2R−3

LRU-S: 複合オブジェクト間の参照情報を用いるバッファ管理手法

沈 虹* 陳 漢 雄† 山口和紀‡ 北川博之§ 大保信夫§ 藤 原 譲§

*筑波大学理工学研究科　‡東京大学教養学部計算機図形教室
†筑波大学工学研究科　§筑波大学電子情報工学系

1 動機と背景

OS、関係データベースのバッファ管理では伝統的なLRUやMRUなどが有効な手法としてよく用いられてきた。

一方、CADなどの応用分野では、オブジェクト指向データベースシステムの導入が増えつつある。これらのシステムでは、複合オブジェクトのトラバースやオブジェクト間のナビゲーションがよく見られるアクセスのパターンといえる。このような環境に於いては、バッファ管理においてオブジェクト間の参照関係を考慮することが有効であると考えられる。本論文では、オブジェクト指向バッファマネジャーの中で、複合オブジェクトの参照関係を有効に利用する新しい構造LRU法（LRU-S）を提案し、その性質を示す。

2 オブジェクト間の参照関係

定義（複合オブジェクトスキーマ） 複合オブジェクトスキーマは

$$O = (OID, A_1, ..., A_n, O_1, ..., O_m)$$

で定義される。ここで、OIDはオブジェクト識別子、O_iは要素オブジェクスキーマ、A_iは単純属性を表す。要素オブジェクトがまた複合オブジェクトであることも有り得る。

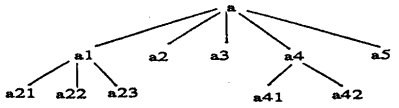

図 1: 複合オブジェクト例

$o_i \prec o$はoがo_iを直接／間接的に参照することを表す。$o_j \prec o_i$及び$o_i \prec o$が成り立てば$o_j \prec o$も成り立つ。

DAG：上で述べたように、オブジェクト間の参照関係はDAGで表される。ここで、オブジェクト、あるいはページはDAG中のノードと対応する。以後NでDAGの節点集合（ノードセット）を表す。$a \prec b$はノードbが直接／間接的にノードaを参照することを示す。

$\preceq$は$\prec$の拡張で、反射性を持つ。従って、すべての$a \in N$に対して、$a \preceq a$が成り立つ.

ノードリスト：バッファ管理では、アクセスノードとバッファ状態との関係は極めて重要である。この二つの概念は以下のように表す。

一つのアクセス列は一つのノードリスト$r = r_1 \ldots r_m$で構成される。（ここで、$r_i \in N$。）

バッファ状態はリスト$b = b_1 b_2 \ldots b_s$ $(\forall b_i \in N)$で表示され、リスト末尾要素b_sは最近使われたノードである。

リストsに対して、s_iと$|s|$はそれぞれsの第i番目要素とsの長さを表す。バッファスタックの最上位の要素はb_s、最下位の要素はb_1である。$hd(b)$はバッファbの最下位のノードを示す。即ち、$hd(b) = b_1$。$tl(b)$はバッファbの最下位のノード以外のノードで構成されたリストである。即ち、$tl(b) = b_2 \ldots b_m$

参照関係：

条件fが与えられた時、$s|_f$はリストsから条件fを満たす各ノードs_{l_i}で構成されたリストである。

$s \downarrow_f \triangleq s_{l_1} s_{l_2} \ldots s_{l_t}$

また、$s \uparrow_f \triangleq s \downarrow_{\neg f}$は$s$の条件fを満たさないノードで構成されたリストである。

例えば、$(b \downarrow_{\lambda x. x \prec n})$はリスト$b$から、ノード$n$に直接／間接的に参照されたノードを抽出して生成した新たなリストである。

3 LRU-Sとは

LRU-Sの基本的な考え方は、複合オブジェクトと要素オブジェクト間の参照関係に基づき、関連するオブジェクトをなるべく一緒にバッファに保存するということである。

アルゴリズム：使用可能なバッファのサイズをs_{max}、バッファの状態を$b = b_1 \ldots b_s$、次のアクセスノードをnとすれば、LRU−Sにより新たなバッファ状態$(b|_n)$は次のように定義される。

ケース1：$n \in b$ならば、

$$b|_n = (b \uparrow_{\lambda x. x \preceq n})(b \downarrow_{\lambda x. x \prec n})n$$

即ち、アクセスノードnはバッファの中にあるノードb_kである時、nをバッファの最上位に移す。そしてバッファの中で、nに参照されているすべてのノードもnのすぐ下のバッファに移す。

ケース2：$n \notin b$、かつ$|b| < s_{max}$ならば

$$b|_n = (b \uparrow_{\lambda x. x \prec n})(b \downarrow_{\lambda x. x \prec n})n$$

即ち、nがバッファの中に存在せず、かつバッファが満たされていない場合、nをバッファの最上位に追加する。そしてバッファの中で、nに参照されているすべてのノードもnのすぐ下に移す。

