データベース更新差分を用いた範囲検索の IO コスト推定

Incremental IO Cost Estimation of Range Scan Using Update Difference of Database

星野 喬♠ 合田 和生♦ 喜連川 優♦

Takashi HOSHINO Kazuo GODA
Masaru KITSUREGAWA

DB 再編成は，データ再配置をすることで，更新による DB の構造劣化を解消し，性能を回復するための処理であり，DBMS にとって不可欠である．本稿では再編成の自立化を目的とする．再編成自立化のためには，DB の構造劣化を低オーバーヘッドで正確に把握する必要がある．本稿では，ストレージ内の IO 性能特性を考慮した IO コストモデルを構築した．それを用いてクラスタ表に対する範囲検索の構造劣化量を IO コストで表現し，DB 更新差分を利用することで，低オーバーヘッドかつ高精度に推定する手法を提案した．提案手法を MySQL に実装し，TPC-H ベンチマークのデータに対する範囲検索を用いて評価した．提案手法は再編成タイミングの判断に有用であることが期待される．

Database reorganization that counteracts structural deterioration caused by data updates to recover performance is an essential task in database administration. This paper targets autonomic database reorganization. Autonomic database reorganization requires accurate measurement of structural deterioration with a little overhead. In this paper, we proposed a method to estimate IO cost of range scan of clustered table accurately considering IO characteristics inside hard disk drive. The method requires only database updates without fully table scan and it can keep estimated IO cost incrementally with little overhead. We implemented the method on MySQL and evaluated it with TPC-H benchmark.

1. はじめに

近年，人類の扱うデジタルデータの容量は飛躍的に増大している．そのため データベース(以下，DB)管理はより困難化し，コスト増大が問題となっている．この問題を解決するため，DB 管理の自立化を目指す試みが増えている[3].

現在のデータベース管理システム(以下，DBMS)において実際に動作している自立管理機構として，オンライン表空間拡張機能や自動バックアップがある．より高度な判断を要するDB構成変更，性能チューニング，再編成などを自立化する試みは，未だ実現していない．DB表空間内のデータは，

♠ 学生会員 東京大学大学院情報理工学系研究科
hoshino@tkl.iis.u-tokyo.ac.jp
♦ 正会員 東京大学生産技術研究所
{kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

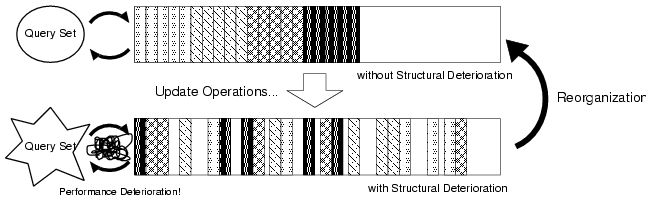


図 1 データベースの構造劣化と再編成
Fig.1 Structural deterioration and Reorganization of Database

更新操作を繰り返すことによりその構造が劣化し，本来想定している配置と大きく乖離した状態となり，性能を大幅に低下させる場合がある（図1）．このため，DB管理者は再編成を行うことにより，ストレージ内のデータを再配置し，性能低下を予め防ぐ必要がある．現状では，構造劣化を同定し再編成を開始することの判断は難しく，また，再編成自体にかかる時間は膨大であることから，再編成は高度な管理業務と考えられており，再編成を自立化させることの意義は大きい.

現在，DBMSにおける再編成では，DB管理者がDBMSから得られる性能などの統計情報や，空間情報などから構造の劣化度合を推定し，再編成が必要であるかどうかを判断し，運用に合わせて再編成処理をスケジューリングする．これらの情報に基づく判断は，管理者の勘と経験によるノウハウに多くを依存しており，具体的に体系化されたものではない．結果として，設計時に更新量を想定して，余裕をもって定期的に再編成を実行する運用が多い．しかし，DBは一度設計すると長期的に使われることが多く，設計時に想定している構造の劣化度合を越えることがあり得る．また，無駄な再編成を行うとより高性能なシステムが必要になるため，コストも高くなる.

