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

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

樫山 俊彦* 花井 知広* 田中 美智子*
今木 常之* 西澤 格*

Toshihiko KASHIYAMA Tomohiro HANAI
Michiko TANAKA Tsuneyuki IMAKI
Itaru NISHIZAWA

RFID やセンサ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，データを永続化しないため，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が，格納処理に伴う遅延によるリアルタイム処理性能低下の問題があった．本稿では，アーカイブ高速化手法である非同期バルクストア，処理性能を維持しつつ処理結果の DB 格納保証が可能なアーカイブバッファ同期バルクストアを提案した．アーカイブ処理量がリアルタイム処理出力量を下回るバルクストア限界点以前では，アーカイブバッファ同期バルクストアは，スループットがアーカイブなしと比較し最大 5.4%減，レイテンシが 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．

RFID and sensor network systems produce huge amount of data, and stream data processing (SDP) which continuously processes the data has proposed as a new data processing paradigm. For future data analysis or logging, database (DB) archive is required because SDP system disposes used data. However, real-time data processing throughput is reduced because DB archive contains additional store processing costs. This paper proposes two faster archive methods, Asynchronous Bulk Store (ABS) and Archive-buffer-Synchronous Bulk Store (ASBS) for SDP. ASBS guarantees that the processed data is archived into DB before the data is output from SDP system. System throughput drops by 5.4% compared to No-Archive case, and data processing latency is kept within 200 msec unless real-time data processing throughput exceeds maximum DB archive throughput.

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．

これらの要求に対し，データ生成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が注目されている[1][2]．我々の研究グループでは，業務毎に異なるデータ処理内容を DB 標準検索言語 SQL (Structured Query Language)と同様の宣言型言語で容易に定義できるストリームデータ処理のプロトタイプを開発している．

ストリームデータ処理では，データ永続化なしで検索するため，事前に登録した条件以外で検索できない．しかし，実応用では過去のデータに対する検索条件の変更や，過去のデータと現在のデータの比較等のニーズがある．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，必要となるストリームデータ，またはクエリ処理結果をアーカイブする機構が必要となる．事後検索容易性からアーカイブ先は DB が有効であり，DB へのアーカイブ処理が可能なシステムも提案されている[3][4][6]．

しかしながら，DB アーカイブは格納処理に伴う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．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する．

次節以降の構成は以下の通りである．2 節では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，アーカイブバッファ同期バルクストア方式の説明，および提案方式実装の評価実験結果を示す．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．

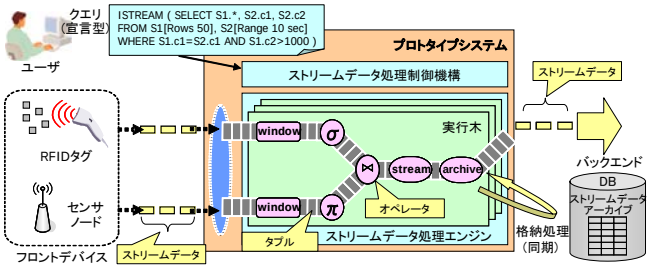

図 1 プロトタイプシステムの構成
Fig.1 Configuration of Prototype System

2. プロトタイプシステムと要件定義

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．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，クエリを予め登録することで入力データをリアルタイム処理できる．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

*正会員 株式会社日立製作所 中央研究所
{toshihiko.kashiyama.ez, tomohiro.hanai.sk, michiko.tanaka.ry, tsuneyuki.imaki.nn, itaru.nishizawa.cw}@hitachi.com

