Vol.2011-ITS-45 No.4
2011/6/24

ストリーム処理を用いた車々間通信データのフィルタリング方式

勝沼聡[†]　山田真大[†]　本田晋也[†]　佐藤健哉[†,††]
高田広章[†]

車両統合制御アプリケーションで活用される車々間通信データは，周辺車両の状況によりデータ量が大きく変化する．したがってストリーム処理を用いた車載データ統合アーキテクチャでは，車々間通信データをフィルタリングすることでデータ量を一定以内に抑えることが課題となる．本稿で提案するフィルタリング方式では，ストリーム処理を用いてデータの重要度を様々な指標で定義可能とする．そしてストリーム処理により，アプリケーションの動作周期に同期し，重要度に従ってアプリケーションへのデータが一定件数になるように抽出する．フィルタリング方式を実装し評価した結果，処理遅延を抑えることを確認し，ストリーム処理による車々間通信データのフィルタリングを実現できることを示した．

Stream Processing Method for Filtering Data Received by Inter-Vehicle Communications

Satoshi KATSUNUMA[†]　Masahiro YAMADA[†]
Shinya HONDA[†]　Kenya SATO[†,††]
and Hiroaki TAKADA[†]

In vehicle integrated control systems, the number of data received by inter-vehicle communications changes widely. Therefore, in data integration architecture for automotive, this stream processing method needs to filter these data. In this paper, we propose the stream processing method for filtering data received by inter-vehicle communications. This technique defines important degree of these data with the stream processing method. In the light of the important degree, this technique maintains the number of these data constant in one cycle. We implement this proposed technique and evaluate its latency. Experimental result shows that the latency of this technique is low.

1. はじめに

近年，プリクラッシュセーフティ技術など，車両の状態や周辺状況を判断し，ドライバへの警告や自動制御により運転の支援を行う車両統合制御アプリケーションが登場している 1)．例えば車両に搭載された複数のセンサからの情報に基づき，操舵回避の支援を行い，衝突が避けられない状況では介入ブレーキを作動させることで衝突衝撃を緩和し被害を軽減するアプリケーションがある 2)3)．また車両追従，レーン逸脱警告，自動駐車アプリケーションなどもある 4)．車両統合制御アプリケーションにおいて，周辺の物体を検知するためにミリ波レーダやレーザレーダ，カメラを始め車輪速センサ，加速度センサ，位置検出センサなど多様なセンサを複数搭載し，車々間通信や路車間通信の利用も加わって，相互に情報交換を行う必要がある．

このような車載システムのアプリケーション（以下，車載アプリケーション）では，センサやアプリケーションの増加に伴い，システムの設計，開発に要するコストが増加している．そのため複数アプリケーションの処理を統一化し開発コストを削減する様々なアプローチが検討されている 5)6)7)．我々のプロジェクトではアプリケーションからセンサ部を切り離し，センサから得られたデータを統合管理し，複数のアプリケーションから共通に利用可能とする車載データ統合アーキテクチャ 8)を検討している．車載データ統合アーキテクチャではストリームとしてデータを管理する 9)10)．これにより様々なデータ処理の共有化によるコスト削減及び，そのデータ処理結果の高速な配信を可能にする．

車載統合アーキテクチャでは，アプリケーションのリアルタイム性を確保するために，高頻度に発生する通信データやセンサデータへの対応が求められる．特に車々間通信データは周辺車両の状況によりデータ量が大きく変化し，また車々間通信データを活用したアプリケーションとして，衝突検知 18)等，高いリアルタイム性を要求する処理が想定されるため，データのフィルタリングが必要となる．

データの削減のためにストリーム処理のためのフィルタリングとしてロードシェディング方式 12)13)14)15)16)が提案されている．しかしロードシェディング方式は，株自動取引や電子マネー等の汎用システムの，逐次的に実行するアプリケーションへの適用を想定している．したがって周期的に実行する車載アプリケーションに対して，配信するデータ量を保証するのが難しい．また重要度の高いデータを削除する可能性がある．

そこで本稿では，車々間通信データのフィルタリングの要件を満たす，Periodic

[†]名古屋大学大学院情報科学研究科附属組込みシステム研究センター
Center for Embedded Computing Systems, Nagoya University
[††]同志社大学理工学部情報システムデザイン学科
Department of Information Systems Design, Doshisha University

Priority-Base Filtering (以下，PPBF)を提案する．PPBF ではアプリケーションの動作周期を考慮し，重要度の指標に従ってデータを取捨選択しつつ，アプリケーションに配信するデータを一定件数に抑制する．またデータの重要度について，ストリーム処理を用いることによりアプリケーション毎の様々な指標で定義可能にする．

本稿の構成は以下の通りである．まず 2 節で車載データ統合アーキテクチャの課題について述べる．そして，次に 3 節において本稿で提案する PPBF を説明し，4 節でその評価について述べる．5 節で関連研究をまとめ，最後に 6 節でまとめと今後の課題を述べる．

