DBSJ Journal, Vol.8, No.1
June 2009

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

Vision and Preliminary Experiments for Out-of-Order Database Engine (OoODE)

喜連川 優♥　　合田 和生♥

Masaru KITSUREGAWA　Kazuo GODA

本論文では，アウトオブオーダ型データベースエンジン（OoODE）と称する新しい実行原理に基づく高性能データベースエンジンの構成法について論じる．当該データベースエンジンは，従前のインオーダ型の実行ではなく，問合せ処理をアンフォールドすることにより多数のプロセッサコアを活用し，また，複数の非同期入出力を同時に発行することにより多数のディスクドライブを活用し，これにより性能向上を目指すものである．本論文ではアウトオブオーダ型データベースエンジンの開発初期段階におけるプロトタイプ実装を示すとともに，実験によりその有効性を検証する．

This paper presents our study on basic design of *Out-of-Order Database Engine (OoODE)*, a high-performance database engine based on novel execution mechanism. OoODE processes a given query while dynamically unfolding the query processing and issuing a number of asynchronous inputs/outputs. These execution properties are helpful for exploiting many processor cores and many disk spindles efficiently. Considerable performance improvement for data analysis queries is expected in comparison with the conventional in-order execution engine. The paper presents our preliminary implementation of the engine and explores its potential benefits through the experiments.

1. はじめに

近年，データベースシステムの研究の方向性としては，XML，ストリーム，リネージ並びにプライバシなど，応用からの研究が多く，所謂コアなエンジン部分に関する研究は少なくなっている．サービスから技術を引き出すという観点では，この傾向は極めて健全であると言えるものの，一方で，ハードウェアの変革も大きく，データベースエンジンの再考に値する段階に入ったとも考えられる．ディスクアレイ技術は広く浸透し，最近ではストレージ仮想化が広く実用化されつつある[4]．インターネットサービスプロバイダのニーズに支えられながら生まれたシンプロビジョニング[1]は，物理領域を事前には割り当てず，実行時に割り当てる方式であり，即ち，仮想化技術の一種であるが，これにより， ISPの負担を大幅に軽減するに至った．更には，管理コストの低減を目的として，大規模なコンソリデーションが進みつつあり，1000ドライブを超えるディスクアレイも珍しくない時代となってきた[11]．即ち，1000ドライブの帯域活用を想定してデータベースエンジンを再考する時期が来たと言えよう．同時に，プロセッサにもマルチコアなどの大きな変化が見られることは周知の通りである．既に16コア／チップの動作は見られ，100コア／チップの時代もそう遠くはないだろう[2]．莫大なストレージとプロセッサコアを有効活用するデータベースエンジンを考えることは，必須のものと考えられる．

本論文では，このような背景のもと，**アウトオブオーダ型データベースエンジン**，即ち OoODE（Out-of-Order Database Engine）と称する新しい実行原理に基づく高性能データベースエンジンの構成法について論じる．第2章では，アウトオブオーダ型データベースエンジンの仕組みを述べ，第3章では，それが有効な適用領域を議論する．第4章では，著者らが開発を行っている初期段階のプロトタイプ実装とそれによる評価実験を示して，その有効性を議論し，第5章においてさらに本格的な実装に向けた技術的課題を議論する．最後に第6章で，本論文をまとめる．

なお，データベースエンジンの構成法に関する最近の研究としては，A. Ailamaki らが QPipe[6]を提案している．これは複数の問合せが実行される環境において，各々の問合せ処理を分割し，これらを並列パイプライン実行することにより，問合せ間でデータと演算の共有を図るものである．対して，本論文で提案する OoODE は，単一の問合せに対して，問合せの持つ内部並列度を活用することにより大幅な性能向上を目指すものである．また，K. Ross らは Cray MTA-2 を対象に，データベースエンジンを改良し，特定のデータベース演算を並列化する手法[3]を提案している．プロセッサレベルのハードウェアマルチスレッド化に焦点が当てられており，入出力は旧来の形態を前提としており，得られる性能向上は限定的なものに留まっている．著者らの知る限りにおいて，類似の提案はなされていない．

