高速OSIディレクトリ用DBMSにおけるバッファリング方式

3N−4

横田 英俊　西山 智　小花 貞夫　浅見 徹
国際電信電話株式会社 研究所

1. はじめに

筆者らは、ユニバーサルパーソナル通信(UPT)等の高度な通信サービスのネームサーバとして適用可能な高速なOSIディレクトリを実現するために、ディレクトリ情報ベース(DIB)のデータモデルと操作を直接提供する専用DBMSの開発を行なっている[1]。一般にDBMSではバッファリング方式がDBMSの処理性能に影響する。本DBMSでは、DIBに対するアクセスの性質を考慮した2段階の優先度付きのLRUアルゴリズムをバッファリング方式として用いている。性能評価の結果、本DBMSでは大量のエントリが格納された場合にバッファのヒット率が低下し、それが応答速度に大きく影響を与えることが分かった[2]。そこで本DBMSのバッファ効率を向上させるために各ページのアクセス頻度を考慮して重み付けを行なうLRUに基づくバッファリング方式について検討したので報告する。

2. 高速OSIディレクトリ用DBMSのデータ格納方式

本DBMSのデータ格納方式を表1に示す。

表1: 高速OSIディレクトリのデータ格納方式

エントリ格納	格納方法	直接クラスタリング(エントリ単位)
	格納の効率化	フラグメンテーション
	アクセス手法	B-木
名前管理	実現方式	識別名インデックス
	アクセス手法	木構造

DIBについては各オブジェクトの名前管理のために、論理的にディレクトリ情報木(DIT)と呼ばれる木構造で表現される。本DBMSでは、DITに対して識別名から検索対象のエントリを得る処理の高速化を図るために、識別名に対してインデックスを生成する。識別名インデックスはDITの木構造に対応した木構造アクセス手法により格納する。この木構造アクセス手法を図1に示す。各エントリに対応するノードは、無限長の論理ブロックにより実現される。論理ブロックはバッファリングの単位であるページにより構成される。論理ブロックに格納されるレコードはキーとデータ内容からなる。キーとしては下位エントリの相対識別名を用いる。データ内容には下位エントリのノードへのポインタが格納される。また、エントリに関してはB-木を用いて格納される。

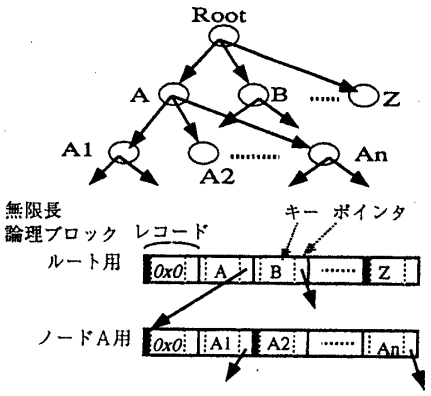

図1: 木構造のアクセス手法

3. 従来のバッファリング方式とその問題点

3.1 従来のバッファリング方式の概要

OSIディレクトリでは各エントリに対してランダムにアクセスされることが想定される。したがって本DBMSにおいてもページの大半を占めるエントリ用B-木の葉ノードのページについては殆んどランダムにアクセスされると考えられる。LRUのようなアクセスの局所性を用いたアルゴリズムでは、ランダムアクセスされる葉ノードのページに対しては効率的なバッファリングが行なえない。このため本DBMSでは識別名インデックスの木およびエントリ用B-木の各ノードのうちランダムにアクセスされるノードと、比較的局所性のあるノードを分けてバッファリングを行なう、2段階の優先度付きLRUに基づくバッファリング方式を用いた。この方式では、全バッファ領域のうち優先度が「高」のページのためのバッファと「低」のページのためのバッファに分ける(以下、優先度が「高」のページのためのバッファがバッファ領域全体に占める割合を優先バッファ率と呼ぶ)。アクセス頻度が小さいB-木の葉ノードについては優先度を「低」、その他のノードおよびDITの全てのノードについては優先度を「高」とする。優先度が「低」のページが要求された時には優先度「低」のバッファのみでLRUを行ない、優先度が「高」のページが要求された時には両方のバッファでLRUを行なう。実装においてはCLOCKアルゴリズムを用いた。

3.2 従来のバッファリング方式の問題点

このバッファリング方式の問題点としては次のようなことが挙げられる。すなわち優先度が「高」のページにおいてもアクセス頻度は異なるが、CLOCKアルゴリズムでは優先度が「高」のページのLRUを行なう際、そ

"Buffering Method of DBMS for High Performance OSI Directory" Hidetoshi YOKOTA, Satoshi NISHIYAMA, Sadao OBANA, and Tohru ASAMI
KDD R & D Laboratories

れらのアクセス頻度の差は考慮されず全て同等に扱われ、それがヒット率の低下を招く。この問題点を解決するために、より細かな優先制御を行なえるバッファリング方式を提案し、以下にその概要を述べる。

