フラッシュストレージ環境における アウトオブオーダ型データベースエンジン OoODE の 実験的クエリ処理性能評価

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　〒 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
††† 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: †haya@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，クエリ処理の動的タスク分解と高多重非同期入出力発行に基づき，入出力帯域および並列演算性能を高効率に活用することで高速なクエリ処理を実現するデータベースエンジンである．本論文では近年高速ストレージとして注目を集めるフラッシュストレージ環境を用いた評価実験により，OoODE のクエリ処理性能評価を行う．

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，フラッシュストレージ，クエリ処理

1. はじめに

計算機システムにおけるストレージの新たな選択肢として，フラッシュメモリを用いたストレージが近年注目を集めている．今日のフラッシュストレージは同一規模の磁気ディスクストレージと比較して高スループット・低遅延な入出力性能を持ち，高性能なエンタープライズシステムにおける活用が広まりつつある．主要ベンダが先端的なデータベースシステムの性能を競う TPC ベンチマーク [1] において，フラッシュストレージを採用するシステムが増えていることはその好例である．フラッシュメモリの低価格化・大容量化が進んだ結果，もはやフラッシュメモリのみから成る高速ストレージ環境も珍しいものではない．IT システムの扱うデータ量は加速度的に増加し続けると言われており [2]，計算機システムのデータ管理を担うデータベースシステムにおいて，フラッシュストレージの性能活用はその高性能化に大きく資することが期待される．

データベースシステムにおけるフラッシュストレージの本格的な性能活用を実現するためには，その入出力帯域の高効率な活用が鍵となる．外部記憶装置として用いられる一般的なフラッシュストレージは，SSD に見られるように，NAND フラッシュチップを多チャンネル接続してモジュール化し，これを多数配列することにより高い入出力帯域を実現している．即ち，フラッシュストレージの入出力スループットを最大限に引き出すためには，多数の入出力チャネル・フラッシュチップの帯域を飽和させるに足る入出力要求を多重発行する必要がある．加えて，磁気ディスクストレージと比べて高入出力スループット・低遅延という特性により，システム全体の性能ボトルネックが相対的にプロセッサ側へとシフトすることが予測される [3]．近年はプロセッサにおけるクロック遅延の縮減はもはや見込まれず，マルチコア化による演算性能向上の傾向が明白であることから，フラッシュストレージの性能活用においてマルチコア性能の活用は必須であるといえよう．

著者らの研究グループでは，**アウトオブオーダ型データベースエンジン OoODE** [4] [5] [6] 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．OoODE は**アウトオブオーダ型**と称するクエリ実行方式に基づき，クエリ実行における演算処理を動的にタスク分解して，演算に要するデータに対して高多重に非同期入出力要求を発行し，入出力完了を契機として並列演算を駆動することでクエリ実行を行うデータベースエンジンである．アウトオブオーダ型データベースエンジンは，従前の**インオーダ型クエリ実行方式**，即ち同期的入出力と逐次的演算実行に基づくクエリ実行方式に基づくデータベースエンジンと比べて，ストレージの入出力帯域，及びマルチコアプロセッサの並列演算性能を高効率に活用することができる．即ち，アウトオブオーダ型データベースエンジンは，フラッシュストレージの入出力性能を本格的に活用することを可能とし，非常に高いクエリ処理性能を発揮することが期待される．本論文ではマルチコアサーバおよびエンタープライズ級フラッシュアレイから成るデータベースシステムにおいて，著者らが開発するアウトオブオーダ型データベースエンジン試作実装を用いて実験を行い，フラッシュストレージ環境におけるクエリ処理性能評価を行った．

本論文の構成は次の通りである．第 2 節においてアウトオブオーダ型クエリ実行方式によるフラッシュストレージ活用の可能性について議論し，第 3 節で著者らが構築した実験環境について紹介し，第 4 節において TPC-H データセットを用いたクエリ処理性能評価の実験結果について説明する．第 5 節で関連する研究について言及し，第 6 節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によるフラッシュストレージ入出力性能活用