DB再編成契機の判断を明確に行うためには，構造劣化を定量的に表現し，常に把握できるようにすることが必要である．性能低下のみを判断に用いようとする場合は，構造劣化以外の原因との切り分けがしにくいため，本稿では直接構造劣化を表現することを試みる．構造の劣化度合を把握する機構を実現すれば，再編成の自立化に大きく貢献すると同時に，これまでに実施していた余分な再編成の回数を減らすこともできる.

DBMSを含む現状のデータインテンシブアプリケーションでは，性能のボトルネックがストレージIOに起因することが多い．そこで，これらのアプリケーションを対象に，構造劣化の度合を，DBの状態から期待されるIOコストとして表現する．我々は，ストレージ内のIO性能特性を考慮した上で，MySQL[8]クラスタ表の範囲検索を対象に，実際に範囲検索問い合わせを実行した場合に期待されるIOコストを高精度かつ低オーバーヘッドでリアルタイムに推定する手法を提案する．これにより，構造劣化の度合を定量的に計測することが可能になる．本稿の提案手法は再編成契機の判断に有用であり，余分な再編成の回数を減らし，再編成の自立化に大きく貢献することが期待される.

本稿は以下のように構成される．まず，第2章で関連研究について述べ，次に第3章で提案手法を説明する．第4章で提案手法の評価を行い，最後に第5章にて結論と今後の課題を述べる.

2. 関連研究

DB 再編成に関する研究には，実行方式の高度化及び実行スケジューリングの最適化を中心に行われてきた．前者につ

いては，近年オンライン再編成方式についての研究が行われてきている．[2], [12]．後者については，再編成契機の最適化に関する研究が行われてきた[1], [5], [7], [9], [11]．再編成の自立化は，再編成契機を判断する必要があるため，後者に属する．これらの研究では，表空間を構成するストレージの性能特性は考慮せず，再編成スケジューリング手法が考察された．

文献[6]は，DB バッファ等の影響を考慮して，B+Tree 索引と表をスキャンするのに必要な IO 数がコストとしてモデル化を提案している．

著者らは，文献[13]において，DB の構造劣化の表現手法と，その定量的推定手法を提案した．一方，本稿では，表空間の構成要素であるストレージ内の IO 性能特性を考慮し，また，DB 更新差分を利用して，高精度かつ低オーバーヘッドな IO コスト推定手法を提案する．

3. 範囲検索の IO コスト推定手法

本章では，まず IO コスト推定の対象である MySQL におけるクラスタ表の実装とその構造劣化について説明する．次に，ディスクドライブの特徴に着目した IO 性能モデルを構築し，当該モデルを用いて，範囲検索の IO コスト推定手法を提案する．さらに，DB 更新差分を用いた低オーバーヘッドの推定手法について述べる．

