アウトオブオーダ型データベースエンジン OoODE による クエリ処理性能の実験的評価

早水　悠登† 　合田　和生†† 　喜連川　優††

† 東京大学大学院情報理工学系研究科電子情報学専攻　〒 113–8656 東京都文京区本郷 7-3-1
†† 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: †{haya,kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし　アウトオブオーダ型データベースエンジン OoODE は，大規模なデータベースに対する選択的な分析クエリを高速に実行することを目して著者らが開発を進めるデータベースエンジンである．OoODE はクエリ処理を動的に複数のタスクに分解し，タスクを多重実行することでストレージに対して多数の非同期入出力を発行する．そして入出力が完了するたびに入出力対象データに係るデータベース演算処理をデータ駆動で実行する．よって，OoODE はマルチコアや多数のディスクといったハードウェア資源を効率的に活用することが可能であり，クエリ処理の高速化が期待される．本論文では，ミッドレンジクラスのマルチコアサーバおよびディスクストレージを用いた実験環境を構築し，OoODE プロトタイプ実装のクエリ処理性能を評価する実験を行った．
キーワード　OoODE, データベースエンジン, アウトオブオーダ型実行, マルチコア, 問合せ処理, 性能評価

Yuto HAYAMIZU†, Kazuo GODA††, and Masaru KITSUREGAWA††

† Information of Communication Engineering, Graduate School of Information Science and Technology, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113–8656 Japan
†† Institute of Industrial Science, the University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153–8505 Japan
E-mail: †{haya,kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

1. はじめに

社会が高度に情報化されるにつれて，人類の生み出すデジタルデータの量は爆発的な勢いで増加の一途を辿っている．米調査会社 IDC の報告では，2012 年に世界中で生成・複製されたデータ量の総計は 2.8 ZB (28 兆 PB) にも上り，2020 年までには 40 ZB に到達するとの予測がなされている [1].

このように増加を続ける膨大なデータを，経済活動や新たな社会的サービスの創出などに積極的に活用しようという機運が高まりを見せている．米国でオンライン通信販売やオークションを手がける eBay は，2011 年時点で 60 PB を超える規模のデータ解析基盤を運用しており，ウェブサイトの最適化などを目的として 1 日あたり 100PB 以上のデータ分析を行っている [2]．また，風力発電タービンの製造，販売，設置などを手がけるデンマークのエネルギー会社 Vestas Wind System A/S は 2 PB を超える気象，潮汐，地理情報，衛生写真などのデータを分析することで，風力発電タービンの最適な設置場所，面積，方角を効率的かつ迅速に決定することに役立てている [3]. McKinsey Global Institute はこのように新たな価値創出の源となる巨大なデータを Big Data と称し，その活用が経済や社会にもたらす影響の大きさを指摘している [4].

今後もデータの生み出される勢いが爆発的に増え続けてゆくであろうことを鑑みると，選択的なデータに対する分析技術の重要性が増してゆくことが予想される．例えばある個人に着目したデータ分析を行う場合には，絶対量としては一定のデータが分析対象となるが，利用者数が増加するほどその相対的な割合は小さくなる．また情報サービスの多様化や情報家電など自らデータを生み出す機器が今後も発展してゆくことを考えると，データの量だけではなくデータの種類も増加することは想像に難くない．その場合，ある特定の種類のデータに着目した分析を行う場合，そのデータが全体に占める割合は相対的に小さくなる．つまり，蓄積された膨大なデータの活用方法として，関心のあるごく一部のデータに対する分析処理がより広く行われるようになると考えられる．

著者らが開発を進めるアウトオブオーダ型データベースエンジン OoODE は，前述のような選択的データ分析クエリを高速に実行することを目するデータベースエンジンである [5]. OoODE はクエリの実行を動的に複数のタスクに分解し，これらのタスク処理を並列に実行することでクエリ処理を行う．各タスクの処理に必要とするデータに対しては非同期 I/O を発行し，I/O が完了するたびに I/O 対象データに係る演算処理の実行を駆動する．この実行原理を最大限に適用してクエリ処理を行った場合には，論理的に依存関係のない I/O はそれぞれ並列に発行可能であり，I/O 完了に伴って実行される演算処理もまた並列に実行可能である．これにより，OoODE はストレージの有する並列 I/O 処理性能やマルチコアプロセッサの演算性能を高効率に活用可能であり，従前のデータベースエンジンに対して高いクエリ処理性能を実現することが期待される．本論文では，ミッドレンジクラスのマルチコアサーバおよびディスクストレージからなるデータベースシステムにおいて，著者らが開発を進める OoODE プロトタイプ実装のクエリ処理性能を評価した．