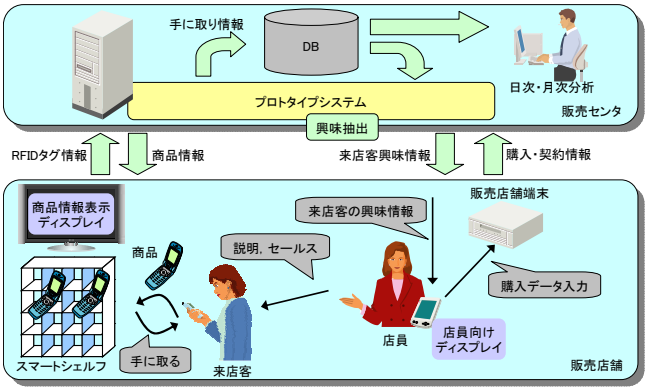

図 2 スマートシェルフシステム
Fig. 2 Smart Shelf System

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 へ格納処理し，格納処理完了後に処理結果を出力する．以下では，本アーカイブ方式を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[5]．スマートシェルフシステムは，取り出された商品，および関連する情報を表示でき，リアルタイム表示するため約 1 秒に 1 回 RFID リーダがタグ情報を送信する．よって，入力データは膨大な量となり，数千から数万タプル/秒と推測される．

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ストリームデータ処理を適用し，図 2 に示す携帯電話の店頭販売を対象としたデモシステムを構築した．本システムは，適用しない場合から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できる．本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)来店客が手に取った複数の商品からの来店客興味情報の抽出，(4)手に取りイベントや購入情報，在庫集計情報などの DB アーカイブ，が挙げられる．

図 1 に示す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 へ格納されることを保証するため，DB 格納処理完了後に次のタプルを処理していた．しかし，DB 格納処理はレイテンシが大きく，アーカイブ処理に起因したリアルタイム処理スループット低下が予想された．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評価したところ，アーカイブした場合のリアルタイム処理性能が，アーカイブしない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の目標処理性能に届かなかった．目標処理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．

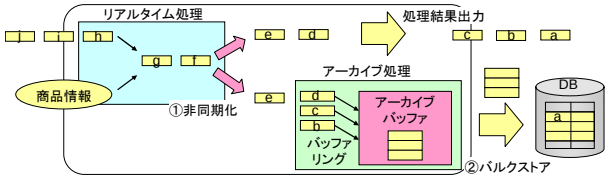

図 3 非同期バルクストア
Fig. 3 Asynchronous Bulk Store

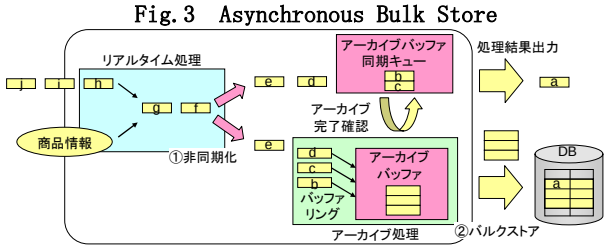

図 4 アーカイブバッファ同期バルクストア
Fig. 4 Archive-buffer-Synchronous Bulk Store

3. アーカイブ処理高速化の提案手法

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．従来方式の SSS では，DB へ格納処理するアーカイブオペレータが実行木中に配置され，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スループットが低下していた．ABS では，DB 格納処理完了を待たずに他のオペレータを実行する．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 格納処理しており，必要なアーカイブ処理性能を確保できなかった．ABS では，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できる．しかしながら，ABS ではアーカイブ完了していない処理結果を出力してしまうため，ストリームデータ処理システムダウン時にアーカイブデータ欠損が発生する可能性がある．よって，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を扱うシステムでは，必要に応じて再度情報を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，データ欠損なしを保証しなければならない場合もある．

上記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入力する．アーカイブバッファ同期キューは，アーカイブバッファ中のデータの DB 格納処理完了を確認し，完了後にキューから出力する．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のオーバーヘッドが

表 1　測定環境
Table 1　Measurement Environment

	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
N/W	1000BASE-T	1000BASE-T
OS	Windows XP	Windows Server 2003
備考	Sun JRE 1.5.0_11 VM メモリ割当：512MB	PostgreSQL 8.1

