機能分散型計算機における データ・サブシステム

吉田浩・田中英彦・元岡達
（東京大学　工学部）

1. はじめに

機能分散型計算機の構成を考える場合の最大の問題は，一つのまとまった計算機システムとして必要な機能を，どのように複数のサブシステムに分割して実現するかということである。その分割法の一つとして，図1に示すように，システム全体を次の4つのサブシステムによって構成する方法が考えられる。

(1) 機能サブシステム
(2) データ・サブシステム
(3) 入出力サブシステム
(4) システム管理サブシステム

機能サブシステムは，数値計算や言語の処理などを行なう。データ・サブシステムは，ファイルやデータベースの管理を行なう。入出力サブシステムは各種入出力装置の制御を行ない，システム管理サブシステムは，各サブシステム間の通信の仲介などを行なってサブシステム間の処理の同期をとる。

本報告では，以上の4つのサブシステムのうちからデータ・サブシステム[1]を取上げ，その機能について検討する。また，このデータ・サブシステムの実験システムとして，ファームウェア化された分散型マシン用のファイル・アクセス法μ-VSAMを実際に構成し，評価・検討を加えた結果について述べる。

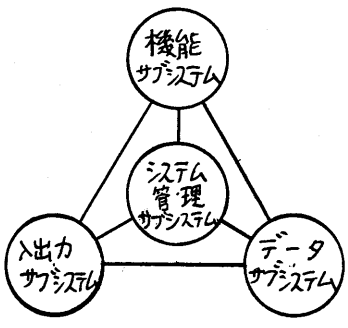

図1. 機能分散型計算機の構成

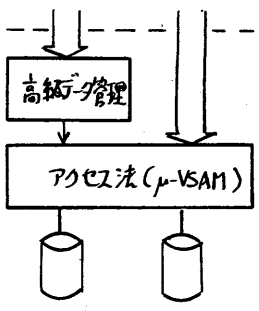

図2. データサブシステムのインタフェース

2. データ・サブシステムの機能

機能分散型計算機においては，複数のサブシステムが通信し合いながらジョブを処理してゆく。このため，各サブシステムへの機能の分割法が不適切であると，サブシステム間の通信量が増加し，オーバヘッドが過大になって，システム全体のスループットが低下する。また各サブシステムの構成のしやすさを考えると，その分割法はなるべく自然なものであることが望ましい。以上の理由により，ここでは各サブシステムを，通常のオペレーティング・システムの主要な機能の単位で分割する方法を考えた。

このようにして考えられたデータ・サブシステムの機能は，次のようなものである。

(1) 通常のオペレーティング・システムにおけるデータ管理の機能を独立させて，サブシステム化する。すなわち，システム全体で用いられる大容量のファイル記憶を管理し，他サブシステムに対しアクセスの手段を提供する。
(2) さらに進んだ機能として，関係データベースの管理や，ファイルに対するアクセスの多いユーティリティの機能をサポートする。これによりサブシステム間の通信量を一層減少させることができる。

データ・サブシステムの機能は，このように2つに大別できる。このうち後者の機能（高級データ管理機能）は，(1)の基本的な機能を基礎として，その上部に構築される。結局データサブシステムの，外部に対するインタフェースは，図2のように2レベルのものになる。1つは，アクセス法レベルのイ

ンタフェースであり，いわゆるデータ管理マクロ命令に類似のコマンドで，ファイル中のデータにアクセスすることができる。もう一つは，高級データ管理機能のインタフェースであり，ここではデータ操作言語（DML）や，各種ユーティリティの操作コマンドなどが，システムに対する命令となる。

なお，この機能分散型計算機システムの用途としては，最近とみにその重要性を増してきた会話型処理を考える。一般に会話型処理ではファイルを多用するため，このデータ・サブシステムはシステム全体の要となる部分であるとも言える。また，データ・サブシステムを，多数の知能端末と通信回線で接続することによって，ネットワーク中の大容量のファイルやデータベースの集中管理を専門にするような一種のデータベース・マシンとしての用途も考えられる。

