複数OLAPエンジンを持つHTAPの更新反映と問合せ割当手法の評価

御喜家奈波† 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: †{mikiya,hikida,hanhlh}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし データ分析の需要が高まる中で，フレッシュなデータに対する分析処理を可能にする HTAP が注目されている．HTAP 研究では OLTP 性能，OLAP 性能，データのフレッシュネスを向上させることが主題となっている．しかし，多くの研究では単一 OLAP エンジン環境が想定されており，OLAP エンジンの種類や数を複数に拡張した場合，処理エンジン選択による OLAP 性能向上の余地がある．本研究では，OLAP エンジンを複数用意した HTAP において，OLTP での差分を迅速に更新反映するため，エンジンやデータセットの配置構成を検討し，問合せ内容に応じて使用する OLAP エンジンとコア数を動的に割り当てることによる性能評価を行う．

キーワード 問合せ処理，HTAP，複数 OLAP エンジン，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トランザクション処理を行う OLTP(OnLine Transaction Processing) と分析処理を行う OLAP(OnLine Analytical Processing) に大別できる．大規模データベースに対して分析処理を実行することができる OLAP エンジンについては，種類による性能を比較する研究 [4], [11] が行われており，実験結果より，クエリに応じて処理に適したエンジンが異なる．データ分析の需要が高まる中で，OLTP における更新内容をできるだけ瞬時に OLAP に適用することでフレッシュなデータに対する分析処理を可能にする，HTAP(Hybrid transactional/analytical processing) [7] が注目されている．

HTAP 研究では，OLTP 処理性能，OLAP 処理性能，OLAP で参照するデータのフレッシュネスを向上させることが主題となっている．HTAP 構成における OLAP エンジンの種類や数を複数に拡張した場合，分散処理や処理エンジン選択による OLAP 性能向上の余地がある．しかし多くの研究では，単一 OLAP エンジン構成の HTAP におけるこれらの性能向上について議論されており，複数 OLAP エンジン環境を想定した研究は十分にされていない．これより，HTAP 構成における OLAP エンジンの種類や数を複数に拡張した場合，クエリの分散処理や処理エンジン選択による OLAP 性能向上の余地がある．

そこで本研究では，HTAP 実行において OLTP 処理性能と OLAP データのフレッシュネスを維持しながら，OLAP 処理性能を向上させることを目的とする．OLAP 処理性能向上のため，OLAP エンジンとして OSS で導入が容易な PostgreSQL と SparkSQL [1] を用意した HTAP システムを考える．このシステムにおいて，OLTP での更新差分を迅速に OLAP に反映してフレッシュネス性能を維持するため，OLAP エンジンやデータセットの配置構成を検討する．OLTP エンジンには我々の研究グループで進めている，HTAP システムを Serializable にする手法である RSS(Read Safe Snapshot) [8] を組み込んだ PostgreSQL を使用する．Spark SQL は非商用で使用する場合データの差分更新が行えず，また RSS 非対応のため，RSS によってすでに更新された PostgreSQL のデータをコピーすることでデータセットを作成する．また，問い合わせ内容に応じて使用する OLAP エンジンを動的に割り当てる手法を提案する．考案した構成，割当手法に対して，TPC-C [9] に基づくトランザクションと TPC-H [10] と同等のクエリを並列実行することができる CH-benCHmark [2] を用いて，OLTP 処理性能，OLAP 処理性能を計測する．得られた計測結果を用いて，単一 OLAP エンジン環境が想定された HTAP システムの先行研究 [13] の性能と比較し評価する．

1.2 本稿の構成

本稿は以下の通りに構成される．2 節では背景知識と関連研究について説明し，3 節では本研究における手法を提案する．4 節では実験内容について述べ，実験結果から考察を行う．最後に 5 節で本稿の結論を述べる．

2 背景知識

2.1 HTAP(Hybrid Transactional/Analytical Processing)

OLTP はデータの挿入，更新，削除を主に行う書き込み指向で，操作に関係するデータサイズは比較的小さいが，大量のトランザクションが発生する可能性がある．一方，OLAP はデータセットを分析するため読み込み指向で，時には膨大なデータセットを読み込む必要があることや，分析処理の複雑化により作業負荷が重い．このため，データベース自体を OLTP と OLAP に分け，トランザクション処理の ACID 性を確保したシステムが主流であった．しかし，これらが分離されたシステムでは，OLTP によるデータの更新が常に行われているにもかかわらず，OLAP で最新のデータを使用することが困難であった．

HTAP はこれら 2 種類の処理を 1 つのシステムに集約したも

のであり，OLTP によるデータの更新を瞬時に OLAP が参照するデータに適用することで，高速な OLTP とフレッシュネスの高いデータを用いた OLAP を可能にする．

