Spark SQL と PostgreSQL での処理に適したクエリ特徴の考察

御喜家奈波† 諸岡 大輝† LeHieu 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,morooka,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，近年，フレッシュなデータに対する分析処理を可能にする HTAP に注目が集まっている．一方で OLAP エンジン単体でも，リレーショナルデータベースに対して並列処理をすることでデータを迅速に処理する Spark SQL が注目されている．しかし従来のクエリ処理エンジンと比較した時に，すべてのクエリにおいて Spark が高い性能を出すというわけではない．本研究では，HTAP 上において複数の OLAP エンジンによる処理を考慮したクエリ分類法の提案を目的とし，CH-benCHmark を用いて Spark SQL と PostgreSQL によるクエリ処理時間の計測を複数のデータサイズ上で行う．計測結果に基づいて Spark SQL と PostgreSQL のそれぞれに適したクエリの特徴を考察する．

キーワード HTAP，Spark SQL，PostgreSQL，並列データ処理

1 はじめに

本研究は，HTAP(Hybrid transactional/analytical processing) 上において複数の OLAP (OnLine Analytical Processing) エンジンの中から最も効率的にクエリを処理できるエンジンを選択するシステムの構築を目指して，クエリ特性やデータサイズが実行時間にどのように影響するのかを調べることで，クエリ分類法を提案することを目的とする．

1.1 研究背景

近年，OLTP (OnLine Transaction Processing) のトランザクション処理における更新内容を，OLAP にできるだけ瞬時に適用することで，フレッシュなデータに対する分析処理の実行を可能とする HTAP に注目が集まっている．OLAP エンジン単体でも，企業の戦略的な意思決定を支援することができる分析処理の需要は高く，近年リレーショナルデータベースに対して並列処理をすることでデータを迅速に処理する Spark SQL [1] が注目されている．一方で従来のクエリ処理エンジンの中でも，オープンソースのデータベース管理システムとして分析処理を含めて多用されている PostgreSQL [2] は，拡張機能を記述するためのサポートがあるため様々なシステムに採用されている．Spark SQL と PostgreSQL の性能を比較する研究は複数行われてきた [3], [4] が，これらの実験結果より，すべてのクエリにおいて Spark SQL が高い性能を出すというわけではない．

そこで本研究では，Spark SQL と PostgreSQL の計測実験を行い，様々なクエリがどちらのクエリ処理エンジンに適しているのか分類することを目的とする．TPC-C に基づくトランザクションワークロードと TPC-H と同等の OLAP クエリを並列実行することができる CH-benCHmark [5], [6] を用いて，96 論理コアのマシン上で Spark SQL と PostgreSQL によるクエリ処理時間の計測を複数のデータサイズで行う．計測結果に基づいて，各クエリに対してデータサイズや各 OLAP エンジンが生成するクエリプランがどのように影響するのかを調べることで，Spark SQL と PostgreSQL のうちどちらのクエリ処理エンジンを使えばより効率的に処理できるのか考察する．

1.2 本稿の構成

本稿は以下の通りに構成される．2 節では背景知識について説明し，3 節では関連する研究について述べる．4 節では本研究における手法を提案する．5 節では実験環境，実験内容について述べ，実験結果から考察を行う．最後に 6 節で本稿の結論を述べる．

2 背景知識

2.1 Spark SQL

リレーショナルクエリと複雑な手続き型アルゴリズムをシームレスに組み合わせた，Apache Spark のモジュールの 1 つである．Spark SQL の特徴の 1 つにデータの並列処理が挙げられる．Spark SQL は JDBC を使用して他のデータベースからデータを読み込むことができ，その際にパーティションを設定することで DataFrame を複数に分割して並列処理することが可能になる．パーティション設定には，分割の基準となるカラム名，そのカラム内要素の最小値，最大値，パーティション数を指定する必要がある．本研究ではこの JDBC を利用して，PostgreSQL で管理するデータベースを読み込み使用する．

2.2 PostgreSQL

カリフォルニア大学バークレー校で POSTGRES プロジェクトとして実装が始まり，1996 年に PostgreSQL として正式にリリースされた．その後も約 30 年に渡って，機能の追加やバグ修正などが行われ続けている，オープンソースのリレーショナルデータベース管理システムである．オープンソースのソフトウェアのため導入時の利用料がかからず，データベース内の拡張機能サポートが優れており数多くのオープンソースソフト

ウェアと連携できるため，様々なシステムに使われている．その拡張性に加えて，OLTP, OLAP ワークロードの両方において機能豊富であり，標準 SQL に準拠しているためデータ管理に適している．

PostgreSQL と同じオープンソースデータベースである MySQL と比較されることもある．MySQL では JOIN 方式が Nested Loop Join のみのサポートであるのに比べて，PostgreSQL では Nested Loop Join の他に Hash Join, Merge Join をサポートしている．Nested Loop Join は結合対象のデータが小さい場合に適しているのに対して，複雑で大量なデータを扱う場合には Hash Join や Merge Join が向いているため PostgreSQL が適している [7].

2.3 CH–benCHmark

OLTP のための TPC-C と OLAP のための TPC-H というそれぞれ確立されたワークロード間のギャップを埋めるために開発された，複雑な混合ワークロードを実行するベンチマークである [5], [6].