3. データ・サブシステムのファイル・アクセス法μ-VSAM

以上のようなデータ・サブシステムの機能のうち，特にいの実現法を検討・評価するために実験システムを構成した。以下このシステムにおけるデータ・サブシステム用のファイル・アクセス法について述べる。これは基本的には，現在大型機のオペレーティング・システムを中心に用いられている仮想記憶アクセス法（VSAM）の処理アルゴリズムを基本として拡張しファームウェア化を図ったもので，μ-VSAMと呼ぶ。

μ-VSAMは，前節のように，システム全体のファイルを管理し他に対してアクセスの手段を提供する他に，(2)の高級データ管理機能を構築する基礎となる。このような要請からμ-VSAMの設計に当っては，以下の方針をとった。

(1) 同一ファイルに対して種々のアクセス形態（順アクセス，乱アクセス）を可能にする。

(2) サブシステムの外部に，ファイルの物理的構造や制約を極力意識させないようにする。

(3) アクセス法のコマンド体系は，できるだけ高級で，かつ単純なものとして，少数のコマンドで多くの処理ができるようにする。

(4) 外部とのデータの転送量を減らすのに有効な機能をできるだけ導入する。具体的には，前述のコマンドの高級化の他に，レコード中の特定フィールドの抽出・更新や，特定フィールドがある条件を満たすレコードを探索する機能，1回のコマンドで複数のレコードを処理する機能などである。

(5) 関係データベースや，テキスト・エディタなどを上部に構築することを考慮し，さらにこれらの機能の一部（たとえば関係データベースにおける射影や選択の演算など）をアクセス法に吸収して，上部の負担の軽減を図る。

(6) ファイルの作成，消去，コピー，形式変換，再編成などのファイル保守コマンドをアクセス法レベルで実現する。これらは通常は複数のコマンドの発行を要するものであるため，こうすることにより外部との通信量が低減できる。

(7) 専用プロセッサの構成という立場から，アクセス法全体をファームウェア化する。

(8) マイクロプログラムのレベルでマルチプログラミングを行ない，複数ユーザのコマンドを並行処理し，2次記憶へのアクセス待ち時間を有効に利用する。

(9) ファイルを作成するための2次記憶としては，磁気ディスク等のランダム・アクセス装置を考える。

前述のように，μ-VSAMは通常使わ

れているVSAMのデータ構造やアルゴリズムに準拠しているが，これは主に上記の(1)(2)の要求を満たすためである。

なおμ-VSAMにおけるファイルの構造は，VSAMのキー順データ・セット（KSDS）に類似のものであるが，その他に，記憶領域の利用効率を考慮して順編成ファイルも取扱っている。ただし，あくまでもKSDSを中心に考え，順編成ファイルはKSDSの記憶領域の使い方に適合するように制約をつけてサポートしている。

4. μ-VSAMのコマンド体系

今まで述べてきたように，機能分散型計算機のサブシステムの場合，インタフェースの設定が特に重要になる。データ・サブシステムのコマンド体系を考えるに当っては，先に述べた設計方針を十分考慮した。

表1にμ-VSAMのコマンドを示す。μ-VSAMのコマンドは，ファイル保守コマンドと，ファイル・アクセス・コマンドの2種に大別される。

ファイル保守コマンドの機能は，通常のオペレーティング・システムでは，ジョブ制御文やユーティリティを使って行なわれるものであるが，μ-VSAMにおいてはアクセス法のレベルでこれを実現している。なお，一般にVSAMファイルにおいては，ファイル作成時にかなり多くのパラメータを指定しなければならない。しかしμ-VSAMでは，特に会話型処理の便を考え，多少効率を犠牲にして，ほとんどのパラメータについて，これを固定化したり標準値を設けたりしている。そのためcreateコマンドなどはかなり簡単なものになっている。

表1. μ-VSAMのコマンド

コマンド名	処理内容
CREATE	ファイルの作成（レコード即時入力可）
DELETE	ファイルの消去
COPY	ファイルのコピー，形式変換，再編成
OPEN	ファイルの使用開始処理
CLOSE	ファイルの使用終了処理
GET	レコードの読出し（フィールド指定，条件指定，複数レコード読出し可）
PUT	レコードの挿入・更新（フィールド指定可）
ERASE	レコードの削除（複数レコードの削除可）
PAGN	レコードの更新と次レコードの読出し（順アクセスのみ）