HTAP データベースにはスケールアップとスケールアウトの 2 つのスケーリング方法があり，それぞれの代表的なデータベースに HyPer [5] と SAP HANA [6] が挙げられる．

HyPer [5] は単一サーバ上で NUMA を利用してノードを複数作成することで，スケールアップによる性能向上を図っている．単一サーバ構成の場合，サーバ間のデータ転送が不要であり，全てのデータをメモリ上でアクセスできるため処理時間が短縮される．しかし全ての機能を 1 つのサーバ上で動かすため，大規模システムを構築する際には大容量のメインメモリを搭載した強力なサーバが必要となる．

SAP HANA [6] は OLTP ワークロードを 1 台のサーバに保持し，NUMA を利用して処理にコアを複数使用するが，さらにワークロードを他のサーバにスケールアウトさせる機能を持つ．また，OLTP 用データセットと OLAP 用データセットを分離することで両処理の効率化を実現しながら，定期バッチ処理よりも高い頻度で OLAP 用データセットに更新を適用するためフレッシュネスの高いデータを用いた分析処理が可能である．しかし，OLTP での更新を OLAP データへ適用するマージ処理が行われる際に OLTP と OLAP の性能が一時的に低下してしまう．また，高負荷時には OLTP，OLAP，更新処理がリソースを競合するため，フレッシュネスの高い OLAP 実行が困難である [12].

2.2 HTAL(Hybrid Transactional/Analytical processing Load balancer)

HTAL [13] は単一ノード，複数コア環境での HTAP システムにおいて，各処理への割り当てコア数を動的に変動させるロードバランサであり，主に OLTP 処理性能と OLAP データのフレッシュネスの向上を目指した研究である．OLTP エンジン，OLAP エンジンともに PostgreSQL を使用し，データセットについては OLTP と OLAP がそれぞれ独立したデータセットを持つ Decoupled Storage アーキテクチャをとっている．このアーキテクチャでは OLTP による更新を随時 OLAP が参照するデータに反映し，データのフレッシュネスを保つ必要がある．そのため，OLTP データに対して実行されたトランザクション処理の実行ログを OLAP データに送り，同じトランザクション処理を OLAP データに対しても行うことで新しいデータを取得する Log Shipping を行う．HTAL では PostgreSQL の Streaming Replication 機能を用いることでこの Log Shipping を行なっている．この時，OLAP データに対して Log Shipping による書き込み処理と OLAP クエリによる読み込み処理が同時に発生することを考慮して，Log Shipping による差分更新用と OLAP クエリ処理用の 2 つのデータセットを用意している．これらのデータセットは定期的に用途の切り替えを行うことで，OLAP データのフレッシュネスを維持する．

このアーキテクチャにおいて複数コアを活用するため，OLTP，OLAP，Log Shipping，コアアサイン処理の各処理専用のコア

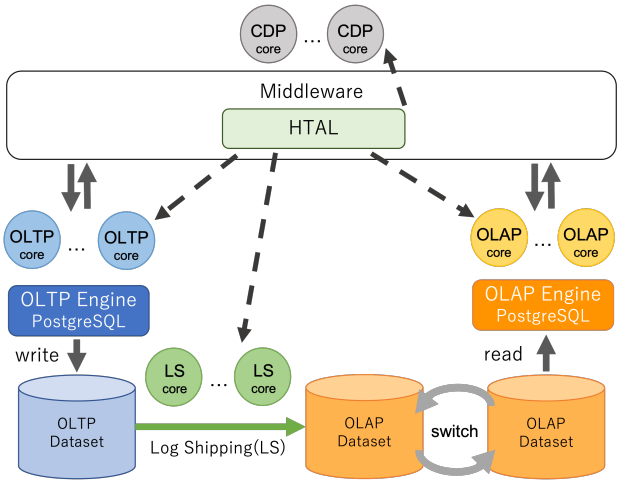

図 1　HTAL を用いたシステムのアーキテクチャ

を設定し，処理負荷に応じて動的にコアをアサインする HTAL が提案されている．図 1 は HTAL を使用したシステムのアーキテクチャを示している．OLTP は，トランザクションクエリがアクセスするデータの主キーに応じて専用のコアを割り当てることでキャッシュヒット率を高め，OLTP 性能の向上を図っている．OLAP は OLAP 専用コアのうちいずれかで実行される．Log Shipping 用コアはトランザクションログの送受信や OLAP データへの更新適用に使用される．コアアサイン処理用コアは各処理へのコアの動的割り当てや OLTP，OLAP によるクエリの送受信に使用される．

HTAL の性能は CH-benCHmark [2] を使用して OLTP 処理性能と OLAP 処理性能が計測され，HTAL を使用しない場合と比較して OLTP 処理性能が向上したことが確認された．また OLAP データのフレッシュネスも，HTAL を使用しない場合と比較して向上した [13].

