Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29

車載システム向けストリームデータ処理の提案と評価

勝沼 聡[†1] 杉本明加[†1] 山口晃広[†1]
山田真大[†1] 金 榮柱[†1] 本田晋也[†1]
佐藤健哉[†2,†1] 高田広章[†1]

車載システムでは車載データ増加に伴うアプリケーションの開発コスト削減を目的とし，ストリームデータ処理を用いた車載データの統合管理が検討されている．しかし従来のストリームデータ処理は一般的に汎用システムを想定しており，組込みシステムである車載システム上での実行が困難である．本研究では車載システム向けのストリームデータ処理を提案する．提案方式ではクエリ及び，実行環境のハードウェア/OS の情報に従って，ストリームデータ処理の必要コンポーネントを含むコードを静的に生成する．これによりバイナリファイルサイズを最小限に抑え，また様々なハードウェア/OS 上の実行を容易化する．提案方式を制御系及び情報系 ECU を想定した組込み環境で評価することにより有用性を示す．

Proposal and Evaluation of Stream Data Processing for Vehicle Systems

Satoshi Katsunuma,[†1] Meika Sugimoto,[†1]
Akihiro Yamaguchi,[†1] Masahiro Yamada,[†1]
Youngjoo Kim,[†1] Shinya Honda,[†1] Kenya Sato[†2,†1]
and Hiroaki Takada [†1]

In vehicle integrated control systems, the number of sensors and applications are increasing. Therefore, we develop the stream data processing system to manage vehicle data. However, since existing stream data processing techniques are executed in general-purpose systems, it is difficult to excute these in automotive systems, which are embedded systems. In this paper, we propose the stream processing method for automotive systems. This method determines whethrer it compiles the components according to query and the informations of hardware/OS. We evaluate this proposed technique in embedded systems equal to control ECU and informaion ECU. Experimental result shows that this technique is efficient.

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．我々はこの問題解決するため，アプリケーションからセンサ部を切り離し，プラットホームでセンサから取得したデータを統合的に管理する車載データ統合プラットフォーム[8)] を提案している．車載データ統合プラットフォームでは，センサから得られたデータをストリームデータ処理により管理することで，様々なデータ処理の共有化によるコスト削減しつつ，そのデータ処理結果の高速な配信を可能にする．

STREAM[9)]，Aurora[10)]，Borealis[12)] などの従来のストリームデータ処理は一般的に汎用システムを想定しており，動的な処理内容の追加や変更，処理最適化が行われる．そのためプラットフォームに多くの機能が組み込まれバイナリファイルサイズが大きい．また車載システムは ECU によってハードウェアや OS が異なるが，多種多様な環境に対応する柔軟性がない．さらに車載システムは静的に処理内容を決定する必要があるため，そのままではストリームデータ処理を適用することができない．

そこで本研究では車載システムに適応可能なストリームデータ処理を提案する．提案方式では**図 1** に示すように，設計時に（静的に）定義したクエリに従って，静的に必要最小限

†1 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
†2 同志社大学モビリティ研究センター
Mobility Research Center, Doshisha University

Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29

のコンポーネントを含むコードを生成する．そして様々な ECU のハードウェア/OS の構成に対応するためにコンポーネントの追加や削除，置換えを可能とする．

本稿の構成は以下の通りである．まず 2 節で車載システムへのストリームデータ処理適用に向けた課題を述べる．そして次に 3 節において本稿で提案する，車載システム向けストリームデータ処理を説明し，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．