2. 車載データ統合アーキテクチャの課題

2.1 車載データ統合アーキテクチャの概要

我々が検討している車載データ統合アーキテクチャの概略について図 1 に示す．車載データ統合アーキテクチャでは，複数の車載アプリケーションにおいて別個に管理されていたセンサデータや車々間，路車間の通信データの処理を統合管理する．これにより様々なアプリケーションでデータ処理を共通利用可能にする．例えば図 1 では，アプリⅠ～Ⅲに対して，従来，各アプリケーションで行われていた位置や周辺他車状況，周辺状況の算出処理をアプリケーションから切り離し，車載データ処理として統合管理する． これにより例えば，位置情報をアプリⅠに配信すると共に，アプリⅡ，Ⅲで必要とする周辺他車や，周辺状況の算出に活用するなど，データの共通利用が可能になる．車載データ統合アーキテクチャでは，車載データをストリームとして管理する．これにより従来のリレーショナルデータベース（RDB）とは異なり，高頻度で到着するデータの高速処理が各アプリケーションで共通利用可能になる．次節ではストリーム処理について説明する．

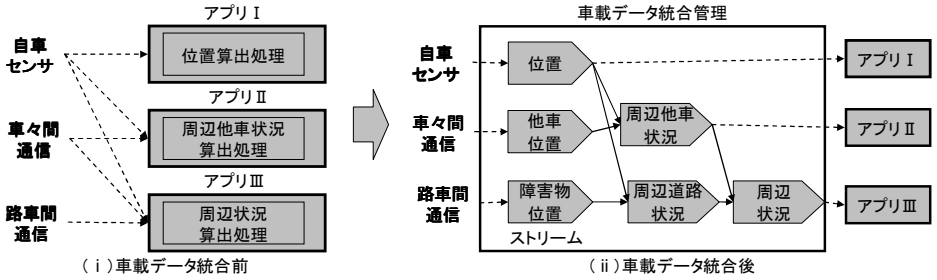

図 1　車載データ統合アーキテクチャ

2.2 ストリーム処理

株自動取引，電子マネー，車両位置，携帯操作などを扱う汎用システムに向けてストリーム処理が検討されている．本節ではこのような汎用システム向けの一般的なストリーム処理の概要及び，スケジューリング方法について説明する．

2.2.1 ストリーム処理の概要

ストリーム処理は，センサデータのような更新頻度の高いデータをアプリケーションが共通に利用可能とするデータ管理システムである．ストリーム処理は従来の RDB と異なり，過去に入力されたセンサデータをメモリ上に保持し，新たなデータが入力される度に保持したデータを更新し，各アプリケーションに配信する．これにより高頻度で到着するデータの高速な配信を実現する．またストリーム処理では，データ処理の定義に CQL （Continuous Query Language）11)を用いる．CQL では，取り扱うデータをストリームとして定義し，ストリーム間のデータ処理に表 1 に示すような Project，Join，Aggregate，Limit などのオペレータを用いる．ユーザは各オペレータのパラメータを設定することで処理ブロックを定義し，その処理ブロック間のデータフローを指定することにより,データ処理を定義する.したがって処理ブロックを置換，追加することにより，データ処理内容を比較的容易に変更することが可能である．また複数のデータ処理の処理ブロックを共有化することにより処理の最適化を実現する.

図 2 では CQL によるデータ処理の定義例として，他車と交差点間の距離（以下，交差点距離）及び，他車が交差点に到着する時刻（以下，交差点到着時間）の算出処理を示す．交差点距離の算出処理では，まず他車情報及び交差点情報のストリームを入力し，Join オペレータによる処理ブロック「交差点距離算出」により他車と交差点間の距離を算出し，その算出結果をストリームとして出力する．一方，交差点到着時刻の算出処理では，交差点距離の算出結果から，Aggregate 及び Project による処理ブロック「前距離算出」,「交差点到着距離算出」を用いて交差点到着時刻を算出する．したがって交差点到着時間及び交差点距離の算出処理では，処理ブロック「交差点距離算出」を共有化することによりコスト低減を実現する．

2.2.2 ストリーム処理のスケジューリング

ストリーム処理では，センサデータや通信データがシステムに入力される度に，各処理ブロックを逐次実行し，アプリケーションに配信する．これにより主に汎用システムの逐次実行を行うアプリケーションに対して，処理結果を低オーバヘッドに配信可能である．これにより例えば株自動取引等のアプリケーションでは，売買の判断を迅速に行うことが可能になる．図 2 に示すデータ処理では，他車データを他車情報ストリームに入力する度に，各処理ブロックを実行し，他車の交差点距離及び交差点到着時間の情報をアプリケーションに配信する．逐次的に実行するアプリケーションでは，データを受信する度にそのデータを処理するため，ストリーム処理を用いることで高速処理が可能になる．

表 1　ストリーム処理のオペレータの例

#	オペレータ	オペレータの動作
1	Project	単一ストリームのデータに対して，関数や演算を実行する．そして実行した結果をストリームとして出力する．
2	Join	複数ストリームの過去の特定期間のデータをメモリ上に保持し，特定の条件によって保持したデータを結合し１つのストリームとして出力する．
3	Aggregate	単一ストリームのデータを一定期間，メモリ上に保持し，保持したデータに対し集計関数を実行した結果をストリームとして出力する．
4	Limit	単一ストリームのデータを一定期間，メモリ上に保持し，保持したデータの属性値の小さい(または大きい)上位 N 件をストリームとして出力する．

