社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－DBS－143 (56)
2007／7／3

ストリームデータ処理におけるデータベースアーカイブ処理高速化の提案と評価

樫山 俊彦 花井 知広 田中 美智子 今木 常之 西澤 格

株式会社 日立製作所 中央研究所 〒185-8601 東京都国分寺市東恋ヶ窪 1-280
E-mail: toshihiko.kashiyama.ez@hitachi.com, tomohiro.hanai.sk@hitachi.com,
michiko.tanaka.ry@hitachi.com, tsuneyuki.imaki.nn@hitachi.com, itaru.nishizawa.cw@hitachi.com

あらまし RFID やセンサデータなど高いレートで生成されるデータを継続的にリアルタイム処理するストリームデータ処理が重要性を増している．ストリームデータ処理システムでは，データの永続化を行わないため，事後解析やロギングのための DB アーカイブも必須である．しかしながら，DB アーカイブはコミット処理や通信による遅延が発生するため，アーカイブ処理によりリアルタイム処理性能も低下する問題があった．本稿では，DB アーカイブ高速化手法である非同期バルクストア，および処理性能を維持しつつも処理結果が DB に格納されていることを保証するアーカイブバッファ同期バルクストアを提案した．プロトタイプシステムによる評価の結果，アーカイブ処理性能がリアルタイム処理出力量を下回る DB 限界に達する以前では，アーカイブバッファ同期バルクストアは，アーカイブしない場合と比較し，スループットが最大 5.4%減，レイテンシが最大 200 ミリ秒以内となり，リアルタイム処理性能を維持可能であることを示した．さらに，提案手法を組み込んだスマートシェルフシステムを構築し，高速化の効果を確認した．

キーワード ストリーム，アーカイブ，データベース，連続問合せ，RFID

Proposal and Evaluation of Fast Database Archive Method in Stream Data Processing

Toshihiko KASHIYAMA, Tomohiro HANAI, Michiko TANAKA,
Tsuneyuki IMAKI and Itaru NISHIZAWA

Hitachi, Ltd.,Central Research Laboratory, 1-280 Higashi-Koigakubo, Kokubunji-shi, Tokyo, 185-8601 Japan
E-mail: toshihiko.kashiyama.ez@hitachi.com, tomohiro.hanai.sk@hitachi.com,
michiko.tanaka.ry@hitachi.com, tsuneyuki.imaki.nn@hitachi.com, itaru.nishizawa.cw@hitachi.com

Abstract RFID technology and sensor network systems produce huge amount of data, and stream data processing, which continuously processes the data in the real-time fashion, has proposed and accepted as a new data processing paradigm. For post hoc analysis or logging, database (DB) archive is required because stream data processing system disposes used data. However, archive processing throughput is low because DB archive contains disk I/O, and real-time data processing throughput is reduced by DB archive. This paper proposes two faster archive method, Asynchronous Bulk Store method and Archive-buffer-Synchronous Bulk Store method. The latter method can guarantee processing data are output after they are archived to DB. The result of evaluation in the prototype system shows throughput drop within 5.4% compared to No-Archive, and latency is within 200 milli-seconds unless real-time data processing throughput exceeds max DB archive throughput. Moreover, proposed archive methods are of advantage in smart-shelf system which is one of RFID applications.

Keyword Stream, Archive, Database, Continuous Query, RFID

1. はじめに

ユビキタス社会の到来に伴い，RFID (Radio Frequency IDentification)タグを利用したトレーサビリティ/在庫管理/生産管理，センサノードによる監視/計測，株価情報を用いたアルゴリズム取引，SOX (Sarbanes-Oxley)法対応の業務モニタリング(Business Activity Monitoring － BAM)，システムの運用管理を容易にするシステムモニタリングなど，毎秒数百～数十万個という高いレートで生成されるデータをリアルタイムに処理する業務が増加している．

このような要求が増大する中，データが一つ生成される時にそれに起因する結果の差分のみを計算することで処理量を削減し，高レートデータのリアルタイム処理を実現するストリームデータ処理方式が注目され

ている[1][2][3][4][5]. 我々の研究グループでは，ストリームデータ処理システムのプロトタイプを開発している.本プロトタイプシステムは,業務毎に異なるデータ処理の内容を，RDBMS の標準検索言語 SQL(Structured Query Language)と同様の宣言型言語で定義できるため，RDBMS の開発容易性を満たしつつ，高速処理が可能となる.

