UNIX−DBの検索アクセスコストの考察

4H−7

尾崎敬二
九州東海大学

1．はじめに

WS-PC(Workstation-Personal Computer) LANシステムによる中規模図書館の電算化は実用的であることを示し、その検証を行って来ている。[1]図書館業務は多岐にわたっているが、実用的なシステムとするために、今回は、特に、図書検索についてさらに検討を加えたので報告する。実用的な図書検索が行われるための重要な要素のひとつは、検索待ち時間である。入力された書誌データのレコード件数に対して検索待ち時間は、リニアに増加するのではないことが、示されている。[2]これは、単純な順次アクセス法ではなく、高速なバッファを利用したバッファ付き順次アクセス法を採用したためであることを示す。また、B-treeインデックスを利用したアクセス法において、検索待ち時間と総入力件数の関係から磁気ディスク装置における単位アクセスコストが見積れることを示す。

2．検索待ち時間のアクセスコストの評価

今回使用しているシステムの構成は処理速度が約5 MIPSのWSをホストとし、PC端末が6台のLANシステムである。データベースはUNIX-DBMSであるUnify(Ver.4.0)を使用して構築したものである。現在の累積総入力件数n_tは約5万3千件であり、2つのテーブル間のリレーションだけが設定してある。書誌データの1件あたりの定義平均長は約1.5KBであり、実質的な長さは約900ﾊﾞｲﾄである。検索待ち時間の測定は出来るだけ検索に要するコスト以外の要素が入り込まないように、簡素化したSQL文を使用し、UNIXの/bin/timeコマンドのreal timeの値を採用しているが、n_tが3万件程度までは、図書検索用のアプリケーション実行時に、ストップウォッチで計時したものである。測定に使用したSQL文の一例は以下の通りである。

```
lines 0
select count(*) from wa_sho
    where [書名1='*文学*'] /
```

同じ条件下では、ストップウォッチ計時の値とreal timeの値は同一である。
また、特に検索待ち時間は以下の式(1)で表されるように、総入力件数n_t、選択されたレコード件数n_a、ホストマシンの負荷状況xの関数とみなされるので、n_aやxは、ほぼ同一条件となるようにして、測定されている。

$$t = f\ (n_t, n_a, x) \qquad (1)$$

総入力件数n_tに対する検索待ち時間tの変化を両対数グラフにしたものが図2である。
ここで、t_1は、中間一致条件（*日本*）、t_2は、前方一致条件（91/9/26）、t_3は高速前方一致条件（夢*）によるものである。t_1とt_2はバッファ付きの順次アクセス法（buffered sequential）であり、t_3はB-treeインデックスによる高速アクセス法である。

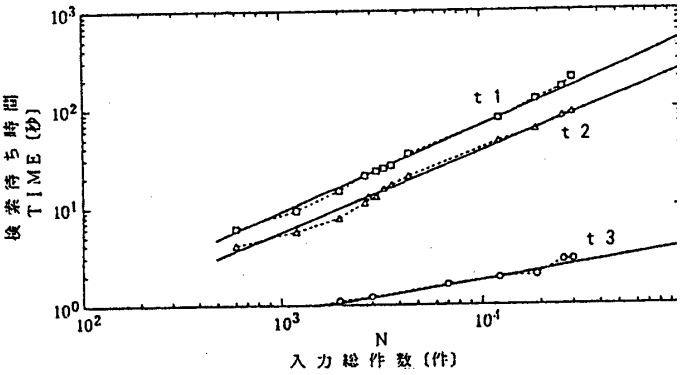

図1．総入力件数に対する検索待ち時間

検索待ち時間t_1〜t_3は総入力件数n_tのべき乗にほぼ比例しており、このグラフの傾きから、そのべき指数βが求められる。
アクセスコストを計算するために、以下のようにメモリ上の単位コストを定義する。
τ_1をメインメモリ上のバッファにおける単位コスト、τ_2を磁気ディスク上における単位コストとする。ディスプレイ装置に表示するコストは選択されたレコード件数n_aが小さい場

Consideration of the Retrieval Access Cost of the UNIX-Database System
Keiji OSAKI
KYUSHU TOKAI Univ.