OLTP システムはトランザクション指向のアプリケーションにサービスを提供しており，多数のユーザーが広く使えることと低いレイテンシに重点を置いている．一方で OLAP システムは戦略的な意思決定をサポートするため，大量のデータを分析する．その中でも製品やサービスを管理・販売・配布しなければならない小売業者をモデルとしている TPC–C と TPC–H を CH–benCHmark では組み合わせている．これら 2 つのベンチマークはそれぞれ異なるスケーリングモデルに従うが，CH–benCHmark ではパフォーマンスに関係なくデータベースサイズが SF (Scale Factor) で設定される TPC–H のスケーリングモデルを適用している．トランザクションは TPC–C に準拠した 5 つのトランザクションを，クエリは TPC–H の 22 個のクエリを CH–benCHmark に適応させて使用している．

3 関 連 研 究

3.1 HTAP

複数コア環境での HTAP 性能を向上させる研究, [8], [9] がされており，処理内容に応じて実行コアを指定するコアアサイン手法を用いて 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で，OLAP 対象データのフレッシュネス向上を図っている．しかしこの手法では OLTP の更新を OLAP へ適用する更新適用性能を重視しており，OLAP の高速処理が困難である．単一のエンジンで OLTP と OLAP の両方をサポートする HTAP システムは多く存在するが，本研究では OLAP のみに注目し，OLAP 性能を向上させることを目標とする．

3.2 複数 OLAP エンジンの性能評価

Flare [4] はクエリ実行時間の大幅な高速化を実現するために開発された Spark のバックエンドである．Spark SQL のクエリオプティマイザから取得できるクエリプランをネイティブコードにコンパイルし，非効率的な抽象化レイヤーを回避することで実行時間の高速化を実現している．開発者の Essertel らは Flare の性能を調べるためにリレーショナルワークロードと機械学習カーネルで実験を行った．リレーショナルワークロード上での実験では，PostgreSQL, Spark SQL, HyPer, そして Flare の性能を比較するために SF10 のデータベースを用いて TPC–H ベンチマークを実行し，それぞれデータの読み込み時間を除いた実行時間を比較している．

この実験結果より PostgreSQL と Spark SQL を比較すると，Spark SQL の性能はほとんどのクエリで PostgreSQL の性能に匹敵しているが，全てのクエリで Spark SQL の方が優れているわけではない．また，各 OLAP エンジンがどのようなクエリの処理を得意としているのかについての考察は行われていない．この課題点を解決するために，数ある OLAP エンジンの中から本稿では導入が容易な Spark SQL と PostgreSQL の 2 種類の OLAP エンジンに絞り，クエリを実行した際のクエリプランを参照することで各 OLAP エンジンが得意とするクエリの特徴を考察する．本稿では，第一段階として異なる OLAP エンジンでのクエリプランの違いを見るために 2 つの OLAP エンジンに絞るが，今後，同様のアプローチを他の OLAP エンジンにもできるか検討を進めていく．

4 提 案 手 法

本研究では HTAP の OLAP 側で入力された OLAP クエリに対して，複数の OLAP エンジンの中から最も効率的にクエリ処理できるエンジンを選択することで迅速に分析結果を出力することができる，図 1 のようなシステムの構築を最終目標とする．OLTP 側では PostgreSQL がデータの挿入，削除，更新を担い，随時 OLTP 更新の適用を OLAP 側に行う．OLAP 側には複数の OLAP エンジンが用意されており，入力される OLAP クエリを分類器で解析することでクエリに適したエンジンを選択する．

この分類器作成の切り口として，本節ではクエリを Spark SQL と PostgreSQL のどちらのエンジンで処理されるべきか選択するためのクエリ分類法を提案する．このシステムは HTAP を前提としているため，OLAP に対する OLTP 更新の適用を考慮する必要があるが，本稿では OLTP 更新適用は考慮せず，OLAP エンジンの性能評価と各 OLAP エンジンに適したクエリ分類の考察を行う．

前節を踏まえて，各 OLAP エンジンでの処理に適したクエリ特徴を分類する手法として次の 2 点に着目する．

4.1 データサイズ

一般的にデータサイズが増加すると処理しなければならない対象データも増えるため，クエリ処理にかかる時間も増加する．データサイズと実行時間が比例する場合，データサイズに関わらず Spark SQL に適したクエリ，もしくは PostgreSQL に適したクエリは一定である．一方で，データサイズの増加と実行時間が比例しない場合，データサイズによって各 OLAP エンジンに適したクエリは変化するため，データサイズを考慮したクエリ分類が必要になる．データサイズと各 OLAP エンジン

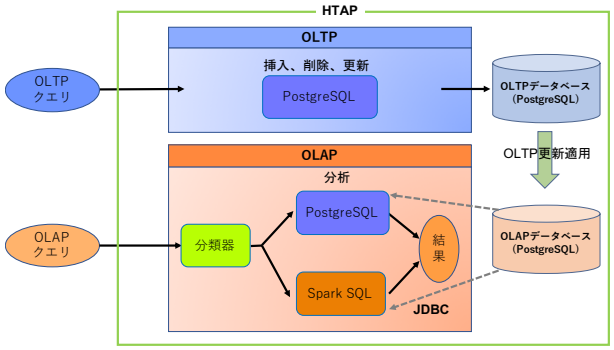

図 1　想定するシステム

での実行時間の関係を調べるため，Spark SQL と PostgreSQL のそれぞれで複数のデータサイズのデータベースを用いて CH-benCHmark の OLAP クエリを実行し，それぞれのクエリ処理時間を計測する．

