オンライン情報蓄積・検索システム "UNIQ-1" の性能評価と最適化について†

小畑 征二郎†† 松沢 茂†† 宮崎 正俊††

情報蓄積・検索システムの性能は，データの蓄積に要する時間，検索における応答時間，データベースを構成するファイル・スペース量などで評価される．本論文では，オンライン情報蓄積・検索システム "UNIQ-1" を対象として，実験による性能評価を試み，それによって得られるデータをもとに決定論的観点からシステム動作を解析し，データの性格に応じたシステムの最適化手法について考察する．実験では，システム性能の一つの決定要因であるインデックス・ファイルのデータ・ポインタ部のサイズを変化させて，データの蓄積時間，検索時間，ファイル・スペース量を測定した．これから，データ・ポインタ部の最適サイズが求められた．また，蓄積時間と検索時間は，それぞれにおけるインデックス・ファイルの I/O 回数に比例することがわかった．決定論的解析による計算では，蓄積時間，検索時間，インデックス・ファイル・サイズともほぼ実測値に近い値が得られた．これより，蓄積しようとするデータのために，最も適したデータ・ポインタ部のサイズを求めることができた．

1. はじめに

最近は各所で商業ベースの情報検索サービスが行われるようになっているが，大学などの研究者の間ではデータベースを自分なりに構築して，学術情報の整理と検索をしたいという要望も多い．われわれはこのような要求に応えるため通常の学術文献データの検索サービスのほかに，個人もしくは研究室などが単位での学術文献データベースの構築とその利用もサポートできるオンライン情報蓄積・検索システム UNIQ-1 (Universal Information Query) を開発した[1),6)]．

このシステムは，現在東北大学大型計算機センターの ACOS システム 1000 (OS は ACOS-6) のもとで実用に供されている[7,8)]．その形態は一般利用者ジョブと共存して動作するものであり，データベースを構成するファイルも一般利用者のファイルと同じスペースを共有する．また，扱われているデータは，各人の研究分野によって，内容，量，利用面などでかなりの相違がある．このため，データベースを構築する場合には，扱うデータに適したシステム構成をとらないと効率の悪いものとなり，他のジョブ処理に悪影響を及ぼすのみならず，よけいな使用料を支払うことにもなる．

† A Performance Evaluation and an Optimization of the Online Information Retrieval System "UNIQ-1" by SEIJIRO OBATA, SHIGERU MATSUZAWA and MASATOSHI MIYAZAKI (Tohoku University Computer Center).

†† 東北大学大型計算機センター

ところが，既製品の情報検索システムを使ってデータベースを構築する場合，そのシステムで作成するデータベースの論理的構造は確定しているので，システムの最適化を図るには，一般に，扱うデータの性格に応じたファイルの設計が唯一の手段となる．

本稿では，UNIQ-1 を使ってデータベースを構築する場合のデータの性格に応じたファイルの設計に関して，実験と決定論的観点から考察し，最後にその最適化について述べる．

2. システム性能の決定要因

UNIQ-1 は，文献データのような文字情報の蓄積と検索を主目的としたオンライン型の情報検索システムであり，利用者はこのシステムを使うことによってデータベースの定義から，データの蓄積，その利用と管理までを一貫して TSS 端末から行うことができる．これを使って作成されるデータベース・システムの構成概念を示すと**図1**のようになる．つまり，UNIQ-1 は，ACOS-6 の TSS の一つのサブシステムとして動作し，誰でも利用できるようになっている．

UNIQ-1 を使って作成されるデータベースは，データベース定義ファイル (以後 DDF: Database Definition File という)，インデックス・ファイルおよびデータ・ファイルからなる．DDF には，データベースの種々の属性が書き込まれており，ここを通してデータベースを構成するすべてのファイルがアクセスされる．インデックス・ファイルには，データの

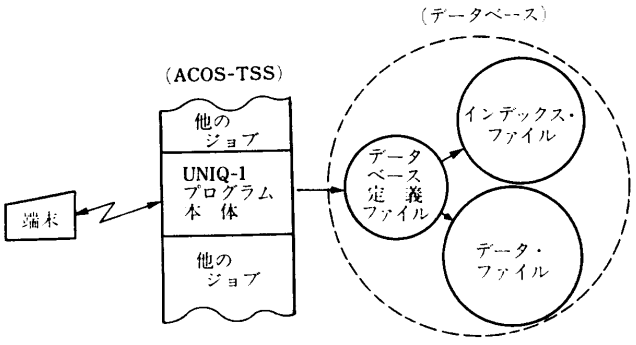


図 1 データベース・システムの構成
Fig. 1 Configuration of database system.

