複数コア環境における HTAP を想定した OLTP 更新の OLAP への適用手法の検討・評価

諸岡　大輝† 　塩井　隆円† 　引田　諭之† 　Le Hieu Hanh† 　横田　治夫†

† 東京工業大学情報理工学院情報工学系　〒 152-8550 東京都目黒区大岡山 2-12-1
E-mail: †{morooka,shioi,hikida,hanhlh}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし　近年,OLTP のデータ更新を OLAP の対象データへリアルタイムに適用することで, 最新のデータを用いた OLAP が可能になる HTAP について研究が進められている. HTAP では OLTP 更新を適用する時間が短いほど, より最新のデータを OLAP の対象データにすることが可能なため,OLTP 更新の適用時間短縮が求められる. 本研究では OLTP 更新を適用する方法として Log shipping を採用し,Log shipping を実行するコアを割り当てる手法を用いることで複数コア環境下における HTAP を想定した OLTP 更新の OLAP への適用について検討・評価を行う.

キーワード　複数コア,OLTP,OLAP,Log shipping

1 はじめに

1.1 研究背景

データベース処理は大きく分けてトランザクション処理を表す OLTP とデータ分析処理を表す OLAP の 2 種類に分類される.OLTP は更新処理に伴う write 処理が多いが,OLAP は read 処理を多く実行する. この 2 つの処理はロック取得などでパフォーマンスが低下することを防ぐために, 対象データセットを分離し,OLTP のデータはリアルタイムで更新するが,OLAP のデータは定期バッチ処理を用いて更新することが多い. しかし, 近年では OLTP のデータ更新を OLAP の対象データにリアルタイムに適用する Hybrid Transactional Analytical Processing(HTAP) が注目を集めている [1, 2].HTAP ではより最新のデータ, すなわち Freshness が高いデータを用いた OLAP が可能となる. ビックデータ・IoT・AI の伸長によりデータ分析の需要が高まり,Freshness の高いデータを対象とした OLAP が求められるようになったため,HTAP の需要も高まっている. HTAP では OLTP の更新を定期バッチ処理よりも高い頻度で,OLAP 対象データへ適用する必要がある. したがって, より最新のデータを対象に OLAP を実行するには,OLTP 更新の OLAP への適用時間を短縮する必要がある.

同様に近年の傾向として複数コア CPU の多用化が挙げられる. クロック周波数を上昇させることで CPU の処理性能を向上させる手法は限界に達し, 高い処理性能が求められる分野では, 各々のコアのクロック周波数は低いが, 処理を複数コアで分散させることで全体として高性能処理を実現する複数コア CPU を用いる傾向にある [3]. したがって処理を分散させることで複数コアを有効活用し, 処理性能を向上させるアプリケーションの開発が必要不可欠である. しかしながら, 複数コアを用いて処理を分散させながら,OLTP 更新を OLAP 対象データへ適用する手法については依然として研究が十分に行われていない.

そこで本研究は, 複数コア環境における OLTP 更新の OLAP 対象データへの適用時間短縮を目的とする. 更新適用手法として Log shipping を用いた手法を採用した上で, 処理内容に応じて実行するコアの割り当てを行い, 処理を分散させることで複数コア環境を有効に活用した更新適用手法の提案・評価を行う.

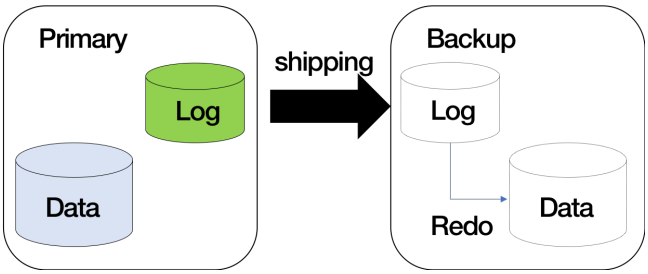

図 1　Log shipping

1.2 本稿の構成

本稿は以下の通りに構成される. 2 節では背景となる知識とそれに関連する研究について述べる.3 節では本研究における OLTP 更新の OLAP 対象データへの適用手法を提案する.4 節では実験環境や実験内容と実験結果について述べる. 最後に 5 節で本稿のまとめと今後の課題を述べる.

2 背景知識

本節では, 本研究を理解する際に前提とする知識とそれに関連する研究について説明を行う.

