データベースエンジン省エネルギー実行による業務模擬処理における省エネルギー効果の評価

茂木 和彦† 西川 記史† 木村 耕治† 早水 悠登‡ 合田 和生‡ 喜連川 優‡

†株式会社日立製作所 〒185-8601 東京都国分寺市東恋ヶ窪 1-280
‡東京大学 生産技術研究所 〒153-8505 東京都目黒区駒場 4-6-1
E-mail: †{kazuhiko.mogi.uv,norifumi.nishikawa.mn,kohji.kimura.zn}@hitachi.com
‡{haya,kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし IoT や AI 技術の普及に伴うデータセンタのエネルギー消費の急増により，提供される IT サービスの性能を維持しつつエネルギー消費を削減することが求められている．本稿では，データ利活用処理向けのデータベースエンジンにおける省エネルギー実行方式とその効果について論じる．データ利活用処理は 2 種類に大別できることを述べ，続いて記憶デバイスの電源 On/Off によるデータベースエンジンの省エネルギー実行方式を説明する．非順序型実行原理を用いるデータベースエンジンに本方式を組み込み，実験により効果を確認した．集計処理においては，実験にて最大で従来比 25%の消費エネルギーを削減する効果を確認した．特定事象に関連するデータの抽出処理においては実験にて 99%の消費エネルギー削減(エネルギー消費効率が約 120 倍向上)する効果を確認した．

キーワード 非順序型実行原理，データベースエンジン，省エネルギー，電源制御，IoT 応用

1. はじめに

IoT や AI 技術の普及に伴うデータセンタのエネルギー消費の急増により，提供される IT サービスの性能を維持しつつ IT 機器のエネルギー消費を削減することが求められている．Nature Web 版[1]によると，2018 年のデータセンタの消費電力量はおよそ 500TWh と推定され，2030 年にはおよそ 3000TWh にまで急増し，これは全消費電力量のおよそ 8%にあたると予測される．データ処理量は増加し続けると考えられる現在，消費電力の削減にはデータセンタにおけるエネルギー消費効率の向上が必須であり，今後益々社会的に重要な課題になると考えられる．

これまで，データセンタのエネルギー消費効率の計測の指標として，データセンタの設備や IT 機器のエネルギー消費効率を計測するため PUE [2] や REF [3], ITEEsv [4] などの指標が標準化されている．更に，応用（ソフトウェア）による省エネルギー効果の評価を目的とした，そのエネルギー消費効率を計測する指標[5]が Application Platform Energy Effectiveness (APEE) として現在標準化プロセスの終盤[6]にある．これにより機器から応用までの複数のレイヤでエネルギー消費効率の計測・評価が可能になり，機器のみならず応用（ソフトウェア）まで含めたデータセンタの省エネルギー化への注目が集まるようになると考えられる．

IT の処理は，実装によりその性能や消費エネルギーが変化する．例えば文献[7] では，ストレージ内のデータ配置を変更することにより，若干の性能低下はあるもののストレージの消費電力を最大 60%以上削減できることが示されている．また文献[8] では，RDBMS の問合せ実行計画やストレージ構成，処理の実行方式により同一処理を実行しても，それらに応じて消費エネルギーが異なることが示されている．つまり，IT における処理性能とエネルギー消費の双方を意識した実装の選択は困難であり，その容易化が課題である．

IT 機器において特に処理当たりの消費エネルギーが多いのは，多量のデータを扱うデータ利活用のための処理を行うシステムのものであると考えられる．このシステムでは多量のデータを格納するストレージが必要になり，その分消費エネルギーが増えると考えられるためである．そこで，この処理パターンを抽出し，それらにおいてエネルギー消費効率を向上させる実装方針を明確化することにより，IT における処理性能とエネルギー消費の双方を意識した実装を容易化することが可能になると考えられる．

日立のデータ利活用処理向け RDBMS 製品である Hitachi Advanced Data Binder(HADB)[1]の利用事例やその他公開情報を基にデータアクセスパターンを分類した．その結果，大きくは 2 パターンに分類できることがわかった．1 つは，多量データへの表全査処理を多用する従来型の分析処理である．もう 1 つは，特定事象・特定対象（必要に応じてそれらに関連する事象・対象を含める）に関する少量のデータを高速に抽出する処理である．今回，それぞれのアクセスパターンに

[1] Hitachi Advanced Data Binder は，内閣府の最先端研究開発支援プログラム「超巨大データベース時代に向けた最高速データベースエンジンの開発と当該エンジンを核とする戦略的社会サービスの実証・評価」(中心研究者：喜連川 東大教授／国立情報学研究所所長)の成果を利用しています．

おける実装方式を検討し，その効果を業務模擬処理にて評価した．

本稿は以下の構成を取る．2 章において事例を基に分類した 2 種類のアクセスパターンについて説明する．3 章では今回評価するデータベースエンジンの省エネルギー実行方式の概略と，2 章で述べたアクセスパターンにおいて適切と考えられる省エネルギー実行方式の最適化の観点について述べる．4 章では，業務模擬処理を用いて評価した省エネルギー化の効果実験とその結果について述べる．5 章でデータベースエンジンの省エネルギー化に関連した関連研究について述べ，6 章で本稿をまとめる．

2. データ利活用処理におけるデータベースエンジンのデータアクセスパターンの分類

はじめに，データ利活用処理向けの日立 RDBMS 製品である HADB の利用事例やその他公開されている情報を基にデータ利活用処理の処理パターンを分類する．

まず，データ利活用処理の 1 つの基本パターンは従来型の分析処理になると考えられる．これは大雑把に言って，多量のデータに対して集計や統計解析等を実施するものである．このとき，これら処理を RDBMS で行うケース[9,10]や，データ分析ツールや統計処理言語で処理を実施し，RDBMS はそれらが利用するデータを管理，要求に応じてそれらにデータを供給するケース[9]がある．これらのケースにおいては，処理対象となるデータがあまり絞り込まれていないことは珍しくない．そのため，RDBMS のデータアクセスパターンとしては表全走査処理のケースが多い．この高速化のため，近年では分析処理向けにカラムストア型のデータ格納方式を採用する RDBMS が多い[11]．

