5P−3

統合化ファイルアクセス法（ＩＦＡＭ）の性能評価*

鈴鹿 豊明　盛屋 邦彦　古舘 丈裕†
日立ソフトウェアエンジニアリング（株）‡

1　はじめに

筆者らは既存ファイルアクセス法の構造及び機能的な包含、新しいアプリケーションへ対応するための機能的な拡張を目指した統合化ファイルアクセス法（IFAM : Integrated File Access Method）を提案し [盛屋 92]、試作を行った。次の段階としては、実際にIFAMが既存ファイルアクセス法の代用として利用可能か、またデータベースの構築や新しいアプリケーションに対応可能かを検証する必要がある。

そこで本稿では、まず既存アクセス法への理論的な対応方法及び性能予測を行い、実際の動作テストから比較／評価を行った。その結果、まだ物理的なアクセス方法の改善点はあるものの実用の見通しを得ることができた。

2　IFAMの評価基準

IFAMの目標の一つに既存ファイルアクセス法の構造／機能的な包含がある。これはいままでに蓄積された既存アクセス法を使ったアプリケーションがIFAMにファイルシステムを移行した後もそのまま利用できるようにすることである。以前筆者らは従来のファイルアクセス法はその性質からストリーム型とランダムアクセス型の2つに大別されることを示した。前者はそれ以上分けられないデータ要素（バイト、ワード等）を単位に、要素の順序位置をもとにしたアクセスであり、後者はいわゆるレコードをアクセス単位とし、レコードに付けられたキーをもとにしたアクセスである。ランダムアクセス型のファイルアクセス法への対応はIFAMでは、基本的にランダムアクセス型の構造及び機能を持っているので、そのまま拡張したものとして利用可能である。一方ストリーム型のファイルアクセス法に対してはレコードが無制限長で内容がアトム列となっており、それに対する操作も豊富なため、

1. ストリーム型ファイルアクセス法における1つのファイルに対してはIFAMの1つのレコードを対応させる。
2. ストリームファイルアクセス法のファイル操作についてはIFAMのレコード内操作を対応させる。

ことで、表現できる。一方、新しいアプリケーションへの対応としては、長大レコードのサポート、それに対する挿入／削除（レコード内操作）をサポートしているがこの実用上の性能も検討する必要がある。またIFAMをデータベースの基本アクセス法として利用するためには、該当レコードの集合処理の性能も考慮する必要がある。これは主にキーアクセス（IFAMではタグによるアクセス）により得られたレコード集合同士の和や積をとるための性能ということになる。まとめると、性能の基準としては、

1. レコードへのタグによるアクセス性能、2つのレコード集合同士の和／積 演算性能
2. レコード内操作 挿入／削除 の性能

ということに設定した。

*Performance of Integrated File Access Method
†Toyoaki Suzuka, Kunihiko Moriya, Takehiro Furudate
‡Hitachi Software Engineering Co., Ltd.

3　IFAMのレコードアクセス性能

ここでは、IFAMのレコードに対するタグによるアクセス及び、レコード集合同士の和／積 演算性能を評価する。

3.1　理論的予測

IFAMのタグ管理については、インデックスの一般的実現手法であるB^+-treeを採用している。したがって、論理的には同じタグが一つのレコードのみに付いている場合は全タグ数をNとすると、計算量はその木の高さである log N のオーダとなる。しかし、IFAMでは同じタグが複数のレコードに付けることができるため、この場合には、集合操作の計算量を考慮する必要がある。同じタグの付いたレコード集合はレコードを代表するレコードIDをインデックスとしたB^+-treeを用いている。これによりレコード集合はレコードIDによりソートされた順序集合になっている。したがって、レコードを特定するには更に順番を指定することになる。この時の計算量は同じタグを持つレコードの数をMとすると log M のオーダとなり、タグを指定して目的のレコードIDを得る計算量は合わせて

$$\log N + \log M$$

のオーダとなる。レコード集合同士の和／積に関しては、各レコード集合のレコードIDはソートされているため、単純なつき合わせにより求めることができる。全タグ数をN、レコード集合Aのレコード数をM1、レコード集合Bのレコード数をM2とすれば、和／積とも計算量は

$$M1(\log N + \log M1) + M2(\log N + \log M2)$$

のオーダとなる。

3.2　レコードアクセス性能の実測

実測では

1. 全タグ数を固定、同じタグを持つレコード数を変化させた時のディスクアクセス回数の推移
2. 同じタグを持つレコード数を固定、全タグ数を変化させた時のディスクアクセス回数の推移

