DBSJ Journal, Vol.12, No.1
June 2013

アウトオブオーダ型データベースエンジンOoODEの試作実装と小規模実験環境におけるソフトウェア実行挙動の観測

Prototype Implementation of Out-of-Order Database Engine (OoODE) and Observation of Its Software Execution Behavior in a Small Experimental Environment

合田　和生♡ 豊田 正史♢ 喜連川 優♠

Kazuo GODA　Masashi TOYODA
Masaru KITSUREGAWA

アウトオブオーダ型データベースエンジン **(OoODE)** と称する著者らの提案する高速データベースエンジンは，問合せ処理を動的にアンフォールドし，すなわち，タスク分解して，高多重のタスク並行実行ならびに高多重の非同期入出力発行を行う．マルチコアプロセッサの演算パワーとディスクストレージの入出力帯域を高効率に活用することを可能とすることにより，中程度の選択性を有する解析系の問合せにおいて，従来型の逐次的な実行方式によるエンジンと比べて，問合せ処理の高速化が期待される．本論文では，これまで著者らがオープンソース **DBMS** をベースとして進めてきたアウトオブオーダ型データベースエンジンの試作機の実装を示すとともに，当該データベースエンジンの特徴であるソフトウェア動作挙動の非決定性を検証し，当該データベースエンジンの有効性を明らかにする．

Out-of-Order Database Engine (OoODE), a high-speed database engine we have proposed, has a novel mechanism for unfolding query processing at runtime. This unique property can introduce a new opportunity for the database engine to make much better use of processing capacity of multicore processors and IO bandwidth of disk storage systems. Our observation is that substantial performance boosting could be achieved for analytical queries with moderate selectiveness in comparison with other conventional database engines. This paper presents prototype implementation of OoODE that we have developed on the basis of open-source database software and investigates its nondeterministic behavior, an unique software property of OoODE, to clarify the potential benefits of the new engine.

♡ 正会員　東京大学生産技術研究所
kgoda@tkl.iis.u-tokyo.ac.jp
♢ 正会員　東京大学生産技術研究所
toyoda@tkl.iis.u-tokyo.ac.jp
♠ 正会員　国立情報学研究所，東京大学生産技術研究所
kitsure@tkl.iis.u-tokyo.ac.jp

1. 序論

TPC [18] の性能ランキングリストに掲載されているハードウェア構成をみると，単一のシステムにおいて，64 以上のプロセッサコアと 1,000 以上のディスクドライブが強結合されていることはもはや珍しくなく，当該傾向はハイエンド機種からミッドレンジ機種へと波及している．市場においては，プロセッサコア数の増加は確実に進展しており，また，所謂ビッグデータブームに牽引され，ストレージシステムの大型化・高密度化が著しいことは周知の通りである．莫大なプロセッサ資源とディスクドライブ資源を有効活用することは，巨大データの管理を担うデータベース技術にとって，必須のものと言えよう．

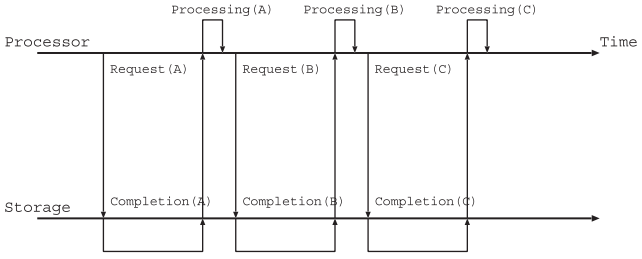
著者らは，アウトオブオーダ型データベースエンジン (OoODE) と称する高速データベースエンジンの開発を進めている [22]．当該データベースエンジンは，問合せ処理を動的にアンフォールドし，すなわち，タスク分解する機能を有しており，高多重のタスク並行実行ならびに高多重の非同期入出力発行を行うことによって問合せを処理を行う．マルチコアプロセッサの演算パワーとディスクストレージの入出力帯域を高効率に活用することが可能とすることにより，中程度の選択性を有する問合せにおいて，従来型の逐次的な実行方式によるエンジンと比べて，問合せ処理の高速化が期待される．本論文では，これまで著者らが進めてきたオープンソース DBMS をベースとするアウトオブオーダ型データベースエンジンの試作機の実装を示すほか，当該データベースエンジンの特徴である動作挙動の非決定性の検証を行い，当該データベースエンジンの有効性を明らかにする．

