Vol.2010-DPS-142 No.48
Vol.2010-CSEC-48 No.48
2010/3/5

ストリーム型ログデータ蓄積処理向け 無停止 DB 分割方式の提案

長谷川 知洋，内藤 一兵衛，毛受 崇，
赤間 浩樹，山室 雅司†

ライフログ蓄積処理システム等で使用され，停止することが許されないストリーム型ログデータ蓄積用 DB が処理全体のボトルネックになることを防ぐために無停止で DB を分割する方式を提案する．具体的には，分割元 DB と分割先 DB の両方の DB に対するクエリ実行と分割元 DB から分割先 DB へのデータコピーを並行して行いながら，クエリの振り分け先を段階的に切り替えることで，無停止で DB を分割して処理のスケールアウトを実現する．評価実験の結果から，データがバースト的に追記された場合でも定常的に追記される場合でも無停止で DB 分割を実行しながら処理の動的スケールアウトが実現できることを示す．

Non stop Database Division Method for Data Stream Management System

Tomohiro Hasegawa, Ichibe Naito, Takashi Menjo,
Hiroki Akama and Masashi Yamamuro†

Various data streams are easily generated, like sensor data, logs on various information systems, video and photo data, etc. To deal with these stream data, we are developing Data stream Management Systems in WRITE ONCE and READ MANY manner. In this system, Lifelog data is stored to database in real-time processing. The database might become a bottleneck as data increases. To solve this problem, we propose Non stop Database Division method for Data Stream Management System, and describe some evaluation results.

1. はじめに

センシング機器から発生する情報や，携帯機器の発達に伴って収集される様々な個人の活動履歴情報，各種情報システムのログ情報など，多種多様な時系列データが時々刻々と大量に生成されている．これらのデータストリームを追記型で収集・蓄積・管理し，利用者の目的に応じて各種情報を処理・統合して参照側に提供する追記・参照型データ管理システムの研究開発に取り組んでいる[1][2]．現在開発中の追記・参照型データ管理システムDMS（Data Management System）は，1 台から数百台規模のPCクラスタ上で動作させることができ，情報源の追加や処理量の増加に対する処理の動的スケールアウト（規模成長）や，連続して到着するストリームデータに対する処理内容の動的変更（機能成長）を実現している[3]．

ストリーム型ログデータ活用サービスでは，ネットワーク上のセンタサーバで情報源ごとの膨大なログを長期間にわたって蓄積・管理したり，多数のデータから統計的な情報を生成したり，収集されるログの種類の増加に伴って既存の処理を変更したり，新たな処理を追加したりする必要がある．これらの要求を満たすために，システム無停止で規模成長と機能成長が実現可能な DMS をログ活用システムとして適用することを考え，ストリーム型ログデータ分散蓄積処理基盤の検討を行っている．

本論文では，2章でストリーム型ログデータ分散蓄積処理基盤について述べた後，3章でストリーム型ログデータを蓄積するDBがボトルネックとなった場合の解決手法として，無停止DB分割方式を提案する．4章では実験システムを用いて行った提案方式の評価結果を示し，5章でまとめと今後の課題を示す．

2. ストリーム型ログデータ分散蓄積処理基盤

2.1 追記・参照型データ管理システム DMS

追記・参照型データ管理システム DMS は図 1 に示すように，データの受付・蓄積を行う追記部 Q と，受け付けたデータを処理したり，蓄積済みのデータと統合したりするフィルタ部 F と，個々の応用にフィルタ部での処理結果を提供するビュー部 V とから構成される．サービスの規模に応じて 1 台から数百台規模の PC クラスタ上で動作させることができ，データ量／処理量の増加に対しては，フィルタ部 F を動的に追加していくことで処理のスループットを向上させ，サービス無停止での動的スケールアウト（規模成長）を実現している．また，フィルタ部 F は自身がデータを処理可能になったタイミングで追記部 Q からデータを取得してきて処理をするというサイクルを繰り返し実行しているため，追記部 F から次に処理すべきデータを取得する直前に自身の処理内容を動的に追加・変更（機能成長）することが可能である．