3. 全タグ数を固定、あるタグを持つレコード集合を固定、更に別のタグを持つレコード集合の大きさを変化させ、集合同士の積を求めるディスクアクセス回数の推移

のテストを行った。以下に理論値及び実測値のグラフ図1、図2、図3を示す。

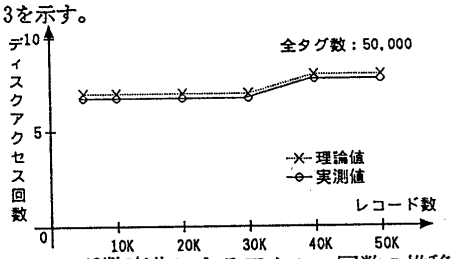

図 1: レコード数変化によるアクセス回数の推移

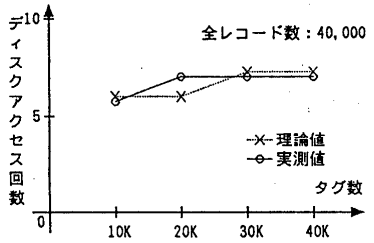

図 2: 全タグ数変化によるアクセス回数の推移

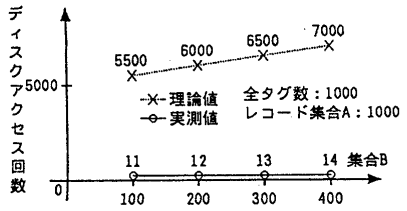

図 3: 積集合を求めるアクセス回数の推移

図 1、図 2を見ればわかるようにレコードへのアクセスの実測値はほぼ理論値通りの推移を示した。ただレコード集合及びタグを構成する木は分岐数が大きく（約100分岐）、これは log の底なのでレコード数やタグ数の影響が少ないことがわかる。したがって、大量の件数のレコードを扱うアプリケーションに対しては有利であることが考えられる。それに対して積集合を求めるテストでは、ディスクアクセス回数の実測値が理論値のそれに比べかなり下回った。これはIFAM 内部でバッファを持っており LRU に基づくディスク／主記憶のアドレス管理を行っているため、連続したアクセスではディスクへのアクセスが軽減されるためであると考えられる。したがってデータベースのような大量データの選択／付き合わせが主となる処理に関しては実用に耐え得ると思われる。

3.3 IFAM のレコード内操作の性能

ここではレコード内操作の性能の評価を行う。IFAM ではマルチメディアデータのような長大レコードを扱うことを考慮に入れ、既存アクセス法には無い挿入／削除の機能を持たせている。したがって、特にこのデータ編集機能の性能を評価する必要がある。

3.4 理論的予測

一つのレコードのデータ構造としては OB-tree[STON84] を用いている。これは、B^+-tree のリーフノードに逐次的に要素を蓄積し、中間ノードにはキーの代わりに自分の子部分木中の要素数（これを重みという）を持たせる手法である。したがって、B^+-tree と同様に挿入／削除の各計算量はレコード内の全アトム数 N に対して log N のオーダとなる。

3.5 レコード内操作の実測

実測では

1. レコード長を変化させた時の挿入操作に対するディスクアクセス回数の推移
2. レコード長を変化させた時の削除操作に対するディスクアクセス回数の推移

のテストを行った。以下に理論値及び実測値のグラフ図4を示す。

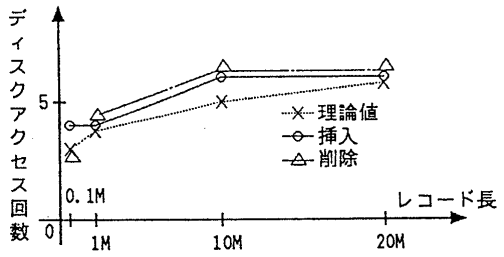

図 4: レコード長に対する各操作のアクセス回数推移

図 4を見ればわかるように、挿入と削除に関しては木のデータ充填率が影響を与え、場合によっては分割併合が起こるためか、ある程度アクセス回数がばらついている。環境を変え、テストの回数を増やせば理論値に近付くと考えられる。このテストでもレコードを構成する木の分岐数が多い（約100）ため、レコード長がアクセス回数にあまり影響を与えてないことがわかる。音声のような長いレコードの編集作業には有利なものと思われる。

4 むすび

本稿では、統合化ファイルアクセス法の性能の実測を行い、理論値との比較を行った。筆者らはこの結果から統合化環境におけるアクセス法として利用できる見通しを得たと考えている。今後の課題としてはレコードを構成する OB-tree はまだ改善の余地が残っており、さらに長大レコードの内部操作の効率化を図っていく予定である。

参考文献

[盛屋 92] 盛屋、鈴鹿: 統合化環境を目指したファイルアクセス法の提案, 情報処理学会第 44 回全国大会, 1992.

[STON84] M. Stonebraker, L. A. Rowe: Database Portals :A New Application Program Interface, Proc.10th VLDB, 1984.