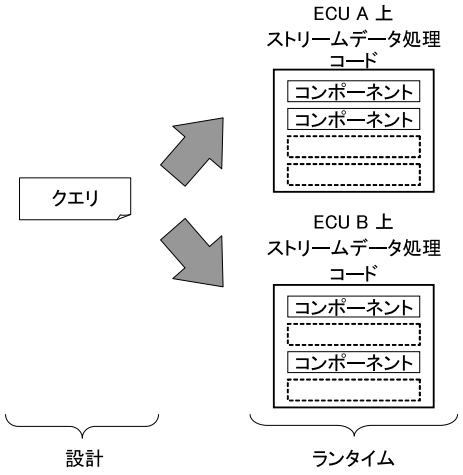


図 1 車載システム向けのストリームデータ処理の方針

2. 車載システムへのストリームデータ処理適用に向けた課題

2.1 汎用システム向けストーリームデータ処理

ストリーム処理は主に株自動取引，電子マネー，携帯操作などを扱う汎用システムを対象として開発され用いられている．本節ではこのような汎用システム向けのストリームデータ処理の概要及び動作例について説明する．

2.1.1 概　　要

ストリームデータ処理は，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．ストリームデータ処理は従来の RDB（Relational DataBase）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．またストリームデータ処理では，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 Filter，Union，Map，Join，Aggregate などのオペレータ（**表 1**）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．

表 1 ストリームデータ処理のオペレータの例

オペレータ	オペレータの動作
Filter	単一ストリームのデータを条件に従って間引きを行い，間引きしたデータをストリームとして出力する．
Union	複数ストリームのデータを単一のストリームとして出力する．
Map	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．
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．
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．

2.1.2 動　作　例

ストリームデータ処理の動作例について**図 2** に示す．ここでは他ノードから受信したデータを Aggregate オペレータで処理する．そしてその処理結果を別の Aggregate オペレータで処理し他ノードに送信する．ストリームデータ処理では Aggregate などのオペレータやデータ通信部はストリームのキューを用いてデータを受け渡しする．各オペレータの実行順や実行タイミングはスケジューラによって決定する．そして各オペレータは，小容量のメモリ領域であるウィンドウにデータを読み書きする．例えば Aggregate オペレータではウィンドウに指定時間あるいは指定件数格納し，そのウィンドウ内のデータに対し合計や平均などの集計を行う．

2.2 車載システムの要件

車載システムとしては情報系及び制御系 ECU 上のシステムを想定する．情報系 ECU ではカーナビなどが実現される．また制御系 ECU 上のアプリケーションとしてはプリクラッ

Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29

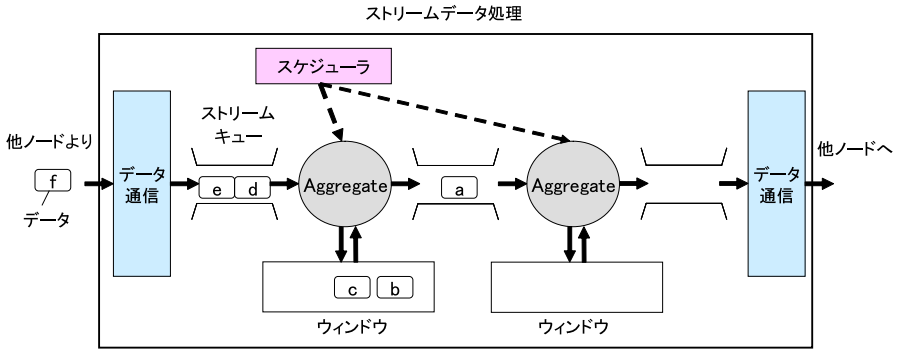


図 2 ストリームデータ処理の動作例

シュセーフティ技術などによるブレーキ制御や，自動パーキング，レーンキーピング，追随走行などが挙げられる．情報系及び制御系 ECU でストリームデータ処理を実行するために，PC と異なる以下の要件が挙げられる．

- 1. ROM へのコード搭載や，コードの信頼性確保のため，静的に処理内容を決定しコードを生成する必要がある．
- 2. 費用低減や物理的制約などの理由からリソース制約が厳しく，プログラムのコード等を割り当てる ROM の容量も限られているため，ストリームデータ処理のバイナリファイルサイズを抑制することが求められる．
- 3. リソース制約が厳しく RAM の容量も限られているため，動的にメモリを確保するのではなく静的に確保した固定長のメモリ領域を活用することが求められる．
- 4. TCP などの汎用システム向けの通信プロトコルのみならず，CAN 等の車載向け通信プロトコルの活用が求められる．
- 5. OS が AUTOSAR 等の RTOS（Real-Time Operating System）であることが多く，RTOS を活用した優先度付きのスケジューリングが求められる．