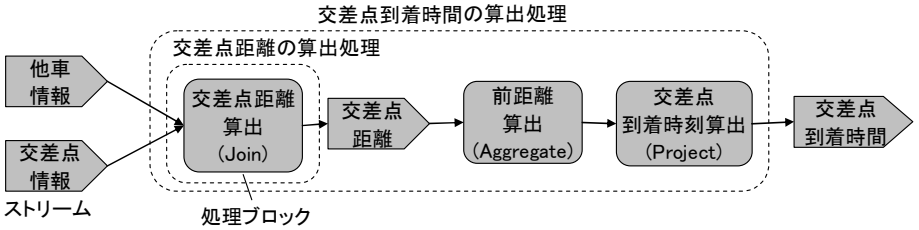

図 2　CQL によるデータ処理の定義例

2.3　車々間通信データのフィルタリングの必要性と要件

本節では，車々間通信データのフィルタリングの必要性を説明するために，車載アプリケーションのスケジューリング及び，車々間通信データの利用に向けた課題について説明する．そして車々間通信データのフィルタリングの要件を挙げる．

2.3.1 車載アプリケーションのスケジューリング

車載アプリケーションは，2.2.2 節で述べた汎用システムのアプリケーションとは異なり，制御周期から動作周期を決定し，その動作周期に合わせて周期的に実行する．また車載アプリケーションが扱う，車速/車輪速センサやステアリングセンサなどの自車センサのデータ数は一定である．したがって各アプリケーションが扱うセンサのデータ数から処理コストを見積もり，その処理コストに合わせて各アプリケーションを静的にスケジューリングすることにより，デッドラインを保証することが可能である．

2.3.2 車々間通信データの利用に向けた課題

しかし前述のように車載アプリケーションの多様化，高度化に伴い，自車センサだけでなく，車々間の通信データの活用が求められている．実際，車々間通信データを活用することにより，従来実現できなかった高度な制御が車載アプリケーションで可能となる．例えば車々間通信データを活用した衝突検知アプリケーションとして，Tu ら 18)は交差点における右折時の衝突検知を提案している．Tu らの方式では車々間通信により，他車の位置，速度，加速度，ヨーレートを取得し，他車の数秒後の経路を予測する．そして自車との車間距離が一定以内となる場合に衝突の可能性があると判定する．

このような車々間通信データは周辺車両の状況によりデータ量が大きく変化する．例えば文献 17)では最大 500 台の他車からデータを想定している．この想定の場合，1 台の車からの送信間隔を 100 ミリ秒に 1 回とすると，データ数は 500 台/100 ミリ秒より最大で毎秒 5000 個となる．しかし経路予測は，例えば Caveney ら 19)のアルゴリズムでは 1 秒先の予測に対しても約 7.1 ミリ秒の時間を要するため，1 / (7.1 ミリ秒 * 0.001)より，毎秒 150 個以上のデータを処理できない．したがって車々間通信データを活用した車載アプリケーションのデッドラインを保証するためには，車々間通信データをフィルタリングし，アプリケーションで処理するデータ量を一定量に抑える必要がある．

2.3.3 車々間通信データのフィルタリングの要件

車々間通信データのフィルタリングの要件として，アプリケーションの動作周期に合わせ，その処理データ量を一定以内に抑制することが挙げられる（以下，要件Ⅰ）．これにより自車センサを処理する場合と同様に，アプリケーションの処理コストを一定以内に抑え，デッドラインを保証することが可能になる．

また 2.3.2 節で述べた，車々間通信データを活用する衝突検知アプリケーションは，衝突を検知した場合には衝突前のドライバへの警告や，さらにブレーキの自動制御も想定するため，高い信頼性が要求される．したがってフィルタリングにより可能な限り，重要度の高いデータが削除されないことが要件として挙げられる（以下，要件Ⅱ）．

さらに車々間通信データを用いるアプリケーションとしては，前述した右折時の衝突検知アプリケーションの他に，交差時衝突や前方衝突検知など様々である．そしてアプリケーション毎に，同じ車々間通信データであっても重要度の高いデータは異なる．例えば右折時の衝突検知では，対向車のデータを必要とするのに対し，交差時の衝突検知には，直交する車のデータが求められる．したがってアプリケーション毎の指標に従ってデータの重要度を定義可能とすることがフィルタリングの要件として挙げられる（以下，要件Ⅲ）．

2.4　既存のストリーム処理によるフィルタリングの課題

本節では既存のストリーム処理のフィルタリングが，車々間通信データのフィルタリングに適用できないことを示す．ストリーム処理によるフィルタリングとして，ロードシェディング方式 12)13)14)15)16)が提案されている．ロードシェディング方式は，

入力データ量の増加によりストリーム処理の遅延が増大することを抑えるために，処理対象のデータを削減する方式である．ロードシェディング方式では，ランダム方式14)15)16)とセマンティック方式 12)13)と呼ばれる二種類の方式が検討されている．

