4G−10

同時実行性を高めた動的ハッシュ*

平野泰宏, 三浦史光, 武田英昭†
NTT情報通信網研究所‡

1 まえがき

データ量 n の変化に柔軟に対応でき、検索時間が $O(n)$ となる索引方式である Extendible Hash, Linear Hash などの動的ハッシュ法が種々提案されている [1].

Extendible Hash はエントリの追加／更新によって溢れたバケットを必ず分割するため、溢れたバケットが分割されるとは限らない Linear Hash よりも安定した検索速度が得られる。しかし、Extendible Hash では、バケット分割の際に多くのディレクトリエントリを更新する必要があり、クリティカルセクションが長くなるため同時実行性が低下するという欠点があった.

本稿では、Extendible Hash を改良し、同時実行性を高めた新しい動的ハッシュ法を提案する.

2 Extendible Hash

Extendible Hash の構成例を図1 (1) に示す. Extendible Hash は、大域深さと、(2の大域深さ乗) 個のエントリを持つディレクトリと、局所深さを持ち該エントリからポイントされるバケットから構成される. 以下に特徴を示す.

(1) ハッシュ値の下位または上位から大域深さと等しい数のビットを用いてディレクトリエントリを参照し、検索・更新を行なう.
(2) データの挿入によって溢れるバケットを2つに分割する. (局所深さ) < (大域深さ) の場合、そのバケットを指していたエントリを更新し、局所深さに1を加える (図1 (2)). (局所深さ) = (大域深さ) の場合、ディレクトリのエントリ数を2倍にし、追加したエントリを更新し、局所深さと大域深さに1を加える (図1 (3)).
(3) 2の (大域深さ−局所深さ) 乗個のエントリが、同一のバケットを指す.

3 改良 Extendible Hash

3.1 Extendible Hash の問題点

Extendible Hash では、複数のエントリが同一のバケットを指す場合があるため、

(1) バケット分割の際に、2の (大域深さ−局所深さ−1) 乗個のエントリの更新を行なう必要がある
(2) ディレクトリ拡張の際に、2の大域深さ乗個のエントリと大域深さの更新を行なう必要がある

ことにより、データを挿入するプロセスの応答時間が長くなる要因となっていた. また、その間、他のプロセスは分割されたバケットやディレクトリを更新できないため、同時実行性が低下するという問題があった.

*A Dynamic Hashing Enhancing Concurrency
†Yasuhiro Hirano, Fumiaki Miura, Hideaki Takeda
‡NTT Network Information Systems Laboratories

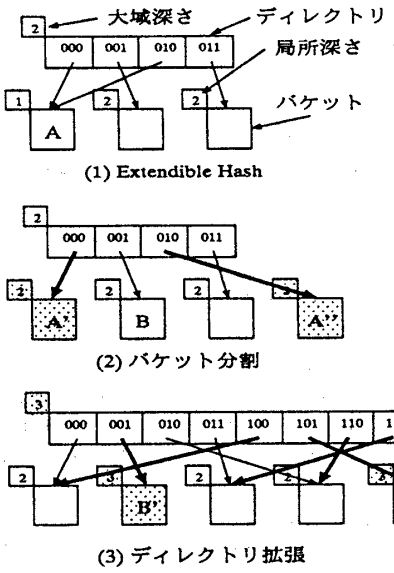

図 1: Extendible Hash

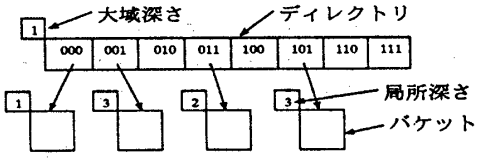

図 2: 改良 Extendible Hash の構成

3.2 構成

溢れたバケットを必ず分割するという Extendible Hash の特徴はそのままで、検索アルゴリズムを工夫することによって、バケット分割の際に更新すべきエントリ数を少なくした改良 Extendible Hash を提案する. 改良 Extendible Hash の構成を図2に示す.

改良 Extendible Hash は Extendible Hash と同様に、局所深さを持つバケット、ハッシュディレクトリ、大域深さで構成される. 改良 Extendible Hash の特徴は以下の通りである.

(1) ハッシュ値を下位ビットから使用する.
(2) ディレクトリエントリの一部が未使用になりうる. 同じバケットを指すエントリの数が Extendible Hash 以下になる.
(3) 大域深さの意味が Extendible Hash と異なる. 2の大域深さ乗個のディレクトリエントリは全て使用されているという意味である.

3.3 検索アルゴリズム

検索アルゴリズムを図3に、実施例を図4 (1) に示す. ハッシュ値の局所深さ以上のビットを使用するまで使用するビットを1ビットずつ増やしながらディレクトリ参照を繰り返すことが特徴である.