3.1 MySQL クラスタ表の実装と構造劣化

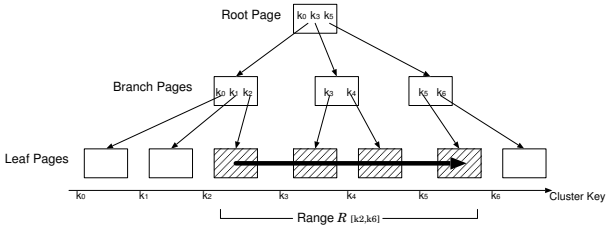


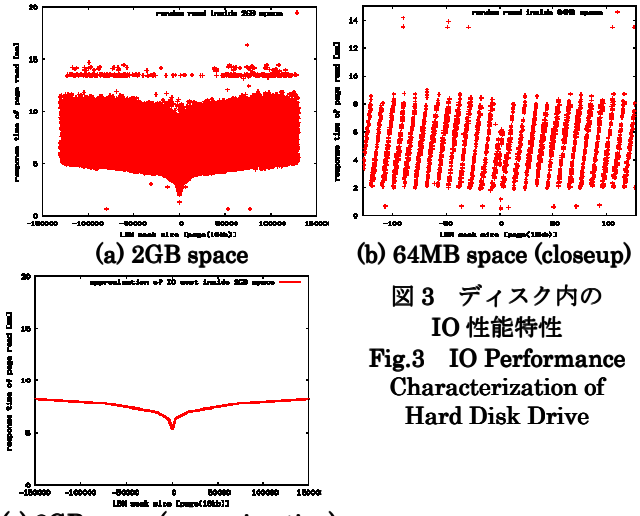
図 2 MySQL クラスタ表の B+Tree 構造
Fig.2 B+Tree Structure of Clustered Table on MySQL

MySQL のクラスタ表は，B+Tree 構造を有しており，枝及び根ページには鍵と下位ページへのポインタが，葉ページにはデータ行が格納されている(図 2 参照)．葉ページは，クラスタ鍵順に並んでいる．範囲検索は，対象となる鍵範囲(例えば，図 2 における範囲 R)に属する葉ページ群を，クラスタ鍵順序で読み込む走査である．B+Tree における葉ページの鍵順とストレージ内の物理順序が対応していることが期待されているが，更新によって B+Tree の構造変化，即ち葉ページの結合，分割が起きると，レコードの物理的な配置が乱雑化する．このため，範囲検索のディスクアクセスはランダム化し，範囲検索の性能は大幅に低下する．本稿では上記の構造劣化を議論の対象とする．

3.2 ストレージ内 IO 性能特性モデル

本稿ではデータベースは 1 台のディスクに格納されるものとする．ディスクの IO 性能特性を考慮することにより，ディスク上に直接表空間を作成した基本的な場合における精度の高い IO コスト推定が可能になる．本稿では Seagate のディスク[1]を用いて，その特性をモデル化した．

対象ディスクに負荷をかけ，図 3 の結果が得られた．横軸



(a) 2GB space (b) 64MB space (closeup)

図 3 ディスク内の IO 性能特性
Fig.3 IO Performance Characterization of Hard Disk Drive

(c) 2GB space (approximation)

に 16KB ページサイズを単位とする論理ページ番号 LBN[2] でのシーク距離を ΔLBN で示し，縦軸に各 IO のレスポンスタイムを示した．IO は全て 16KB ページ単位で発行されている．図 3(a) は，2GB のディスク空間に対し，ランダムリード負荷を与えたもので，図 3(b) は，64MB 空間にランダムリード負荷を与え，特に $|\Delta LBN| < 128$ の区間を示したものである．本稿では，対象となるページを読むのに必要な時間(レスポンスタイム)をそのページの IO コスト C_{access} とする．C_{access} はディスクの性質から，

$$C_{access} = C_{seek} + C_{rotation} + C_{transfer}$$

で近似される．ここに C_{seek} はヘッドシーク時間，$C_{rotation}$ は回転待ち時間，$C_{transfer}$ は転送時間を意味する．図 3(a) において，$|\Delta LBN|$ が大きくなるに連れ，レスポンスタイムが大きくなっている．この成分が C_{seek} である．縦軸方向へのプロットの広がりは $C_{rotation}$ の分布である．図 3(b) では，$C_{rotation}$ が ΔLBN に対して周期的に分布していることが確認できる．また，$\Delta LBN = 1$ の場合，ディスクはすぐ次のページにアクセスするため，このとき，C_{seek} 及び $C_{rotation}$ は 0 である．C_{access} は ΔLBN の値から以下のように近似することができる．

$$C_{access} = f(\Delta LBN)$$

は図 3(b) の値を， は，図 3(a) のプロットを区間ごとに線形近似した図 3(c) の値を用いる． では， の振る舞いも考慮するが， では， をディスクが一律回転する一定時間 として近似する．

[1] Cheetah 18LPST318203FC[4].