キーワードが登録されているが，キーワードのアドレッシングにはハッシュ法を用い，ハッシングの衝突の処理には連鎖法 (Chaining 法) を用いている．そのインデックス・ファイルの構成は，**図 2** に示すようにハッシュ・レコード部，コリジョン・レコード部，オーバフロー・レコード部からなる．また，データ・ファイルは，**図 3** に示すようにプライマリ・レコード部とオーバフロー・レコード部からなる．そして，これらの各部のサイズは，任意に設定することができる．したがって，データの性格に応じたファイルの最適化とは，これらの最適サイズを決定することである．

ところが，データ・ファイルは単純で，かつアクセスもデータの蓄積時と検索結果の出力時に各 1 回と少ない．また，パーソナルな文献データベースではデータの項目も厳選され，かつアブストラクトを入れることも少ないので，**図 4** の例のようにほぼ一定サイズになるのが通常である．したがって，データ・ファイルの最適サイズの決定は，比較的簡単である．問題になるのは，よ

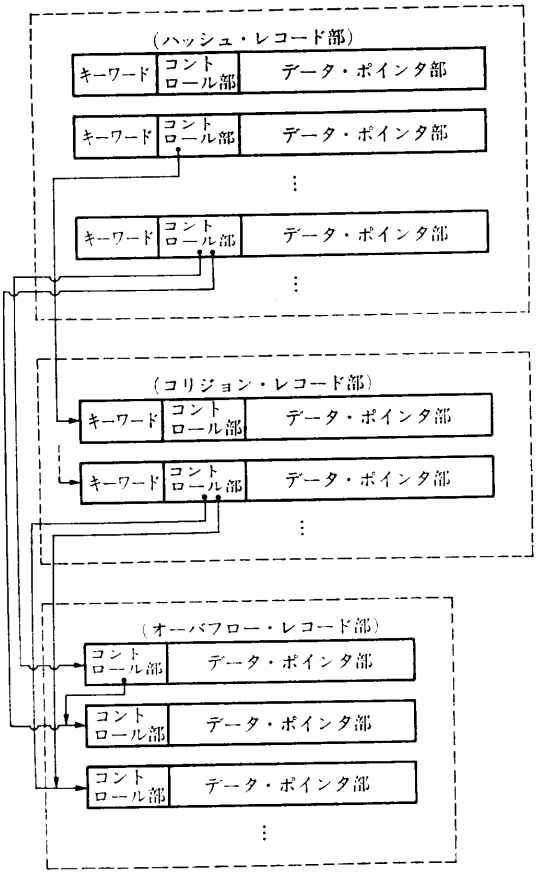


図 2 インデックス・ファイルの構造
Fig. 2 Structure of index file.

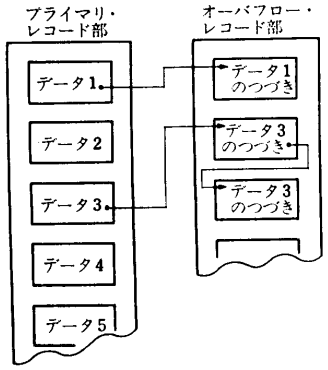


図 3 データ・ファイルの構造
Fig. 3 Structure of data file.

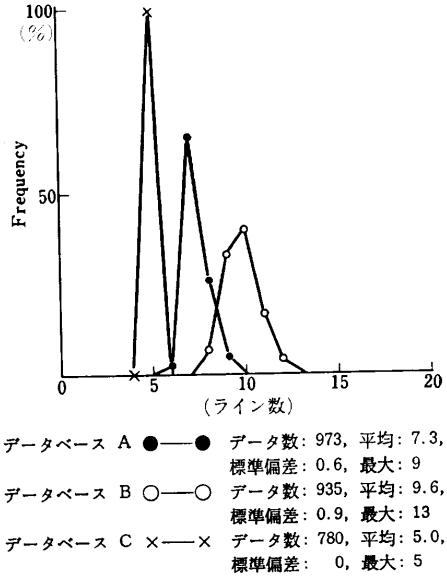


図 4 1 データのライン数の分布
Fig. 4 Distribution of line per data.

り複雑な構造をし，かつ検索時などに頻繁にアクセスのあるインデックス・ファイルの設計である[2),3)].

そこで，本稿ではインデックス・ファイルの最適化に関して考察するが，インデックス・ファイルの論理的な構造は決まっているので，インデックス・ファイルの最適化とは，蓄積するデータに最も適したデータ・ポインタ部のポインタの記録許容数（データ・ポインタ部のサイズという）を決定することである[4),5)]．また，このときの評価尺度は，計算機システムに対して最も負荷の大きい，使用 CPU 時間とファイル・スペース量である．つまり，評価尺度は，(1)データ蓄積に要する CPU 時間，(2)検索に要する CPU 時間，(3)インデックス・ファイルのスペース量である．

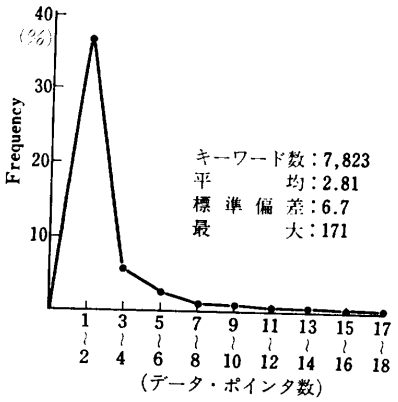


図 5 データ・ポインタ数の分布
Fig. 5 Distribution of data pointers.

