DEIM Forum 2021 B31-1

OLTP 性能を維持し高データ鮮度を実現する HTAP の 複数コア環境での動的負荷分散

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　トランザクション処理を行う OLTP での更新を, 高いデータ鮮度でデータ分析処理を行う OLAP に適用可能な HTAP 技術が注目されている. 更新適用の際にワークロードが変化しても,OLTP の性能が劣化することは可能な限り避けなければならない. 複数コア環境で OLTP 性能を維持しながら, 高いデータ鮮度で HTAP を実現するためにはリソースを有効に活用する必要がある. そこで本研究ではリアルタイムに変化するワークロードに応じて OLTP 性能の維持とデータ鮮度を優先して, 複数コアのリソースを動的に割り当てる負荷分散手法を提案する. また, 実際の複数コア環境においてベンチマークを用いた評価を行うことで提案手法の効果を検証する.

キーワード　HTAP, 複数コア, データ鮮度, 負荷分散, リソース管理

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 (OnLine Transaction Processing) とデータ分析処理を表す OLAP (OnLine Analytical Processing) の 2 種類に分類される.OLTP は更新処理に伴う write 処理が多く短時間の排他制御が必要だが,OLAP では長時間での read 処理を多く実行する. 従来ではこの 2 つの処理はロック取得などによって双方のパフォーマンスが低下することを防ぐため, 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アーキテクチャが多く採用されていた.

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可能な HTAP (Hybrid Transactional Analytical Processing) が注目を集めている [1,2]. HTAP を用いることでより最新のデータ, すなわちデータ鮮度 (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ことから HTAP の必要性が高まっている. HTAP では OLTP と OLAP を組み合わせた多様なワークロードに対して,OLTP・OLAP 処理性能と,OLTP による更新を OLAP データに適用する更新適用性能を得る必要がある. この 2 つを得るためには計算リソースを適切に配分し, 処理を分散させる必要がある.

また近年では Intel Optane DC Persistent Memory などといった, より高速にアクセス可能な NVDIMM の開発や SSD の低廉化によって, 使用されるストレージの高速化が活発である. 高速にアクセス可能なストレージを使用した環境においては,Disk I/O ではなく CPU ボトルネックが発生する傾向が強く, 処理性能を向上させるには CPU ボトルネックの解消が必要である. しかしながら近年の CPU は, 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 CPU ボトルネックを解消し高性能化を実現するには, 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環境での HTAP において, 動的に各コアへ処理を割り当てることで処理を分散させ,OLTP・OLAP の高い処理性能と高い更新適用性能を実現する研究は十分には行われていない.

そこで本研究では,HTAP における性能を OLTP 処理性能・更新適用性能・OLAP 処理性能の順に優先して考え, 複数コア環境での HTAP 実行において OLTP 処理性能と OLAP データの Freshness を維持しながら, 可能な限りの OLAP 処理性能を取得することを目的とする. この目的を実現するため, 本研究では処理内容に応じて実行するコアを割り当てるコアアサイン手法を用いる.HTAP に必要な OLTP 処理・更新適用処理・OLAP 処理に加えてコアアサイン処理の 4 つの処理専用のコアを設定し, 各コアに対して処理のアサインを行う. また,HTAP 向けのロードバランス手法を提案することで, リアルタイムに変化する多様なワークロード各々に適したコアアサインの動的な取得を可能とし, 高い処理性能と更新適用性能を実現する手法を提案し, その手法の評価を行う.

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コアに対する HTAP の処理の割り当てと負荷分散手法を提案する.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.

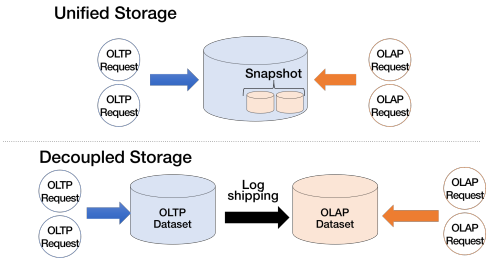

図 1　HTAP アーキテクチャ

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　HTAP (Hybrid Transactional Analytical Processing)

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 対象データに適用することで, 高速な OLTP と Freshness の高いデータを用いた OLAP の両立を可能にすることである.HTAP は OLTP と OLAP 両方を実行するため,2 種類の処理が混合した多様なワークロードに対応し,OLTP 処理性能と OLAP データの Freshness を維持することが求められている.

また HTAP のアーキテクチャは大きく分けて 2 種類に分類される [4]. 両者の違いを表したものが図 1 である. 一つ目は同一のデータを対象にスナップショット機能などを用いて OLTP と OLAP を実行する Unified Storage である.Unified Storage では OLAP 実行のたびにスナップショットなどによって OLAP クエリ実行時点のデータを取得するため,Freshness が最新のデータを対象に OLAP が可能であるが,OLTP 性能へのオーバーヘッドが大きいという特徴を持つ. もう一つのアーキテクチャは Decoupled Storage である.Decoupled Storage は OLTP と OLAP は別々のデータを対象に実行され,OLTP による更新を随時 OLAP データに反映するアーキテクチャである.Decoupled Storage は Unified Storage よりも OLAP の Freshness が劣るが, 高い OLTP 性能が実現できるという特徴を持つ.