2.3 従来のストリームデータ処理の課題

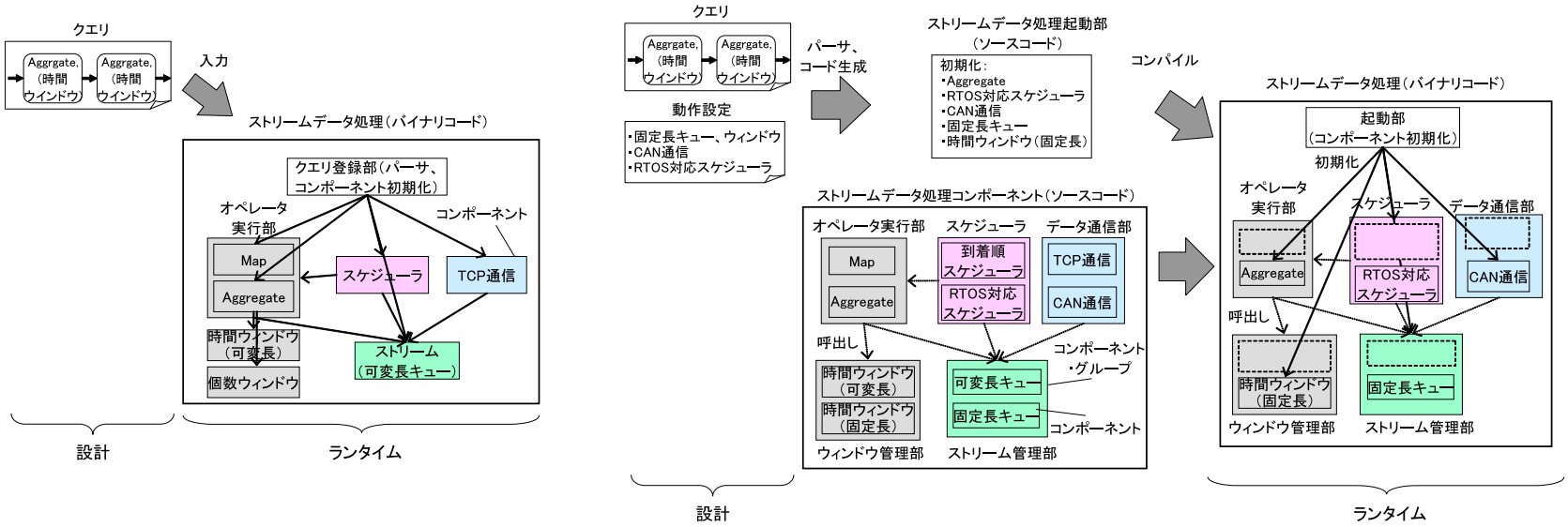
従来の汎用システム向けのストリームデータ処理では**図 3**（a）に示すように，あらかじめストリームデータ処理のソースコードをコンパイルしバイナリコードを生成し，システムにインストールする．そしてインストール済みのストリームデータ処理に対し，データ処理内容が定義されたクエリを登録する．そして登録されたクエリに従ってストリームデータ処理の各コンポーネントを初期化しストリームデータ処理を実行する．したがってストリームデータ処理は動的にクエリを登録するため静的にコードを決定することができず要件 1 を満たさない．また動的なクエリの追加や変更，さらに処理の最適化のために多くの機能が組み込まれ，ストリームデータ処理のバイナリファイルサイズが大きくなり要件 2 が満たせない．また汎用システム向けであるため，a. 動的なメモリ確保，b. TCP 通信，c. Linux 等の汎用システム向けの OS を想定しており，要件 3〜5 を満たさない．