一般的なデータベースシステムは，クエリを受け付けると具

体的な演算方式を表すクエリ実行プラン木を生成し，各演算ノードを逐次的に辿りながら演算を繰り返すことでクエリ実行を行う．クエリ実行の過程において演算対象データの取得が必要な場合，当該データの格納されたストレージに対して同期的に入出力要求を発行し，要求完了を待ってから演算を行う．このようなクエリ実行方式を**インオーダ型クエリ実行方式**と称する．Ingres [7] にみられる初期のデータベースシステム実装以降から今日に至るまで，インオーダ型クエリ実行方式は最も基本的なクエリ実行方式として用いられている [8]．

インオーダ型クエリ実行方式を用いた場合，1 つのクエリ実行に際してデータベースシステムから同時発行される論理的な入出力要求は高々1 である．これは，フラッシュストレージの性能活用という観点において好ましくない．外部記憶装置として用いられるフラッシュストレージは，一般に SSD 等のフラッシュモジュールを多数配列することで，高い入出力スループットを有する構成をとる．同時発行される入出力要求数が限られている場合，これら多数のフラッシュモジュールのごく一部のみが利用されるに留まり，フラッシュストレージの入出力性能を最大限に引き出すことは難しい．また演算は逐次実行されるため，クエリ実行に使用されるプロセッサ資源は限定的である．2005 年頃よりプロセッサのシングルスレッド処理性能には大きな変化がみられない一方で，NAND フラッシュは毎年 30 から 40%の容積密度の向上 [9]，および毎年 40 から 50%の容量単価の低下傾向 [10] にあり，フラッシュストレージはより高密度化，高スループット化する傾向にあることを鑑みると，インオーダ型クエリ実行ではプロセッサにおける逐次演算実行が律速要因となってゆくことが懸念される．つまり，フラッシュストレージの性能活用においては，多数の入出力要求を同時発行するとともに，複数のプロセッサコアによる並列演算性能を活用することは必須であるといえよう．

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，高多重非同期入出力発行による入出力帯域の活用と，入出力駆動の並列演算実行による並列演算性能の活用によって，特に中程度の選択性を有する分析的クエリ実行で高速なクエリ実行を実現するデータベースエンジンである．磁気ディスクストレージ環境においては高いクエリ処理性能を示すことが既に知られているが [6]，高多重非同期入出力発行，並列演算実行というアウトオブオーダ型データベースエンジンの特徴は，フラッシュストレージ環境においてもその性能活用に有効であると期待される．ただし，フラッシュストレージは磁気ディスクストレージと比べて高スループット・低遅延であるため，プロセッサ資源の効率的な活用がより重要になると予見される．本論文では，フラッシュストレージ環境においてマルチスレッド化されたアウトオブオーダ型データベースエンジン試作実装を用いた実験を行うことで，そのクエリ処理性能評価を行い，フラッシュストレージの性能活用に関する考察を行った．

3. 評価実験環境

アウトオブオーダ型データベースエンジンのクエリ処理性能の評価実験を行うべく，著者らはフラッシュストレージを用いたデータベースシステムを構築した．表 1 にその諸元を示す．サーバ・フラッシュストレージ間の入出力パスは，サーバ側 SAS HBA とストレージ側 SAS Expander が総計 8 本の 4 ワイドポート SAS ケーブルによって結線され，各 SAS Expander あたり 10 台，総じて 40 台の SSD が接続されている．

表 1　実験システム諸元

Server: Sun Fire X4470 M2	
Processor	4x Intel Xeon E7-4870 10-core 2.4GHz Processor
Memory	64x 4GB DDR3-1333MHz DIMM
HBA	1x 6Gbbs SAS PCIe HBA internal 8port (for OS) 4x 6Gbps SAS PCIe HBA 8port (for database)
HDD	4x 300GB 10000rpm 2.5inch SAS HDD (for OS)
Flash Storage: Sun Storage F5100	
Interconnect	8x 4-wide 3Gbps SAS cable 4x SAS 36port Expander
SSD	40x 24GB SATA SLC (for database)