ランダム方式は，データの属性値を考慮せず，削除するデータをランダムに選択する方式である．したがって重要度に高いデータから優先的に抽出することができず，フィルタリングの要件Ⅱを満たすことができない．

一方，セマンティック方式では，データの属性値を考慮して，削除対象のデータを選択する．したがって重要度の高いデータを抽出対象と指定することが可能である．しかしセマンティック方式では， 2.2.2 節で述べた逐次的に実行するアプリケーションを想定し，その処理遅延を抑えることを目的とする．したがって周期的に実行する車載アプリケーションに適用した場合，動作周期に合わせ一定件数抽出することができず，フィルタリングの要件Ⅰを満たなさない．また以下の例に示すように，重要度の高いデータを削除する可能性があり，フィルタリングの要件Ⅱを満たなさない．

図 3 の例ではセマンティック方式により,重要度の高いデータが削除されることを示す．動作例では，交差点までの距離が短い他車情報がより重要度が高いと設定した場合のフィルタリングを示す．セマンティック方式におけるフィルタリングのアルゴリズムは様々であるが，動作例では以下の手順に従う．

1. 一定の期間（図 3 のα～ω）毎に入力データ数を算出する．
2. 1 で算出した入力データ数が閾値（例では 1 個）を超えた場合には，それ以降，同一期間において重要度の低いデータを一定数（例では 1 個）削除する．
3. 1 で算出した入力データ数が閾値以下になった場合には，それ以降のデータを削除しない．

図 3 の例では，期間αでデータが 2 個入力されたため，手順 2 に従って期間βで，車 D のデータよりも交差点距離が長い車 C のデータを削除する．また期間γではデータが 1 個も入力されなかったため，手順 3 に従って期間ωで入力される車 B を削除しない．これにより図 3（ⅰ）に示すように，逐次的に実行するアプリケーションに適用した場合には，車 C のデータを削除することにより，重要度の高い車 D のデータを遅延することなく配信するなど，処理遅延の増加を抑えつつ重要度の高いデータの配信が可能になる．しかし図 3（ⅱ）に示すように，周期的に実行するアプリケーションに適用した場合には，重要度の高いデータを削除する．すなわち車 B，C はいずれも，時刻 200 ミリ秒に実行開始するアプリケーションで処理するにも関わらず，車 B のデータを抽出し，車 B よりも交差点距離が短い，すなわち重要度が高い車 C のデータを削除する．

上記アルゴリズムはセマンティック方式の一例であるが，いずれの方法を用いても周期実行のアプリケーションに適用することができず，車載アプリケーションではその動作周期を考慮したフィルタリングが求められる．

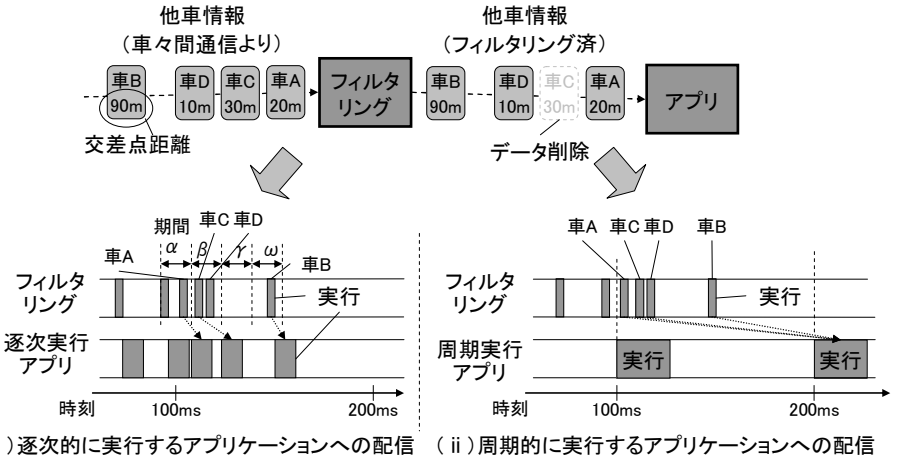

図 3　ロードシェディングのセマンティック方式の動作例

3. フィルタリング方式の提案

3.1 PPBF の概要

本稿では，2 節で述べた車々間通信データのフィルタリングの要件Ⅰ～Ⅲを満たす方式として PPBF を提案する．

PPBF の動作としては，まずアプリケーション毎に定義される，データの重要度の指標を算出する処理（以下，重要度算出処理）を実行する．そして図 4 の例に示すデータ削減処理では，重要度が高い順に入力データをソートし，一定件数，メモリ上に保持し，データが入力される度に，保持しているデータを更新する．そしてアプリケーションの実行開始時刻に，保持しているデータを取出し，入力順にソートした後にアプリケーションに配信する．アプリケーションはデータの配信後に実行を開始する．