† 日本電信電話株式会社　NTT サイバースペース研究所
NTT Cyber Space Laboratories, NTT Corporation

Vol.2010-DPS-142 No.48
Vol.2010-CSEC-48 No.48
2010/3/5

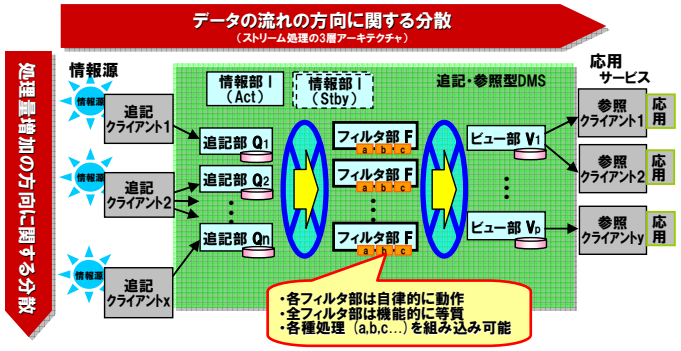

図 1　追記・参照型データ管理システム DMS

2.2　ストリーム型ログデータ分散蓄積処理基盤のアーキテクチャ

ストリーム型ログデータ分散蓄積処理基盤は，図 2 のように 2 段に配置した DMS とストリーム型ログデータ蓄積処理向け DB から構成される．1 段目の DMS（情報源毎ログ蓄積フェーズ）では，情報源側クライアントから逐次送られてくる個別のログを受け付けてストリーム型ログデータ蓄積処理向け DB に蓄積し，2 段目の DMS（サービス実行フェーズ）では，ストリーム型ログデータ蓄積処理向け DB に蓄積されたログを参照し，各種サービスごとに登録された処理を実行し，返却域に結果を出力するアーキテクチャとなっている．1 段目のフィルタ部を追加していくことにより，情報源の増加に対する処理のスループット向上に対応し，2 段目のフィルタ部を追加していくことにより，サービス数の増加に対する処理のスループット向上に対応可能な構成となっている．

各情報源がセンタシステムに対して，情報源側クライアント経由でログを送信したり，ログを活用したサービスの実行を要求したりすると，センタシステムは各リクエストに対する受付 ID を返却する．情報源側クライアントは返却された受付 ID をキーにして返却域からサービスの実行結果を取得する．このように，情報源側クライアントからのリクエストとサービスの実行結果取得のタイミングを非同期にすることにより，サービスの実行結果生成に時間がかかる場合でも情報源側クライアントが自身の都合が良いタイミングで処理結果を取得することができる．

2.3　ストリーム型ログデータ蓄積処理向け DB

情報源ごとに送られてくるログデータは 1 段目の情報源ログ蓄積フェーズで情報源毎ログ DB に蓄積される．この DB には GPS から取得した位置情報，購入した商品の履歴情報や店舗情報，Web サイトの閲覧履歴，撮影した写真のメタ情報などの様々な種類のログが蓄積されることが想定されるため，これらを統一的に扱えることが望ましい．また，今後新たに登場するかもしれない未知のセンサー機器や新たなサービス

