センサノード上で動作する汎用データ管理基盤の開発

山口　卓郎† 　渡辺　陽介†† 　北川　博之†,†††

† 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: †tttakuro@kde.cs.tsukuba.ac.jp, ††watanabe@de.cs.titech.ac.jp, †††kitagawa@cs.tsukuba.ac.jp

あらまし　近年、情報源から逐次送信されるストリームデータが増加しており、アプリケーションにデータの種類を意識させることなく、ストリームデータに対する問合せ処理を実現するストリームデータ処理エンジンが開発されている。我々の研究グループでも StreamSpinner を研究・開発している。本システムには、複数のノード同士の連携によってデータ処理を実現するための、分散ストリーム処理の機能がある。これまで、ストリームデータの１つである、センサネットワークから得られるデータに対する問合せの演算処理は PC 上で行われており、各センサノードはセンシングした全てのデータを本システムが搭載されている PC へデータ転送する必要があった。しかし、センサノード上で演算処理が実行できれば、不要なデータ転送を減らせる可能性がある。そこで、本研究では各センサノードも、問合せ演算処理に利用するような分散ストリーム環境の実現を目指す。本稿では、センサノード上への StreamSpinner の搭載、またセンサネットワークの特性を考慮した分散問合せ最適化手法の提案を行う。
キーワード　データストリーム、分散問合せ最適化、分散ストリーム処理、センサネットワーク

Development of Data Management System Operating on Sensor Nodes

Takuro YAMAGUCHI†, Yousuke WATANABE††, and Hiroyuki KITAGAWA†,†††

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

Abstract　Today, stream data which is continuously delivered from information sources has been increasing. Stream processing engines for query processing to stream data have been developed for the applications using stream data. We are researching and developing a stream processing engine named StreamSpinner. It provides a distributed stream processing scheme. In our previous work, we assume our systems are deployed on commodity PCs. Sensor nodes had to send all of sensed data to StreamSpinner on PC. But, if a stream processing engine runs on each sensor node and processes sensed data in the node, we can reduce unnecessary data transfers. So we aim to develop the distributed stream environment that includes stream processing engines on sensor nodes. In this paper, we describe the implementation of StreamSpinner on sensor nodes and propose an optimization method.
Key words　data stream, ditributed query optimization, distributed stream processing, sensor network

1. はじめに

近年、センサデバイスやモバイルコンピューティング環境の発展により、実世界からリアルタイムに上がってくるデータの取得が容易になりつつある。情報源から時々刻々と変化する、逐次送信されるデータをストリームデータと呼ぶ。ストリームデータの１つとして、図 1 に示すセンサネットワーク [8] を介して取得することができるセンサデータがある。センサネットワークは温度や湿度、照度など実世界情報を電気信号に変換するセンサに CPU、メモリ、通信デバイスが搭載され、バッテリで動作する複数のセンサノードによって構成される。センサノード同士は自律的にアドホックなネットワークを形成して

いる。

アプリケーションがデータを利用する際、アプリケーション側に情報源に合わせたデータ取得の枠組みが必要となり、システム開発者の大きな負担となっている。これはセンサデータのみならず、カメラの映像データや GPS を用いた位置情報データなど他のストリームデータを用いる時も同様の問題が起こる。そこでアプリケーションがデータの形式の違いを意識することなく、ストリームデータに対する問合せ処理を実現するストリームデータ処理エンジン [1][2][3] が開発されており、我々の研究グループにおいても StreamSpinner[9] を研究・開発している。

本システムには、複数のノード同士の連携によってデータ処理を実現するための、分散ストリーム処理の機能がある [7]。これまで、センサデータに対する問合せの演算処理は PC 上で動作する StreamSpinner によって行われており、各センサノードはセンシングした全てのデータを本システムが搭載されている PC へデータ転送する必要があった。しかし、センサノード上で演算処理が実行できれば、そこで必要ないデータをフィルタリングすることができ、不要なデータ転送を減らせる可能性がある。また、PC とセンサノードにそれぞれ独立した異種のシステムが搭載されている場合より、両者に同種のシステムが搭載された方が、それぞれの特性を考慮した柔軟な問合せ最適化が行えると考えられる。

