アウトオブオーダ型データベースエンジンにおける 2 表結合問合せの処理時間見積り方式の提案と評価

土田 隼之† 清水 晃† 田中 美智子† 藤原 真二‡ 茂木 和彦‡ 合田 和生¶ 喜連川 優¶§

†㈱日立製作所 中央研究所 〒244-0817 神奈川県横浜市戸塚区吉田町 292 番地
‡㈱日立製作所 情報・通信システム社 IT プラットフォーム事業本部
〒244-0817 神奈川県横浜市戸塚区吉田町 292 番地
¶東京大学生産技術研究所 〒153-8503 東京都目黒区駒場 4-6-1
§国立情報学研究所 〒101-8430 東京都千代田区一ツ橋 2-1-2
E-mail: †{ takayuki.tsuchida.aj, akira.shimizu.wv, michiko.tanaka.ry }@hitachi.com,
‡{ shinji.fujiwara.yc, kazuhiko.mogi.uv }@hitachi.com, ¶{ kgoda, kitsure }@tkl.iis.u-tokyo.ac.jp

あらまし 実世界で発生するデータが爆発的に増加しており，ビッグデータ利活用への期待が高まっている．このような中，我々は内閣府最先端研究開発支援プログラムにおいて，アウトオブオーダ型データベースエンジン（OoODE）と称する実行原理に基づく超高速データベースエンジンの研究開発を推進している．OoODE はアンフォールドした処理を複数並行して進めるため，インデックスアクセス時に高多重リード要求を行う．ストレージ性能の限界に達する場合， その影響を考慮しないと 2 表結合であっても処理時間見積りに誤差が生じる．本論文では，OoODE の特性を踏まえた 2 表結合問合せの時間見積り方式の検討と評価結果について報告する．
キーワード OoODE，アウトオブオーダ型，データベースエンジン，結合方式，処理時間見積り

Takayuki TSUCHIDA† Akira SHIMIZU† Michiko TANAKA† Shinji FUJIWARA‡
Kazuhiko MOGI‡ Kazuo GODA¶ and Masaru KITSUREGAWA¶§

†Central Research Laboratory, Hitachi, Ltd. 292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817 Japan
‡IT Platform Division Group, Information & Telecommunication Systems Company, Hitachi, Ltd.
292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817 Japan
¶Institute of Industrial Science, The University of Tokyo 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505 Japan
§National Institute of Infomatics 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan
E-mail: †{ takayuki.tsuchida.aj, akira.shimizu.wv, michiko.tanaka.ry }@hitachi.com,
‡{ shinji.fujiwara.yc, kazuhiko.mogi.uv }@hitachi.com, ¶{ kgoda, kitsure }@tkl.iis.u-tokyo.ac.jp

1. はじめに

近年，実世界で発生するデータが爆発的に増加しており，ビッグデータの利活用への期待が高まっている．IT 専門調査会社 IDC によると，2012 年の国内外付型ディスクストレージシステムの出荷容量は 903PB であり，2012 年～2017 年のディスクストレージシステムの出荷容量成長率の予測値は 41.1%に達する[1].

このような大規模なデータに対する分析ニーズが高まっている状況の中，我々は，内閣府最先端研究開発支援プログラムにおいて，アウトオブオーダ型データベースエンジン (OoODE) と称する超高速データベースエンジンの研究開発を進めている[2]. 従来型のデータベースエンジンでは，レコードのフェッチからレコード処理まで，プログラムされた決定的な順序で問合せ処理を進めていくのに対し，OoODE は，互いに独立な処理を複数駆動し，ストレージ帯域などの資源を十分に活用して実行可能な順に処理を行う．そのため，インデックスアクセス時に高多重リード要求を行い，高速に処理を行うことが出来る[3][4][5].