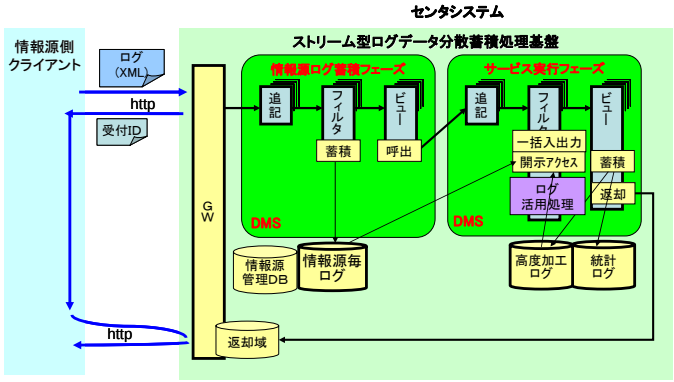

図 2　ストリーム型ログデータ分散蓄積処理基盤のアーキテクチャ

によって提供される情報など，DB に蓄積すべきログの種類は今後も増加していくことが予想される．

このように，ストリーム型ログデータ蓄積処理向けDBは様々な種類のログを統一的に扱うことができ，更に新たなログ項目の追加にも柔軟に対応可能なテーブル構造であることが求められる．そこで，Bigtableのデータモデル[4]を参考にしたKey-Value Storeライクなテーブル構造を採用し，新たなログ項目の追加時でも既存テーブルへのカラム追加ではなく，レコード追加として対応し，テーブル構造の変更を不要にしている（図 3）．

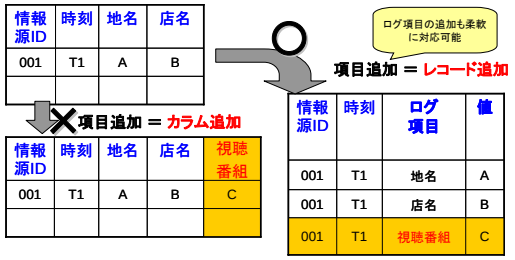

図 3　新たなログ項目追加への対応

Vol.2010-DPS-142 No.48
Vol.2010-CSEC-48 No.48
2010/3/5

2.4 従来技術と課題

ストリーム型ログデータ蓄積処理向け DB に大量のデータが蓄積され，情報源側クライアントから追記されるログデータの入力頻度や量が増加し，更にサービス実行フェーズで DB に蓄積されたデータを参照する頻度や量が増加してくると，いくらフィルタ部を追加してもストリーム型ログデータ蓄積処理向け DB がボトルネックとなって処理が遅延し，処理全体のスケールアウトが得られなくなることが予想される．このような場合は，論理的に 1 つのテーブルを物理的に複数のサーバ上に分散配置（水平分割）することで，個々のサーバ上で処理すべきテーブルのサイズが小さくなり，アクセス高速化の効果が期待できる．また，アクセス対象となるサーバが分散することによって入出力の分散による効率改善の効果が期待できる．ストリーム型ログデータ蓄積処理向け DB には，情報源単位に追記型でデータが登録され，レコードごとの独立性が高いという特長がある．そのため，情報源 ID などをキーにして，比較的容易に複数のサーバに分散配置することが可能である．

しかし, 多くのDBはデータの一貫性を保つためにテーブルの分割が完了するまで新たなクエリを受け付けることができないという課題があった．各情報源から絶え間なく送信されてくる多種多様なログはそのデータ特性から，一時的にクエリ受付を停止することは非常に困難である. 無停止でDB分割を実現する技術として, NonStop SQL[5] が知られているが，専用のハードウェア上でOSと一体化したDBを利用しているため，汎用性に欠けるという課題がある．

以上をまとめると，ストリーム型ログデータ蓄積処理向け DB は次の 3 つの特性がある。

1. 処理が止められない（ストリーム処理）
2. クエリパターンは追記・参照が中心（更新は僅か）
3. レコードごとの独立性が高い（情報源単位のレコード）

これらの特性を考慮して，サービスを停止させることなく，ストリーム型ログデータ蓄積処理向け DB を動的に分割し，処理のスケールアウトを実現するための無停止 DB 分割方式を提案する。