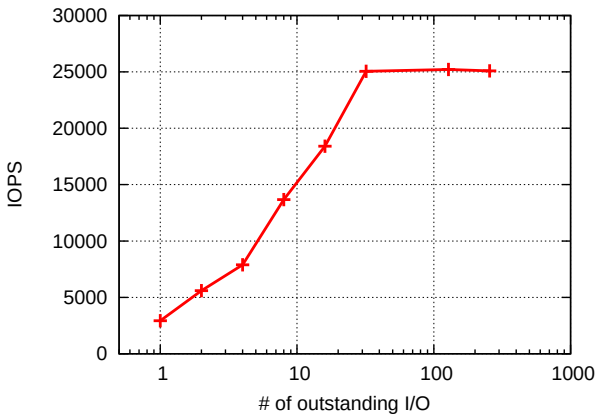

図 1　SSD 単体の IOPS 性能

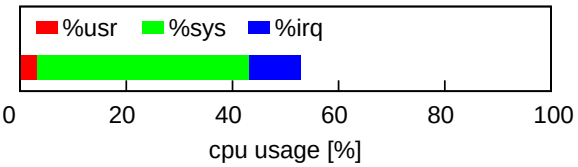

図 2　SSD 単体，64 並列入出力発行時の CPU 利用率

フラッシュストレージ環境におけるクエリ処理性能の評価に先立ち，マイクロベンチマークを用いたフラッシュストレージの入出力性能測定を行った．当該マイクロベンチマークは Linux カーネルの非同期入出力機構を利用し，4KB 単位の入出力要求を論理ブロックアドレス空間内でランダムに発行するプログラムであり，並列入出力発行数および入出力発行を行う CPU コアを指定することが可能である．

まず，フラッシュストレージを構成する SSD 単体の入出力性能を明らかにするため，並列入出力発行数を 1 から 256 まで変化させて IOPS (注1) を測定した．入出力発行コアとしては 1

(注1)：IOPS: 1 秒あたりに処理される入出力要求数

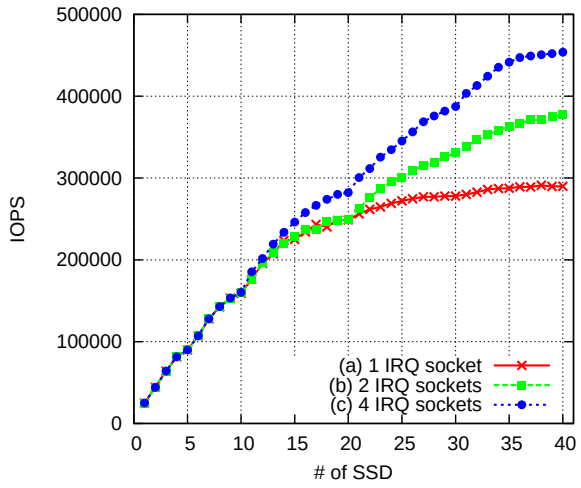

図 3　SSD 利用数と IOPS 性能

プロセッサコアのみを用いるよう指定した．結果を図 1 に示す．並列入出力発行数が 1 のときは 2,920 IOPS であり，32 まで増加するに伴い IOPS は上昇し，32 以上ではほぼ一定値を取り，その性能は 25,000 IOPS であった．この結果より，SSD 単体の入出力性能活用においても入出力要求の多重発行が必要であり，その帯域を飽和させるためには 32 以上の入出力要求を並列発行することが必要であるとわかる．図 2 に 64 並列入出力発行時の CPU 利用率を示す．SSD 単体の入出力帯域を使い切るには，入出力発行処理が 3.2%，OS カーネル内処理が 40%，割り込み関連処理が 9.7%と，総じて 50%以上の CPU コア資源が必要であることが読み取れる．