図 4 はデータ削減処理の動作例を示す．PPBF では，時刻 100～200 ミリ秒においてデータを入力する度に重要度の高い 3 件を更新する．そしてアプリケーションの実行開始時刻 200 ミリ秒になったときに，上位 3 件のデータ，すなわち車 D，A，C のデータを取出し，入力順にソートしアプリケーションに配信する．従って図 3 に示すロードシェディングのセマンティック方式とは異なり，システムで決めたタイミングでなく，アプリケーションの動作周期に合わせて重要度の高いデータを抽出するため，時刻 100～200 ミリ秒で入力されたデータの中で，車 B のデータを削除し，より重要

度が高い車 C のデータを抽出することが可能になる．

これにより PPBF では，ロードシェディングのセマンティック方式とは異なり，車載アプリケーションの動作周期に合わせ，抽出データを一定件数に抑えるため，フィルタリングの要件 I を満たす．また動作周期に合わせ，より重要度の高いデータを優先的に抽出するため，フィルタリングの要件Ⅱを満たす．さらに PPBF では，データの重要度について，CQL によりユーザが定義する．これにより各車載アプリケーションに応じて,様々な重要度の指標を定義可能となるため,要件Ⅲを満たす.以下で PPBF の動作詳細について説明する．

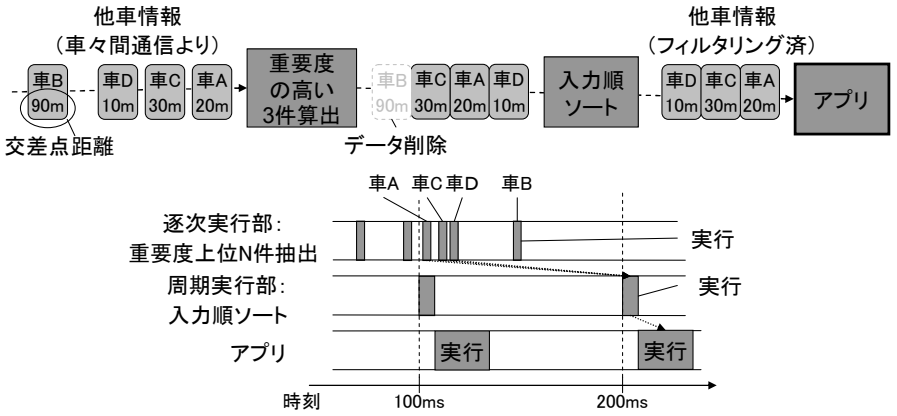

図 4　データ削減処理の動作例

3.2 重要度算出処理の定義

重要度算出処理は，車々間通信を通して受信する他車データを要素とする他車情報ストリームを入力とする．そしてユーザ定義の処理ブロックにより他車データの重要度を算出し,その重要度を出力する.例えば重要度の指標を,アプリ A が交差点距離,アプリ B が交差点到着時間とした場合には，図 2 に示すように，他車情報ストリームを入力とする交差点距離及び，交差点到着時間の算出処理を，重要度算出処理として定義する．なお PPBF では重要度の指標の値が小さいほど重要度が高いとみなす．例えばアプリ A，B の場合には，交差点距離，交差点到着時間が短いデータの重要度が高い．このように PPBF では重要度算出処理を CQL で定義し，車載データ統合アーキテクチャに登録することにより，2.2 節で述べたように，処理内容の追加や変更の容易化．また処理の最適化を実現する．

3.3 データ削減処理の生成

データ削減処理は，ユーザが指定したアプリケーションの動作周期及び，その周期毎の抽出件数に従ってシステムが生成する．データ削減処理は，図 5 に示すように,CQL の Limit オペレータ(表 1#4)による処理ブロック「重要度上位 N 件算出」(表 2#1)及び「入力順ソート」（表 2#2）から構成され，重要度を付加した他車情報を入力し，フィルタリングされた他車情報として出力する．具体的には以下のように動作する．

1. 重要度を付加した他車情報を入力する度に,処理ブロック「重要度上位 N 件算出」により,重要度の値の小さい上位 N 件を算出し,算出結果をメモリ上に保持する．
2. 各周期におけるアプリケーションの実行開始時刻となったら，メモリ上に保持している上位 N 件のデータを抽出し，抽出後，保持したデータを削除する．
3. 抽出したデータを，処理ブロック「入力順ソート」により車々間通信データの入力順にソートし，フィルタリングされた他車情報として出力する．

例えばアプリケーションに対し動作周期 100 ミリ秒，抽出件数 3 件と定義された場合には，100 ミリ秒毎に交差点距離の値の小さい上位 3 件のデータを抽出する．図 4 では，時刻 100 ミリ秒から 200 ミリ秒の間に車 A，D，C，B のデータが入力された場合に，時刻 200 ミリ秒で車 A，D，C のデータを抽出されることを示す.