[2] Logical Block Number. ここでは LBN=LBA/16KB.

3.3 範囲検索の IO コスト推定

MySQL クラスタ表を構成する B+Tree 葉ページ群のディスク上での位置(*LBN*)は表空間をスキャンすれば把握でき，範囲検索で読み込まれるページ群の論理シーク量 ΔLBN が予測できる．B+Tree に属する全ての葉ページを p_i $\{p_i : i = 0,1,2,...,N-1\}$ とする．i はクラスタ鍵の順序に従う．ページ p_i が範囲検索で読み込まれる場合に期待される論理シーク量を $\Delta LBN_i (= LBN_i - LBN_{i-1}$，$\Delta LBN_0$ は不定)として定義する．p_i を読み込むために必要な IO コストを C_i (但し，$C_0 = 0$ とする)とすれば，$C_i = f(\Delta LBN_i)$ として計算できる．任意の範囲 R (n ページ)と対応する葉ページ範囲を $[p_{i_0},..., p_{i_0+n-1}]$ とすると，R を範囲検索するための IO コスト C_R は，

$$C_R = \sum_{i=i_0}^{i_0+n-1} C_i$$

として推定できる[3].

3.4 DB 更新差分を用いた低オーバーヘッド推定

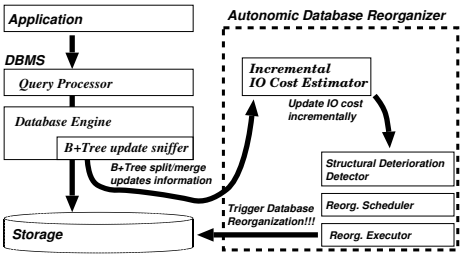


図 4 データベース更新差分抽出機構を用いた差分 IO コスト推定

Fig.4 Architecture of Incremental IO Cost Estimator Using Update Sniffer of Database

図 4 に，低オーバーヘッドで範囲検索の IO コスト推定が出来る手法の概要を示した．従来の手法[13]では，IO コスト推定が必要である毎に，表空間をスキャンする必要があった．DB システム稼働中の表空間スキャンは大きなオーバーヘッドであり，他のオンライン DB アクセスの性能を低下させうる．

提案手法では B+Tree 構造が変化したとき，すなわちページ分割及びページ結合によるときのみその情報を更新差分抽出機構が抽出し，管理ツールに送り，IO コストを差分計算する．前節で示した IO コスト推定式は，ΔLBN_i の変化したページ p_i の IO コスト C_i をページ単位で差分計算できる．範囲　を検索するための IO コストが　　であり，DB 更新した後，　　になるとすれば，差分計算式は，以下の式で表される．

は，削除されたページ集合を，　　は追加されたページ集合を表す．構造変化の前後でページに対する i の割り当てが変化し得るため i^{old} と i^{new} は区別している．本手法を用いれば，IO コスト推定の度に表空間をスキャンする必要がなくなり，オーバーヘッドは構造変化分の情報を抽出する操作のみになる．これらの情報は更新される時点でメインメモリに読み込まれているため，差分抽出操作に対してディスクへのアクセスは一切発生しない．

[3] 範囲 R におけるページ　　を読み込む直前のヘッド位置が不確定であるため IO コストは不定である．本稿では，$10 << n$ を満たす範囲検索を対象としており，　　とすることによる誤差は無視できるものとして推定を行う．

4. 評価

提案手法を MySQL[8]上にプロトタイプ実装し，評価した．DBMS は MySQL 4.1.12 InnoDB エンジンを使用した．TPC-H[10]ベンチマークを用いて実験を行った．