2.1 Log shipping

Log shipping とはデータベースサーバのバックアップ手法の一つである. 図 1 が Log shipping を表した図である. まずバックアップサーバ上に, ある時点のプライマリーサーバのデータを保有する. そこで, プライマリーサーバで実行されたトランザクションの実行ログをバックアップサーバに送信する, 送信されたトランザクションログをバックアップサーバー上のデー

タに対して Redo することでデータの復旧が可能になる手法が Log shipping である.

Log shipping の特徴として 2 点あげられる.1 点目はプライマリーサーバを停止させずにバックアップを取得可能なため,Log shipping はホットバックアップである.2 点目は Commit 済みトランザクションのログのみを送信するため,OLAP のトランザクション分離レベルを Read committed にできる点である. 本研究ではこの 2 つの特徴を活用するため, 更新適用手法に Log shipping を採用する.

2.2 HTAP

1 節で述べたように,HTAP とは OLTP の更新をリアルタイムに OLAP 対象データに適用し,Freshness の高いデータを用いた OLAP を可能にすることである. HTAP を実現しているデータベースとして SAP HANA [4] が挙げられる. SAP HANA では OLTP 用データセットをデルタストレージに保存,OLAP 用データセットをメインストレージに保存することで OLTP と OLAP の対象データを分離させ, 双方の高効率化を実現している. また, デルタストレージの情報は定期的にメインストレージにマージされるため, 定期バッチ処理よりも高い頻度で OLAP 対象データが更新され, より最新のデータを対象にした OLAP が実行可能である.

しかしながら,SAP HANA では OLAP 対象データを更新するマージ処理によって, データベース処理の性能が一時的に低下してしまうことが報告されている [5]. また, 高負荷時にはマージ処理がデータベース処理とリソースを競合し, 更新適用処理に時間がかかってしまうため,Freshness の高い OLAP 実行が困難であるという課題点が挙げられる.

2.3 複数コア DB システム

1 節で述べたように, 近年の CPU は搭載するコア数の増加が著しく, 複数コア環境に対応したデータベースシステムが必要である. そこで, 関連研究 [6, 7] では複数コア環境に対応したデータベースシステムの提案を行った.

図 2 が関連研究 [7] の概要図である. この研究ではトランザクションのアクセス範囲に応じてトランザクションを処理するコアを変化させるコアアサイン手法を用いることで, 各コアの L1・L2 キャッシュヒット率が向上することを示した.

しかしながら, この研究では OLAP の対象データが更新されないため, 最新データを用いた OLAP が不可能であり HTAP を実現できていないという課題点が挙げられる.

本研究ではこれらの課題点を解決するために, 複数コア環境で OLTP の更新を OLAP 対象データに適用する時間を短縮することを目的とする.

3 提案手法

前節を踏まえて, 本研究では処理内容に応じたコアアサインを行った上で, 更新適用手法として Log shipping を用いる手法を提案する.