例えば Borealis の場合図 3（a）に示すように，クエリを，システムにインストールしたストリームデータ処理に動的に登録するため要件 1 を満たさない．またインストールする機能を変更は一部しか対応しておらず，最小構成の場合にも全てのオペレータ及び，ストリーム，ウィンドウ，スケジューラ，TCP 通信部がインストールされる．またクエリ追加のためのパーサ機能や，データの間引き，データ流の制御機能など制御系では必ずしも必要としない機能も搭載される．したがってバイナリファイルサイズは全体で約 2.9MB となり，これよりもバイナリファイルサイズが小さいコードを生成できない．ITRON WG/調査研究 WG[19] によると，組込みシステムではバイナリプログラムのサイズが 1MB 未満のプログラムが 56 %，その内 256KB 未満のプログラムが 35 %を占めると述べられており，Borealis は車載システムを初めとした組込みシステムとしてはバイナリファイルサイズが大きい．したがって要件 2 を満たすことができない．

また Borealis は動的にメモリを確保し，また TCP 通信を標準とし，RTOS にも非対応であるため要件 3〜5 を満たさない．また Borealis では互いに依存関係にあるコンポーネントが多い．例えば図 3（a）に示すように，ストリームは各オペレータや TCP 通信部から呼び出され，ウィンドウも各オペレータから呼び出される．したがって動的なメモリ割当てができない環境でストリームやウィンドウを固定長から可変長に変更する場合には，各オペレータや TCP 通信部の変更も必要になる．また同様にスケジューラや TCP 通信部も他コンポーネントから呼び出されるため，RTOS 向けのスケジューラや車載通信の追加に対しても全体の変更するコード量が多い．したがって車載システム向けの機能拡張も困難である．

なお組込みシステム向けのストリームデータ処理としては，Gigascope[13]，Muller の手法[14] が挙げられる．これらの方式では静的にコードを生成するため要件 1 を満たす．また Gigascope ではクエリからソースコードを生成するためバイナリファイルが小さくなり要件 2 を満たす．一方，Muller の手法は仮想マシンを用いてストリームデータ処理を実現するため，クエリに関係なく仮想マシンのコードが生成されるためバイナリファイルサイズが大きくなり要件 2 は満たさない．また Gigascope ではネットワーク機器，Muller の手法で

Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29



(a) 汎用システム向けストリームデータ処理　　(b) 車載システム向けストリームデータ処理

図 3　汎用システム/車載システム向けのストリームデータ処理

は無線センサノードを想定しており車載システムの要件 3〜5 は満たさない.

3. 車載システム向けストリームデータ処理

3.1 概　　要

車載システム向けのストリームデータ処理では 2.3 節で示した要件 1〜5 を満たすために従来のストリームデータ処理とは異なる構成を取る．まず図 3（b）に示すようにクエリや動作設定に従って静的にソースコードを生成する（要件 1 の満足）．そしてストリームデータ処理のコンポーネントの追加や削除，置換えを他コンポーネントに影響することなく可能とし，クエリに従って必要コンポーネントのコードのみを生成しバイナリファイルサイズを最小限に抑制する（要件 2 の満足）．また動作設定に従って，a. ストリームやウィンドウのコンポーネントを可変長から固定長への置換え，b. TCP 通信機能の削除，c. RTOS に対応したスケジューラの追加，を他コンポーネントの変更なしに可能とする（要件 3〜5 の満足）.

3.2 ストリームデータ処理のコンポーネント

提案方式ではストリームデータ処理は複数のコンポーネントにより構成され，各コンポーネントはコンポーネント・グループ毎に共通のインタフェースを通して他のコンポーネントを呼び出す．**表 2** に示すようにコンポーネント・グループとしては，ストリーム管理部，オペレータ実行部，ウィンドウ管理部，スケジューラ，データ通信部を定義する．各コンポーネント・グループに属するコンポーネントの一例を表 2 に示し，コンポーネント・グループごとに定義されるインタフェースの一例について**図 4** に示す.