そこで、本研究では各センサノードも、問合せ演算処理に利用するような分散ストリーム環境の実現を目指す。そのためにセンサノード上に StreamSpinner を搭載し、センサネットワークの特性を考慮した分散問合せ最適化手法の提案を行う。それにより、センサネットワークから得られるデータを要求に合わせて効率的に取り出し、運用の効率化も図る。

本稿では、2. において StreamSpinner と本システムを用いた分散問合せ最適化について、3. においてセンサノードに搭載可能とした StreamSpinner の実装とその動作例について述べる。また、4. において分散問合せ処理について、5. において本稿で提案する分散問合せ最適化手法の提案について、6. において関連研究について述べ、7. においてまとめとする。

2. StreamSpinner

本節では StreamSpinner のアーキテクチャ、分散ストリーム処理、本システムを用いた分散問合せ最適化について述べる。

本システムのアーキテクチャを図 2 に示す。本システムはラッパーと呼ばれるモジュールによって各情報源が持つ固有のデータ形式をリレーションのタプル形式に変換し、メディエータにデータを送信できるようにする。アプリケーションが問合せを登録すると、それに基づいた処理木が算出され、処理プランが決定される。データの到着や時間の経過といったイベントに連動して、メディエータは処理木に基づいた演算評価を行い、処理結果をアプリケーションに返す。

次に、分散ストリーム処理について述べる。

分散ストリーム処理とは、カメラやセンサなど実世界ストリーム情報源が地理的に分散して存在することが多いため、効

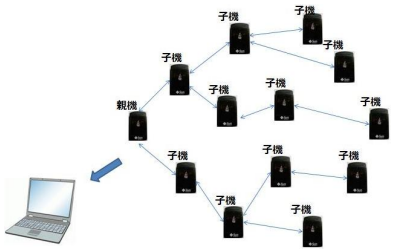


図 1　センサネットワーク

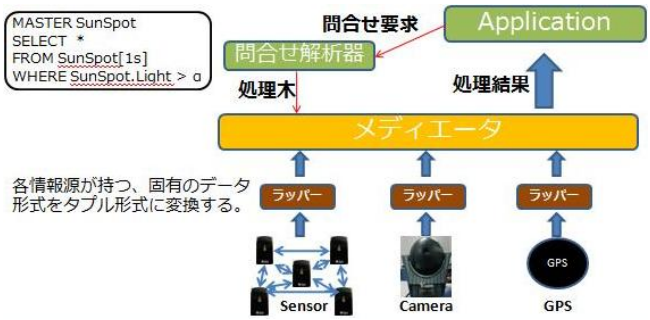


図 2　StreamSpinner のアーキテクチャ

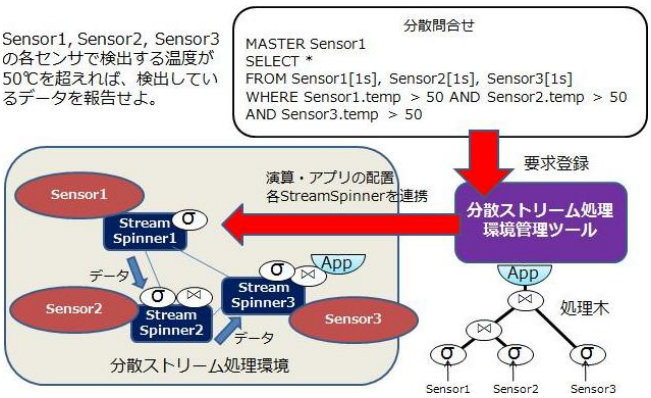


図 3　分散ストリーム処理の流れ

率的な処理のためにストリーム処理エンジンを地理的に分散配置し、各エンジンが連携して分散ストリーム処理環境全体で効率的な処理を行おうとするものである。我々の研究グループの先行研究 [7] では StreamSpinner を用いて分散ストリーム処理を行っている。処理の流れを図 3 に示す。