3. 実験による考察

3.1 実験方法と使用データ

実験では，ハッシュ関数は，組込みのものを使い，インデックス・ファイルの各レコード・サイズを変化させたときの，データの蓄積と検索に要する CPU 時間およびファイル・スペース量の変化について測定した．インデックス・ファイルのハッシュ・レコードとコリジョン・レコードは，キーワード部，コントロール部およびデータ・ポインタ部からなり，オーバフロー・レコードはコントロール部とデータ・ポインタ部からなる．各レコードのコントロール部には，そのレコードのデータ・ポインタ部に記録されたポインタ数，そのレコードのキーワードと衝突したキーワードを格納するコリジョン・レコードのアドレス，データ・ポインタ部のあふれ部分を記録するオーバフロー部の最初のレコードのアドレスと最後のレコードのアドレスが記録される．また，キーワード部は 12 文字に固定してある．したがって，インデックス・ファイルのレコード・サイズを変えるとは，データ・ポインタ部のサイズを変えることにほかならない．なお，システムでは，各部（ハッシュ・レコード部，コリジョン・レコード部，オーバフロー・レコード部）のデータ・ポインタ部のサイズは別々に変化させることができるが，実験では一律に変化させるようにした．

測定に用いたデータは，INSPEC（英国電気学会，情報システム）の 1975 年から 1976 年までの Computer and Control Abstracts から任意に 3,000 件を抽出し，各データの項目をタイトル，著者，掲載誌（名，巻号，年），アブストラクト誌番号に限定した．つまり，実験に関係しない項目は除くと同時に，パーソナルなデータベースに近いものとした．そして，キーワードの切出しはタイトルから行った．それによるとキーワードの数は，7,823 個で，その延べ数は，22,016 個である．つまり，1 キーレコード当りの平均データ・ポインタ数は，約 2.8 個である．キーレコード数は当然キーワードの数と同じになる．また，データ・ポインタ数の度数分布は**図5**のようになる．なお，キーワードの切出しのためのデリミタは，「空白」，「，」，「．」，を使い，不要語は，冠詞，前置詞，助動詞などとした．

3.2 データの蓄積

UNIQ-1 のデータの蓄積では，1 個のデータを読むごとに指定の項目からキーワードを切り出し，インデックス・ファイルに登録する．そのとき，キーワードが初めて登録されるものであれば，ハッシングで決められたレコードにキーワードとデータのアドレス（データ・ポインタ）を書き込み，キーワードがすでに登録されているものであれば，登録されているレコードのデータ・ポインタ部の最後に，そのデータのアドレスを書き込む．このときのデータの蓄積法には，すでに蓄積されているデータを修正しながら蓄積する方法（オンラインの会話型蓄積）と新規データだけを順番に蓄積する方法（バッチ型蓄積）がある．前者を蓄積法(1)と呼び，後者を蓄積法(2)と呼ぶことにする．

蓄積法(1)では，すでに蓄積されているデータの行単位の修正や行単位の追加を行うことができる．修正情報や追加情報からキーワードを切り出して登録する際，すでに登録されているキーワードでは，そのキーレコードのデータ・ポインタ部に，修正するデータのアドレスが，すでに登録されているかどうかのチェッ

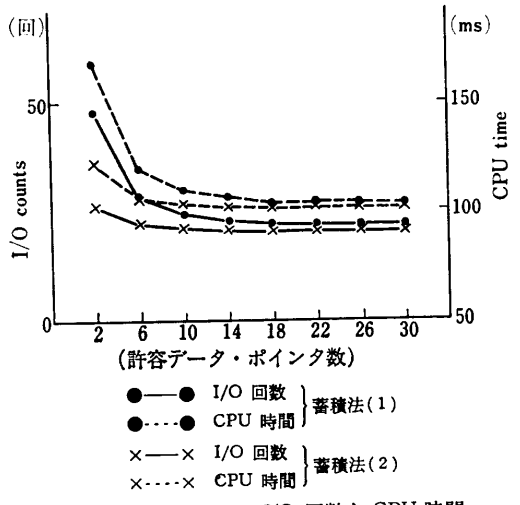


図 6 蓄積における I/O 回数と CPU 時間
(1データ当り)
Fig. 6 I/O counts and CPU time for store.