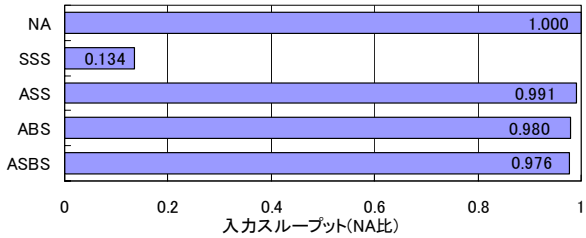

図 5　アーカイブ方式比較(相対値)
Fig. 5　Comparison of archive methods

伴うが，リアルタイム処理とアーカイブ処理を非同期で処理できるため，高スループットなリアルタイム処理が実現できる．しかし，DB 格納処理完了まで処理結果を出力できないため，NA や ABS と比較しレイテンシが増大する．また，アーカイブ完了後のデータがまとめて出力されるため，クエリを多段につなげた場合に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が，これらの問題に関しては本稿では検討せず，今後の課題とする．

4. 提案手法の評価

4.1　実験 1: アーカイブ方式比較

提案アーカイブ方式の効果を確認するため，表 1 に示す測定環境で，以下に示すスキーマのストリーム S1，および基本クエリによってベンチマークを実施した．

ストリームのスキーマ

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

クエリ

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

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た．また，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し，本実験ではアーカイブ割合をスマートシェルフシステムの 2 倍程度である 0.5%(FILTER_PARAM=5)とした．バルクストアのバッファリング間隔は 16 ミリ秒とした．

図 5 に各アーカイブ方式の入力スループット(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入力スループットが約 1/7 に低下する．ASS は 0.9%の低下であり，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，ASS 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，アーカイブバッファ中のタプルが増加し続けてしまう(詳細は 4.2 節に示す)．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．実システムでは，アーカイブバッファ中のデータ量がメモリ量を超えない限

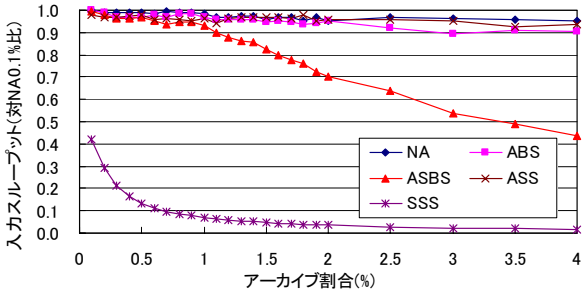

図 6 アーカイブ割合と入力スループット(相対値)
Fig. 6　Archive Ratio vs. Input Throughput

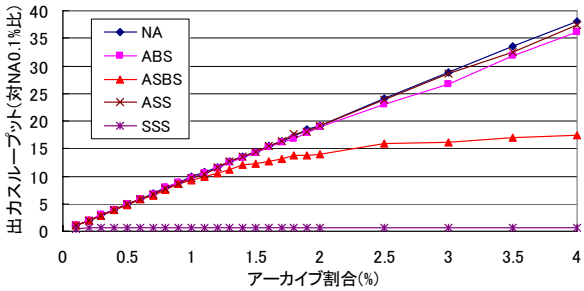

図 7　アーカイブ割合と出力スループット(相対値)
Fig. 7　Archive Ratio vs. Output Throughput

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場合もあるが，本稿では上記は考えないものとする．

ABS は 2.0%の低下となる．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なため，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 は 2.4%の低下となる．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．

4.2　実験 2: アーカイブ割合

次に，アーカイブ割合変動時のスループット，レイテンシを測定する．4.1 節のクエリにおいてアーカイブ割合を 0.1%から 4%(FILTER_PARAM を 1 から 40)まで変動させる．入力データ，バッファリング間隔は実験 1 と同一である．

図 6 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，アーカイブ割合増加に伴いスループットが低下する．ASBS は，アーカイブ割合 1.0%までは最大で NA 比 5.4%，ABS 比 3.9%の低下に留まり，ほぼ同等となるが，アーカイブ割合 1.1%以降ではスループットが低下する．ASS，ABS は NA と同等であるが，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処理結果出力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 7 に示すグラフとなる．出力スループットは対 NA0.1%比の相対値であり，アーカイブ失敗とならない範囲ではアーカイブ処理スループットと同一となる．SSS では，単一タプルアーカイブのため出力スループットが低く，約 0.5 で限界となる．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 格納処理スループットが上昇し，単一タプルストア限界スループットを上回る．しかしながら， アーカイブ割合増加に伴い約 17 で限界となる．この