3. 無停止 DB 分割方式

3.1 3 つの処理プロセス

提案方式では，(1)管理情報蓄積プロセスと(2)クエリ処理プロセスと(3)データコピープロセスという 3 つの処理プロセスを導入する．

(1) 管理情報蓄積プロセス
DB の分割およびクエリの振り分けに必要な管理情報を管理･蓄積する．管理情報には参照系クエリ用振り分けルールと更新系クエリ用振り分けルールと版番号が存在する. 参照系クエリ用振り分けルールは参照クエリを適用する DB の管理に使用し, 更新系クエリ用振り分けルールは挿入クエリ／更新クエリ／削除クエリを適用する DB の管理に使用し, 版番号は DB 中のデータの世代管理に使用する. 参照系クエリ用振り分けルールと更新系クエリ用振り分けルールは DB 分割処理のステージごとに振り分けルールが更新されることが特長である.

(2) クエリ処理プロセス
クエリ処理プロセスはフィルタ部で動作し, クライアントからのクエリを受け付けてクエリの種類を解析し, 解析結果に基づいて管理情報蓄積プロセスが管理する振り分けルールに従ってクエリを適用する DB を振り分け，振り分けられた DB に対してクエリを実行し，クエリの実行結果を返却する．クエリ実行によるデータ削除時（削除クエリの場合）は DB 中の削除対象データを物理的に削除せずに削除フラグを設定するのみとし, 参照クエリおよび更新クエリ実行時に削除フラグが設定されたデータをクエリ実行結果から除外することと, クエリ実行によって挿入／更新／削除された DB 中のデータに管理情報蓄積プロセスが管理する版番号を付加することが特長である.

(3) データコピープロセス
分割先の DB を新たに作成してから, 分散環境上に存在するすべてのクエリ処理プロセスに DB の分割開始を通知し, 管理情報蓄積プロセスが管理する振り分けルールと版番号に従って分割元 DB のデータを分割先 DB へコピーする. この時, クエリ処理プロセスによるクエリ実行とデータコピープロセスによるデータコピーが並行実行されることが特長である．また，データコピーが完了したら，管理情報蓄積プロセスが管理する振り分けルールを変更し, すべてのクエリ処理プロセスに DB の分割終了を通知するステップを 2 回に分けて実行するという特長がある．1 回目の分割終了通知により，分割元 DB を参照系クエリの適用先から除外し, すべてのクエリ処理プロセスが振り分けルールを変更し終わるのを待って，2 回目の分割終了通知を行う．2 回目の分割終了通知により，分割元 DB を更新系クエリの適用先からも除外し, すべてのクエリ処理プロセスが振り分けルールを変更し終わるのを待って処理を終了する．このように分割終了通知を 2 回に分けて実行することで, 終了通知の受信タイミングが異なるクエリ処理プロセスが存在する場合でも, 個々のクエリの実行結果に差異が生じないようにしている.

提案方式の概要を以下にまとめる（図 4）.

1. 更新系クエリ実行時に DB 分割の世代管理用の版番号をデータに付与する.
2. 参照系クエリと更新系クエリの適用先を振り分けルールとして管理し，DB 分割処理のステージごとに上記振り分けルールを更新する.
3. データの版番号と振り分けルールに応じてクエリ処理とデータコピーを

Vol.2010-DPS-142 No.48
Vol.2010-CSEC-48 No.48
2010/3/5

並行実行する．

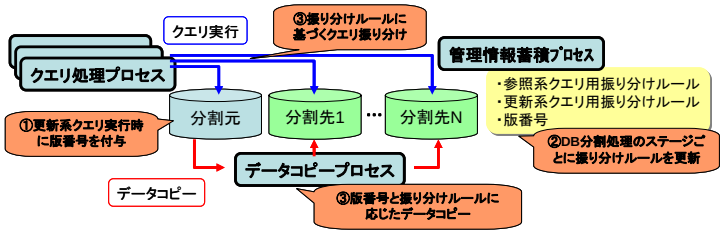

図 4　無停止 DB 分割方式のポイント

3.2 分割手順

提案方式による DB 分割手順について説明する．ここでは，全国のユーザ情報を蓄積している全国 DB（分割元 DB）を西日本 DB（分割先 DB1）と東日本 DB（分割先 DB2）に分割する例を用いて説明する．