2.3 RSS(Read Safe Snapshot)

HTAP システムにおける課題の 1 つはデータの一貫性である．OLTP による更新データの一貫性を保証する並行性制御が必要不可欠であり，トランザクション同士の独立性が求められる．しかし多くの HTAP システムは，全体では Serializability を保証していない．例えば，2.1 小節でも紹介した SAP HANA [6] はシステム全体の分離レベルとして Snapshot Isolation を採用している．これは，より高い分離レベルである Serializable を HTAP システム全体で実現するためにはリソースコストがかかり，OLTP，OLAP 処理性能の大幅な低下とフレッシュネスの低下を招くためである．

これに対して，RSS [8] は Serializability を保証する Snapshot を低コストで提供するため，データの一貫性を保証しながら OLTP 処理性能への影響を少なくして，高フレッシュネスを実現する．

本研究ではこの RSS を利用することでデータの一貫性を保証しながら処理性能やフレッシュネスを低下させることなく HTAP システムを構築する．

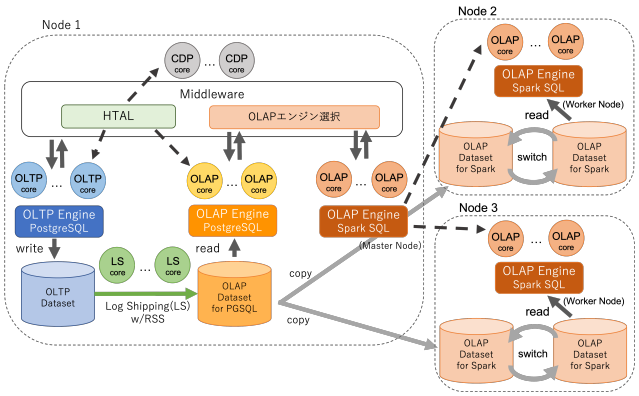

図 2　構　成　案

2.4　既存研究の課題点

先行研究 [13] の課題点として OLAP 処理性能の向上に余地がある点が挙げられる．HTAP のビジネス利用上，OLTP のレスポンスタイムの短縮が求められている背景から，先行研究ではシステムアーキテクチャや動的コアアサインにおいて OLTP を最優先にしているため，OLAP 処理性能の低下がみられた．

これに対して本研究では，OLAP エンジンの種類を増やし，これらの OLAP エンジンが十分なリソースを使用できるよう OLAP 専用サーバを追加で用意する．この OLAP エンジンを拡張した構成において，HTAL のアルゴリズムを利用して OLTP 処理性能とフレッシュネスを維持しつつ，クエリの分散処理による OLAP 処理性能の向上を目指す．

3　提 案 手 法

本研究では HTAP における OLAP 処理性能を向上させるべく，図 2 に示す構成を検討する．図 2 は 3 ノードの場合の構成を示したものである。構成の詳細を以下で説明する．

3.1　複数ノード

OLAP では，大量のデータを高速に処理する方法の 1 つに分散処理がある．分散処理は大きく分けて 2 つあり，1 つは 1 台のマシンに多数の CPU やコアを搭載することで処理を分散するスケールアップ．もう 1 つは処理を複数のマシンに分散させるスケールアウトである．本研究ではノード数を複数にスケールアウトすることによって OLAP 処理に使用できるコアを増やし，OLAP の分散処理が発揮できる環境を構築する．提案する構成案では，先行研究 [13] で提案された単一ノードで HTAP を実行できるノードの他にノードを新たに用意し，ノード内全てのコアを分散処理を得意とする Spark SQL に使用することで OLAP 処理性能の向上を図る．

3.2　複数 OLAP エンジン

OSS で導入の容易な PostgreSQL と Spark SQL を OLAP エンジンとして用意し，後述の OLAP エンジン選択手法によりクエリに応じて処理の速いエンジンを選択，そしてクエリの並列処理によって OLAP 処理性能の向上を図る．

表 1　実 験 環 境

プロセッサ	Intel Xeon Platinum 8176
ソケット数	2
論理コア数	112
L1d キャッシュ	1.8 MiB
L1i キャッシュ	1.8 MiB
L2 キャッシュ	56 MiB
L3 キャッシュ	77 MiB
RAM	1 TB
OS	Ubuntu 20.04.4

3.2.1　PostgreSQL

本実験では OLTP エンジン，OLAP エンジンの両方で PostgreSQL を使用する．HTAP のアーキテクチャには Unified Storage と Decoupled Storage の 2 種があり，PostgreSQL はどちらにも対応している．本実験では OLTP と OLAP のデータが分離されており，それぞれの処理が競合しない分，高い OLTP 処理性能を実現できる Decoupled Storage を採用する．