プロトタイプシステムなどストリームデータ処理システムでは，データの永続化を行わないため，事前に登録した条件以外ではデータを検索できない．しかし，実応用では過去のデータに対して条件を変更して検索したり，過去のデータと現在のデータを比較したりするニーズがある．また，法令により履歴データの保存や検索を義務付けられる場合もある．そのため，ストリームデータ，またはクエリ処理結果のうち，必要となるデータをアーカイブする機構が必要となる．事後検索の容易性からアーカイブ先として，DB へのアーカイブが有効である．従来，DB へのアーカイブ処理が可能なストリームデータ処理システムも提案されている[6][7][8][10].

しかしながら，DB アーカイブはコミット処理や通信による遅延が発生するため，アーカイブ処理性能が低く，それに伴いリアルタイム処理性能が劣化する問題があった．本稿では，DB アーカイブ高速化手法である非同期バルクストア，および処理性能を維持しつつも処理結果が DB に格納されていることを保証するアーカイブバッファ同期バルクストアを提案する．提案手法を組み込んだスマートシェルフシステムを構築し，高速化の効果を確認する.

次節以降の構成は以下の通りである． 2節で，プロトタイプシステムの構成，応用例におけるシステム要件定義を示す．3節，4節ではそれぞれ，提案する非同期バルクストア方式，およびアーカイブバッファ同期バルクストア方式の説明，および提案方式の実装に対する評価実験の結果を示す． 5節で関連研究について述べ，最後に6節でまとめと今後の課題を述べる.

2. プロトタイプシステムとシステム要件定義

本節では，プロトタイプシステムの構成，およびスマートシェルフシステムを応用例としたシステム要件定義を示す.

2.1. プロトタイプシステムの構成

プロトタイプシステムの構成図を図 1に示す．プロトタイプシステムでは,RFID タグやセンサノードから読み出された時々刻々と到着する実世界の情報がストリームデータとして入力される．ユーザは，クエリを

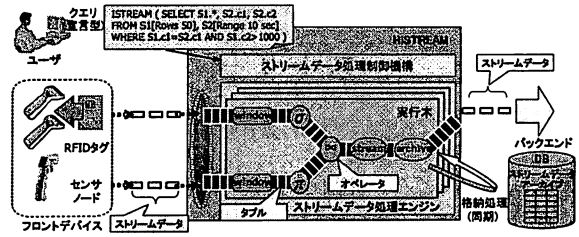

図 1: プロトタイプシステムの構成

予め登録しておくことで入力データをリアルタイム処理することができる．ストリームデータ処理エンジンでは，データ処理単位のオペレータが木構造に配置される(本構造を実行木と呼ぶ)．ストリームデータは，指定されたデータ数，または時間間隔に応じてその処理対象範囲がタプル(データレコード)として切り出される．その後，タプル集合に対してデータ処理(選択処理，射影処理，結合処理など)が実行され，最後に必要に応じて再びストリームデータとして出力される.

また，アーカイブオペレータは，DB への格納処理を行い，格納処理完了後に処理結果を出力する．以下では，同期単一タプルストア(Synchronous Single-tuple Store: SSS)と呼ぶ.

2.2. システム要件定義

ストリームデータ処理の応用例として想定される RFID タグやセンサノードを用いたシステムでは,様々な実証実験が行われ，実用化が進んでいる．例えば，商品に RFID タグを付加し，棚から取り出された商品の情報を提供するスマートシェルフの実証実験が大手百貨店において実施された[9]．スマートシェルフシステムでは，取り出された商品，および関連する情報を表示することができる．スマートシェルフシステムの場合，リアルタイム表示を可能とするため，RFID リーダが約 1 秒に 1 回タグ情報を送信する．よって，処理しなければならないデータは膨大な量となり，数千から数万タプル/秒の入力データになると推測される.