本論文の構成は以下の通りである．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基本アイデアを紹介し，第 3 章では，著者らが進めている試作機の実装を示す．第 4 章では，当該試作機において TPC-H ベンチマークデータセットを用いて行った動作挙動の検証実験を報告し，第 5 章において本論文を纏める．

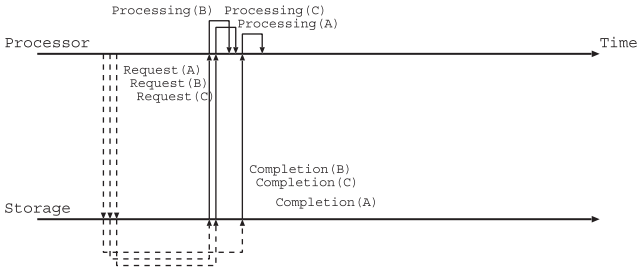
2. アウトオブオーダ型データベースエンジン

多くのデータベースエンジンは，問合せを受け付けると，問合せ実行木を生成し，実行木における演算ノードを逐次的に辿って実行することにより，当該問合せを処理する．演算ノードの実行においては，対象となるデータをデータベースから取得する必要がある場合，データベースが格納されたストレージに入出力命令を発行しその完了を待って演算を実行することを，対象の全てのデータを処理し終えるまで繰り返す．当該挙動を模式的に図 1(a) に示す．入出力の発行と演算の実行は事前にプログラムされた順序に基づき行われ，すなわち，その挙動は決定的であると言え，同じデータベースに対して同じ問合せを処理する場合は，常に同じ順序で入出力と演算が実行され，同じ結果が返される．Ingres [14] 等初期のデータベース実装以来，多くのデータベースエンジンがインオーダ型の実行方式を採用してきている [4,19]．Right Deep ハッシュ結合に見られるパイプライン処理 [12] や，主記憶指向データベースエンジンに見られる SIMD 型演算実行 [2] 等，幾多の工夫が行われてきたが，基本的な実行方式そのものには変化は見られない．本論文では，このようなソフトウェアの実行方式をインオーダ型と称することとする．

関係モデルに基づくデータベースエンジンの実装が試みられ始めた当時，プロセッサ性能と主記憶容量は今日と比べて遙かに限



(a) インオーダ型実行.



(b) アウトオブオーダ型実行.

図 1: データベースエンジンの実行方式の比較.

定的であった．ディスクドライブに代表されるストレージ（二次記憶）の性能も今日と比べて限定的ではあったものの，プロセッサと主記憶が相対的に高価であり，これらを中心としてシステムは構築された．すなわち，演算資源と入出力資源という観点では，圧倒的に演算資源が高価であり，ソフトウェア開発においても，演算にかかるコードパスとフットプリントを節約することが肝要とされ，インオーダ型の実行方式は妥当な選択肢であったと言えよう．しかしながら，今日のハードウェア環境においては，プロセッサは著しく高速化しており，大容量の主記憶が利用可能である．他方で，相対的なストレージの性能は低下しており，ストレージがシステム全体の性能を律速するケースが増え，システム設計の主眼はむしろストレージに注がれるようになってきた [5,6]．新たなハードウェアの特性バランスに基づき，従前とは異なるソフトウェアアーキテクチャを再考する段階に入ったとも言えよう [15,16]．圧縮技法が提案され，市場で注目を集めるようになってきている [1,10,11] が，著者らは，より根源的なアプローチに挑戦することとし，データベースソフトウェアの実行方式そのものを変革し，即ち，ソフトウェアの非同期化によって，相対的に安価になりつつある演算資源を活用し，入出力資源を高効率に利用するアウトオブオーダ型と称するデータベースエンジンの実行方式を考案した [22].