IoT 等により収集した詳細・多量データの利活用が普及するにつれ，集計や統計解析だけではなく，特定事象等に関して詳細調査する目的に収集した詳細データをそのまま参照するケースが増えてきた．例えば，金融業において，不正取引が疑われた場合に関連する取引履歴データを抽出する処理を行う事例[12]がある．他の例としては，北米送電事業者では送電網の詳細モニタデータを収集し，必要に応じて可視化やモデルを用いた障害把握等に利用している[13]ことを挙げることができる．また，所謂 cyber physical system 的な用途で，多数のパターンで事前シミュレーションを行いその結果を DB に保存し，特定条件の状態を事前シミュレーション結果から取得もしくは近傍条件の結果から計算する事例[14,15]がある．

これらの処理をまとめて表現すると，特定事象・特定対象（必要に応じてそれらに関連する事象・対象を含める）に関する少量のデータを高速に抽出する処理となる．この処理を高速に行うための RDBMS におけるデータアクセスパターンは，B-Tree 索引等の 2 次索引を用いて処理対象データを絞り込む 2 次索引走査となる可能性が高い．

以上の様に，データ利活用におけるデータアクセスパターンは大きくは 2 パターンに分類できる．1 つは集計・統計解析等における表全走査処理を行い，もう 1 つは 2 次索引を用いて部分的にデータを取得するものである．

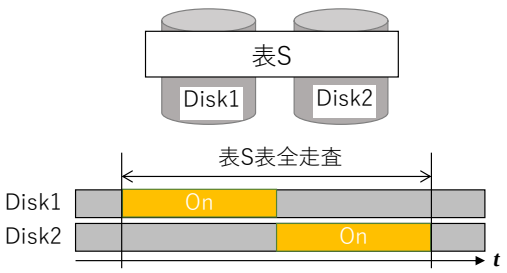
3. データアクセスパターン毎のデータベースエンジン省エネルギー実行方式の概略

3.1. データベースエンジン省エネルギー実行方式の基本動作

文献[8]では，基本的な関係データベース演算における消費エネルギーコストモデルと，基本的な関係データベース演算における演算実行方式と処理実行時にアクセスされる HDD 台数を変化させた際の消費エネルギーの変化について論じている．その結果から，関係データベース演算を行う際には，サーバにおける CPU 性能に応じた最適な HDD 台数のとき，つまり，CPU の処理スループットと HDD の IO スループットが均衡したときに処理当たりの消費電力が最小になることが分かる．従って，HDD 等の記憶デバイスの電源 On/Off の処理時間や消費電力のオーバヘッドの影響があるものの，基本的に，CPU の処理スループットと記憶デバイスの IO スループットが等しくなるように記憶デバイスの電源を On/Off することにより処理に必要な消費エネルギーを最小化できると考えられる．

ただし，この電源制御を簡単に実行できるかどうかはデータの走査処理方式に依存する．表全走査処理の場合，各行へのアクセス順に依存関係は存在しない．つまり，処理対象データが保存される記憶デバイスを順次電源を On にしてその中にあるデータを取得しても問題は発生しない．従って，CPU の処理スループットに応じた台数の記憶デバイスを順次電源 On/Off する制御によりエネルギー消費を削減しながら表全走査

図 1 表全走査処理における電源制御のイメージ



処理が可能である．（図 1）

一方の 2 次索引走査処理の場合，索引をアクセスした後に索引により判別された対象行にアクセスする必要がある．従って，索引と表のデータが入っている記憶デバイスを全て電源 On にしない限り，IO 毎にアクセス先記憶デバイスの電源状態を確認し，On でない場合には On になる迄待たせる制御が必要になる．

3.2. 時系列データにおける 2 次索引走査処理の省エネルギー実行方式

時系列データやその他大規模データの場合，古いデータの部分削除の容易化やプルーニングによる性能向上の観点からパーティショニング等のデータ分割機構（分割方法を指示する論理分割は必須で，物理分割も併用する実装が多い）を利用することが多い．このとき，2 次索引をそのデータ分割内ローカルに作成し，同じ分割単位に含まれる表データと索引を同じ記憶デバイス群に保存する制約を課した場合，その記憶デバイス群の電源を On にするとその分割単位においては 2 次索引走査が可能になる．つまり，この制約を設けた場合，表全走査と同様に分割単位を順次アクセスし，それに対応する記憶デバイス群を処理に応じて電源 On するよう制御することにより 2 次索引走査におけるエネルギー消費を削減することができる．（図 2）

3.3. 多段ネステッドループ結合処理における一時表利用による省エネルギー実行方式

関係データベース演算の処理は，入力データに対して原則パイプライン動作をする方式と，入力データが一定量蓄積されないと実行が開始できない方式が存在する．記憶デバイスの電源 On/Off によるエネルギー消費の削減を考えた場合，パイプライン的に動作をするとその分同時に処理対象となるデータが増え，同時に電源 On すべき記憶デバイスの数が増えることになる．これは，処理の消費エネルギーを増加させる要因となる．

