5P-5

BTRONファイル管理システムの開発
— アクセスモデルシミュレーション —

坪崎　徹　　杉下　幸司　　今井　良彦
松下電器産業株式会社　情報システム研究所

1. はじめに

BTRONファイルシステムでは、管理情報に対するアクセスの頻度が高いが、全ての管理情報をメモリ上に持ち難い。従って、管理情報のアクセスの高速化のために各種のアルゴリズムに対してシミュレーションを行い、この結果をインプリメントに反映させた。本稿ではシミュレーションの結果について報告する。

2. BTRONファイル管理[1][2]

BTRONでは、複数可変長レコードから成るファイルを管理するために以下の管理情報を持っている。

(1) ファイルヘッダ部(192B)とレコードインデックス部から構成されるファイルヘッダブロック(1KB)
(2) レコードインデックス部
　① レコード数がある数より少ない場合、レコードの位置を示すポインタを格納(レベル0、図1)
　② レコード数がある数より多い場合、レコードインデックスブロックの位置を示すポインタを格納
　レコードインデックスブロックは、レコードの位置を示すポインタを格納(レベル1、図2)

3. アルゴリズム

シミュレーションを行なったアルゴリズムは大別するとキャッシュ(ディスクバッファ)を獲得するアルゴリズムとFCBのサイズの2種である。

(1) キャッシュ獲得アルゴリズム
① 優先度付き

キャッシュを使用するデータは、ファイルヘッダブロック(FH)、レコードインデックスブロック(RI)、一般データの端数(GD)の3種類がある。これら3種類のデータに優先度を付して優先度の高いデータはできるだけキャッシュに残すアルゴリズムである。同一優先度の場合は新しいものをキャッシュに残す。優先度の付け方は3種類のデータにすべて優先度を付ける方法が3！ = 6通り、1種類のデータのみ優先度を付ける方法が3通り、優先度を付けない方法が1通りの計10通りである。本稿では、この内優先度を付けない(FIFO)、RI, FH, GDの順に優先度を付ける(rfd)、FH, RI, GDの順に優先度を付ける(frd)の3つについて報告する。

② 使用順序付き(LRU)

キャッシュのデータを入れ替える場合、最も長い間アクセスされていないデータを除去の対象とする。

(2) FCBのサイズ

FCBに格納するデータとして、ファイルヘッダ部のみとレコードインデックス部を含むヘッダブロック全体とが考えられる。後者は、高速化が見込まれるが、OSのサイズがかなり増える。このため、FCBの個数を減らしても同様の効果があるか否かも調べてみた。

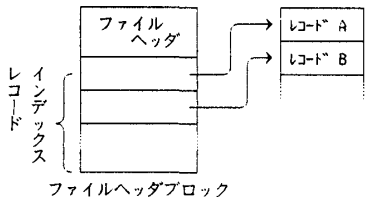

図1　レベル0ファイルの構成

4. アクセスモデル

BTRONでは仮身を次々に開いていくタイプが一般的である。ウィンドウを開いた場合、その実身内の仮身を表示する必要がある。このとき、ファイルのオープン、レコードの読込みが少なくとも行なわれる。この組をオープンウィンドウ処理と名付け、これを中核としてアクセスモデルを構築した。

1. オープンウィンドウ処理中のファイルのオープン、レコードの読込みの対象ファイルはランダムに選択する。
2. オープンウィンドウ処理の間に、ランダムにファイルの生成、削除を挿む。
3. ウィンドウをオープンした場合、そのファイルに対するリード/ライトを数回行なう。

5. 測定方法

BTRONのファイル管理モデルをUNIX上に構築し、BTRONで想定されているファイルアクセスモデルを設定することによりシミュレーションを行った。BTRONの各ファイルの管理情報にアクセスするとき、メモリ上にあるものにアクセスした場合、”ヒットした”と呼び、ディスクへの実IOが必要になった場合、”ヒットしない”と呼ぶ。全アクセス数のうち、ヒットした回数の比率を”ヒット率”と呼び、このヒット率が高いものを良い方法であると判断した。各アルゴリズムに対して、キャッシュ数、

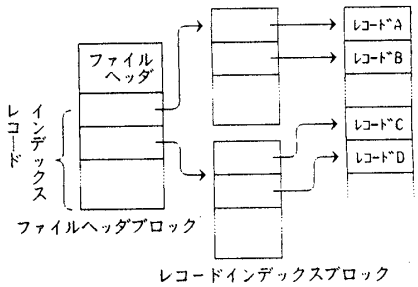

図2　レベル1ファイルの構成

A Development of BTRON File Management System
Toru Tsubosaki, Kouji Sugisita, Yoshihiko Imai
Matsushita Electric Industrial Co., Ltd.