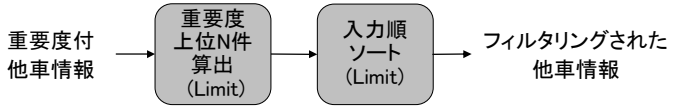

図 5　データ削減処理の定義

表 2　データ削減処理を構成する処理ブロック

#	処理ブロック名	オペレータ	処理ブロックの説明
1	重要度 上位 N 件抽出	Limit	動作周期毎に，入力データの重要度の値が小さい順に N 件（抽出件数）抽出する．
2	入力順ソート	Limit	入力データを時刻順にソートし出力する．

3.4 重要度算出処理とデータ削減処理の結合

重要度算出処理とデータ削減処理から，以下に示すフィルタリング処理(図 6)をシステムで生成する

1. 他車情報ストリーム（表 3 #1）を重要度算出処理に入力する．
2. 重要度算出処理の出力と他車情報ストリームを，Join オペレータによる処理ブロック「重要度付他車情報生成」(表 4 #1)に入力し，重要度を付加した他車情報を出力する．

3. 重要度付きの他車情報をデータ削減処理に入力する．そしてデータ削減処理の出力を Project オペレータによる処理ブロック「フィルタ済他車情報生成」(表 4 #2)に入力し，他車情報から重要度の情報を除き，車々間通信により受信した他車情報と同一の形式にする．そしてアプリケーションに配信するフィルタ済他車情報ストリーム（表 3 #2）を出力する．

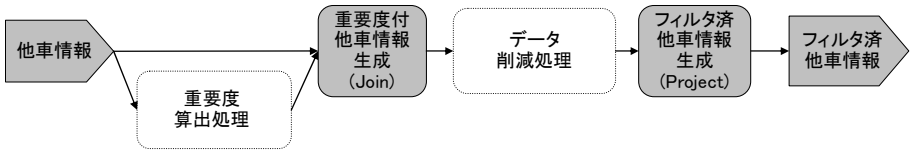

図 6 フィルタリング処理の定義

表 3 フィルタリング処理を構成するストリーム

#	ストリーム名	ストリームの説明
1	他車情報	車々間通信で受信する他車データを格納する．
2	フィルタ済 他車情報	フィルタリング後のアプリケーションに配信する他車データを格納する．

表 4 フィルタリング処理を構成する処理ブロック

#	処理ブロック名	オペレータ	処理ブロックの説明
1	重要度付 他車情報生成	Join	他車データと重要度を結合し，重要度が付加した他車データを生成する．
2	フィルタ済 他車情報生成	Project	他車情報ストリームに格納されたデータとフォーマットが同一の他車データを生成する．

3.5 フィルタリング処理のスケジューリング

定義したフィルタリング処理を実行することで，車々間通信データのフィルタリングを実現する．図 7 に示すように，車々間通信を通して他車データを受信し，その他車データを他車情報ストリームに対応するキューに格納する．そしてその他車データを入力とし，フィルタリング処理の各処理ブロックをデータフローに従って実行する．処理ブロックの実行については，処理ブロック「重要度上位 N 件算出」までの処理（以下，逐次実行部）は車々間通信データを受信する度に逐次実行する．一方，処理ブロック「入力順ソート」以降の処理（以下，周期実行部）は周期実行する．すなわち各周期でアプリケーションの実行開始時刻となり，逐次実行部でその時刻までの受信データを処理後，周期実行部の実行を開始する．そして周期実行部ではその処理結果を，フィルタ済他車情報ストリームに対応するキューに格納する．最後に周期実行部によるキューへの格納が完了した後に，アプリケーションの実行を開始する．

例えば図 4 では，動作周期が 100 ミリ秒，すなわちアプリケーションの実行開始時刻を 100 ミリ秒，200 ミリ秒とした場合である．動作例では逐次実行部で時刻 100 ミリ秒～200 ミリ秒に受信した車 A～D のデータを処理した後に，周期実行部，アプリケーションの順に実行する．

したがってアプリケーションは，逐次実行部及び周期実行部と同期するために遅延が生じる可能性があるため，次節で処理の遅延について評価する．

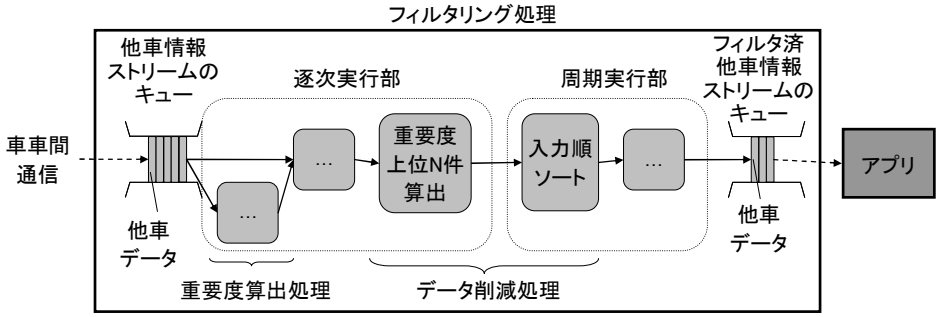

図 7 フィルタリング処理の実行

4. フィルタリング方式の評価

4.1 評価方法，環境

PPBF を，ストリーム処理システム Borealis 20)をベースに我々が開発した車載データ統合プラットフォームを拡張することで，プロトタイプ実装した．そしてプロトタイプ上でフィルタリングの処理遅延について測定し，フィルタリングをしない場合（以下，フィルタリングなし方式）と比較した．評価環境は PC であり，PC 上の VM（プロセッサ数 2，メモリ 2560MB 割当て)に Linux（Ubuntu 8.04）を搭載しプロトタイプを動作させた．