主要な結合処理方式について考えると，ネステッドループ結合処理はパイプライン動作するもので，ハッシュ結合処理は一定量蓄積されないと実行が開始できないものになる．つまり，多段ネステッドループ結合処理の場合，各段の結合処理対象のデータが異なる記憶デバイスに保存されているとその分消費エネルギーが増加することになる．また，3.2 節で述べた分割を行った表同士をネステッドループス結合を行った場合，分割数が増えると結合処理のために必要な記憶デバイスの組合せが分割数の掛け算で増え，その際の記憶デバイスの電源 On/Off のオーバヘッド（遷移待ち時間とその間の消費電力，HDD 電源 On 時の突入電流による消費電力）が増加することになる．これらの場合，各段毎の処理結果を一時表として保存することによりエネルギー消費を削減できる可能性がある[16]．（図 3）この一時表利用の判断は，文献[8]で述べられた消費エネルギーコストモデルを用いて問合せ最適化処理にて行うことが適切と考えられる．

図 2 分割を用いた 2 次索引走査における電源制御のイメージ

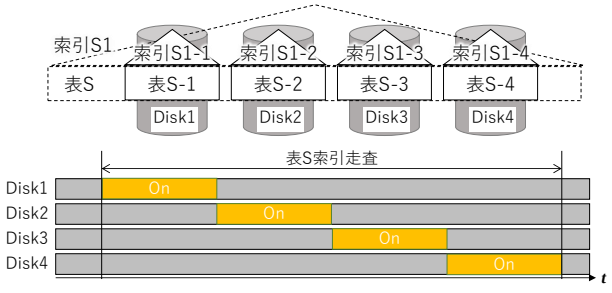
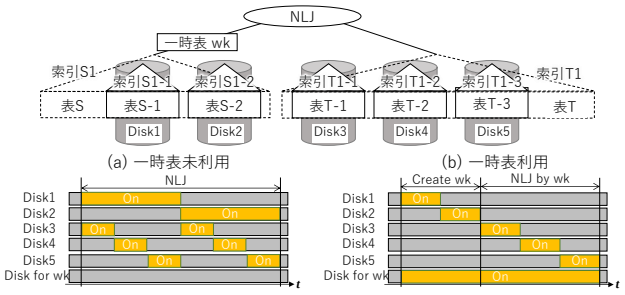


図 3 一時表利用多段ネステッドループ結合における電源制御のイメージ



4. 処理パターン毎の業務模擬処理による省エネルギー効果の評価実験

4.1. 実験環境

表 1 に評価実験に用いるハードウェア構成を示す．データセンタでの処理を想定し，汎用サーバとエンタ

表 1 実験ハードウェア構成

機器	概要
サーバ	日立 HA8000/RS220AN CPU: 12 物理コア/socket×2 socket (24 物理コア) Intel® Xeon® CPU E5-2690 v3 @ 2.60GHz OS: RHEL 6.6
IO パス	8Gbps Fibre Channel × 4 パス
ストレージ	日立 HUS130 15 krpm 300 GB SAS HDD×(256 + 8) 台 ((7D + 1P)×32 RG + 8HDD spares) DB を(7D + 1P)×32 RG 上に構築 表・索引: 31 RG, 小規模表・索引，ワーク: 1RG(常時電源 On)
電力計	横川電機 WT1800

ープライズストレージを用いた構成で実験する．RDBMS として HADB v04-03 に 3 章で説明した省エネルギー実行方式を試験的に組み込んだものを利用する．

本実験で利用するストレージは記憶デバイスである HDD の電源を RAID Group(RG: RAID の冗長構成を取る HDD の組)単位で On/Off するソフトウェア IF を有しており，これを利用して RDBMS から省エネルギー実行時の HDD への電源 On/Off 指示を行う．なお，常時電源 On される RG(1 つ)があり，ここにマスタデータ表等の小規模表や，一時表格納等に利用するワーク領域を割当てる．

なお，HADB は非順序型実行原理[17][2]を採用している．そのため，文献[8]で論じられているように 2 次索引走査の処理性能が高く，その効果で処理当たりの消費エネルギーが従来型の実行方式より大幅に削減されることを期待できる．

4.2. 実験データベースの設計と評価処理

2 章で説明したデータアクセスパターンの分類に従いそれぞれに適したデータベーススキーマとデータを利用して評価実験を行う．以下，省エネルギー実行に関連するデータベース設計を説明する．

集計処理（表全走査）

従来型の分析処理を模した評価を目的とし，データアクセスパターンとしては表全走査を想定する．この分野で著名なベンチマークである TPC Benchmark™ H (TPC-H)[18]のデータベーススキーマ・データセットを用いる．TPC-H は販売履歴分析(データ期間 7 年分)を模したものであるが，その中から Q1(単一表走査), Q8(8 表結合), Q19(2 表結合)の 3 処理により評価する．データの規模は Scale factor=10,000(10TB)とする．

lineitem, orders の各表は時系列データである．そこで, HADB が有する“チャンク”と呼ぶ論理構造を用い，lineitem 表は l_shipdate, orders 表は o_orderdate の値(年月）毎にチャンクを分割し，各チャンク毎に同じチャンクに属するデータは 1 つの RG に閉じで記憶するよう制御する．履歴表データを格納する RG 26 個にチャンク（年月）毎にラウンドロビンで割当てる．part, supplier, customer, partsupp の各表は履歴表とは異なる RG 5 個に格納する．nation, region 表は極めて小さいため常時電源 On となる RG に格納する．本評価では表データをカラムストア型で格納し表全走査処理を高速化する．また lineitem, orders の各表においては，DATE 型のカラムについてチャンク毎にその最大値/最小値を保持する補助情報を持ち，可能であれば処理対象日付条件によりチャンクのプルーニングを行う．(Q1 の lineitem 表, Q8 の orders 表)