また Decoupled Storage における OLAP データを更新する手法の一つとして Log shipping があげられる. OLAP データは常にある時点の OLTP データが保存されている. そこで OLTP データに対して実行されたトランザクションの実行ログを OLAP データに送信し, 送信されたトランザクションログを OLAP データに対して Redo することで OLTP データと同等な最新のデータを取得する手法が Log shipping である. したがって,Log shipping を採用する Decoupled Storage では Unified Storage と比べてトランザクションログを Redo する処理が必要なためマシン上で実行する処理の種類が増え, よりリソース配分の重要性が高いと言える.

OLTP ではレスポンスタイムが QoS のユーザ設定要件になるため,HTAP においても OLTP 処理性能を高くすることは最も優先すべき要件である. したがって, 本研究では HTAP において OLTP 処理性能を最も優先するために, アーキテクチャとして Decoupled Storage, 更新適用手法として Log shipping を採用する.

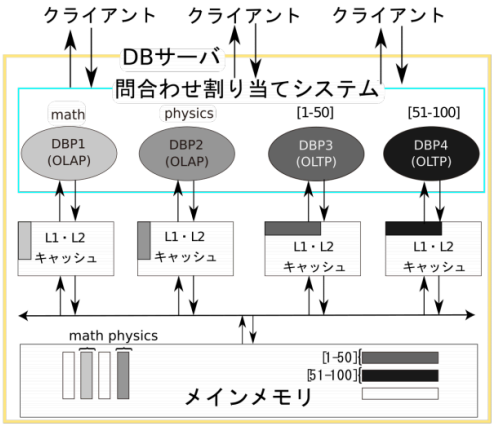

図 2　アクセス範囲に応じたコアアサイン

2.2　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. そこで本研究の先行研究 [5,6] 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 が先行研究 [6] の概要図である. この研究では OLTP と OLAP の同時実行において OLTP 処理コアと OLAP 処理コアを別々に設け,OLTP 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.

また複数コア環境に対応した HTAP に関する研究として関連研究 [4] が挙げられる. この研究では複数コア環境上での HTAP において,OLAP データの鮮度を表す Freshness Rate [7] に応じた状態遷移を行い,Unified Storage と Decoupled Storage を切り替えることでワークロードに応じたリソース配分が可能な HTAP システムを提案している.

2.3　既存研究の課題点

先行研究 [6] の課題点として,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点が挙げられる.HTAP を実現するためには,OLTP による更新を OLAP データに適用する処理も含めてコアアサインを考える必要がある. また, コアアサインが静的なため初期状態からコア配分を変更できないため, ワークロードに応じたコアアサインが実現できない点も課題点である.

関連研究 [4] の課題点として, 各処理に割り当てるコア数の配分が状態ごとに静的であるため, ワークロードによっては処理内容の割り当てに改善の余地が残されている点が挙げられる. 具体的には,OLAP ヘビーなワークロードにおいて,Freshness を優先する Unified Storage を使用するアーキテクチャに切り替えることは可能だが,OLTP 実行に割り当てるコアの数は静的

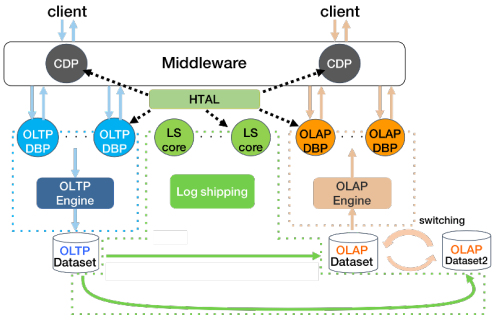

図 3　全体アーキテクチャ

なため,OLAP 実行に割り当てるコアが不足してしまう可能性が考えられる. また OLTP と OLAP の処理性能を維持するためのリソース配分は行われているが, 更新適用に関してはアーキテクチャを切り替えるのみで, 更新適用性能に対するリソース配分が行われていないという点も課題点の一つである.

本研究ではこれらの課題点を解決するため,HTAP の多様なワークロードに対して複数コア環境を想定し, 動的にコアの割り当てを変更することでリソース配分を行い,OLTP 処理性能と高 Freshness を維持しながら, 可能な限りの OLAP 処理性能を得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い, さらにコアアサインを動的に調節するロードバランサである HTAL (Hybrid Transactional Analytical processing Load balancer) を導入した手法を提案する.

本節ではまず全体のアーキテクチャについての説明を行い, 次に HTAL を適用するコアアサイン手法について述べ, 最後に HTAL のアルゴリズムや処理についての説明を行う.

3.1　全　体　像

図 3 が HTAL を用いたシステムの全体アーキテクチャを表している. クライアントはミドルウェア経由でクエリを送信し, ミドルウェアがクエリに応じて指定された実行コアに送信する.