ファイル・アクセス・コマンドは実際にファイル中のデータにアクセスするためのものである。これらは，通常のデータ管理におけるマクロ命令とほぼ同じような名称と種類をもっているが，従来複数のコマンドを必要としていた処理のいくつかは，1つのコマンドで処理できるようになっている。たとえば，順アクセスの開始点を定めるために従来はpoint命令を用いていたが，μ-VSAMでは最初のget命令で開始点の設定とレコードの読出しを同時に行なえる。また，レコードの削除の際，通常のアクセス法ではget命令でレコードを読出してからerase命令を発行するが，μ-VSAMではget命令は不要である。さらに，順アクセスでファイル中のレコードを次々と更新してゆくような処理のために，レコードの書出しと次レコードの読込みを同時に行なうpagn（put and get next）コマンド

(1) 乱アクセスによるput（レコードの挿入または更新）

```
open   fn=XX,access=WRITE,psw=--
put    fn=XX,mode=RANDOM,key=--
put    fn=XX,mode=RANDOM,key=--
close  fn=XX
```

(2) 順アクセスによるget

```
get    fn=XX,mode=SQ,key=--
get    fn=XX
get    fn=XX
```

(3) 順アクセスによるレコードの更新

```
get    fn=XX,mode=SQ,key=--
pagn   fn=XX
pagn   fn=XX
```

(4) キー範囲指定，フィールド抽出，フィールド条件指定を組合せたget

```
get    fn=XX,mode=SQ,key1=A,key2=B,
       field=(7,71),cond=((1,5),GE,'10000')
```

(5) キー範囲指定をしてレコードを削除する場合

```
erase  fn=XX,mode=SQ,key1=A,key2=B
```

図3. μ-VSAMのファイル・アクセス・コマンドの使用例

なども用意されている。

μ-VSAMにおいて順アクセスを行なう場合，基本的には待機アクセス法のように1レコードずつの処理がなされる。しかし，場合によっては開始キーと終了キーの2つを指定することにより，1コマンドで複数のレコードを処理することもできる。またレコードよりも細かい単位として，レコード中のフィールドの抽出・更新や，フィールドの値を調べてレコードを読出すことなども可能である。ファイル・アクセス・コマンドの使用例を図3に示す。

なお前述のように，μ-VSAMは関係データベースを実装する際の基礎となることを意識して作られている。この場合，関係モデルと物理的なファイルとは，関係とファイル，組とレコード，属性とフィールドという対応をとる。そして射影(projection)と選択(selection)はμ-VSAMの側である程度サポートするため，たとえば，2つの関係にそれぞれ射影と選択を施してからこれらを結合するという処理は，図4のように記述することができる。

5. μ-VSAMの実装

5-1. 実験システムのハードウェア

```
begin
time:=0;
get(t1,fn=R,mode=SQ,key=a1,cond=F1');
if found(R) then
  while ¬eof(R)∧t1.rkey=<b1 do begin
    get(t2,fn=S,mode=SQ,key=lowvalue,field=A');
    while ¬eof(S) do begin
      if t1.AθB then begin
        if θ='EQ' then t3:=(t1,t2.B̄)
        else           t3:=(t1,t2);
        if time=0 then begin
          put(t3,fn=T,mode=SQ,key=t3.tkey);
          time:=1 end
        else put(t3,fn=T) end;
      get(t2,fn=S) end;
    get(t1,fn=R) end
end.
```

注. 関係 R(rkey, r1, ⋯, rn), S(skey, s1, ⋯, sn) について次の演算を行なう。t1, t2, t3は組を表わす変数である。

選択 R[F1] → T1 (F1 = (a1≦rkey≦b1)∧F1', F1'は属性に関する条件)
射影 S.A' → T2 (A' = (skey, sj1, ⋯, sjk) という属性並び)
結合 T1[AθB]T2 → T (A∈A')