2. アウトオブオーダ型データベースエンジン

従来型のデータベースエンジンは，基本的にインオーダ型の手続き実行に基づいている．即ち，レコードのフェッチから当該レコードの処理に到る一連の動作は，通常は事前にプログラムされた決定的な順序に基づくものとなる．これに対し，ここではアウトオブオーダ型の動作が可能なデータベースエンジンを考えたい．

簡単のため，2つのリレーションの結合演算を考えることとする．この場合，多くはネステッドループ結合かハッシュ結合が利用されるが，ここではネステッドループ結合を考えよう．これは，片方のリレーション（外表と称される）から1タプルを取り出し，これに基づき，もう片方のリレーション（内表と称される）から結合条件に合致するタプルを探し出すことを繰り返すものである．この際，一般には，外表としてより小さいリレーションを選択することにより効率化を図り，また，内表には結合属性に対してインデクスを作成しておく．即ち，ネステッドループ結合の基本動作は，まず外表から1タプルをフェッチし，結合条件に従って内表のインデクスの検索を行い，更に条件に合致するタプルが見つかる場合には，内表からタプルをフェッチし，この一連の動作が終了した後に，次に，外表から次の1タプルをフェッチし，以降，逐次的に同様の手続きを行うものとなる．Ingres[10]

♥ 正会員　東京大学 生産技術研究所
{kitsure, kgoda}@tkl.iis.u-tokyo.ac.jp

以来，多くのDBMSがまさしくこの実装方式を利用している．これは，主記憶が高価であり，またプロセッサ性能が低い時代においては，最も妥当な選択肢であった．しかしながら，現状のサーバの主記憶は著しく大容量化し，同時にプロセッサは著しく高速化している一方，ディスクドライブのランダムアクセス性能[1]は殆ど向上しておらず，両者の性能は大きく乖離している[7,13]．新たなハードウェア特性のバランスに基づき，データベースエンジンを再考する段階に入ったと言えよう．

ここで，ネステッドループ結合の動作においては，例えば，外表のそれぞれのタプルから駆動される一連の処理は相互に独立であることに着目したい．外表の第1タプルのフェッチから駆動される一連の内表へのアクセスと，第2タプルのフェッチから駆動される一連の内表へのアクセスとは，どちらが先行して処理されたとしても，最終的な問合せ結果に影響を及ぼさない．即ち，ネステッドループ結合におけるループはアンフォールドし，例えば，外表のそれぞれのタプルから駆動される一連の処理に対して，別々のスレッドを割り当てて実行することが考えられる．アンフォールドは，必ずしも外表のレベルに限定されたものではなく，例えば，3段以上の結合演算であれば，外表，第一段階の内表のそれぞれレベルでアンフォールドを行い，実行時にスレッドを割り当てることも可能であろう[2]．

ネステッドループ結合におけるアンフォールド自体は容易に考案されるものである．過去にも外表をいくつかのサブリレーションに分割し，それぞれにプロセッサを割り当てるという方式が考えられたが，これは単に分割してインオーダ処理を行うものにすぎない．