本論文の構成は次の通りである．2 節では大規模データに対して選択率の低いクエリを実行する際の処理性能について，従前のデータベースエンジンに対する OoODE の高速性を考察する．3 節ではクエリ処理性能の評価に用いる実験環境について説明し，4 節では当該環境におけるストレージの基本性能測定の結果を示す．5 節では OoODE の評価実験結果を示し，6 節で本論文をまとめる．

2. 選択的な分析クエリの処理性能

一般にデータベースシステムにおいてはクエリ処理のためのアクセスパスや結合方式は複数用意されており，その組み合わせによってクエリ実行プランが表現され，データベースエンジンがクエリ実行プランに基づいてクエリ処理を実行する．クエリ実行プランの生成は，問い合わせ対象であるリレーションの選択率に基づいてクエリオプティマイザが行う．OoODE が対象とするような選択率の低いクエリの場合，通常は索引によるリレーション走査とネステッドループ結合の組み合わせが用いられる．

従前のデータベースエンジンにおいては，多くの場合クエリ処理における I/O とその結果に対する演算は逐次的に行われる．つまり，演算対象となるデータに対して同期的に I/O を発行し，その I/O の完了を待ってから演算を実行する，という過程の繰り返しによってクエリ処理が進行する．データの I/O，および演算が実行される順序は，同じクエリ処理については事前にプログラムされた通りの決定的な順序で行われる．このような実行方式をインオーダ型と呼ぶことにする．

論理的には，インオーダ型データベースエンジンが発行するアウトスタンディングな I/O の数は高々1 である．しかし物理的に発行されるアウトスタンディングな I/O の数はその限りではない．OS のファイルシステムやストレージコントローラは，連続する I/O に規則性が見出される場合にはそれに基づいて先読みを行うため，インオーダ型データベースエンジンから発行

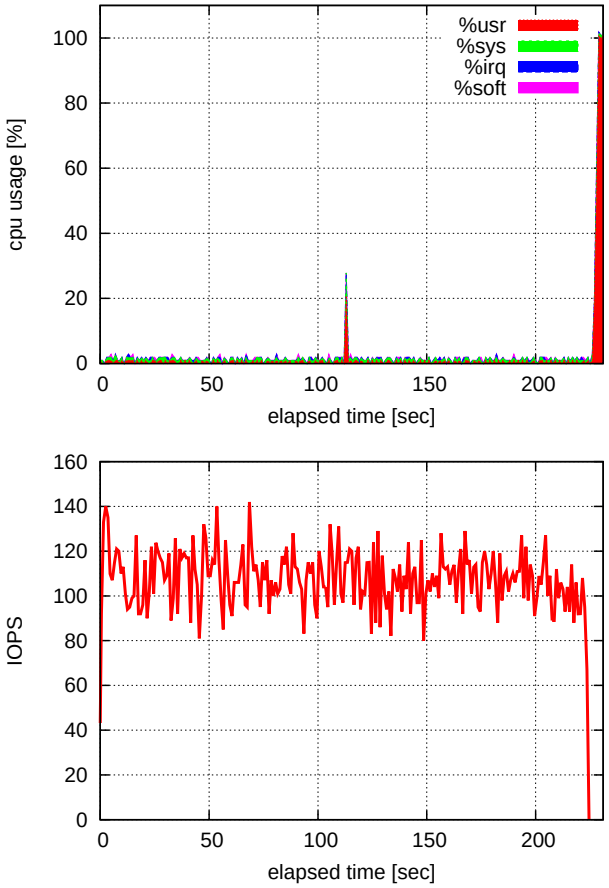

図 1　約 10 億行のデータから 0.01%のデータを選択するクエリを実行した場合の CPU 利用率と IOPS の経時変化

される 1 つの I/O が，実際にはストレージコントローラやディスクに対する複数の I/O となる場合もある．リレーションの全件走査に伴うシーケンシャルアクセスでは，前述の機構によりストレージの帯域を有効に活用することができる．