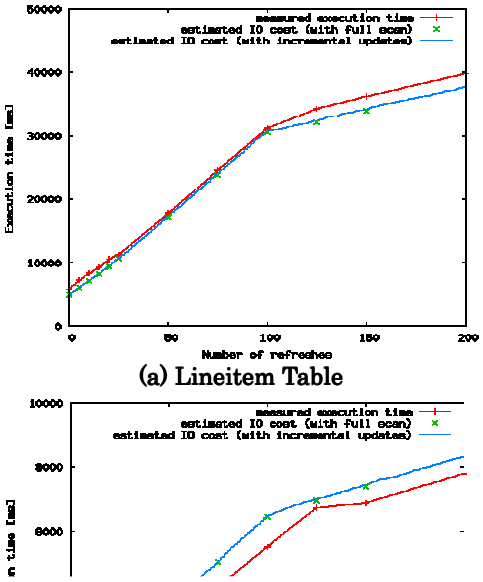
4.1 実験環境

Linux 2.4 が動作する PC 上でディスク 1 台を raw デバイスとして使用した．ディスクは IO 性能モデルを作成したものと同じ Cheetah 10K 18GB を用いた．

4.2 TPC-H ベンチマークによる実験

TPC-H スケールファクタは 0.1 として実験を行なった．表空間は実験環境の raw デバイス上で先頭から 1GB 確保し，表空間を作成した．ページサイズは 16KB とし，クラスタ表を用いて TPC-H スキーマを構築した．表空間へのデータロード時は，索引なども含めて約 220MB の空間を使用されており，更新が繰り返されることにより最大で約 300MB の空間にデータが分布した．

4.2.1 範囲検索 IO コスト推定精度



(a) Lineitem Table

(b) Orders Table

図 5 範囲検索 IO 推定コスト(スキャン，差分)と実測値

Fig.5 Estimated IO Cost (scan, diff) and Measured Time of Full Scan

初期ロードした TPC-H DB に対して，以下の更新操作を 200 回繰り返し,検索操作を適度な更新間隔で実行した．

更新操作: TPC-H ベンチマークのリフレッシュ関数 RF1 (orders と lineitem のバルク挿入)，RF2 (orders と lineitem

のバルク削除）をそれぞれ 1 回ずつ実行する．これでデータの 1%が更新される．更新によってデータ量や主鍵値の分布はほぼ変わらない．

検索操作: orders 表と lineitem 表の範囲検索(範囲 100%)を行い，その応答時間を測定する．同時に提案手法(推定の度に表空間スキャン及び差分を用いた手法)を用いて IO コストを推定する．

図 5 に，更新操作に伴う範囲検索性能の，提案手法による IO コスト推定値と，クエリ実行時間の実測値を示した．範囲検索は，IO インテンシブな操作であり，IO コストがほぼ実行時間を占めるものと考えて良い．更新操作に伴い，推定値，実測値が共に増加している．推定誤差は最大で orders 表については 17%，lineitem 表については 6%であった．また，IO コスト推定時，表空間をスキャンする手法と，更新差分のみを用いる手法では，推定精度は変化しなかった．

4.2.2 更新差分抽出機構のオーバーヘッド

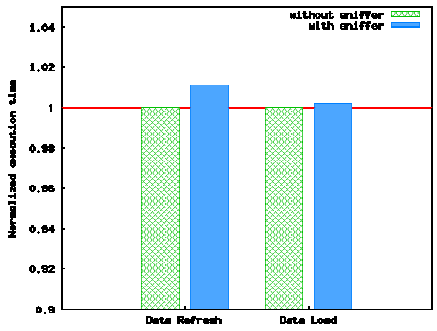


図 6 更新差分抽出機構のオーバーヘッド
Fig.6 Overhead of B+Tree Update Sniffer

DB 更新差分抽出機構を実装した MySQL と，当該機構を有しない MySQL それぞれについて，更新操作（Data Refresh）とデータロード（Data Load）の実行時間を測定し，更新差分抽出機構のオーバーヘッドを算出した．図 6 に実験結果を示した．