(手順1) 初期状態として，1 つの管理情報蓄積プロセスと 1 つ以上のクエリ処理プロセスと 1 つのデータコピープロセスが存在し，DB としては分割元 DB である全国 DB のみが存在する．管理情報蓄積プロセスが管理している版番号は 1 であり，参照系クエリ用振り分けルールと更新系クエリ用振り分けルールの内容は共に等しく，すべてのクエリを全国 DB に振り分ける設定になっている．

(手順2) クエリ処理プロセスは起動時に，管理情報蓄積プロセスから最新の版番号（=1）と最新の振り分けルールを取得して自身にキャッシュとして保持し，取得した振り分けルールに従って全国 DB に対してクエリを実行する．この時，更新系クエリの場合は全国 DB 中のデータに対して，版番号=1 を付加して記録する．

(手順3) データコピープロセスは，クエリ処理プロセスがクエリを実行している裏側で，分割先 DB となる 2 つの新たな DB（西日本 DB と東日本 DB）を作成する．次に管理情報蓄積プロセスが管理する版番号を 1 から 2 に変更し，更新系クエリ用振り分けルールを，条件 A（例えば，ユーザの住所などで分類）を満たす場合は全国 DB と西日本 DB の両方にクエリを振り分け，条件 B を満たす場合は全国 DB と東日本 DB の両方にクエリを振り分けるように変更し，すべてのクエリ処理プロセスに分割開始を通知する．

(手順4) データコピープロセスから分割開始の通知を受けたクエリ処理プロセスは，管理情報蓄積プロセスから最新の版番号（=2）と最新の振り分けルールを取得して自身にキャッシュとして保持した後，データコピープロセスに分割開始通知に対する受領確認を通知し，取得した振り分けルールに従って更新系クエリの場合は全国 DB と西日本 DB，あるいは全国 DB と東日本 DB に対してクエリを実行する．この時，更新系クエリ実行時に更新対象の DB 中のデータに対して，版番号=2 を付加して記録する．

(手順5) データコピープロセスは，すべてのクエリ処理プロセスから分割開始通知に対する受領確認を受け取ると，クエリ処理プロセスがクエリを実行している裏側で，全国 DB のデータの中の版番号が 2 よりも小さいデータに対して振り分けルールを適用し，条件 A を満たすデータは西日本 DB にコピーし，条件 B を満たすデータは東日本 DB にコピーする．次に管理情報蓄積プロセスが管理する参照系クエリ用振り分けルールを，条件 A を満たす場合は西日本 DB にクエリを振り分け，条件 B を満たす場合は東日本 DB にクエリを振り分けるように変更し，すべてのクエリ処理プロセスに 1 回目の分割終了を通知する．

(手順6) データコピープロセスから 1 回目の分割終了の通知を受けたクエリ処理プロセスは，管理情報蓄積プロセスから最新の振り分けルールを取得して自身にキャッシュとして保持した後，データコピープロセスに 1 回目の分割終了通知に対する受領確認を通知し，取得した振り分けルールに従ってクエリを実行する．

(手順7) データコピープロセスは，すべてのクエリ処理プロセスから 1 回目の分割終了通知に対する受領確認を受け取ると，クエリ処理プロセスがクエリを実行している裏側で，管理情報蓄積プロセスが管理する更新系クエリ用振り分けルールを，条件 A を満たす場合は西日本 DB にクエリを振り分け，条件 B を満たす場合は東日本 DB にクエリを振り分けるように変更し，すべてのクエリ処理プロセスに 2 回目の分割終了を通知する．