LRU-S: A Buffer Mangement Method Using Object Reference Information

Hong Shen * Hanxiong Chen † Kazunori Yamaguchi ‡ Hiroyuki Kitakawa §Nobuo Ohbo § Yuzuru Fujiwara §

*Master Course in Science and Engineering, University of Tsukuba　‡Department of Graphic and Computer Science, The University of Tokyo
†Doctoral Program in Engineering,University of Tsukuba　§Institute Of Information Science and Electronics,University of Tsukuba

ケース 3 : $n \notin b$、かつ $|b| = s_{max}$ ならば、

$$b|_n = \begin{cases} tl(b)n, & if \ (b \uparrow_{\lambda x.x \prec n}) = \varepsilon \\ (b' \downarrow_{cond1})(b' \uparrow_{cond2})(b \downarrow_{\lambda x.x \prec n})n, & otherwise \end{cases}$$

ここで、

$b' = (b \uparrow_{\lambda x.x \prec n}), n' = hd(b')$

$cond1 = \lambda x.(x \prec n' \wedge \forall b_i(\neg(b_i \prec n') -> \neg(x \prec b_i)))$

$cond2 = cond1 \vee (x = n')$

即ち、アクセスノード n がバッファの中に存在せず、かつバッファが一杯である場合、n から参照されているノードが存在する時、スタック最下位のノードをバッファから取り出し (swap out)、n をバッファの最上位に追加する。そしてバッファの中で、n から参照されているすべてのノードも n のすぐ下に移し、最後に n に参照されていないが、取り出したノード n' から参照されているすべてのノードをバッファの下部に移す。但し、共有ノードをなるべくバッファに置くため、n' から参照されていない他のバッファのノードから参照されているノードは優先順位を変えないままでバッファの中に保存する。

もし、バッファ中のすべてのノードが n から参照されている場合、スタック最下位のノードをバッファ中から取り出す。

4 シミュレーション

シミュレーション の結果は図 2、図 3、図 4 のように示す。[Chou, Falo91, Ng91] 及び Hot Set の理論と同じように、LRU-S に於いても、オブジェクトのサイズと問い合わせパターンにより、最適的なバッファサイズは変化する。

LRU-S は適当なバッファサイズの区間において、LRU、MRU より有効である。通常、ある特定のアクセスパターンの中に、繰り返し現れるノードの列を Ref_s とし、それらのノードから参照される可能性のあるノードの列を Ref_r とすれば、LRU-S のバッファサイズは [$|Ref_s|, |Ref_s| + |Ref_r|$] の間で、効率がいい。図 2 では [45, 75]、図 3 では [14, 23]、図 4 では [18, 36] においてこれが成り立つことがわかる。

明らかに、極端に小さいあるいは大きいバッファサイズに対して、全てのバッファ管理手法が同じように動作することはいうまでもない。実際、上で述べた区間以外では LRU-S の効率が LRU や MRU のとほぼ同じであることがわかる。また、不規則な参照列に対しても、LRU-S が LRU や MRU に劣らないことがわかった。

5 結び

本研究では関連オブジェクト間の参照関係を考慮したバッファ管理方法 LRU-S を提案した。これからの課題として、オブジェクト間の関係のセマンティクスをより正確に分析して、LRU-S を改良することがあげられる。

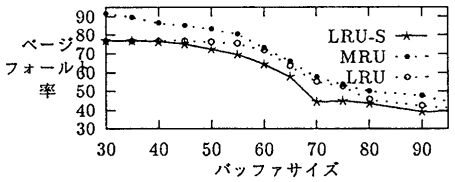

図 2: ケース 1 : $|Ref_s| \approx 45$、$|Ref_r| \approx 30$

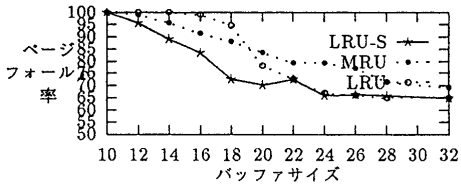

図 3: ケース 2 : $|Ref_s| \approx 14$、$|Ref_r| \approx 9$

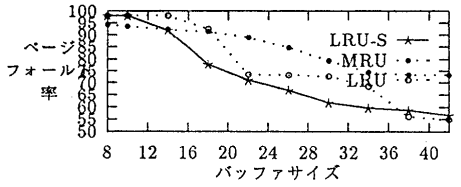

図 4: ケース 3 : $|Ref_s| \approx 18$、$|Ref_r| \approx 18$

参考文献

[Chou] H. T. Chou and D. J. DeWitt: An Evaluation of Buffer Management Strategies for Relational Database System. *"Readings in DATABASE SYSTEMS", edited by M. Stonebraker*, Morgan Kufman Publishers, INC.

[Ng91] R. Ng, C. Faloutsos and T.Sellis: Flexible Buffer Allocation based on Marginal Gains, *Proc. of ACM SIGMOD'91*, 1991.

[Falo91] C. Faloutsos, R. Ng and T. Sellis: Predictive Load Control for Flexible Buffer Allocation, *Proc. of the 17th Conf. on VLDB*, Barcelona, Sep., 1991.