スレッド分割の適正化による交通情報生成の高速化

今井 照之 藤山 健一郎 喜田 弘司 中村 暢達
NEC サービスプラットフォーム研究所

1 はじめに

車両から位置や速度等を収集し，交通情報を生成するプローブ情報システムが注目されている．現在のプローブ情報システムは地域や車両を限定し，数分毎に数千台から1万台の車両データを処理している．今後，全国レベルで実用化するために，より多くの道路や車両の情報を高速に処理する必要がある．例えば，東京都下の営業車だけでも15万台，都道以上の道路を100m単位に区切ると5.4万区間となる．本稿では，東京都下規模の車両データを1台の計算機で処理可能な交通情報生成処理の高速化について述べる．

2 課題分析

2.1 プローブ情報システム

従来のプローブ情報システムの構成を図1に示す．通常，収集，リンクマッチング，道路別集約，渋滞判定，出力の順に処理がなされる．収集では車両からの位置，速度(センサデータ)を収集する．リンクマッチングでは，車両の位置から，走行中の道路区間のID(リンクID)を判定する．センサデータと対応するリンクIDも併せて車両情報と呼ぶ．道路別集約では，道路区間毎に車両情報をまとめ，全ての道路区間の平均速度，走行台数などの状況を求める．渋滞判定では，各道路区間の状況から，その道路が渋滞している度合を判定する．結果出力では，渋滞判定の結果を利用者に提供する．

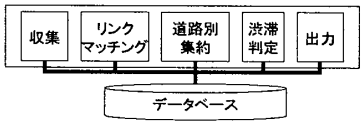

図1: 従来のプローブ情報システムの構成

従来，各処理結果の中間データをデータベース(DB)に蓄積し，必要に応じてDBからデータを取り出して処理していた．しかし，DBを用いた方式は，以下の理由から大規模化が困難である．第一に，データをまとめて処理すると扱うデータ量が大きくなるので，大量のリソース，ディスクI/Oが必要となる．第二に，大量のデータを蓄積し続けるためディスクが溢れる．

Faster Production of Traffic Information by Inter-Thread Flow Control
Teruyuki Imai, Ken-ichiro Fujiyama, Koji Kida, and Nobutatsu Nakamura
Service Platforms Research Laboratories, NEC Corporation

2.2 データストリーム処理

これらの課題に対し，処理をパイプライン化し，データをDBに蓄積せず，発生毎に順次処理するデータストリーム処理に関する研究に取り組んでいる[1]．データストリーム処理では，処理全体を独立したスレッドに分割し，スレッド間は，キューを介して接続される．各スレッドの処理は，入力されたデータを蓄積せず順次メモリ上で処理し，結果を次のキューに入力する．処理の済んだデータは基本的に破棄する．

プローブ情報システムにデータストリーム処理を適用した例を図2に示す．

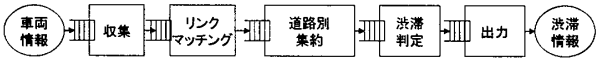

図2: データストリーム処理型プローブ情報システムの構成

データストリーム処理は，第一の課題に対しては，発生したデータのみを分割されたスレッド単位で順次処理するため，必要なリソースが少なく，メモリ上で高速に処理できる．第二の課題に対しては，データを蓄積せず破棄するため，ディスク溢れを回避できる．

2.3 データストリーム処理の課題

従来のDBを用いたシステムに単純にデータストリーム処理を適用すると，データの流れに停滞が生じる．負荷の高いスレッドにおいては，そのスレッドの次の処理が開始するまでにキューに入力されるデータ数が増大し，キューが溢れる．キュー溢れは，スワップを引き起こし，処理速度が低下する．

スレッドを駆動するトリガーには，データの到着数に応じたデータ駆動型と，一定時間間隔の時間駆動型とがある．時間駆動型の方が，入力されるデータ数が不確定であるため，上記の問題が発生しやすい．

3 スレッド分割の適正化

前節の課題に対応するためには，各スレッドの処理負荷を小さくし，できるだけデータ駆動型となるようにスレッドを分割することが重要である．本稿では，時間駆動型スレッドに対してより適切なスレッド分割を行うことを提案し，その性能向上効果について述べる．以下，プローブ情報システムへの適用を例に説明する．

道路別集約は，2.1に示したように，車両情報の発生毎に頻繁に各道路の状況を更新する個別状況更新処理の後，一定時間経過毎にその時点の全ての道路の状況情報を生成する全状況生成処理を行う．2つの処理は

直列に実行されるため，全状況生成処理の実行中は個別状況更新処理が行われず，新たに発生した車両情報はキューに蓄積される．処理する道路区間の数が増加すると，全状況生成に掛かる時間が増大するため，より多くの車両情報がキューに蓄積される．我々の実験では，6 万区間の道路ではメモリが溢れ，処理不能に陥った．