特定事象関連データ抽出処理（2 次索引走査）

特定事象に関連したデータの抽出処理を模した評価を目的とし，データアクセスパターンとしては 2 次索引走査を含むことを想定する．文献[19]においてデータベース処理のエネルギー消費効率評価に用いられたデータベーススキーマ・データセット[20]に固定装置の位置情報に関する表を追加したものを用いる．これは IoT アプリケーション，具体的には鉱山の露天掘りにおける鉱石運搬車両管理アプリケーションを模したものである.データはシミュレータにより生成する．シミュレーションに登場する装置は鉱石をトラックに積み込むローダ，鉱石を運搬するトラック，鉱石を降ろすダンプ，の 3 種類である．シミュレータでは，トラックがローダで資源等を積み込み，それをダンプの場所まで運び，空になったら再度ローダに行き，資源等を積み込むサイクルを繰り返す．ローダ及びダンプには生産量の累積値を計測するセンサが，トラックには積載量の累積値，移動距離の累積値，現在位置を示す GPS センサが取り付けられ，一定時間間隔でデータを送信する．また，ローダとダンプの間には複数のフィールドセンサ（Beacon）が置かれ，近くに来たトラックの ID を送信する．ローダ，ダンプは累積生産量が一定値を超えると，トラックは累積積載量または累積の移動距離が閾値を超えると保守モードに入る．データ期間は 270 日（約 9 ヶ月）とする．処理として以下の 3 つを用いて評価する. その問合せ SQL 文を図 4 に示す.

Qa　ある指定された日（1 日）に指定されたイベントが発生したダンプに対し指定イベント発生時の直前 1 週間以内にそのダンプで鉱石を下したトラックについて，その際の積載量センサ値を取得

Qb　ある領域にあるフィールドセンサにセンスされたトラックについて, センス時の速度情報を計算するために必要な位置情報（連続 2 サンプル）を取得

Qc　ある指定された日（1 日）に保守から稼働状態に戻ったトラックについて，稼働状態に戻ってから 1 週間以内に利用したローダ・ダンプの利用時のローダ・ダンプの処理量センサ値を取得

データの規模は CSV ファイルでおよそ 7.5TiB になるようにパラメータを設定した.（ローダ・ダンプはそれぞれ 450 機，トラックは 9000 台） このときイベント・センサデータの小規模表は約 2.5 億行，中規模表は約 42 億行，大規模表約 420 億行になる．

[2] 喜連川東大教授／国立情報学研究所所長・合田東大准教授が考案した原理.

図 4 特定事象関連データ抽出処理の問合せ

(a) Qa

```
select ae1.EQUIPMENT_ID, ae3.EQUIPMENT_ID, es.TIME_STAMP, es."VALUE"
  from ((((((ACTIVITY_EVENT_MEASURE ae1
             inner join STATIC_EQUIPMENT_LOCATION sel on (sel.EQUIPMENT_ID = ae1.EQUIPMENT_ID))
           inner join ACTIVITY_EVENT_MEASURE ae2 on (ae1.EQUIPMENT_ID = ae2.EQUIPMENT_ID
                 and ae2.BEGIN_TIMESTAMP between ae1.BEGIN_TIMESTAMP - 7 day and ae1.BEGIN_TIMESTAMP))
         inner join ACTIVITY_EVENT_MEASURE ae3 on (ae2.END_TIMESTAMP = ae3.END_TIMESTAMP))
        inner join EQUIPMENT_SENSOR_RELATIONSHIP esr1 on (ae3.EQUIPMENT_ID = esr1.EQUIPMENT_ID))
       inner join MOVING_SENSOR_LOCATION_MEASURE msl on (esr1.SENSOR_ID = msl.SENSOR_ID
             and msl.TIME_STAMP > ae2.END_TIMESTAMP -5 second and msl.TIME_STAMP <= ae2.END_TIMESTAMP
             and msl.GPS_X between sel.GPS_LOCATION_X -5.0 and sel.GPS_LOCATION_X +5.0
             and msl.GPS_Y between sel.GPS_LOCATION_Y -5.0 and sel.GPS_LOCATION_Y +5.0))
      inner join EQUIPMENT_SENSOR_RELATIONSHIP esr2 on (ae3.EQUIPMENT_ID = esr2.EQUIPMENT_ID))
     inner join EQUIPMENT_SENSOR_MEASURE es on (esr2.SENSOR_ID = es.SENSOR_ID
           and es.TIME_STAMP > ae2.END_TIMESTAMP -5 second and es.TIME_STAMP <= ae2.END_TIMESTAMP)
  where ae1.EVENT_ID     = 'E993'
    and ae2.EVENT_ID     = 'E112'
    and ae3.EVENT_ID     = 'E004'
    and esr1.SENSOR_ID like 'HGP%'
    and esr2.SENSOR_ID like 'HLW%'
    and ae1.BEGIN_TIMESTAMP between TIMESTAMP'2015-08-09 00:00:00.000' and TIMESTAMP'2015-08-09 23:59:59.999'
    and ae2.END_TIMESTAMP between TIMESTAMP'2015-08-02 00:00:00.000' and TIMESTAMP'2015-08-09 23:59:59.999'
    and msl.TIME_STAMP    between TIMESTAMP'2015-08-01 23:59:55.000' and TIMESTAMP'2015-08-09 23:59:59.999'
    and ae3.END_TIMESTAMP between TIMESTAMP'2015-08-02 00:00:00.000' and TIMESTAMP'2015-08-09 23:59:59.999'
    and es.TIME_STAMP     between TIMESTAMP'2015-08-01 23:59:55.000' and TIMESTAMP'2015-08-09 23:59:59.999'
```