次に，複数の SSD に対して同時に入出力を行う際の性能を明らかにすべく，同時利用 SSD 数を 1 から 40 まで変化させた際の IOPS 性能を測定した．全プロセッサコアがそれぞれ入出力対象の全 SSD に入出力要求を発行するよう指定し，SSD 1 台あたりが同時に受ける入出力要求数が 64 となるよう指定を行った．また，入出力要求完了の割り込み処理を行う CPU ソケット (IRQ ソケット) の数を変化させ，複数の場合について測定を行った．結果のグラフを図 3 に示す．グラフの系列はそれぞれ (a) 1 IRQ ソケット，(b) 2 IRQ ソケット，(c) 4 IRQ ソケットの場合を示す．入出力対象 SSD が 10 台未満の場合にはいずれも性能に目立った差はなく，台数に応じた IOPS の増加がみられた．一方，SSD10 台以上になると (c) の 4 IRQ ソケットの場合がそれ以外に比べて高い IOPS を示し，20 台以上では (b) の 2 IRQ ソケットが (a) の 1 IRQ ソケットよりも高い IOPS を示す結果となった．4 IRQ ソケットの下で SSD40 台に対し総計 2,560 並列入出力要求を行った際に，最大性能 454,000 IOPS が得られた．この結果は，フラッシュストレージ活用においてプロセッサ資源配分がシステム全体の性能に大きく影響を与えることを示唆している．

4. TPC-H データセットを用いたクエリ処理性能評価

第 3 節で紹介した実験環境において，著者らの開発するアウトオブオーダ型データベースエンジン試作実装を用いたクエリ処理性能評価実験を実施した．試作実装の詳細については [5] [6] を参照されたい．評価に際しては，意思決定支援系の業界標準ベンチマーク TPC-H [1] のデータセットを用いることとし，仕様にしたがってスキーマを定義したテーブル空間に，`dbgen` をスケールファクタ 100 で実行することで生成したデータセットを読み込み，その上でクエリ実行に必要な索引の構築を行った．索引を含めたデータベース容量は約 250GB 程度であった．データベースはフラッシュストレージ上の SSD40 台に対して 4KB ストライプで直接展開し格納した．また入出力の割り込み処理には 4CPU ソケットを割り当てることとした．

```
SELECT COUNT(*), SUM(L_QUANTITY)
  FROM CUSTOMER C
    INNER JOIN ORDERS O
      ON C.C_CUSTKEY = O.O_CUSTKEY
    INNER JOIN LINEITEM L
      ON O.O_DERKEY = L.L_ORDERKEY
  WHERE [CUSTOMER selection condition]
```

図 4　性能評価用クエリ

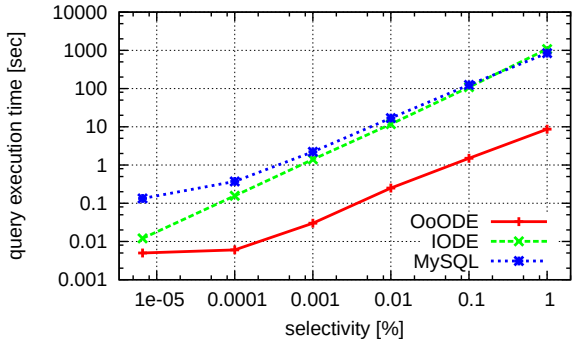

図 5　クエリ選択率と実行時間

4.1 選択率とクエリ処理性能

アウトオブオーダ型データベースエンジンにより，高速なクエリ処理性能が達成される有効領域を明らかにすべく，図 4 に示すクエリの選択条件を調整することにより選択率を 6.6×10^{-6}%から 1.0%まで変化させながら，その実行時間の測定を行った．アウトオブオーダ型データベースエンジン試作実装においては，当該クエリの実行には索引走査・ネステッドループ結合からなる実行プランを用いた．またアウトオブオーダ型クエリ実行による処理性能向上の効果を確認するため，試作実装における入出力を同期入出力に切り替え，インオーダ型クエリ実行を行う場合についても併せて測定を行った．さらにこれに加えて，試作実装の妥当性を確認するためにオープンソース DBMS として代表的な MySQL を用いて同様の測定を行った．

結果を図 5 に示す．凡例の “IODE” は試作実装によりインオーダ型クエリ実行を行った場合，“OoODE” は試作実装によりアウトオブオーダ型クエリ実行を行った場合，“MySQL” は MySQL によりクエリ実行を行った場合を表す．索引走査