図4. μ-VSAMを用いた関係代数の実現例

図5に，μ-VSAMを実装した機能分散型計算機の実験システムの構成を示す。データ・サブシステムとして使用したポリプロセッサ・システムPPS-1[2]は，3台のマイクロプログラム制御のプロセッサが主記憶を共有しているもので，ダイナミック・マイクロプログラミングが可能である。主記憶にはさらにチャネルが接続され，容量9MBの磁気ディスク装置2台および入出力サブシステムとデータ転送を行なう。各プロセッサのマイクロ命令は24ビットの垂直型であり，1命令の実行時間は440 nsec.である。実験システムではPU2にμ-VSAMを実装し，PU1で入出力サブシステムとの通信を行ない，PU3は高級データ管理用とした。

入出力サブシステム[3]は，マイクロプロセッサM6800とAm2900を用いて試作されたものである。M6800により端末の制御およびPPS-1の主記憶とのデータのDMA転送を行ない，Am2900はサブシステム制御プロセッサとして，M-6800とPPS-1との間で割込みによる通信を行なう。

現在この2つのサブシステムにより分散処理を行なうことができるようになっている。その一例として，入出力サブシステムで端末から入力されたコマンドを中間形式に変換し，データ・サブシステムの高級データ管理プロセッサでこれをμ-VSAMのコマンドに直し，

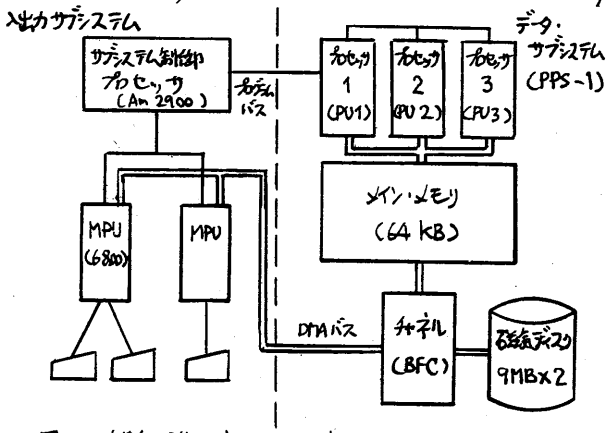

図5. 機能分散型計算機の実験システムの構成

μ-VSAMプロセッサで実際のファイルにアクセスするような「分散型テキスト・エディタ」の実装を行なった。μ-VSAMのコマンドが高級なために，この高級データ管理プロセッサのプログラムなどはかなり簡単になっている。

5-2. μ-VSAMのファイル構造と処理

図6にμ-VSAMにおけるファイルの構造を示す。μ-VSAMにおいては，コントロール・インタバル（CI），コントロール・エリア（CA）の大きさは固定であり，またデータ用のCIと，インデクス用のCIは同じ大きさである。CAはディスクの1シリンダに対応し，データ用のCAにおいては，その中の1つのCIは必らず対応するシーケンス・セットCIになっていて，いわゆるimbed方式に類似の方法がとられている。この他に，1つのCAにアクセスしている間は，対応するシーケンス・セットCIは必らず主記憶上に常駐させて処理の高速化を図っている。なお通常のVSAMで行なわれているキーの圧縮やインデクス・エントリのセクション化は，ここでは行なっていない。

アクセスの単位であるレコードは可変長であり，定位置に固定長のキーをもっている。またスパンド・レコードは許していないので，レコードの最大長には制限がつけられている。

システム中のファイル全体を管理するために，カタログが設けられている。カタログ自体も1つのファイルの形式をとっており，各レコードには，ファイル名をキーとして，そのファイルの最上位のインデクスCIへのポインタと，キー情報などが収められている。