(b) Qb

```
select fe.EQUIPMENT_ID, nmsl.TIME_STAMP2, nmsl.GPS_X1, nmsl.GPS_Y1, nmsl.GPS_X2, nmsl.GPS_Y2
  from ((FIELD_SENSOR_LOCATION fsl
           inner join FIELD_EQUIPMENT_MEASURE fe on (fsl.SENSOR_ID = fe.SENSOR_ID))
         inner join EQUIPMENT_SENSOR_RELATIONSHIP esr on (fe.EQUIPMENT_ID = esr.EQUIPMENT_ID))
        inner join NEXT_MOVING_SENSOR_LOCATION_MEASURE nmsl on (esr.SENSOR_ID = nmsl.SENSOR_ID2
              and fe.TIME_STAMP = nmsl.TIME_STAMP2)
  where fsl.GPS_LOCATION_X between  500.0 and  700.0
    and fsl.GPS_LOCATION_Y between 1000.0 and 1050.0
    and esr.SENSOR_ID like 'HGP%'
```

(c) Qc

```
with tgt_equipment as
(
  select EQUIPMENT_ID, END_TIMESTAMP as TIME_STAMP
    from ACTIVITY_EVENT_MEASURE
    where EVENT_ID in ('E990', 'E991')
      and END_TIMESTAMP between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-06 23:59:59.999'
)
select te.EQUIPMENT_ID, ae2.EQUIPMENT_ID, oae.TIME_STAMP, oae."VALUE"
  from ((((((tgt_equipment te
               inner join ACTIVITY_EVENT_MEASURE ae1 on (te.EQUIPMENT_ID = ae1.EQUIPMENT_ID
                     and ae1.BEGIN_TIMESTAMP between te.TIME_STAMP and te.TIME_STAMP +7 day))
             inner join EQUIPMENT_SENSOR_RELATIONSHIP esr1 on (ae1.EQUIPMENT_ID = esr1.EQUIPMENT_ID))
            inner join MOVING_SENSOR_LOCATION_MEASURE msl on (esr1.SENSOR_ID = msl.SENSOR_ID and
                  msl.TIME_STAMP >= ae1.END_TIMESTAMP and msl.TIME_STAMP < ae1.END_TIMESTAMP +5 second))
           inner join ACTIVITY_EVENT_MEASURE ae2 on (ae1.END_TIMESTAMP = ae2.END_TIMESTAMP))
          inner join STATIC_EQUIPMENT_LOCATION sel on (sel.EQUIPMENT_ID = ae2.EQUIPMENT_ID
                and msl.GPS_X between sel.GPS_LOCATION_X -5.0 and sel.GPS_LOCATION_X +5.0
                and msl.GPS_Y between sel.GPS_LOCATION_Y -5.0 and sel.GPS_LOCATION_Y +5.0))
         inner join EQUIPMENT_SENSOR_RELATIONSHIP esr2 on (ae2.EQUIPMENT_ID = esr2.EQUIPMENT_ID))
        inner join OBJECT_ATTRIBUTE_ESTIMATION_MEASURE oae on (esr2.SENSOR_ID = oae.SENSOR_ID
              and oae.TIME_STAMP >= ae1.END_TIMESTAMP and oae.TIME_STAMP < ae1.END_TIMESTAMP +5 second)
where ae1.EVENT_ID = 'E001'
    and ae2.EVENT_ID = 'E102'
    and esr1.SENSOR_ID like 'HGP%'
    and ae1.END_TIMESTAMP between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:00.000'
    and ae2.END_TIMESTAMP between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:00.000'
    and msl.TIME_STAMP    between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:05.000'
    and oae.TIME_STAMP    between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:05.000'
union all
select te.EQUIPMENT_ID, ae2.EQUIPMENT_ID, oae.TIME_STAMP, oae."VALUE"
  from ((((((tgt_equipment te
               inner join ACTIVITY_EVENT_MEASURE ae1 on (te.EQUIPMENT_ID = ae1.EQUIPMENT_ID
                     and ae1.BEGIN_TIMESTAMP between te.TIME_STAMP and te.TIME_STAMP +7 day))
             inner join EQUIPMENT_SENSOR_RELATIONSHIP esr1 on (ae1.EQUIPMENT_ID = esr1.EQUIPMENT_ID))
            inner join MOVING_SENSOR_LOCATION_MEASURE msl on (esr1.SENSOR_ID = msl.SENSOR_ID and
                  msl.TIME_STAMP > ae1.END_TIMESTAMP -5 second and msl.TIME_STAMP <= ae1.END_TIMESTAMP))
           inner join ACTIVITY_EVENT_MEASURE ae2 on (ae1.END_TIMESTAMP = ae2.END_TIMESTAMP))
          inner join STATIC_EQUIPMENT_LOCATION sel on (sel.EQUIPMENT_ID = ae2.EQUIPMENT_ID
                and msl.GPS_X between sel.GPS_LOCATION_X -0.1 and sel.GPS_LOCATION_X +0.1
                and msl.GPS_Y between sel.GPS_LOCATION_Y -0.1 and sel.GPS_LOCATION_Y +0.1))
         inner join EQUIPMENT_SENSOR_RELATIONSHIP esr2 on (ae2.EQUIPMENT_ID = esr2.EQUIPMENT_ID))
        inner join OBJECT_ATTRIBUTE_ESTIMATION_MEASURE oae on (esr2.SENSOR_ID = oae.SENSOR_ID
              and oae.TIME_STAMP > ae1.END_TIMESTAMP -5 second and oae.TIME_STAMP <= ae1.END_TIMESTAMP)
  where ae1.EVENT_ID = 'E004'
    and ae2.EVENT_ID = 'E112'
    and esr1.SENSOR_ID like 'HGP%'
    and ae1.END_TIMESTAMP between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:00.000'
    and ae2.END_TIMESTAMP between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:00.000'
    and msl.TIME_STAMP    between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:05.000'
    and oae.TIME_STAMP    between TIMESTAMP'2015-06-04 00:00:00.000' and TIMESTAMP'2015-06-14 01:00:05.000'
```