上述する従前のインオーダ型の実行方式では，問合せ処理の実行時に，演算ノードにおいて入出力と演算を逐次的に繰り返し，実行時間としては，入出力にかかる応答時間が蓄積することとなる．これに対して，著者らの考案したアウトオブオーダ型の実行方式では，問合せ処理の実行時に，演算ノードにおいて新たな入出力を発行する必要が生じると，都度にタスク分解を行い，分解された並行実行可能なタスク上で入出力と関連する演算を行う．すなわち，入出力を非同期化する．ソフトウェアとしての実装は多様な方式が考えられるが，問合せ処理の実行時に，実行論理の許す限りにおいて，必要に応じてタスク分解が行われ，多数の非同期入出力がストレージに発行されることとなる．当該挙動を模式的に図 1(b) に示す．現実的には，資源量は有限であるため，利用可能な主記憶の容量によって並行実行するタスクの数の上限は制約され，また，ディスクドライブやディスクアレイ，ホストバスアダプタや OS カーネルの処理能力によって同時処理可能な非同期入出力の数の上限が制約されるものの，最近のサーバにおいては数百程度の主記憶容量は珍しくなく，またストレージシステムも数百のディスクドライブを搭載するに到っており [17]，従来に対してより多くのタスクと入出力の同時処理が可能となってきている．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 [13,20] 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の処理がなされるのが通例である．即ち，アウトオブオーダ型の実行方式では，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．従来のインオーダ型の実行方式の下では，プロセッサ性能と主記憶容量を節約するべく，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実行方式においては，ソフトウェアの非同期化によって，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の達成を狙う．リレーションを複数のパーティションに分割し，パーティション毎に事前に問合せ処理をタスク分割する技法も提案されてきたが，これはパーティション毎にインオーダ型の実行方式で問合せを処理しているに過ぎず，アウトオブオーダ型の実行方式とは根源的に異なる．

問合せ処理におけるデータベースのアクセスパスという観点では，大きく分けて，リレーション全体の走査か，索引経由のアクセスかの選択が鍵となる．リレーション間の結合は，問合せ実行時間の多くを占める場合が多い演算であるが，基底となるリレーションに対する選択性に基づき，問合せ最適化器によって，ハッシュ結合 [3,7] と索引（ネステッドループ）結合のうちいずれかが選択され，用いられてきた．前者は，リレーション全体の走査を基本とし[1]，選択性が低い（選択率の高い）場合に有効とされるが，リレーション規模がテラバイトからペタバイトへと巨大化するにつれ [8]，単純な走査が可能な機会はより限られてくるはずであり，より選択性の高い（選択率の低い）問合せの重要性が強まる可能性が高い．即ち，現状における多くの所謂ビッグデータ解析は，母集団の表面的な統計集約に留まっているものの，今後はより精緻に現実を振り返るスタイルの解析が不可欠となり，この際，インタラクティブなドリルダウン型のクエリの機動的な処理が鍵となるとみられ，索引結合に頼る比重が増えるものと推察される．アウトオブオーダ型の実行方式は，当該結合方式に対して，圧倒的な高速化ポテンシャルを有している．

アウトオブオーダ型の実行方式では，入出力の発行と演算の実行は事前にプログラムされた順序で行われるのではない．同じデータベースに対して同じ問合せを処理する場合において，同じ結果が返されることは無論のことであるが，入出力と演算が実行される順序は，その試行毎に異なる可能性が高い．非決定的な動作挙動は，他のシステムソフトウェアには見られない極めて稀な特徴であり，動作正当性の検証方法などを含めて，詳細な検討を今後

[1]索引を用いたハッシュ結合の可能性もあり，当該手段を含めたより広範な有効性の検証は別稿で議論したい．

表 1: 実験システムの諸元.