4.2　クエリプラン

OLAP エンジンはクエリを受け取ってから実行するまでに，SQL 解析やクエリ最適化を行う．Spark SQL の場合，Catalyst Optimizer というフレームワークによってクエリを解析，最適化し，並列実行するための Physical Plan を生成する．PostgreSQL の場合にも，クエリの構造とデータの性質に適したクエリプランを選択するためのプランナが存在する．これらのクエリプランは，同じクエリを実行した場合でも OLAP エンジンによって異なる．全体のクエリ実行時間は処理に時間がかかるクエリ演算子の数が影響すると考えられるため，各 OLAP エンジンが生成したクエリプランのクエリ演算子の出現回数に着目して，この回数と実行時間に関係があるか調べる．具体的には，Spark SQL と PostgreSQL 両方のクエリプランに共通して存在する主要なクエリ演算子の出現回数を数える．この回数の多寡から，Spark SQL と PostgreSQL のどちらで処理した方が好ましいのかクエリを分類する．

5　実　　験

5.1　実 験 環 境

表 1 に本実験で用いるマシンの環境を示す．

Spark SQL の環境，主要な設定

- バージョン：3.0.1
- Java バージョン：1.8.0

PostgreSQL の環境，主要な設定

- バージョン：12.3
- shared_buffers：30 GB
- work_mem：1 GB
- max_connections：300
- max_wal_size：16 GB

Spark SQL のクエリ実行と計測については Python による独自実装をする．JDBC を通して PostgreSQL で管理しているデータベースを読み込み，sparkmeasure を用いて処理時間の計測を行う．PostgreSQL では oltpbench [10] を用いること

表 1　実 験 環 境

CPU(s)	インテル® Xeon ® Platinum 8260 プロセッサー
Thread(s) per core	2
Core(s) per socket	24
Socket(s)	2
NUMA node(s)	2
CPU MHz	2.4 GHz
L1d cache	32 KB
L1i cache	32 KB
L2 cache	1,024 KB
L3 cache	36,608 KB
memory	384 GB

でクエリの実行と処理時間の計測を行う．データベースは 1.5 TB の NVDIMM に格納し，PostgreSQL で管理する．

5.2　実 験 内 容

Spark SQL と PostgreSQL それぞれで CH-benCHmark の OLAP クエリのみを SF100, SF200, SF300, SF400, SF500, SF1000（それぞれ 10 GB, 20 GB, 30 GB, 40 GB, 50 GB, 100 GB）の計 6 種類のデータサイズ上で実行して，クエリ処理時間の計測を行う．Spark SQL は PostgreSQL で管理しているデータベースを JDBC を使用して読み込み、この読み込みにかかる時間もクエリ処理時間に含めて計測する．この時，DataFrame を分割して並列処理するためのパーティション数は，SF と同じ値に設定する．PostgreSQL にも複数の CPU を使用することでクエリ処理時間を短くするパラレルクエリがあるが，今回の実験では使用しない．Spark SQL，PostgreSQL のそれぞれで 4 回ずつ計測を行い，それらの平均値を計測値として扱う．各回実行前には，Spark SQL の場合は OS のメモリキャッシュを削除し，PostgreSQL の場合にはデータベースの shared_buffer の削除と OS のメモリキャッシュ削除を行う．

5.3　予 備 実 験

Spark SQL は JDBC を通して PostgreSQL で管理しているデータベースを読み込む際にパーティション数を指定して実行することができるため，各スケールファクターに適したパーティション数を設定することができる．このパーティション数を決定するため予備実験では 5.1 節で説明した実験環境下において，SF100, 200, 300, 400, 500, 1000 と複数のデータサイズで実験を行う．

5.3.1　実 験 内 容

Spark SQL で CH–benCHmark の OLAP クエリのみを実行してクエリ処理時間の計測を行う．SF200 のデータベースに対してはパーティション数を 200, 500 に設定し，SF500 のデータベースに対してはパーティション数を 200, 500, 1000 に設定してそれぞれ実行する．それぞれ 4 回ずつ計測を行い，それらの平均値を計測値として扱う．各回実行前には OS のメモリキャッシュ削除を行う．

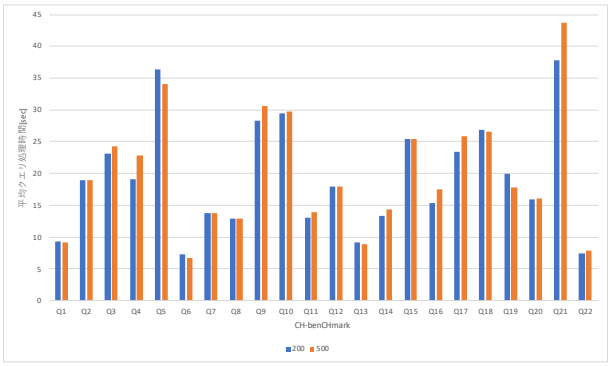

図 2　SF200 のデータベース上でパーティション数を変更した時の平均クエリ処理時間

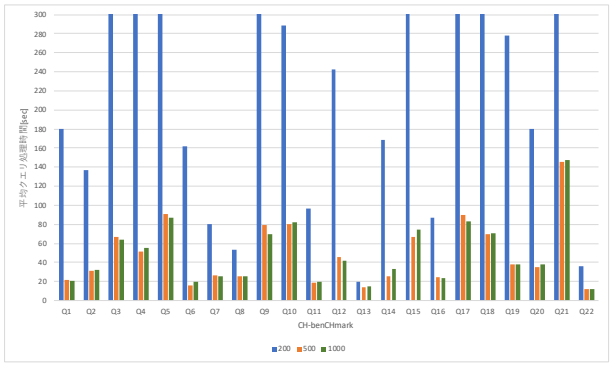

図 3　SF500 のデータベース上でパーティション数を変更した時の平均クエリ処理時間

5.3.2　実験結果

SF200,500 のデータベースでそれぞれ複数のパーティション数に設定して CH–benCHmark クエリを実行した際の平均クエリ処理時間を図 2,3 に表す．ただし，クエリ処理時間が 300 秒を超えた時点で計測を中断している．