図 2 に示すストリームデータ処理の実行における，コンポーネントの呼出しの例を示す．まずデータ通信部は他のノードからデータを受信した際に，ストリーム管理部を呼出し，ストリームのキューにデータを書き込む．そしてスケジューラは Aggregate 等のオペレータ実行部を起動し，起動された各オペレータ実行部は，ストリームのキューのデータを参照し，その処理結果を書き込む．こうして処理された結果をデータ通信部でストリームのキューのデータを参照し他ノードに送信する.

Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29

表 2 ストリームデータ処理のコンポーネントの例

コンポーネント・グループ	コンポーネント
ストリーム管理部	可変長キュー，固定長キュー
オペレータ実行部	Filter，Map，Union，Join，Aggregate オペレータ
ウィンドウ管理部	個数，時間（固定長，可変長）ウィンドウ
スケジューラ	到着順，RTOS 対応スケジューラ
データ通信部	TCP，UDP，CAN 通信

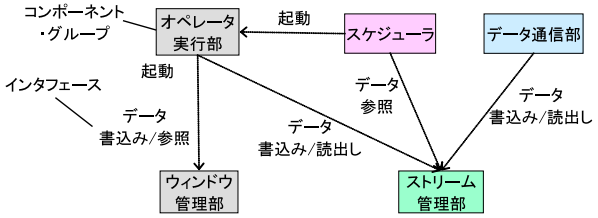


図 4 ストリームデータ処理のコンポーネント間の呼出し

3.3 ストリームデータ処理起動部のソースコード生成

提案方式ではあらかじめ定義されたコンポーネントの中からクエリ，動作設定に従って，必要とするコンポーネントを決定する．そして必要なコンポーネントを初期化するストリームデータ処理起動部の C/C++ソースコードを生成する．例えば図 3（b）に示すクエリでは Aggregate オペレータのみが定義されているため，Aggregate オペレータを初期化するコードを生成する．また動作設定としては固定長キュー，ウィンドウ，CAN 通信，RTOS 対応スケジューラが指定されている．したがって固定長キューのストリーム，固定長の時間ウィンドウ，CAN 通信部，RTOS 対応スケジューラを初期化するコードも生成する．

3.4 ストリームデータ処理のバイナリコード生成

最後に生成したストリームデータ処理の起動部のソースコード及び，必要コンポーネントのソースコードをコンパイル，リンクすることにより，ストリームデータ処理のバイナリコードを生成する．例えば図 3（b）では起動部及び，Aggregate オペレータの実行部，固定長キューのストリーム，固定長の時間ウィンドウ，CAN 通信部，RTOS 対応スケジューラのソースコードをコンパイル，リンクし，バイナリコードを生成する．

3.5 実　　装

提案方式は，ストリームデータ処理のコンポーネント及び，ストリームデータ処理起動部のソースコードを生成するツール（以下，ソースコード生成ツール）から構成される．実装

表 3 実装したストリームデータ処理のコンポーネント

コンポーネント・グループ	コンポーネント
ストリーム管理部	可変長，固定長キュー
オペレータ実行部	Filter，Map，Union，Aggregate（可変長/固定長テーブル），Join オペレータ
ウィンドウ管理部	個数，時間（可変長/固定長）ウィンドウ
スケジューラ	到着順，RTOS 対応スケジューラ
データ通信部	TCP，UDP 通信
集計関数	Count，Sum，Average，Max，Min 関数

したストリームデータ処理のコンポーネントを**表 3** に示す．各コンポーネントは C/C++言語を用いて定義した．またソースコード生成ツールは Java で記述し，XML 形式で定義されたクエリ及び動作設定に対し XML をパースし中間コードを生成する．そして中間コードに対して処理最適化[17] を行い C/C++ソースコードを生成する．

4. 車載システム向けストリームデータ処理の評価

情報系及び制御系 ECU を想定した環境におけるハードウェア/OS への対応（要件 3〜5）及び，バイナリファイルサイズ（要件 2）について提案方式を Borealis と比較し評価する．