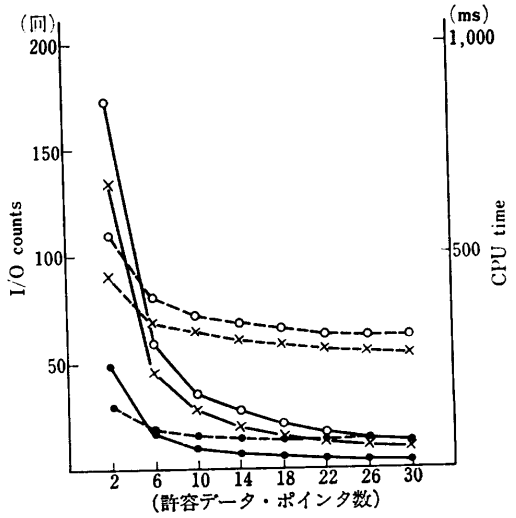
クを行って重複をさける必要がある．すなわち，そのキーレコードにリンクされているレコードをすべて読み出してチェックを行うことになる．

一方，蓄積法(2)では，次々と入力されるデータのキーワードが，すでに登録されているものであってもそのキーレコードにリンクされている最後のレコードのデータ・ポインタ部に，たんにデータのアドレスを追加すればよい．ハッシュ・レコードとコリジョン・レコードのコントロール部には，リンクされているオーバフロー・レコードの最終レコードのポインタも書き込まれている．

図6は，3,000件のデータをタイトルからキーワードを切り出して蓄積するときの1データ当りの蓄積に要するインデックス・ファイルの平均I/O回数と，平均 CPU 時間をデータ・ポインタ部のサイズごとに示したものである．実線は I/O 回数で，点線は CPU 時間である．なお，使用している CPU は約 15 MIPS の速度である．これによるとデータ・ポインタ部のサイズは，蓄積法(1)の場合は約18より大きくしても，また，蓄積法(2)の場合は約10より大きくしても，意味がないことがわかる．また，I/O回数と CPU 時間はかなり強い相関があることがわかる．なお，実験ではデータはあらかじめ普通のシーケンシャル・ファイルに入れておき，そこからデータベースに蓄積するようにした．

3.3 データの検索

図7は，検索におけるインデックス・ファイルの I/O 回数と所要 CPU 時間をデータ・ポインタ部のサイズごとに示したものである．検索におけるキーワードとしては，サイズごとの値の比較をしやすいようにデータ・ポインタ数の比較的多い語を用いた．それらを A，B，C とすると，それぞれのデータ・ポインタ数は，98，128，75 個である．測定は，A だけで検索した場合，AとBの論理積で検索した場合，A，B，C の3語を論理積で与えた場合に分けて行った．図より所要 CPU 時間に関しては，データ・ポインタ部の大きさを約18より大きくしても意味がないことがわかる．



—— I/O 回数，---- CPU 時間，● A の場合，
× A・B の場合，○ A・B・C の場合．

図 7 検索における I/O 回数と CPU 時間
Fig. 7 I/O counts and CPU time for retrieval.

3.4 インデックス・ファイルのスペース

図8は，データ・ポインタ部のサイズとインデック

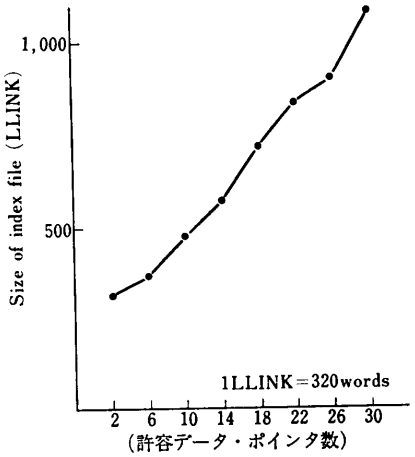


図 8 インデックス・ファイル・サイズ
Fig. 8 Size of index file.

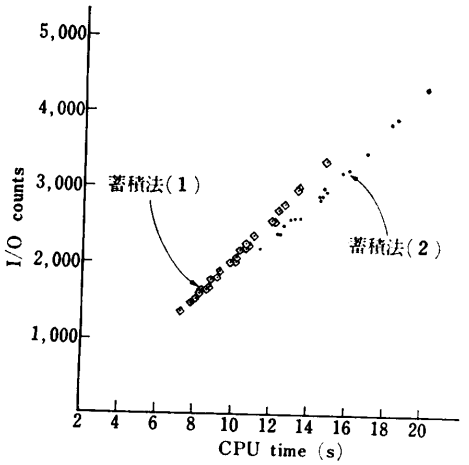


図 9 I/O 回数と CPU 時間の関係
Fig. 9 Relation between I/O counts and CPU time.