SF200 のデータベースに対してパーティション数を 200 と 500 に設定してそれぞれクエリを実行した場合，クエリによってどちらの設定がより速く処理できるかは異なり，クエリ処理時間の差も高々 6 秒である．これよりどちらか一方に優位性は見られない．

SF500 のデータベースに対してパーティション数を 200 に設定してそれぞれクエリを実行した場合，多くのクエリが 300 秒以内にクエリを処理できなかった．一方でパーティション数を 500 に設定した場合，全てのクエリにおいてパーティション数 200 でかかった時間の 2 分の 1 以下でクエリを処理できた．パーティション数を 1000 に設定した場合のクエリ処理時間は，500 に設定した時の結果と比較するとクエリによってどちらがより速く処理できるかは異なり，どちらか一方に優位性は見られない．

これらの結果より，Spark SQL でデータを外部のテーブルから読み込む際のパーティション数はスケールファクターと同数に設定した場合に並列実行の効果が処理時間に現れ，スケールファクターより大きい値に設定してもクエリ処理時間には影響を与えないことが示された．本実験ではパーティション数をスケールファクターと同数に設定して計測を行う．

5.4　実験結果

PostgreSQL と Spark SQL をそれぞれ 6 種類のデータサイズ上で実行した際の平均クエリ処理時間を図 4,5 に示す．ただし，PostgreSQL において 300 秒を計測時間の上限とし，クエリ処理時間が 300 秒を超えたものは全て 300 秒として記録している．各データサイズで Spark SQL に有利なクエリを表 2 に示す．これらの計測結果を基に考察を行う．

5.5　考　　察

実験から得られた Spark SQL, PostgreSQL それぞれでのクエリ実行時間の計測結果を用いて，データサイズを変化させた時の実行時間の変化，そしてクエリプランからわかるクエリ特徴を考察する．

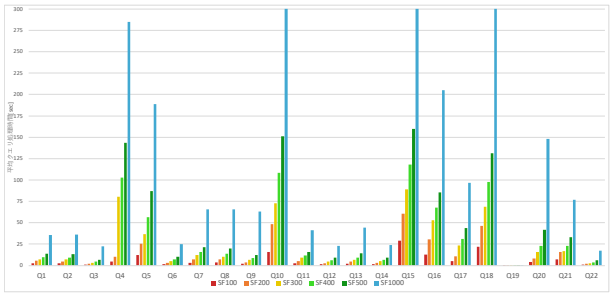

図 4　データサイズを変更した時の PostgreSQL の平均クエリ処理時間

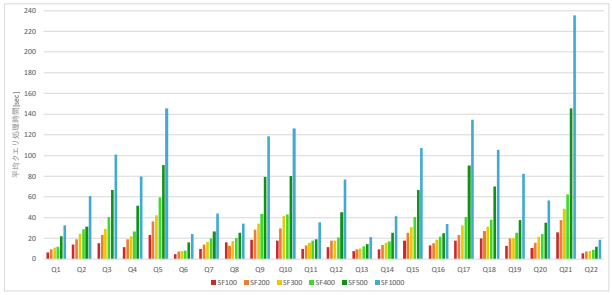

図 5　データサイズを変更した時の Spark SQL の平均クエリ処理時間

表 2　各データサイズで Spark SQL の方がクエリ処理が速かったクエリ

SF100	Q15, Q18
SF200	Q10, Q15, Q16, Q18
SF300	Q4, Q10, Q15, Q16, Q18
SF400	Q4, Q10, Q15, Q16, Q18
SF500	Q4, Q10, Q13, Q15, Q16, Q18, Q20
SF1000	Q1, Q4, Q5, Q6, Q7, Q8, Q10, Q11, Q13, Q15, Q16, Q18, Q20

5.5.1　データサイズ

各データサイズ，各クエリでの PostgreSQL の平均実行時間を 1 に固定した時の Spark SQL の平均実行時間の割合を表したグラフが図 6 である．Q19 のみデータサイズの上昇に伴い，PostgreSQL での平均クエリ処理時間に対する Spark SQL で