我々の研究グループでは，上記のようなスマートシェルフシステムに対してストリームデータ処理を用いることで，さらに進んで顧客が取り出した複数の商品からの興味情報抽出や棚ごとの在庫集計をリアルタイムに処理することを可能としている．デモシステムは，携帯電話の店頭販売を対象としており，システムの概要を図 2に示す.

本デモシステムにおけるデータ処理の要件として，(1)商品タグ読み出し情報から手に取りイベントへの変換，(2)来店客が手に取った商品（携帯電話）のタグ ID と商品情報の対応付け，(3)来店客が手に取っている複数の商品からの来店客興味情報の抽出，(4)手に取

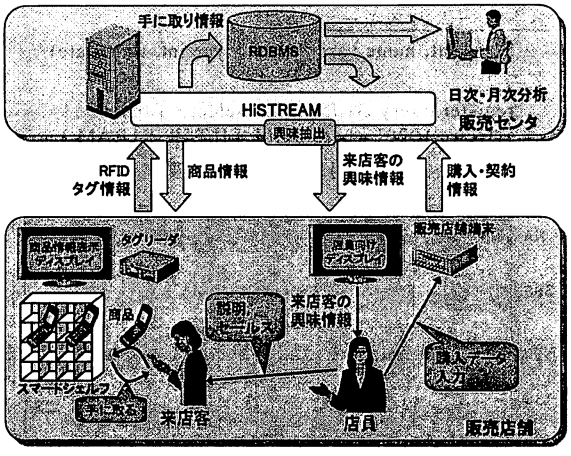

図 2: スマートシェルフシステム

りイベントや購入情報，在庫集計情報などの DB へのアーカイブ，が挙げられる．

図 1に示す従来のプロトタイプシステムのアーカイブ処理では，出力データが確実に DB へ格納されていることを保証するため，DB への格納処理完了後に次のタプルを処理していた．しかし，DB への格納処理はレイテンシが大きく，アーカイブ処理によりリアルタイム処理のスループットが低下することが予想された．モデルケースとして上記システム要件を満たすクエリ，およびアーカイブ処理を定義し，プロトタイプシステムの評価を行ったところ，アーカイブしない場合のリアルタイム処理性能が，アーカイブした場合のリアルタイム処理性能の約 1/4 となり，スマートシェルフシステムで想定される処理性能に届かなかった．想定性能に到達させるため，リアルタイム処理に影響を与えないアーカイブ処理，およびアーカイブ処理のスループット向上が必要であった．

3. アーカイブ処理高速化の提案手法

本研究で提案するアーカイブ処理方式である非同期バルクストア方式，およびアーカイブバッファ同期バルクストア方式を示す．

3.1. 非同期バルクストア

図 3に提案する非同期バルクストア(Asynchronous Bulk Store: ABS)を示す．従来方式である SSS では，DB 格納処理を行うアーカイブオペレータが実行木中に配置されていた．そのため，レイテンシの大きいアーカイブオペレータに起因してリアルタイム処理のスループットが低下していた．ABS では，DB 格納処理を待たずに他のオペレータを処理する方式とする．すなわち，アーカイブオペレータを実行木外に配置し，アーカイブ処理を非同期化することで，リアルタイム処理

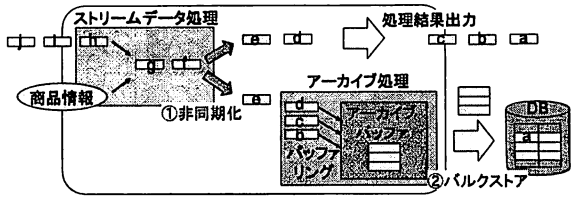

図 3: 非同期バルクストア

を高速化する(図 3①).

また，SSS では DB 格納処理完了後に処理結果を出力するため，1 タプルずつ DB 格納処理を行っていた．そのため，必要なアーカイブ処理性能を確保できなかった．ABS では，アーカイブオペレータ中に一定時間のタプルをバッファリングするアーカイブバッファを用意する．そして，バッファリングしたデータを DB にまとめて格納(バルクストア)することで DB 格納処理のレイテンシを隠蔽し，アーカイブ処理を高速化する(図 3②).