分散ストリーム環境には StreamSpinner が搭載された PC が複数存在し、各ノードはセンサデータを取得している。ユーザは分散問合せを分散ストリーム処理環境管理ツールに登録する。分散問合せ要求の処理に対しては、連続的問合せを用いている。これはストリームデータが到達する度に演算を評価し、直前までの処理結果との差分を求めて、その結果を返すというものである。分散問合せ要求の記述は、一部異なる部分はあるものの基本的には SQL に即している。異なるのは MASTER 節が新たに設けられていることと FROM 節にウインドウサイズを指定できることである。MASTER 節は、そこで指定されたストリームデータを受信したときに処理の実行を開始するということを意味する。ウインドウサイズは、処理を実行する際にどの時間範囲のデータを処理する対象に捉えるか指定するも

のである。

図中の分散問合せの意味は Sensor1, Sensor2, Sensor3 の各センサで検出する温度が 50 ℃を超えれば、検出しているデータを報告せよというものである。問合せ登録を受けた分散ストリーム処理環境管理ツールは問合せを基に処理木を作成し、その処理木に沿って演算とアプリケーションを各ノードに配置する。図中の演算配置と分散問合せでは、StreamSpinner1 が Sensor1 から受け取ったデータにフィルタリングをかけ温度が 50 ℃を超えた時の測定データを StreamSpinner2 に送信する。StreamSpinner2 は Sensor2 から受け取ったデータにフィルタリングをかけ温度が 50 ℃を超えた時の測定データを StreamSpinner1 から受け取ったデータと結合処理を行い、結果を StreamSpinner3 に送信する。StreamSpinner3 は Sensor3 から受け取ったデータにフィルタリングをかけ、温度が 50 ℃を超えた時の測定データを StreamSpinner2 から受け取ったデータと結合処理を行い、結果をアプリケーションに渡す、という処理を行っている。

演算と配置先ノードの組合せは複数あるが、その中からネットワーク帯域使用量などのコスト関数を最小化するものを探す処理を分散問合せ最適化という。それによって、通信に用いるネットワークの負荷が最も低くなると見積もることができる組合せを選んでいる。

本稿では、これまでの分散ストリーム処理の枠組みをベースに、PC だけでなくセンサーノード自身をも StreamSpinner の動作ノードとして考慮したものへと拡張する。

3. StreamSpinner のセンサノードへの搭載

本節では StreamSpinner のセンサノードへの搭載に当たる工夫、そしてその動作の確認について述べる。

本システムの搭載には Sun Microsystems の SunSPOT[8] を用いる。SunSPOT の仕様を図 4 に示す。開発に Java 言語を用いることができ、温度・照度・3 軸加速度を測定可能で、個々のノードに固有の Mac アドレスを持つ。SunSPOT には 2 種類あり、それぞれを親機と子機と呼ぶ。親機は基本的に PC と USB 接続して使用するもので、子機から送られたデータを受信して、それをそのまま PC に送る役割を果たす。子機はバッテリを搭載しており、親機との通信が可能な範囲であればどこでもデータをセンシングして、親機にそのデータを送信することができる。親機との通信が困難な距離にあっても、他の子機を経由して通信を行うマルチホップ通信が可能である。親機と子機は基本的に同じ基盤を使用しており、異なるのは子機には存在する複数のセンサを搭載したセンサボードとバッテリが親機には搭載されていないことである。

搭載するにあたり、Java6.0 レベルで実装されている StreamSpinner を J2ME レベルのソースコードへの変換を行った。そのために、Java5.0 と 6.0 で新たに採用された機能と J2ME に非対応のライブラリ使用部分の再実装を行った。また、SunSPOT がもつセンサから取得したデータをリレーションへ変換を行うラッパーの開発を行った。図 5 に表すように、SunSPOT のセンサから取り込んだデータのリレーショナルスキーマは (Time Stamp, Temp, Light, AccelX, AccelY, AccelZ) となる。完成したシステムを StreamSpinner on SunSPOT と呼ぶ。現在の実装では、選択演算、結合演算、射影演算が処理可能である。