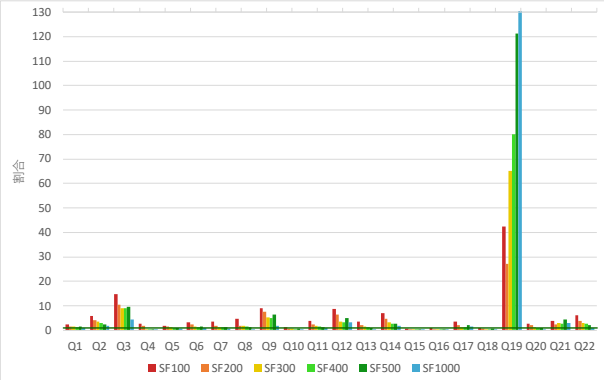

図 6　PostgreSQL の実行時間を 1 に固定した場合の Spark SQL の実行時間の割合

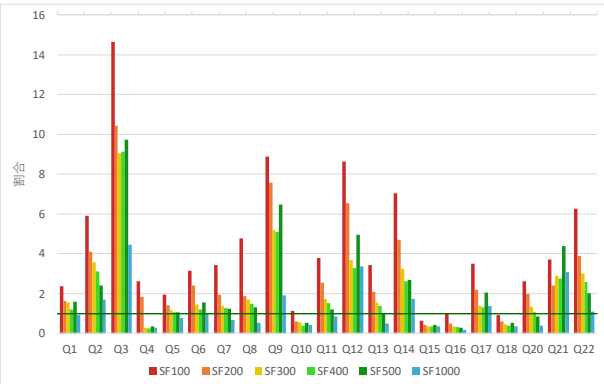

図 7　PostgreSQL の実行時間を 1 に固定した場合の Spark SQL の実行時間の割合 (Q19 を除いたもの)

の平均クエリ処理時間の割合が大幅に増加するため，Q19 の結果のみを除いたグラフが図 7 である．Spark SQL の割合が 1 より小さい場合は，PostgreSQL と比べて Spark SQL の実行時間が短いことを表し，割合が 1 より大きい場合は，PostgreSQL よりも実行時間が長いことを表す．

図 6,7 より，多くのクエリではデータサイズの上昇に伴い PostgreSQL に対する Spark SQL の平均クエリ処理時間が減少する．表 2 からもデータサイズが増えるほど Spark SQL での処理に適したクエリは増加し，あるデータサイズで Spark SQL に有利なクエリはそれより大きいデータサイズでも Spark SQL に有利である．これらより多くのクエリでは，データサイズが大きいほど Spark SQL での処理に適しており，SF1000 の時点では PostgreSQL の処理性能の方が高いクエリでもデータサイズを大きくした場合に Spark SQL に有利なクエリに切り替わる可能性がある．

一方で Q19 を始めとするいくつかのクエリは，データサイズの上昇に伴い PostgreSQL に対する Spark SQL のクエリ実行時間も増加，あるいは減少しない．これらのクエリは，データサイズを SF1000 よりも大きくしたとしても PostgreSQL に有利なクエリである可能性が高い．

5.5.2　クエリプラン

クエリプランで使用される主要なクエリ演算子をソート演算子，集計演算子，結合演算子，スキャン演算子の 4 種類に分類する．クエリプランはデータサイズに依存しないと仮定して，SF200 のデータサイズ上で CH–benCHmark の OLAP クエリを実行した際に，Spark SQL と PostgreSQL のクエリプラン内で 4 種類のクエリ演算子がそれぞれ何回実行されているのかを調べる．各 OLAP エンジンでそれぞれどのようなクエリ演算子が使用されているかを表 3 に示す．

ある SF の大きさで Spark SQL の方が平均クエリ処理時間が速いクエリは，それより大きい SF でも Spark SQL が優位である．この性質を利用して，図 8 に示すように CH–benCHmark の OLAP クエリを次のように色分けした．

- SF100 以上で Spark SQL に優位なクエリ：赤
- SF200 以上で Spark SQL に優位なクエリ：橙
- SF300 以上で Spark SQL に優位なクエリ：黄
- SF400 以上で Spark SQL に優位なクエリ：黄緑
- SF500 以上で Spark SQL に優位なクエリ：緑
- SF1000 以上で Spark SQL に優位なクエリ：水色
- SF1000 以下では PostgreSQL に優位なクエリ：青

ただし，本研究では SF1000 までのデータベースで実験を行ったため，「SF1000 以下で PostgreSQL に有利なクエリ」には SF1000 よりも大きいデータサイズで Spark SQL に優位に切り替わる可能性があるクエリが含まれる．

SF200 のデータサイズ上で CH–benCHmark の OLAP クエリを実行した際の集計データを視覚的に捉えるため，クエリ演算子の種類毎に各クエリとクエリプラン内の演算子数の関係を示すグラフを作成する．

4 種類に分類したクエリ演算子が各クエリのクエリプランでそれぞれ計画された回数を図 9〜12 にまとめた．各点は CH–benCHmark の 22 個のクエリを表している．Spark SQL と PostgreSQL でのクエリ演算子の回数が同じクエリが複数ある場合，より小さいデータサイズで Spark SQL に有利なクエリの色を優先し，隠れているクエリに関しては括弧内に記している．

図 9 より，Spark SQL のクエリプラン内でのソート演算子数は PostgreSQL のものよりも多いクエリが多い．これは，Spark SQL では結合の際に SortMergeJoin が使用され，1 回の結合につき大抵 2 回ソート作業が行われるためである．

図 10 より，多くのクエリプランで集計演算子が行われる回数は重複しており，クエリによって集計が行われる回数は大きく変わらない．これより集計演算子の合計回数からクエリ特徴を見出すことは難しい．

多くのクエリでは各クエリで使用するテーブル数と同程度の回数スキャンが行われる．このため，Spark SQL と PostgreSQL それぞれのクエリプランで出現する回数が等しいクエリが多いことが図 12 よりわかる．

Spark SQL に適したクエリの特徴

ここでは，データサイズが 300 以下の場合に Spark SQL でのクエリ処理時間の方が速いクエリ，言い換えれば赤色，橙色，黄色に分類されるクエリを Spark SQL に適したクエリと定める．Spark SQL に適したクエリの集合を Slike とおくと，

$$\mathrm{Slike} = \{\mathrm{Q4}, \mathrm{Q10}, \mathrm{Q15}, \mathrm{Q16}, \mathrm{Q18}\}$$