以下では，非同期化のみを行う方式を非同期単一タプルストア(Asynchronous Single-tuple Store: ASS)と呼び，アーカイブ処理を行わない場合を NA(No Archive)と呼ぶ．

3.2. アーカイブバッファ同期バルクストア

3.1節に示した ABS により，リアルタイム処理性能を維持しつつ，アーカイブ処理を高速化することが可能となる．しかしながら，ABS とすることでアーカイブが完了していないデータを出力してしまうため，ストリームデータ処理システムのダウンによりアーカイブデータ欠損が生じる可能性があり，出力データが確実に DB へ格納されていることを保証できない．

RFID タグやセンサノードなどを扱うシステムは，必要に応じてタグ情報やセンサ情報を再度取得することでデータ欠損を許容できる場合もある．しかしながら，データ欠損を許容できるシステムは一部であり，データ欠損がないことを保証しなければならない場合もある．

上記のようなニーズに対応するため，図 4に示すアーカイブバッファ同期キューを用意し,クエリの処理結果を出力する際に，アーカイブバッファ同期キューに入力する．アーカイブバッファ同期キューでは，アーカイブバッファにバッファリングされたデータが DB にアーカイブ完了したかを確認し，アーカイブ完了後に処理結果を出力する．

以下では，本方式をアーカイブバッファ同期バルクストア(Archive-buffer-Synchronous Bulk Store: ASBS)と呼ぶ．ASBS では，アーカイブ完了確認を行うオーバーヘッドが伴うものの，リアルタイム処理とアーカ

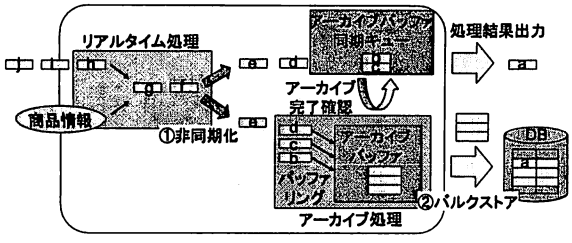

図 4: アーカイブバッファ同期バルクストアイブ処理を非同期で処理することが可能なため，リアルタイム処理を高スループットに行うことができる．しかし，DB 格納処理完了まで処理結果が出力できないため，NA や ABS と比較しレイテンシが大きくなる．また，クエリを多段につなげた場合に，アーカイブ完了後のデータがまとめて出力されるため，処理結果出力タイミング，およびレイテンシが一定とならない問題も生じる．処理結果出力タイミング，およびレイテンシが一定とならない問題に関しては本稿では検討せず，今後の課題とする．

以上より，各アーカイブ方式をまとめると表 1となる．

表 1: アーカイブ処理方式比較

	NA	SSS	ASS	ABS	ASBS
リアルタイム処理性能	○	×	○	○	○
アーカイブ処理性能	−	×	×	○	○
レイテンシ	○	×	○	○	△
結果格納保証	−	○	×	×	○

4. 評価

提案アーカイブ方式の効果を確認するため，比較評価を行う．なお，測定環境は以下の表の通りである．

表 2: 測定環境

	プロトタイプサーバ	DB サーバ
CPU	Intel Core2 Duo E6600 (2.4GHz)	Intel Xeon 2.4GHz
メモリ	1GB	2GB
OS	Windows XP Professional SP2	Windows Server 2003
備考	JRE 1.5.0_11 VM メモリ割り当て：512MB	PostgreSQL 8.1

4.1. 実験 1: アーカイブ方式比較

図 5に示すスキーマのストリーム S1，および基本クエリ Q1 によってベンチマークを実施した．ここで，入力データはカラム id が 0 から 999 までランダムに発生する系列とし，500,000 個のデータを事前に入力し，最大入力スループットを測定した．また，FILTER_PARAM (選択条件指定，この値によりアーカイブ量が異なる) を 5 (入力の 0.5%をアーカイブ) とし，バルクストアのバッファリング間隔を 16 ミリ秒とした．なお，FILTER_PARAM により設定される入力データに対するアーカイブデータの割合をアーカイブ割合と定義する．

ストリームのスキーマ

S1(id int, name varchar(10), type int, time date)