さて，実行時のアンフォールドによっては，多数の非同期的な入出力がストレージ装置に発行されることとなるが，この際に発行可能な入出力数は，現実的には，例えば，ディスクドライブやディスクアレイコントローラのタグ長，ホストバスアダプタの最大キュー長，および，カーネルの入出力処理モジュールなどのシステムの多様な有限資源によって制約を受ける．ストレージ装置は近年莫大な数のドライブを搭載するに至り，従来に対してより多くの入出力の同時処理が可能となってきており，また，ディスクドライブ内，ディスクアレイコントローラ内さらにはOS内の高度なスケジューリング機構により，論理的な入出力発行順序とは異なる順序で入出力処理がなされるのが通例である．即ち，ストレージ装置からの入出力完了は非順序的になされ，ここで提案するアウトオブオーダ型のデータベースエンジンでは，入出力完了によって手続きが駆動されるべく制御がなされることとなる．従来のデータベースエンジンでは，インオーダ型の動作によって極めて少量の入出力のみが発行されていたのに対し，アウトオブオーダ型のエンジンは，実行論理が許す限り，大量の入出力を発行することを可能とし，これにより，極めて高速化の進んでいるプロセッサ資源の活用と，膨大な数のディスクドライブの並列アクセスによって，効果的な性能バランスを狙うものである．当然のことながら，スタンディング入出力数は1000を超えることも予想され，カーネル内のメモリスペースを従来に比して大幅に使用することとなるなど，従来型のエンジンとは動作方式が大きく異なることとなる．1000以上の非同期入出力を発行すること自体，殆ど試みられてこなかったことが実情であり，データベースエンジンだけではなく，OSを含めたシステムの設計，構築が必要となる．

[1] ネステッドループ結合における内表のタプルアクセスは多くの場合，ランダムアクセスとなる．

[2] アンフォールドの粒度設定には多くのバリエーションが考えられる．例えば，二次記憶上のデータ構造によっては，インデクス検索と，それに基づくリレーションからのタプルフェッチのそれぞれのレベルでアンフォールドを行うことが可能である．また，インデクスツリー探索においてもポインタトラバースの機会の度に，アンフォールドを行うことが可能である場合もある．

3. アウトオブオーダ型データベースエンジンの適用領域

ここまでアウトオブオーダ化により高速化を目指す新しいデータベースエンジンの構成について述べた．本章では，その適用範囲について考えたい．

データベースにおける問合せ処理方式の歴史を振り返ると，ネステッドループ結合が当初用いられてきたが，大規模リレーションに対しては極めて非効率的であることが明らかとなり，80 年代後半よりハッシュ結合[5,8]が用いられるようになってきた．これは，ハッシュ結合はとりわけ選択率がある程度大きな場合においては，圧倒的に高速であるためである．一方，ハッシュ結合は，多くの場合，プローブ側のリレーションのスキャンが支配的となり，巨大なリレーションに対しては必ずしも有効でなくなる可能性がある．即ち，リレーション規模がテラバイトからペタバイトへと超巨大化する中で，単純なスキャンは大きなコストとなることは明らかであり，人間が処理可能なデータ長は絶対的な大きさで制約されるであろうから，選択率は相対的に縮小する方向になると予想される．即ち，従来のオプティマイザは選択率がある一定以上の場合はハッシュ結合を，それ以下の場合にはネステッドループ結合を用いていたが，今後，アウトオブオーダ実行による高速化により後者の重要性が増すものと考えられる．

アプリケーションとしては，大規模トレーサビリティセンタなどが考えられる．例えば，ある工場のあるロットのある日時の製品の具合が悪いと判明した場合には，当該製品を構成部品とする最終製品に対してどのような影響を与えているかをチェックする必要があり，これには，多段の結合演算を要するが，選択率は極めて小さなものとなることが予見される．

4. アウトオブオーダ型データベースエンジンのプロトタイプ実装と評価実験

先述の通り，アウトオブオーダ実行を実現することにより，データベース問合せ処理の多大な性能向上が期待される．このような性能向上効果を検証するために，著者らは，アウトオブオーダ型データベースエンジンのプロトタイプの構築を進めている．本章では構築中のプロトタイプを用いた性能評価実験を示し，提案エンジンの有効性を議論する．

4.1 アウトオブオーダ型データベースエンジンのプロトタイプ実装

著者らは MySQL InnoDB ストレージエンジン[9]のデー

タフォーマットを直接マウントすることのできるアウトオブオーダ型データベースエンジンの構築を行っている．現状のプロトタイプでは，インデクススキャンやネステッドループ結合などの基本的な演算処理のアウトオブオーダ実行が可能である．