本節ではまず, 提案手法におけるコアアサインについて述べ,

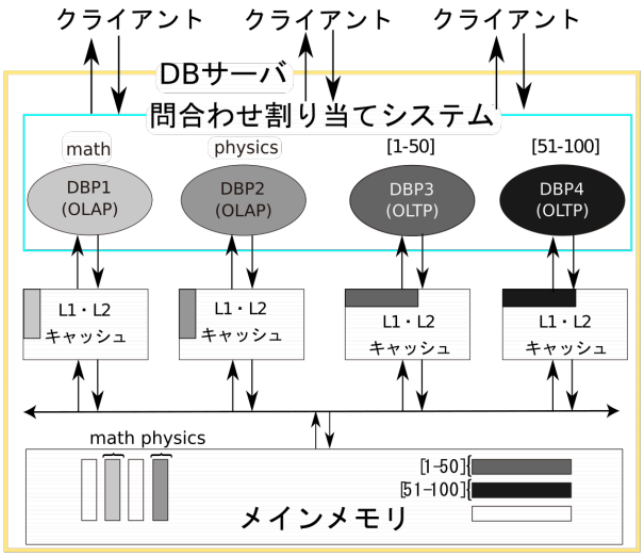

図 2　アクセス範囲に応じたコアアサイン

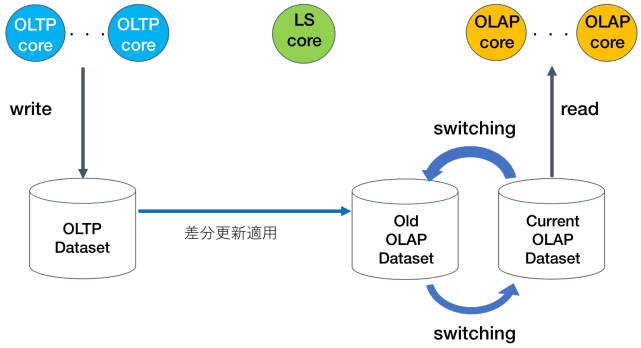

図 3　コアアサイン

次に更新適用手法について説明を行う.

3.1 コアアサイン

2 節で述べた関連研究 [7] 同様に複数コア環境を有効に活用するため, 各コアに対して OLTP 処理・OLAP 処理・更新適用処理の 3 種類のいずれかを実行するようにコアアサインを行う. 図 3 のように OLTP データと OLAP データを分離させ,OLTP 処理コアは OLTP データにのみアクセスを行い,OLAP 処理コアは OLAP データにのみアクセスを行う.

更新適用処理を行うコア (以下,LS コア) は OLTP・OLAP の実行とは独立して処理を実行し,OLTP 実行による差分を OLAP データに適用する処理を行う.

また, 更新適用処理を実行している間も OLAP を実行し続けるため,OLAP データを 2 つ保有し, 更新が適用される OLAP データと OLAP 実行の対象になる OLAP データを交互に切り替える構造を採用する.

このように LS コアを設定することによって, 高負荷時にも更新適用時間や OLTP・OLAP 処理性能に与える影響を極力小さくした更新適用が可能であると考えられる.

3.2 更新適用手法

提案手法における更新適用とは OLTP のデータセットを V(n)

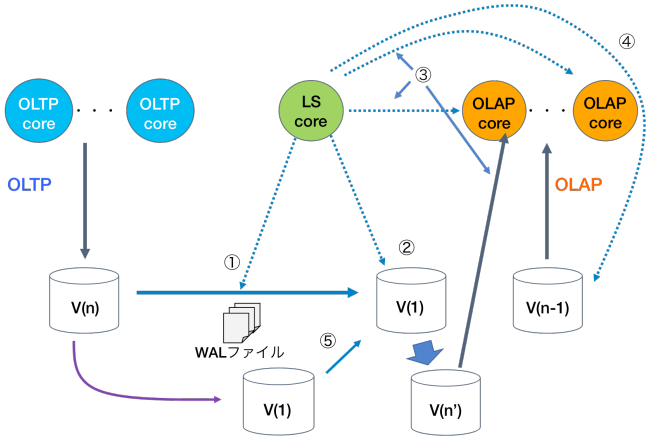

図 4　Base Version Relation

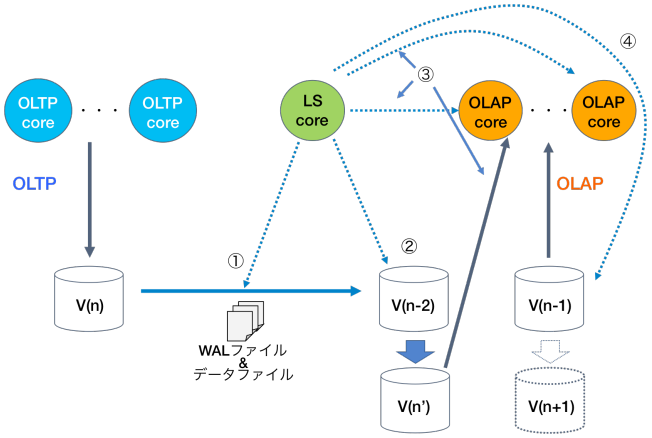

図 5　Latest Two-Version Relations

とした際に,OLAP 対象データを V(n-1) から V(n') に変更することであると定義する.

本研究では更新適用手法として, 最初のバージョン V(1) から差分更新を行う Base Version Relation(以下,BVR) と最も新しい 2 つのバージョンから差分更新を行う Latest Two-Version Relations(以下,LTVR) という 2 種類の手法を用いる. 本小節では, この 2 種類の手法について説明を行う.

3.2.1　更新適用手法 1：Base Version Relation

BVR では最初に OLTP データから V(1) を表す Base Version を取得し, 常に V(1) に対して差分更新を行う.