センサデータ等のイベント時系列データはそのイベント時刻の値（日付）でチャンクを分割し，31 個の RG に日付毎にラウンドロビンで割当てる．その他の表（マスタ表）は極めて小さいため，イベント時系列データとは異なる常時電源 On となる RG に格納する．本実験では 2 次索引表走査処理に合わせて表データをローストア型で格納する．またイベント時系列データの各表に関しては，TIMESTAMP 型のカラムについてチャンク毎にその最大値/最小値を保持する補助情報を持ち，処理対象日付によりチャンクのプルーニングを行う．

4.3. 評価実験結果

非順序型実行原理を利用せず，かつ，HDD の電源 On/Off の制御を行わない従来型実行を基準とした処理当たりの消費エネルギーがどの程度削減されたかを示す消費エネルギー削減率と，従来型実行を基準とした相対消費エネルギーの逆数であるエネルギー消費効率向上率で評価を行う．また，結果解釈の材料として従来型実行を基準とした相対処理時間の逆数である処理性能向上率も同時に示す．なお従来型実行は，3 章で説明した省エネルギー実行方式を試験的に組み込んだデータベースエンジンにて省エネルギー制御を実行せず，表全走査処理においては全 RG を均等に処理する設定で計測する．そのため，表全走査処理における負荷分散挙動等，一部製品版 HADB とは異なる挙動にて計測している．

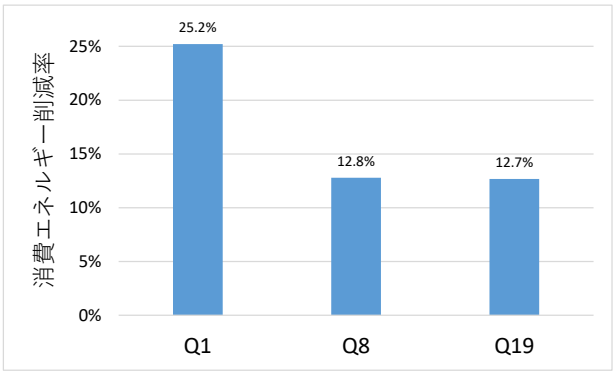
集計処理（表全走査）

実験結果として，図 5 に消費エネルギー削減率，エネルギー消費効率向上率，処理性能向上率を示す．図の横軸は実行した問合せを，図の縦軸はそれぞれの計測値を示す．3.1 節で説明したデータベースエンジンの省エネルギー実行方式に従い，HDD を順次電源 On/Off する制御によりシステム全体の消費電力を下げることができ，消費エネルギー削減率が 12～25%となる結果を得た．なお本実験においては，part, supplier, customer, partsupp の各表を処理する場合には RG5 つを同時に電源 On してアクセスさせ，orders, lineitem の各表においてはアクセス開始時には 4 つの RG の電源を On にし，処理スレッドにおける IO 完了待ちが多い場合には最大 9 まで順次電源 On して同時に処理対象とする RG 数を増やす制御を行った．

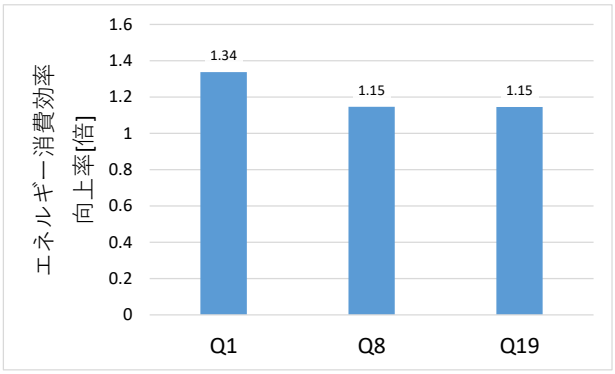
クエリにより消費エネルギー削減率の差が見られるが，これは問合せの処理特性の影響を受けたためと考える．CPU 負荷が高い Q1 は少ない HDD 台数で処理を行うことができ，また，処理時間も他より長めのために相対的に HDD の電源 On/Off のオーバヘッドが小さく処理時間の悪化割合も小さいため，最大となる 25%の消費エネルギー削減率を示したと考える．なお，削減率がこの程度であるのは，サーバやストレージの電源 Off できない部分の消費電力がシステム全体の消費電力のおよそ 65%（無負荷時）に達するためと考えられる．この値を小さくすることにより消費エネルギー削減率を向上させることができると考えられる．

図 5 集計処理の実験結果

(a) 消費エネルギー削減率



(b) エネルギー消費効率向上率



(c) 処理性能向上率

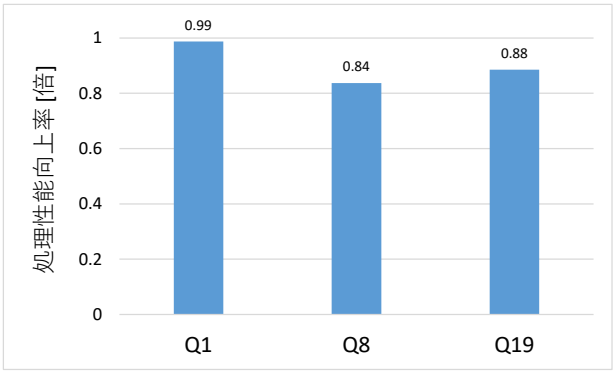
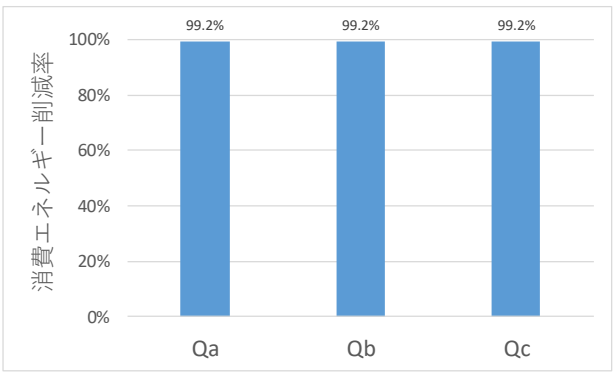
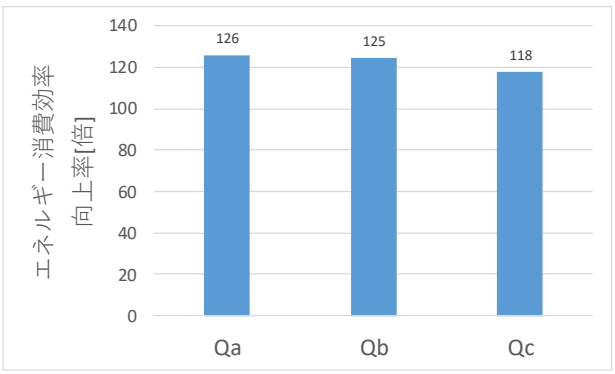