従来型のデータベースエンジンにおいては，インデクススキャンを行う場合，インデクス検索，さらに対応するリレーションからのレコードフェッチが逐次的に繰り返される．また，ネステッドループ結合の問合せ処理を行う場合には，同様に決定的な処理順序に従い，外表のレコードフェッチ，対応する内表のインデクス検索，更に対応する内表のレコードフェッチが繰り返される．即ち，入出力命令が次の動作をブロックするため，同時に極めて少数の入出力が発行されるにすぎない．これに対して，アウトオブオーダ型のデータベースエンジンにおける場合，インデクススキャンについてはインデクス検索とリレーションのレコードフェッチが重層的に行われ，また，ネステッドループ結合の処理では，イベント駆動型の動作モデルに従い，外表のレコードフェッチ，内表のインデクス検索，内表のレコードフェッチが重層的に行われ，よってその処理順序が実行時に決定される．この際，問合せ処理は非同期入出力を用い複数の入出力を同時に発行することがなされるため，多大な性能向上が期待される．

なお，現状の実装プロトタイプでは問合せ処理のアンフォールドを必ずしもレコードレベルに限定せず，たとえば，データベースのインデクスツリー探索においても垂直ポインタのトラバースの単位でアンフォールドを行うこととしている．たとえば，インデクスツリーのルートページ P0 から 3 つの子ページ P1，P2 および P3 にページポインタが張られているとする．インオーダ型処理系でインデクススキャンをする場合，一般には P0 のフェッチ，P0 の解釈，P1 のフェッチ，P1 の解釈，P2 のフェッチ，P2 の解釈などと逐次的に処理がなされる．これに対して，プロトタイプにおいては，P0 の解釈ののち，P1，P2 および P3 に対して非同期的なフェッチ要求を行い，データがフェッチされたものから解釈を行う．より細粒度でのアンフォールドを行うことにより問合せの有する潜在的な並列度を導き出すことが期待される[3].

4.2 アウトオブオーダ型データベースエンジンの評価実験

前節で述べたデータベースエンジンのプロトタイプ用いて，ネステッドループ結合におけるアウトオブオーダ型実行の有効性を検証した．

実験に際し，データベースサーバとディスクアレイから構成される実験システムを構築した．データベースサーバは 8 個の 4 コア Xeon プロセッサ，64GB の主記憶を有し，OS として EM64T モードの RedHat Enterprise Linux 5 が動作する．一方，ディスクアレイは SATA 規格のディスクドライブを搭載可能であり，データベースサーバと合計 8 本の 4Gbps ファイバチャネル接続を有する．

実験においては，8 台のディスクドライブを用いてパリティなしストライピング(RAID-0)編成のボリュームを構成し，この際のセグメントサイズを 128KB とした．当該ボリュームとサーバ間の接続帯域としては 4Gbps のファイバチャネル 1 本を用意した．ボリュームにはファイルシステムを構築せず，所謂ローデバイスとして扱い，16KB のページ長を以て 128GB の InnoDB 表空間を構築した．また，表空間には 1.6GB のデータベースバッファを主記憶に割り当てた．

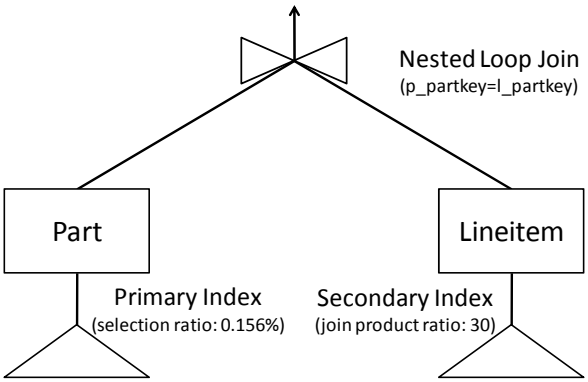


図 1 問合せ実行プランの例
Fig. 1 An example of query execution plan

データベースエンジンの性能測定においては，TPC-H ベンチマーク[12]を用いた．即ち，スケールファクタを 32 とした初期データセットを生成し，これを表空間に読込ませ，その後，TPC-H 規定の問合せ 14 及び 17 を参考に作成した図 1 に示す問合せの処理を実行し，その性能を計測した．即ち，Part 表を外表とし，Lineitem を内表とするネステッドループ結合であり，この際，Part 表の選択率を 0.156%とし，Part 表から Lineitem 表に対する結合比率を 30 とした．