図 4 は BVR によって OLAP データセットを更新する処理を表している.BVR を実行する直前まで OLAP は V(n-1) を対象データセットとして実行する.BVR を実行すると V(1) に対して Write Ahead Log(以下,WAL) ファイルを適用して差分更新を行い, データセットを V(n') を構築する. その後,OLAP の参照するデータセットを V(n-1) から V(n') に変更することで更新適用を行う.

BVR における LS コアは以下の 5 つの処理を繰り返し実行する.

① V(1) に対する差分 WAL ファイルをコピー
② 差分ファイルを用いて V(1) を更新し,V(n') を構築
③ OLAP コアの接続 DB を V(n-1) から V(n') に変更
④ V(n-1) の DB サーバを停止
⑤ V(n-1) を V(1) に書き換える

この手法では V(1) から V(n) までの全てのバージョンデータを構築可能なため, 任意バージョンに対しての OLAP 実行が可能になる.

本手法の評価実験で使用する PostgreSQL では V(n) の DB が起動していると同一 DB の起動を防ぐ機能が備わっており,V(n') を起動するには V(n) を停止する必要があった. したがって,V(n) の起動前に V(1) である Base Version を取得することで,V(n') と V(n) を別々の DB として起動可能な BVR を提案した.

しかし DB 動作時に生成される postmaster.pid ファイルを削除することで V(n) を停止せずに V(n') を起動することが可能になったため, 更新適用手法 2 の LTVR を用いた更新適用を考案した.

3.2.2　更新適用手法 2：Latest Two-Version Relations

LTVR の主な処理の流れは BVR と同様だが,LTVR では V(1) ではなく V(n-2) に対して差分更新を行う.

図 5 が LTVR によって OLAP データを更新する処理を表している.BVR では WAL ファイルのみを用いて更新適用を行ったが,LTVR では WAL ファイルとデータファイルを用いて更新適用を行う. LTVR では CHECKPOINT 済みの更新に対してはデータファイルをコピーすることで更新し,CHECKPOINT されていない更新は WAL ファイルを適用することで更新適用を行う. このように差分データファイルと差分 WAL ファイルを用いて V(n-2) を更新して V(n') を構築し,OLAP の参照データセットに変更する手法が LTVR である.

LTVR における LS コアは以下の 4 つの処理を繰り返し実行する.

① V(n-2) に対する差分 WAL ファイルと差分データファイルをコピー
② 差分ファイルを用いて V(n-2) を更新し,V(n') を構築
③ OLAP コアの接続 DB を V(n-1) から V(n') に変更
④ V(n-1) の DB サーバを停止

この手法では V(n-2) に対して更新適用が実行されるため, 必要となる差分ファイル数を削減でき, より短時間での更新適用が可能と考えられる.

また, どちらの手法においても WAL ファイルは Commit 済みのトランザクションのみを適用するため, 未 Commit のトランザクションは OLAP 対象データに更新されない. 多くの DBMS は Commit 済みトランザクションの更新を参照可能とする Read committed をデフォルトのトランザクション分離レベルとしていることから, 本手法は OLAP 対象データに対して最低限の Consistency が確保できていると考えられる.

4　実　　験

提案手法の性能を調べるため,3 節の提案手法を実装し, 評価実験を行った.

表 1　実験環境

	クライアント	サーバ
プロセッサ	Intel Xeon E7-4860	AMD Opteron 6174
ソケット数	4	4
論理コア数	80	48
クロック周波数	2.27GHz	2.20GHz
L1 キャッシュ	32KB+32KB	64KB+64KB
L2 キャッシュ	256KB	512KB
L3 キャッシュ	24576KB	5118KB
RAM	32GB	32GB
DISK	SSD(200GB)	SSD(100GB) × 2
OS	Ubuntu 16.04.3	Ubuntu 16.04.3
PostgreSQL	11.5	11.5

4.1　実験環境

表 1 に本実験で用いた実験環境を示す.

OLTP と OLAP を実行する DBMS として PostgreSQL を使用し,WAL ファイルを用いた更新適用は PostgreSQL の Point In Time Recovery(PITR) 機能を用いて行った.

また, 本研究は Linux カーネル 2.6 以降の環境で使用可能な「sched_setaffnity()」API を用いて CPU affinity を設定することで処理内容に応じたコアアサインを適用した.

