較

	最大 (ms)	平均 (ms)	制約充足率 (%)
提案方式 (AS)	96.1	76.1	100%
提案方式 (A-SAS)	100.0	100.0	100%
ランダム方式	150.2	111.7	22%
ラウンドロビン方式	157.5	108.0	44%
上流方式	126.4	86.3	83%

5.3.1　実 験 結 果

実験結果を表 1 に示す．表 1 は，各方式の，最大リカバリ時間と平均リカバリ時間，100 問合せ中のリカバリ時間制約充足率を示している．

表 1 より，提案方式 (AS) は，リカバリ時間制約を 100ms と設定した本実験環境において，全ての問合せについて制約を充足することがわかる．また，提案方式 (A-SAS) では，実験 1 で示したようにネットワーク使用量を小さく抑えながら，リカバリ時間を制約と同じ 100ms 程度に調整している．

これに対し，ランダム方式，ラウンドロビン方式では，制約充足率が 22%，44%と低い値になっている．ランダム方式，ラウンドロビン方式では，ネットワーク使用量や通信遅延時間を考慮した配置を一切行わない．ゆえに，リカバリ時間において非効率的な配置が多く存在し，制約充足率が低くなっている．

また，上流方式では制約充足率 83%と提案方式より低くなっている．これは上流方式がセカンダリを起点座標に配置することにより，もう一方のプライマリが仮想座標系上離れた計算機にある場合，リカバリ時間制約が守れなくなることがあるためであると考えられる．

5.4　実験 3:CPU 使用量の比較

分散ストリーム処理環境全体の CPU 使用量の分散を比較する．本実験では，CPU 使用量の比較を行うために，各計算機に対して登録されているオペレータ数を CPU 使用量と見なした．リカバリ時間制約を 100ms，高信頼化における Ack 送信間隔を 200ms と設定し，提案方式，ランダム方式，ラウンドロビン方式，上流方式の各方式において比較を行った．

5.4.1　実 験 結 果

実験結果を表 2 に示す．表 2 は，各方式により問合せの配置を行った際の，最大オペレータ配置数とオペレータ配置数の分

表 2　CPU 使用量の比較

	最大配置数	分散
提案方式	19	1.6
ランダム方式	27	16.1
ラウンドロビン方式	14	0
上流方式	28	20.1

散値である．

表 2 より，ラウンドロビン方式，提案方式，ランダム方式，上流方式の順に CPU 使用量を分散させることがわかる．ラウンドロビン方式はオペレータを，CPU 使用量の少ない計算機に対して配置する．ゆえに，各計算機に割り当てられるオペレータ数は最も平均的になる．しかしながら，実験 1，実験 2 で示したように，ラウンドロビン方式はネットワーク使用量の削減やリカバリ時間制約の充足は難しい方式である．

提案方式では，分散値 1.6 と，最もラウンドロビン方式に近い分散値を示している．提案方式では，仮想座標系を構築する際に，CPU 使用量を示す次元軸を追加する．これにより，実験 1，実験 2 で示した様な，低ネットワーク使用量，高リカバリ時間制約充足率を実現し，CPU 使用量を可能なかぎり分散させる．

ランダム方式，上流方式では配置オペレータ数に関する分散値がそれぞれ，16.1，20.1 と大きな値になっている．ランダム方式では，各計算機の CPU 使用量を一切考慮しないため，このような結果となる．また，上流方式では，上流側にあるプライマリとセカンダリが同じ計算機上に存在するため，CPU 使用量に偏りが生じやすくなる．

以上より，高信頼化を考慮した問合せ最適化を行う際には，提案方式がより有効であると考えられる．

6. まとめと今後の課題

本稿では，分散ストリーム処理環境において，単一障害点回避を考慮した問合せ最適化方式の提案を行った．本方式ではまず，仮想座標系 [5] と Relaxation アルゴリズム [9] を利用しネットワーク使用量を最小化するようなプライマリの配置を決定する．そして，プライマリの配置に基づいて，ネットワーク使用量を最小化する起点座標を決定し，起点座標を中心としてリカバリ時間制約を充足するセカンダリの配置を探索する．問合せの配置決定後，先行研究で提案した適応型 Semi-Active Standby 方式 [10] を利用して高信頼化処理を実行する．

評価実験の結果，本方式はネットワーク使用量を最小化する上流方式と同程度のネットワーク使用量を示しつつ，リカバリ時間制約充足率を 100%とすることを確認した．また，CPU 使用量に関する評価実験結果より，本方式は上流方式よりも効率的に CPU 使用量を分散させることを確認し，本方式の有効性を示した．

今後の課題として，提案方式が効率的に動作する環境，非効率的に動作する環境の考察をする必要がある．例えば，処理木が，他入力のオペレータのみから構成される場合や，オペレータの共有を許した非巡回有向グラフのような形になる場合，各方式の示す性質が変化すると考えられる．

また，本稿で提案した問合せ最適化方式と先行研究である Semi-Active Standby 方式，適応型 Semi-Active Standby 方式を含む高信頼化フレームワークの実装を考えている．そして，このフレームワークを実際の SPE と分散ストリーム処理環境に対して適用し，性能評価を行う予定である．

謝辞　本研究の一部は，科学研究費補助金基盤研究 (A)(#21240005)，科学研究費補助金特定領域研究 (#21013004) による．

文　献

[1] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.

[2] Y. Ahmad, B. Berg, U. Cetintemel, M. Humphrey, J.-H. Hwang, A. Jhingran, A. Maskey, O. Papaemmanouil, A. Rasin, N. Tatbul, W. Xing, Y. Xing, and S. Zdonik. Distributed operation in the borealis stream processing engine. In *SIGMOD*, pp. 882–884, 2005.

[3] S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, et al. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *CIDR*, pp. 269–280, 2003.

[4] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik. Scalable distributed stream processing. In *In CIDR*, 2003.

[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.*, 34:15–26, August 2004.

[6] J. Hwang, M. Balazinska, A. Rasin, U. Çetintemel, M. Stonebraker, and S. Zdonik. High-availability algorithms for distributed stream processing. In *ICDE*, pp. 779–790, 2005.

[7] J. Hwang, U. Çetintemel, and S. Zdonik. Fast and highly-available stream processing over wide area networks. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pp. 804–813, 2008.

[8] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. S. Manku, C. Olston, J. Rosenstein, and R. Varma. Query processing, approximation, and resource management in a data stream management system. In *CIDR*, pp. 245–256, 2003.

[9] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer. Network-aware operator placement for stream-processing systems. *Data Engineering, International Conference on*, 0:49, 2006.

[10] H. Shiokawa, H. Kitagawa, and H. Kawashima. A-sas: An adaptive high-availability scheme for distributed stream processing systems. In *Mobile Data Management*, pp. 413–418, 2010.

[11] Y. Xing. Dynamic load distribution in the borealis stream processor. In *In ICDE*, pp. 791–802, 2005.

[12] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for internet topology. *IEEE/ACM Trans. Netw.*, 5:770–783, December 1997.

[13] 塩川, 北川, 川島, 渡辺. 分散ストリーム処理システムにおける高信頼化方式の提案. 電子情報通信学会論文誌 D, Vol.J93-D(No.6):767–780, 06 2010.