表 3　各 OLAP エンジンのクエリ演算子

クエリ演算子の種類	Spark SQL	PostgreSQL
ソート	Sort	Sort
集計	HashAggregate	Aggregate GroupAggregate HashAggregate
結合	SortMergeJoin	NestedLoopJoin MergeJoin HashJoin
スキャン	Scan JDBC Relation	SeqScan IndexScan IndexOnlyScan BitmapIndexScan BitmapHeapScan

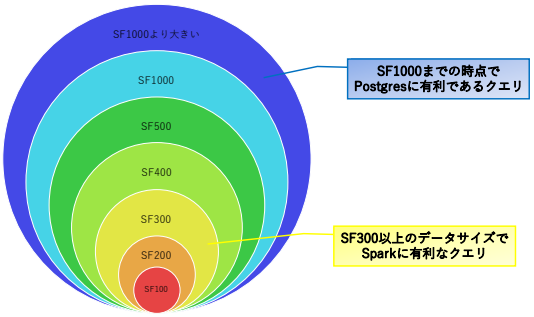

図 8　Spark SQL が有利になる SF による色分け

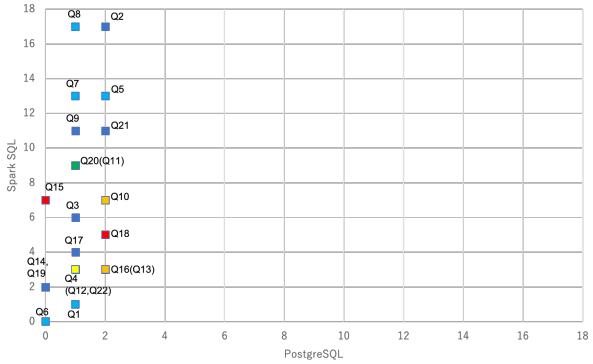

図 9　Spark SQL, PostgreSQL それぞれのクエリプラン内でのソート演算子の数

各クエリと結合演算子の数の関係を表す図 11 より，グラフの左下の部分，具体的には Spark SQL のクエリプランでの結合演算子が 1 以上 3 以下の範囲（この範囲を S とおく）には，Slike のクエリが集まっている．しかしこの範囲には PostgreSQL に適したクエリも複数存在する．範囲 S 内にある PostgreSQL に適したクエリの集合を Plike_{fj} (fj は，with a few join operations を意味する) とおくと，

$$\text{Plike}_{\text{fj}} = \{\text{Q3}, \text{Q12}, \text{Q13}, \text{Q14}, \text{Q17}, \text{Q19}, \text{Q22}\}$$

この範囲 S 内にあるクエリの内，Spark SQL に有利なクエリのみが持つ特徴，もしくは PostgreSQL に有利なクエリのみが持つ特徴が見つけられれば，範囲 S から Plike_{fj} を除外し Slike のみを抽出することができる．

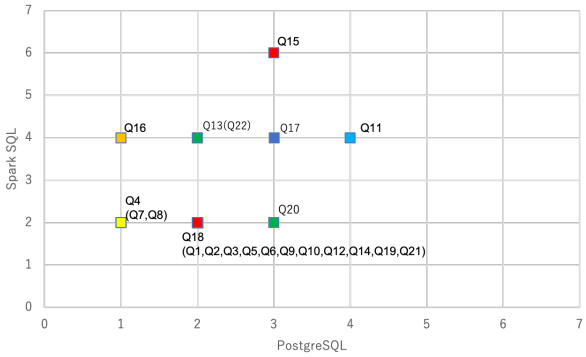

図 10　Spark SQL, PostgreSQL それぞれのクエリプラン内での集計演算子の数

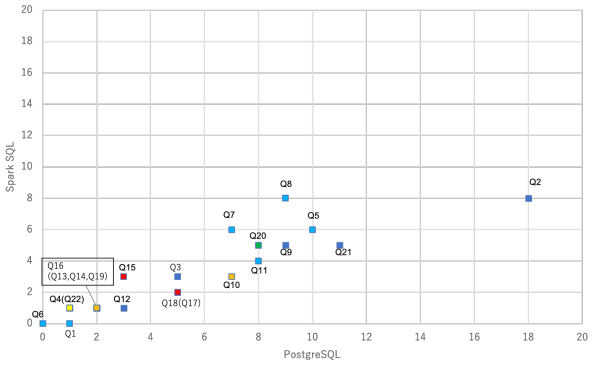

図 11　Spark SQL, PostgreSQL それぞれのクエリプラン内での結合演算子の数

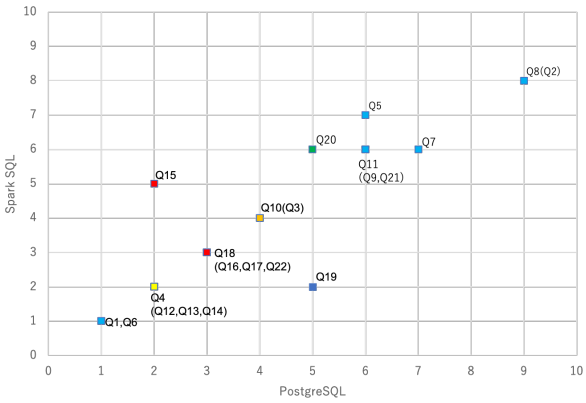

図 12　Spark SQL, PostgreSQL それぞれのクエリプラン内でのスキャン演算子の数

ここでソート演算子の数に注目すると，Spark SQL でのソート回数が 2 以上 6 以下，かつ PostgreSQL でのソート回数が 0 回, 1 回の範囲（この範囲を P とおく）には Plike_{fj} のクエリが集まっている．範囲 S 内にあるクエリと，P 内にあるクエリを比較すると表 4 のようになる．これよりソート回数を Plike_{fj} のクエリのみが持つ特徴として，範囲 S 内にあるクエリから PostgreSQL に有利なクエリを除外することで Spark SQL に適したクエリを残すことができる．