しかし，一般に索引を用いたネステッドループ結合のクエリ処理における I/O のパターンはランダムアクセスであり，インオーダ型で当該処理を行った場合にはディスクへの I/O に要する時間がほぼすべて直列化されクエリ実行時間に反映される．現在のコンピュータシステムにおいては，多くの場合 1 タプルの処理にプロセッサが要する時間は 1μs に満たない一方で，ディスクに対する 1 回の I/O 完了には数 ms を要する．結果として，インオーダ型データベースエンジンが選択率の低いクエリ処理を実行する場合，クエリ実行時間がほぼ I/O 待ちで占められ，アウトスタンディング I/O の数もほぼ 1 であり，プロセッサの利用率が極めて低い上にストレージの並列 I/O 処理性能が全く活用されない状態に陥ると予想される．実際に約 10 億行のデータが格納されたデータベースに対して，0.01%のデータを選択するクエリを実行した場合の CPU 使用率と IOPS を図 1 に示す．このクエリは 3. 節にて説明する評価実験環境で，MySQL 5.5 において実行したものである．クエリ実行時間は 224 秒であり，システムが有する 24 コア全てを利用した場合の CPU 利用率を 2,400%としたときに，当該クエリの実行では平均 1.5%しか利用されていない．またストレージは 160 台

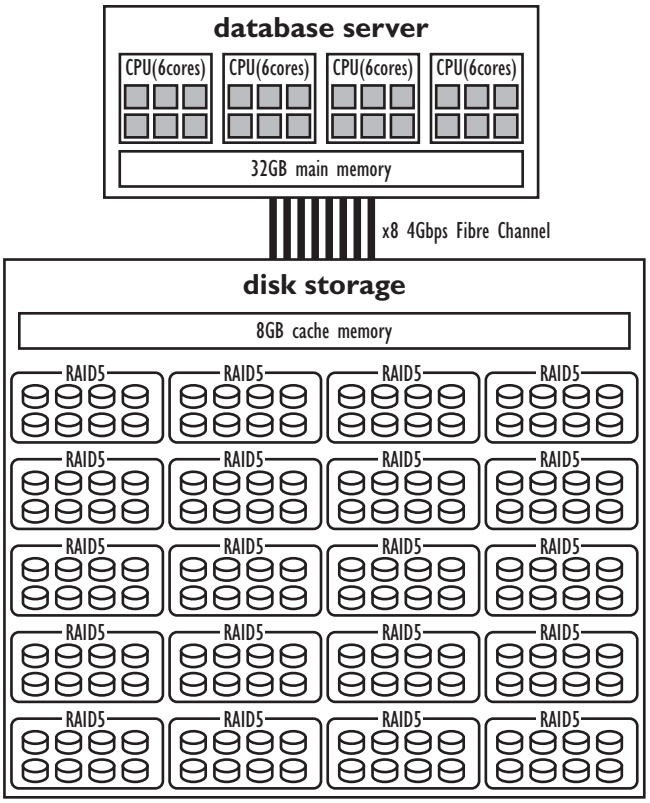

図 2　評価実験環境の概要

のディスクで構成され 50,000 IOPS を超えるランダムアクセスの処理性能を持つにも関わらず，当該クエリの実行に際して発行された I/O は平均 107 IOPS であった．単純にストレージの帯域という観点からすると，107 IOPS はディスク 1 台でも十分達成可能な数値である．

ディスクのランダムアクセスに要する応答時間は 2000 年以降ほとんど改善しておらず，依然としてプロセッサの処理速度とストレージの応答時間の差は大きい．つまり，ストレージのランダムアクセスの応答時間に律速されるインオーダ型という実行原理に依拠する限り，選択率の小さいクエリ処理の大幅な改善は見込まれないばかりか，その実行時間はデータの絶対量が増えるほど長くなるであろう．つまり，大規模なデータに対する選択的なクエリを高速に処理するためには，データベースエンジンの実行原理自体を大きく転換する必要がある．そこで，喜連川らはクエリ処理を動的にタスク分割して実行を非同期化し，多数のプロセッサコアやストレージの並列 I/O 処理性能を高効率に利用するアウトオブオーダ型データベースエンジン OoODE を考案し，エントリークラスのサーバおよびストレージにおいてその有効性を確認した [5]．本論文では，マルチスレッド化された OoODE プロトタイプ実装を用いて，ミッドレンジクラスのサーバおよびストレージを用いた環境においても OoODE がその性能を十分に活用し，高いクエリ処理性能を実現することができるか，実験により評価を行った．