リレーショナル型のデータベースエンジンへの問合せ言語には，SQL (Structured Query Language) が広く用いられる．SQL は宣言的な言語のため，データベースエンジンで実行プラン(内部的な処理実行方法)を決める必要がある．実行プランを複数作成可能な場合，最も処理時間が短いと思われる実行プランを選択する必要がある．この際，実行プランの処理時間見積りが重要となり，OoODE においても例外ではない．

リレーショナル型データベースにおける問合せ実行の基本処理の 1 つである結合(ジョイン)処理の実行方式として，現在ではネストループジョイン(NLJ) と

ハッシュジョイン(HJ)が一般的に用いられる．ここで結合処理方式として NLJ の処理時間が HJ より長くなるのは，処理実行時に選択されるレコード数が多い場合である．OoODE においては，前述の高多重リード要求の効果により，従来型エンジンと比較して NLJ の処理時間の方が短い条件がよりレコード数が多い範囲に拡大する．この点を鑑みると，計算機システムの上限 I/O 性能で処理が進められるとして処理時間見積りを行い結合処理方式を選択するのが良いと考えられる．

ただ，OoODE は互いに独立な処理を実行可能な順に処理を行うため，連続アクセスを行うテーブルスキャンとランダムアクセスを行うインデックススキャンを同時に行う混在アクセスを行う場合がある．この場合，ストレージ性能の限界に達してしまい，その影響を考慮しないと 2 表結合であっても処理時間見積りに誤差が生じると考えられる．

本論文では，上記を踏まえて行ったリレーショナルデータベースエンジンの基本的な処理である 2 表結合における OoODE の処理時間見積り方式の検討と評価結果について報告する．本論文の構成は以下の通りである．2 章では OoODE における 2 表結合問合せの処理時間見積り方式として，従来型データベースエンジンと同様に I/O 処理性能が固定であると近似した OoODE の処理時間見積り方式と，互いに独立な処理を実行可能な順に処理を行う OoODE の動作特性を考慮した処理時間見積り方式の 2 種類を説明する．3 章では実際の処理時間と比較することで各方式の評価を行い，最後に 4 章で本論文をまとめる．

2. アウトオブオーダ型データベースエンジンの 2 表結合問合せの処理時間見積り

アウトオブオーダ型データベースエンジン OoODE における問合せの見積り処理時間 (以降，見積り処理時間を「コスト」と呼ぶ) の計算方法，特に本論文では 2 表結合の問合せについて述べる．

リレーショナルデータベースエンジンへの問合せ言語として広く使われている SQL は，取得対象のデータ属性のみ表現している．このため，表の結合順序などの取得対象データの生成手順である実行プランをデータベースエンジンで決定する必要がある．問合せから実行プランを決定する機能を問合せ最適化と呼ぶ．1 つの問合せに対して実行プランが複数存在する場合があり，その場合問合せ最適化では実行プランを一つ選ぶ必要がある．例えば，複数の表を結合する場合，ネストループジョイン (NLJ) やハッシュジョイン (HJ) などの実行方式を選択可能である．表のサイズやレコードの件数，問合せの条件により適切な結合処理方式が異なるため，問合せ最適化に関しては幅広く研究がおこなわれている．コストが小さい実行プランを選ぶコストベース最適化[6]がその代表例である．コストベース最適化では，適切な実行プランを選択するためにコストを精度良く見積ることが重要となる．

OoODE の問合せ言語に SQL を用いる場合，従来と同様に問合せ最適化が必要となる．コストベース最適化を用いる際に課題となるのが，OoODE における処理コストの見積りである．

以下，OoODE におけるコストベース最適化のために，SQL で基本的な処理である 2 表結合のコスト見積り方式の検討を行う．初めに，簡素なコストモデルを構築すること目的に，従来型データベースエンジンと同様に I/O 性能が I/O 処理方式により固定であると近似した OoODE のコスト見積り方式の検討を行う．次に，より見積り精度を高めることを目的に，互いに独立な処理を実行可能な順に処理を行う OoODE の動作特性を考慮したコスト見積り方式の検討を行う．

2.1. アウトオブオーダ型データベースエンジンの動作

OoODE では，実行順序に依存関係のない DB 処理をレコード単位レベルまで細分化して複数のタスクに割り当て，これらのタスクを並列動作させて非同期 I/O を発行し，I/O 完了順に検索処理を実行する．これにより，超高多重な I/O 処理を可能とし，ストレージ装置の持つ I/O 性能を最大限引き出すことで処理時間が大幅に短縮可能である．