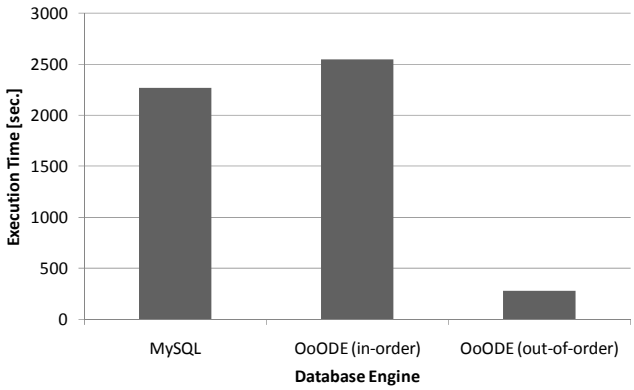
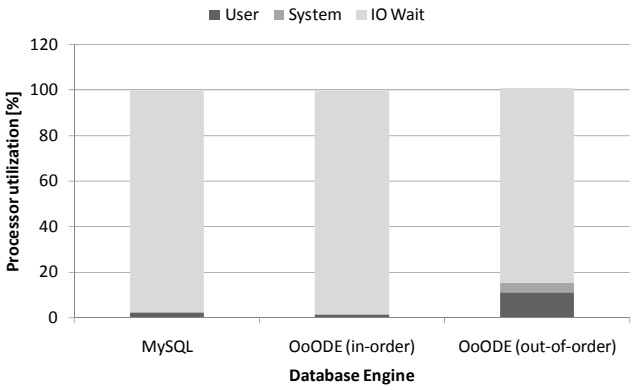


図 2 問合せ実行時間の比較
Fig. 2 Comparison of query execution time

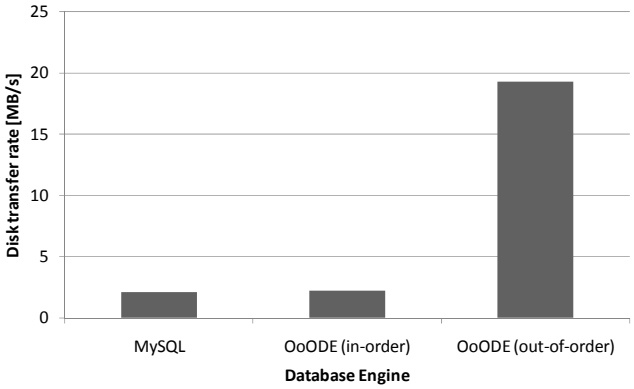
図 2 にプロトタイプで当該問合せ処理を行った際の実行時間を示す．この際，インオーダ実行とアウトオブオーダ実行とを比較する他，参考のために MySQL データベースエンジンでインオーダ実行を行った場合を示す．当然のことながら，MySQL データベースエンジンはインオーダ実行を行うものであることから，MySQL データベースエンジンにおける問合せ実行と，プロトタイプにおけるインオーダ実行での実行時間には大きな差は見られない．一方，アウトオブオーダ実行に関しては，大幅な実行時間の改善が見られ，概ねイ

[3] 本論文はアウトオブオーダ型データベースエンジンの基本的な構成法について焦点を絞っており，プロトタイプ実装におけるアンフォールド手法の比較については議論を別稿に譲りたい．

ンオーダ実行と比較して約 8 倍の性能向上が見られた.