ハードウェア (Dell PowerEdge 720xd)	
プロセッサ	2x Intel Xeon E5-2680 2.70GHz (2p/16c)
主記憶	1,600MHz 64GB
RAID コントローラ	Dell PERC H710p
ディスクドライブ	2x 15,000rpm 300GB 2.5in SAS HDDs (OS) 24x 10,000rpm 900GB 2.5in SAS HDDs (database)
ソフトウェア	
OS	CentOS 5.8 x86_64 (カスタマイズ済みカーネル)
DBMS	OoODE (MySQL ベースの試作機) MySQL 5.5.24

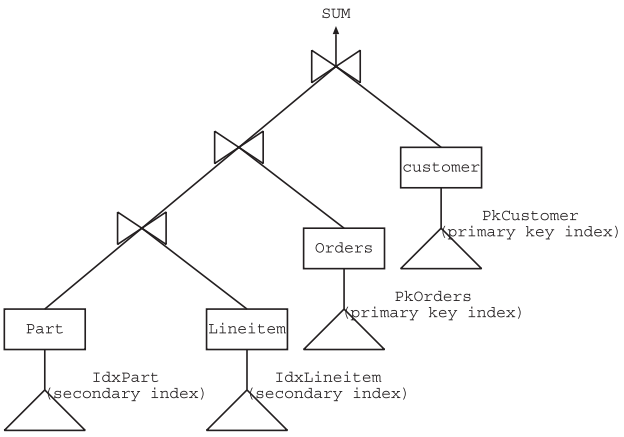


図 2: 実験に用いた問合せの実行木.

深めていきたいと考えているが，本稿では，アウトオブオーダ型のデータベースエンジンの試作実装を用いて，実験的にその動作挙動の確認を行う.

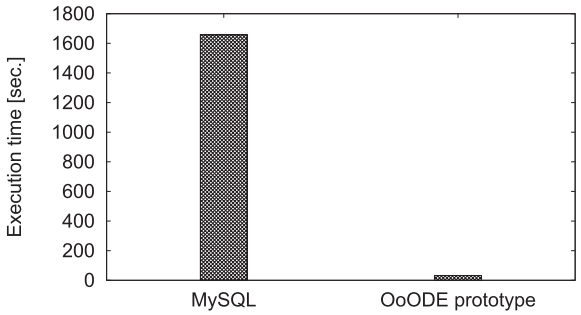
3. アウトオブオーダ型データベースエンジンの試作実装

著者らは，これまで，オープンソースのデータベースソフトウェアをベースとしたアウトオブオーダ型データベースエンジンの試作実装を行ってきた．本論文では，その一実装である MySQL [9] をベースとする試作実装を紹介する[2].

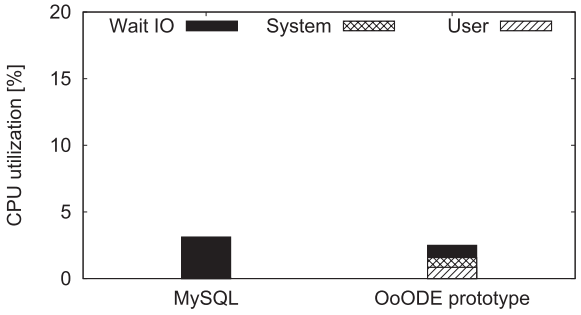
開発を行っているアウトオブオーダ型データベースエンジンは，主に問合せ処理器，ストレージエンジンならびに挙動モニタから構成される[3]．問合せ処理器は，問合せ実行木に基づき問合せ処理を実行する．演算ノードにおいて新たな入出力を発行する必要が生じると，都度にタスク分解を行い，非同期入出力をストレージエンジンに要求する．ストレージエンジンは，内部にデータベースバッファを備えており，当該バッファを経由して，問合せ処理器から要求された非同期入出力を OS を介してストレージに発行し，

[2]試作実装は InnoDB ストレージエンジンのデータフォーマットに対応している.