4.2　実験方法

HTAP 向けベンチマークである CH-benCHmark [8] のデータを SF=10 で生成し, 評価実験を行った.

CH-benCHmark は OLTP 処理性能を測る TPC-C [9] と OLAP 処理性能を測る TPC-H [10] を混合したベンチマークであり,TPC-C で更新されたデータを対象として TPC-H を実行することが可能である. このベンチマークを用いて, 更新適用時間と OLTP・OLAP 処理性能について計測を行う. なお,CH-benCHmark のデータ生成やクエリは oltpbench [11] に基づいて行った.

また, アサインするコア数に関しては論理コア数 48 のサーバマシンに対して OLTP:OLAP:LS コアを 24:23:1 というコア配分を静的に適用した.

4.3　実験内容・結果

4.3.1　実験 1：更新適用時間

3.2 節で述べた 2 種類の更新適用手法を用いた場合の適用時間を把握するため, トランザクションを一定数実行した後に更新適用を実行し, 更新適用時間の比較を行う.

トランザクションは CH-benCHmark に準拠し,TPC-C の New-Order,Payment,Order-Status,Delivery,Stock-Level という 5 種類のトランザクションを 10:10:1:1:1 の比率で実行を行う.

実験 1 の結果が図 6 である. 実験 1 では LTVR は 0 トランザクション時には約 30 秒,10000 トランザクション以上実行した際には約 15 秒早く更新適用が完了し, 全てのケースにおいて BVR より高速な結果になった. これは BVR が Base Version から全てのデータをコピーするのに対し,LTVR では前バージョンからの差分のみをデータコピーしているため扱うデータ量が

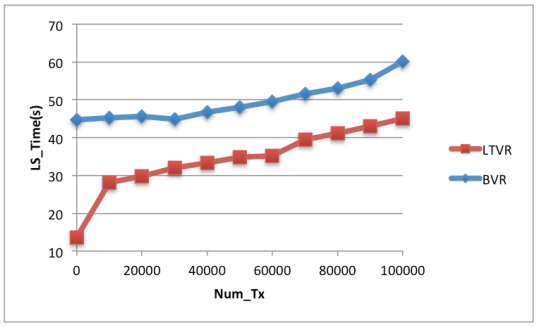

図 6　実験 1：更新適用時間

表 2　性能評価対象

	OLTP・OLAP データ	コアアサイン	Log shipping
Baseline	統一	なし	なし
LTVR	分離	あり	あり

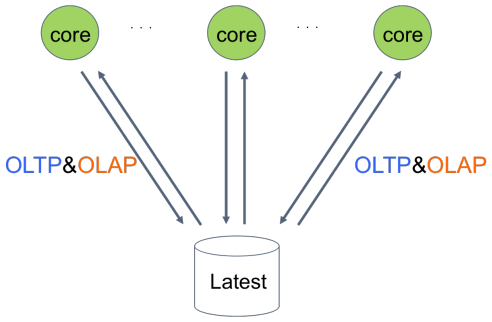

図 7　Baseline

BVR よりも小さいためであると考えられる.

また,LTVR の更新適用時間は適用する OLTP 更新が存在しない 0 トランザクション時を除いて, 更新適用のデータ量に対してほぼ線形に増加することが確認できた.

4.3.2　実験 2：OLTP・OLAP 処理性能

コアアサインを実行しながら更新適用を実行した際の OLTP・OLAP 処理性能を評価するため, 実験 1 において更新適用時間が優れていた LTVR を用いて実験 2 を行う.

実験 2 では CH-benCHmark における OLTP と OLAP を実行するクライアント数を変化させることで負荷を変化させながら処理性能を計測する.OLTP を実行するクライアントと OLAP を実行するクライアントの数は等しいものとし,OLTP のクライアント数が 12 の際には,OLAP クライアント数も 12 になり, サーバに対して合計 24 クライアントが同時接続している状況とする. OLTP 処理性能は 1 分間に処理可能なトランザクション数を表す tpmC,OLAP 処理性能は 1 時間で処理可能なクエリ数を表す QphH を用いて評価する.

LTVR はクエリのアクセス範囲に応じて実行するコアを割り当てるため,OLTP では主キーによってパーティショニングを行い, それに基づいて実行コアを指定した.OLAP では OLAP 実行コアとして設定したいずれかのコアが OLAP クエリを実行するように指定した.