(a) Processor utilization
(a) プロセッサ利用率



(b) Disk transfer rate
(b) ディスク転送レート

図 3 プロセッサ利用率ならびにディスク転送レートの比較
Fig. 3 Comparison of processor utilization and disk transfer rate

図 3 には, 上記の実験時におけるプロトタイプにおけるサーバのプロセッサ利用率ならびにディスク転送レートを示す. 今回の実験では, 1 プロセッサコアのみを用いることとしたため, 例えば図中の 100%は 1 コアが完全にビジーであることを意味する. インオーダ実行では, 処理動作が同期入出力によってブロックされているため, プロセッサ資源の内, 実質的に利用されているものは 3%に満たず, 残りは入出力待ちに利用されている. これに対して, アウトオブオーダ実行では, 非同期入出力を用いることにより同時に多数の入出力を発行することが可能となることから, 約 20%の資源が実質的なプロセッサ処理に有効利用されるようになった. また, ディスク転送レートに関しても同様に, インオーダ処理からアウトオブオーダ処理で大幅なスループットの向上が確認された. 即ち, アウトオブオーダ実行により, プロセッサ資源とディスクドライブ資源が有効に活用され, 問合せ処理の性能が向上することが確認された.

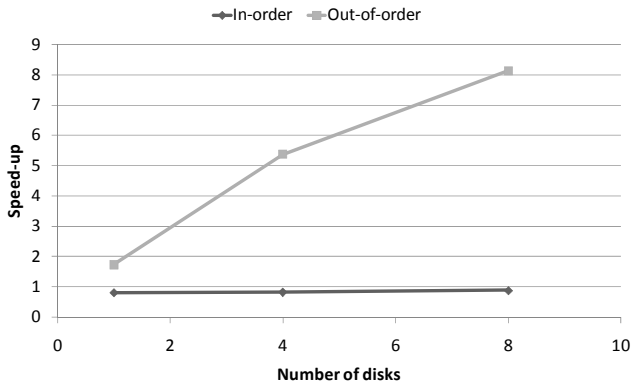


図 4 ディスクドライブ数に対する
問合せ処理のスピードアップの変化
Fig. 4 Speed-up of query processing along the number of disk drives

次に, 構築したプロトタイプに関して, ボリュームを構成するディスクドライブの数を 1 から 8 まで段階的に変化させ, 性能向上の変化を確認した. 図 4 に, MySQL を比較対象とした場合のスピードアップ曲線を示す. インオーダ実行に関しては, 同時に極めて少数の入出力が発行されるために, ディスクドライブ数の増加に対して性能向上が殆ど観測されない. これに対して, アウトオブオーダ実行に関しては, 同時に複数の入出力を発行することが可能となるため, ディスクドライブ数の増加に伴い性能向上が見られ, 8 台のドライブを用いて概ね 8 倍の性能向上が確認された. ただ, このスピードアップ値については, 前述の通り, システム中の様々な有限資源による制約により, 十分に入出力が発行されていない可能性があり, なおも改善が期待されると考えている. 今後の課題として, 更なる性能向上に向けた改善を検討したい.

これらの実験結果から, アウトオブオーダ型のデータベースエンジンは, 特に多くのコアを活用するデータベースサーバ, ならびに多くのドライブを活用するディスクアレイの資源を有効活用することにより, 従来型のデータベースエンジンと比較して高い性能上の優位性を持つ可能性が示された. また同時に, 更なる性能向上に向けた改善の重要性も明らかになったと言えよう.

5. 本格的な実装に向けて

本論文では, ここまでアウトオブオーダ型データベースエンジンの基本アイデアと構成法, ならびに初期実験を議論してきた. 現時点で著者らが構築を進めているプロトタイプについては, アウトオブオーダ型処理が基本的なデータベース演算に限定されているなど, さらに実装を進展させる必要がある. 本章では, 本格的な実装に向けた技術的課題を議論したい.