4.1 評 価 環 境

情報系 ECU を想定した環境としては，ZMP 社の RoboCarR 1/10 / ZMP RC-Z（以下，RoboCar）[15] を活用する．CPU は AMD Geode LX800 Processor 500MHz であり，メモリサイズは 478MByte である．OS は Linux（Fedore 10）である．一方，制御系 ECU を想定した環境としては，Altera 社の Nios II 開発キット, Cyclone III エディション（3C25）（以下，Altera ボード）[16] を活用する．CPU は Nios II/f processor core であり，メモリは DDR SDRAM メモリ 133 MHz, 16 bits のサイズは 32 MBytes であり，また Synchronous SRAM メモリのサイズは 1 MByte である．OS は TOPPERS/ATK2（AUTOSAR 準拠）である．

4.2 実行環境のハードウェア/OS への対応

提案方式では RoboCar，Altera ボードにおいて各々**図 5**（a）（b）に示すようなストリームデータ処理コンポーネントを選択することで動作を確認した．RoboCar では PC 上と同様のコードで動作することが可能であった．また Altera ボードは Robocar とは異なり動的なメモリ割当て (ヒープ) に OS が対応していなかったため，ストリームのキュー，ウィンドウ，Aggregate オペレータに固定長のメモリ領域を活用する構成とした．また TCP，UDP 通信は非対応であったためコードから除いた．またスケジューラに RTOS を活用したもの

Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29

を追加した．そして RoboCar と Altera ボードで Filter，Map，Union，Join オペレータ，個数ウィンドウは共通のコンポーネントを活用することができ，追加や削除，置換えを行った他コンポーネントの影響を受けないことが示された．

また RoboCar 上では 3 種類の衝突を検知するクエリ[17] を動作させ，全ての衝突を検知する場合にもレイテンシの平均 2285us，最大 3977us と妥当であることを確認した（**表 4**）．また Altera ボード上で単一オペレータのクエリを動作させ，その結果，レイテンシが最大でも Aggregate オペレータの場合で 168us 以内におさまり妥当であることを確認した（**表 5**）．Altera ボード上の実際の車載システムを想定したクエリによる性能評価は今後の課題とする．

一方，Borealis では RoboCar は前述のように PC と同様の環境であるため動作が可能であった．しかし Altera ボード上では，動的なメモリ確保や，TCP 通信，RTOS を対応しないスケジューラが最小の構成にも含まれているためコンパイルができない．また Altera ボードに対応するためのコードの変更についても 2.3 節で述べたようにオペレータも含めた変更が必要となる．したがって実行環境のハードウェア/OS への対応に関して提案方式の優位性を示すことができた．