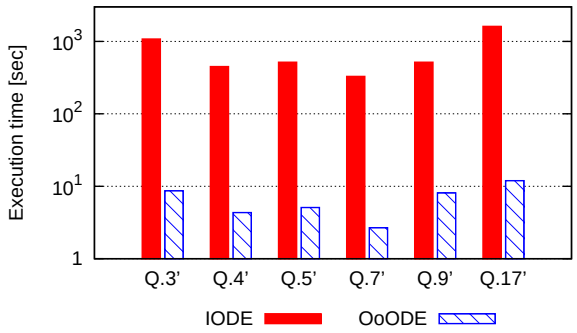

図 6　TPC-H 類似クエリ実行時間

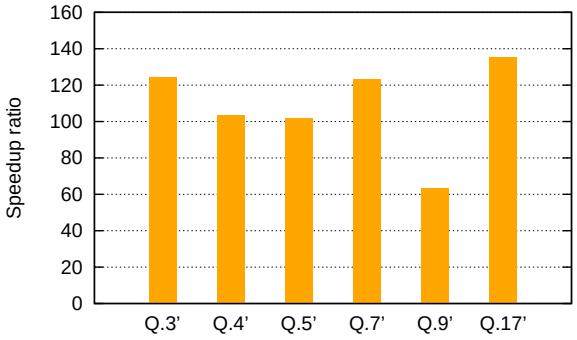

図 7　OoODE による TPC-H 類似クエリ性能向上率

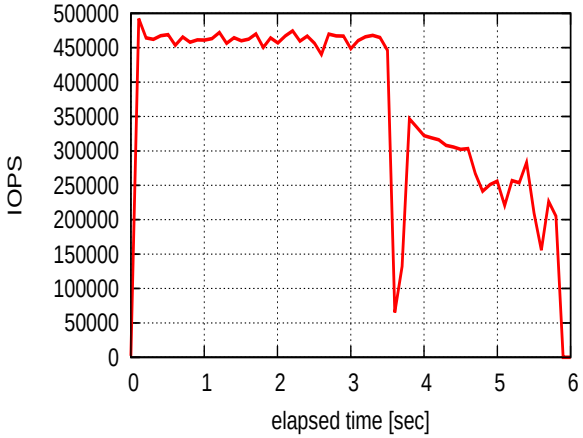

図 8　TPC-H Q.3 類似クエリ実行時の IOPS 経時変化

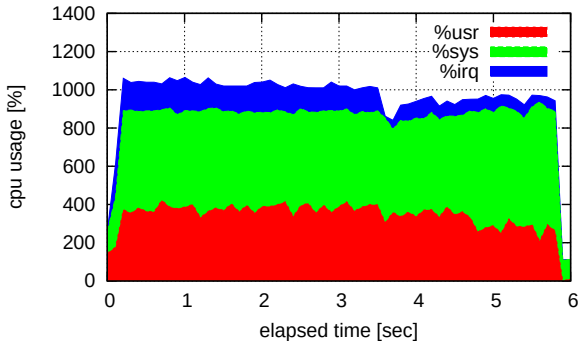

図 9　TPC-H Q.3 類似クエリ実行時の CPU 使用率経時変化

およびネステッドループ結合によってクエリ実行を行う場合，選択率が大きくなるに従って入出力を行うデータ量は増加し，それに伴いクエリ実行時間が長くなる．実際に IODE および MySQL においてその傾向が確認できる．MySQL は IODE と同じくインオーダ型クエリ実行方式を採用しており，クエリ実行開始に要する 0.1 秒程度のオーバヘッドが顕在化する選択率 0.00001%以下の領域を除くと，両者の実行時間に大きな差は見られない．このことから，試作実装のインオーダ型クエリ実行は妥当なものであると判断できる．アウトオブオーダ型クエリ実行を行う場合には，選択率 0.0001%以上において大幅な実行時間の短縮が確認でき，高いクエリ処理性能を実現していることがわかる．インオーダ型クエリ実行に対する性能向上率は，本測定においては選択率 1.0%において最大となり 124 倍であった．この結果より，幅広いクエリ選択率の領域において，アウトオブオーダ型データベースエンジンが高いクエリ処理性能を発揮することが示された．

