カラムストアとローストアを利用したOLAP問合せ処理における消費電力と処理速度の関係について

福澤　優† 宮崎　純† 山本豪志朗† 加藤　博一†

† 奈良先端科学技術大学院大学　〒 630–0192 奈良県生駒市高山町 8916-5
E-mail: †tofirstplace@gmail.com, ††{miyazaki,goshiro,kato}@is.naist.jp

あらまし　デジタルデータの爆発的な増加が問題となってる一方で，蓄積されたビッグデータに対して分析を中心とした OLAP 処理の利用が広がっている．またビッグデータを扱うデータセンターにおいては消費電力量への懸念が抱かれている．これらを背景とし本稿では，これまで処理速度中心の改善を続けてきた DBMS において，OLAP 問合せに着目しローストアとカラムストアのハイブリッド構成の DBMS 利用した消費電力量の考慮を行っていく．このアプローチによって TPC-H ベンチマーク (ScaleFactor=20) のクエリ 5 においてハイブリッド構成で，実行時間がローストアの 74.9%，カラムストアの 23.4% 消費電力量ではローストアの 75.8%，カラムストアの 19.8% という結果が得られ，提案手法の有用性を示した．

キーワード　消費電力，OLAP，問合せ処理

On Relationship between Processing Time and Energy Consumption in OLAP Query Processing using Row and Column Stores

Yu FUKUZAWA†, Jun MIYAZAKI†, Goshiro YAMAMOTO†, and Hirokazu KATO†

† Nara Institute of Science and Technology
8916–5 Takayama, Ikoma, Nara, 630–0192 Japan
E-mail: †tofirstplace@gmail.com, ††{miyazaki,goshiro,kato}@is.naist.jp

1. はじめに

計算機の発展により，ムーアの法則にのっとり計算機の処理性能は大きく向上してきた．しかし計算機の処理性能向上と利用範囲の拡大の一方で，計算機によって消費される電力量が地球温暖化などに代表される環境問題や企業におけるコスト削減の需要から問題視されるようになり，計算機の発展においては処理速度の向上のみならず消費電力量の削減の需要が高まり，より高い処理速度を維持しながら，消費電力を削減するといった電力管理も考慮したシステムが求められている．

データベース管理システム (DBMS) は現在の情報化社会において大量のデータを扱うために多くのシステムの根幹を担っており，銀行のオンラインシステムや電子商取引などのシステムにおいては必要不可欠なソフトウェアである．近年では，情報化の発展により計算機で扱うデータ量の急速な増加が見られており米国 IDC が EMC の協賛によって発行した “Digital Universe“ [3] のレポートにおいては，2020 年におけるデータ量は 2010 年の 44 倍にあたる 35ZB(ゼタバイト) にもなると予測している．それに伴い，DBMS 上のビッグデータから有意義なデータの抽出を行ったり，その規則性の発見を目的としたデータマイニングやビジネスインテリジェンスなどのシステムがデータベースの分野で需要が高まっている．

またクラウドコンピューティング技術の発展により，機器導入のコストや消費電力の削減を目的として多くの企業においてクラウドコンピューティング技術の導入が進行している．その一方でクラウド化によってデータセンターに多くのデータが集約されることとなり，より一層ビッグデータに対する取り組みが注目されている．

本稿ではデータベースの処理の中でも，これらのビッグデータに対して行われる OLAP(OnLine Analytical Processing) の処理に着目し，従来の DBMS において最も重要とされてきた処理速度の向上だけではなく，消費電力量削減に対する問題を考慮した OLAP 問合せ処理手法の提案と評価を行う．処理手法においては，これまで OLTP(OnLine Transaction Processing) の処理を想定した DBMS において行われているローストアのデータ格納方法と，OLAP 処理において着目されてきたカラム

ストアのデータ格納方法をハイブリッドで利用することで，それぞれの利点を利用した柔軟な問合せ実行が行えると考えた．これによって複数の問合せ実行プランからユーザの実行時間または消費電力を削減したいというニーズに対応できるとも考えられる．

このハイブリッド構成のアプローチは，テーブルスキャン，選択演算，ならびに TPC-H(ScaleFactor=20) のクエリ 5 において，実行時間や消費電力量に対して有効であることを示す．

2. 関 連 研 究

従来からの PostgreSQL や MySQL などの RDBMS の設計においては，テーブルデータの格納方法にレコード単位でページに格納していくローストアという手法がとられてきた．一方，近年ではデータ量の爆発的な増加が伴う中で OLAP 処理に有効な DBMS の設計が考えられてきている．なかでもローストアに対して C-Store [6] に代表されるテーブルをカラム単位，つまり同じ属性データだけををまとめてページに格納していくカラムストア [2] のデータ格納が効果的とされており SybaseIQ, Vertica Analytic Database などの商用 DBMS においてもカラムストアの概念が取り入れられている．OLAP 処理においてカラムストア DBMS である C-Store が従来のローストアの DBMS と比べ多くの利点が挙げられている [1] がスタースキーマ構造を前提とし，評価は簡単な問合せにおける比較にとどまっている．このことから，より多くの特性をもつ問合せに対しての検証を行う必要性があると考える．

またローストアとカラムストアの両方を利用したハイブリッドな DBMS を構成することでクエリやデータの特性によるローストア DBMS とカラムストア DBMS のそれぞれの利点を利用する Fractured Mirrors [5] が提案された．Fractured Mirrors ではローストアとカラムストアのデータをミラーリングして格納する．このハイブリッドなデータ格納を利用した DBMS によってローストアまたはカラムストアのみでのクエリ実行よりも柔軟かつ有効なクエリ実行が可能になっている．

