D-008

ロックフリー索引のための基礎ベンチマークの作成及び性能検証

Benchmark Design for Lock-Free Indexing and Performance Verification

牧田 直樹 Naoki Makita
杉浦 健人 Kento Sugiura
石川 佳治 Yoshiharu Ishikawa
陸 可鏡 Lu Kejing

1 はじめに

近年，ムーアの法則の終焉によるメニーコア環境への転換に伴い，マルチスレッド向けの索引に対する需要が増加している．特に ATM や電子商取引に用いられる OLTP システムにはリアルタイムかつ高速なデータベースの更新が要求されるため，その性能は特に重要である．

索引のマルチスレッドへの対応として，スレッドが構造へのロックを取得して排他処理を行う方法がある．しかし，ロックを取得したスレッドが作業を行う間，ロックを取得できないスレッドは待ち状態となる．そのため，ロックによる同時実行制御では並列数に対し性能がスケールしにくいという問題がある．

そうした背景から，他のスレッドの干渉を受けずアトミックに構造の更新を行うロックフリーアルゴリズムに基づく索引が注目されている．一般にロックフリーアルゴリズムは CPU 命令の一つである CAS（compare-and-swap）命令によって実現される．CAS 命令はあるメモリ位置の内容と指定された値を比較し，等しければそのメモリ位置に別の指定された値を格納する．これらの比較と置換は同時に行われ，他の操作は割り込まない．この CAS 命令において，変更を行おうとしているメモリ位置の値と以前そのメモリ位置から取得した値を比較することで他のスレッドから変更があったかどうかを確認できる．変更がなければ同時に新しい値に置換され，変更があれば CAS は失敗するため，ロックフリーな更新を実現できる．同一のレコードを変更しようと複数のスレッドで CAS が同時に実行されるとき，唯一のスレッドのみが CAS を成功させ，他の CAS は失敗する．

一方で，Bw 木 [1] や Bz 木 [2] などの B+ 木をベースとした既存のロックフリー索引に関して，統一的な環境・実装での比較実験は不足している．そこで，本研究ではロックフリー索引の性能検証を目的とし，基礎的なベンチマークの作成及び既存索引の再現実装を行い，統一的な環境下での比較によってその性能特徴を検証する．

2 関連研究

2.1 Bw 木

Bw 木は Levandoski ら [1] によって提案された．Bw 木は差分ノードと CAS 命令を組み合わせてアトミックな更新を行うことによりロックフリーな操作を実現している．また，マッピングテーブルによるノード間のポインタの仮想化により，ストレージ内でのノード領域を柔軟に変更できる．

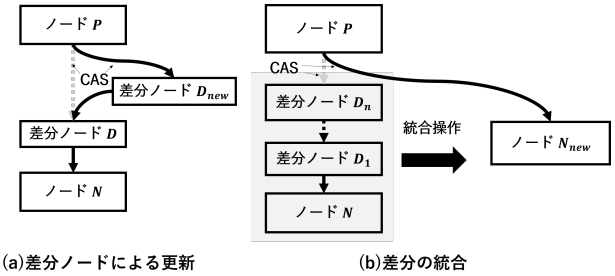

図 1 Bw 木における差分ノードを用いた更新

Bw 木では差分ノードによって木へのレコード書き込みおよび構造変更を表現することで不正な中間状態の生成と観測を回避している．差分ノードは図 1 の（a）のようにノードの親の部分に挿入する．差分ノードが既に存在する場合はそれに連なる形で挿入される．この挿入操作は CAS 命令によってアトミックに行われるため，構造に対してロックを取得する必要はない．差分ノードの連なりは探索の効率を悪化させるため，差分ノードが一定数を超えると図 1 の（b）のように統合操作が実行される．

ノードの分割やマージはノード内のレコードの数によってトリガする．構造変更操作はいくつかのステップに分割され，複数のスレッドが協調することによって行われる．構造変更操作の進度は差分ノードの挿入によって他のスレッドに知らされ，中間ステップを観測したスレッドはその操作を後追いする．最終的に各ステップで唯一のスレッドが構造変更操作を完了させ，他のスレッドの構造変更操作は失敗する．

Ziqi ら [3] は Levandoski ら [1] の提案した Bw 木に対して非ユニークキーのサポートと探索効率の改善のためのアルゴリズムを提案し，それらを実装した OSS である OpenBw 木を開発した．本研究ではこの OpenBw 木に対し性能調査を行う．

2.2 Bz 木

Bz 木は Arulraj ら [2] によって提案された不揮発性メモリを想定した索引構造である．B+ 木をベースにした構造であり，Persistence Multi-word CAS（PMwCAS）を使用してロックフリーな構造操作を実現している．不揮発性メモリ上の動作による更新の永続化や迅速なリカバリが利点として挙げられる一方で，揮発性メモリ上においても動作が可能である．

PMwCAS は，CAS 命令を単一ワードから複数ワードに拡張した命令である．Bz 木では各操作において PMwCAS を用いることでアトミックな更新を実現している．

名古屋大学大学院情報学研究科 Graduate School of Informatics, Nagoya University

葉ノードの操作では，PMwCAS を用いてノードのメタデータの更新によって書き込み領域の予約やレコードの可視状態の変更を行う．葉ノードでのレコード挿入は非ソート領域へ行われる．読み取り性能の低下を防ぐため，非ソートレコードの数がしきい値を超えるとノードの再編成が行われノード内のレコードがソートされる．