PostgreSQL に適したクエリの特徴

ここでは，データサイズが SF500 以下の場合に PostgreSQL

表 4　結合演算子とソート演算子の数に注目した際のクエリ

S 内にある Slike クエリ	Q4,Q10,Q15,Q16,Q18
S 内にある Plike_{fj} クエリ	Q3,Q12,Q13, Q14,Q17,Q19,Q22
P 内にあるクエリ	Q3,Q4,Q12, Q14,Q17,Q19,Q22
{S 内にあるクエリ }-{P 内にあるクエリ }	Q10,Q13,Q15,Q16,Q18

でのクエリ処理時間の方が速いクエリ，言い換えると水色，青色に分類されるクエリを PostgreSQL に適したクエリと定める．PostgeSQL に適したクエリの集合を Plike とすると，

$$\text{Plike} = \{Q1, Q2, Q3, Q5, Q6, Q7, Q8, Q9, Q11, Q12, Q14, Q17, Q19, Q21, Q22\}$$

各クエリとクエリ演算子の数の関係を示すグラフを視覚的に捉えると，ソート演算子，結合演算子，スキャン演算子の数が極端に少ない，または極端に多いクエリは PostgreSQL に適したクエリとなっている．具体的には，以下のいずれかに該当するクエリは PostgreSQL に適したクエリである．

- Spark SQL のクエリプランでのソート演算子の数が 2 以下，または 11 以上
- Spark SQL のクエリプランでの結合演算子の数が 0，または PostgreSQL のクエリプランでの結合演算子の数が 9 以上
- PostgeSQL のクエリプランでのスキャン演算子の数が 6 以上

それぞれの条件を満たすクエリは表 5 であり，Plike のクエリ 15 個の内，上記のいずれかの条件を満たすクエリは 11 個である．

PostgreSQL でのクエリプランに出てくるスキャンの内，インデックスを使用したスキャンに注目する．これらのスキャンでは，Index Cond という条件が設けられており，この条件を満たす場合のみインデックスを使用したスキャンが実行される．今回の実験で使用した CH–benCHmark の OLAP クエリの内，Q3, Q4, Q5, Q7, Q8, Q9, Q19, Q22 ではインデックスを使用したスキャンが計画されている．これらの中で，Q4 を除いた 7 つのクエリは PostgreSQL での処理に適した Plike なクエリである．ここで，PostgreSQL のクエリプランだけでなく実際にクエリが実行された時の処理時間等の情報を調べると，Q4 のみ Index Cond の条件を満たさず，実際の実行時にはインデックスを使用したスキャンが行われていないことがわかった．このことより，PostgeSQL でのクエリプランでインデックスを使用するスキャンが計画され，尚且つ，Index Cond を満たし実際の実行時にもこれらのスキャンが行われるクエリは，PostgreSQL での処理に適した Plike なクエリであることがわかる．

6　おわりに

6.1　まとめ

本稿では，CH-benCHmark の OLAP クエリを用いて Spark SQL と PostgreSQL におけるクエリ実行時間の計測実験を行った．計測結果に基づいて，各クエリに対してデータサイズ，各クエリ処理エンジンでのクエリプランがどのように影響するのかを調べることで，Spark SQL と PostgreSQL のうちどちらのクエリ処理エンジンを使えばより効率的に処理できるのか考察した．

表 5　PostgreSQL に適したクエリの条件

Spark SQL でのソート回数が 2 以下，または 11 以上	Q1,Q2,Q5,Q6,Q7, Q8,Q9,Q14,Q19,Q21
Spark SQL での結合回数が 0, または PostgreSQL での結合回数が 9 以上	Q1,Q2,Q5, Q6,Q8,Q9,Q21
PostgreSQL でのスキャン回数が 6 以上	Q2,Q5,Q7, Q8,Q9,Q11,Q21
上記のいずれかを満たすクエリ	Q1,Q2,Q5,Q6,Q7, Q8,Q9,Q11Q14,Q19,Q21

複数のデータサイズ上で CH–benCHmark の OLAP クエリを実行した際のクエリ実行時間計測結果より，多くのクエリではデータサイズの上昇に伴い，PostgreSQL に対する Spark SQL の相対的なクエリ処理時間が減少する傾向にある．これらのクエリは SF1000 の時点で PostgreSQL の処理性能の方が高いクエリであったとしても，データサイズを更に大きくした場合に Spark SQL に有利なクエリに切り替わる可能性がある．一方でデータサイズを上昇させると PostgreSQL の処理性能が高くなるクエリも一部あり，これらのクエリはデータサイズに関係なく PostgreSQL での処理に適していると考えられる．データサイズを上昇させた時の PostgreSQL に対する Spark SQL のクエリ処理時間割合の増減は，クエリにより異なる傾向があることがわかった．