このアーキテクチャでは Decoupled Storage を採用するため,OLTP データと OLAP データは分離し,OLTP による更新は Log shipping によって OLAP データに反映される. また Log shipping による OLAP データ更新が原因で実行結果に不整合が生じることを防ぐため,OLAP データを 2 つ保有し, 片方のデータに対しては OLAP を実行するが, もう一つのデータに対しては Log shipping による更新適用を行う. このように OLAP データに関しては更新が適用されるデータと OLAP 実行の対象になるデータを定期的に切り替えるアーキテクチャを採用する.

3.2　コアアサイン

2 節で述べた先行研究 [6] 同様に複数コア環境を有効に活用するため, 各コアに対して処理内容に応じたコアアサインを行う. 本手法では OLTP 処理・OLAP 処理・更新適用処理・コアアサイン処理の 4 種類のいずれかを実行するようにコアアサインの設定を行う.

Algorithm 1 状態遷移

Data: W_T : OLTP 実行キューの待機数
Data: F : Freshness Rate
StateDecision
　$F \Leftarrow 1 - \frac{\text{更新が未適用なタプル数}}{\text{総タプル数}}$
　if $(W_T \geqq \alpha)$ **then**
　　SetState(S_1) // OLTP mode
　else
　　if $(F \leqq \beta)$ **then**
　　SetState(S_2) // LS mode
　　else
　　SetState(S_3) // OLAP mode

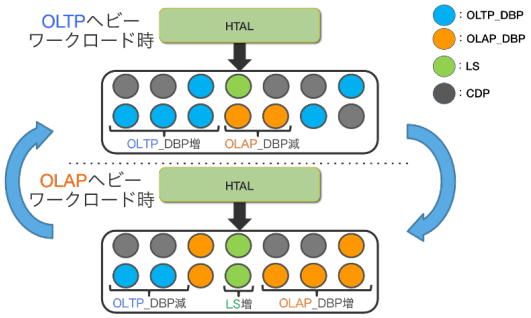

図 4　HTAL 動作例

OLTP 処理コア (以下,OLTP_DBP) と OLAP 処理コア (以下,OLAP_DBP) はそれぞれ OLTP データと OLAP データに対してのみアクセスし, 各クエリの実行を行う. 先行研究 [6] 同様に,OLTP データは主キーで Range パーティショニングされており, クエリがアクセスする主キーに応じて専用の OLTP_DBP に実行を割り当てることで, クエリ実行のキャッシュヒット率を高める. 一方で OLAP データは OLAP 専用コアのいずれかで実行されるように設定されている.

更新適用処理を行うコア (以下,LS) は Log shipping を実行するコアを表す.Log shipping では Log の送受信や更新適用処理が並列に行われているため,LS を増やすことで Log shipping の更新適用性能を高めることが可能である.

コアアサイン処理を行うコア（以下,CDP）はミドルウェアを動作させるためのコアを表す. CDP は 2 種類の役割を担う.1 つ目の役割はクライアントや DBP との通信を行う Query Dispatcher の役割である.CDP はクライアントから受け取ったクエリに応じて適切な OLTP_DBP や OLAP_DBP を指定し, クエリの送受信を行う. もう一つの役割は HTAL を動作させ, 動的にコアの割り当てを変化させることでコア全体での負荷分散を行う Balancer の役割である.

このように 4 種類の処理専用のコアを設定することで, 高負荷なワークロードにおいても,OLTP_DBP や LS に与える影響を最小限化し,OLTP 処理性能と OLAP データの Freshness を維持することが可能であると考えられる.

3.3　ロードバランサ

HTAL では多様なワークロード各々に適したリソース配分を実現するため, 各処理を割り当てたコアの再配分を実行する. 図 4 は OLTP ヘビーなワークロードと OLAP ヘビーなワークロードを実行している場合における HTAL の動作を表している. HTAL は各コアの負荷状況を把握し,OLTP ヘビーなワークロードと判断した場合には OLAP 処理のリソースを OLTP 処理を実行するようにアサインの変更を行う. 一方で OLAP ヘビーなワークロードと判断した場合には OLTP 処理のリソースを OLAP 処理を実行するようにアサインの変更を行う. 接続するクライアントが増加した場合や即時性の高い OLTP ヘビーな場合には DBP だけではなく,Query Dispatcher を行う CDP に対する負荷が増加することを考慮し,HTAL は OLTP_DBP・OLAP_DBP・LS・CDP の 4 種類全てに対して動的な負荷分散を行う.

HTAL はアルゴリズム 1 とアルゴリズム 2 を基に動作する. アルゴリズム 1 では現在実行されているワークロードにおいて, どの処理に対して優先的にコアを割り当てる必要があるか判断し, 優先する処理の状態に遷移する. 入力値としては OLTP_DBP が現在処理しているクライアント数を表す W_T と OLAP データの Freshness を表す Freshness Rate [7] が与えられる.