2 表結合時の動作を説明する．説明に用いる問合せを図 1 に示し，問合せの実行プランを図 2 に示す．なお，ここでは意思決定支援システムのデータベースベンチマークである TPC-H[7]の表定義を想定している．

```
SELECT o_orderstatus,
        sum(l_extendedprice), count(*)
FROM   lineitem, orders
WHERE l_orderkey = o_orderkey
       and l_returnflag = 'R'and l_shipmode = 'AIR'
       and l_shipinstruct = 'DELIVER IN PERSON'
       and l_quantity <= 30   ← この条件で
GROUP BY o_orderstatus          絞込み率を調整
```

図 1：2 表結合の問合せ例

HJ プラン (図 2 の(a)) では，OoODE は lineitem 表のテーブルスキャン (TS) と Orders 表の TS は同時に処理できる．しかし，HJ において lineitem 表の処理対象となる全レコードのハッシュ値が作成されなければ orders レコードを使った Probe 処理を完了できない．そこで，大局的に見た場合 Build 側処理を行った後に Probe 側処理を行うとする．

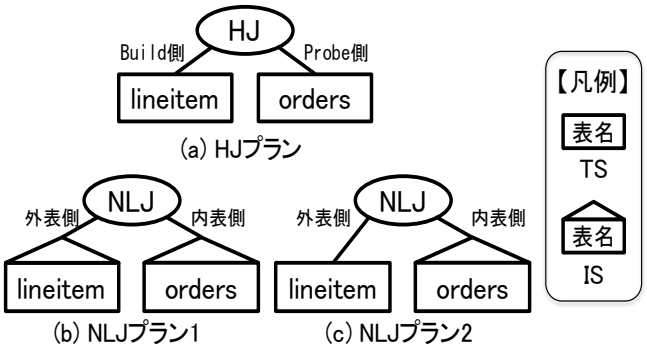

図 2：2 表結合問合せの実行プラン例

NLJ プランの処理は，外表側レコードを用いて内表側レコードを探すといった依存関係以外は互いに独立なレコード単位の処理である．このため，OoODE ではレコードごとに外表側の処理と内表側の処理を同時に実行する．外表側処理と内表側処理がともにインデックススキャン (IS) である NLJ プラン 1 (図 2 の(b)) は全てランダムアクセスとなるが，外表側処理が TS であり，内表側処理が IS である NLJ プラン 2 (図 2 の(c)) では，TS の連続アクセスの間に IS のランダムアクセスが混ざることになる．連続アクセスとランダムアクセスが混ざる I/O パターンを混在アクセスと呼ぶ．

OoODE は，実行プランに内在する並列性を活用し，高多重なリード要求を行うことでストレージ装置の性能を引き出している．このため，外表側の選択率が高く，引き出せる I/O 多重度が高い場合により高い性能を発揮する特徴がある．[4]

2.2. 固定 I/O 性能による OoODE コスト見積り方式

最初に，従来型のデータベースエンジンにおける処理コストについて簡単に述べる．従来型のデータベースエンジンでは，基本的に CPU 処理と I/O 処理が逐次的に行われるため，CPU 処理のコスト (CPU コスト) と I/O 処理のコスト (I/O コスト) の和を問合せのコストと考える．CPU コストは，問合せで必要な CPU 処理の総時間を見積り，処理を行うコア数で割った値と考える．I/O コストは，レコードへのアクセス方式によって計算方法が異なる．表を構成するページを連続的にアクセスする TS では，表サイズを単位時間当たりのデータ転送量 (これをスループットと呼ぶ) で割った値が I/O コストとなる．一方，インデックスを使ってレコードをアクセスする IS では，多くの場合ページ単位でランダムにアクセスするため，総 I/O 回数を単位時間当たりの処理 I/O 数 (これを IOPS (I/O per second) と呼ぶ) で割った値を I/O コストと考える．TS と IS を行う問合せの場合は，TS の I/O と IS の I/O が逐次的に処理されていると考え，TS の I/O コストと IS の I/O コストの和が問合せの I/O コストと考える．