これらの研究はクエリ実行時間の高速化を目指して行われており，その評価も実行時間に対してのみ行われている．しかしデータセンターなどにおける大規模システムを中心とした背景に省電力化の需要が高まってくると，これまでのクエリ実行における処理速度の向上だけでなくクエリ実行時に消費される電力量のコストも抑えたクエリ最適化の考えが出てきた．文献 [7] においては，PostgreSQL で行われているさまざまなクエリプランの中から実行時間のコストパラメータによる最適化の手法に，新たに演算子ごとの消費電力のパラメータを付加させることにより消費電力と実行時間両方のコストパラメータを用いたコスト関数を用いてクエリ最適化が可能となるシステムを構築している．このシステムでは OLTP 処理中心の TPC-C ベンチマークにおいては実行時間を 2.5% 改善した上で，消費電力量も 19.0% 低減しているのに対して，OLAP 処理中心の TPC-H ベンチマークにおいては実行時間が 19.7% の低下する一方で，3.3% の消費電力量削減しか行えていない．また消費電力量に関して，ディスク 1 台の消費電力は CPU の消費電力に比べ十分に小さいとし，ディスクの消費電力量は考慮せず CPU の消費電力量の計測のみに留まっている．しかし，現在の OLAP などの処理を行うシステムでは複数のディスクを論理的に 1 つにまとめて利用するディスクアレイを構築するのが一般的であるため，このような構成においてはディスクの消費電力量も考慮すべきである．

本稿ではこの結果に対して OLAP 処理において，より効果的にクエリ実行時間を維持しながら低消費電力化を実現するために，これまで OLAP 処理で行われてきた実行時間高速化手法を用いながら消費電力量の低減を可能とするクエリ処理方法に関して考察する．

3. ローストアとカラムストア

ここでは DBMS におけるデータ格納について述べる．従来の RDBMS においてはローストアによるデータ格納方法が採用されてきた．これは電子商取引など OLTP 処理のようなタプルに対する更新が頻繁におこる処理を中心に考えられ設計されてきたからである．ローストアはリレーショナルデータベースのテーブルのレコードデータを 1 つの単位としてディスク上の各ページに格納していく格納方法である．

一方，データの増加による経営の意思決定支援システムなど OLAP 処理の需要が高まりカラムストアによるデータ格納方法が注目されてきた．カラムストアはリレーショナルデータベースのテーブルデータのカラムを 1 つの単位としてディスク上の各ページに格納していく格納方法である．OLTP 処理のような更新中心の処理においては，ある特定のタプルが更新の対象になる場合が多く，属性 A が a の値をとるタプルの属性 B の値を更新するなど，更新対象のタプルを特定して更新する処理が主となるのでタプル全体の情報が重要になることが多い．また重要なのは，OLTP 処理においてはトランザクションが処理の単位となることである．あるタプルをロックすることを考えるとローストアの方がロック粒度が小さくなることがいえる．このような理由から従来からの OLTP 処理を前提に設計されている DBMS ではローストアのデータ格納方法が採用されている．

OLTP 処理と異なり OLAP 処理においてはデータセットに対して，分析的な処理になるためデータアクセスのほとんどは読み込み処理のために行われる．このことから，OLTP のような占有ロックにおける粒度問題は発生しない．また分析の対象となるのがテーブル中の特定の属性データのみとなることが多い．このときローストアにおいてはレコード単位での格納であるため必要な特定の属性データ以外に，クエリでは利用しない属性データまでも読み込み，このことによってディスク I/O が多くなってしまう．これはレコード中の属性数が多ければ多いほど如実に現れる．現在の計算機の CPU 処理速度に対してディスク I/O 速度が大幅に遅いという特徴においては，このような多くのディスク I/O はクリティカルな問題となる．しかしカラムストアにおいてはカラム単位でのデータ格納が行われているため，特定の属性データの読み込む際には，他の不必要な属性データを読み込まずに処理が可能である．よって OLAP 処理においてはローストアに比べカラムストアの方が，計算機

上の処理における主なボトルネックとなっているディスク I/O の発生が低減できるために有効とされている．

4. システムの概要

4.1 システムの構成

本研究のシステムは [8] において，OLAP 処理においてローストア DBMS とカラムストア DBMS それぞれに対して処理速度と消費電力量の観点からの有効なクエリ特性が存在することが検証されているため，Fractured Mirrors のようにローストアとカラムストアのハイブリッドによるシステム構成を用いた．ハイブリッドのデータ格納にすることによって，演算を行うときに処理速度または消費電力量に対して効率的に行うことが出来る格納データに対して処理を行うことが可能となり，この問合せの柔軟さを利用することで処理速度と消費電力量の両方を考慮した問合せの実現を目指した．

またクエリ処理の柔軟性だけではなく，ローストアとカラムストア両方に重複するデータをそれぞれ異なるディスク上に保持する，つまりローストアとカラムストア間でのデータのミラーリングをすることにより，障害によってあるデータの参照が困難になったときに，異なるディスクに存在するもう一方のデータ格納方法のデータを参照することによって容易に復元できるという信頼性に関わる効果もうまれる．

一方ミラーリングすることによって，クエリ特性によっては片方のディスクへアクセスが集中してしまうという問題も考えられる．これに対してはそれぞれのデータ格納方法を水平分割し，ローストアとカラムストアがそれぞれのディスクにおいて同じ割合で配置するようなディスク上のデータ配置を実現すればディスクアクセスの偏りは解消されると考える．