クエリ

```
register query Q1 istream ( select * from
s1[rows 1000] where S1.id<FILTER_PARAM )
```

図 5: 実験に用いたスキーマ及びクエリ

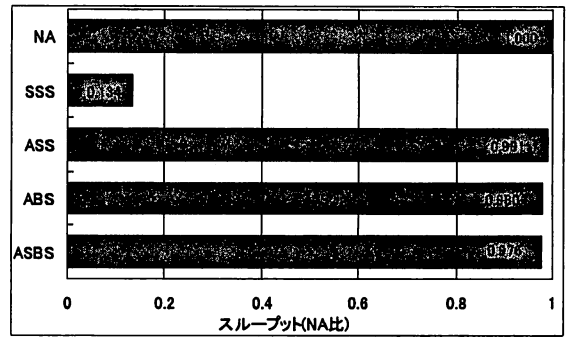

図 6: アーカイブ方式比較(相対値)

図 6に各アーカイブ方式における入力スループット(対 NA 比の相対値)を示す．SSS は，DB 格納処理のコストが大きく，スループットが約 1/7 に低下してしまう．ASS は，NA と比較し 0.9%のスループット低下となり，非同期化によるリアルタイム処理高速化が実現されている．しかし，単一タプルアーカイブでは，処理結果出力スループットに対し，DB 格納処理スループットが下回り(詳細は4.2節に示す)，アーカイブバッファにバッファリングされているタプルが増加し続けてしまう．ここで，DB 格納処理スループットが処理結果出力スループットを下回ることをアーカイブ失敗と定義する．実システムにおいては，アーカイブバッファにバッファリングされているデータ量がメモリ量を超えない限りは，アーカイブ負荷減少時にアーカイブ処理が追いつくことで回復できる場合もあるが，本稿では上記は考えないものとする．

ABS では，NA と比較し 2.0%のスループット低下となる．本パラメータの場合，ASS がアーカイブ失敗となるのに対し，ABS ではすべてのデータをアーカイブ処理可能であるため，ASS よりもアーカイブ処理コストが大きくなり，スループットが低下する．ASBS では，NA と比較し 2.4%のスループット低下となる．ABS からさらにアーカイブバッファ同期のコストが加わるため，ABS よりもスループットが低下するものの，処理結果が DB に格納されていることを保証できる．

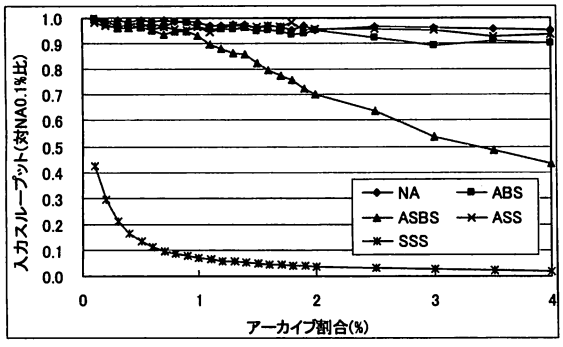

図 7: アーカイブ割合と入力スループット(相対値)

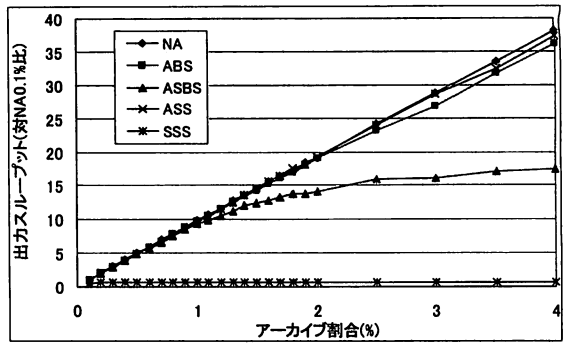

図 8: アーカイブ割合と出力スループット(相対値)