4.2　複数の TPC-H 類似クエリによる評価

複数の異なる種類のクエリ実行において，アウトオブオーダ型クエリ実行によるクエリ処理性能の向上を評価するため，TPC-H 規定クエリについて一部条件を簡略化したものを，選択率が 0.1 から 1.0%程度となるよう調整することでクエリを作成し，これらを実行してその実行時間を測定した．

図 6 に各クエリの実行時間を示す．凡例の IODE はアウトオブオーダ型データベースエンジン試作実装をインオーダ型クエリ実行方式で動作させた場合，OoODE はアウトオブオーダ型クエリ実行方式で動作させた場合を表す．また図 7 に IODE の実行時間を基準としたときのアウトオブオーダ型クエリ実行による性能向上率を示す．クエリ毎のテーブル選択条件や結合濃度が異なることに起因するばらつきはあるものの，総じて IODE に対し OoODE は非常に短い時間でクエリ実行を完了することが確認でき，性能向上率にすると 5 つのクエリは 100 倍を超える性能を達成し，Q.17 類似クエリにおいて最大の 135 倍であった．この結果より，アウトオブオーダ型データベースエンジンは複数の異なるワークロードにおいても有効にフラッシュストレージの入出力性能を活用し，インオーダ型エンジンに比べて非常に高いクエリ処理性能を達成可能であるといえる．

4.3　クエリ実行時挙動

アウトオブオーダ型データベースエンジンのクエリ実行挙動をより詳細に確認するため，TPC-H Q.3 類似クエリをアウトオブオーダ型データベースエンジンにより実行し，実行中の経時的な IOPS および CPU 利用率の測定を行った．結果を図 8, 9 に示す．クエリ実行開始直後より IOPS および CPU 使用率が上昇し，約 3.5 秒の間 460,000 IOPS，1,000%前後を推移した．直後の IOPS，CPU 使用率の落ち込みは，エンジン内部のタスク分解によって生成された処理単位タスクの総数が増加から減少に転じるたことで，入出力帯域を飽和させるに足る非同期入出力発行が行われなくなったことに起因する．その後は処理単位タスク数の減少に伴い，IOPS は徐々に低下してゆく．CPU 使用率に関してもデータベース演算に係る %usr は低下してゆくが，プロセッサコア間でタスク再分散を行うコヒーレンシコスト増加により%sys が増加し，CPU 使用率全体として

は大幅な増減なくクエリ実行完了まで推移した．本実験結果より，クエリ実行がシステム性能を引き出すに足る処理並列性を有する段階においては，アウトオブオーダ型データベースエンジンはフラッシュストレージの入出力スループットを最大限に活用し，またマルチコア資源を有効しながらクエリ実行が可能であることが分かる．本実験に用いた試作実装においては，処理並列性がある程度減少した段階で IOPS の低下がみられたが，その際にも最大時の 50%を超える程度には入出力帯域を活用しており，全体を通して高いクエリ処理性能を実現するに至っていることが確認できた．

5. 関連研究

データベースシステムにおけるフラッシュストレージ性能活用の取り組みとしては，その低遅延性を活かしバッファプールの拡張領域として SSD を用いる手法 [11] [12] や，高速なトランザクションログ格納領域として用いる手法 [13] [14]，入出力並列性に着目した B^+ 木索引構造の改善 [15] などがある．また NAND フラッシュはランダム書き込みに対し順次書き込みが高速である点に着目し，書き込みを順次書き込みに変換可能なデータレイアウトを用いることで性能改善を図る取り組み [16] [17] もみられる．一方，クエリ実行方式そのものの転換によるフラッシュストレージの性能活用は著者らの知る限りにおいてこれまで行われておらず，新規な取り組みであるといえる．