さらにはミラーリングを行うことによりデータの整合性が損なわれることが考えられる．これはデータに対する更新処理が行われたとき，ローストアとカラムストアの両方のデータに対して更新処理を行う必要性が生じるためである．このことから，ローストアデータとカラムストアデータが常に一致するとは限らず，クエリを同時実行する場合においては両方のデータ格納データに対して整合性を保証した同時実行制御を行う必要性が生じる．しかし，本研究で対象としている処理は意思決定支援システムのような OLAP 処理であるため，データに対しては読み込み処理が中心である．また意思決定支援システムなどにおけるデータの更新は OLTP データベースからのまとまったデータの挿入処理が定期的に発生するのみである．このことから同時実行制御におけるローストアとカラムストアのデータの整合性の保証は厳密には行わなくてもよく，OLTP データベースなどからのデータ挿入処理のみ，読み込み中心の問合せ処理から独立して行えばよいと考える．よって本研究ではデータに対して読み込みを行う問合せ処理のみを考える．

またローストアとカラムストアでデータのミラーリングを行うことによって，単純に考えるとローストアまたはカラムストアのどちらかのデータ格納を用いた場合と比べ，格納するべきデータ量が 2 倍になることがいえる．これによってデータを格納するのに必要になるディスクも 2 倍となると考えられ，消費電力量の観点のみからであればローストアとカラムストアの間でのミラーリングは消費電力量の増加の要因となることが考えられる．しかし，データベース管理システムにおいてはデータベース上のデータがディスク故障によって破壊されることは多くの業務で多大な損失を生じる可能性があるために，処理に必要なディスクとは別のディスクにデータをバックアップをとっておくことが考えられる．これによってディスクに対する障害が発生したときには，バックアップをとっておいたディスクよりデータを復元し稼働を再開させる．さらにデータを復元する際にも，システム全体の稼働率を下げないためにバックアップデータが格納されているディスクをそのまま用いることが一般的である．また，障害が起きたときの更なるバックアップの作成時間に時間を割かないためや，正常時に想定されている以外のバックアップによって，システムがアクセスするためのディスクに余計な I/O を発生させてシステムの応答時間を低下させないためにも，一般的には 3 重以上でデータのバックアップをとることが行われる．だが，これらのバックアップを行うディスクは正常時においては電源を落とすことで，電力消費が行われないことも考えられる，これに対して本研究で提案するローストアとカラムストアでデータのミラーリングではあるディスクに障害が起きたとしても，もう一方のデータ格納を参照することによって，障害時における復帰時間は実質 0 秒とすることができ，先ほどのバックアップディスクに電力を供給していない場合に対して稼働率は高いシステムを構成できると考えられる．ここではディスク電力と稼働率においてトレードオフが発生するが，本研究のアプローチではディスクの消費電力量はどちらかのデータ格納を用いる場合に対して犠牲になるが，データベース処理であることを考え稼働率を上げ，さらにミラーリングによって処理全体のコストを下げることが出来ることを優先的に考えた．また，一般的なデータベース管理システムにおいては大きなボトルネックとなる I/O のレイテンシを短くするために，ディスクを余分に追加してまでディスクアレイを構成するのは一般的であるため，ローストアとカラムストアのミラーリングによりディスクの追加は 2 倍ほど多くならないことが考えられる．さらに，同じ問合せにおいてはハイブリッド構成になってもディスクに対するアクセス量は変わらないと考えられ，アクセスのない時間はアイドル時間となっていると思われる．これらのことから，ハイブリッド構成によってローストアまたはカラムストアのどちらかのデータ格納を用いた場合に対して単純に消費電力量が 2 倍になるわけではないといえる．

4.2 実　　装

システムの実装においては，まず研究の最終的な評価対象として TPC-H ベンチマークを想定し，システム上で扱うデータ型としては TPC-H ベンチマークで扱われているデータ型を基準として実装を行っている．また，ローストアとカラムストアのそれぞれのデータ格納を Key-Value ストア型の組み込みデータベースである BerkeleyDB を利用することによって実現した．ローストアでは 1 組の Key-Value をリレーショナルテーブルの 1 つのタプルに対応して格納した（図 1）．

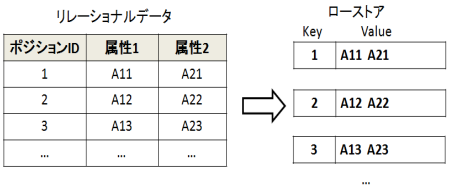

図 1　ローストアの実装

Key にはリレーショナルテーブルにおける格納タプルのポジション ID (リレーショナルテーブルを先頭タプルから順次走査していくとき何番目のタプルか) を対応させる．Value にはタプル中における各属性データを順次格納した配列データを生成し，その配列データを対応させ格納を行っていく．また現在のローストア DBMS ではディスクアクセスへの効率化のために，データに対して B^+-Tree に代表されるインデックスを利用しているため，本研究のシステムにおいても BerkeleyDB のセカンダリ DB の機能を用いてローストアデータに対するインデックスの実装を行った．セカンダリ DB の機能を用いることにより，ローストアデータのある属性に対してセカンダリ DB を生成した場合，セカンダリ DB に対し定数値を指定したときにセカンダリ DB の対象とした属性において，その定数値をもつタプルデータのプライマリ DB を内部的に参照することで読み出すことが出来る．