3.2.2　Spark SQL

Spark SQL は複数ノード構成に対応した cluster mode で使用し，Cluster Manager として YARN を用いる．この YARN についてはクエリごとに Spark アプリケーションを起動する，短期実行に適した client mode を使用する．

Spark SQL の OLAP データを永続化するため Hive Metastore を使用する．これにより，メタデータと実データが作成される．メタデータは Spark SQL の Master Node となるノードに配置され，実データは Spark SQL が受け取ったクエリを実際に処理する Worker Node に配置される．本実験では OLAP 実行時に別ノードのデータを収集するノード間 I/O を減らすため，全ての Worker Node に配置する．この時，ミドルウェアである Apache Hive を使わずに Spark のみで Hive Metastore を使用することができ，ローカルファイルシステムに実データを保存する．一方で，Apache Hive を使用して Hive Metastore に実データを保存すると，HDFS 上に保存される．実データをローカルに保存した場合と HDFS に保存した場合の違いとして，まずデータの保存時間が挙げられる．ローカルファイルシステムに保存する場合，SF200(20GB) では約 3 分かかるのに対し，HDFS に保存する場合 SF200 では約 20 分かかる．また，実データの保存場所によるクエリ実行時間比較を行うため計測を行った．Master Node と 1 つの Worker Node の 2 ノード構成で計測を行い，これら 2 つのノードは表 1 に示す環境を使用した．実行するベンチマークには SF(Scale Factor)200 の CH-benCHmark を使用し，OLAP クエリを実行した．実験結果を図 3 に表す．実データをローカルファイルシステムに保存した場合，Q3, Q13, Q22 は実行時エラーにより計測できなかった．一方で HDFS に実データを保存した場合，一部のクエリは実行に時間がかかるが，全てのクエリを実行することができた．これより，本研究ではクエリ実行の確実性を優先して HDFS 上に実データを保存する．

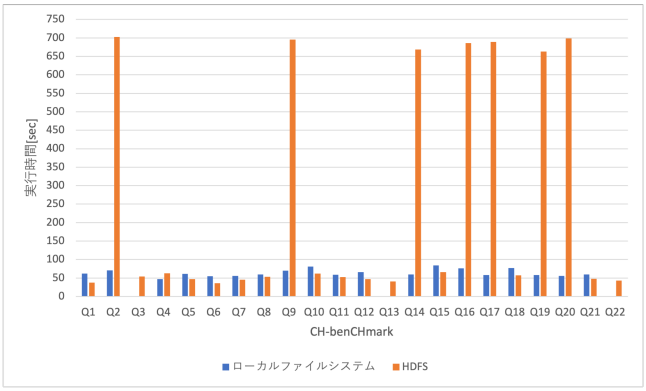

図 3　実データ保存場所によるクエリ実行時間の比較

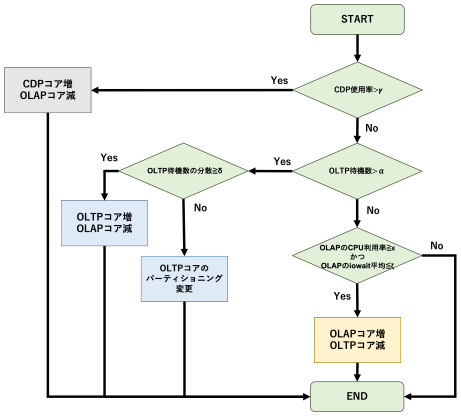

図 4　HTAL を基にしたコアアサイン手法

3.3　コアアサイン

構成案の Node 1 におけるコアアサインは，HTAL [13] を基に OLTP，OLAP，コアアサイン処理の各処理に対して専用のコアを設定し，処理負荷に応じて動的にコアをアサインする．HTAL では OLAP データセットについて，差分更新用と OLAP クエリ処理用に 2 つ保持するため，OLAP データのフレッシュネスを表す数値をもとにデータセットの切り替えや Log Shipping のためのコア数を動的にアサインしていた．それに対して本研究では，OLTP の更新差分がデータセットの切り替えによる中断なく常に PostgreSQL の OLAP データセットに適用されるため，静的にコアをアサインし，システム動作中はコア数を変化させないことにする．

本研究で用いる，HTAL から Log Shipping の動的コアアサインを除いたコアアサインアルゴリズムを図 4 に示す．このアルゴリズムにおいて，OLTP，OLAP 処理を行う PostgreSQL へのコアアサインやその他各処理へのコアアサインは，Linux の sched_setaffinity API を用いることで使用コアを指定する．