ストリーム処理システムへ入力する車々間通信データについては，PreScan 21)により，交差点走行時の周辺他車との車々間通信をシミュレーションすることにより生成した．評価に用いるフィルタリング内容を表 5 に示す．アプリ C はストリーム処理システムからの他車データを受信する処理のみを実行するアプリケーションである．一方，フィルタリングなし方式では，車々間通信データを受信し，ストリームのキュー

に格納し,そのデータを処理ブロックに渡すことなく全データをアプリ C に配信する.

表 5　評価に用いたフィルタリング内容

アプリケーション	重要度の指標	動作周期	抽出件数
アプリ C	交差点到着時間	100 ミリ秒	10 件

4.2 評価結果

PPBF 及びフィルタリングなし方式の処理遅延の評価結果を，図 8 に示す．処理遅延は，アプリ C が各周期で実行開始の時刻となってから，他車データの取得が完了するまでの時間を測定する．車々間通信データの入力データ数は毎秒 100 個～2000 個の範囲とする．評価の結果，PPBF の処理遅延は，入力データ数が毎秒 1400 個までの場合には 4 ミリ秒以内であるが,入力データ数が毎秒 1400 個よりも大きい場合には最大で 40 ミリ秒以上に増加した．一方，フィルタリングなしの方式の処理遅延は，入力データ数が毎秒 2000 個以内で 5 ミリ秒以内に収まった．

4.3 考察

フィルタリングなし方式の処理遅延の原因は，アプリ C のデータの取得に要する時間と考えられ，アプリ C はデータの受信処理のみを実行するアプリケーションであるため処理遅延への影響は小さかった．しかし実アプリケーションを用いた場合には入力データ数の増加に従って，アプリケーションの処理コストが増加し，その結果，処理時間が増加すると考えられる．例えば Caveney ら 19)のアルゴリズムでは 1 秒先の起動予測に 7.1 ミリ秒の処理時間を要するため, 2.3 節で述べたようにフィルタリングなしの方式では毎秒 150 個以上のデータを処理できない．

一方，PPBF の処理遅延は，入力データ数が毎秒 1400 個以下の場合は 3.5 節で述べた周期実行部の処理時間が主な原因である．周期実行部は各周期で抽出件数と同数のデータを処理するために，入力データ数に関係なく処理コストは一定であるため，処理遅延が 4 ミリ秒以内に収まった．一方，入力データ数が毎秒 1400 個以上の場合は逐次実行部の処理コストの増大が原因と考えられる．逐次実行部は入力デー多数に比例して処理コストが増加するために，処理遅延が最大で 40 ミリ以上に増加した．

PPBF では 100 ミリ秒毎に 10 件以内のデータを抽出するため，入力データ数が毎秒 1400 個以下の場合，Caveney ら 19)のアルゴリズムではフィルタリングの遅延を考慮しても,処理時間は各周期で 7.1 ミリ秒 * 10 件 +4 ミリ秒より 75 ミリ秒であるため，今回，評価に用いたフィルタリングでは，入力データ数が毎秒 1400 個まで処理することが可能である．したがって PPBF が有用性であることを示せた．

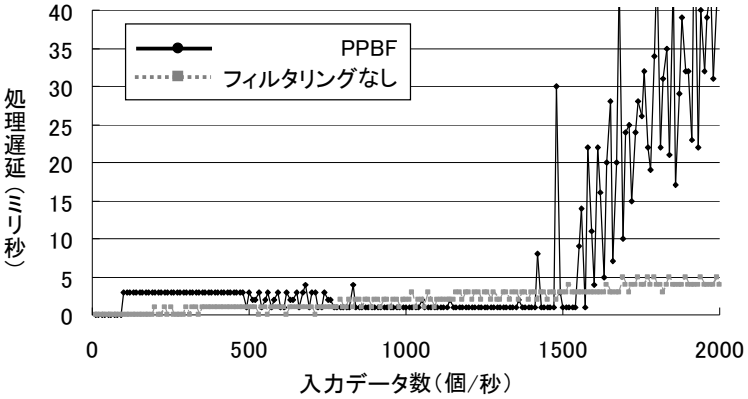

図 8　PPBF の処理遅延の評価

5. 関連研究

車載統合制御システムにおけるアプリケーションの処理共有化としては， RDB を用いたデータ管理手法 5)6)がある．しかし RDB ではストリーム処理とは異なり，一般的にメモリ上でなくハードディスク上に格納され，データ入力時に逐次処理を行わないため，リアルタイム性が要求される処理で活用するのが難しい．また Reinholtz ら 7)は自車の周辺状況の認識情報を共有化し，各アプリケーションに配信するシステムを提案している. しかしこの方式では認識情報の算出部分はストリーム処理を用いず，作り込まれている. したがって PPBF とは異なり, ストリーム処理による処理の追加，変更の容易化や処理最適化が実現されない．