3. 評価実験環境

OoODE プロトタイプ実装の性能評価を実施するにあたり，ミッドレンジクラスのサーバとディスクストレージを備える実験システムを構築した．サーバは，4 プロセッサ 24 物理コア(注1)と 32 GB の主記憶を搭載し，64bit 版 RedHat Enterprise Linux Server 5.8(注2)が OS として動作する．Hyper Threading は無効化され，1 物理コアは OS から 1 論理コアとして認識される．ディスクストレージは 160 台の 450GB 15,000rpm FC HDD と，8GB のキャッシュメモリをもつディスクシステム IBM DS5300 から成る．ディスクシステム内では 8 HDD ごとに 1 つの RAID5 ディスクグループが編成され，各ディスクグループが 1 つの論理ユニット (LU) として，合計 20 の LU がサーバから認識される．サーバとディスクシステムは，8 本の 4 Gbps ファイバチャネルにより接続される．

3.1 ディスクストレージの基本 I/O 性能

OoODE がストレージの並列 I/O 処理性能を高効率に活用可能であるかを評価するためには，ストレージの I/O 処理性能を正確に把握する必要がある．OoODE プロトタイプ実装では，MySQL の InnoDB ファイルフォーマットを採用しているため，データベースのページサイズは 16KB である．すなわち，OoODE から発行される I/O のワークロードは 16KB ランダム読込みとなる．そこで OoODE プロトタイプ実装の性能評価に先立ち，評価環境のストレージが有する 16KB ランダム読込みの I/O 性能について測定を行った．

基本 I/O 性能測定には，Linux の非同期 I/O 機能を利用して対象デバイスに I/O を発行し続けるテストプログラムを利用した．テストプログラムは `open(2)` システムコールを `O_DIRECT` フラグ付きで呼び出し，Linux のページキャッシュを迂回して直接デバイスへと I/O を発行する．また本論文の実験において，OoODE がデータベース領域として利用するのは，各 LU の先頭から 512GB をソフトウェア RAID0 によりストライピングした領域である．よって，ソフトウェア RAID0 を利用せずに各 LU に直接 I/O を発行する場合にも，その I/O 発行範囲は先頭から 512GB の領域とした．

3.1.1 各 LU を直接利用する場合の I/O 性能

図 3 は，1 LU のみをアクセス対象として，テストプログラムが発行するアウトスタンディング I/O 数を変化させた場合の IOPS 性能を示す．アウトスタンディング I/O 数が 128 以下ではアウトスタンディング I/O 数の増加に伴って IOPS が上昇するが，128 のときに 3,672 IOPS であり，128 以上の場合にはほぼ一定の IOPS をとる結果となった．これより，1 LU の有するランダム読込み性能を活用するためには，アウトスタンディング I/O 数が 128 以上必要であることがわかる．

図 4 は，1 LU あたり 128 アウトスタンディング I/O を発行し，利用 LU 数を変化させた場合のシステム全体の IOPS 性能を表すグラフである．測定に際してはソフトウェア RAID は利用せず，各 LU ごとにテストプログラムを立ち上げ，I/O 発行を行った．利用 LU 数にほぼ比例する形でシステム全体の

(注1)：1 プロセッサあたり 6 物理コアを有する Intel Xeon X7460 2.66GHz を 4 プロセッサ搭載
(注2)：OS カーネルは Linux 2.6.18-308.el5

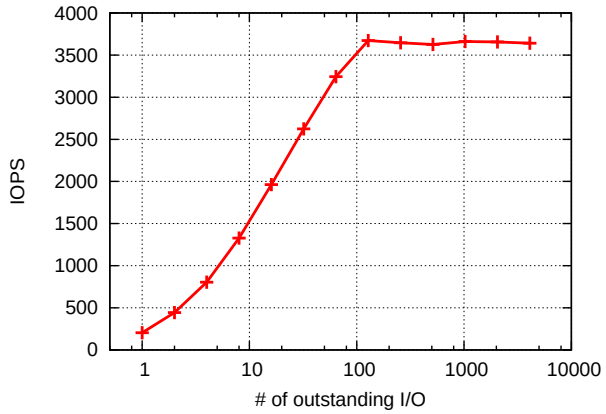

図 3　1 LU 利用時，16KB ランダム読込みにおけるアウトスタンディング I/O 数と IOPS 性能の関係

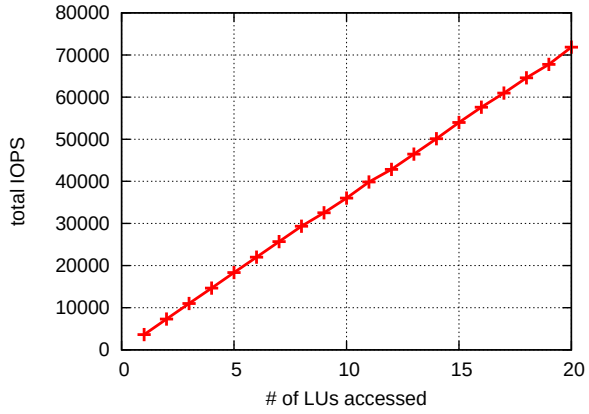

図 4　同時利用 LU 数と IOPS 性能の関係 (1 LU あたり 128 アウトスタンディング I/O を発行)