2 節で述べた通り, 本手法では OLTP 処理性能・更新適用性能・OLAP 処理性能の順に優先的にコアの割り当てを行うため,OLTP・LS・OLAP の順に状態遷移の判断を行う.OLTP ヘビーなワークロードでは W_T が閾値を越えるため,OLTP モードに遷移する. 一方で OLAP ヘビーなワークロードであっても,W_T や Freshness Rate が閾値を超えない場合にのみ OLAP モードに遷移するため,OLAP 処理性能の優先度は最も低く設定されている.

アルゴリズム 2 ではアルゴリズム 1 で得た状態に基づいて, 各処理に割り当てられたコア数を変化させる必要があるか判断を行う. 入力値として CPU 使用率平均と iowait 平均が与えられる. 入力値から OLTP_DBP・OLAP_DBP・CDP に関する負荷状況を判断し, コア数を増やす必要がある場合には AssignmentChange によって対象のコアアサインを交換する.AssignmentChange は対象のコアをインクリメントする関数であり, 各状態において優先していない処理を実行しているコアの中からデクリメント可能なコアを決定し, 処理の入れ替えを行う関数である.

なお OLTP に関しては Range パーティショニングを基に OLTP_DBP に対して割り当てを行っているため,skew が生じているワークロードでは一部の OLTP_DBP では負荷が過多だが, 別の OLTP_DBP に余裕がある状況が生じる可能性が考えられる. そのようなワークロードに対応するため, アクセスに偏りが生じている場合には PartitionAssign によって,Range パーティショニングの範囲を変化させることで OLTP_DBP 間での負荷分散を行う.

以上のようなアルゴリズムを用いて動的にコア数を増減し計算リソースを配分することで, OLTP 処理性能・更新適用性能・OLAP 処理性能の順に優先付けされたロードバランスが可能と考えられる.

Algorithm 2 コア数バランシング

Data: U_C : CPU 使用率平均 (CDP)
Data: D_T : OLTP 実行キュー待機数の分散
Data: C_{LS} : LS のコア数
Data: C_{ML} : LS の最高効率に必要なコア数
Data: U_A : CPU 使用率平均 (OLAP_DBP)
Data: W_A : iowait 平均 (OLAP_DBP)

```
Core_Balancing
    while True do
        State ⇐ GetState()
        if (State = S1) then
                if (U_C ≧ γ) then
                        AssignmentChange(CDP)
                else if (D_T ≧ δ) then
                        AssignmentChange(OLTP)
                else
                        PartitionAssign() // partition 変更
        else if (State = S2 and C_LS ≦ C_ML) then
                AssignmentChange(LS)
        else if (State = S3) then
                if (U_C ≧ γ) then
                        AssignmentChange(CDP)
                else if (U_A ≧ ϵ and W_A ≦ ζ) then
                        AssignmentChange(OLAP)
```

表 1　実 験 環 境

	クライアント	サーバ
プロセッサ	Intel Xeon E7-4860	Intel Xeon Platinum 8260M Processor
ソケット数	4	2
論理コア数	80	96
L1 キャッシュ	32 KB+32 KB	32 KB+32 KB
L2 キャッシュ	256 KB	1,024 KB
L3 キャッシュ	24,576 KB	36,608 KB
RAM	32 GB	384 GB
二次記憶装置	DELL MZ-5EA2000-0D3 SSD(200 GB)	OLTP : Intel Optane Persistent Memory(1.5 TB) OLAP : Dell H8X3X SSD(880 GB) × 2
OS	Ubuntu 16.04.3	Ubuntu 18.04.5
PostgreSQL	11.5	13.0

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.

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,Log shipping として PostgreSQL の Streaming Replication 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,PostgreSQL のバックエンドプロセスの 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 実験構成

表 2 は実験で使用した手法を表している.Baseline としてコアアサイン・HTAL によるロードバランス・OLAP データセットの切り替え・Log shipping を実行しない手法を用いる. なお, 図 5 のように Baseline は Unified Storage と Decoupled Storage の 2 種類のアーキテクチャを用いた. また, 比較手法として HTAL を適用せずにコアアサインのみを行う手法を用いて実験を行った.Baseline と提案手法を比較することでコアアサイン手法や OLAP データセットを OLTP データから分離させて二重化するアーキテクチャの効果検証. 比較手法と提案手法を比較することで HTAL によるロードバランス機能の効果を検証を行った. 各手法のトランザクション分離レベルは Baseline (Unified) のみ OLTP と OLAP 両方を SERIALIZABLE に設定. それ以外の 3 つの手法では OLTP は SERIALIZABLE,OLAP は REPEATABLE READ に設定して実験を行った.

3.3 小節で述べた HTAL のアルゴリズムで使用するパラメータは表 3 のように設定を行う. 比較手法は 120 秒おきに OLAP データセットの切り替えを行い, 提案手法は HTAL が Freshness Rate の低下を検知した際に OLAP データセットの切り替えが発生する.