本研究で HTAP システム全体のデータ一貫性を保証するために導入する RSS は常に一定数のコアを必要とする．そのため，RSS の作動に必要なコアは静的にコアアサインを行う．

OLAP エンジンである Spark SQL へのコアアサインは，Master Node となる Node 1 では使用コアを固定し，Node 2 以降の Worker Node では Spark が提供する Dynamic Resource Allocation を用いることで，動的に使用コアを調整する．

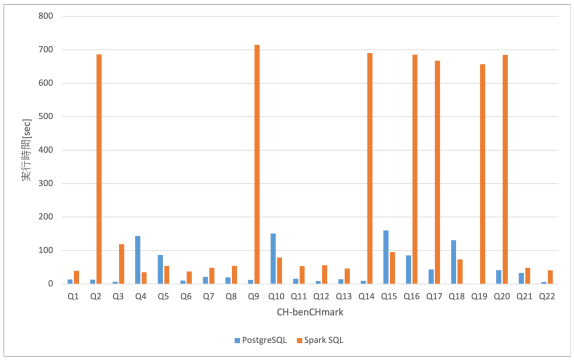

図 5　PostgreSQL と Spark SQL による OLAP クエリの処理時間

3.4　OLAP エンジン選択

OLAP エンジンとして PostgreSQL と Spark SQL を用意したが，クエリによって高速に処理できるエンジンは異なる [11]．本研究で使用する環境，構成での PostgreSQL と Spark SQL で OLAP クエリを実行した時のクエリ処理時間を比較するため，それぞれ計測を行った．Spark SQL の構成については Master Node1 台，Worker Node1 台の 2 ノード構成を用いた．実行ベンチマークは SF200 のデータに対し，CH-benCHmark の OLAP クエリを使用した．計測結果を図 5 に示す．Spark SQL を local モードで使用した関連研究 [11] と比較すると，本研究で用いる cluster モードの Spark SQL は OLAP クエリ処理に時間がかかる．しかし，一部のクエリでは PostgreSQL よりも短い時間でクエリを処理できることから，本研究の構成においてもクエリに応じて適した OLAP エンジンを割り当てることができれば，OLAP 処理性能の向上が期待できる．

本研究ではクエリに応じたエンジンを選択する方法として，実行計画の内容から処理エンジンを決定する．この時，PostgreSQL はクエリを実行することなく実行計画のみを取得できるが，Spark SQL はクエリを実行して初めて実行計画を取得できるため，本研究では PostgreSQL による実行計画から OLAP エンジン選択を行う．PostgreSQL に実行計画を取得するクエリを投げ，得られた実行計画に対して出現するクエリ演算子を種類ごとに集計する．PostgreSQL の実行計画に Index Scan が含まれている場合，テーブルフルスキャンと比較して I/O 量が減少し処理性能が高くなるため，このクエリは処理エンジンとして PostgreSQL を選択する．それ以外のもの，Index Scan が含まれていないクエリは Spark SQL を選択する．

3.5　データ更新適用

本研究で提案した構成案では OLTP 性能を維持するため，Decoupled Storage を採用している．そのため，OLTP による更新を随時 OLAP が参照するデータに反映し，データのフレッシュネスを保つ必要がある．

PostgreSQL が参照する OLAP データへの更新適用は PostgreSQL の Streaming Replication を用いる．このとき，Serializable な Snapshot を使用するため RSS [8] を導入する．RSS は Decoupled Storage アーキテクチャに対応しているため組み込み可能である．RSS を導入すると，OLTP での更新が随

表 2　PostgreSQL の設定

version	12.0
max_connections	1000
max_wal_size	16 GB
default_transaction_isolation (OLTP)	serializable
default_transaction_isolation (OLAP)	repeatable read

表 3　Hadoop の設定

version	3.3.1
Java version	OpenJDK 11.0.17

時適用される OLAP データを直接 OLAP クエリ処理のデータ読み込みに使用できるため，Freshness Rate は実験により高いことを確認している。これに対して、RSS を導入しない先行研究 [13] の構成では OLAP 用のデータセットとして Log Shipping 用と OLAP クエリ処理用の 2 つを用意し，定期的に切り替えていたため，最新のデータを使用した OLAP は困難であり，データのフレッシュネスを示す Freashness Rate は時間と共に低下し，540 秒経過した時点で約 80%になっていることがわかる．

Spark SQL が参照する OLAP データへの更新適用は，JDBC 経由で OLAP エンジンである PostgreSQL のデータを読み込み，保存する．この時 Spark SQL が参照するためのデータは，テーブルを永続的に管理できる Hive metastore に保存する．ただし，Hive metastore への保存はデータの上書きしか行えず，差分更新が行えないため，データ更新用データセットと OLAP の参照用データセットの 2 つを用意する．OLAP 実行とデータ更新適用を同時に行い，定期的に 2 つのデータセットを交換することで，常に Spark SQL を OLAP エンジンとして使用できる状態にする．なお，SF200(20GB) のデータ保存には約 20 分かかるため，最速でデータセット切り替えを行った場合でも，20 分前に更新されたデータまでしか分析対象にすることができない．