- プロセッサボード仕様
  - 180MHz 32ビット ARM920T コア
  - 512KB RAM/4MB フラッシュメモリー
  - 2.4GHz IEEE 802.15.4無線（アンテナ付）
  - USBインタフェース
  - 3.7V 720mAh リチウムイオンバッテリー
- センサボード仕様
  - 2G/6G 3軸加速度センサー
  - 温度センサー
  - 照度センサー

図 4　SunSPOT 仕様

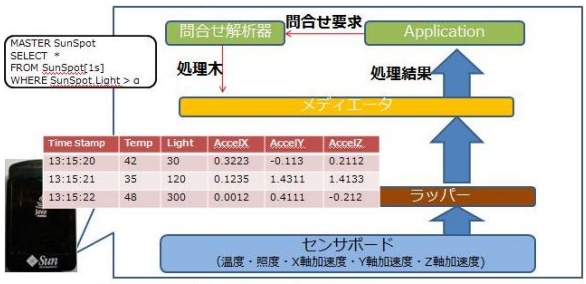


図 5　StreamSpinner on SunSPOT

典型的な利用例を図 5 に示す。例として図中の問合せを StreamSpinner on SunSPOT に登録すると、周囲が暗い間はデータ送信を行っていないが、観測している照度が閾値 α を超えたとき、測定しているデータの値を送信するなど、選択演算にあたる処理を正しく行えていることを確認した。

予備実験として選択演算を含む問合せを登録して、子機が親機にデータを送信する回数を減らすことで、どれだけバッテリの寿命に違いがあるかを確認した。センサノードはバッテリで動作しており、消費電力が大きいと充電やバッテリ交換などのコストが発生する。そのため、できるだけ消費電力を少なくすることが求められている。センシングした全てのデータをフィルタリングして送信するノードと、センシングした全てのデータを送信するノード同士でバッテリの持ちを比較した。

双方のノードを満充電にして同時にセンシングを開始させたところ、後者は 13 時間 12 分ほどでバッテリがなくなり動作しなくなったのに対し、前者は 40 時間以上動作し続けた。不必要なデータをセンサノード上で選択して送信することでセンサノードは消費電力を大幅に削減することがわかった。

4. 分散問合せ処理

本節では、PC で動作する StreamSpinner に加え、センサノード上の StreamSpinner on SunSPOT が混在する環境における分散問合せ処理について述べる。

まず、本稿で提案するシステムにおいて想定している状況を述べる。ここでは、部屋などの場所に備え付けのセンサノードと、人や物に付いて一緒に移動するセンサノードが複数のエリアに混在するような環境を想定する。そして通信可能な範囲にいるセンサノード同士でアドホックネットワークを形成する。その上流には PC 上で動作する StreamSpinner が存在する。ま