表 4 は比較手法におけるコア数配分と提案手法における初期状態のコア数配分を表している. 論理コア数が 96 のサーバに対して,OLTP と OLAP の処理コアがおおよそ 1 対 1 になるように設定し,CDP 数に関しては先行研究 [6] と同様に設定した. 比較手法は HTAL を使用しないためコア数配分は表 4 の状態から変化しない. 一方で提案手法は表 4 の状態を初期状態にし, 負荷状況に応じて各処理のコア数を増減させる.

4.3 実験内容

本実験では 2 種類の実験を行った. 一つ目の実験は最優先の性能要件である OLTP に負荷が偏った OLTP ヘビーなワークロードに対する性能評価. 二つ目の実験では OLTP と OLAP の負荷割合がリアルタイムに変化するワークロードを用いて評価を行った. 本小節では二つの実験の内容について説明を述べる.

4.3.1 実験 1：OLTP ヘビーワークロード

実験 1 では HTAP 向けベンチマークである CH-benCHmark [8] のデータを用いて評価実験を行った.

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,OLTP・OLAP 処理性能と更新適用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 SF (Scale Factor)200 のデータ (約 20 GB) を使用した.

なお, 今回の実験では CH-benCHmark における 22 種類の OLAP クエリでは実行時間に散らばりが大きいため, 実行時間が似た値であり関連研究 [4] でも評価に使用されている Q1,Q6,Q19 の 3 種類を用いて実験を行った.

OLTP 処理性能として 1 秒間に処理したトランザクション数を表す tps (transaction per second),OLAP 処理性能として OLAP 処理性能として 1 時間で処理が可能なクエリ数を表す qph (query per hour), 更新適用性能として Freshness Rate の 3 つについて計測を行った.Freshness Rate は Algorithm1 で示した式で算出される. 本実験では PostgreSQL の統計情報であり,Streaming Replication によって更新されているデータは何秒遅延しているかを表す replay_lag を用いて OLAP データの遅延秒数を取得することで更新が未適用なタプル数を取得し,Freshness Rate を算出した.

実験 1 では CH-benCHmark における client 数を OLTP client 数を 100,OLAP client 数を 20 に設定し,OLTP に比重が偏ったワークロードを実行した際の各手法における性能評価を行った.

4.3.2 実験 2：CH-benCHmark+

実験 1 で使用した CH-benCHmark では OLTP と OLAP の負荷比率の調整をリアルタイムに変更することが不可能である.CH-benCHmark における OLTP と OLAP の負荷比率はデータベースに同時接続する OLTP と OLAP の stream 数を事前にパラメータとして設定するため, ベンチマーク実行時に OLTP と OLAP の負荷比率は変更されない. 本研究で提案する提案手法は多様なワークロード各々に適したコアアサインの動的な取得を行うため, 性能評価で使用するベンチマークは OLTP と OLAP の負荷比率を動的に調整する必要がある.

そこで実験 2 では CH-benCHmark$^+$ という CH-benCHmark に OLTP と OLAP の負荷比率調整機能を追加したベンチマークを使用する. CH-benCHmark$^+$ はデータベース全体に同時接続する client 数と時間変化する OLTP・OLAP の比率をパラメータとして設定する. そして 1client は定期的に Load Ratio にアクセスし,OLTP と OLAP の比率に応じて OLTP のトランザクションと OLAP クエリどちらを実行するか決定する. 事前に指定された時間が経過すると OLTP・OLAP を実行する client 数が変化し,OLTP・OLAP の負荷比率がリアルタイムに変更されるワークロードである. 以上のような処理を行う CH-benCHmark$^+$ を用いて実験 2 を行った.

実験 2 で使用する OLAP クエリは実験 1 同様 Q1,Q6,Q19 の 3 種類を用いて評価を行い, データサイズに関しても同様に SF 200(約 20 GB) で生成したデータを使用した. 評価指標に関しても実験 1 同様に tps,qph,Freshness Rate の 3 つに加えて,OLTP トランザクションの ABORT 率を計測した.

リアルタイムに変化する OLTP・OLAP の負荷は 180 秒毎に client 数が 50：50,70:30,30:70 の順に変化させることで, 負荷比率が釣り合う状態,OLTP ヘビーな状態,OLAP ヘビーな状態の順にワークロードを変化させて評価を行った.

4.4 実験結果

本小節では実験 1 と実験 2 の実験結果について述べる.

4.4.1 実験 1：OLTP ヘビーワークロード

図 6〜図 8 が実験 1 の実験結果である. OLTP 処理性能である tps に関して比較手法と提案手法を比較すると, 比較手法は時間経過に対して性能変化が見られないが, 提案手法は時間経過とともに OLTP 処理を行う OLTP_DBP が増加するこ

表 2　手 法 一 覧

	コアアサイン	HTAL	OLAP データ切り替え	LS	OLAP データ分離	OLTP isolation_level	OLAP isolation_level
Baseline (Unified)						SERIALIZABLE	SERIALIZABLE
Baseline (Decoupled)					✓	SERIALIZABLE	REPEATABLE READ
比較手法	✓		✓	✓	✓	SERIALIZABLE	REPEATABLE READ
提案手法	✓	✓	✓	✓	✓	SERIALIZABLE	REPEATABLE READ