従来型データベースエンジンとの差分の形で OoODE のコスト見積り方式を考える．一番大きな違いは， OoODE では CPU 処理と I/O 処理をオーバーラップさせて動いていると考えられるため，CPU コストと I/O コストの大きい方が問合せのコストとなる．

ここで，CPU コストの計算は，基本的に従来型データベースエンジンと同様の方式により見積もれると考えられる．一方の I/O コストに関しては，OoODE における I/O 処理挙動は従来型データベースエンジンと大きく異なるため，どのように扱うかが問題になる．

2.1 節で述べたように，OoODE は実行プランから引き出せる多重度が高い場合により高い性能を発揮する．そこで，まず簡素なコスト見積り方式として I/O 性能が I/O 処理方式により固定であると近似する I/O コスト見積り方式を考える．I/O コストの計算には，計算機システムの最大スループット (上限スループット) と計算機システムの最大 IOPS (上限 IOPS) を用いて計算する．

なお，選択率が小さい場合は処理すべきレコード数が少なく，それが原因で I/O 多重度を出せずに上限 IOPS より低い性能で結合処理を行う可能性があり，その場合 NLJ の I/O コストは実際より小さく見積もられる．しかし，OoODE の問合せ処理における IOPS 性能がシステム上限性能から乖離するほど選択率が低い場合は，一般に処理時間が短いためプラン選択を誤っても実用上は問題無いと考えられる．

従来型データベースエンジン向けコスト見積りを，OoODE に適用した場合の I/O コスト式を下に示す(数式 1).

$$\frac{TSのデータ量[MB]}{上限スループット[MB/s]}+\frac{ISのI/O数[個]}{上限IOPS[iops]}$$

数式 1：固定上限 I/O 性能による I/O コスト式

2.1 節で動作の説明に用いた図 1 の問合せと，図 2 の実行プランを用いて，2 表結合の処理時間見積りを説明する．CPU コストは従来型データベースエンジンと同様の方式となるため，主に I/O コストについてのみ説明する．

HJ プラン (図 2 の(a)) では Build 側処理を行った後に Probe 側処理を行うため，Build 側処理のコストと Probe 側処理のコストの和が HJ プランのコストと考えることができる．Build 側処理および Probe 側処理のコストは，それぞれの CPU コストと I/O コストの最大値をとる．表へのアクセスは TS か IS のいづれかで I/O コストが小さいほうを採用する．

NLJ プランでは，外表側の処理と内表側の処理を同時に実行する．このため，外表側処理と内表側処理をまとめて CPU コストと I/O コストを計算し，その最大値が NLJ プランのコストとなる．NLJ プラン 1 (図 2

の(b)) の場合は外表側と内表側ともに IS のため, 全てランダムアクセスとなり，その I/O コストは総 I/O 数を上限 IOPS で割った値となる. 一方, NLJ プラン 2 (図 2 の(c)) では外表側の TS の I/O コストと内表側の IS の I/O コストの和が NLJ プランの I/O コストとなる.

NLJ プラン 2 では，TS と IS を同時に行う．本 I/O コストの見積り方式では，TS と IS が共に上限 I/O 性能で処理されると近似したコストとなる．実際は，上限スループットと上限 IOPS を維持しながら処理を進められない可能性があり，この点が誤差要因になり得る.

2.3. 混在アクセス時の I/O 性能を考慮した OoODE コスト見積り方式

前述のように，連続アクセスとランダムアクセスが混在するような混在アクセス時の I/O コストに誤差が生じる可能性がある．つまり,混在アクセス時の I/O 性能をより正確に反映することで，従来方式のコスト見積り方式に比べ実際の処理時間との誤差が小さくなることが期待される．そこで，混在アクセス時の I/O 性能を考慮した OoODE のコスト見積り方式を考える.

混在アクセスでは，図 3 に示すように，連続アクセスとランダムアクセスの双方ともスループットと IOPS の両性能値に寄与する. あるスループットで行われている連続アクセスは，それに応じた IOPS で処理している．一方，ある IOPS で行われているランダムアクセスは，それに応じたスループットで処理している. 数式 1 で示した I/O コスト式では, TS はスループットに対する寄与のみ，IS は IOPS に対する寄与のみを考えており誤差を生じることになる.