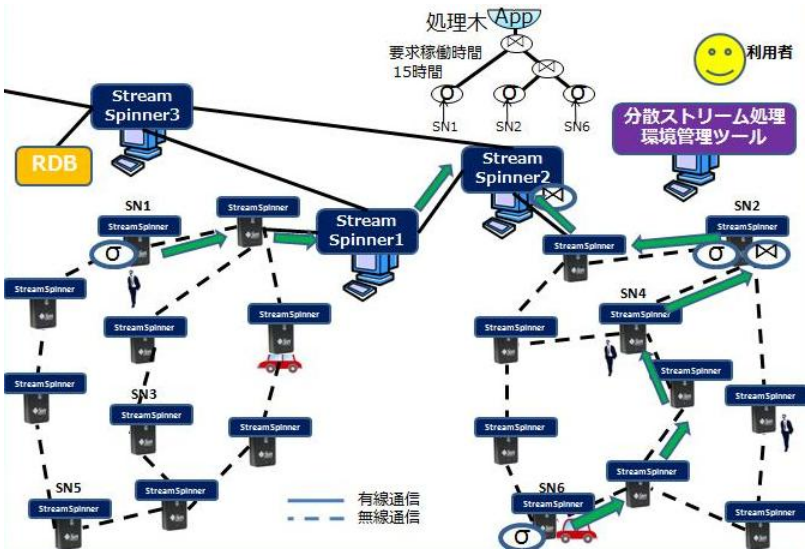


図 6　センサノードも含めた分散問合せ最適化

た、StreamSpinner はラッパーを介し他の RDB のデータを取り込むことが可能であり、それらも問合せの対象とみなすことができる。

図 6 で表しているように、センサノードには SN1 や SN2 のようにノード番号が割り当てられており、ユーザはどこに何番のノードがあると把握できるものとする。センサノードでは温度・照度・3 軸加速度が測定可能であり、ユーザはそれに基づき分散ストリームを収集するための問合せ要求をすることができる。

ただ、センサノードはバッテリ駆動であり、本稿ではその位置はダイナミックに変わるものと想定している。よって、演算が配置されたノードにバッテリ切れで通信できなくなったり、通信を行う周りのノードが変わることがある。これらの問題に対するアプローチは 5 節で述べる。

稼働可能時間を満たすノードを探す上で必要な入力情報として、ユーザは「これだけの時間は動き続けて欲しい」という要求稼働時間を指定する。その為に LIFETIME 節を導入する。問合せ例を図 7 に示す。LIFETIME 節は時間によって定義することができる。図 7 の問合せではノード番号 1, 2, 6 のノードで得られるデータが 10 ℃を超えたら報告するという処理を、15 時間継続して欲しいという要求である。

この問合せを分散ストリーム処理環境管理ツールが受け取った後、先の問合せを演算単位に分割し、各ノードに演算を配置した後、各ノードを連携処理させることで分散ストリーム処理を行う。図 7 の問合せでは、各センサノードでデータが取得される度に演算の評価が実行され、処理結果が次の演算のあるノードへ転送される。

また、問題を簡略化するため、本研究では問合せ実行タイミングを与える MASTER 節が FROM 節と同一である問合せを対象とし、それ以外の外部からのイベントをトリガとする問合せについては扱わないものとする。

```
MASTER  SN1, SN2, SN6
  SELECT ＊
    FROM  SN1[1s], SN2[1s], SN6[1s]
  WHERE  SN1.Temp > 10
     AND  SN2.Temp > 10 AND SN6.Temp > 10
LIFETIME 15h
```

図 7　分散問合せの例

5.　分散問合せ最適化手法の提案

本節では PC で動作する StreamSpinner に加え、センサノード上の StreamSpinner まで対象にした分散問合せ最適化手法について述べる。

[7] では固定ネットワーク、PC での動作が前提での最適化手法であった。しかし、本研究ではセンサノードを演算の配置対象に含めているため、それに伴う以下の問題を扱う必要がある。

- バッテリ切れや通信不安定によって演算が配置されたノードに通信できない
- センサノードの位置がダイナミックに変わり、最適な演算配置場所が変化する

以降では、これらの問題点に対する本研究のアプローチについて述べる。

5.1　最適化の基本方針

最適化の基本方針を以下に示す。

- ネットワーク帯域使用量を極力抑えられる演算配置を選ぶ
- センサノードの移動やバッテリ寿命などの影響を受けずに一定期間は処理が続けられる演算配置を選ぶ

最適化のために、データレートやネットワーク遅延、各センサノードの移動までの時間、バッテリ情報などの統計情報を用い、ネットワーク帯域使用量が低く、かつ、一定期間処理を続けることができる演算の配置プランを選択する。まず、5.2 で最適化に必要な統計情報の収集について述べ、次に 5.3 で演算

の配置プラン選択方法、最後に 5.4 で演算の再配置をいつ行うのか、そのタイミングについて述べる。

5.2 最適化に必要な統計情報の収集

最適化に必要な各統計情報の収集について述べる。

入力データレートとネットワーク遅延は、ネットワーク帯域使用量を算出するために収集される。各センサノードの移動履歴と稼働可能時間は、よく移動するノードやバッテリ寿命の短くなっているノードへの演算配置を避けるために収集する。これらの情報は、各ノードが定期的に取得し、最新の値を分散ストリーム処理管理ツールにノード番号とともに報告するものとする。