一方，カラムストアに対する格納方法は 1 組の Key-Value に複数のカラムデータを格納していく．具体的には Value として記憶装置に対するページサイズと同じ大きさの領域を設定し（厳密には BerkeleyDB などのオーバヘッドを考慮しページサイズより少し小さい領域），この領域（以下，格納ページ）に複数のカラムデータを格納していく．このとき注意しなければならないのがカラムストアはローストアよりもポジション ID を考慮しなければならない点である．これは問合せ処理中にテーブルの一部を再構築する必要が出てくると，ポジション ID を基準としてカラム同士の結合を行うからである．しかし，すべてのカラムデータがポジション ID を保持する (図 2) ことになると格納すべきデータ量が増加してしまうばかりでなく，データ読み込み時にその分のディスク I/O が発生してしまうという問題が起こる．

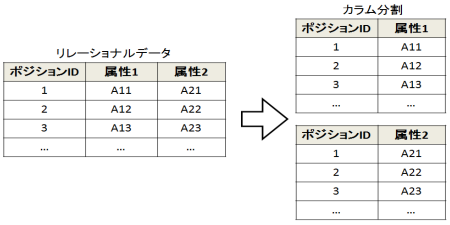

図 2　カラム分割

そこでテーブル内のカラムデータをポジション ID に対してシーケンシャルに格納することで，格納ページ内にヘッダ情報としてページ中の最初に格納するデータのポジション ID を格納しておけば，ヘッダ情報から各データのポジション ID を得ることが出来る．これによって，データ量の増加やディスク I/O の増加が最小限に抑えることが可能となる．このポジション ID による圧縮を利用した方法によって BerkeleyDB を利用したカラムストアの実装は図 3 のようになる．

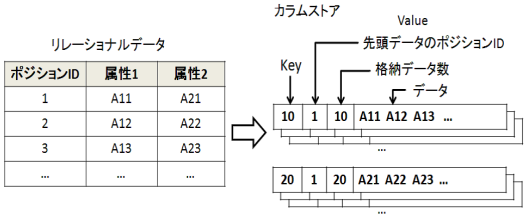

図 3　カラムストアの実装

BerkeleyDB に格納する Value の値としては，カラムデータをシーケンシャルに並べて格納していくが，ポジション ID による圧縮を行うために Value 内にヘッダを設けて，格納するデータの先頭のポジション ID とデータ数をそれぞれヘッダ情報として格納する．また BerkeleyDB の Key の値としては格納ページ内に格納されたカラムデータのうち末尾に格納したカラムデータのポジション ID を格納する．これは BerkeleyDB から読み込みを行う場合にポジション ID を元に検索ができ，指定したポジション ID から対応する格納ページを参照が可能となるからである

クエリ実行に対する実装では，現在多くの DBMS では問合せ演算の処理を Volcano-style [4] を基として実装されている．Volcano-style では各演算子の入出力データやインターフェースを統一し，演算子を組み合わせることによって問合せの実行を行う．ある演算子が処理を行う際には，下位の演算子からデータを受け取り処理を行い上位の演算子に処理データを渡すことによって，全体として各タプルが演算子間をパイルライン処理のようにやり取りされる．また演算子間で入出力データやインターフェースを統一しているため，演算子の拡張性にも優れている．本研究の問合せ処理も，この Volcano-style を参考にして実装を行った．まず演算子間で受け渡しを行うデータとしては，演算子で扱う属性データのスキーマ情報とタプルデータとタプルに対するポジション ID である．ポジション ID はカラムストアに対して行われる演算の演算子によって使用される．問合せ処理を行う際には演算子をノードとした実行木を生成し，始めに演算子間で属性データのスキーマ情報の受け渡しを行った後，順次タプルを受け取り演算子ごとの処理を行っていく．なお，結合演算などで生じる一時データなどの扱いはすべて BerkeleyDB を通じて行われているため，メモリからディスクへの一時データの退避などの処理は BerkeleyDB によって行われている．

5. 実　　験

ローストアとカラムストアのハイブリッド構成において処理速度と消費電力量の間にどのような関係が見られるかの計測実

験を行った結果を示していく．本稿で示す実験は大きく分けて3つである．ローストアとカラムストアとのハイブリッド構成のアプローチにおいて，データを読み込む際にどちらかのデータ格納からデータを取り出すかに大きな差異がでてくる．まず，このローストアとカラムストアのハイブリッド構成において起こるデータ格納の使い分けが考えられる問合せの局所的な処理に対して行った2つの実験を示す．ここでは主に，ローストアとカラムストアのハイブリッド構成における処理時間と消費電力量のトレードオフの関係に着目していく．その後，実際のTPC-H ベンチマークを利用した実験について述べていく．

5.1 実験環境

今回の実験環境を表1に示す．

表1 実験環境

CPU	Intel Corei7-2600K(3.40GHz)
メモリ	DDR3 4GB × 2
HDD	SAMSUNG HD103SJ 1TB × 2
マザーボード	P67A-G45
OS	Ubuntu 11.10
クランプメーター	SANWA CL-22AD
計測器	HIOKI メモリハイロガー LR8400

実験における消費電力量の計測は電源ユニットからマザーボード (MB) への供給電力，電源ユニットから CPU への供給電力，電源ユニットから HDD への供給電源をそれぞれ測定し MB，CPU，HDD における消費電力量の合計値をクエリにおける消費電力量として示している．計測器としてクランプメーターと HIOKI のメモリハイロガー LR8400 を利用して，電源ユニットから HDD における低電流回路にはシャント抵抗，電源ユニットから MB や CPU における高電流回路に対してはクランプメーターを使用してメモリハイロガーによって 10ms ごとの測定を行い，電圧と電流から消費電力量を算出している．処理時間の計測は実時間，ユーザ時間，システム時間それぞれにおいて計測を行った．