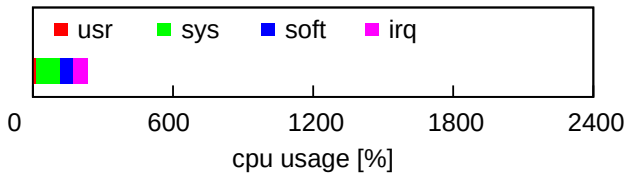

図 5　1 LU あたり 128 アウトスタンディング I/O，20 LU 利用時の CPU 利用率 (システム全体で 2,400%)

IOPS が増加し，20 LU 利用時には 71,844 IOPS となった．また 20 LU 利用時の CPU 利用率を図 5 に示す．システム全体で 24 コア 2,400%のうち利用されているのは 234%である．つまり，本環境が有するストレージの I/O 性能を活用するためには，I/O 発行に関わる処理だけで最低 3 コア分の処理性能が必要とされることを意味する．

3.1.2　ソフトウェア RAID0 を用いた場合の I/O 性能

図 6 は，20 LU をソフトウェア RAID0 によりストライピングした領域に対して，テストプログラムから発行するアウトスタンディング I/O 数を変化させた場合の IOPS 性能を示す．1024 まではアウトスタンディング I/O 数の増加に伴う IOPS 性能の大きな向上がみられるが，それ以上のアウトスタンディング I/O 数では IOPS 性能の増加は比較的小さい．ソフトウェア RAID0 を用いない場合には，20 LU に対してそれぞれ 128

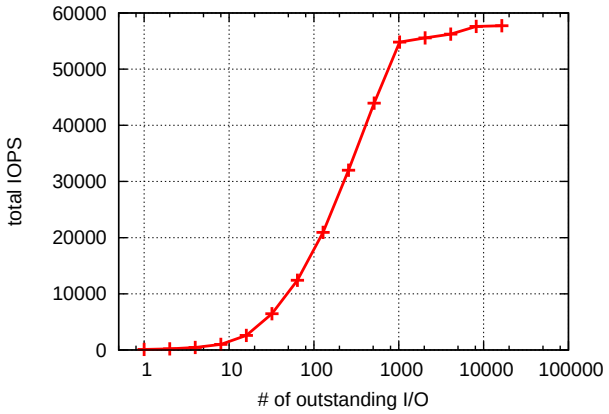

図 6　20 LU をソフトウェア RAID0 によりストライピングした場合の IOPS 性能

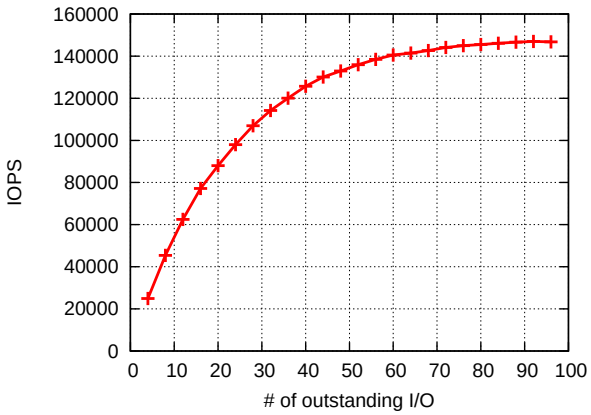

図 7　サーバ・ストレージコントローラ間接続の IOPS 性能

アウトスタンディング I/O，すなわち計 2560 アウトスタンディング I/O を発行したときに 71,844 IOPS であった一方で，ソフトウェア RAID0 を利用した場合には，4096 アウトスタンディング I/O を発行したときに 56,225 IOPS であり，ソフトウェア RAID0 のオーバヘッドにより 2 割程度の IOPS 性能低下が確認された．