更新操作は 1.0%，データロードは 0.2%のオーバーヘッドであった．更新差分のみを用いた提案手法は低オーバーヘッドであると言える．

5. おわりに

本稿は，DB 更新差分を用いて，クラスタ表の範囲検索における IO コスト推定手法を提案した．ディスク性能特性を活用することにより，推定精度を高めるとともに，データベースの更新差分情報を巧みに利用することにより極めて低いオーバーヘッドで IO コスト推定を可能とした．プロトタイプ実装を用いて，リアルタイム IO コスト推定ができることを示した．

今後の課題は，範囲検索以外のクエリへの対応，より詳細な表空間のリアルタイム分析の実現，RAID 等の高度ストレージへの対応が挙げられる．

[謝辞]

本研究の一部は，文部科学省リーディングプロジェクト e-society 基盤ソフトウェアの総合開発「先進的なストレージ技術」の助成により行われた．協力企業である株式会社日立製作所より多くの有益なコメントを頂戴した．深謝する次第である．

[文献]

[1] Batory, D. S.: “Optimal file designs and reorganization points”. ACM TODS 7(1), pp.60-81 (1982).
[2](Ed.) Lomet, D.: “Special Issue on Online Reorganization”. IEEE Data Eng. Bull. 19(2) (1996).
[3] Lightstone, S., Schiefer, B., Zilio, D. and Kleewein, J.: “Autonomic Computing for Relational Databases: The Ten Year Vision”. In Proc. of AUCOPA2003 (2003).
[4] Seagate Technology LLC.: “Cheetah 18LP FC Disk Drive Product Manual Volume 1”, (1999).
[5] Lohman, G. M., and Muckstadt, J. A.: “Optimal policy for batch operations: Backup, checkpointing, reorganization, and updating”. In Proc. of SIGMOD1977, pp.157 (1977).
[6] Mackert, L. F., and Lohman, G. M.: “Index Scans Using a Finite LRU Buffer: A Validated I/O Model”. ACM TODS 14(3), pp.401–424 (1989).
[7] Maruyama, K. and Smith, S. E.: “Optimal reorganization of distributed space disk files”. Commun. ACM 19(11), pp.634–642 (1976).
[8] MySQL: The World’s Most Popular Open Source Database. http://www.mysql.com/.
[9] Shneiderman, B.: “Optimum data base reorganization points”. Commun. ACM 16(6) pp.362–365 (1973).
[10] TPC: Transaction Processing Performance Council. http://www.tpc.org/.
[11] Yao, S. B., Das, K. S. and Teorey, T. J.: “A Dynamic Database Reorganization Algorithm”. ACM TODS 1(2) pp.159–174 (1976).
[12] 合田和生，喜連川優．：“データベース再編成機構を有するストレージシステム”．情報処理学会論文誌データベース，46(TOD (SIG8)) pp.130-147 (2005).
[13] 星野喬，合田和生，喜連川優．：“データベース再編成契機決定のための性能劣化同定方式”．DEWS2005 5B-o2 (2005).

星野 喬 Takashi HOSHINO
東京大学大学院情報理工学系研究科博士課程在学中．2003 年東京大学工学部電子情報工学科卒業．2005 年東京大学大学院情報理工学系研究科修士課程修了．データベースシステムの研究に従事．情報処理学会，日本データベース学会学生会員．

合田 和生 Kazuo GODA
東京大学生産技術研究所産学官連携研究員．2000 年東京大学工学部電気工学科卒業．2005 年東京大学大学院情報理工学系研究科博士課程単位取得満期退学．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．情報処理学会，日本データベース学会会員．

喜連川 優 Masaru KITSUREGAWA
東京大学生産技術研究所教授．同所戦略情報融合国際研究センター長．1983 年東京大学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，並列処理，Web マイニングに関する研究に従事．本会理事，情報処理学会フェロー，SNIA-Japan 顧問，ACM SIGMOD Japan Chapter Chair (H11-H14)，電子情報通信学会データ工学研究専門委員会委員長(H9,10)．VLDB Trustee, IEEE TKDE Assoc. Editor, IEEE ICDE, PAKDD, WAIM Steering Comm. Member.