(手順8) データコピープロセスから 2 回目の分割終了の通知を受けたクエリ処理プロセスは，管理情報蓄積プロセスから最新の振り分けルールを取得して自身にキャッシュとして保持した後，データコピープロセスに 2 回目の分割終了通知に対する受領確認を通知し，取得した振り分けルールに従ってクエリを実行する．

(手順9) データコピープロセスは，すべてのクエリ処理プロセスから 2 回目の分割終了通知に対する受領確認を受け取ると，参照系クエリ用振り分けルールと更新系クエリ用振り分けルールの両方から除外されたことによりクエリが振り分けられなくなった全国 DB を切り離して DB 分割処理を終了する．

万一，分割中に分割元 DB と分割先 DB のうちの片方に対するクエリ実行が失敗した場合でも，分割元 DB だけを有効として運用を継続し，分割処理を初めからやり直すことでリカバリが可能である．また，管理情報蓄積プロセスが管理する版番号と振り分けルールの設定により，分割先 DB を更に再分割すること（分割の多段化）や一

Vol.2010-DPS-142 No.48
Vol.2010-CSEC-48 No.48
2010/3/5

度の分割で N 個の DB に分割すること（N 分割）も可能である．

4. 評価

4.1 実験環境

提案方式の有効性を評価するために実験システムを構築して評価を行った．DMS の構成として追記部 1 台，フィルタ部 10 台，ビュー部 1 台のサーバ（CPU: Intel Core 2 Duo 2.53GHz x2, Memory: 2GB, OS: CentOS 5.2）を使用し，1 段目の処理と 2 段目の処理をそれぞれ割り当てた．DB 分割については，分割元 DB を分割先 DB1 と分割先 DB2 へ分割することとし，それぞれの DB に 1 台ずつ計 3 台の DB サーバ（CPU: Intel Pentium4 3.00GHz x2, Memory: 2GB, OS: Fedora Core 5）を使用した．

1 段目の DMS にログデータを連続投入し，DB 分割前後の追記部におけるデータの入力スループット，出力スループット，残レコード件数の変化を 1 段目の DMS および 2 段目の DMS について測定した．

4.2 測定 A（バースト的なデータ投入）

分割元 DB の初期データ件数を 10 万件とし，新規に 2 万件のログデータを可能な限り連続で投入し，擬似的にバースト的なデータ投入の状態を発生させた．1 段目の DMS の追記部へのデータ投入が完了し，未処理のデータ（残レコード）が存在する状態で分割元 DB の分割処理を開始した．1 段目の DMS および 2 段目の DMS の追記部におけるデータの入力スループット，出力スループット，残レコード件数の 1 秒ごとの測定結果を図 5 および図 6 に示す．

1 段目の DMS ではバースト的なデータ投入による影響により，入力スループットが初めに急上昇し，データ投入完了とともに一気に下降している．入力スループットの上昇と連動して残レコード数も一気に上昇し，データの投入が完了すると徐々に残レコード数が減少している．更に分割終了後は出力スループットが大幅に向上した影響で残レコード数が急激に減少している．出力スループットについては，分割中もスループットの低下が僅かで，分割終了後に大幅に向上している．2 段目の DMS でもほぼ同様の傾向となっているが，1 段目の DMS の出力が 2 段目の DMS の入力となるため，測定開始後のバースト的なデータ投入の影響は少ない．

また，分割元 DB の初期データ件数を 1 万件／10 万件／100 万件と変化させながら，新規に 2 万件のログデータを可能な限り連続で投入した場合に DB 分割に要する時間を測定した（表 1）．いずれの場合も 1 秒あたり約 200～250 件のデータ分割を実行していることがわかり，初期データ件数に依らず一定の処理量で分割を実行していることがわかる．

表 1　DB 分割に要する時間

DBの初期データ件数	1万	10万	100万
事実上の分割対象データ件数	3万	12万	102万
所要時間(秒)	142	562	4145
単位時間あたりの処理量(件／秒)	211.27	213.52	246.08