ストリーム処理については Stanford 大, MIT などにより様々な方式が研究されている 9)10). またストリーム処理によるフィルタリングとしては, 2.4 節で述べたように，ロードシェディングのランダム方式及びセマンティック方式が検討されている．

ロードシェディングのランダム方式としてはアルタイム処理向けの方式が提案されている 14)15)16)．これらの方式では入力データ数と，直前のデータ数を用いて，処理に渡すデータ量を制御する．したがってアプリケーションの処理データ数を保証することが可能であるが, PPBF とは異なり, 削除対象のデータはランダムに選択され，データの重要度に基づいた取捨選択はできない．

一方，ロードシェディングのセマンティック方式では，データの属性値に従って重要度を定義し，重要度に基づいてフィルタリング対象を選択する 12)13)．セマンティ

ック方式では，PPBF とは異なりアプリケーションの動作周期を考慮していない．したがって入力データ数が一定以上増加した場合には，同一周期で処理するデータにおいて重要度がより高いデータが削除される可能性がある．

6. おわりに

本稿では，車々間通信データのフィルタリングの要件を満たす方式として，PPBF を提案した．PPBF では重要度の高いデータから順に，アプリケーションの動作周期に合わせ，一定件数のデータを抽出する．これによりアプリケーションは，重要度の高いデータを一定件数処理することが可能になる．また PPBF では，データの重要度の算出処理について CQL を用いて定義することで, アプリケーション毎の指標に従って重要度を決定できる．PPBF を実装し性能評価した結果， フィルタリングの処理遅延を抑えられることを確認し，ストリーム処理による車々間通信データのフィルタリングの実現性を示した．今回の実装はシステム構築を容易に行うために PC を利用したが, 組込み環境においてシステムを構築し評価を実施することが今後の課題となる．

謝辞 本研究の一部は科研費(22240003) の助成を受けたものである．

参考文献

1) 浅沼信吉，加世山秀樹:安全運転支援のための車両予知・予測技術のとりまく状況，国際交通安全学会誌,Vol.31, 2006.
2) W.D. Jones: Keeping Cars from Crashing, IEEE Spectrum, Vol.38, 2001.
3) 藤田浩一, 宇佐見祐之, 山田幸則, 所節夫: 衝突危険性のセンシング技術，自動車技術, Vol.61, 2007.
4) 西垣戸貴臣，大塚裕史，坂本博史，大辻信也: 予防安全の高度化を実現するセンササーフュージョン技術，日立評論, Vol.89, 2007.
5) D. Caveney: Hierarchical Software Architectures and Vehicular Path Prediction for Cooperative Driving Applications, International IEEE Conference on Intelligent Transportation Systems (ITSC), 2008.
6) D. Nystromy, A. Tesanovic, C. Norstromy, J. Hansson, and N-E. Bankestadz: Data Management Issues in Vehicle Control Systems: a Case Study, EUROMICRO Conference on Real-Time Systems, 2002.
7) C. Reinholtz: DARPA Urban Challenge Technical Paper, 2007.
8) 佐藤健哉: 自動車走行環境認識のためのセンサデータ処理機構，電子情報通信学会技術研究報告. DE, データ工学，Vol.110, 2010.
9) R. Motwani, J. Widom， A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma: Query Processing, Resource Management, and Approximation in a Data Stream Management System, Conference on Innovative Data Systems Research (CIDR), 2003.
10) D. J. Abadi, D. Carney, U. Cetintemel, 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, Vol.12, 2003.
11) A. Arasu, S. Babu, and J. Widom: The CQL continuous query language: semantic foundations and query execution, The VLDB Journal, Vol. 15, 2006.
12) N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker: Load Shedding in a Data Stream Manager, International Conference on Very Large Data Bases (VLDB), 2003.
13) J. Park and H. Cho: Semantic Load Shedding for Prioritized Continuous Queries over Data Stream, International Symposium on *Computer and Information Sciences* (ISCIS), 2005.
14) Y. Wei, S. H. Son, and J. A. Stankovic: RTSTREAM: Real-Time Query Processing for Data Streams, IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), 2006.
15) Y. C. Tu, S. Liu, S. Prabhakar, and B. Yao: Load Shedding in Stream Databases: A Control-Based Approach, International Conference on Very Large Data Bases (VLDB), 2006.
16) X. Li, L. Ma, K. Li, K. Wang, and H. A. Wang: Adaptive Load Management over Real-time Data Streams, International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2007.
17) 総務省「 ITS 無線システムの高度化に関する研究会報告書」 http://www.soumu.go.jp/main_content/000025426.pdf
18) L. Tu, C-M Huang: Forwards: A Map-Free Intersection Collision-Warning System for All Road Patterns, IEEE Transactions on Vehicular Technology, 2010.
19) D. Caveney: Numerical Integration for Future Vehicle Path Prediction, American Control Conference (ACC), 2007.
20) Borealis Distributed Stream Processing Engine, http://www.cs.brown.edu/research/borealis/public/
21) Tass, PreScan, http://www.tno.co.jp/PreScan