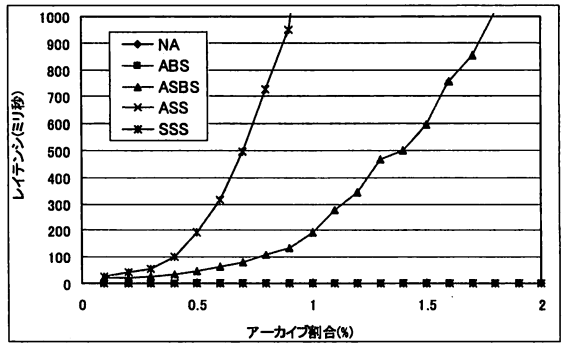

図 9: アーカイブ割合とレイテンシ

4.2. 実験 2: アーカイブ割合

次に，アーカイブ割合を変動させた場合のスループット，レイテンシを測定する．図 5に示すスキーマ・クエリにおいて，また，FILTER_PARAM を 1(0.1%アーカイブ)から 40(4%アーカイブ)まで変動させる．入力データ，およびバッファリング間隔は実験 1 と同一である．

図 7に入力スループット(対 NA0.1%比の相対値)の測定結果を示す．SSS は，アーカイブ割合が増加に伴い，スループットが減少していく．ASBS は，アーカイブ割合が 1.0%までは，最大で NA 比 5.4%，ABS 比で 3.9%(いずれも 1.0%アーカイブ)のスループット低下に留まり，ほぼ同等のスループットを確保できる．しかし，アーカイブ割合が 1.1%以降では，スループットが低下していく．ASS，ABS は，NA 同等のスループットとなるとなるものの，ASS では 0.1%，ABS では 1.1%以降において，DB 格納処理スループットがリアルタイム処理スループットを下回るアーカイブ失敗状態となる．

アーカイブ失敗状態を確認するため，縦軸を出力スループット(入力スループット×アーカイブ割合)とすると，図 8に示すグラフとなる．なお，出力スループットは対 NA0.1%比の相対値であり，アーカイブ処理スループットと同一となる．

SSS では，単一タプルのアーカイブとなるため，DB 格納のスループットが低くなり，出力スループットが 0.5 程度で一定となってしまう．この値を単一タプルストア限界スループットとする．ASBS では，バルクストアにより DB 格納のスループットが上昇するため，単一タプルストア限界スループットを超える出力スループットを得ることができる．しかしながら，アーカイブ割合が増加すると,出力スループットは 17 程度で一定となってしまう．この値をバルクストア限界スループットとする．1.1%以降では，ABS はアーカイブバッファにデータが増え続ける状態となり，アーカイブ失敗となる．ここで，ABS においてアーカイブ失敗するアーカイブ割合をバルクストア限界点とする．ベンチマーククエリの場合，バルクストア限界点は 1.1%となる．

図 7に示すように，ASBS は，バルクストア限界点以降ではバルクストア限界スループットに引っ張られ，入力スループットが急激に低下する．すなわち，図 7において，アーカイブ割合がバルクストア限界点までは，プロトタイプシステムにおけるリアルタイム処理ネック，バルクストア限界点以降は DB 格納処理ネックとなる．

続いて，アーカイブ割合を変動させたときのレイテンシを測定する．図 9にレイテンシの測定結果を示す．NA，ASS，ABS がアーカイブ割合に依存せず 1 ミリ秒以内となるのに対し，ASBS は，DB 格納処理完了までの時間が含まれるため，NA や ABS と比較しレイテンシが最小でも数十ミリ秒となる．DB 限界点に向けレイテンシはさらに増加し，DB 限界点を境に急激に上昇する．これは，フィルタ処理を完了し，アーカイブバッファに入っている DB 格納処理待ちのデータが増えるためである．SSS では，本現象がさらに顕著になり，急激にレイテンシが上昇する．

ASBS はレイテンシが大きくなるという問題を抱えるが，DB 限界点付近までは，200 ミリ秒以内(1.0%アー

カイブ)のレイテンシとなる．そのため，商品を取り出した瞬間に商品情報を表示するスマートシェルフのようなシステムにおいて，バルクストア限界点を超えない範囲でアーカイブする場合，レイテンシは許容可能な値となる．

4.3. 実験 3: スマートシェルフシステム

最後に，2.2節で述べたスマートシェルフシステムにおけるスループットを測定した．2.2節に示した要件を満たす 10 個のクエリを実行したところ，NA に対しそれぞれ，ABS が 0.2%，ASBS が 0.3%のスループット減となり，同程度のスループットとなることが確認できた．ここで，スマートシェルフのアーカイブ量は図 9 に示すアーカイブ割合の 0.2%程度であるため，レイテンシも問題とはならなかった．