また HDD の構成として容量 1TB のディスク2つを利用し，ソフトウェア上で RAID0 のストライピング構成にすることによって，読み込み処理の高速化を図っている．さらには本実験では最小限の電力パフォーマンス下で実験を行うために，BIOS 設定で不要な機能を全てオフにした．また OS 側においても ACPI を利用した CPU の制御を行う cpufreqd デーモンを用いて，CPU の周波数幅の利用を最大幅に CPU 動作モード (government) を ondemand に設定した．これによって CPU がアイドル時には周波数を下げて，CPU の負荷が一定値に達した場合に周波数を急激に上げるといった動作が可能となる．本実験においては，CPU 負荷のしきい値としては 60% に設定して動作させている．また今回用いた CPU では 1.60GHz から 3.40GHz までの周波数幅で 0.20GHz の刻み幅で制御が可能となっている．

クエリの実行に関しては前提条件としては，クエリ処理の前に BerkeleyDB によって管理されているローストアとカラムストアのデータベースファイルが開かれていることとしている．これはクエリの実行だけに焦点をあてるためである．また，意思決定支援システムなどにおけるアドホックなクエリであることも想定している．これによって，OLTP 処理などにおけるデータの局所性からメモリ内にデータ参照に有効なインデックスデータが存在するなどの状態は考えず，常にクエリに関わるデータはメモリ内に存在していないこととして実験を行っている．これは OLAP 処理の対象が大量のデータセットであることからも想定できる環境である．

5.2 フルスキャン処理

まずテーブルに対するフルスキャンの処理に対して計測を行った．ローストアに対してフルスキャンを行う場合には，1つのタプルの属性データがまとまって格納されているので1つのタプルデータを順次と読み込んで行くことは単純な読み込みで処理が可能である．一方カラムストアに対してフルスキャンを行う場合には，1つのタプルの属性データが異なるページ上に位置するために，まずそのページをすべて読み込まなければならない．その後に読み出した同一タプルの属性データを連結する処理を行わなければならない．しかしフルスキャンにおいて必要なデータがテーブル中の全属性データではなく，一部の属性データである場合は読み込み対象テーブルのすべてのカラムデータに対して読み込む必要はなく，問合せにおいて必要な属性のカラムデータのみを読み込みポジション ID によって結合して出力をすればよい．

この実験では対象データは TPC-H ベンチマークで用いられている LINEITEM テーブルを利用した．この際データの規模を示す ScaleFactor は 20 に設定した．カラムストアにおいては問合せによっては処理に不必要な属性データには読み込み処理を行わなくても良いため，読み込みの属性数を先頭の属性より2属性から 16 属性まで1ずつ変化させながらフルスキャンする処理に対して計測を行った．

その結果を処理時間 (図 4) と消費電力量 (図 5) それぞれについて示す．図 4,5 の横軸では処理のパターンを表しており，2 ～ 16 の値はカラムストアに対する処理であり，数字が読み込む属性数を示している．また”row ”はローストアに対する処理を示している．

ここでカラムストアにおいて 15 属性を抽出する処理に着目すると，ローストアからのフルスキャンに比べカラムストアの実行時間は約 4.8% 多いのに対して，消費電力量では約 8.1% 削減された値となっている．

5.3 選択・射影処理

クエリ実行における選択演算はタプル中の特定属性に対して条件を与えて，その条件に一致するタプル全体を出力する演算である．一方，射影演算はタプルから特定の属性のみを抽出する演算である．この2つの演算処理はクエリ処理全体を考えたとき，処理中に扱うデータ量を減らす為に重要な演算処理であり，クエリ木において比較的，葉ノードに近い場所で行われる．このため選択・射影処理はローストアとカラムストアのどちらを利用するかによって，その具体的な処理内容が異なってくる．本実験では以下のような4パターンについてのデータの絞りこ

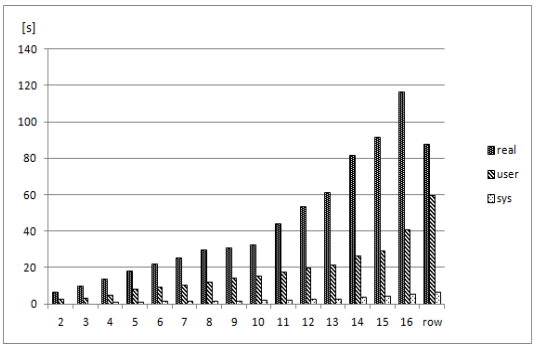

図 4　フルスキャン処理における処理時間

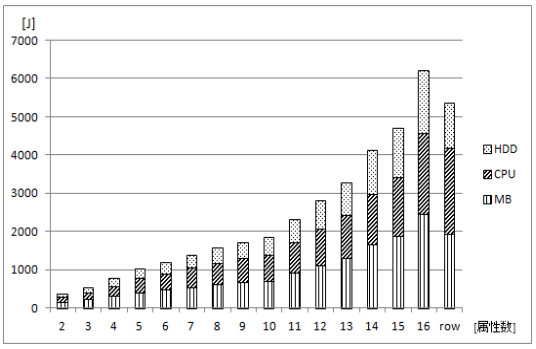

図 5　フルスキャン処理における消費電力量

みを行う選択・射影の処理を想定した．

パターン 1　ローストアに対して射影フィルターを利用したフルスキャンを行い，選択演算を行う．
パターン 2　ローストアに対して選択演算の条件からインデックスを利用してデータを取り込み必要属性の抽出を行う．
パターン 3　カラムストアに対して抽出に必要なデータを各カラムデータから全て読み込み，結合してタプルを具体化した後に選択演算を行う．
パターン 4　カラムストアに対して選択演算に必要なカラムデータのみを抽出し，選択を行った後に出力に必要な属性データを読み込みながら結合し，タプルを具体化して出力を行う．