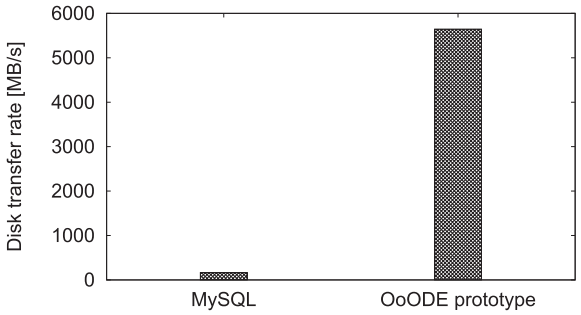
[3]本論文ではエンジンのカーネル部分に焦点を当て，他のコンポーネントである問合せ最適化器，更新・リカバリ機構，レプリケーション機構等に関しては，別稿に譲りたい.



(a) 実行時間.



(b) プロセッサ利用率.



(c) ディスク転送レート.

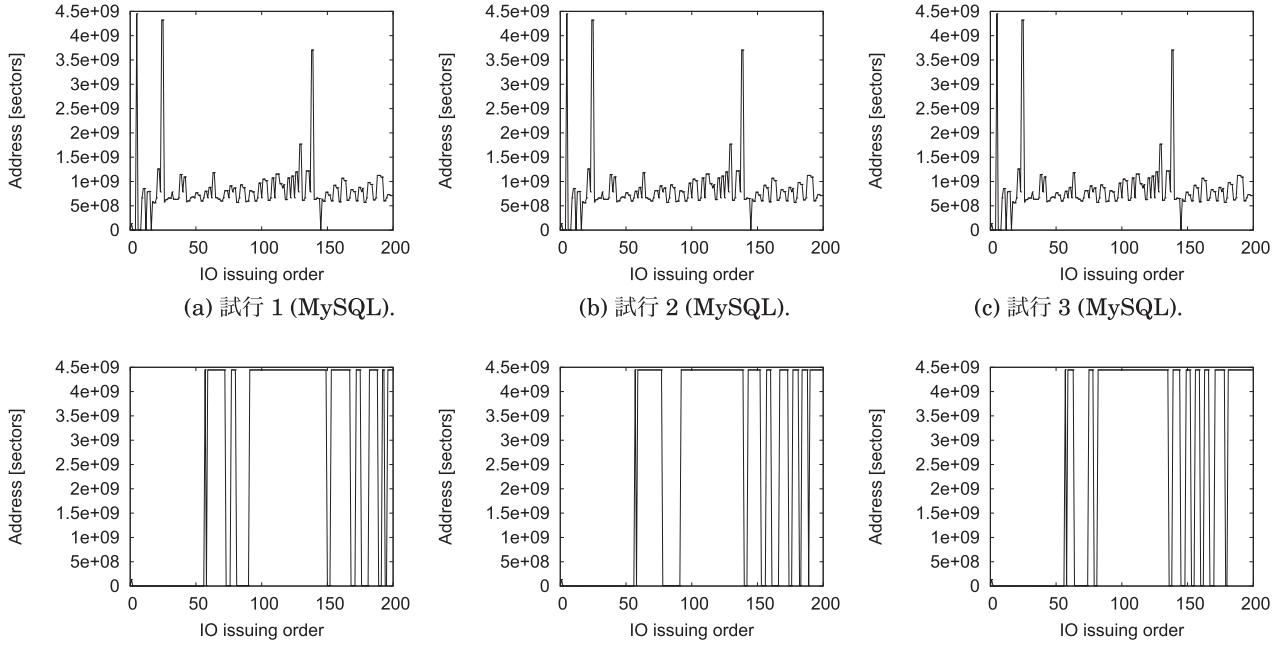
図 3: MySQL とアウトオブオーダ型データベースエンジン試作実装の性能比較.

その応答を受け付ける．ストレージエンジンが非同期入出力の完了を通知すると，問合せ処理器は当該非同期入出力に関連する演算を実行する．挙動モニタは，データベースエンジンの問合せ処理には必ずしも不可欠のものではないが，実験的な実装であることから，問合せ処理器ならびにストレージエンジンの動作挙動を精緻に捕捉するためのものであり，内部バッファを用いて問合せ処理器ならびにストレージエンジンのイベントを取得し，微視的な挙動把握によって動作デバッグ等に資するためのものである.