分割やマージなどの構造変更操作では，ノードを凍結状態にすることにより進行途中の更新操作の完了を防いで一貫性を保つ．凍結状態のノードを観測したスレッドはノードの構造変更操作を後追いする．最終的に，最も早く操作を終えたスレッドが PMwCAS によってノードをスワップし構造変更操作を完了させる．

Bw 木と異なりノード間のポインタをマッピングテーブルを介さずそのまま保持するため，余分なポインタ参照がなく高速にノード間を探索できる．Arulraj ら [2] は様々なワークロードにおいて Bw 木より高いパフォーマンスを発揮したことを報告している．

3 ベンチマークの設計

本章では作成する基礎ベンチマークの概要と Bw 木及び Bz 木の構造から予測される性能特性について述べる．

3.1 ベンチマークの概要

本ベンチマークはマルチスレッドをサポートし，ユーザの指定したワークロード下での索引のスループット及びパーセンタイルの遅延を測定する．データセットには 8 バイト整数を想定しているが，可変長型のサポートも予定している．

ベンチマークがサポートする可変なパラメータとして以下を想定している．

■木の初期サイズ　ベンチマーク実行前の初期化時に行う挿入操作の回数を設定する．read 操作のみのワークロードの実行や木の高さが各操作に与える影響を計測する際に有用である．

■キーの最大数及び偏り　式 (1) の分布に従う Zipf の法則に基づき，パラメータを変更することでキーの偏りを調整する．式 (1) において N はキーの最大数，k はキーの出現頻度の順位，s は偏りを調整するパラメータである．s を増加させるとキーの偏りは大きくなる．

$$f(k;s,N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s} \tag{1}$$

■各操作の割合　本ベンチマークでは索引への操作として read，scan，upsert（write），insert（write if not exist），update（write if exist），delete（write if exist）の六つを想定する．これら六つの操作の割合をワークロードとして指定し，ベンチマークとして実行する．索引の対応していない操作は割合として 0 を指定する．

3.2 各索引の予測される性能特性

性能検証の対象となる Bw 木と Bz 木を比較し，各索引の予測される性能特性について述べる．

■Bw 木　Bw 木は基本的には操作を差分ノードの追加で済ませるため各操作の遅延は小さく抑えられると推測される．

Bw 木は差分ノードの連なりにより探索性能を悪化させるケースが存在する．read 操作はリーフノードに接続する差分ノードをたどる必要がある．そのため，特に write 操作と read 操作を行うようなワークロードでは read 操作やレコードの存在確認が必要となる update 操作の遅延が大きくなると考えられる．

Bw 木ではスレッドが処理途中の構造変更操作を観測すると，現在の処理を保留して構造変更操作を後追い実行する．そのため，キーの偏りが大きいワークロードでは多くのスレッドが同一ノードの構造変更操作を後追いする可能性があり，全体的な性能が低下すると考えられる．

■Bz 木　葉ノードの非ソート領域が大きくなると read 操作時に線形探索が必要な領域が増加するため read 操作の性能が悪くなる．これは Bw 木の差分ノードの連なりと対応する．

Bz 木は PMwCAS を主体として各操作を行うが，その命令コストは CAS と比べると大きい．そのため，CAS を用いた Bw 木より平均的な遅延は大きくなると推測される．特にキーの偏りが大きく同一ノードに複数スレッドの write 操作が行われるようなワークロードでは PMwCAS の失敗率は高くなり，遅延は大きくなると考えられる．

Bz 木ではスレッドが凍結状態のノードを観測すると現在の処理を保留して構造変更操作を後追い実行するため，Bw 木と同じような問題が存在する．Bz 木での構造変更操作は段階的に分割されておらず後追い実行のコストが Bz 木と比べると高価であるため，性能に与える影響はより大きいと考えられる．

4 おわりに

本稿ではロックフリー索引の性能検証に着目した．関連研究としてロックフリー索引である Bw 木と Bz 木の概要を述べ，それぞれの特徴を挙げた．また，作成するベンチマークの概要とサポートするパラメータについて述べ，関連研究で挙げた Bw 木と Bz 木について予測される性能特性を考察した．

今後はこれらの方針にしたがってベンチマークを開発し，性能調査を行うことでロックフリー索引の性能特徴を検証する．

謝辞

本研究は JSPS 科研費（16H01722, 20K19804）の助成, 及び国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP16007）から得られた結果による．

参考文献

[1] J. Levandoski, D. B. Lomet, and S. Sengupta, “The Bw-tree: A B-tree for new hardware platforms,” in *Proc. ICDE*, pp. 302–313, 2013.

[2] J. Arulraj, J. Levandoski, U. F. Minhas, and P.-A. Larson, “BzTree: A high-performance latch-free range index for non-volatile memory,” *PVLDB*, vol. 11, no. 5, pp. 553–565, 2018.

[3] Z. Wang, A. Pavlo, H. Lim, V. Leis, H. Zhang, M. Kaminsky, and D. G. Andersen, “Building a Bw-tree takes more than just buzz words,” in *Proc. SIGMOD*, pp. 473–488, 2018.