ス・ファイルの全スペース量の関係を示したものである．なお，1データ・ポインタは1語に格納され，1 LLINK は 320 語である．

3.5 評　価

この実験では，蓄積におけるインデックス・ファイルの I/O 回数と所要 CPU 時間，検索におけるインデックス・ファイルの I/O 回数と所要 CPU 時間，およびデータ・ポインタ部のサイズとインデックス・ファイルの大きさの関係を調べた．これによると，3,000 件ぐらいの文献のタイトルのような文章からキーワードを切り出すと，データ・ポインタ部のサイズは 6〜10 語ぐらいが最適なことがわかる．また，蓄積における所要 CPU 時間も検索における所要 CPU 時間もインデックス・ファイルの I/O 回数に強い影響を受けることがわかる．

図9は，蓄積法(1)と蓄積法(2)における I/O 回数と CPU 時間の相関図である．これは 3,000 件のデータを蓄積する過程において，蓄積法(1)では 100 件単位で，蓄積法(2)では 150 件単位で計測したものである．これによると I/O 回数と所要 CPU 時間は比例関係にあることがわかる．

4. 決定論的考察

実際にデータを蓄積する場合，そのインデックス・ファイルの最適化を，3 章で述べたような試行錯誤で行うことは不可能である．したがって，蓄積しようとするデータをあらかじめ分析し，それをもとに最適なインデックス・ファイルの設計を行う必要性が生じる．

4.1 蓄積における考察

(1) 蓄積法(1)の場合

蓄積法(1)では，あるキーレコードに1個のデータ・ポインタを追加するたびに，データ・ポインタが重複しないように，リンクされているすべてのレコードを読み出して調べる必要がある．したがって，k 個のデータ・ポインタをもつキーワードを登録するときのインデックス・ファイルの I/O 回数は，

$$\sum_{j=1}^{k}[(j-1)/s]+2\cdot k \qquad (1)$$

となる．ただし，s は1レコードのデータ・ポインタ部のサイズ（単位は語）であり，[ ] はガウスの記号である．したがって，k 個のデータ・ポインタをもつキーワードの度数を $F(k)$ とすると，全キーワードを格納するための I/O 回数 N は

$$N=\sum_{k=1}^{K}\left(F(k)\left(\sum_{j=1}^{k}[(j-1)/s]+2\cdot k\right)+[F(k)\cdot(k+1)\cdot\alpha]\right) \qquad (2)$$

となる．Kは区間数 ($1\leqq K\leqq$全キーワード数)，α は衝突率である．

ところが，蓄積における I/O 回数と所要 CPU 時間の関係は，3 章で述べたように1次関数となる．それを実測値から求めると

$$T=2.42N+51.48 \qquad (3)$$

となる．N が I/O 回数で，T が所要 CPU 時間である．

図10は，図5に示したデータ・ポインタ数の度数分布をもとに1データ当りの I/O 回数と所要 CPU 時間を計算し，実測値と比較したものである．ただし，

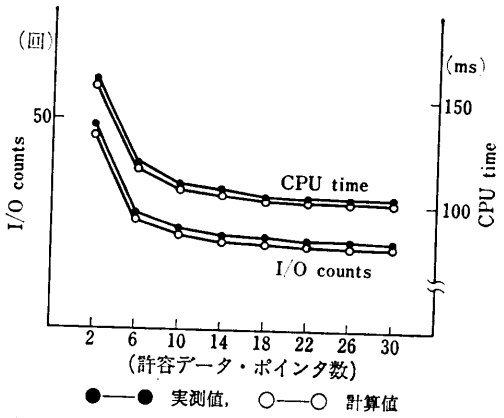


図 10 蓄積法(1)における計算値と実測値の比較
Fig. 10 Comparison between calculated values and measured values on the method (1) of store.

計算における衝突率は，実測とほぼ等しい30%と仮定した．図によると計算値と実測値はほぼ一致することがわかる．

（2） 蓄積法(2)の場合

蓄積法(2)では，k 個のデータ・ポインタをもつキーワードを格納するときの I/O 回数は

$k \leqq s$ ならば $2k$ (4)

$k > s$ ならば $2k+(k-s)-[(k-1)/s]$ (5)

となる．したがって，k 個のポインタをもつキーワードの度数を $F(k)$ とすると，全キーワードを格納するための I/O 回数 N は

$$N=\sum_{k=1}^{s} F(k)(2k+(k+1)\cdot\alpha) + \sum_{k=s+1}^{K} F(k)(3k-s-[(k-1)/s] +(k+1)\cdot\alpha) \quad (6)$$

となる．

蓄積法(2)の場合の I/O 回数と所要 CPU 時間の関係は，実測値から求めると

$$T=4.55N+8.745 \quad (7)$$

となる．

図11は，図5に示すデータ・ポインタ数の度数分布をもとに1データ当りの I/O 回数と所要 CPU 時間を計算し，実測値と比較したものである．蓄積法(2)の場合も計算値と実測値は，ほぼ一致することがわかる．