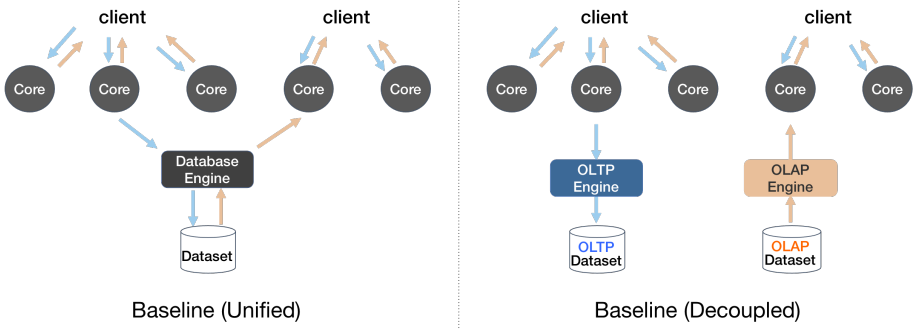

図 5　Baseline 概要図

表 3　アルゴリズムパラメータ

algorithm1		algorithm2			
α	β	γ	δ	ϵ	ζ
2	85%	60%	8	80%	10%

表 4　比較手法コア配分

OLTP_DBP	40
OLTP_CDP	8
OLAP_DBP	44
OLAP_CDP	2
HTAL_CDP	1
LS	1

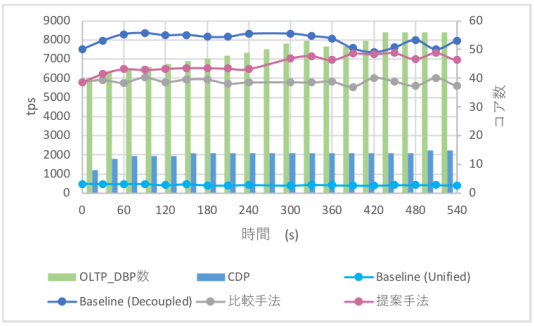

図 6　実験 1：OLTP 処理性能

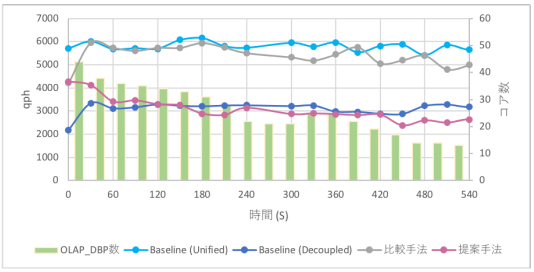

図 7　実験 1：OLAP 処理性能

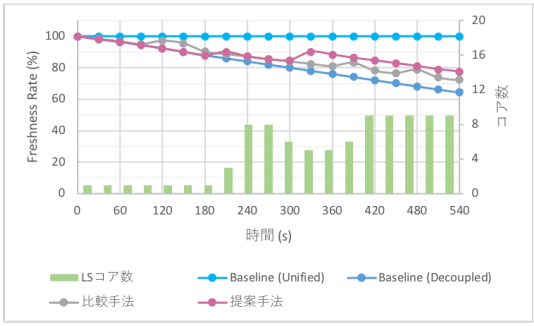

図 8　実験 1：更新適用性能

とで性能が向上していることが確認でき, 実験終了直前には比較手法より 32.6%OLTP 処理性能が向上する結果になった. この結果から HTAL は OLTP ヘビーなワークロードに対して OLTP_DBP を増加させる動作ができていることがわかる. また Baseline (Unified) の OLTP 処理性能は OLAP 実行に伴う read 命令によって OLTP が妨げられることで極端に低い性能であることがわかる. 図 7 の OLAP 処理性能も考慮すると,Baseline (Unified) は OLAP 性能が優先されてしまい OLTP 性能を維持した HTAP 実行が実現できていないことが確認できる. Baseline (Decoupled) と提案手法を比較すると,Baseline (Decoupled) は実験開始時から提案手法より優れた OLTP 処理性能であり, 時間経過に対して OLTP 処理性能に変化は見られない. 実験終了直前において提案手法は Baseline (Decoupled) より 8.56%性能が低い結果になった. このような結果になった理由として, 提案手法は Baseline (Decoupled) では行っていない Log shipping やコアアサイン処理を行っているため, それらの処理を実行するための LS や CDP が設定されている. 一方で Baseline (Decoupled) はそれらのオーバーヘッドがないため, 提案手法より優れた OLTP 処理性能になったと考えられる.

次に図 7 の OLAP 処理性能である qph に関して考察を行う. 提案手法では時間経過とともに OLAP_DBP 数が減少していることが確認できる. これは OLTP ヘビーなワークロードに対して HTAL が OLTP_DBP 数を増加させていることが原因である. したがって, 提案手法は OLAP_DBP の減少に伴って OLAP 処理性能は低下していき, 比較手法と比べて 55.6%性能が低下する結果になった. 提案手法は OLAP 処理性能よりも OLTP 処理性能と更新適用性能を優先してロードバランスを行っているためこのような結果になったと考えられる.