またソフトウェア RAID0 を利用している場合に，ストレージコントローラとサーバ間でやり取りが可能な IOPS 性能の上限値，すなわちサーバ・コントローラ間の実効的な帯域幅を明らかにするために，あらかじめソフトウェア RAID0 領域の先頭 1GB を読みだしてコントローラのキャッシュに載せた状態で，当該の 1GB 領域に対してテストプログラムから I/O を発行し，アウトスタンディング I/O 数の変化にともなう IOPS 性能を測定した．結果のグラフを図 7 に示す．アウトスタンディング I/O 数の増加にともなって IOPS 性能は増加するが，増加の程度は徐々に小さくなり，アウトスタンディング I/O 数が 90 前後でほとんど増加しなくなる．この測定における最大値は，アウトスタンディング I/O 数が 92 のときの 146,964 IOPS であった．

4.　TPC-H による性能評価実験

3. 節に示した環境において，OoODE のクエリ処理性能を評価する実験を行った．評価に用いた OoODE プロトタイプ実装

```
SELECT count(*), sum(l_quantity)
  FROM customer, orders, lineitem
  WHERE
    c_custkey < [value]
    AND c_custkey = o_custkey
    AND l_orderkey = o_orderkey;
```

図 8　評価に用いた TPC-H Query 3 の修正クエリ: [value] を変更することで選択率の調整を行った

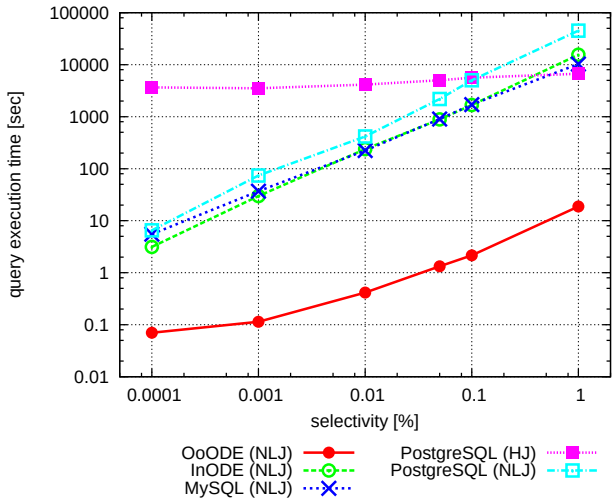

図 9　クエリ処理 (結合) 方式ごとの選択率とクエリ実行時間の関係

は，MySQL の InnoDB ファイルフォーマットを採用している．また当該実装はマルチスレッド化されており，事前の予備実験の結果に基づいて，本性能評価実験においては 12 スレッド，1 スレッドあたり最大で 1024 アウトスタンディング I/O を発行する設定を用いることとした．なお，OoODE プロトタイプ実装の詳細については [6] を参照されたい．

評価実験には意思決定支援系のベンチマークの業界標準である TPC-H ベンチマークを利用した．データベースの規模を表す Scale Factor は 100 であり，InnoDB フォーマットにロードした状態のファイルサイズは索引を含めて 250GB である．データベースファイルはストレージの全 20 LU の先頭 512GB をソフトウェア RAID0 によってストライピングした領域に配置した．評価に用いたクエリは，Query 3 の WHERE 句を変更し，選択率を実験にあわせて適宜変更したものである (図 8)．クエリの実行に際しては，毎回データベースバッファおよびファイルシステムキャッシュを消去した上でクエリ実行を開始し，その実行時間を測定した．

4.1　選択率とクエリ実行時間

選択率を変化させた場合のクエリ実行時間を，クエリ処理方式ごとに測定した結果を図 9 に示す．凡例における NLJ はネステッドループ結合，HJ はハッシュ結合を意味する．また凡例の “OoODE” は OoODE プロトタイプ実装を用いた場合を，“InODE” は OoODE プロトタイプ実装のインオーダ型動作版を用いた場合を，“MySQL” は MySQL 5.5 を用いた場合を表す．またクエリ実行時間の参考として，PostgreSQL 9.2 によるクエリ実行時間の測定も行った．測定に用いたデータベースは，ソフトウェア RAID0 領域を ext4 ファイルシステムでフォーマットした上に，TPC-H ベンチマーク Scale Factor = 100 のデータをロードしたものを利用した．

ネステッドループ結合におけるクエリ処理コストは選択するデータ量に比例する．実際に InODE，MySQL，PostgreSQL のいずれにおいても選択率とクエリ実行時間はほぼ比例関係にあるといえる．OoODE のクエリ処理はインオーダ型のネステッドループ結合処理が多重化された分だけ性能向上すると期待され，実際に選択率 1.0%の場合には実行時間が 18.73 秒で MySQL に対して 548 倍の性能向上率，0.1%の場合には実行時間が 2.15 秒で MySQL に対して 788 倍の性能向上率が確認された．また選択率がある一定以上よりも小さくなると，クエリ処理におけるデータ並列性がストレージの並列 I/O 処理性能を下回り，OoODE の性能向上率は小さくなると予想されるが，この傾向も測定結果のグラフと一致する．