4.2 検索における考察

データ・ポインタ部のサイズを s とし，J 個のデータ・ポインタをもつキーワードの検索に要するインデックス・ファイルの I/O 回数 I は

$$I=[(J-1)/s] \quad (8)$$

である．したがって，n 個のキーワードの検索に要する I/O 回数 N は

$$N=\sum_{i=1}^{n}[(J_i-1)/s]+[\alpha\cdot n] \quad (9)$$

となる．

I/O 回数と所要 CPU 時間の関係は，実測値から求めると，キーワード1個（データ・ポインタ数は 98）の場合は

$$T=1.6N+61.8 \quad (10)$$

となり，キーワード2個（データ・ポインタ数98と128）の論理積の場合は

$$T=1.3N+286.65 \quad (11)$$

となる．式(10)と(11)の定数項の違いは論理積の処理の有無による．

図12は，検索における I/O 回数と所要 CPU 時間の計算値と実測値の比較である．いずれの場合も I/O 回数と所要 CPU 時間は，計算値と実測値がほぼ一致することがわかる．

4.3 インデックス・ファイル・サイズ

インデックス・ファイル・サイズに関しては，衝突のためハッシュ・レコード部に登録できないキーワードを登録するコリジョン・レコード部の大きさと，各レコードのデータ・ポインタ部に入り切らないデータ・ポインタの処理のためのオーバフロー・レコード

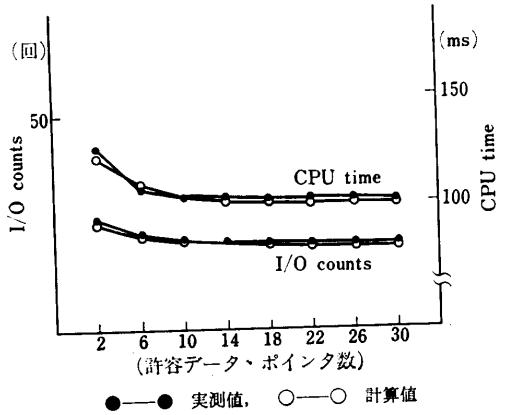


図11 蓄積法(2)における計算値と実測値の比較
Fig. 11 Comparison between calculated values and measured values on the method (2) of store.

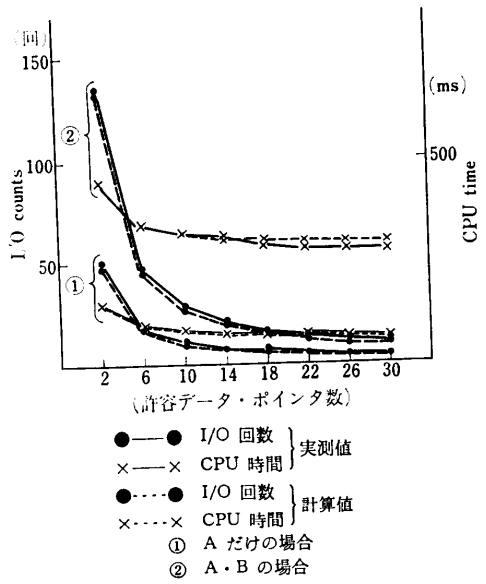


図12 検索における計算値と実測値の比較
Fig. 12 Comparison between calculated values and measured values for retrieval.

部の大きさが問題となる．いま，ハッシュ・レコード部のレコード数を定数 c とし，各レコードのデータ・ポインタ部のサイズを s とすると，k 個のデータ・ポインタをもつキーワードの度数が $F(k)$ である全キーワードを格納するためのインデックス・ファイル・サイズ M (単位は語) は

$$M=(s+7)\left\{c+\left[\alpha\cdot\sum_{k=1}^{K}F(k)\right]\right\}+(s+2)\left\{\sum_{k=1}^{K}([(k-1)/s]\cdot F(k))\right\} \quad (12)$$

となる．$(s+7)$ はハッシュ・レコードとコリジョン・レコードのサイズで，$(s+2)$ はオーバフロー・レコードのサイズである．

図 13 は，式(12)で求めた値を LLINK (=320 語)* 単位に直して実測値と比較したものである．これによるとインデックス・ファイル・サイズにおいても計算値と実測値はほぼ一致することがわかる．

4.4 決定論的考察における結論

データの蓄積や検索におけるインデックス・ファイルの I/O 回数，所要 CPU 時間およびインデックス・ファイル・サイズは，かなり正確に計算できることがわかった．しかし，所要 CPU 時間と I/O 回数の関係を示す一次関数の係数や定数項は，蓄積しようとするデータの性格，つまり，データ数，キーワード数，キーワードの分布などによって異なる[5]．これを蓄積するデータごとに求めるには，実際に蓄積を試みなければならず，非常に面倒で手間がかかる．