4 実　　験

提案手法の性能を検証するため，OLTP 処理性能と OLAP 処理性能を測定する実験を行った．

4.1 実験環境

Spark SQL の実データ保存場所による実行時間の計測を行った環境，表 1 を本実験でも用いる．全てのノードでこの環境のマシンを使用する．

使用ソフトウェアのバージョンや設定詳細は表 2，3，4 の通りである．

4.2 実験構成

表 5 は実験で使用した手法を表している．先行研究 [13] は想定されている単一ノード構成で HTAL アルゴリズムを使用する手法を用いて実験を行った．また提案手法として，RSS を

表 4　Spark SQL の設定

version	3.2.0
spark.yarn.am.cores	4
spark.driver.memory	10 GB
spark.executor.cores	4
spark.dynamicAllocation.enabled	true
spark.shuffle.service.enabled	true

Log Shipping に適用せずに OLAP エンジンの拡張のみを行う手法を用いて実験を行った．先行研究と RSS なしの提案手法を比較することで，OLAP エンジン拡張の効果を検証，RSS ありと RSS なしの提案手法を比較することで Log Shipping に RSS を導入したことによる性能の評価を行う．また，RSS ありの提案手法においてノード数を変化させることで，Spark SQL の分散処理による OLAP 処理性能を評価する．

3.3 小節で述べたコアアサイン手法のパラメータは先行研究と同様，表 6 のように設定を行う．

表 7 は先行研究と提案手法の初期状態のコア数配分を表している．論理コア数が 112 のノードに対して，CDP 数は先行研究と同様に設定した．また，OLTP_DBP コアの数は OLTP_CDP コア数によって決定されるため，先行研究と提案手法では同数に設定している．先行研究の手法では全てのコアが動的コアアサインの対象になるが，提案手法では LS コアと Spark SQL のコアは静的にアサインし，コア数は変化しない．

4.3 実験内容

全ての実験で CH-benCHmark [2] を用いて，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で 360 秒間，OLTP と OLAP を同時実行した際の各手法における処理性能を評価した．CH-benCHmark のデータ生成は oltpbench [3] を用い，SF200 のデータ (20GB) を使用する．OLTP は 5 種のトランザクションの中から，OLAP は 22 種のクエリの中からランダムに実行するものが選ばれる．計測は各 5 回ずつ行い，実験の度にデータは作成し直す．これはトランザクション処理によりデータサイズが増加するためである．OLAP 処理性能として 1 秒あたりの処理トランザクション数を表す tps(transaction per second)，OLAP 処理性能として 1 時間あたりの処理クエリ数を表す qph(query per hour) を用い，30 秒ごとにそれぞれ算出する．OLAP 処理性能については，CH-benCHmark の全 22 種類のクエリの実行時間にばらつきがあるため，360 秒間の平均 qph も示す．

4.4 実験結果

図 6，7 が実験結果である．また，OLAP 処理性能の平均は図 8 のようになった．

まず，先行研究と RSS なしの提案手法を比較すると，OLTP 処理性能は平均 25%向上した．これは一部の OLAP クエリが Node 2 で処理を行う Spark SQL に割り当てられたことで，Node 1 で OLAP 処理をする PostgreSQL へ割り当てられるコアが減り，OLTP に使用できるコアが増えたことによるものだと考えられる．OLAP 処理性能は PostgreSQL 1 台で OLAP

表 5　手法一覧

	ノード数	PostgreSQL	Spark SQL	OLAP エンジン選択	RSS
先行研究	1	✓			
提案手法 RSS なし	2	✓	✓	✓	
提案手法 RSS あり (2 ノード)	2	✓	✓	✓	✓
提案手法 RSS あり (3 ノード)	3	✓	✓	✓	✓
提案手法 RSS あり (4 ノード)	4	✓	✓	✓	✓

表 6　コアアサインパラメータ

α	γ	δ	ϵ	ζ
2	60%	8	80%	10%

表 7　初期コア数配分

コアの種類	先行研究	提案手法
OLTP_DBP	40	40
OLTP_CDP	8	8
OLAP_DBP	60	52
OLAP_CDP	2	2
HTAL_CBP	1	1
LS	1	5
Spark SQL(Master Node)	0	4