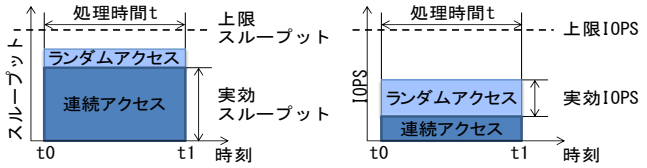

図 3：混在アクセス時のスループットと IOPS

ここで,連続アクセスにおける実際の処理速度である実効スループットと，ランダムアクセスにおける実際の処理速度である実効 IOPS を考える．なお，実効スループットと実効 IOPS をまとめて実効 I/O 性能と呼ぶ．その定義から，実効スループットで連続アクセスのデータ量を割ることにより求める処理時間と，ランダムアクセスの I/O 数を実効 IOPS で割ることにより求める処理時間は一致し，実際の処理時間になると考えられる.

この際に問題となるのは，実効スループットと実効 IOPS の決め方である．混在アクセスでは，連続アクセスとランダムアクセスが時間方向に均一に分布していることを仮定すると，連続アクセスの量とランダムアクセスの量の割合により，実効スループットと実効 IOPS が決まると考えられる．そこで，I/O のみ行うテストプログラムで連続アクセスの量とランダムアクセスの量の割合を変化させた測定を行い，測定結果から定まる実効スループットと実効 IOPS の関係をモデル化した式 (混在 I/O 性能モデル式) を用いて処理時間を見積る．具体的には，混在 I/O 性能モデル式が実効 IOPS を引数として実効スループットを返す関数 f により表現される場合，数式 2 の連立方程式を解き，そこで求まった実効スループットもしくは実効 IOPS により処理時間を見積る．この見積り処理時間が I/O コストとなる.

$$\text{実効スループット} = f(\text{実効}IOPS)$$

$$\frac{TS\text{のデータ量}[MB]}{\text{実効スループット}[MB/s]} = \frac{IS\text{の}I/O\text{数}[\text{個}]}{\text{実効}IOPS[iops]}$$

数式 2：実効 I/O 性能を決める連立方程式

数式 2 を用いて実効 I/O 性能で I/O コストを求める方式を実効 I/O 性能方式と呼ぶ．これに対し，2.2 節で説明したように上限スループットと上限 IOPS の上限 I/O 性能で I/O コストを求める方式を上限 I/O 性能方式と呼ぶ.

3. I/O コスト見積り方式の評価

本章では，上述した I/O コスト見積り方式の評価を行う．初めに，2.1 節で述べた選択率による I/O コスト見積りに対する影響を図 4 の問合せを用いて確認する．次に，図 1 の問合せを用いて，混在アクセスにおけるコスト見積り式を検証する．図 1 の問合せから生成される，図 2 の HJ プランおよび NLJ プラン 1 は混在アクセスが発生しないため，NLJ プラン 2 のコストのみを扱う．また，I/O コスト見積り方式の違いを評価するために，I/O ネックとなる環境において NLJ プラン 2 の実際の処理時間を測定し，上限 I/O 性能方式の I/O コストおよび実効 I/O 性能方式の I/O コストとの比較を行う.

```
SELECT
      count(l_shipdate) l_year,
      sum( case
            when n2.n_name = 'CHINA'
            then l_extendedprice
            else 0
      end) /
            sum(l_extendedprice * (1 - l_discount))
```

```
mkt_share,
      n2.n_name
 FROM
      part,          customer,
      supplier,      lineitem,
      orders,        nation n1,          この条件で
      nation n2,     region              絞込み率を調整
 WHERE
      p_partkey = l_partkey
      and s_suppkey = l_suppkey
      and l_orderkey = o_orderkey
      and o_custkey = c_custkey
      and c_nationkey = n1.n_nationkey
      and n1.n_regionkey = r_regionkey
      and r_name = 'AMERICA'
      and s_nationkey = n2.n_nationkey
      and o_orderdate between DATE '1995-01-01' and
DATE '1996-12-31'
      and p_type = 'ECONOMY ANODIZED STEEL'
      and p_size = 3
      and p_brand = 'Brand#24'
      and p_partkey < 100000
 GROUP BY    n2.n_name
 ORDER BY    n2.n_name;
```