この 4 つのパターンに対してデータの絞り込みを行う処理を実施した．この実験では属性ごとのデータ型の違いにより処理時間や消費電力量の偏りが発生しないように integer 型に揃え，interger 型の属性 5 つで，10 億レコードを持つテーブルをデータセットを用いて実験を行った．また選択演算の選択条件を 1 つの属性に対しての等号条件を満たすタプルを抽出する条件式にして，選択率を 0.001% として行った．つまり演算の対象データのタプル数が 10 億タプルなので，選択演算によって 1000 タプルの抽出がされる．また選択演算によって抽出されるタプルは演算対象のリレーショナルデータ中には一様分布で存在しているものとしている．また出力する属性数を選択演算の選択条件の対象となる属性を含めて 3 属性としている．

実験の処理時間に関する結果を図 6 に，消費電力量に関する結果を図 7 に示す．結果からパターン 1 が実行時間も消費電力量の他の 3 つのパターンに比べて明らかに大きいことが分かる．ここでローストアのインデックスを用いたアプローチであるパターン 2 とカラムストアを用いた処理のうち，実行時間が短いパターン 3 とを比較してみると，パターン 2 ではパターン 3 に比べ 42% 長い処理時間がかかっているが，消費電力量が 19.1% 削減され処理が行えている．

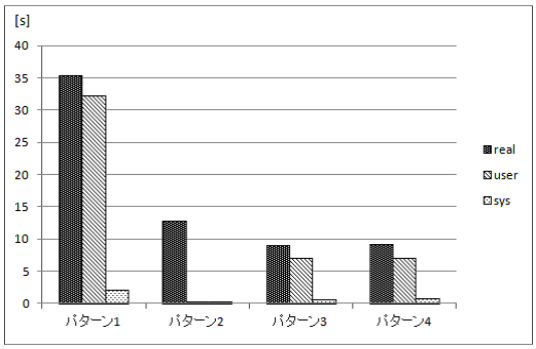

図 6　選択・射影処理における処理時間

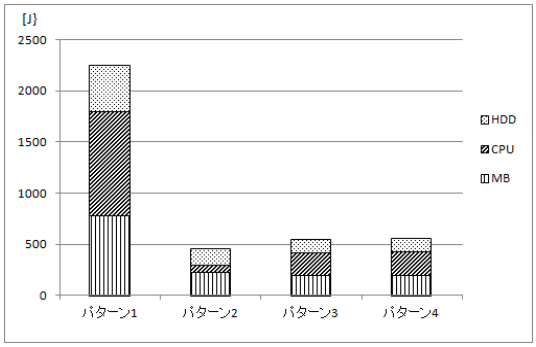

図 7　選択・射影処理における消費電力量

5.4　TPC-H クエリ 5

本実験では TPC-H ベンチマークのクエリ処理を対象に実験を行った結果を示す．TPC-H は意思決定支援システムのアドホックなクエリを想定したベンチマークである．本実験では ScaleFactor(SF) が 1 と 20 の TPC-H のクエリ 5 に対して実験を行った．ローストアとカラムストアのデータ格納方法の違いを検証するために，ローストアに対するアクセスのみでの実行，カラムストアに対するアクセスのみでの実行，ローストアとカラムストアのハイブリッド構成における実行の 3 つのパターンを分類してクエリ実行を行った．またハイブリッドにおけるデータ増加の問題からディスク 2 台の RAID 構成に追加しディスク 1 台の RAID 構成なしでの環境においての実験も行った．それぞれの実行プランは PostgreSQL における実行プランをもとに生成を行った．ハイブリッド構成においては，LINEITEM テーブル以外はカラムストアで読み込みを行い，LINEITEM テーブルはローストアのインデックスを用いた読み取りを行っている．まず SF=1 での結果の実行時間を図 8，消費電力量を図 9 に示す．横軸の接尾語が disk1 はディスク 1 台で RAID 構成なし，disk2 はディスク 2 台で RAID 構成ありをそれぞれ示す．