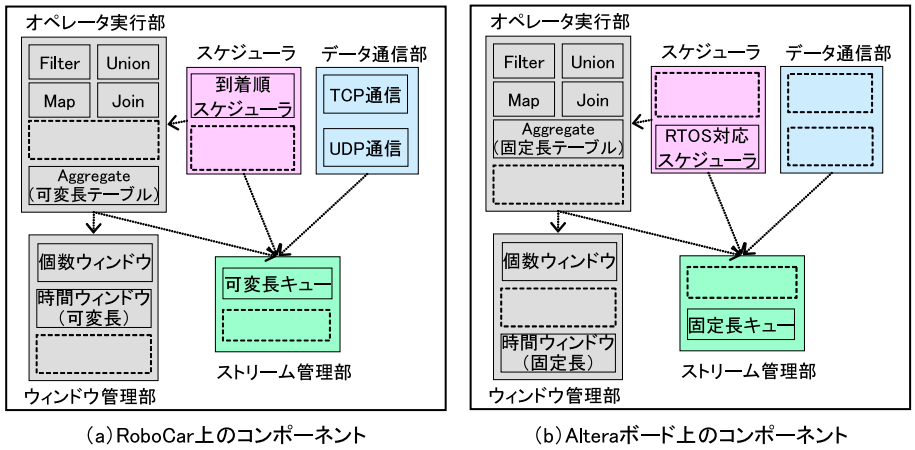


図 5 RoboCar/Altera ボード上のストリームデータ処理のコンポーネント

4.3 バイナリファイルサイズ

提案方式において，RoboCar，Altera ボード上で異なるオペレータから構成されるクエリを実行する場合のバイナリファイルサイズを評価する．RoboCar 上では**図 6** に示すように，単一オペレータのクエリの場合には，全オペレータのクエリの場合と比べて，最大で Filter オペレータの場合に 61,051Byte（#6 と#2 の合計の差）と 49%削減することを確認した．一方，Altera ボード上では**図 7** に示すように，単一オペレータのクエリの場合には，全オペレータのクエリの場合と比べて，最大で Filter オペレータの場合に 32,531Byte（#6 と#2 の合計の差）と 40%削減することを確認した．したがってクエリに従ってコンパイル対象のオペレータを限定することによりバイナリファイルサイズが削減することが確認できた．

表 4 RoboCar 上の処理レイテンシ評価

オペレータ	前方衝突	交差時衝突	右折時衝突	全衝突
平均レイテンシ	1,701us	1,749us	1,328us	2,285us
最大レイテンシ	2,555us	3,789us	2,240us	3,977us

表 5 Altera ボード上の処理レイテンシ評価

オペレータ	Filter	Union	Map	Join	Aggregate
平均レイテンシ	89us	94us	102us	147us	156us
最大レイテンシ	90us	95us	104us	147us	168us

また提案方式の RoboCar 上におけるバイナリファイルサイズはデータ通信部も含めて最大で 133,495 byte となった．一方，Borealis では最小構成の場合にもバイナリファイルサイズは 2.9MB である．これは Borealis はクエリのパーサ等の機能も実行環境に組み込まれていることが原因である．ITRON WG/調査研究 WG[19] によると，組込みシステムではバイナリファイルサイズが 256KB 以上のプログラムが 60 %以上，1MB 以上のプログラムが 40 %以上を占めると述べられており，提案方式によりストリームデータ処理のバイナリファイルサイズが，標準的な組込み向けプログラムと同等となることが確認できた．

5. 関 連 研 究

汎用システム向けのストリームデータ処理としては，STREAM[9]，Aurora[10]，Borealis[12] などがある．2.3 節で述べたように汎用システム向けのストリームデータ処理は提案方式とは異なり，動的にクエリを登録するため静的にコードを決定できない．またクエリに従ってコンパイル対象のコンポーネントを選択することができずバイナリファイルサイズが大きい．また車載システムにおける様々なハードウェア/OS に対応していない．そして車載システム向けのコード変更についても，提案方式とは異なりコンポーネントの追加，削除，置

Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29

表 6 RoboCar 上のバイナリファイルサイズの評価（単位：byte）

#	ストリームデータ処理の機能	text	data	bss	合計
1	Filter オペレータ対応	58,365	520	564	59,449
2	Union オペレータ対応	59,978	512	564	61,054
3	Map オペレータ対応	62,070	516	564	63,150
4	Join オペレータ対応	80,723	528	592	81,843
5	Aggregate オペレータ対応	84,709	540	688	85,937
6	全オペレータ対応	119,224	556	720	120,500
7	全オペレータ対応，TCP，UDP 通信対応	132,083	624	788	133,495

表 7 Altera ボード上のバイナリファイルサイズの評価（単位：byte）

#	ストリームデータ処理の機能	text	rodata	rwdata	bss	合計
1	Filter オペレータ対応	35,552	4,407	172	8,744	48,873
2	Union オペレータ対応	36,128	4,139	180	9,176	49,623
3	Map オペレータ対応	34,824	4,435	172	8,908	48,339
4	Join オペレータ対応	41,888	4,571	192	1,058	57,239
5	Aggregate オペレータ対応	45,944	5,564	212	12,708	64,428
6	全オペレータ対応	62,304	6,180	212	12,708	81,404