図 4：問合せ(低選択率での性能確認)

3.1. データベースと評価環境

データベースのスキーマは TPC-H で定義されているものを用いる．図 1 の問合せで使用するデータは Scale Factor(SF)が 1,000 のもので，lineitem レコードが 60 億件，orders レコードが 15 億件，合計約 1TB のデータである．図 4 の問合せで使用するデータは SF が 100,000 のものでそのデータサイズは約 100TB である．OoODE の測定には，内閣府最先端研究開発支援プログラムの成果を基にした商用アウトオブオーダ型データベースエンジン Hitachi Advanced Data Binder[8]を用いた．

評価環境を表 1 に示す．I/O コストを確認するため，I/O ネックとなる環境にて測定を行った．

表 1：測定環境の構成

サーバ	日立製ブレードサーバ BS2000 (128 論理コア，1TB メモリ)
ストレージ	日立製ストレージ装置 AMS2500 (16 筐体，15krpm HDD 合計 2,048 台)

3.2. 実効 I/O 性能モデル式

評価環境での実効 I/O 性能モデル式を求めるために，I/O のみ行うテストプログラム (TP) を用いて混在アクセス時の I/O 性能を測定した．orders 領域へのランダムアクセスと lineitem 領域への連続アクセスの割合を変化させ，lineitem 領域の実効スループットと orders 領域の実効 IOPS を採取した．

TP 測定における実効スループットと実効 IOPS の関係を上限性能比で示した(図 5)．このため，本測定環境における実効性能モデル曲線を図 5 の実線とした．

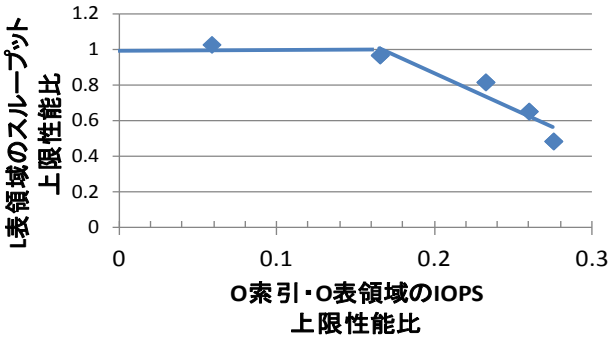

図 5：TP 測定結果と実効 I/O 性能モデル曲線

3.3. 選択率による I/O コスト見積りへの影響に関する考察

図 4 の問合せをベースに，part 表の選択率を 5E-11～5E-6 に変化させた時の測定結果を，5E-11 の処理時間を 1 として相対値でプロットした(図 6)．本評価においては，図 2 の NLJ プラン 1 のように，すべてのテーブルアクセスをインデックスアクセスとなっている．結果を見ると，1.0E-7 以下の絞込み率の領域で I/O コスト見積り値との差が広がっているが，この領域では処理時間がおよそ 1 秒以下となっている．

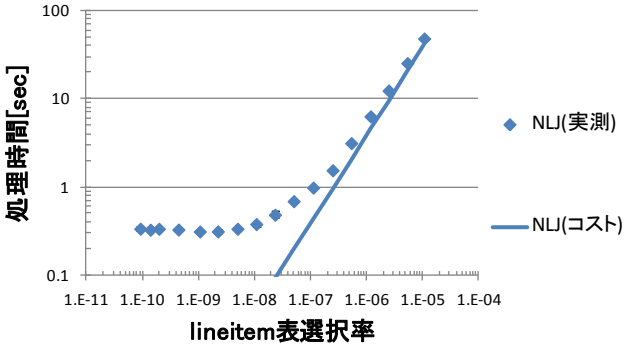

図 6：実測処理時間

ハードウェア構成を変化させることを考えたとき，一番影響が大きいと考えられるのは HDD 性能が低下する場合である．この場合，IOPS 性能のシステム上限に達する I/O 多重度に必要なレコード件数はあまり変