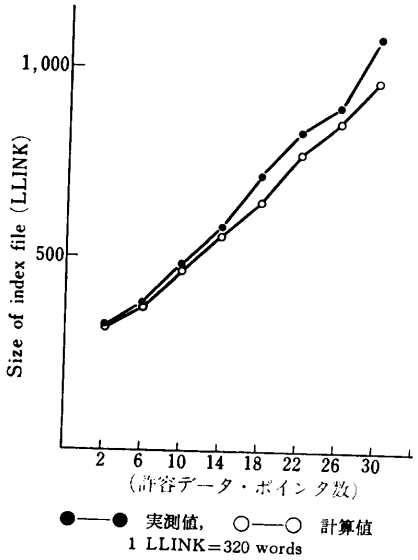


図 13 インデックス・ファイル・サイズの計算値と実測値の比較

Fig. 13 Comparison between calculated values and measured values on the size of index file.

* これには BCW (Block Control Word) 1語と，この中にとられる各論理レコードに付く RCW (Record Control Word) が含まれる．

5. 最適化手法とその応用

5.1 最適化手法

蓄積するデータのためのインデックス・ファイルの最適レコード・サイズを求めるには，データ・ポインタ部のサイズごとに蓄積や検索に要する CPU 時間とインデックス・ファイルのサイズを計算して，データ・ポインタ部のサイズごとにトータル・コストを求める必要がある．ところが，実験と決定論的考察より以下のことがわかっている．

(a) 蓄積法(1)の蓄積では，キーワードを登録するごとに検索をして，そのデータ・ポインタ部の最後に，蓄積しているデータのアドレスを追加することを繰り返す．つまり，蓄積法(1)の式(2)は，検索の式(9)を包含している．

(b) 蓄積におけるインデックス・ファイルの I/O 回数 N と所要 CPU 時間 T は

$$T=\alpha N+\beta \quad (13)$$

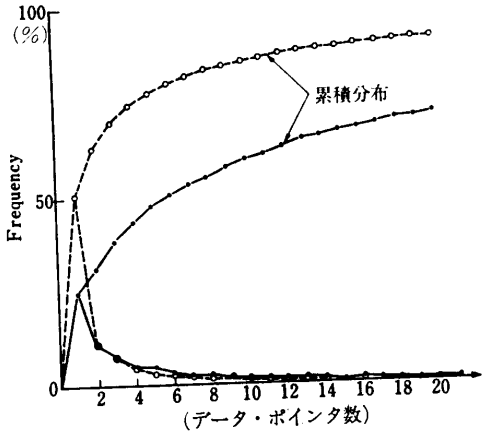
の関係がある．ただし，α と β は蓄積するデータの性格によって異なる．

すなわち，(a)の関係より，レコード・サイズの決定は，蓄積法(1)による所要 CPU 時間とインデックス・ファイル・サイズの関係から求められることがわかり，(b)の関係からは，必ずしも所要 CPU 時間は求める必要はなく，I/O 回数とインデックス・ファイル・サイズの値をデータ・ポインタ部のサイズごとに求め，相対的な比較を行えばよいことがわかる．

5.2 最適化手法の適用例

5.1 節で述べた最適化の手法を，ここでは，東北大学大型計算機センターでサービスしている METADEX (金属学関係文献データベース) に関して，適用した例を示す．

図 14 は，この METADEX (1980 年 1 月～3 月) 10,000 件の TITLE 項目と INDEX 項目からキーワードを切り出したときのデータ・ポインタ数の度数分布である．キーワードを切り出すときのデリミタは「空白」，「,」，「;」，「/」，「:」とし，不要語としては，冠詞，前置詞，助動詞，接続詞を使用した．同じデータであってもキーワードを切り出す項目によって，そ

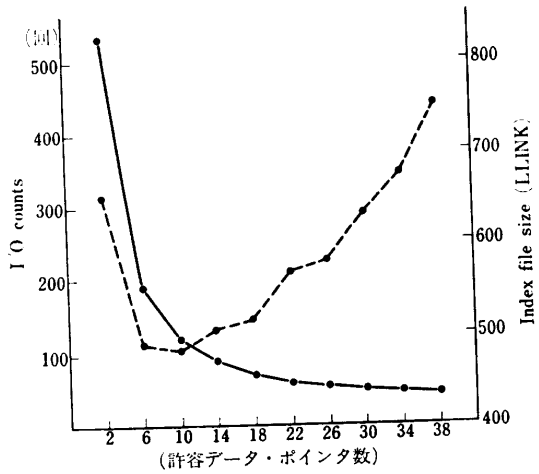


実線: INDEX 項目から
キーワード数: 2,570
データ・ポインタ数 108,453
点線: TITLE 項目から
キーワード数: 8,610
データ・ポインタ数 77,270

図 14 METADEX のキーワードの分布
Fig. 14 Distribution of keywords in METADEX.