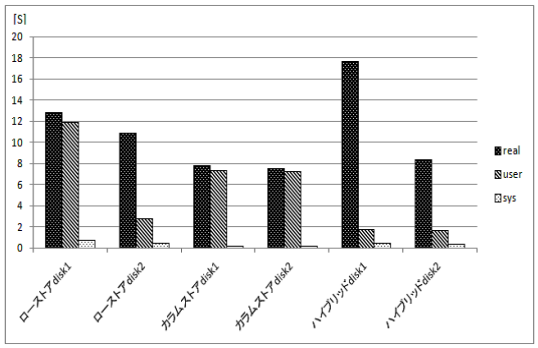

図 8　TPC-H(SF=1) クエリ 5 における処理時間

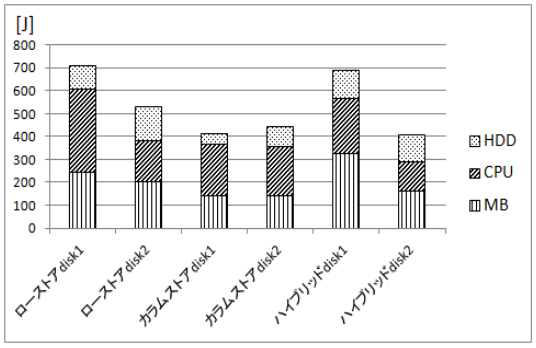

図 9　TPC-H(SF=1) クエリ 5 における消費電力量

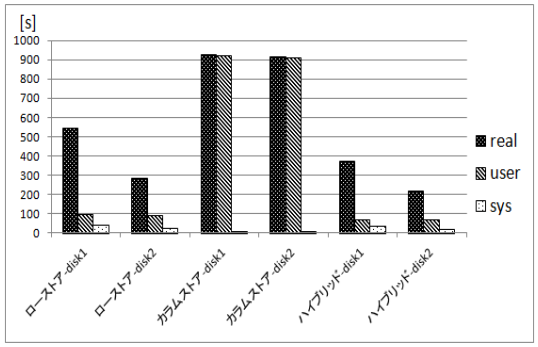

図 10　TPC-H(SF=20) クエリ 5 における処理時間

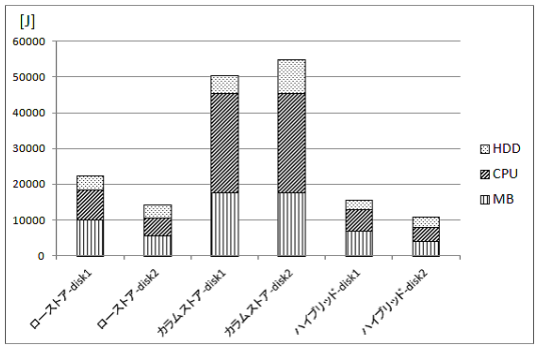

図 11　TPC-H(SF=20) クエリ 5 における消費電力量

実験の結果から RAID 構成でのカラムストアとハイブリッドを比較してみると，ハイブリッドがカラムストアに対して実行時間が 11.5% 増加しているのに対して，消費電力量は 8.0% の減少が見られる．

次に SF=20 での結果の実行時間を図 10，消費電力量を図 11 に示す．SF=20 場合を見てみると，先ほどの SF=1 の場合とは異なり，カラムストアの実行時間と消費電力量のコストがもっとも大きいことがわかる．こちらは RAID 構成でのハイブリッドがもっとも良く，RAID 構成上での比較を行うとハイブリッドの実行時間がローストアの 74.9%，カラムストアの 23.4% となっており，消費電力量ではローストアの 75.8%，カラムストアの 19.8% という結果になった．

5.5　考　　察

5.2 節の実験ではカラムストアでの属性の抽出が 15 属性の場合にローストアとカラムストアの間で処理時間と消費電力量のトレードオフが見られた，よってある程度の属性数を読み込むときには，ローストアからの読み込みとカラムストアからの読み込み関して実行時間と消費電力量の間にトレードオフが発生するため，最適化処理においては実行時間コストと消費電力量コストの小さくしたい方のコストに対してローストアとカラムストアどちらの格納データから読み込みを行うかを選択すればよいことになる．しかし，この実験ではデータセットとして LINEITEM テーブルが用いられているため，各属性のデータ型が一致していないことがいえる．このためフルスキャン処理においてローストアかカラムストアのどちらの方がコストが小さく実行できるかの要因としてはローストアにおいて読み込む属性数だけではなく，読み込む属性数のデータ型に対しても考慮する必要がある．

次に 5.3 節における処理時間の結果を見ると全体的にローストアをフルスキャンした後に選択演算を行うパターン 1 では実行時間，消費電力量共にコストが大きいことがいえる．ここでローストアに対するインデックスを利用しているパターン 2 とカラムストアを利用したアプローチを比較してみると，実行時間においてはパターン 3,4 のほうがパターン 2 よりも少ない時間で実行出来ているのに対して，消費電力量においてはパターン 3,4 よりパターン 2 のほうが低消費電力量で実行できている．これはパターン 3,4 のカラムストアに対するアプローチに比べパターン 2 においてユーザ時間やシステム時間がわずかであることや，CPU の消費電力量が少ないことから，インデックスの利用によってディスクに対するランダムアクセスが多く発生することが原因であると考えられる．つまりランダムアクセスが多く発生することによって，プロセスが待機状態となる時間が多く発生し，その間における CPU での消費電力量が低減されていると考えられるため，パターン 2 ではカラムストアを利用する場合よりも処理時間は多くかかってしまうが，処理にかかる消費電力量は少なく実行できていると考えられる．

5.2 節と 5.3 節における実験ではローストアとカラムストアのハイブリッド構成において起こるデータ格納の使い分けが起こる問合せの局所的な処理に対しての実験を行った．その中で実行時間と消費電力量のトレードオフが存在することを示したが，フルスキャンにおいては読み込み属性数や属性のデータ型，絞り込み処理においては選択率や選択条件などに対してさまざまな状況が考えられる．その中でローストアとカラムストアのアクセス対象の違いによって実行時間と消費電力量間のトレードオフが起こらない場合も考えられるが，複雑な TPC-H のよ