5.2.1 入力データレート

データレートは単位時間当たりに流れるデータ量のことである。入力データレートは StreamSpinner on SunSPOT がセンサデータを取得する間隔で決定される。例えば、10 秒に 1 回、データをセンシングして 1 タプルにする場合、それによって毎分 6 タプルが生成され、データレートは平均 6 タプル/分となる。データレートは各センサノードが独立に設定し、1 度決定した値は電池が切れるまで変わることはないため、固定値とする。

5.2.2 ネットワーク遅延

本研究におけるネットワーク遅延は、StreamSpinner の動作するノード間の通信において発生する遅延時間とする。本研究では、通信するノードの種別ごとに異なる方針でネットワーク遅延を見積もる。

（1） PC ノード間のネットワーク遅延：実際にパケットを送信することで測定することができる。

（2） 親機子機の関係にある PC ノードとセンサノード間の遅延：分散ストリーム処理環境管理ツールがデータを処理する際には、比較的高頻度に通信が行われる。センサノードのバッテリを節約する観点から新たにパケットを送信してネットワーク遅延を求めるのではなく、行われている通信から遅延を見積もる。

（3） 親機子機の関係にはない PC ノードとセンサノード間の遅延：(1) と (2) のネットワーク遅延の値の和によって間接的に見積もる。それによって、センサノードのバッテリも節約することができる。

（4） 同じエリア内のセンサノード間の遅延：ここでは、一連の無線アドホックネットワークによって子機同士が通信可能な範囲のことをエリアと呼んでいる。子機同士でパケット通信を行うことで遅延を求めることも可能であるが、子機同士の組み合わせが多いと、遅延の測定のために必要以上にバッテリを消費してしまうと考えられる。その遅延は各センサノードと親機間の遅延時間を基に見積もる。例えばセンサノード A-センサノード B 間の遅延は、センサノード A-親機+親機-センサノード B の遅延の値を用いる。

（5） 異なるエリア内のセンサノード間の遅延：各エリアの (2) のネットワーク遅延と (1) のネットワーク遅延の 3 つの値の和によって求めることができる。

5.2.3 各センサノードの移動履歴

本研究で考慮しなければならないノードの移動とは、周辺のノードとの通信状況を大きく変化させるような場合だけである。特に、別のエリアにあるアドホックネットワークへ参加するような移動を検出の対象とする。これは無線通信可能な親機の通信状態を定期的に監視することで、実現可能であると考えられる。移動した先で周辺ノードから新たな親機の情報を取得する。

センサノードの移動履歴は次の移動までの時間の見積もりに使用する。これは過去によく移動しているセンサノードは、今後もよく移動する可能性が高いという推測に基づく。近隣のノードと通信を開始したときからの経過時間をノード番号とともに分散ストリーム処理環境管理ツールに報告する。よく移動するセンサノードほど、移動までの平均時間が少ないといえる。

5.2.4 各センサノードの稼働可能時間

SunSPOT にはバッテリ残量を測定するメソッドが存在し、残量は何パーセントという情報を得ることができる。満充電で N 時間動作することができると仮定すると、残量が 50 %のとき N/2 時間動作することがわかる。バッテリ残量をノード番号とともに定期的に分散ストリーム処理環境管理ツールに報告する。

5.3 配置プラン選択方法

5.2 で述べた通り、最適化に必要な統計情報はノード番号と共に分散ストリーム処理環境管理ツールに送信される。管理ツールは、ユーザから与えられた問合せを基に統計情報と併せて配置プランを作成していく。

まず、問合せに対応する演算の処理木を構築する。演算の実行順序は分散環境ではない通常の問合せ最適化手法 [12] 等によって求めるものとする。生成された処理木を基に、各演算をどこのノードへ配置すればよいかを以下の手順で決定する。