換えが容易ではないため難しい．

組込みシステム向けのストリームデータ処理としては，Gigascope[13]，Muller らの方式[14]が挙げられる．Gigascope は基地局等にあるネットワーク機器向けのストリームデータ処理であり，あらかじめクエリから C/C++のソースコードを生成することで高スループットな処理を実現する．Gigascope では提案方式と同様にクエリからソースコードを生成しているがその方法は公開されていない．また実行環境のハードウェア/OS に従ったコンポーネントの選択については検討されていない．Muller らの方式は無線センサノード向けのストリームデータ処理である．クエリから独自の中間言語に変換し仮想マシン上でその中間言語を実行する．提案方式とは異なりクエリからソースコードを生成しないためバイナリファイルサイズが大きい．また実行環境のハードウェア/OS に従ったコンポーネントの選択については検討されていない．

組込みシステム向けのソフトウェアとしては，組込みデータベース[18]や，Matlab/Simlink の Real-Time Workshop などが挙げられる．これらのソフトウェアでは提案方式と同様に，特定の言語で記述されたクエリやプログラムからソースコードへの変換が行われる．しかし提案方式ではストリームデータ処理に特化したオペレータ，ストリーム，スケジューラに着目しコンポーネントを設計しているため，ストリームデータ処理以外のソフトウェアとは構成が異なる．

6. おわりに

本稿では，車載システム向けのストリームデータ処理方式を提案した．提案方式では従来の汎用システム向けのストリームデータ処理とは異なり，処理内容や，実行環境のハードウェア/OS に従ってストリームデータ処理のコードを静的に生成する．そのためにストリームデータ処理を複数のコンポーネントから構成し，あるコンポーネントは他コンポーネントに影響することなく，追加，削除，置換えを可能とする．提案方式を実装し，情報系及び制御系 ECU を想定した組込み環境で評価した．その結果，提案方式は従来のストリームデータ処理とは異なり，情報系及び制御系 ECU を想定した環境のハードウェア/OS への対応が容易であることを示した．またバイナリファイルサイズについて処理内容により最大で 49%削減できることを示した．今後の課題としては，車載システムを想定した処理を用いて性能を詳細評価することが挙げられる．

謝辞 本研究の一部は科研費 (22240003) の助成を受けている．

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, 2006.
2) W.D. Jones: Keeping Cars from Crashing, IEEE Spectrum, 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, 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, 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, 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,

Vol.2011-OS-119 No.1
Vol.2011-EMB-23 No.1
2011/11/29

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, 2003.
11) A. Arasu, S. Babu, and J. Widom: The CQL continuous query language: semantic foundations and query execution, The VLDB Journal, 2006.
12) Borealis Distributed Stream Processing Engine, http://www.cs.brown.edu/research/borealis/public/.
13) C. Cranor and V. Shkapenyuk: Gigascope: A Stream Database for Network Applications, SIGMOD, 2003.
14) R. Muller: Data Stream Processing on Embedded Devices, Degree of Doctor of Sciences, 2010.
15) ZMP: RoboCarR 1/10 / ZMP RC-Z, http://www.zmp.co.jp/e-nuvo/jp/robocar-110.html.
16) Altera: Nios II 3C25 Microprocessor with LCD Controller Data Sheet, http://www.altera.co.jp/literature/ds/ds_nios2_3c25_lcd.pdf
17) 山口晃広, 山田真大, 勝沼聡, 本田晋也, 佐藤健哉, 高田広章: 車載 DSMS における静的クエリ最適化, WebDB Forum, 2011.
18) ユビキタス: DeviceSQL, http://www.ubiquitous.co.jp/products/middleware/devicesql/.
19) ITRON WG/調査研究 WG: 2009 年度組込みシステムにおけるリアルタイム OS の利用動向関するアンケート調査結果報告, http://www.t-engine.org/ja/wp-content/themes/wp.vicuna/pdf/ja/TEF071-W002-100625_02.pdf