クエリプランで使用される主要なクエリ演算子をソート演算子，集計演算子，結合演算子，スキャン演算子の 4 種類に分類し，SF200 のデータサイズ上で CH–benCHmark の OLAP クエリを実行した際に，Spark SQL と PostgreSQL のクエリプラン内で 4 種類のクエリ演算子がそれぞれ何回実行されているのかを調べた．結合演算子とソート演算子の数がそれぞれある一定の範囲内にあるクエリは Spark SQL での処理に適したクエリであった．ソート演算子，結合演算子，スキャン演算子の数が極端に少ない，または極端に多いクエリは PostgreSQL での処理に適したクエリとなっていた．これより，Spark SQL に適したクエリ，あるいは PostgreSQL に適したクエリの分類に，クエリプランで使用されるクエリ演算子の回数の閾値を使用できる可能性があることがわかった．

PostgreSQL でのクエリプランではスキャンが複数種類あり，その内のインデックスを使用したスキャンに注目すると，インデックスを使用したスキャンがクエリプランにおいて計画され，尚且つ，クエリ実行時に実際にインデックスを使用したスキャンが実行されるものについては，PostgreSQL での処理に適したクエリに分類できる可能性があることがわかった．

6.2 今後の課題

本稿では，OLAP エンジンはデータサイズに依存することなくクエリプランを生成すると仮定して考察を行った．Spark SQL で生成されるクエリプランについては，データサイズを変更してもクエリプランが変化することはなかった．しかし PostgreSQL で生成されるクエリプランについて，一部のクエリではデータサイズを変更するとクエリプランも異なるものが生成される．例として，Q11 のクエリプランは SF200 以下では図 13，SF300 以上では図 14 となる．ただし両方に共通する部分は省略してある．図 13,14 からわかるように，SF300 以上ではソート演算が追加されており，これはクエリプラン内で特定の演算子が計画された回数を表すグラフに影響する．このため，演算子の数から Spark SQL に適したクエリ，PostgreSQL に適したクエリを分別するためには，データサイズを変更した時のクエリプランの変化を考慮する必要がある．

クエリプランから各 OLAP エンジンに適したクエリを分類する際に，Spark SQL に適したクエリ・PostgreSQL に適したクエリを定義したが，目標とする分類器が実際に使用される環境を考慮した上でそれぞれの定義をする必要がある．

クエリプランから各 OLAP エンジンに適したクエリを分類する方法として提示した複数の条件の評価として，CH-benCHmark の OLAP クエリしか対象にしていない．CH-benCHmark 以外のクエリも用いて評価するとともに，それぞれの条件の論理的な根拠を解明し，より汎用的な分類法を目指す．

Spark SQL の並列分散処理という特徴を考えると，実験環境として分散環境でクエリ実行時間の計測実験を行い，PostgreSQL での実行時間と比較した上で各 OLAP エンジンに適したクエリを分類することが望ましい．

HyPer や Flare などのように性能の高い OLAP エンジンは数多く存在するが，本稿ではその中でも導入が容易な Spark SQL と PostgreSQL を用いてクエリ分類を試みた．しかし，クエリに適した OLAP エンジンを選択するための分類器作成による OLAP 性能向上を図るためには，実験対象とする OLAP エンジンを増やし，各クエリの処理により適したエンジンを追究することが必要である．

本研究の最終目標は，HTAP の OLAP 側で入力された OLAP クエリに対して分析結果を迅速に出力することができるシステムの構築である．本稿では OLAP エンジン性能評価のため，OLTP 更新適用を評価の対象としなかったが，OLTP から OLAP への更新適用を考慮した実験が必要がある．

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．

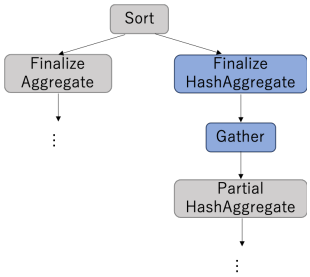

図 13　SF200 以下で PostgreSQL が生成する Q11 のクエリプラン

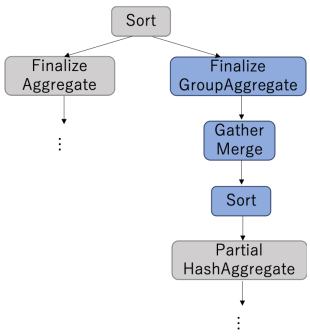

図 14　SF300 以上で PostgreSQL が生成する Q11 のクエリプラン

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] PostgreSQL, https://www.postgresql.org/

[3] Joshua Powers, “Apache Spark Performance Compared to a Traditional Relational Database using Open Source Big Data Health Software,” School of Computer Science Graduate Student Publications, 2016.

[4] Gregory 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 preprint arXiv:1703.08219, 2017.

[5] CH-benCHmark, https://db.in.tum.de/research/projects/CHbenCHmark/.

[6] Richard Cole, Florian Funke, Leo Giakoumakis, Wey Guy, Alfons Kemper, Stefan Krompass, Harumi Kuno, Raghunath Nambiar, Thomas Neumann, Meikel Poess, Kai-Uwe Sattler, Michael Seibold, Eric Simon, Florian Waas “The mixed workload CH-benCHmark,” Proceedings of the Fourth International Workshop on Testing Database Systems, No. 8, pp. 1–6, 2011.

[7] Let’s POSTGRES PostgreSQL の機能と他の RDBMS の比較，https://lets.postgresql.jp/documents/tutorial/rdbms-hikaku/

[8] 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.

[9] 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，2020.

[10] oltpbenchmark.oltpbench，https://github.com/oltpbenchmark/oltpbench.