（1） センサノードおよび通常のノードを含む全ノードのうち、次の移動までの平均時間が短くよく移動していると判断できるノードと、バッテリ寿命が LIFETIME 節によって要求される稼働可能時間を満たさないノードを配置先の候補から外す。

（2） 残った候補ノードと演算の組み合わせの中から、ネットワーク帯域使用量を最小化する配置プランを選択し、演算の配置を行う。

各配置プランにおけるネットワーク帯域使用量の見積りは、従来の分散ストリーム処理の最適化 [11] でも用いられていた式 1 を用いる。

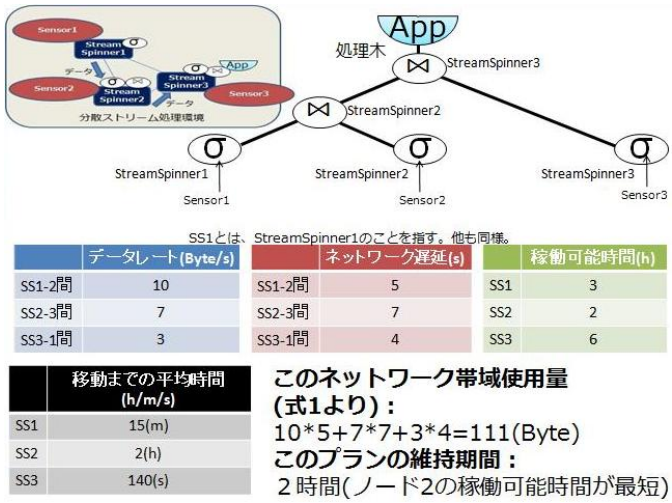
$$u(q) = \sum_{l \in \mathbf{L}} DR(l) Lat(l) \qquad \text{(式 1)}$$

L : 問合せ処理のデータ転送に必要なノード間の回線の集合

$DR(l)$: 回線 l における入力データレート

$Lat(l)$: 回線 l におけるネットワーク遅延

これにより、入力レート情報とネットワーク遅延情報を基に、ネットワーク帯域使用量を最小とする演算配置プランを導出することができる。本研究では、従来の分散ストリーム処理と同



SS1とは、StreamSpinner1のことを指す。他も同様。

	データレート(Byte/s)
SS1-2間	10
SS2-3間	7
SS3-1間	3

	ネットワーク遅延(s)
SS1-2間	5
SS2-3間	7
SS3-1間	4

	稼働可能時間(h)
SS1	3
SS2	2
SS3	6

	移動までの平均時間 (h/m/s)
SS1	15(m)
SS2	2(h)
SS3	140(s)

このネットワーク帯域使用量 (式1より)：
10*5+7*7+3*4=111(Byte)
このプランの維持期間：
２時間(ノード2の稼働可能時間が最短)

図 8　コスト算出例

じネットワーク帯域使用量の算出式を使用しているが、よりセンサネットの特性を考慮するならば、別の算出式を用いることも検討すべきである。それについては今後の課題とする。

図 8 にコスト算出例を示す。このプランにおける統計情報は下の表 4 つに記載された通りである。これを基に式 1 を用いてネットワーク帯域使用量を求めることができる。また、稼働可能時間が最短のノードの時間によってプランの維持期間が求められる。また、このネットワークにおいては移動までの平均時間が 3 時間を下回るノードには演算を配置しないとしているが、今回は全てのノードがその閾値を上回っているため、演算の配置対象としている。

5.4　演算の再配置タイミング

先に述べた通りノードはダイナミックに移動するため、定期的に演算配置を見直す必要がある。本来はノードの移動が確認された時点で演算配置を見直すことが最善ではあるが、何度も再配置を行うことでより多くの電力を消費してしまう。

