2N-03

Bw木およびBz木における範囲走査性能の評価

平野 匠真† 杉浦 健人†† 石川 佳治†† 陸 可鏡††

† 名古屋大学情報学部コンピュータ科学科 †† 名古屋大学大学院情報学研究科

1 はじめに

コンピュータ技術が発展し，膨大な量のデータが管理されている．扱うデータは蓄積され続け，データの処理要求はより高度化される．そのため，より効率的にデータを扱える索引構造が必要とされている．

B^+ 木をロックフリー化させた索引構造として Bw 木 [1] および Bz 木 [2] が提案されている．Bw 木は全ての兄弟ノードが単方向リンクを持つ一方，Bz 木は親ノードから子ノードへの単方向リンクしか持たない．そのため，範囲走査において Bw 木は兄弟リンクをたどり葉ノードを読んでいくが，Bz 木は次の葉ノードを根ノードから読み直す必要がある．そのような動作にも関わらず，Bz 木での元論文では Bw 木よりも高い範囲走査性能が得られたと示されている．しかし，元論文では 10 レコード程度の短い範囲走査しか行われていないため，検証が不十分であると考えられる．そこで，本稿では兄弟リンクを持つ Bw 木と持たない Bz 木における範囲走査をより網羅的に検証し，それぞれの索引構造が範囲走査に与える影響を再評価する．

本稿では，まず Bw 木および Bz 木について，両木の木構造について述べる．次に，範囲走査において両木の動作の共通点および相違点を述べる．最後に両木の範囲走査の性能の比較および評価方法について述べる．

2 Bw木およびBz木の概要

Bw 木および Bz 木は B^+ 木を拡張したロックフリー索引構造である．挿入・削除を始めとするレコード操作や分割などの木の構造変更操作において compare-and-swap（CAS）命令を用いることにより，ロックフリーでの更新を実現している．以下では，それぞれの木について概要を説明する．

2.1 Bw木

Bw 木は，直感的にはロックフリーの単方向連結リストと B^+ 木を組み合わせた索引構造であり，図 1 のような構造を持つ．更新内容が記述された差分レコード（delta record）をノードの前に連結リストとして挿入し，索引に対する全ての更新を表す．差分レコードのリストが一定以上の長さを持つときノードへの統合操作が行われ，統合後は B^+ 木のノードと同様のものが生成される．また，単一の CAS 命令を用いてノード間のポインタをインストールできるようメモリ内のデータ構造を設計

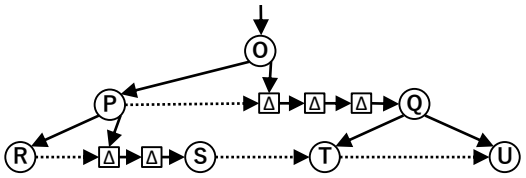

図 1 Bw 木の簡単な木構造

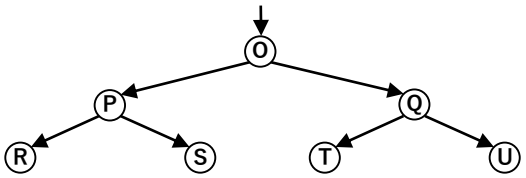

図 2 Bz 木の簡単な木構造

することにより，木のロックフリー化を実現している [3].

B^+ 木と同様に，Bw 木は索引部およびデータ部によって構成される．索引部のノード（中間ノード）にはキーでソートされた分割キーと子ノードへのポインタの組を格納し，木の下方向に対しての検索を補助する．最下層のデータ部のノード（葉ノード）はキーと対応する値の組を格納している．なお，前述のとおり各ノードがそれぞれ差分レコードのリストを持つ点に注意する．また，全てのノードはそのノード内に格納可能な最小キーおよび最大キーを情報として持つ．各ノードは右の兄弟ノードへの単方向リンクも持つため，自身の最大キーよりも大きなキーが与えられた際は右の兄弟ノードを続けて検索できる．例えば，図 1 においてノード S に目的のキーが含まれていない場合，右の兄弟ノードであるノード T を続けて検索できる．

2.2 Bz木

Bz 木は，直感的には B^+ 木の葉ノード内部にロックフリーの固定長配列を持つ索引構造であり，図 2 のような構造を持つ．Bz 木は単一の CAS 命令ではなく，メモリ上の複数ワードを対象とする multi-word compare-and-swap（MwCAS）命令を用いて木のロックフリー化を実現している．なお，Bz 木は元々永続メモリ向けの索引として提案されたが，本稿では永続化については議論せずロックフリー索引としてのみ扱う．

Bz 木は Bw 木同様に索引部，データ部によって構成される．Bz 木のノード内のデータについて説明する．各ノードにはヘッダ，メタデータ，レコードが格納されている．ヘッダはノードの先頭に配置され，ノード自身の情報が格納されている．格納されている情報としてノード自身の大きさ，レコード数，レ

Evaluation of Range Scanning Performance in Bw-tree and BzTree
Shoma Hirano†, Kento Sugiura††, Yoshiharu Ishikawa††, and Kejing Lu††
†Department of Computer Science, School of Informatics, Nagoya University
††Graduate School of Informatics, Nagoya University