そこで，道路別集約処理を，図 3 のようにデータ駆動型の個別状況更新と時間駆動型の全状況生成に分割し，発生し続ける車両情報に対して，キューに入ってから次に全状況生成処理の駆動される時まで待たずに個別状況更新を非同期に処理できるようにする．

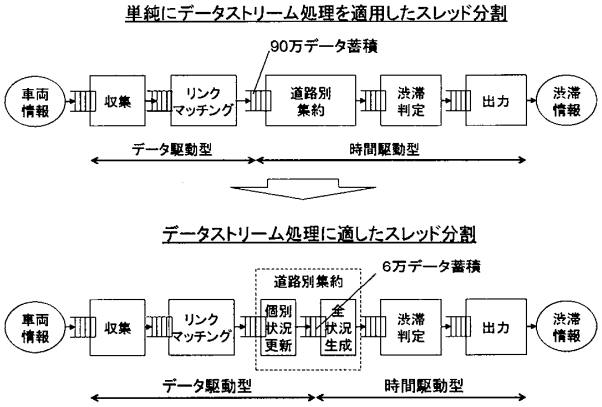

図 3: 処理分割 (スレッド構成) の適正化

東京都下を想定した表 1 の動作条件で，15 万台の車両情報を処理する場合を例に提案手法の効果を述べる．単純にデータストリーム処理を適用すると，15 万データ × (60 秒/10 秒) = 90 万データ が道路別集約のキューに停滞する．これに対し，スレッドを前記のように分割すると，個別状況更新処理はデータ駆動型であるためデータの流れは停滞しない．個別状況更新処理は 15 万台分の車両情報を 6 万道路区間の状況に変換し，全状況生成のキューに出力する．ただし，キュー内に同じ道路区間のデータがあればそれを更新する．よって，蓄積されるデータ数は入力される車両情報の数にかかわらず 6 万である．このように，キューに蓄積されるデータ量を削減できる．よって，従来に比べ，大量の車両情報を高頻度に処理可能となる．

表 1: 想定する動作条件

道路区間数	6 万区間
センサデータの収集間隔	10 秒
渋滞情報の生成間隔	60 秒

典型的な統計処理，つまり一定時間毎のデータに対する個数や平均などの統計量を求める処理は，個々のデータに対して逐次的に処理できる統計計算と，時間単位の結果を集計する処理からなる．そのため，一般的な統計処理は，データ駆動型の統計計算スレッドと，時間駆動型の集計スレッドに分割可能である。

統計計算の結果発生するデータ量はデータ駆動型の統計計算スレッドのキューに入力されるデータ量より少ないため，統計計算と集計に分割するとキューに蓄えられるデータ量は減少する．したがって，メモリ溢れを回避し，大規模なデータを処理可能となる．

4 評価

スレッド分割の適正化による交通情報生成の高速化を評価する．

4.1 評価方法

東京都下における交通情報生成を想定し，表 1 に示す条件で提案方式によりスレッド分割を適正化したプローブ情報システムが 1 台の PC 上で扱える車両台数の限界値を求める．

4.2 結果

評価の結果，単純にデータストリーム処理を適用した方式では 2 万台のデータを扱えなかった．一方，提案方式では図 4 に示すように，14 万台までの車両が扱え，15 万台の車両はメモリ不足のため扱えなかった．提案方式により東京都下規模の交通情報生成が可能な処理速度が得られることが示せた．

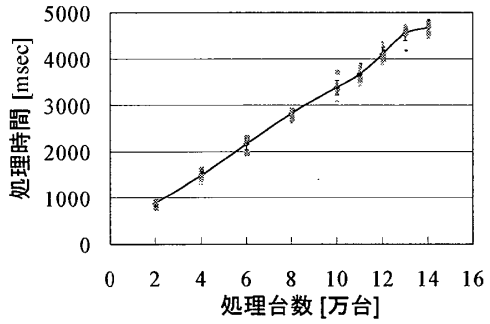

図 4: 提案システムの処理性能

5 おわりに

データストリーム処理では，データ処理の流れの停滞を防ぐ必要がある．本稿では，スレッド分割を適正化し，データ駆動で処理可能な部分を非同期化することでメモリ溢れを回避し，大量の車両情報を高速に処理する手法を述べた．さらに，交通情報生成処理を対象に，道路状況生成を高速化し，東京都下に相当する交通状況情報の生成が可能であることを示した．

参考文献

[1] 喜田弘司 他：データストリーム処理による大規模プローブカーシステムの開発と評価, 情処研報 ITS, No. 34 (2008).