図 6 特定事象関連データ抽出処理の実験結果

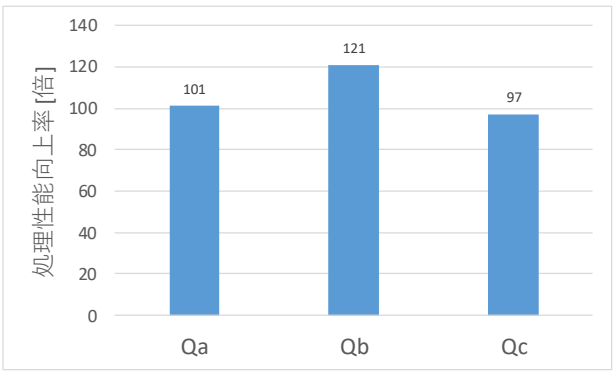
(a) 消費エネルギー削減率



(b) エネルギー消費効率向上率



(c) 処理性能向上率



一方 CPU 負荷が中程度の Q8 と CPU 負荷が小さい Q18 においては，Q1 に比して同時に電源 On してアクセスする HDD 台数が多く，また，CPU 負荷が低い分処理時間が Q1 より短くその分 HDD の電源 On/Off のオーバヘッドが大きくなり Q1 ほどの省エネルギー削減率を示すことができなかったと考える．

特定事象関連データ抽出処理（2 次索引走査）

実験結果として，図 6 に消費エネルギー削減率，エネルギー消費効率向上率，処理性能向上率を示す．図の横軸は実行した問合せを，図の縦軸はそれぞれの計測値を示す．本実験においては，すべての問合せで消費エネルギー削減率が 99%以上，エネルギー消費効率向上率がおよそ 120 倍(118～126 倍)の結果を得た．なお本実験においては，2 次索引走査処理の性能向上効果を鑑み，同時に電源 On にして処理対象とする RG の数の上限を 12RG(96 HDD)とした．結合処理は全てネストループ結合で実行する．内表が電源 On/Off の対象となる RG に格納されている場合，3.3 節で述べた外表側の処理結果で一時表を作成しそれを用いて内表をアクセスする手法を用いる．

文献[8]で論じられているように，非順序型実行原理を用いる場合，高多重で IO を発行することにより HDD 内部の IO スケジューリング効果が得られ，(同時処理 HDD 台数（電源 On する台数)) ×2 倍程度の性能向上効果が期待でき，その効果が図 2(c)の処理性能向

上率として示される．その一方で HDD の電源 On/Off 遷移のオーバヘッドが消費エネルギーを上昇させる方向に働き，その兼ね合いでどの程度消費エネルギーが削減されるかが定まる．Qa, Qc においては処理対象データが日付により絞られており，処理でアクセスする HDD が限られる．その数は設定上限に届かず，最大で 9RG(72HDD)であった．その際，同期間のデータをアクセスするため同じ HDD 群をアクセスし続け電源 On/Off 回数が少なくそのためのオーバヘッドが少なくて済み，HDD の電源 Off によるエネルギー消費の削減効果がより強く出て処理性能向上率より高いネルギー消費効率向上率となったと考える．

一方 Qb では，2 種類のイベント時系列データにおいて両表ともに全チャンクが処理対象となり，それぞれの表をアクセスするのに HDD の電源 On/Off が必要でそのオーバヘッドが無視できず，処理性能向上率と同程度のエネルギー消費効率向上率となったと考える．

5. 関連研究

DBMS が動作するシステムのエネルギー消費を削減する，もしくはエネルギー消費を考慮する技術の研究が行われている．主流の考え方は，問合せ最適化機能がエネルギー消費を削減する問合せ実行計画を生成するものである[21,22]．これは文献[8]で論じられているように，消費エネルギーをコスト指標としたコストモデルを構築可能であり，かつ，現在の問合せ最適化処理はコストベースで実施することが多いことに起因すると考える．

DBMS の処理特性を考慮してハードウェアを効率的に利用する技術の研究は以下のものがある．ストレージに関しては，文献[7]において IO パターン把握に基づくデータのプリロード・データ配置最適化とストレージ（記憶デバイス）の電源管理によりシステムの消費電力を削減する手法について論じている．CPU に関しては，文献[23]において，分析指向問合せ処理における CPU の動作クロック変更によりサーバの消費電力を変化させた場合のシステム全体の消費エネルギーの変化について論じている．

