ロックフリー索引のトライ木化による高速化に関する研究

井戸 佑[1,a)]　杉浦 健人[1,b)]　石川 佳治[1,c)]　陸 可鏡[1,d)]

概要：代表的な索引構造である B+木は様々なデータを格納可能な汎用性の高い索引である．一方で，扱うキーに制限を加え，索引構造を最適化させることで性能を向上させる研究も行われてきた．本研究では，著者らの研究室で開発しているロックフリー B+木に対し同様の拡張および性能改善を行う．具体的には，扱うキーをバイナリ比較可能なものに制限することでトライ木の構造を適用し，空間利用効率の向上およびそれに伴う検索性能の向上を図る．

1. はじめに

2. 関連研究

2.1 B^c 木

2.2 Masstree

3. B^c forest の構造

4. B^c forest のノード操作

本節では，B^c forest のノード操作について述べる．B^c forest は以下の読み取りおよび書き込み操作をサポートする．

4.1 書き込み

書き込み（Write）はキーとペイロードを差分レコード領域に挿入する操作である．まず，キーの先頭 0〜7 byte が属するの葉ノードを根ノードから二分探索により特定する（Layer0）．次に，葉ノード内のイミュータブル領域を二分探索し，接頭辞 8 byte が共通するキーについて下層へのポインタの有無を確認する．ポインタが無い場合，本葉ノードを挿入先の葉ノードとして特定する．ポインタがある場合，下層の根ノードへ移動し，キーの 8〜15 byte が属する葉ノードを二分探索により特定する（Layer1）．本操作を下層ポインタがなくなるまで繰り返し，葉ノードを特定する．挿入先の葉ノードに到達後，その葉ノードの差分レコード領域に値を挿入する．

書き込み操作は差分レコード領域の予約とレコード挿入および可視化の 2 ステップで行われる．図 1 に B^c 木における差分レコードの挿入を示す．ノードフッタにはミュータブル領域の状態を表すステータスワードを用意し，この中のレコード数と使用済みブロックサイズを加算することで差分レコード用の領域を予約する．また，B^c 木ではノードの生成時にミュータブル領域をゼロ埋めしている．B^c 木では，メタデータがゼロ埋めされている場合を処理途中として表すため，差分レコード用の領域を予約した時点ではレコードが可視化されていないことを認識できる．確保した領域へ差分レコードを書き込み，対応するレコードメタデータの値を更新することでレコードを可視化し，挿入処理を完了する．

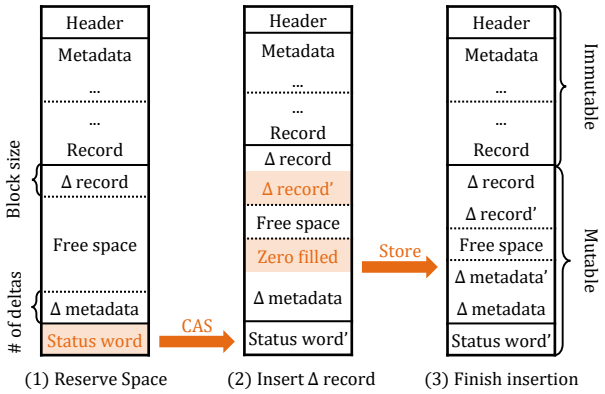

図 1　B^c 木における差分レコードの挿入

以上の書き込み操作によって差分レコードを挿入していくが，挿入後の差分レコードの総数またはノード容量のしきい値を越える場合，差分レコードの統合操作または構造変更操作が行われる．しきい値の確認はステータスワードの更新時に行われ，しきい値を越えた場合はレコードの書き込み後に統合操作，構造変更操作いずれかの操作を行うか判定する．

[1] 名古屋大学大学院情報学研究科
Graduate School of Informatics, Nagoya University
[a)] ido@db.is.i.nagoya-u.ac.jp
[b)] sugiura@i.nagoya-u.ac.jp
[c)] ishikawa@i.nagoya-u.ac.jp
[d)] lu@db.is.i.nagoya-u.ac.jp

書き込み処理はロックフリーに動作する．書き込み同士の競合はステータスワードで解決され，差分レコード用領域の予約，つまり差分レコードの書き込み順は CAS 命令の成功順によって順序付けられる．また，後述する構造変更操作との競合によって待ち時間が発生しうる．

4.2 読み取り

読み取り（Read）は与えられた対象キーのペイロードを返す操作である．まず，キーの先頭 0〜7 byte が属するの葉ノードを根ノードから二分探索により特定する（Layer0）．葉ノードに到達した後は，葉ノード内の差分レコードの線形探索とイミュータブルレコードの二分探索の 2 ステップで読み取り操作を行う．最新の値は差分レコード領域に書き込まれるため，まずミュータブル領域を線形探索し，一致するキーがあるか確認する．このとき，差分レコードの数はステータスワードから読み取り，差分レコードが可視化されていなければスキップする．

差分レコード中に対象のレコードが存在しなければ，ノードヘッダからイミュータブルレコードの数を読み取り，二分探索によって対象キーの有無を確認する．接頭辞 8 byte が共通するキーについて posting list が存在する場合には，posting list 内を線形探索し，接尾辞が一致するキーがあるか確認する．接頭辞 8 byte が共通するキーについて下層へのポインタが存在する場合には，下層の根ノードへ移動する（Layer0 → Layer1）．Layer1 移動後，キーの 8〜15 byte が属する葉ノードを二分探索により特定し，同様の 2 ステップを行う（Layer1）．この操作を下層へのポインタがなくなるまで繰り返し，読み取り操作を行う．先頭ノードが統合操作の途中である場合には，差分レコードを読み終わった時点で物理ポインタをたどり古いノードへ移動し，同様の 2 ステップを古いノード上で行う．

読み取り処理は wait-free に動作する．上述したとおり読み取り命令は一切ノードの状態を変更せず，読み取った状態に応じて適切な手続きを選択する．そのため，読み取り命令においてリトライなどは発生せず，有限時間内で必ず処理が終了する．

5. B^c forest の構造変更操作

5.1 統合
5.2 分割
5.3 新層作成

6. おわりに

謝辞 本研究は JSPS 科研費 JP20K19804，JP21H03555，JP22H03594，JP22H03903 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