うなクエリにおいては 1 回の問合せ処理で何度もデータの絞り込みの処理を行わなければいけない場合が多く，データの絞り込みの処理においてトレードオフを考慮して最適化を行うことは有効であると考える．

また 5.4 節の TPC-H ベンチマークにおけるクエリ 5 の実験では，前の 2 つの実験とは異なり実際のシステムを想定したクエリの実行による評価が出来ている．まず SF=1 の場合では，実行時間のコストがもっとも良いのは RAID 構成のカラムストア，消費電力量のコストでは RAID 構成のハイブリッドであることがわかる．一方 SF=20 の場合について考えていくと，カラムストアの実行時間と消費電力量のコストが他と比べ大きくなっていることがわかる．これは他と比べカラムストアではカラムごとの結合を行う必要があり CPU への負荷がかなり多いことで，このような結果になったことが考えられる．また実行時間, 消費電力量共に RAID 構成のハイブリッドがもっとも良いことがわかる．このことからハイブリッドにすることにより，データのミラーリングに必要なディスクスペースが増加し，それによるディスクの追加分の消費電力量の増加が考えられるが，必ずしもそうでないことが言える．実験で示すように，ローストアやハイブリッドの場合ではディスク 1 台より 2 台による RAID 構成を行った方が実行時間の高速化が見られており，それに伴い消費電力量の値もディスクを 2 台用いた場合の方が全体的に減少していることがわかる．しかしカラムストアにおいては実行時間は減少しているが，ディスク I/O の負荷がローストアやハイブリッドと比べ少ないことから大幅な減少ではなく消費電力量ではディスクが 1 台の場合の方が少ないという結果になっている．これらのことから消費電力量を考慮した OLAP 処理の最適化を行う際には，扱うデータ量や属性数などのパラメータだけではなく，ディスク構成などのハードウェアの構成もパラメータの 1 つとして考えながら最適化を行わなければならないと考えられる．

これらのことからデータ格納方法においては，ローストアとカラムストアのハイブリッド構成が, これまでのローストアもしくはカラムストアどちらかのデータ格納を用いるアプローチよりも有効であることが TPC-H ベンチマークにおけるクエリ 5 で示すことが出来た．特に SF=20 の場合においてはどちらかのデータ格納方法を用いるより実行時間と消費電力量共に大きなコスト削減が見られた．

6. まとめと今後の課題

本稿では OLAP 処理に対して低消費電力を考慮した問合せ処理を行うために，ローストアとカラムストアの両方のデータ格納方法を利用したハイブリッド構成による問合せ処理を提案し，ハイブリッド構成での OLAP 問合せ処理において実行時間だけでなく消費電力量削減を目指した．このアプローチで限られた条件ではあるが，ハイブリッド構成においてローストアとカラムストアの使い分けが考えられる問合せの部分的な処理において，ローストアとカラムストアのそれぞれを利用した場合に，実行時間と消費電力量の間にトレードオフの関係があることを述べた．これによってハイブリッド構成下ではユーザが実行時間または消費電力量のどちらかのコストを重視して実行を行いたい場合，重視したいコストに対してアクセスする格納データを変更し，実行することによってユーザのニーズに対応して問合せ最適化が行えると考える．さらに TPC-H ベンチマークを用いた実験では，クエリ 5 においてハイブリッド構成で実行時間と消費電力量の削減が行うことが出来た．またハイブリッド構成において問題とされているデータ量とディスクの増加による，消費電力量の問題に対してもディスクを追加することによってディスクのレイテンシが短縮することが考えられ，実際の実験においても実行時間が短縮したことが観測できた．また時間の短縮によって消費電力量の減少も観測された．よってディスクの追加による消費電力量の増加は，場合によっては問題にならないと思われる．しかし，単位時間あたりの消費電力量である電力は増加してしまうということがある．

今後の課題として，消費電力を考慮したオプティマイザの設計，カラムストアに対するデータ圧縮の導入，電力を消費する原因となるデータを SSD に配置する等が挙げられる．

謝　辞

研究の一部は，科研費補助金基盤研究 (A)(課題番号:22240005) の支援による．ここに記して謝意を表す．

文　献

[1] Daniel J. Abadi, Samuel R. Madden, and Nabil Hachem. Column-stores vs. row-stores: how different are they really? In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 967–980, New York, NY, USA, 2008. ACM.

[2] George P. Copeland and Setrag N. Khoshafian. A decomposition storage model. In *Proceedings of the 1985 ACM SIGMOD international conference on Management of data*, SIGMOD '85, pages 268–279, New York, NY, USA, 1985. ACM.

[3] John Gantz and David Reinsel. The digital universe decade, are you ready? *IDC*, 2009(May):1–16, 2010.

[4] G. Graefe. Volcano- an extensible and parallel query evaluation system. *IEEE Trans. on Knowl. and Data Eng.*, 6:120–135, February 1994.

[5] Ravishankar Ramamurthy, David J. DeWitt, and Qi Su. A case for fractured mirrors. *The VLDB Journal*, 12:89–101, August 2003.

[6] Mike Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, and Stan Zdonik. C-store: a column-oriented DBMS. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 553–564. VLDB Endowment, 2005.

[7] Zichen Xu, Yu-Cheng Tu, and Xiaorui Wang. Exploring power-performance tradeoffs in database systems. *ACM SIGMOD*, pages 485–496, 2010.

[8] 福澤 優, 宮崎 純, 藤澤 誠, 天野 敏之, and 加藤 博一. 消費電力を考慮した olap 問合せ処理. **情報処理学会関西支部 支部大会** (C-03), 2011.