4. 小規模マシンを用いた実験

小規模マシン環境において前節で紹介した試作実装を用いた実験を行い，その動作挙動の確認を行った．中規模環境における性能試験に関しては，[21] を参照されたい.

表 1 に著者らが構築した実験システムの諸元を示す．24 台の SAS ディスクドライブを用いて二重パリティストライピング



(a) 試行 1 (MySQL). (b) 試行 2 (MySQL). (c) 試行 3 (MySQL).

(d) 試行 1 (OoODE 試作実装). (e) 試行 2 (OoODE 試作実装). (f) 試行 3 (OoODE 試作実装).

図 4: 入出力トレースによる MySQL とアウトオブオーダ型データベースエンジン試作実装の微視的挙動.

(RAID-6, セグメントサイズ 64KB) 編成のボリュームを構成し, 当該ボリュームからページ長 16KB, 空間長 2,560GB の InnoDB 表空間を構築した. 当該表空間には 1GB のデータベースバッファを割り当てた.

データベースエンジンの性能試験には, TPC-H ベンチマーク [18] を用いた. TPC-H の仕様に従いスキーマを作成し, この際, 主鍵に対して一次索引 (MySQL においては所謂索引構成表として編成される) を, 外部鍵に対して二次索引を構成し, スケールファクタとして 1,000 のデータセット (ローフォーマットで 1,000GB 程度) を生成して, これを表空間にインポートした. また, TPC-H 既定の問合せを簡略化した図 2 に示す問合せを作成した. この際, 最外表である `Part` リレーションの選択率は 0.0015%程度になるように調整した. 実験においては, アウトオブオーダ型データベースエンジンの試作実装を用いて当該問合せ処理を実行した際の性能を計測するとともに, 挙動モニタを用いて動作を検証した. また, 比較として, MySQL における同一の問合せ処理の実行性能を計測した.

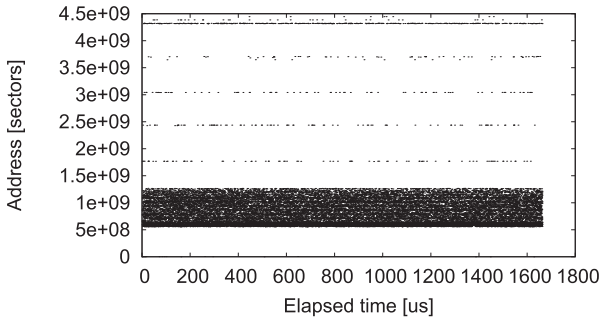
図 3(a) に問合せ処理に要した実行時間を示す. MySQL では, 問合せ処理に 1,650 秒程度を要していたが, これに対してアウトオブオーダ型データベースエンジンの試作実装においては, これを 32 秒程度に短縮し, すなわち, 約 52 倍の高速化を達成した. 図 3(b),(c) には, 問合せ処理におけるサーバのプロセッサ利用率ならびにディスクアクセススループットを示す. MySQL においては, インオーダ型の実行方式によって問合せが処理され, 原則, 1 コアのみが利用される. 問合せ処理が同期入出力によって都度ブロックされ, 当該コアのプロセッサ時間のほとんどは入出力完了の割り込み待ちに利用されており, また, 入出力スループットは高々160 IO/S 程度であって, 24 ドライブ分の帯域を有効利用できていないことが見てとれる. これに対して, アウトオブオーダ型の実行方式においては, 動的なタスク分解によって多数のタスクが並行実行され, また, 多数の非同期入出力が発行され, これによって, プロセッサ利用率は大幅に増加し, また, 入出力スループットは格段に向上している.