LTVR と比較する Baseline としては, 図 7 のようにコアアサインと Log shipping を行わずに, 単一 DB に対して OLTP と

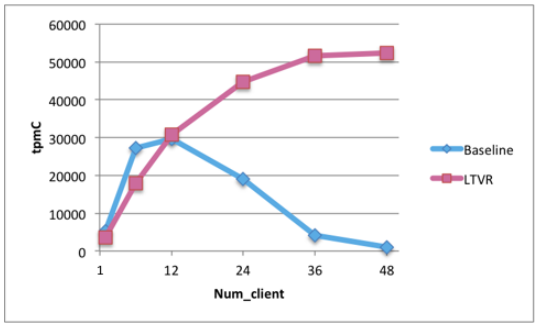

図 8　実験 2：OLTP 処理性能

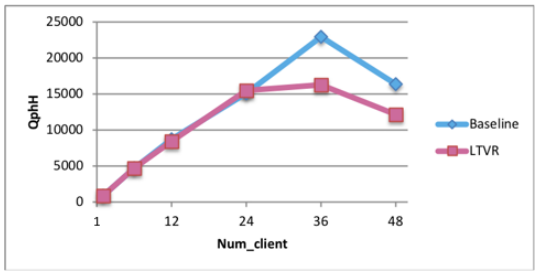

図 9　実験 2：OLAP 処理性能

OLAP を実行する構成を採用した. 表 2 は LTVR と Baseline の違いをまとめたものである.

実験 2 の結果が図 8〜図 9 である. 図 8 は OLTP 処理性能を表し, 図 9 が OLAP 処理性能を表している.

OLTP 処理性能に関しては,Baseline では 24 クライアント以上になると大幅な tpmC 低下が確認された. 一方で LTVR では, クライアント数が増加した際にも tpmC は増加し続けていた.48 クライアント時の tpmC を定量的に比較すると,Baseline は約 1000tpmC に対して LTVR は 52000tpmC になり,LTVR は tpmC が約 51000 向上し,4991.2%処理性能が向上する結果であった. これは Baseline では同時実行した OLAP によって大量の read ロックや Disk I/O が発生し,OLTP の write ロック取得に必要な待ち時間や WAL の書き込みに必要な時間が増加したことが原因で,tpmC が大幅に低下したと考えられる. 一方,LTVR は OLTP と OLAP の対象データが分離し, アクセスする Disk が異なるため,OLAP による read ロックや Disk I/O の影響を受けずに OLTP を実行できるため, クライアント数の増加に対して tpmC の低下が発生しなかったと考えられる.

OLAP 処理性能に関しては Baseline と LTVR はどちらも 36 クライアントまでは QphH が向上しているが,48 クライアント時には QphH が低下している. これは実験に使用したサーバの論理コア数が 48 コアに対し,OLTP と OLAP 合計で 96 クライアントが接続している状況になり,CPU リソースが不足してしまっているためと考えられる.

また,QphH について Baseline と LTVR を比較すると,LTVR は 24 クライアントまでは同等の QphH だがクライアント数が 36 以上の際には Baseline より低い QphH であった.QphH の差が最大であった 36 クライアント時の QphH を定量的に比較すると,Baseline は約 23000QphH に対して LTVR は約 16000QphH になり,LTVR は QphH が約 7000 低下し,29.2%の処理性能が低下する結果になった. これは LTVR では Log shipping が実行され,OLAP データに対して大量の Disk I/O が生じていることや OLAP 処理にアサインするコア数が Baseline と比べて制限されていることが原因であると考えられる.

4.3.3　ま と め

実験 1 では提案手法である BVR と LTVR の 2 つについて更新適用時間の比較を行った. 実験 1 の結果,LTVR が更新適用時間では優れ, 今回の実験では 30〜50 秒程度で OLAP 対象データを更新できることを確認した.

実験 2 では LTVR をコアアサインしながら実行した際の OLTP と OLAP の処理性能を計測した. 実験 2 の結果,OLTP 処理性能はクライアント数の増加に対しても増加し続け,4991.2%性能向上が確認できた.OLAP 処理性能は LTVR による影響を受け,29.2%性能が低下する結果になった.

今回の実験では Baseline としてコアアサインと Log shipping を実行しない手法と比較を行ったが, コアアサインを行わずに Log shipping は実行する手法と比較することで, コアアサインの効果をより詳細に検証できると考えられる.