ハッシュ結合はクエリの対象とするリレーションに対して全件走査を行うため，選択率にかかわらず必ず全件走査のコストが発生するが，選択率増加にともなう処理コストの増加量はネステッドループ結合に比べて軽微である．実際に PostgreSQL におけるハッシュ結合は選択率にかかわらず 3,600 秒以上を要したが，一方で選択率の増加に応じた実行時間の増加は全方式の中で最も小さく，選択率 1.0%の場合には OoODE 以外の全ての方式の中で最もクエリ実行時間が短い．PostgreSQL と OoODE/InODE/MySQL ではデータベースのフォーマットやファイルシステム構成が異なるため直接的な比較はできないが，InnoDB フォーマットのデータベースに対してハッシュ結合を行う場合にも実行時間は同じ傾向にあると考えられる．すなわち，インオーダ型のハッシュ結合に対する OoODE(のネステッドループ結合) の性能向上率は，選択率が大きいほど低くなる傾向にあると考えられる．

本実験の結果より，選択率 0.0001%から 1.0%の範囲においては，測定に用いた OoODE，InODE，MySQL，PostgreSQL のうち，OoODE が他のいずれよりもクエリ実行時間が短いことが確認された．またインオーダ型のクエリ処理に対する OoODE の性能向上率は，ネステッドループ結合については選択率が大きいほど高く，ハッシュ結合については選択率が小さいほど高くなる傾向があると考えられ，実験結果からもそのことが確認できた．

4.2　OoODE のクエリ処理性能

選択率 1.0%のクエリについて，OoODE で実行した際の IOPS の経時変化を図 10 に，CPU 使用率の経時変化を図 11 に示す．

図 10 より，IOPS はクエリ開始直後に上昇し，約 140,000 IOPS に到達した後はクエリ終了直前までわずかに増減しながらその値をほぼ維持する形で推移した．システムの基本性能測定によれば，サーバ・ストレージコントローラ間の実効的な帯域幅は 146,964 IOPS であったから，OoODE はこの帯域の約 95%を活用することができたといえる．OoODE でクエリを実行した際の IOPS の値が基本性能測定におけるランダム読込