図 8 は提案手法と比較手法における OLAP 対象データの Freshness Rate を表している.Baseline (Unified) は OLTP と OLAP のデータセットが同一のため,OLAP 実行における Freshness Rate は 100%であり,Baseline (Decoupled) は Log shipping による更新を行っていないため Freshness Rate は単調に低下していく. 比較手法は 120 秒経過毎に OLAP データセットの切り替えが発生し,Freshness Rate が上昇する. また, 提案手法は比較手法と比較すると LS コア数が HTAL によって増加しているため, 切り替え後の Freshness Rate に関しては比較手法よりも緩やかに低下しているため比較手法と比べると 6.37%向上する結果になった.

4. 4. 2 実験 2 : CH-benCHmark$^+$

図 9〜図 11 が CH-benCHmark$^+$ を用いた実験 2 の実験結果を表している. 提案手法は 50 : 50 の負荷では OLTP_DBP と OLAP_DBP の増減を繰り返し, ワークロードに対してコアの増減に釣り合いが取れている状態である.70 : 30 の負荷では OLAP_DBP が減少し,OLTP_DBP を増加させる OLTP ヘビーなワークロードに対する処理が HTAL によって行われている.30 : 70 の負荷に対しては OLTP_DBP や CDP は減少しているが,OLAP_DBP は増加せずに LS が増加していることが確認できる. この配分は OLAP ヘビーなワークロードにおける Freshness が低い際のロードバランスであり,OLAP ヘビーなワークロードに対して OLAP 処理性能より更新適用性能を優先する HTAL が想定した処理をしていることが確認できる.

OLTP 処理性能に関しては OLTP ヘビーな 180 秒〜360 秒以外では比較手法と同程度の OLTP 処理性能,OLTP ヘビーな時間帯では比較手法を上回り 10.1%性能向上している. 次に 2 つの Baseline と比較すると Baseline (Unified) は実験 1 同様,OLAP 実行によって他の手法より低い OLTP 処理性能であることが確認できる. また,Baseline (Decoupled) と比較した場合には OLTP ヘビーな 180 秒〜360 秒では実験 1 同様, 提案手法は Baseline(Decoupled) に対して 17.8%性能低下している. 一方で OLAP ヘビーな 360 秒〜540 秒の時点では Baseline (Decoupled) を上回る結果になり,34.9%性能向上する結果になった. 提案手法はコアアサインにより OLTP 処理に使用するコアが指定されているため,OLAP 実行によって OLTP 性能に受ける影響が小さかったことに対し,Baseline (Decoupled) はどちらの手法も OLAP ヘビー時には OLAP 実行による影響を受けてしまうためこのような結果になったと考えられる.

表 5 は CH-benCHmark$^+$ 実行時の OLTP トランザクション総実行数や ABORT 率を表している. この結果から Baseline (Unified) は他の手法と比べて ABORT 率が高いことがわかる. これは OLTP と OLAP が同一データセットで実行されていることが原因と考えられる. また Baseline (Decoupled) と提案手法を比べると提案手法は ABORT 率が低く抑えられていることが確認できる. 提案手法はコアアサイン手法によって OLTP 実行時に同一 Range にアクセスする DBP が指定されているため ABORT 率が低い結果になったと考えられ, 提案手法は ABORT を抑制する効果があることが確認できた.

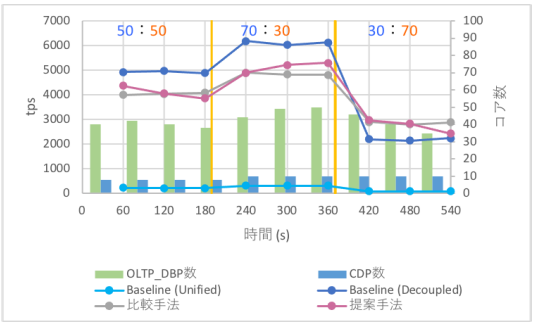

図 9　実験 2 : OLTP 処理性能

表 5　実験 2 : ABORT 率

	Baseline (Unified)	Baseline (Decoupled)	比較手法	提案手法
ABORT 率	10.5%	8.2%	5.4%	6.4%
実行 Tx 数	98,958	2,104,867	1,928,263	2,009,562
ABORT 数	10,444	168,389	104,126	128,611

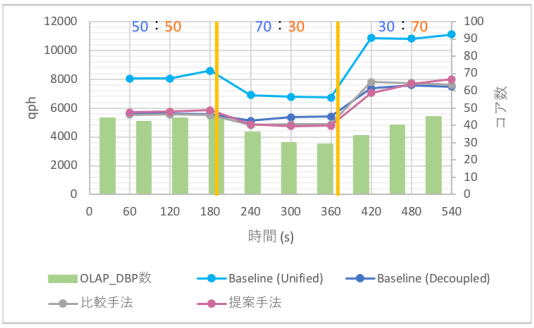

図 10　実験 2 : OLAP 処理性能

次に OLAP 処理性能に関しては, ワークロードの変化に対して OLAP_DBP は減少しているため比較手法より低い処理性能であり,5.3%の性能差が生じている. 提案手法は HTAL によって OLAP_DBP の調整を行っているが,OLAP_DBP は増加せず LS を増加させ更新適用性能を優先していたためこのような結果になった.