化しないと考えられ，HDD 性能の変化に応じ処理時間が延びると考えられる．ただ，このときは CPU 処理の余裕が大きくなり元来有する処理の並列性をより引き出せる状況にあり，HDD 性能の反比例を超えて悪化することはないと考えられる．従って，選択率による I/O コストに対する影響は，処理時間が数秒より大きければこの誤差は十分小さいと考えられ，大規模データにおいて NLJ と HJ の選択に利用しても実用上許容されると考える．

3.4. 混在アクセスにおける I/O コストと実測時間の比較

評価環境における実際の処理時間と，上限 I/O 性能方式による I/O コストおよび実効 I/O 性能方式による I/O コストを比較する．図 1 に示した問合せをベースに，l_quantity の条件を変化させて lineitem 表の選択率を 0.0E-0～5.3E-3 に変化させた問合せ 8 個を用いる．l_quantity 以外の条件が存在するため，選択率の最大値が 5.3E-3 となっている．

図 7 に実測した処理時間と I/O コストをプロットしたグラフを示す．lineitem テーブルの選択率が 0E-0 の時の実測処理時間を 1 とした時の処理時間を示した．

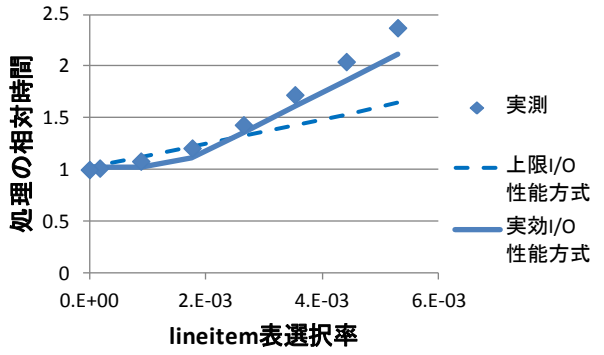

図 7：実測と I/O コストの比較

上限 I/O 性能方式の I/O コストは，2E-3 より小さい選択率では実測との誤差が 4%以下であった．しかし，2E-3 より大きい選択率では，ランダムアクセスの量が増加するに従い，実測と I/O コストの差が拡大していき，最大 31%まで誤差が拡大した．一方，実効 I/O 性能方式の I/O コストは，0E+0 以上 6E-3 以下の選択率の領域で，実測との誤差が 11%以下に抑えられている．

本評価により，実効 I/O 性能方式により，実測との誤差が小さくできていることが確認できた．

4. おわりに

本論文では，著者らが研究開発を進めているアウトオブオーダ型データベースエンジンにおける 2 表結合問合せの処理時間見積りの課題と，その解決方式を報告した．連続アクセスやランダムアクセスそれぞれの実際の I/O 性能である実効 I/O 性能により I/O コストを算出する実効 I/O 性能方式を検討した．TPC-H ベンチマークのデータを用いた評価では，商用 OoODE を用いた実測結果と I/O コストを比較することにより，提案方式が有効であることを確認した．

謝　辞

本研究は，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」の助成により行われた．

文　献

[1] IDC Japan，国内外付型ディスクストレージ市場の実績と予測を発表，http://www.idcjapan.co.jp/Press/Current/20130617Apr.html, 2013

[2] 喜連川優，合田和生,アウトオブオーダ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, 2009.

[3] 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.

[4] 早水悠登，合田和生，喜連川優，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，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.

[5] 清水晃，徳田晴介，田中美智子，茂木和彦，合田和生，喜連川優，アウトオブオーダ型データベースエンジン OoODE におけるタスク管理機構の一実装方式の評価，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.

[6] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. 1979. Access path selection in a relational database management system. In Proceedings of the 1979 ACM SIGMOD international conference on Management of data (SIGMOD '79). ACM, New York, NY, USA, 23-34.

[7] Transaction Processing Performance Council, TPC BENCHMARK™ H, Standard Specification, http://www.tpc.org/tpch/spec/tpch2.16.0.pdf, 2013.

[8] 日立，東京大学との超高速データベースエンジンの共同研究開発成果を製品化，http://www.hitachi.co.jp/New/cnews/month/2012/05/0528.html, 2012.