コードブロックが占有しているスペースのバイト数などがある．メタデータは 1 つのレコードに対して 1 つ存在し，対応するレコードの情報を格納する．レコードはキーおよびレコードへのポインタもしくは実際のペイロードを格納する．また，Bz 木はノード内で差分レコード用の領域を持ち，一部の差分レコードを扱っている．書き込みや削除などの葉ノードへのレコード操作はノード内にある差分レコード用の領域を用いて処理し，統合操作および木の構造変更に関しては MwCAS を用いて処理する．また Bw 木と異なる点として，図 2 に示したように Bz 木のノードは兄弟リンクを持たない．

3 Bw 木および Bz 木における範囲走査

Bw 木および Bz 木は範囲走査をサポートしている．以下では，範囲走査における両木の共通する部分および異なる部分を述べる．

3.1 共通する部分

Bw 木および Bz 木の範囲走査において共通する部分として，範囲走査全体の流れと差分レコードの統合操作が挙げられる．

3.1.1 範囲走査

範囲走査とはユーザ側が始点キーと終点キーを指定することで，そのキー範囲内のノードを検索し，その全てのレコードを返す動作である．始点キー，終点キーは省略が可能であり，その場合には省略した一端がオープンエンドとなる．範囲走査は終点キーにたどり着くまで葉ノードを 1 つずつ走査することで行われる．Bw 木および Bz 木内部のノードはキーによってソートされているため，ある葉ノードの走査が終了したとき次に走査すべき葉ノードは唯一となり，重複なくレコードを走査できる．

始点キーが指定された場合，そのキーを用いて走査対象の葉ノードを検索する．指定されなければ索引内で最も左端の葉ノードを選択する．葉ノードに到達した後は，対象のノードが持つレコードを以下の統合操作によって複製し，複製したレコードを走査するためのイテレータをユーザへ返す．

3.1.2 統合

走査対象のノードに差分レコードが存在している場合，走査する前にノード内のレコードと統合，つまり全レコードをソートする必要がある．

Bw 木の場合は 2.1 章で述べたように，更新内容を記述した差分レコードがベースとなる葉ノードに前置されることによりチェーンが形成されている．そのため，対象ノードの走査を試みた際に差分レコードを発見した場合，その差分レコードで形成された差分チェーンおよび対象ノード内のレコードを統合する．統合操作により走査対象のレコードを複製したノードが作成されるため，その最初のレコードを指すイテレータを返す．

Bz 木の場合は 2.2 章で述べたように，葉ノードへのレコード操作を記述した差分レコードはノード内の差分レコード用領域に格納されている．走査の際に差分レコードが存在していれば，差分レコードとソート済みレコードとを統合したノードの複製を作成する．Bz 木の統合では，差分レコード用領域をシーケンシャルにスキャンできる．そのため，統合処理で単方向リンクをたどる必要がなく，Bw 木に比べキャッシュヒット率を高めている．

3.2 異なる部分

Bw 木および Bz 木の範囲走査において異なる点として，兄弟リンクの利用の有無がある．

Bw 木は兄弟リンクを持つ索引構造である．範囲走査において葉ノードの端まで検索したが，そのノードの最大キーが終点キーより小さい場合，次の葉ノードに向かう必要がある．その際に，葉ノード間のポインタを利用する．ポインタを利用することにより，根ノードから読み直すことなく右の兄弟ノードへ遷移し，効率的に範囲走査を継続できる．

一方で，Bz 木は兄弟リンクを持たない索引構造である．つまり，次の葉ノードに向かう動作をする際には根ノードから読み直す必要があり，中間ノードの追加読み取りが発生してしまう．

以上のとおり，Bw 木では兄弟リンクを用いた葉ノード間の遷移が可能であるのに対し，Bz 木では常に根ノードからたどりなおさなければならない．この点についてのみ着目した場合 Bz 木よりも Bw 木の方が効率的と言えるが，Lomet らは Bz 木の方が範囲走査の性能が高いと報告している [2]．これは前述したとおり Bz 木の統合操作が Bw 木よりも効率的であるためだと考えられるが，その影響の度合いは明らかとなっていない．

4 評価分析

まず通常の Bw 木および Bz 木を構築し，両木の基本的な構造や動作を確認する．正常な動作が確認できた後，各索引における範囲操作の性能を調査する．特に元論文では不足していたレコードの範囲について，網羅的に調査する．具体的には，数レコード程度の低範囲，葉ノードを跨ぐ中範囲，多くのレコードを走査する広範囲の 3 つの範囲における性能を評価する．

5 おわりに

本稿では，ロックフリー索引である Bw 木および Bz 木の概要，範囲走査における動作を述べ，範囲走査の性能比較を提案した．今後は，4 章で述べたように各索引の範囲走査における性能比較のために実験していく予定である．

謝辞

本研究は JSPS 科研費（16H01722，20K19804, 21H03555）の助成，およ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）の結果得られたものである．

参考文献

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

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

[3] A. Petrov, 詳説データベース: ストレージエンジンと分散データシステムの仕組み. オライリー・ジャパン, 2021.