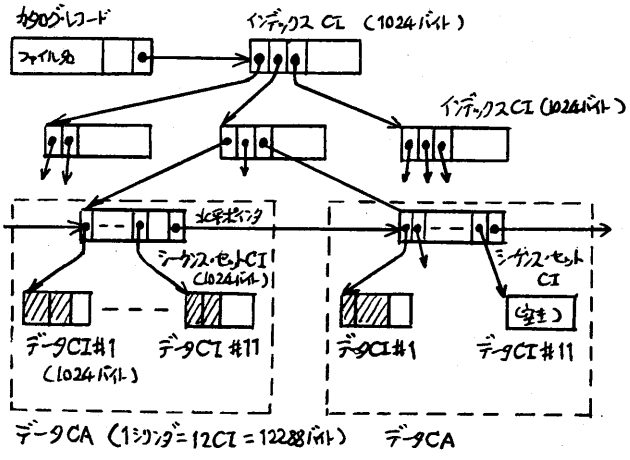

図6. μ-VSAMのファイルの構造（インデクス3段の場合）

ディスクの領域の管理は，システム中に1種類のファイルしか存在しないため非常に簡単になっている。すなわち，ディスク上の領域はすべてCI，CAの単位に分割され，この単位で領域の授受が行なわれる。そしてファイルの作成や拡張の際は，1つのファイルの領域はディスクのなるべく近いトラックにまとまるように確保される。

レコードの探索，読出し，更新，挿入と，それに伴なうCI分割，CA分割などの処理は通常のVSAMとほとんど同じである。なおレコードの削除時には木の再構成を行なわないので，木がバランスしなくなることがある。

5-3. マイクロプログラムの作成

以上のようなμ-VSAMの作成は，マイクロアセンブラ言語PAPPS1を用いて行なわれた。PAPPS1はPascalで記述されたクロス・アセンブラである。μ-VSAMのコーディング行数はコメントを含めて約1万行で，ステップ数は約8kステップである。コーディングとデバッグには約6人・月の労力を要している。

6. μ-VSAMの特性測定と評価

以上のようにして実装されたデータ・サブシステムのアクセス法μ-VSAMについて，ターンアラウンド時間やスループットを測定した。これらは，本来ならば種々の場合を想定して多角的に測定・評価すべきであるが，ここではファイルの形式としてはある限られたものを選び，それに対する処理も代表的なものにとどめた。

まず各コマンドの処理時間（ターン

アラウンド時間）であるが，これは前述のPPS-1の高級データ管理プロセッサ（PU3）から通信制御プロセッサ(PU1)を通じてμ-VSAMプロセッサ（PU2）にコマンドを発行し，その応答が返ってくるまでの時間を，PU3上のマイクロプログラムによって測定した。ファイルの作成，順・乱アクセスそれぞれのget，putコマンドの処理時間の測定結果を表2に示す。また，実測値の一例として，ファイルに対してその初期レコード数の100％増になるまで新レコードをランダムに挿入した場合の，挿入量に対する処理時間の変化の様子を図7に示す。同図は，異なる5本の乱数系列を用いて平均し，さらに40レコードごとに平均をとったものである。VSAMのこのような特性については，すでにいくつかの研究例があるが，[4] ここでもCI分割，CA分割の効果により，ある程度挿入量が多くなっても処理時間は最初とそれほど変わらないという性質が現われている。

ここで測定した値にはディスクのアクセス時間も含まれており，むしろそれが処理時間の大部分を占めている。その意味で，この値は単一のディスクにアクセスが集中するような場合のスループットの下限値であるとも言える。それに対して，ディスク装置やチャネルが十分に多数台あって，同一のディスクに対して別々のユーザのアクセスが競合することがほとんどないような場合を考える。この場合，ディスクのアクセス時間を利用して多重プログラミングを行なっているため，単位時間に処理できるコマンド数はプロセッサの処理時間のみで決まることになり，これがスループットの上限を与える。いくつかの代表的なコマンドについてこのプロセッサでの処理時間を実際のマイクロプログラムのステップ数から計算したものが表3である。

比較のために中型計算機FACOM 230-38の索引順編成ファイル（ISAMファイル）について，同様な処理に要するプロセッサの処理時間を実測した結果を表4に示す。測定はCOBOLで記述したプログラムを実際に実行させて行なった。プロセッサの速度が異なる上に，VSAMとISAMではファイルの構造や

表2. コマンドの処理時間

処理内容	1コマンド当りの処理時間(msec.)
ファイルの作成（1000レコード）	4000
乱アクセスのput（1000レコードずつ5回の平均）	127
順アクセスのput（ファイル最後へのレコード追加，1000レコード平均）	27
乱アクセスのget（ファイル作成直後，1000レコード平均）	88
〃 〃（1000レコード挿入後，1000レコード平均）	97
順アクセスのget（ファイル作成直後，1000レコード平均）	4.0
〃 〃（1000レコード挿入後，1000レコード平均）	3.8