の分布が変わることがわかる．なお，TITLE 項目から切り出されたキーワード数は 8,610 個で，INDEX 項目から切り出されたキーワード数は 2,570 個である．

図 15 は，図 14 で示したキーワードの分布の INDEX 項目のものをもとに蓄積における I/O 回数とインデックス・ファイル・サイズを求めたものである．これによると，INDEX 項目からキーワードを切り出して蓄積する場合のデータ・ポインタ部のサイズは，10 語前後が最適であることがわかる．



●——● I/O 回数／1データ
●----● インデックス・ファイル・サイズ

図 15 蓄積における I/O 回数とインデックス・ファイル・サイズ
Fig. 15 I/O counts and size of index file for store.

METADEX では，キーワードを INDEX 項目から切り出しているが，ハッシュ・レコード部とコリジョン・レコード部のデータ・ポインタ部のサイズは 10 語にしてあり，オーバフロー・レコード部のデータ・ポインタ部のサイズは 15 語にしてある．オーバフロー・レコード部の方を大きくしてあるのは，データ・ポインタ数の多いキーワードの検索における I/O 回数を減らすためである．一般に，データ・ポインタ数の多いキーワードは，検索における利用頻度も大きいので，このことは全体の I/O 回数を減らすことにもなる．

6. おわりに

本稿では，UNIQ-1 の性能を実験により評価し，それをもとに決定論的な観点から，データの性格に応じたインデックス・ファイルの最適化の手法に関して考察した．その結果，蓄積しようとするデータのキーワードの分布から，蓄積法(1)における I/O 回数とインデックス・ファイル・サイズをデータ・ポインタ部のサイズごとに計算することにより，インデックス・ファイルの最適化が可能であることがわかった．

しかし，実際にデータ・ポインタ部のサイズを決定する場合には，ファイル・システムの動作や物理的なファイルの構成法なども考慮する必要がある．また，ファイルの容量面では，ハッシュ・レコード部とコリジョン・レコード部のバランスも重要である．これは換言するとハッシュ関数と衝突の処理方式の問題で，従来から各所で研究されてはいるが[9)~11)]，われわれは UNIQ-1 に適応させるための問題として検討する予定である．また，UNIQ-1 では，オンライン時間やヒット数だけではなく，使用したコマンド，質問式などの詳細な利用記録もとれるので，今後は，これらの利用記録をもとに UNIQ-1 の利用面での評価も試みる予定である．

謝辞 最後に，UNIQ-1 の開発でご指導いただき，実験でもお世話になった東北大学金属材料研究所の小岩昌宏教授に深謝する．

参考文献

1) 小畑，松沢，宮崎：オンライン情報蓄積・検索システム "UNIQ-1" の設計と開発，情報処理学会論文誌，Vol. 23, No. 3, pp. 272-279 (1982).
2) Severance, D. G. and Caris, J. V.: A Practical

Approach to Selecting Record Access Paths, *ACM Comput. Surv.* Vol. 9, No. 4, pp. 259-272 (1977).

3) 小畑, 松沢, 宮崎: キー・ランダマイズ型情報検索システムの性能評価, 情報処理学会第20回全国大会論文集 (1979).

4) 小畑, 松沢, 宮崎: キー・ランダマイズ型インバーテッド・ファイルの評価, 電気関係学会東北支部連合大会論文集 (1979).

5) 小畑, 松沢, 宮崎: キー・ランダマイズ型情報検索システムの最適化に関する一考察, 情報処理学会第21回全国大会論文集 (1980).

6) 小岩昌宏, 小畑征二郎: METADEX-オンライン検索サービスの利用法, 東北大学大型計算機センター広報, *SENAC*, Vol. 13, No. 4 (1980).

7) 小畑, 松沢, 宮崎, 小岩: UNIQ-1 でサービスしている金属学関係文献データベース METADEX の利用分析, 情報処理学会第23回全国大会論文集 (1981).

8) 小畑, 松沢, 宮崎, 小岩: 文献データベース "METADEX" の検索サービスとその利用分析, 情報管理, Vol. 25, No. 1, pp. 25-34 (1982).

9) 古川康一: コンフリント・フラグをもったハッシュ記憶法, 情報処理, Vol. 13, No. 8, pp. 533-539 (1972).

10) 西原清一, 萩原 宏: 予測子を用いた Open Hash 法, 情報処理, Vol. 15, No. 7, pp. 510-515 (1974).

11) 後藤英一, 井田哲雄: ハッシング・プロセッサ, 情報処理, Vol. 18, No. 4, pp. 395-401 (1977).

(昭和57年 7 月7日受付)
(昭和57年12月 6 日採録)