合は、無視できる。まずt_2について評価してみる。t_2を式(2)のように近似する。

$$t_2 = n_t \cdot \tau_2 / b + n_t \cdot \tau_1 \quad (2)$$

すると、t_2に関するべき指数は

$$\beta_2 = \frac{d(log_{10} t_2)}{d(log_{10} n_t)} \quad (3)$$

$$= 1 - \frac{\tau_1}{\tau_2} b \quad (4)$$

となる。ここでbはバッファの大きさをレコード数に換算したものである。これにより、バッファの効果によってβが1以下になることが示される。次に、t_3について検討する。

B-treeインデックスを持つファイルのアクセスコストは、B-treeの高さhに比例するので式(5)のように近似できよう。

$$t_3 = (log_{k+1} n_t + 1) \cdot \tau_2 + n_t \cdot \tau_1 \quad (5)$$

$$\frac{dt_3}{dn_t} = \frac{\tau_2}{n_t \cdot ln2} \quad (6)$$

$$\beta_3 = \frac{1}{\{(h+1) + n_t \cdot \tau_1 / \tau_2\} \cdot ln2} \quad (7)$$

ここで、kはB-treeの度数（order）であり、ここでは、2以上の値である。

t_1では、中間一致条件検索なので、レコードを検索した後にさらに、フィールド内での文字列の検索時間が加わる。しかし、n_tに対する依存性はt_2と同様である。

3．測定結果と考察

図1に示されるt_2のグラフの傾きからβ_2を求めると、0.8〜0.9である。今回使用しているUNIX-DBMSのUnifyでは、バッファの大きさは通常のUNIXのバッファより、かなり大きく、少なくともレコード件数に換算して、10件以上とみられる。通常磁気ディスクのτ_2はメインメモリ上のτ_1と比べて10〜100倍程度大きいとみなされる。式(4)からだけでは、βを見積もれないが、式(6)からτ_2を評価することができる。その前に式(5)を検討してみる。n_tの範囲が1000〜約30000までのt_3に関するデータからdt_3/dn_tを差分近似したものを両対数グラフ上にプロットしたものが、図2である。このグラフの傾きは、ほぼ-1であることから、式(6)に示されるようにdt_3/dn_tがn_tに反比例していることが確認される。

このdt_3/dn_tの値を使用して求めたτ_2の平均値は0.227[sec]となり、今回のホストWSの磁気ディスク上での探索単位コストは約200[ms]であることが、判明した。以上の結果を式(4)に代入してみると、τ_2/τ_1がおよそ100程度

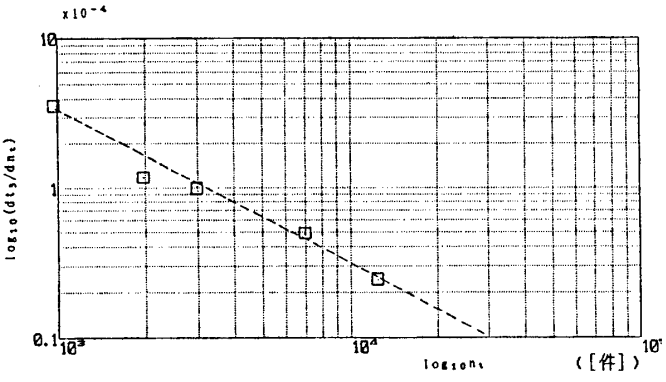

図2．dt_3/dn_tのn_t依存性（両対数グラフ）

とすると、bの値は10〜20程度となる。これは、妥当な値と言えよう。式(7)について検討考察してみる。hの値はn_tが1000〜20000の範囲にあるときは、6〜9の範囲にある。τ_1/τ_2の項を省略すると、β_3の値は0.21〜0.14となる。図1．に示されるt_3のグラフから求まるβ_3の値は約0.3であるので、実測値より、計算値の方が小さい。これは、B-treeの度数がレコード件数によって変動しているためではないかと推測される。

今回SQL文で測定した結果の中で、B-treeインデックスがつけられている項目の複合条件検索を行ったところ、単一の項目での検索待ち時間より、複合条件にした場合の方がかなり大きくなることが判明した。すなわち、OR条件を例に取ると、

$$t_3(c1 \cup c2 \cup c3) > t_3(c1) + t_3(c2) + t_3(c3)$$

ここで、c1〜c3は検索条件を示す。このことから、複合条件検索の場合は高速なB-tree項目については、各条件を独立に検索させて、その結果を結合させる工夫をすると、かなり高速化が図れるものと思われる。

3．まとめ

B-treeを付加された項目について、高速前方一致条件検索時間t_3と総入力件数n_tの両対数グラフから磁気ディスク装置の探索単位アクセスコストが見積もれることが判明した。また、バッファ付きの順次アクセスコストはn_tのべき乗に比例するが、その指数β_2はバッファの大きさに影響されることがアクセスコストの評価式によって明示された。B-tree項目の高速検索におけるβ_3に関しては、実測値との違いが大きく、今後の検討課題である。

（参考文献）

1)九州女子大学紀要；第27巻第1号
尾崎敬二他　；1992年3月
2)情報処理学会第42回全国大会（前期）
尾崎敬二、木村美奈子；1991年3月