注．レコード長は80バイト（固定），キーは2バイト，CI%は33%，CA%は27%である

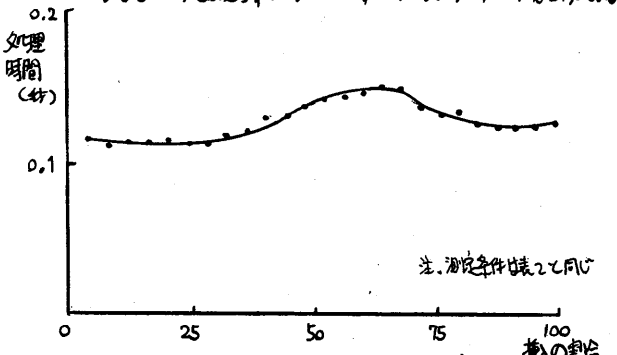

図7. レコード挿入時における処理時間の変化

表3. μ-VSAMのスループット

処理内容	プロセッサ処理時間(msec.)	1秒当りの実行可能回数
乱アクセス get	3.3	300
順アクセス get	0.84	1200
乱アクセス put	5.8	170

注．レコード長は80バイト（固定），キー長は8バイトとした。また，ファイルのインデクスの高さは3段とし，ファイル中のレコードの消去はないものとした。

表4. ISAMファイルにおけるプロセッサの処理時間

処理内容	プロセッサ処理時間(msec.)
乱アクセスget（ファイル作成直後）	3.8
〃 〃（8000レコード挿入後）	4.7
順アクセス get（ファイル作成直後）	0.59
〃 〃（8000レコード挿入後）	0.58
乱アクセス put（8000レコード挿入，平均）	20.1

注．レコード長は80バイト，キー長は8バイト，初期レコード数は10000個，シリンダ内，ブロック内の空きはそれぞれ25%とした。使用OSはOSII/VS，

処理方法もかなり異なるため，これらの結果の単純比較はできないが，それでも今回構成したような実験システムでも，現在の中型計算機と同程度のスループットをあげることは可能であると言える。また，FACOM230-38の命令のいくつかをPPS-1でエミュレートした場合，その実行時間は1.5倍ほどで大きくなると考えられることや，PPS-1のマイクロ命令の実行時間の440 n sec.という値は現在の水準から見ればやや遜色があることなどを考えると，今回の実験システムでとったような方法で，かなりスループットの高いシステムを構成することも不可能ではないと考えられる。

7. 検討と今後の課題

7-1. μ-VSAMの高速化

μ-VSAMの高速化を追求する場合，2つの方向が考えられる。1つはプロセッサによる処理の高速化であり，もう1つはディスクへのアクセス回数を減少させることである。

まず，プロセッサにおける処理の高速化について考える。前述のようにスループットを求めるためにマイクロプログラムのステップ数の分析を行なったが，その際，PPS-1の主記憶上でのデータのブロック転送，チャネルの制御，文字列の比較などの単純な基本的な処理に要する時間が比較的大きな比重をもっていることがわかった。それに対して，たとえばCI分割の際にレコードを2つないし3つのCIに分ける分け方を決める処理などは，判断や分岐が多く複雑であるが，処理時間そのものはあまり大きくない。また，VSAMに特有なCI分割やCA分割は確かに複雑で多くの処理時間を要するが，それでも私アクセスのputコマンドについて，CA分割が起きた場合，CI分割が起きた場合，分割が起きなかった場合のプロセッサにおける処理時間比は5：3：2程度である。そしてこれらの分割の起きる頻度は数回から数十回に1回程度である。このことから，データ・サブシステムのスループットをより向上させるためには，μ-VSAMの処理アルゴリズムを再検討するよりも，むしろ先に述べたような基本的な処理の高速化が重要であると言える。具体的には，これらの処理に適したマイクロ命令セットの設定，文字列の比較処理用の専用ハードウェアの利用，チャネル制御の簡単化やチャネルの高級化などが考えられる。

一方，これに対してターンアラウンド時間を直接に短縮するという意味では，ディスクへのアクセス回数の減少ということも重要な問題である。このためには，通常のVSAMで行なわれているように，キーの圧縮などによりインデクスCI中のエントリ数を増すことの他に，上位のインデクスを主記憶に常駐することや，前回の命令で読み込んだインデクスの再利用などの方法が有効と考えられる。