次に, データベースエンジンのソフトウェア挙動を, より微視的に確認したい. 図 4(a),(b),(c) には, MySQL における問合せ処理において発行される当初 200 個の入出力を, 縦軸として発行順, 横軸として発行先アドレスの空間にプロットしたものを示す. ここでは, 同じ問合せを 3 回試行したものを示すが, MySQL においては試行によって入出力が発行される順序に相違はない. これに対して, 図 4(d),(e),(f) には, アウトオブオーダ型データベースエンジンの試作実装における入出力発行挙動を示す. 同様に 3 回試行したそれぞれの実行を示すが, 明らかに試行毎に入出力の発行順序が異なっている. データベースエンジンが結果的に返す問合せの回答は同じであるものの, その実行過程が非決定的であるという特性が見て取れる.

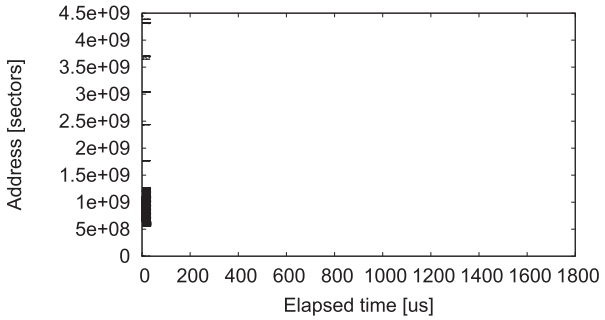
図 5 には, 問合せ処理全体の入出力トレースの結果を示す. ここでは横軸は発行順序ではなく, 経過時間とした. 視覚的に, インオーダ型の実行方式である MySQL と比べて, アウトオブオーダ型データベースエンジンの試作実装は, 格段に稠密に入出力を発行しており, 当該特性が問合せ処理の高速性の源泉となっていることが判る.

5. 結論

本論文では, 著者らが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 (OoODE) と称する高速データベースエンジンを紹介した. 当該データベースエンジンは, 問合せ処理を動的にア



(a) MySQL.



(b) OoODE 試作実装.

図 5: 入出力トレースによる MySQL とアウトオブオーダ型データベースエンジン試作実装の全体挙動.

ンフォールドし，すなわち，実行時にタスク分解する機能を有しており，高多重のタスク並行実行ならびに高多重の非同期入出力発行を行うことによって問合せを処理を行う点に特徴があり，マルチコアプロセッサの演算パワーとディスクストレージの入出力帯域を高効率に活用することを可能とする利点を備えている．中程度の選択性を有する問合せにおいて，従来型の逐次的な実行方式によるエンジンと比べて，問合せ処理の高速化が期待されている．本論文では，これまで著者らが進めてきたオープンソース DBMS である MySQL をベースとするアウトオブオーダ型データベースエンジンの試作機の実装を示し，小規模環境において TPC-H ベンチマークを用いた性能比較を行い，MySQL と比較して 52 倍程度の高速性を確認したほか，実行挙動トレースを用いてその動作挙動の非決定性の検証を行った．今後，多様な問合せに対して大規模な環境を用いた検証を進めていきたい．また，本論文ではエンジンのカーネル部分に焦点を当て論じて来たが，他のコンポーネントである問合せ最適化器，更新・リカバリ機構，レプリケーション機構等に関しても，順次報告してゆきたい．

[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」の助成により行われた．

[文献]

[1] Daniel J. Abadi, Samuel Madden, and Miguel Ferreira. Integrating compression and execution in column-oriented database systems. In *Proc. ACM SIGMOD Conf.*, pp. 671–682, 2006.

[2] Peter A. Boncz, Martin L. Kersten, and Stefan Manegold. Breaking the memory wall in MonetDB. *Comm. ACM*, Vol. 51, No. 12, pp. 77–85, 2008.

[3] 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 *Proc. Int'l. Conf. on Very Large Data Base*, pp. 228–237, 1986.

[4] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison Wesley, 1994.

[5] Kazuo Goda and Masaru Kitsuregawa. The History of Storage Systems. *Proc. of IEEE*, Vol. 100, No. Special Centennial Issue, pp. 1433–1440, 2012.

[6] Masaru Kitsuregawa, Kazuo Goda, and Takashi Hoshino. Storage Fusion. In *Proc. of 2nd Int'l Conf. on Ubiquitous Information Mgmt. and Comm.*, pp. 287–294, 2008.