そこで、前回の配置から一定数以上のノード移動を検出したり、一定の時間が経過したときに再配置要求を管理ツールが出して、再配置を行う。

6. 関 連 研 究

本節では本研究に関連するストリーム処理エンジンやセンサデータ収集の枠組みについて、関連研究を紹介する。

- Aurora[1], STREAM[3], TelegraphCQ[2]:1 台のマシンで動作する集中型のストリーム処理エンジンであり、分散ストリーム処理は考慮されていない。
- Coral8[10]:商用ストリーム処理エンジンであり、単位時間当たりの処理可能データ数が多く、高速処理が可能と謳われている。株式トレーディングなどに利用することを想定している。
- Borealis[4]:分散型ストリーム処理エンジンであるが、分散環境中にアプリケーションを組み込むための明確な枠組みはない。またセンサノード上で稼働していない。
- TinyDB[6]:センサネットワークを 1 つの大きなリレーションとしてみなすことができる。しかし分散ストリーム処理のためにノード毎に異なる演算を配置することはできない。また下流から上がってくる不要な子機データを上流の親機で集約する際に処理するのに対して、本研究では、データを持つ子機上で選択演算して余分なデータはその子機から送信しないなど、不要データ処理に関するアプローチが異なっている
- Abadi らは Borealis と TinyDB を組み合わせた枠組みを提案している [5]。ただし、Borealis の分散問合せ最適化ではセンサノードは対象にされておらず、センサネットワーク内のデータ収集方法は、全て TinyDB に依存している。彼らの枠組みでは、センサネットワークからデータを収集する要求の定義を QoS によって設定できる。この仕組みにより、バッテリ寿命を気にせずサンプリング収集間隔を狭めてデータを収集したり、サンプリング収集間隔を広げてバッテリ寿命を延ばしながらデータを収集したり、といった指定が可能である。収集は power、latency、quality の各条件を、ユーザやアプリが設定する lifetime という寿命を満たすように設定して行われる。これに対し本研究では、PC とセンサノードの両方に同種のシステムを搭載することで、それぞれの特性を考慮した問合せ最適化を実現している。問合せに基づいて、生成された処理木の演算を LIFETIME 節の条件を満たすノードの中で最もコストがかからないプランを選んで配置している。

7. ま と め

センサノードと PC 上で稼働する StreamSpinner が連携して、分散問合せ最適化を行うことでセンサネットワークから取得可能なデータの処理の高度化を図ることができる。実際に StreamSpinner をセンサノードに搭載し、最適化手法を提案した。今後、提案手法の実装と評価、手法の拡張を行っていく。

謝辞　本研究の一部は、科学研究費補助金 (#18200005) による。

文　　献

[1] D. J. Abadi, et al., ”Aurora: a new model and architecture for data stream management ”, VLDB Journal Vol.12, No.2, pp.120-139,2003.

[2] S. Chandrasekaran, et al., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World", Proc. CIDR, 2003.
[3] R. Motwani, et al., "Query Processing, Resource management, and Approximation in a Data Stream Management System", Proc. CIDR, 2003.
[4] Daniel J. Abadi, et al., "The Design of the Borealis Stream Processing Engine", Proc. CIDR, 2005.
[5] Daniel J. Abadi, et al., "An Integration Framework for Sensor Networks and Data Stream Management Systems", Proc. VLDB, 2004.
[6] S. R. Madden, et al., "TinyDB: An Acquisitional Query Processing System for Sensor Networks", ACM Transactions on Database Systems, Vol.30, No.1, pp.122-173, March 2005.
[7] 稲守孝之. 渡辺陽介. 北川博之. 天笠俊之. 川島英之., 広域分散ストリーム処理環境における演算配置最適化手法の評価, DEWS2008.
[8] Sun Microsystems/SunSPOT. http://jp.sun.com/products/software/sunspot/
[9] StreamSpinner. http://www.streamspinner.org
[10] Coral8. http://www.coral8.com/
[11] Peter R. Pietzuch, Jonathan Ledlie, Jeffrey Shneidman, Mema Roussopoulos, Matt Welsh and Margo I. Seltzer. "Network-Aware Operator Placement for Stream-Processing Systems" Proc. ICDE, p.49, 2006.
[12] Ahmed Ayad, Jeffrey F. Naughton, "Static Optimization of Conjunctive Queries with Sliding Windows Over Infinite Streams", Proc. SIGMOD, pp. 419-430, 2004.