7-2. μ-VSAMの機能の拡張

今回実装したμ-VSAMは，実験システムとしても最初のものであるため，機能的には不十分な点が多い。そこで今後さらに機能を拡張してゆく上で特に重要と思われる点を列挙してみる。

まず，ファイルを複数ユーザで同時使用するための機構である。μ-VSAMはデータベース管理システムを作るための基礎となることをも設計の際に考慮しているが，その場合特に，1つのファイルを同時に使用することが必要になる。一般にB-treeを用いたファイルの場合，この種の処理は複雑になるが，これはぜひ考えなければならない機能であると言える。

次に，通常のVSAMではすでにサポートされている交代インデクスの機能で

ある。データベース管理を行なう時，この機能はデータの検索を高速化するために非常に有用である。

最後に，障害回復やエラーのチェックの機能である。現在のμ-VSAMはこの点がまだかなり不備であるが，実用的なシステムを考える上では重要な問題である。前述のスループット等も，これらの機能を備えた上で再評価する必要があろう。

7-3. データ・サブシステムの中核としてのμ-VSAM

今までμ-VSAMについて，主にその実装法などに関連した比較的細かい技術的な観点から論じてきた。ところで前述のように，μ-VSAMはデータ・サブシステムの機能の中核となる重要な部分である。そこで，最後にこのような観点から，μ-VSAMについて総合的に検討してみる。

μ-VSAMの機能としては，まず第一に，システム全体のファイルの管理があげられる。μ-VSAMは，種々の簡略化のために，通常のVSAMと比較すると速度や2次記憶の利用効率などの点で，多少性能が劣っている。しかし基本的な性能やアクセスの特性などは十分保存されており，むしろ簡略化によって，処理アルゴリズム等がより単純・明確になった点もあると言える。

もう一つは，上部にデータベース管理システムなどを作る場合の基礎としての機能である。この点についての評価はまだ十分には行なっていないが，μ-VSAMには，データベース管理に適したコマンドや機能がかなり用意されており，またテキスト・エディタ等の作成経験から，上部のプログラムはかなり簡単化できることがわかる。図4にはμ-VSAMを用いた関係データベース管理システムの一部を示したが，μ-VSAMの存在によって，このようなシステムの作成は，Pascal程度の記述能力のある高級言語を用いればかなり容易に行なえると考えられる。またその処理効率も，たとえばフィールドの抽出等の処理に要するオーバヘッドはわずかであるし，またフィールドの値を調べる機能も，無駄なデータ転送を防ぐことができるため，全体としてはそれほど低くないと思われる。

結局，データベース管理等についてはまだ問題も残ってはいるが，総合的に見て，このような方向を推し進めることによって，かなり高性能のサブシステムを構成することも十分可能であろう。

8. おわりに

本報告では，機能分散型計算機の一つの構成法を提案し，その要素の一つであるデータ・サブシステムについて特に詳細に検討した。またこの実験システムとして，ファームウェア化されたアクセス法μ-VSAMを実際に作成して，このようなサブシステムを構成することが可能であることを示し，その性能等について評価・検討を行なった。

今後の課題として，先に述べたμ-VSAMの改良と，より綿密な評価の他に，関係データベース等の高級機能の実現法についての検討，さらには機能分散型計算機全体の問題として，管理サブシステムの構成法やそのインタフェース，ジョブの実行方法などに関しての検討が必要と考えられる。

参考文献

1. 吉田，田中，元岡，“機能分散型計算機におけるデータ・サブシステム”，情報処理学会第21回全国大会，7K-2，pp.731-732 (1980).
2. 元岡，山室，“ポリプロセッサ・システム PPS-1”，情報処理，Vol. 15，No.7 (1974).
3. 正井，田中，元岡，“機能分散型I/Oサブシステムの一構成法について”，昭和52年度電子通信学会情報・半導体部門全国大会，No.304 (197
4. Keehn, D.G and Lacy, J.O., “VSAM data set design parameters,” IBM Syst. J., No.3, pp.186-212 (1974).