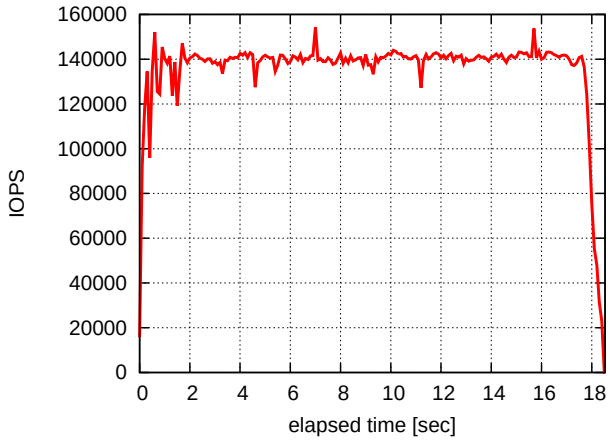

図 10　選択率 1.0%のクエリを OoODE で実行した際の IOPS 経時変化

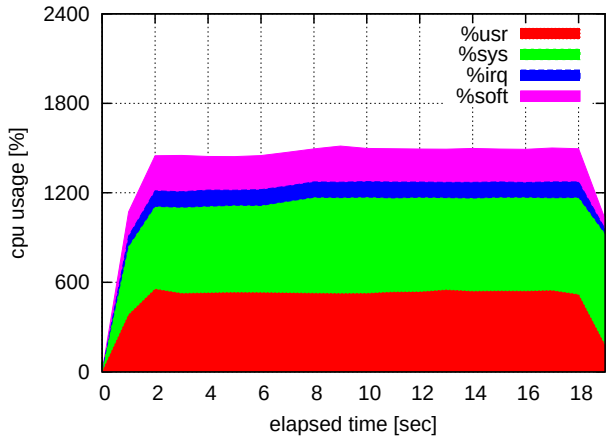

図 11　選択率 1.0%のクエリを OoODE で実行した際の CPU 利用率経時変化

み性能 56,225 IOPS を大幅に上回る理由は，ストレージにおける各ディスクヘッドのシーク範囲の違いに由来するものと考えられる．基本性能測定においては，ソフトウェア RAID0 でストライピングを行った 512GB × 20=10TB の領域全体に対してランダムに I/O を発行するのに対し，選択率 1.0%のクエリでアクセスされるデータ量はおよそ 2.5GB 程度である．それに加えて，索引やリレーションがそれぞれディスクのある位置にまとまって配置されている場合には，I/O が発行されるパターンにランダム性はあるものの，データ全体のうち実際にアクセスされる範囲は極めて局所的となる．すなわち，ごく一部のディスクのトラックにアクセスが集中するため，1 回あたりのディスクへの I/O 処理に必要なシーク時間が大幅に減少し，結果としてディスクへの I/O 処理よりもサーバ・ストレージコントローラ間の実効的な帯域幅がボトルネックとなったと考えられる．

また CPU 利用率に関しては，図 11 に示すようにクエリ開始直後から約 1,400%まで上昇し，その後はクエリ終了直後までわずかに増減しながらその値をほぼ維持する形で推移した．クエリ実行全体を通した平均は 1,478%であった．データの演算処理に相当する%usr については，平均 532%であり，非同期 I/O 完了から駆動される演算処理が複数コアの性能を活用可能であることが確認できる．一方で全体としては CPU 資源の 4 割近くは利用されていない．これは本評価環境においてストレージの I/O 帯域がボトルネックとなっているためであり，より高帯域なストレージを用いた場合には，OoODE はさらに高い CPU 利用率を示すことが予想される．

このように OoODE はストレージ，プロセッサの性能を活用することができ，その結果として MySQL に対して 547 倍という圧倒的に高速なクエリ処理性能を実現可能であることが確認できた．また本評価環境においては OoODE のクエリ処理性能はストレージ性能にほぼ律速されている状態であり，より高帯域なストレージを利用することでさらに高速なクエリ処理性能を達成可能であると期待される．一方でインオーダ型実行ではそもそもストレージ帯域を活用することができないため，ストレージが高帯域化するほど相対的に OoODE の性能向上率もまた高くなると考えられる．

5. おわりに

著者らが開発を進めるアウトオブオーダ型データベースエンジン OoODE は，クエリ処理を動的に複数のタスクに分解することで，大量に非同期 I/O を発行してストレージの並列 I/O 処理性能を活用し，また I/O 完了により駆動される演算処理を割り当てることでマルチコアを活用して，高速にクエリ処理を実行することが期待される．本論文では，ミッドレンジクラスのマルチコアサーバおよびディスクストレージからなる環境を構築し，TPC-H ベンチマークを用いて OoODE のクエリ処理性能を評価する実験を行った．TPC-H Query 3 の選択率を 0.0001%から 1%まで変化させて，InODE(OoODE のインオーダ型動作版)，MySQL，PostgreSQL のそれぞれと OoODE のクエリ処理性能を比較した場合，OoODE は全ての選択率においていずれのものよりも高いクエリ処理性能を示した．特に選択率 1.0%のクエリにおいては，当該環境におけるストレージ性能の 95%を活用し，同じデータベースフォーマットを用いている MySQL に対して 547 倍の性能向上率を達成可能であることが確認された．この実験結果は，ストレージが高帯域化するほど OoODE はより高いクエリ処理性能を実現可能であることを示唆している．

謝　辞

本研究の一部は内閣府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」，および日本学術振興会科学研究費補助金 (特別研究員奨励費) 24·8381 の助成により行われた．

文　　献

[1] John Gantz and David Reinsel. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. Technical report, IDC, Dec 2012.

[2] Tom Fastner. Big data at eBay – A Technical Discussion, Oct 2011. 2011 Teradata PARTNERS Conference & Expo.

[3] Lars Christian. Vestas Wind Systems Turns to IBM Big Data Analytics for Smarter Wind Energy, Oct 2011. IBM Information on Demand 2011.

[4] Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute, May 2011.

[5] 喜連川優, 合田和生. アウトオブオーダー型データベースエンジン OoODE の構想と初期実験. 日本データベース学会論文誌, Vol. 8, No. 1, pp. 131–136, Jun 2009.

[6] 合田和生, 豊田正史, 喜連川優. アウトオブオーダ型データベースエンジン ooode の試作とその実行挙動. 第 5 回データ工学と情報マネジメントに関するフォーラム, 2013. (to appear).