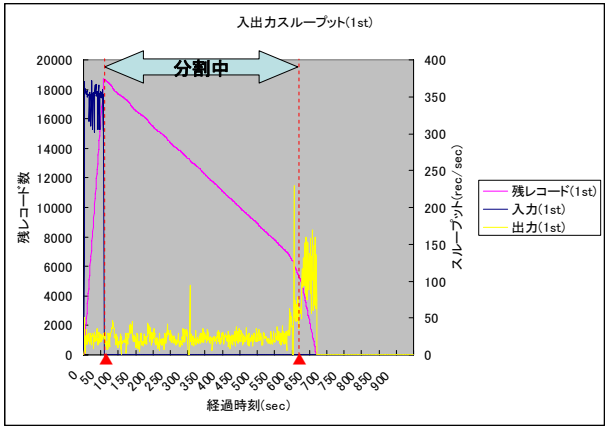

図 5　測定 A の結果（1 段目の DMS）

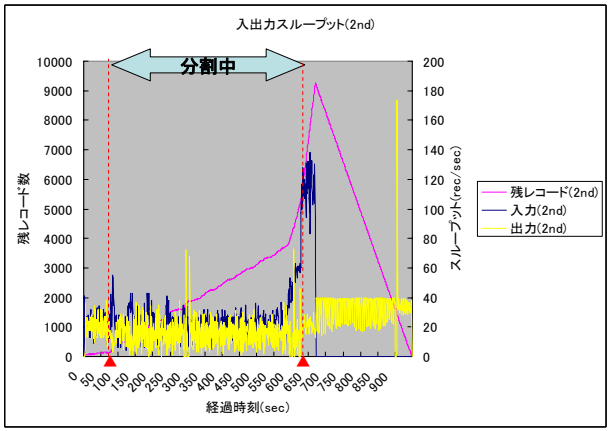

図 6　測定 A の結果（2 段目の DMS）

4.3 測定 B（定常的なデータ投入）

分割元 DB の初期データ件数を 10 万件とし，一定の入力レート（3 件／100 ミリ秒 ≒ 30 件／秒）で新規データを投入し，定常的にデータが投入されている状態を発生させて任意のタイミングで分割元 DB の分割処理を開始した．1 段目の DMS および