図 11 は CH-benCHmark$^+$ 実行時の時間変化による Freshness Rate の推移である. 比較手法に関しては 120 秒毎に OLAP データセットの切り替えが発生しており, 切り替わるたびに Freshness Rate は上昇するが, 時間経過とともに 2 つの OLAP データセット両方の Freshness が低下してしまうため, 切り替え直後の Freshness Rate は次第に低下していく. 提案手法に関しても比較手法と同様の傾向が見られるが,HTAL によって LS コアの増加しているため, 比較手法より緩やかな低下が見

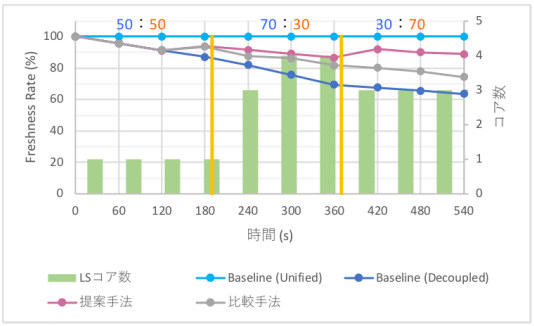

図 11　実験 2：更新適用性能

られ,19.9%向上する結果になった. また実験 1 同様に Baseline (Decoupled) は単調に減少していることから提案手法は Baseline (Decoupled) より優れた Freshness Rate を実現しながら可能な限り OLTP 処理性能を向上させるリソース配分を行っていることが確認できる.

5　おわりに

5.1　まとめ

本研究では複数コア環境においてリアルタイムに変化するワークロードに対して,OLTP 処理性能と Freshness を維持しながら, 可能な限りの OLAP 処理性能の獲得を目的に動的なコアへの処理の割り当てを行う HTAP システムを検討した.

複数コアに適したシステム構成に関する検討を行い,Decoupled Storage を前提とした OLAP データセットの切り替えを行うアーキテクチャを採用し,OLTP・OLAP のデータ形式に関しては更新適用性能を考慮し, 双方を行指向データにするデータ形式を採用した.

また, 検討したシステム構成に対して 4 種類の処理専用コアを設定するコアアサイン手法を提案し, コアアサイン手法に対してのロードバランサである HTAL を提案した.

提案した HTAP システムに対して OLTP ヘビーなワークロードやリアルタイムに OLTP・OLAP の負荷比率を変化させることが可能な CH-benCHmark$^+$ を用いて性能評価実験を行った結果,HTAL を使用しない比較手法と比較して OLTP ヘビーなワークロードでは OLTP 処理性能を 32.6%, 更新適用性能を 6.37%向上した. 同様に CH-benCHmark$^+$ を用いた, リアルタイムに OLTP と OLAP の負荷比率を変化させるワークロードに対しても同様に OLTP 処理性能を 10.1%, 更新適用性能を 19.9%向上させたことから HTAL によるロードバランスの有効性を示した. また OLTP 性能に優れた Decouple Storage を採用した Baseline との比較では OLTP ヘビー時における OLTP 性能は 8.56%低下してしまうが,OLAP ヘビー時にはコアアサインによって OLTP への影響が軽減され,34.9%向上する結果になったことからコアアサイン手法の有効性を部分的に示した.

5.2　今後の課題

本研究ではロードバランスのパラメータとして CPU 使用率やクエリキューの待機数を用いたアルゴリズムを使用したが,CDP 一つが担当する DBP 数などといった別パラメータを用いたアルゴリズムの効果を検証することが今後の課題である.

また OLTP 処理性能をより優先した構成として,OLTP 実行と OLAP 実行を別ノードに分けたマルチノード構成に対して, コアアサイン手法と動的なロードバランスの適用を検証し, より優れた性能要件を実現できるか実験する必要がある.

本研究では OLTP 処理性能と更新適用性能を優先した構成を採用している. したがって,OLAP データエンジンとしても OLAP 処理に有利ではない行指向データベースを使用し, 実験 1 と実験 2 では OLAP 処理性能が低下する結果になった. よって,OLAP データセットのデータ形式や OLAP データエンジンを OLAP 実行に有利な構成に対応させることも今後の課題としてあげられる.

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）の委託事業（JPNP16007）の結果得られたものです.

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] Aunn Raza, Periklis Chrysogelos, Angelos Christos Anadiotis, and Anastasia Ailamaki. Adaptive htap through elastic resource scheduling. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 2043–2054, 2020.

[5] 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.

[6] 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.

[7] Mokrane Bouzeghoub. A framework for analysis of data freshness. In *Proceedings of the 2004 international workshop on Information quality in information systems*, pp. 59–67, 2004.

[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.