4. アクセス頻度を重みとした優先度付き LRU に基づくバッファリング方式の提案

B-木および木構造では木の上のノードほどアクセス頻度が高いことから、その平均アクセス回数を重みとして利用する。まず B-木について考える。Bを 1 ノード当たりの分岐数、Nを木の深さとし、$w_{i,N}$を葉ノードのアクセス頻度を 1 とした時のレベル i のノードの相対的なアクセス頻度とすると、各エントリにランダムにアクセスされると仮定した場合の $w_{i,N}$は $w_{i,N} = B^{N-i}$で与えられるので、これを B-木のページの重みの値として用いる。また B-木の 1 ノード当たりの分岐数 Bは 1 ノードが格納できるデータの最大数にノードの平均利用率 (69%[3]) をかけたものを用いる。木の深さ Nについてはエントリ格納時に計算できるのでそれを用いる。DIT に関しては OSI ディレクトリを使用するアプリケーションにより木の高さ N、総エントリ数などの特性が定まるので、それらの値からノード当たりの分岐数 Bを求めることができる。今回このアルゴリズムを実装するために GCLOCK アルゴリズムを用い、ページが初めて参照された時の参照カウンタの初期値として $w_{i,N}$を与えることとした。

5. 評価

表 2に示した条件のもとで提案したバッファリング方式を用いてランダムに選んだ葉エントリに対する Read 操作を行なった時のバッファのヒット率を求めた。また比較のため優先度のない LRU によるバッファリング方式と従来のバッファリング方式 (優先バッファ率は 50%) を用いた時のヒット率を求め合わせて図 2に示す。本バッファリング方式に関しては、DIT における木の深さ Nとノードの分岐数 Bは評価用 DIB では $N = 4$、$B = 25$ であるので、$w_{1,4} = 25^3$、$w_{2,4} = 25^3$、$w_{3,4} = 25^2$、$w_{4,4} = 25^1$ とした。

表 2: 測定条件

測定環境	Sun SPARCserver 690MP (SunOS4.1.3)
測定に用いた DIT	木の深さ 4 段 ($N = 4$)、2 段目以降が 25 本ずつの枝を持ち各段で均等に分岐する ($B = 25$)。エントリ総数=16276 件。

6. 考察

評価結果より優先度のない LRU によるバッファリング方式よりも従来のバッファリング方式の方がヒット率が高いが、本バッファリング方式ではさらに高いヒット率が得られた。これにより本バッファリング方式の有効性を確認できた。また 1000 ページ付近においては従来のバッファリング方式と本バッファリング方式との間の差が小さくなっているが、これは評価実験に用いた 1

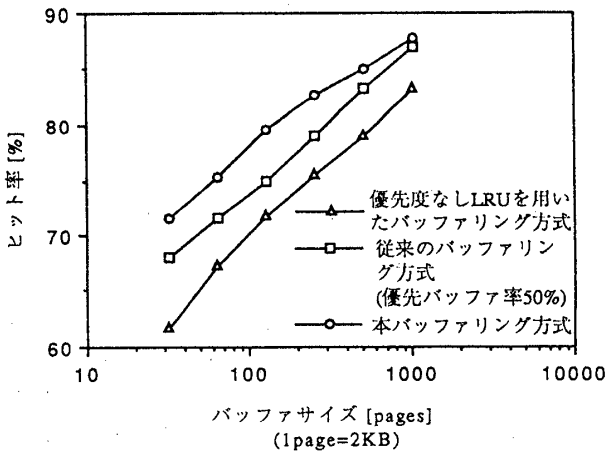

図 2: バッファサイズとヒット率の関係

万 6000 件のエントリに対して優先度が「高」のページの総数は約 1000 ページ程度であり、優先度が「高」のページが全て入るだけの十分なバッファサイズとなっているためである。

評価結果より葉ノードのページ以外の全てのページが入るだけのバッファを取ることができれば従来のバッファリング方式は本バッファリング方式に比べて同程度のヒット率を示す。1 万件程度のエントリについては上述のように約 1000 ページ (約 2MB) 程度のバッファを取ることができれば十分なバッファサイズと言える。しかしエントリの件数が 10 万件、100 万件となった時に、従来のバッファリング方式が本バッファリング方式と同程度の高いヒット率を得るためには、バッファサイズが 10 万件の場合には約 8 千ページ (約 16MB)、100 万件の場合には約 8 万ページ (約 160MB) 程度必要となる。バッファサイズには物理的な制約があることから、件数が増えるにつれて十分なバッファを確保することが不可能となる。したがって大量のデータを格納する場合に本バッファリング方式が特に有効な手法と言える。

7. おわりに

高速な OSI ディレクトリ情報ベースを構築するためにそのバッファリング方式について検討し、すでに実装している従来のバッファリング方式の問題点を解決するためにアクセス頻度を重みとした優先度付き LRU に基づいたバッファリング方式を提案し評価を行なった。その結果、大量データを格納する DIB では提案したバッファリング方式が特に有効であることを示した。

最後に日頃御指導頂く KDD 研究所 小野所長、浦野次長に感謝します。

文献

[1] 西山 他、“高速 OSI ディレクトリ用 DBMS の設計,” 情処第 45 回全大 3R-9, 1992.
[2] 西山 他、“高速 OSI ディレクトリ用 DBMS の評価,” 情処第 46 回全大 3N-3, 1993.
[3] 上林弥彦 “データベース”、昭晃堂 (昭 61 年)