適用演算の拡張

現状の実装プロトタイプにおいては, アウトオブオーダ型処理がインデクススキャンやネステッドループ結合などの基本的なデータベース演算に限定されており, 著者らは適用演算を拡張するべく検討を進めている.

第一には, 上記のほか, 関係データベースで利用される演算のうち, アウトオブオーダ化によって大幅な高速化が期待

される演算は他にもある．たとえば，近年はアプリケーション開発フレームワークが充実し，アプリケーション開発者はフレームワークを通じてデータベース管理システムにアクセスを行う機会が増えており，一般に記述される SQL はビジネスロジックをそのまま転写したものとなってより複雑化している場合がある．すなわち，サブ問合せが頻繁に利用される傾向があるが，そのうち，相関条件を有するサブ問合せについては本質的にアウトオブオーダ化が可能であろう．

第二に，関係データベース演算で利用される演算のうち，ソートやアグリゲーションについては最終的には逐次処理を要するため，完全なアウトオブオーダ化は望めないものの，部分的なアウトオブオーダ化が可能である場合も少なくない．たとえば，ネステッドループ結合を行い，その結果に基づきアグリゲーションとソートを行う問合せは情報解析システムでは頻繁にみられるが，この場合，ネステッドループ結合に引き続きアグリゲーションの一部をそれぞれアウトオブオーダ化しパイプライン実行するなど，実装上の工夫による利得は小さくないだろう．

実行時資源調整の実現

従来的なインオーダ型の処理系では，一般に中間状態を最小化するべく，事前に決定された順序によって逐次的な処理がなされる．これに対して，アウトオブオーダ型の処理系は非同期的な処理を基本とし，よってその振る舞いは事前決定することができず，また，実行多重度は一般に 100 以上となることが想定される．処理系には高度な実行時資源調整機能が求められることとなる．

現状のプロトタイプにおいてはより高位のタスクに対してより高い優先度を与えるスケジューリングを行う比較的単純な実行時資源調整を行っているに過ぎない．8 台のディスクドライブに対して高々8 倍の性能向上が得られているに過ぎず、さらに資源を効率的に利用できる可能性があり、本格的な実装に向けて，より高度な資源調整機能を考案する必要があるだろう．

更新処理のサポート

現状のプロトタイプ実装では，データ解析専用エンジンとし，更新処理をサポートしていないが，バッチ処理など，アウトオブオーダ型実行によってご利益のあるアプリケーションも少なくない．更新処理については，理論・実装双方について検討を深める必要があるだろう．

6. おわりに

本論文では，プロセッサの構成が多数コアアーキテクチャへと大きく変遷するとともに，巨大ストレージシステムの登場により多数のスピンドルが単一筐体内で駆動されるようになりつつあることを鑑み，新しい実行原理に基づく高性能データベースエンジンの構成法について論じた．即ち，従来型のインオーダ型の実行ではなく，問合せ処理をアンフォールドすることにより多数のプロセッサコアを活用し，また，複数の非同期入出力を同時に発行することにより多数のディスクドライブを活用するアウトオブオーダ実行の有効性を議論した．

また，著者らが構築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のプロトタイプの初期実装を示した．ネステッドループ結合を用いた問合せ処理に関する実験においては，プロセッサ，ストレージ資源ともに全く同一の環境において，インオーダからアウトオブオーダに動作を変更することにより，性能を 800%に向上可能であることが示された．現時点のシステムは，未だ，種々のチューニング前の極めて荒削りな段階であり，当該性能よりもはるかに高い性能を引き出すことが可能であるものと考えている．今回は 1 コアのみを利用したが多コア化をはじめ，資源管理方式などを順次報告してゆきたい．

[謝辞]

本研究の一部は，文部科学省次世代 IT 基盤構築のための研究開発「非順序型実行原理に基づく超高性能データベースエンジンの開発」の助成により行われた．プログラムオフィサーである国立情報学研究所坂内所長をはじめ研究開発プロジェクト会議の先生方より有益なコメントを頂戴しており，深く感謝を申し上げる．また，協力企業である株式会社日立製作所に感謝する次第である．