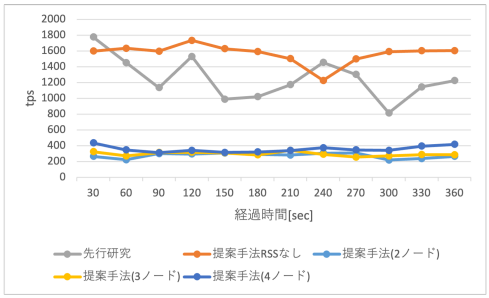

図 6　OLTP 処理性能

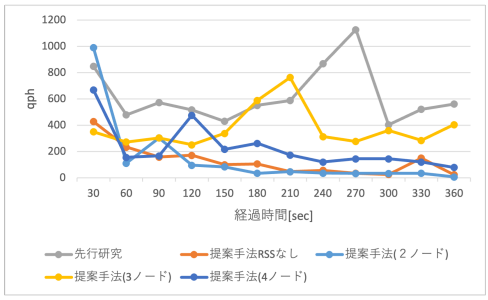

図 7　OLAP 処理性能

クエリを処理する先行研究と比較して，PostgreSQL と Spark SQL にクエリを分散させた提案手法で性能が平均 79%劣化した．このことから，OLAP エンジン選択アルゴリズムの条件が不十分で，1 クエリあたりの処理時間が PostgreSQL 1 台で処理するよりも増加してしまった可能性が考えられる．実際に OLAP エンジン選択アルゴリズムを確認すると，Spark SQL での実行に 600 秒以上かかる Q2，Q14，Q16，Q17，Q19，Q20

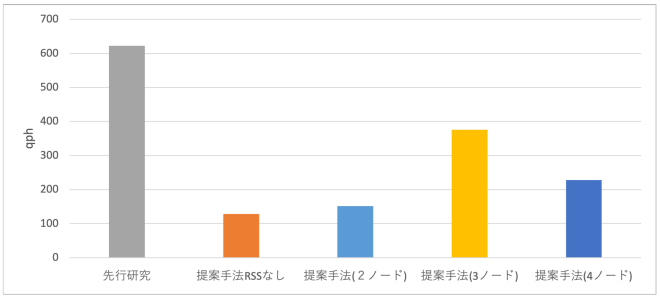

図 8　OLAP 処理性能の平均

のクエリが全て Spark SQL に割り当てられているため，提案手法の qph が劣化したと考えられる．

次に RSS なしの提案手法と RSS ありの提案手法 (2 ノード) を比較すると，OLTP 処理性能は平均 82%劣化した．この理由として，RSS を導入した構成における Node 1 での動的コアアサイン手法は RSS を導入していない構成と同じアルゴリズムを使用しているため，OLTP DBP コアや OLTP CDP コアなど OLTP に関わるコアの動的アサインが正常に行われず，OLTP 処理性能の劣化が起こった可能性が考えられる．OLAP 処理性能は RSS なしと比較して，RSS ありの提案手法では 18%向上した．OLTP 処理性能と同じく，動的コアアサインアルゴリズムが正常に動作しないことによるものとも考えられるが，比較した 2 つの構成は OLAP エンジン構成やノード数は同じため，CH-benCHmark の各クエリの実行時間のばらつきによるものだとも考えられる．

提案手法のうち，ノード数を変えた構成である提案手法 (2 ノード)，提案手法 (3 ノード)，提案手法 (4 ノード) の OLTP 処理性能を比較すると，360 秒間の平均 tps はそれぞれ 275.2 tps，296.5 tps，357.3 tps となっており，ノード数を増やすことで OLTP 処理性能が向上していることがわかる．一方で OLAP 処理性能は 2 ノードから 3 ノードに増やすことで平均 2.5 倍に向上した．これはノード数を増やすことで Spark SQL の分散処理に使用されるコア数が増加し，複数のクエリが Spark SQL で同時実行されるときの各実行時間が減少することで全体の OLAP 処理性能が向上したと考えられる．しかし，4 ノードに増やしたとき，2 ノード構成と比較して OLAP 処理性能は平均 51%向上したが，3 ノード構成と比較すると平均で 39%劣化した．これは CH-benCHmark の各クエリの実行時間のばらつきによる誤差と，20 クライアントによる OLAP 実行では 3

ノード構成で既にリソースが足りており，4 ノード構成にしたときリソース過多となり十分に活用できていない可能性が考えられる．

5 おわりに

5.1 まとめ

本研究では HTAP システムの OLAP 処理性能を向上させるため，複数の OLAP エンジンを持つ HTAP システムの構築を検討した．

OLAP エンジンには PostgreSQL と Spark SQL を採用し，Spark SQL の分散処理を活用するため複数ノードの HTAP システムを構築した．Spark SQL のデータ保存場所には HDFS を採用し，分散処理におけるコア管理には Spark が提供する Dynamic Resource Allocation を採用した．また，Decouple Storage アーキテクチャで処理性能の低下を防ぎながら HTAP システムのデータ一貫性を保証するため，Serializability を保証するスナップショットを提供する RSS を採用した．この構成に対して，OLAP エンジンからクエリに応じて処理エンジンを決定する OLAP エンジン選択手法を提案し，HTAP システムにおける各処理に使用されるコアを動的に割り当てるため，先行研究 [13] で提案されたロードバランサである HTAL を基に動的コアアサイン手法を提案した．