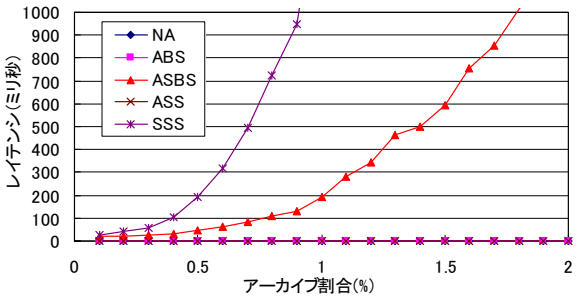

図 8 アーカイブ割合とレイテンシ
Fig.8 Archive Ratio vs. Latency

値をバルクストア限界スループットとする．ABS は，1.1%以降でアーカイブバッファ中の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 でアーカイブ失敗とな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 6 において，ASBS はバルクストア限界点以降で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，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，アーカイブ割合変動時のレイテンシを測定する．図 8 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 格納処理完了確認の時間が含まれるため大きくなる．また，バルクストア限界点に向け緩やかに増加し，限界点を境にアーカイブバッファ中の DB 格納処理待ちデータが増えるため急激に上昇する．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い問題を抱えるが, バルクストア限界点付近までは 200 ミリ秒以内(1.0%アーカイブ)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システムにおいて，バルクストア限界点以前で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入力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%減となり，同程度となることが確認できた．また，アーカイブ量は図 8 に示すアーカイブ割合の 0.2%程度であるため，レイテンシも許容可能である．

以上の実験より，バルクストア限界点以前では，NA と同程度のスループットを確保可能，かつ実システムでレイテンシが許容範囲内となる 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，DB 連携の必要性が認識されてきている．StreamBase では，システムに組込んだ BerkleyDB へアーカイブ処理できる[3]. Coral8 は DB2 と連携したアーカイブ処理が可能である[6]. 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，コスト式を用いてユーザにアーカイブ処理可能か提示できる[4]. また，アーカイブ時の複数問合せ最適化による DB アーカイブ量減少が可能である．しかしながら，バルクストアなどアーカイブ処理高速化の手法は検討されていない．STREAM では，リレーションデータとのジョイン演算が可能であるが，アーカイブ処理は検討されていない[1].

6. まとめと今後の課題

本稿では，ストリームデータ処理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バルクストア限界点以前では，アーカイブバッファ同期バルクストアは，スループットがアーカイブなしと比較し最大 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] R. Motwani, et al. "Query Processing, Resource Management, and Approximation in a Data Stream Management System", CIDR 2003, pp.245-256 (2003).
[2] D. J. Abadi, et al. “The design of the Borealis stream processing engine”, CIDR 2005, pp.277-289 (2005).
[3] M. Stonebraker, et al. “One Size Fits All: An Idea Whose Time has Come and Gone”, ICDE 2005, pp.2-11 (2005).
[4] 山田真一，他．"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).
[5] 経済産業省平成 17 年度電子タグ実証実験 http://www.meti.go.jp/policy/it_policy/tag/tagzissyouzikken.htm.
[6] Coral8 and DB2 9. http://www.coral8.com/products/DB2.html.

樫山 俊彦 Toshihiko KASHIYAMA
2005 年東京工業大学大学院情報理工学研究科修士課程修了．同年より(株)日立製作所中央研究所．FIT 2004 船井ベストペーパー賞受賞．電子情報通信学会会員．

花井 知広 Tomohiro HANAI
2004 年東京工業大学大学院情報理工学研究科修士課程修了．同年より(株)日立製作所中央研究所．情報処理学会会員．

田中 美智子 Michiko TANAKA
2006 年九州大学大学院システム情報科学府修士課程修了．同年より(株)日立製作所中央研究所．情報処理学会会員．

今木 常之 Tsuneyuki IMAKI
1996 年東京大学大学院工学系研究科修士課程修了．同年より(株)日立製作所中央研究所．情報処理学会会員．

西澤 格 Itaru NISHIZAWA
1996 年東京大学大学院工学系研究科博士課程修了．同年より(株)日立製作所中央研究所．主任研究員．2002-2003 年米スタンフォード大客員研究員．ACM, 情報処理学会各会員．