6. まとめ

本稿では，データ利活用処理向けのデータベースエンジンにおける省エネルギー実行方式とそのエネルギー消費の削減効果について論じた．まず，データ利活用処理は 2 種類に大別できることを述べた．その 2 種類の特徴としては，1 つは集計・統計処理等を行うもので，表全走査となることが多い．もう 1 つは特定事象等に関連する少量のデータを抽出する処理であり，2 次索引走査になることが多い．続いて，データベースエンジンの省エネルギー実行方式を説明した．具体的には CPU の処理スループットと記憶デバイスの IO スループットが均衡するように記憶デバイスの電源 On/Off を処理の進捗に合わせて実施するものである．非順序型実行原理を用いるデータベースエンジンにおいてこの省エネルギー実行方式による効果を評価実験により確認した．集計処理においては，実験にて最大で従来比 25%の消費エネルギーを削減する効果を確認した．特定事象に関連するデータの抽出処理においては，実験での処理では消費エネルギー削減率が 99%以上，エネルギー消費効率向上率で見るとおよそ 120 倍との結果を得た．

謝辞

本研究の一部は，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「IoT 推進のための横断技術開発プロジェクト／先進 IoT サービスを実現する革新的超省エネルギー型ビッグデータ基盤の研究開発」および助成事業「高効率・高速処理を可能とする AI チップ次世代コンピューティグの技術開発／高度な IoT 社会を実現する横断技術開発／先進 IoT サービスを実現する革新的超省エネルギー型ビッグデータ基盤の研究開発」の結果得られたものである．

参　考　文　献

[1] N. Jones, “How to stop data centres from gobbling up the world’s electricity,” nature Web, https://www.nature.com/articles/d41586-018-06610-y, 2018.

[2] Information technology - Data centres - Key performance indicators - Part 2: Power usage effectiveness (PUE), ISO/IEC 30134-2:2016, 2016.

[3] Information technology - Data centres - Key performance indicators - Part 3: Renewable energy factor (REF), ISO/IEC 31034-3:2016, 2016.

[4] Information technology - Data centres - Key performance indicators - Part 4: IT Equipment Energy Efficiency for servers (ITEEsv), ISO/IEC 30134-4:2017, 2017.

[5] 五十嵐他，“アプリケーションを含めた IT サービスのエネルギー効率指標の提案,” 第 10 回データ工学と情報マネジメントに関するフォーラム (DEIM 2018), C7-3, 2018.

[6] ISO, “ISO/IEC DIS 23544 Information Technology — Data Centres — Application Platform Energy Effectiveness (APEE),” https://www.iso.org/standard/76000.html.

[7] N. Nishikawa, et.al., “Application Sensitive Energy Management Framework for Storage Systems,” IEEE Transactions on Knowledge and Data Engineering, Vol. 27, Issue 9, pp.2335–2348, 2015.

[8] 早水他，“分析的データベース問合せ処理を対象とするディスクストレージの消費エネルギーコスト推定手法,” 電子情報通信学会論文誌 D, J102-D(1), pp.13-24, 2019.

[9] 酒井他, “レセプト情報等 オンサイトリサーチセンター（京都） パフォーマンステスト結果報告,” 第 32 回レセプト情報等の提供に関する有識者会議, https://www.mhlw.go.jp/file/05-Shingikai-12401000-Hokenkyoku-Soumuka/0000131572.pdf, 2016.

[10] 日立, “超高速データベースエンジン Hitachi Advanced Data Binder :【導入事例】小売（顧客嗜好分析）,” https://www.hitachi.co.jp/products/it/bigdata/platform/data-binder/case/retail/index.html.

[11] 田村, “カラムストアデータベースの技術と活用法,” db tech showcase.com, https://www.db-tech-showcase.com/library/dbts-tokyo-2015/b12-infoframe-databooster-nec.

[12] 日立, “超高速データベースエンジン Hitachi Advanced Data Binder :【導入事例】金融（不正取引検知）,” https://www.hitachi.co.jp/products/it/bigdata/platform/data-binder/case/financial/index.html.

[13] K. D. Jones, “Zero to One: A Digital Transformation at Dominion Energy,” Fall 2019 NASPI, https://www.naspi.org/sites/default/files/2019-10/03_Dominion_Jones_20191029.pdf, 2019.

[14] 日立, “超高速データベースエンジン Hitachi Advanced Data Binder :【導入事例】公共（防災シミュレーション）,” https://www.hitachi.co.jp/products/it/bigdata/platform/data-binder/case/public/index.html.

[15] 日立, “KEK 向けに、磁性材料の磁気構造に関するシミュレーションデータや量子ビーム実験データを解析・可視化するシステムの開発を支援,” https://www.hitachi.co.jp/New/cnews/month/2017/04/0417.html, 2017.

[16] 礒田他, “データベース管理システム及び方法,” 特許第 6823626 号公報, 2021.

[17] K. Goda, et.al., “Out-of-order Execution of Database Queries,” VLDB2020 keynote 3, http://www.vldb.org/pvldb/vol13/p3489-goda.pdf, 2020.

[18] Transaction Processing Performance Council (TPC), “TPC-H benchmark specification,” http://www.tpc.org/tpch/.

[19] 西川他, “産業ビッグデータアプリケーションモデルを用いた RDBMS と NoSQL の電力効率の初期比較,” 日本データベース学会和文論文誌, Vol. 17-J(4), 2019.

[20] U. Dayal, et.al., “An Approach to Benchmarking Industrial Big Data Applications,” Big Data Benchmarking: 5th International Workshop, WBDB 2014, 2014.

[21] Z. Xu, et.al., “Exploring power-performance tradeoffs in database systems,” In Proc. of 2010 IEEE 26th Int. Conf. on Data Engineering(ICDE 2010), pp. 485-496, 2010.

[22] B. Guo, et.al., “A green framework for DBMS based on energy-aware query optimization and energy -efficient query processing,” Journal of Network and Computer Applications, Vol. 84, pp. 118-130, 2017.

[23] 羅他, “プロセッサ動作モード制御による分析指向問合せ処理の省電力化効果の測定,” 日本データベース学会和文論文誌, Vol. 17-J(3), 2019.