5　おわりに

本研究では複数コア環境において HTAP を想定した際に必要となる,OLTP の更新を OLAP 対象データに適用する時間を短縮し,OLAP 対象データの Freshness を向上させることを目的に研究を行った.

アプローチとして Log shipping を用いた更新適用手法をコアアサインしながら実行することで, 各コアに対して処理の分散を実現する手法を示した.

提案手法について実験を行った結果,OLTP の更新をリアルタイムに OLAP 対象データに適用可能であることを確認した. また, 更新適用手法をコアアサインしながら実行することでベースラインと比較して OLTP 処理性能の向上が確認できたが,OLAP 処理性能は低下する結果になった.OLAP 処理性能が低下した原因として SSD の I/O 性能か CPU の処理性能のどちらかが限界に達していることが考えられるため,Disk I/O や CPU 利用率を確認し原因を追求する必要がある.

今後の課題としては提案手法の効果をより詳細に検証するための追加実験を行うことがあげられる. 具体的には, 今回の実験では OLTP と OLAP のクライアント数を同一に設定して実験を行ったが,OLTP のクライアント数を OLAP のクライアント数よりも増やして実験し,Write Heavy なワークロードを与えた際の性能を評価することがあげられる. その他にも, コアアサインを行わずに LTVR を実行する手法と比較実験することでコアアサインの効果をより詳細に評価することが可能である.

同様に, 処理性能の更なる向上も今後の課題である. 具体的なアプローチとして, 今回の実験ではコアアサインを静的に設定して行ったが, 動的なコアアサインが可能なように機能拡張することがあげられる. また,OLAP を実行する DBMS を列指向 DBMS に変更し, 更新適用後のデータ構造を列指向へと変換可能にすることで OLAP 処理性能の更なる向上が期待できる.

謝　　辞

この成果は, 国立研究開発法人新エネルギー・産業技術総合開発機構（ＮＥＤＯ）の委託業務の結果得られたものである.

文　　献

[1] Samuel S Conn. OLTP and OLAP data integration: a review of feasible implementation methods and architectures for real time data analysis. In *Proceeding of the 2005 SoutheastCon*, pp. 515–520. IEEE, 2005.

[2] Fatma Özcan, Yuanyuan Tian, and Pinar Tözün. Hybrid transactional/analytical processing: A survey. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1771–1775. ACM, 2017.

[3] David Geer. Chip makers turn to multicore processors. *Computer*, Vol. 38, No. 5, pp. 11–13, 2005.

[4] Vishal Sikka, Franz Färber, Wolfgang Lehner, Sang Kyun Cha, Thomas Peh, and Christof Bornhövd. Efficient transaction processing in SAP HANA database: the end of a column store myth. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 731–742. ACM, 2012.

[5] SAP HANAPower Systems 出版チーム. SAP HANA 入門 Powered by IBM Power Systems. 翔泳社, 2017.

[6] Fang Xi, Takeshi Mishima, and Haruo Yokota. CARIC-DA: Core affinity with a range index for cache-conscious data access in a multicore environment. In *Proceedings of the International Conference on Database Systems for Advanced Applications*, pp. 282–296. Springer, 2014.

[7] 諸岡大輝, 塩井隆円, Le Hieu Hanh, 横田治夫. 複数コア環境におけるアクセス範囲を考慮した OLTP/OLAP 同時実行手法. 第 11 回データ工学と情報マネジメントに関するフォーラム, 2019.

[8] Richard Cole, Florian Funke, Leo Giakoumakis, Wey Guy, Alfons Kemper, Stefan Krompass, Harumi Kuno, Raghunath Nambiar, Thomas Neumann, Meikel Poess, et al. The mixed workload CH-benCHmark. In *Proceedings of the Fourth International Workshop on Testing Database Systems*, p. 8. ACM, 2011.

[9] TPC. TPC-C Benchmark 5.11.0. `http://www.tpc.org/tpcc/`.

[10] TPC. TPC-H Benchmark 2.17.3. `http://www.tpc.org/tpch/`.

[11] Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, and Philippe Cudre-Mauroux. Oltp-bench: An extensible testbed for benchmarking relational databases. *Proceedings of the VLDB Endowment*, Vol. 7, No. 4, pp. 277–288, 2013.