ＦＣＢの個数を変化させて測定を行った。

6．結果、考察

以上のような条件でシミュレートした結果と考察を述べる。アルゴリズムＡがアルゴリズムＢよりヒット率が高い場合 Ａ ＞ Ｂ、同じ場合は Ａ ＝ Ｂ と記述する。

(1) キャッシュの個数の効果

① ＦＣＢ ＝ １９２Ｂ，６０個（図３）

キャッシュ数が６４個以下の場合、 ＬＲＵ ＝ ＦＩＦＯ ＞ ｆｒｄ ＞ ｒｆｄ であり、 キャッシュ数の増加に伴いヒット率も上昇する。キャッシュ数が２５６個以上の場合アルゴリズムと無関係に一定である。

ＦＣＢヒットを除いたＦＨのアクセスは全アクセスの約５０％を占めているので、優先度を付ける場合はＦＨに最高優先度を付ける方式が最も効果的である。データアクセスの局所性があるため、最高優先度のデータを比較した場合、ＬＲＵ、ＦＩＦＯは優先度付きより多少低くなる程度である。キャッシュ数が少ない場合、最高優先度以外のデータは、優先度付きでは非常に悪くなる。従って、全体のヒット率としては、ＬＲＵ、ＦＩＦＯが最高となる。

② ＦＣＢ ＝ １ＫＢ，６０個（図４）

キャッシュ数が３２個以下の場合、ＬＲＵ ＝ ＦＩＦＯ ＝ ｒｆｄ ＞ ｆｒｄであり、キャッシュ数が増加してもヒット率はあまり変化しない。キャッシュ数が６４以上の場合ＬＲＵ ＝ ＦＩＦＯ ＝ ｒｆｄ ＞ ｆｒｄであり、キャッシュ数を増加すればヒット率も上昇する（キャッシュ数が２５６個でほぼ定常状態）。

ＦＣＢヒットが全アクセス中の約７５％で一定であるので、全体のヒット率はキャッシュ数の影響をあまり受けない。ＦＨはＦＣＢでほとんどヒットするので、優先度を付ける場合は、ＲＩに最高優先度を付ける方法が最も効果的である。キャッシュ数が６４個以上の場合、ＦＨのキャッシュヒット率が著しく上昇する。

(2) ＦＣＢの個数の効果

各アルゴリズムともＦＣＢの増加に対するヒット率の上昇の割合はほぼ同じである。

① ＦＣＢ＝１９２Ｂ，キャッシュ数＝１６個（図５）

ＬＲＵ ＝ ＦＩＦＯ ＞ ｆｒｄ ＞ ｒｆｄである。

② ＦＣＢ＝１ＫＢ，キャッシュ数＝１６個（図６）

ｒｆｄ ＝ ＦＩＦＯ ＝ ＬＲＵ ＞ ｆｒｄである。

7．まとめ

ＦＩＦＯとＬＲＵは、ＦＣＢ数、ＦＣＢサイズ、キャッシュ数にかかわらず、最も安定したヒット率を示す。さらに、優先度を付けた場合、ＧＤのヒット率は極端に悪くなるが、ＦＩＦＯ、ＬＲＵの場合他の種類のデータと同じくらいのヒット率である。

ｒｆｄは、ＦＣＢが１ＫＢの場合、非常に有効であるが、メモリの増加を考慮するとＦＩＦＯ、ＬＲＵに劣っている。

ｆｒｄは、条件を変化させてもあまりよいヒット率は得られず、他のアルゴリズムに比べ有効ではない。

ＦＣＢを１ＫＢにした場合、ｒｆｄ以外のアルゴリズムではメモリの増加に見合う効果は得られない。

キャッシュ数を増加するよりＦＣＢの個数を増加する方がメモリの増分に対するヒット率の上昇分が良い。

8．おわりに

キャッシュ数、ＦＣＢ数、ＦＣＢサイズを変化させ、シミュレーションを行い、その結果を報告した。今後、ＦＣＢ、キャッシュのサーチのオーバヘッドを考慮に入れたシミュレーションを行い、ＢＴＲＯＮの性能の向上を目指す予定である。

参考文献

[1] 岩村他 ：「実身／仮身モデルのインプリメンテーション」、第3回リアルタイムアーキテクチャＴＲＯＮ研究会資料(1987)、pp.2-21

[2] 坂村 ：「ＢＴＲＯＮのファイル管理システム」、第１回リアルタイムＯＳ －ＴＲＯＮ研究会資料(1987)、pp. 42-55

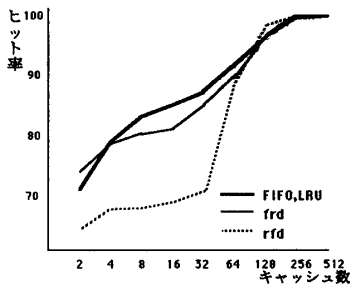

図3 FCB＝192B、60個のシミュレーション結果

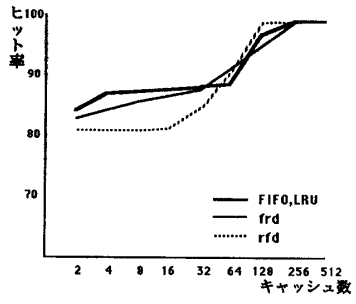

図4 FCB＝1KB、60個のシミュレーション結果

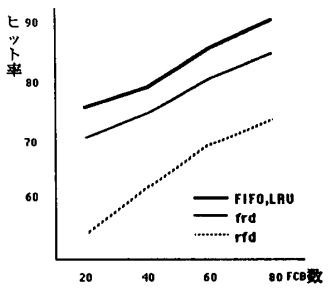

図5 FCB＝192B、キャッシュ数＝16個のシミュレーション結果

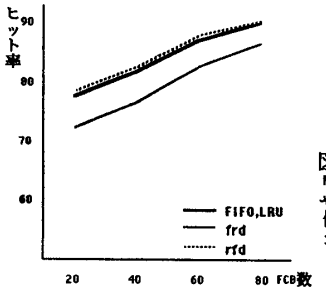

図6 FCB＝1KB、キャッシュ数＝16個のシミュレーション結果