[7] Masaru Kitsuregawa, Hidehiko Tanaka, and Tohru Moto-Oka. Application of Hash to Data Base Machine and Its Architecture. *New Generation Comput.*, Vol. 1, No. 1, pp. 63–74, 1983.

[8] McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity, 2011.

[9] Oracle Corp. MySQL: The World's Most Popular Open Source Database. http://www.mysql.com/.

[10] Oracle Corp. Hybrid Columnar Compression (HCC) on Exadata. An Oracle White Paper, 2012.

[11] Phil Francisco. The Netezza Data Appliance Architecture: A Platform for High Performance Data Warehousing and Analytics. *IBM Redbooks*, 2011.

[12] Donovan A. Schneider and David J. DeWitt. Tradeoffs in processing complex join queries via hashing in multiprocessor database machines. In *Proc. Int'l. Conf. on Very Large Data Base*, pp. 469–480, 1990.

[13] Margo I. Seltzer, Peter M. Chen, and John K. Ousterout. Disk Scheduling Revisited. In *Proc. USENIX Tech. Conf.*, pp. 313–323, 1990.

[14] M. Stonebraker, Eugene Wong, Peter Kreps, and Gerald Held. The Design and Implementation of INGRES. *ACM Trans. Database Syst.*, Vol. 1, No. 3, pp. 189–222, 1976.

[15] Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alex Rasin, Nga Tran, and Stanley B. Zdonik. C-Store: A Column-oriented DBMS. In *Proc. Int'l. Conf. on Very Large Data Base*, pp. 553–564, 2005.

[16] Ivan Sutherland. The tyranny of the clock. *Comm. ACM*, Vol. 55, No. 10, pp. 35–36, 2012.

[17] N. Takahashi and H. Yoshida. Hitachi TagmaStore Universal Storage Platform: Virtualization without Limits. White Paper, Hitachi Ltd., 2004.

[18] Transaction Processing Performance Council. TPC-H, an ad-doc, decision support benchmark. http://www.tpc.org/tpch/.

[19] Eugene Wong and Karel Youssefi. Decomposition — a strategy for query processing. *ACM Trans. Database Syst.*, Vol. 1, No. 3, pp. 223–241, 1976.

[20] Bruce L. Worthington, Gregory R. Ganger, and Yale N. Patt. Scheduling Algorithms for Modern Disk Drives. In *Proc. ACM SIGMETRICS Conf.*, pp. 241–251, 1994.

[21] 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. 電子情報通信学会／日本データベース学会データ工学と情報マネジメントに関するフォーラム, pp. F3–2, 2013.

[22] 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 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.

合田 和生 **Kazuo GODA**
東京大学生産技術研究所特任准教授．2000 東京大学工学部電気工学科卒業．2002 同大学院工学系研究科修士課程修了．2005 同大学院情報理工学系研究科博士課程単位取得満期退学．同年，博士（情報理工学）．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，USENIX 各会員．

豊田 正史 **Masashi TOYODA**
東京大学生産技術研究所准教授．1994 東京工業大学理学部情報科学科卒業．1996 同大学院情報理工学研究科修士課程修了．1999 同大学院情報理工学研究科博士後期課程修了．博士（理学）．同年，科学技術振興事業団計算科学技術研究員．ウェブマイニング，ユーザインタフェース，ビジュアルプログラミングに興味をもつ．ACM，IEEE CS，情報処理学会，日本ソフトウェア科学会各会員．

喜連川 優 **Masaru KITSUREGAWA**
国立情報学研究所所長，東京大学生産技術研究所教授．1978 東京大学工学部電子工学科卒業．1983 同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．データベース工学の研究に従事．本会理事，情報処理学会，電子情報通信学会，ACM，IEEE 各フェロー．電子情報通信学会データ工学研究専門委員会委員長，ACM SIGMOD Japan Chapter Chair，情報処理学会副会長，VLDB Trustee, IEEE ICDE, PAKDD, WAIM などステアリング委員歴任．