2 段目の DMS の追記部におけるデータの入力スループット，出力スループット，残レコード件数の 1 秒ごとの測定結果を図 7 および図 8 に示す．

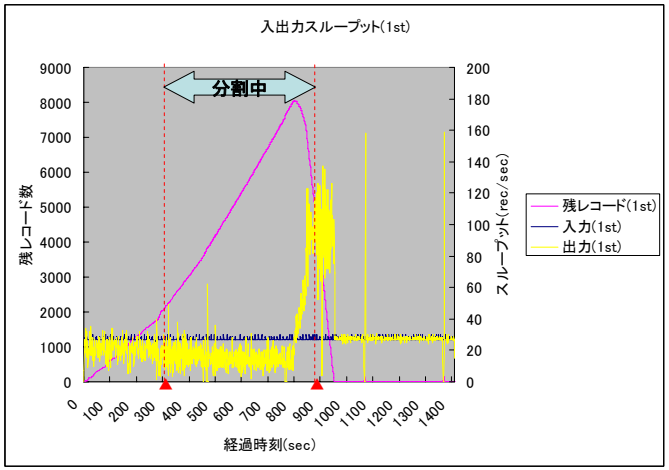

図 7　測定 B の結果（1 段目の DMS）

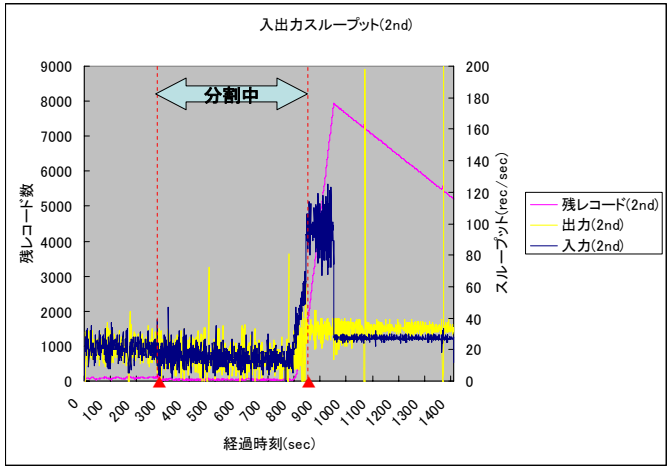

図 8　測定 B の結果（2 段目の DMS）

1 段目の DMS では経過時間が進むにつれて出力スループットが徐々に低下していき，それに伴って残レコード数が徐々に増加している．しかし，分割が終了すると DB ボトルネックが解消して出力スループットが向上し，一定の入力レートで投入され続けるデータに加えて，それまで追記部に蓄積されていた未処理のデータまで処理できるようになり，出力スループットが一気に上昇している．未処理のデータをすべて処理し終えた時点で入力スループットと出力スループットが等しくなり，安定状態に戻っている．2 段目の DMS でもほぼ同様の傾向となっているが，1 段目の DMS の出力が 2 段目の DMS の入力となるため，残レコード数の急激な増加は分割終了後に見られた．

5. おわりに

処理が止められず，クエリパターンは追記・参照が中心であり，レコードごとの独立性が高いという特性をもつストリーム型ログデータ蓄積処理向け DB を，サービスを停止させることなく動的に分割することで処理のスケールアウトを実現する無停止 DB 分割方式を提案した．また，評価結果から DB の分割中も処理のスループット低下は僅かであり，分割終了後は入力スループットと出力スループットが等しくなる安定状態へ戻り，DB ボトルネックが解消することを示した．

GPS機能付きの携帯端末から取得したライフログを活用するサービスの例として，利用者のコンテキスト（時間，現在地，プロファイル情報，操作履歴など）に応じた個人適応型のレコメンドサービス[6][7]などが考えられている．ライフログはストリーム型ログデータの一例であるため，各情報源をユーザとして捉え，情報源ごとのログをユーザごとのライフログとして捉えれば，ストリーム型ログデータ分散蓄積処理基盤を用いてこれらのサービスに対応することが可能であり，現在検証実験中である．今後は大規模データを用いた実験等において提案方式の有効性検証を行っていきたい．

参考文献

[1] 赤間，内山，三浦，西岡，内藤，谷口，山室，櫻井: 追記・参照型データ管理プラットフォームアーキテクチャの提案，情報処理学会 DPSWS 2006，pp199-204 (2006).
[2] 内山，赤間，西岡，内藤，谷口，長谷川，三浦，山室，櫻井: 分散データストリーム処理アーキテクチャの提案，情報処理学会研究報告 2007-DBS-143，DBWS 2007，pp327-332 (2007).
[3] 赤間，内山，西岡，内藤，谷口，長谷川，兵藤，三浦，山室，櫻井: 追記・参照型データ管理システムの設計と評価，情報処理学会論文誌，Vol.49，No.2，pp749-764 (2008).
[4] 西田圭介: Google を支える技術，技術評論社 (2008).
[5] 樋川，渡辺: データベースプロセッサ: NonStop SQL のアーキテクチャ，情報処理, Vol.33, No.12, pp.1436-1440 (1992).
[6] 伊藤，飯塚，村山，小林: 行動支援サービスのためのユーザ理解モデルの検討，信学技報，vol. 109，no. 272，pp.121-128 (2009).
[7] 手塚，中村，茂木，永徳，瀬古，西野，武藤，阿部: GPS 情報に基づくリアルタイムユーザ状況推定システムとフィールド実験，信学技報，vol. 109，no. 272，pp.129-133 (2009).