単一 OLAP エンジン，単一ノード構成の先行研究と比較するため，提案した HTAP システムに対して CH-benCHmark を用いて性能評価実験を行った結果，RSS は導入せず複数 OLAP エンジン，複数ノード構成で OLAP エンジン選択を行う提案手法では OLTP 処理性能が 25%向上した．しかし OLAP 処理性能は 79%劣化してしまったことから，OLAP エンジン選択手法の改善が求められる．また RSS を導入した場合，RSS を導入しない構成と比較して OLTP 処理性能は劣化したが OLAP 処理性能は向上した．このことから，RSS を導入した場合に適用可能な動的コアアサインアルゴリズムの改善が求められる．ノード数を 2 ノードから 4 ノードまで増やした場合は，OLTP 性能は向上し，OLAP 性能は一部向上したことから，ノード数を増やすことによる有効性を示した．

5.2 今後の課題

本研究では PostgreSQL の実行計画の中でも Index Scan に着目した OLAP エンジン選択手法を提案したが、データサイズやクエリの性質に着目した手法を検討することが今後の課題である．

また，動的コアアサイン手法として先行研究で提案された手法を基にアルゴリズムの再構築を行ったが、本研究での構成にも適用できる手法の再検討が必要である．

実験では OLTP client 数が 100、OLAP client 数が 20 の OLTP ヘビーなワークロードで処理性能を計測したが、負荷比率が異なるワークロードでの実験による性能評価が必要である．またデータサイズについても、実世界での使用を想定した場合の大規模データに対応することが望まれる。

本研究では OLAP エンジンの種類を導入の容易さから PostgreSQL と Spark SQL に限定して構成検討を行ったが，幅広い OLAP エンジンに対応した構成，動的コアアサイン手法，OLAP エンジン選択手法が求められる．

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] Michael Armburst, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, Matei Zaharia, “Spark SQL: Relational Data Processing in Spark” Processings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1383–1394, 2015.

[2] 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,” Proceedings of the Fourth International Workshop on Testing Database Systems, Vol. 8 pp. 1-6, 2011.

[3] Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, Philippe CudréMauroux, “OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases,” Proceedings of the VLDB Endowment, Vol. 7, No. 4, pp. 277–288, 2013.

[4] Grégory M. Essertel, Ruby Y. Tahboub, James M. Decker, Kevin J. Brown, Kunle Olukotun, Tiark Rompf, “Flare: Native Compilation for Heterogeneous Workloads in Apache Spark,” arXiv.1703.08219, 2017.

[5] Alfons Kemper, Thomas Neumann, “Hyper: A hybrid oltp olap main memory database system based on virtual memory snapshots,” 2011 IEEE 27th International Conference on Data Engineering, pp. 195–206, 2011.

[6] Norman May, Alexander Böhm, Wolfgang Lehner, “Sap hana – the evolution of an in-memory dbms from pure olap processing towards mixed workloads,” Datenbanksysteme für Business, Technologie und Web, pp. 545–546, 2017.

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

[8] Takamitsu Shioi, Takashi Kambayashi, Suguru Arakawa, Ryoji Kurosawa, Satoshi Hikida, Haruo Yokota, “Serializable HTAP with Abort-/Wait-free Snapshot Read,” arXiv:2201.07993, 2022.

[9] TPC, TPC–C Benchmark, https://www.tpc.org/tpcc/

[10] TPC, TPC–H Benchmark, https://www.tpc.org/tpch/

[11] 御喜家 奈波，諸岡 大輝，Le Hieu Hanh，横田 治夫，“Spark SQL と PostgreSQL での処理に適したクエリ特徴の考察，” 第 13 回データ工学と情報マネジメントに関するフォーラム, B25-2, 2021.

[12] 諸岡 大輝，塩井 隆円，引田 諭之，Le Hieu Hanh，横田 治夫，“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，” 第 12 回データ工学と情報マネジメントに関するフォーラム， H2-2, 2020.

[13] 諸岡 大輝，塩井 隆円，引田 諭之，Le Hieu Hanh，横田 治夫，“メニーコア環境での HTAP 実現に向けた OLTP 性能とデータ鮮度重視の動的負荷分散，” 日本データベース学会和文論文誌， Vol. 20-J, Article No. 8, 2022.