以上の実験より，DB 限界とならない範囲では，NA と同程度のスループットを確保可能であり，レイテンシが実システムにおいて許容範囲内となる ASBS が有効であることを確認した．

5. 関連研究

ストリームデータ処理システムでは，アーカイブ処理や，マスタ表参照など，RDBMS との連携の必要性が認識されてきている．

StreamBase では，ストリームデータ処理システム上に BerkleyDB を組込み，アーカイブ処理を行うことが可能である[6]．Coral8 は DB2 と連携したアーカイブ処理が可能である[10]．Coral8 では，アーカイブ処理方式として，非同期書き込み，バッチ書き込みをサポートしているが，ABS と同等の方式と推測される．

StreamSpinner では，ストリームデータ処理システムにおいて，コスト式を用いてアーカイブ処理可能か計算し，ユーザに提示することが可能である[7]．また，アーカイブ時の複数問合せ最適化を行うことで DB アーカイブ量を減少させることができる[8]．しかしながら，バルクストアなどアーカイブ処理を高速化する手法に関しては検討されていない．

STREAM では，ストリームデータ処理システム上にリレーションデータを保持し，ストリームデータ，およびリレーションデータのジョイン演算を行うことが可能であるが，DB へのアーカイブ処理は検討されていない[2]．

6. おわりに

本稿では，ストリームデータ処理システムのプロトタイプシステム，および DB の処理性能の違いに起因するアーカイブ時のリアルタイム処理性能低下，アーカイブ処理性能不足に対して，DB アーカイブ高速化手法である非同期バルクストア，および処理性能を維持しつつも処理結果が DB に格納されていることを保証するアーカイブバッファ同期バルクストアを提案した．評価の結果，アーカイブ処理がリアルタイム処理に追い付かなくなる DB 限界に達する以前では，アーカイブバッファ同期バルクストアは，アーカイブなしと比較し，スループットが最大 5.4%減，レイテンシが最大 200 ミリ秒以内となり，リアルタイム処理性能を維持可能であることを示した．さらに，提案手法を組み込んだスマートシェルフシステムを構築し，高速化の効果を確認した．

今後の課題として，提案アーカイブ処理を含めた処理性能モデルの定式化，および処理結果出力のレイテンシが一定とならない問題の解消が挙げられる．

文　献

[1] Nauman, Chaudhry,Kevin, Shaw,Mahdi, Abdelguerfi, *Stream Data Management*, Springer (2005).

[2] Motwani, et al. "Query Processing, Resource Management, and Approximation in a Data Stream Management System", CIDR 2003 (2003).

[3] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik, "The design of the Borealis stream processing engine", In Proc. of CIDR 2005, pp.277-289 (2005).

[4] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous dataflow processing for an uncertain world", In Proc. of CIDR 2003, pp. 269-280 (2003).

[5] N. Jain, L. Amini, H. Andrade, R. King, Y. Park, P. Selo, and C. Venkatramani, "Design, Implementation, and Evaluation of the Linear Road Benchmark on the Stream Processing Core", In Proc. of SIGMOD 2006, pp.431-442 (2006).

[6] M. Stonebraker and U. Cetintemel, "One Size Fits All: An Idea Whose Time has Come and Gone", In Proc. of ICDE'05, pp.2-11 (2005).

[7] 山田真一，渡辺陽介，北川博之，天笠俊之．" ストリーム管理システムにおける永続化要求の妥当性評価"，電子情報通信学会技術研究報告 Vol.106, No.149, pp.209-214 (2006).

[8] 山田真一，渡辺陽介，北川博之，天笠俊之，"ストリーム管理システムにおける複数永続化要求最適化手法"，データ工学ワークショップ DEWS2007 (2007).

[9] 経済産業省平成 17 年度電子タグ実証実験事業の成果について．http://www.meti.go.jp/policy/it_policy/tag/tagzissyouzikken.htm.

[10] Coral8 and DB2 9. http://www.coral8.com/products/DB2.html.