6. おわりに

本論文では，高速なデータベースストレージとして注目されるフラッシュストレージ環境において，著者らが開発をすすめるアウトオブオーダ型データベースエンジン OoODE のクエリ処理性能評価を行った．エンタープライズ級フラッシュストレージを用いた実験環境を構築し，100GB 規模の TPC-H データセットを用いてアウトオブオーダ型データベースエンジン試作実装のクエリ処理性能評価を実施したところ，選択率 6.6×10^{-6}%から 1.0%という範囲において高いクエリ処理性能を示し，複数のクエリにおいてインオーダ型クエリ実行に比べ 100 倍以上の性能向上が得られることが確認された．今後は，より多様なクエリに対して大規模な環境を用いた性能評価を進めてゆきたい．また本論文ではデータベースエンジンにおけるクエリ実行方式に焦点を当て考察したが，フラッシュによる高速ストレージ環境に適合した総体的なソフトウェアアーキテクチャについても検討を進めてゆきたい．

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」，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] Transaction Processing Performance Council. http://www.tpc.org/.

[2] 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.

[3] 出射英臣, 久木和也, 藤原真二, 茂木和彦, 合田和生, 喜連川優. フラッシュメモリ構成のストレージ環境における商用アウトオブオーダ型データベースエンジンの性能にプロセッサ省電力モードが与える影響の評価. 第 6 回データ工学と情報マネジメントに関するフォーラム, Mar 2014. (to appear).

[4] 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.

[5] 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, Mar 2013.

[6] 合田和生, 早水悠登, 喜連川優. 100 ドライブ規模のディスクストレージ環境におけるアウトオブオーダ型データベースエンジン OoODE の問合せ処理性能試験. 電子情報通信学会論文誌, Apr 2014. (to appear).

[7] G. D. Held, M. R. Stonebraker, and E. Wong. Ingres: A relational data base system. In *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*, AFIPS '75, pp. 409–416, New York, NY, USA, 1975. ACM.

[8] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 3 edition, 2003.

[9] R.E. Fontana, G.M. Decad, and S.R. Hetzler. The impact of areal density and millions of square inches (msi) of produced memory on petabyte shipments of tape, nand flash, and hdd storage class memories. In *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*, pp. 1–8, 2013.

[10] Robert E. Fontana, S.R. Hetzler, and G. Decad. Technology roadmap comparisons for tape, hdd, and nand flash: Implications for data storage applications. *Magnetics, IEEE Transactions on*, Vol. 48, No. 5, pp. 1692–1696, 2012.

[11] Mustafa Canim, George A. Mihaila, Bishwaranjan Bhattacharjee, Kenneth A. Ross, and Christian A. Lang. Ssd bufferpool extensions for database systems. *Proc. VLDB Endow.*, Vol. 3, No. 1-2, pp. 1435–1446, September 2010.

[12] Jaeyoung Do, Donghui Zhang, Jignesh M. Patel, David J. DeWitt, Jeffrey F. Naughton, and Alan Halverson. Turbocharging dbms buffer pool using ssds. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pp. 1113–1124, New York, NY, USA, 2011. ACM.

[13] Sang-Won Lee, Bongki Moon, Chanik Park, Jae-Myung Kim, and Sang-Woo Kim. A case for flash memory ssd in enterprise database applications. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1075–1086, New York, NY, USA, 2008. ACM.

[14] Shimin Chen. Flashlogging: Exploiting flash devices for synchronous logging performance. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pp. 73–86, New York, NY, USA, 2009. ACM.

[15] Hongchan Roh, Sanghyun Park, Sungho Kim, Mincheol Shin, and Sang-Won Lee. B+-tree index optimization by exploiting internal parallelism of flash-based solid state drives. *Proc. VLDB Endow.*, Vol. 5, No. 4, pp. 286–297, December 2011.

[16] Radu Stoica, Manos Athanassoulis, Ryan Johnson, and Anastasia Ailamaki. Evaluating and repairing write performance on flash devices. In *Proceedings of the Fifth International Workshop on Data Management on New Hardware*, DaMoN '09, pp. 9–14, New York, NY, USA, 2009. ACM.

[17] Sang-Won Lee and Bongki Moon. Design of flash-based

dbms: an in-page logging approach. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pp. 55–66, New York, NY, USA, 2007. ACM.