3.4 挿入アルゴリズム

挿入の実施例を図4 (2) に示す．3.3で述べた検索アルゴリズムにより1つのバケットを指すエントリは1つでよくなるため、バケット分割の際に更新するエントリ数はExtendible Hash 以下になる．

3.5 ディレクトリメンテナンス

検索の際のディレクトリエントリや局所深さの参照回数を削減するために、未使用になっていたエントリを更新して大域深さを大きくする操作をディレクトリメンテナンスと呼ぶ．

ディレクトリメンテナンスの実施例を図4 (3) に示す．ディレクトリメンテナンスは挿入とは非同期に行なってもよいし、Extendible Hash のようにディレクトリ拡張の度に行なってもよい．但し、ディレクトリメンテナンスを行なった場合、バケット分割の際に2の大域深さ乗以下のエントリが全て適切なバケットを指すように更新する必要がある．

4 Extendible Hash との比較

4.1 検索

改良 Extendible Hash では、大域深さより大きい局所深さを持つバケットを検索する場合、ディレクトリエントリや局所深さを参照する回数が (局所深さ) − (大域深さ) 回だけ、Extendible Hash よりも増える．

特に、大域深さよりも局所深さが大きいバケットが多い場合には、検索速度が遅くなる．この欠点は、ディレクトリのメンテナンスを行なうことによって解消できる．

4.2 バケット分割

改良 Extendible Hash では、特にディレクトリのメンテナンスを行なわない場合、バケット分割の際に更新するエントリの数を少なくできるので、データを挿入するプロセスの応答時間を Extendible Hash より短くできる．

ディレクトリメンテナンスを行なうと、更新するエントリの数は Extendible Hash に近付き、特に、大域深さが同じ場合は Extendible Hash と等しくなる．

4.3 ディレクトリ拡張

改良 Extendible Hash では、Extendible Hash のディレクトリ拡張に相当する場合でも1～2個のエントリを更新するだけでよいので、インデックスへのデータ挿入の応答時間を短縮できる．

ディレクトリ拡張時にディレクトリのメンテナンスを行なう場合でも、クリティカルセクションは1～2個のエントリの更新に短縮できる．さらに、ディレクトリのメンテナンス中に他のプロセスが分割後のバケットにアクセスできるので、同時実行性を高めることができる．

5 あとがき

動的ハッシュ法の1つである Extendible Hash を改良し、検索法を工夫することによってバケット分割の際に更新するエントリ数を削減し、ディレクトリ拡張の際のでも矛盾なく検索・更新を行なえる改良 Extendible Hash を提案し、処理速度と同時実行性について考察した．

文献

[1] R. J. Enbody, H. C. Du, "Dynamic Hashing Schemes" ACM Computing Serveys, Vol. 20, No. 2, pp. 85 - 113, June 1988.

```
begin
    i := 大域深さ
    loop
        ハッシュ値の下位 i ビットを用いてディレクトリエントリを参照
        if 局所深さ > i
        then i++
        else バケット内を検索して終了
    endloop
end
```

図 3: 検索アルゴリズム

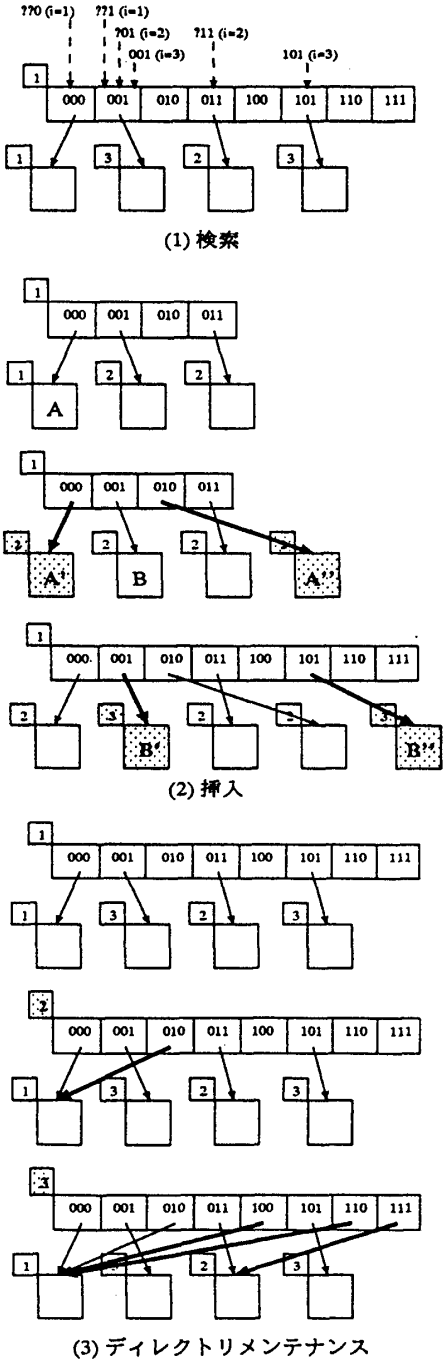

図 4: 改良 Extendible Hash の例