[文献]

[1] 3PAR Inc.: “3PAR Thin Provisioning, Eliminating Allocated but Unused Storage and Accelerating ROI”, White Paper (2008).
[2] Shekhar Borkar: “Thousand Core Chips - A Technology Perspective”, In Proceedings of Annual ACM/IEEE Design Automation Conference, pp.746-749 (2007).
[3] F. Bunn and R. Peglar: “Storage Virtualization I. What, Why, Where and How?”, SNIA Education (2004).
[4] J. Cieslewicz, J. Berry, B. Hendrickson and K. A. Ross: “Realizing Parallelism in Database Operations: Insights from a Massively Multithreaded Architecture”, Proceedings of the 2006 Workshop on Data Management on New Hardware (2006).
[5] David J. DeWitt, Robert H. Gerber, Goetz Graefe, Michael L. Heytens, Krishna B. Kumar, and M. Muralikrishna: “GAMMA - A High Performance Dataflow Database Machine”, In Proceedings of International Conference on Very Large Data Base, pp.228-237 (1986).
[6] S. Harizopoulos, V. Shkapenyuk and A. Ailamaki: “QPipe: A Simultaneously Pipelined Relational Query Engine”, Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp.383-394 (2005).
[7] Masaru Kitsuregawa, Kazuo Goda, and Takashi Hoshino: “Storage Fusion”, In Proceedings of 2nd International Conference on Ubiquitous Information Management and Communication, pp.287-294 (2008).
[8] Masaru Kitsuregawa, Hidehiko Tanaka, and Tohru Moto-Oka: “Application of Hash to Data Base Machine and Its Architecture”, New Generation Computer, Vol.1 No.1 pp.63-74 (1983).
[9] MySQL AB.: “MySQL: The World's Most Popular Open Source Database”, http://www.mysql.com/ (2009).
[10] M. Stonebraker, Eugene Wong, Peter Kreps, and Gerald Held: “The Design and Implementation of INGRES”, ACM Transaction on Database Systems, No.1 Vol.3 pp.189-222 (1976).
[11] N. Takahashi and H. Yoshida: “Hitachi TagmaStore Universal Storage Platform: Virtualization without Limits”, White Paper, Hitachi Ltd. (2004).

[12] Transaction Processing Performance Council: “TPC-H, an ad-doc, decision support benchmark”, http://www.tpc.org/tpch/ (2009).
[13] 喜連川優，合田和生，星野喬，茂木和彦，河村信男，土屋宏嘉，阿部淳，西川記史，大枝高，鈴木芳生，藤原真二，杉江衛，小高俊彦：「ストレージフュージョン：ストレージシステムとデータベース管理システムの融合」 情報処理, Vol.49 No.11 pp.1284-1289 (2008).

喜連川 優 Masaru KITSUREGAWA
1978 東京大学工学部電子工学科卒業．1983 同大大学院工学系研究科情報工学専攻博士課程修了．工学博士．同年同大生産技術研究所講師．現在，同教授．2003 より同所戦略情報融合国際研究センター長．データベース工学，並列処理，Web マイニングに関する研究に従事．現在，本会理事，情報処理学会副会長およびフェロー，電子情報通信学会各フェロー．ACM SIGMOD Japan Chapter Chair，電子情報通信学会データ工学研究専門委員会委員長歴任．VLDB Trustee, IEEE ICDE, PAKDD, WAIM などステアリング委員．

合田 和生 Kazuo GODA
2000 東京大学工学部電気工学科卒業，2005 同大学院情報理工学系研究科電子情報学専攻博士課程単位取得満期退学．同年，博士(情報理工学)．現在，東京大学生産技術研究所特任助教．並列データベースシステム，ストレージシステムの研究に従事．本会，情報処理学会，ACM，IEEE CS，USENIX 会員．