計算機システムの論理体系に関する一考察*

坂　和磨** 石田喬也** 飯川昭一**

Abstract

Recently a computer system becomes to be one of the components of a large information processing system. Thus, first of all, we must formalize the information processing system itself and after that we will be able to design a computer system compatible with a large information processing system such as MIS.

First, we construct the background (information space) of the information processing system using the new concept of File and Volume where a File is defined as a set of information and a Volume is defined as a container of a File. Next, we analyze how the information processing is acted upon this information space.

1. まえがき

計算機システムを情報を処理するひとつのシステムであるとしてとらえる．情報を処理する機能を保持するものを情報処理体と名付けるとすれば，世の中には計算機も含めて無数の情報処理体が存在し，各情報処理体がそれぞれ自分を中心としてひとつの情報処理システムを形作っている．それぞれは独立に，あるいは他の情報処理システムと連絡を保ちつつ情報処理作業をおこなっている．互に関連を持つ情報処理体は集まってさらにひとつの情報処理体を形成し，それがより大きな情報処理システムを形作ることが通常である．

計算機システム自身も他の情報処理システムから遊離した存在となることはできない．すなわち計算機システムをそれを包含するより大きな情報処理システムの一要素としてとらえることが重要である．従来計算機システムに対してはあくまでそれに固有の設計がなされ，常に計算機システム内に閉じこめて考えられてきたゆえにその外界との断絶が大きく，それを使う人間ならびにその集団あるいは他の計算機システムとの情報の交流を困難に，かつ複雑なものとしている．

計算機システムの論理体系に関する考察は，計算機システムの機能ならびに性能の向上の方向を定める上において，また一方ではバランスのとれたシステムを設計する目的に非常に重要な意味を持つ．従来この方面の研究は報告されてなくはない[1,2,3]が，すべて計算機の情報処理の域を出るものでなく，われわれがここで意図するような大局的な立場からの考察はなされていない．

ここでは情報*の集合をファイル，さらにその容器をボリュームと名付け，すべての Computing Resource はファイルでありボリュームであると一貫してとらえることにより，ひとつの情報処理システムとしての計算機システムの論理体系の統一的記述を試みる．このような広義の情報の世界の一部としてながめるとき，Multi-Processor System, Multi-User System, Man-Machine System, Multi-Level Storage System などは何ら特殊なものではなく，計算機システムという共通の基盤の上に立つ，単なるひとつの形態として容易に設計することができるであろう．

2. 情報の世界

情報処理体はひとつの情報の世界の中に存在する情報を処理の対象とし，情報処理体自身も情報の世界の中に存在するものであるとすれば，すべてに先立ちまず情報の世界を定義しなければならない．

情報の任意の集合をファイルと名付け，ファイルの任意の集合はまたファイルであると定義するとき，情報の世界はその世界に存在するすべての情報の集合で

* A Logical Structure of the Computer System, by Kazuma Ban, Takaya Ishida and Shoichi Iikawa (Mitsubishi Electric Corp.)

** 三菱電機株式会社　鎌倉製作所

* ここでは情報という言葉は単に“何らかの意味を持つデータ”の意味で使われている．

あるひとつのファイルであるといえる.

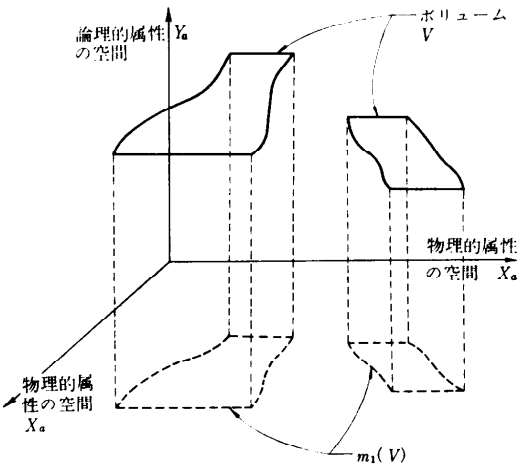
ファイルは単に情報の集合であるにすぎず，それがどこに存在し，どのようにしてアクセスされるかということとは，本質的に独立に定義されるものである．したがって，ファイルがアクセス可能なものとなるためには，それにアクセスの場が与えられなければならない．各ファイルに与えられるアクセスの場をボリュームと名付ける．これは概念的にはファイルの容器とみなされる．ボリュームにとって，その中味であるファイルは，そのボリュームのその時点の Status を表わす．特にファイルを保持しないボリュームは Empty Status にあるといわれる．ボリュームの集合はまた，ボリュームであると定義するとき，情報の世界はその世界の中に定義されるボリュームのすべての集合であるひとつのボリュームであるとみなされる．

そしてユーザはボリュームを介してのみファイルをアクセスできるというのが，この論文において定義される情報の世界である．

2.1 ボリュームの定義

以下にボリュームをより詳細に定義づける．ボリュームは物理的属性と物理的位置，ならびに論理的属性が与えられることにより，情報の世界の中に一意的に位置づけられる．すなわち，ボリュームは物理的属性の空間 (X_a)，物理的位置の空間 (X_l)，ならびに論理的属性の空間 (Y_a) の直積空間 ($X_a \times X_l \times Y_a$) におけるひとつの部分集合 ($V$) で表わされる (**第1図**)．

$X_a \times X_l \times Y_a$ 空間から $X_a \times X_l$ 空間への写像を m_1 で表わせば (m_1: $X_a \times X_l \times Y_a \to X_a \times X_l$)，ボリューム V のしめる物理的領域は $m_1(V)$ で与えられる．



第1図　ボリューム空間

物理的属性は，ファイルが保持される物理的媒体に固有の物理的特性であって，他のボリュームからのそのボリューム中のファイルのアクセス，あるいはそのボリュームの他のボリューム中のファイルのアクセスは，物理的にはこの属性により制御される．

同一の物理的属性を保持する物理媒体の集合は，ひとつの物理的空間を形成する．物理的位置は，その物理的空間内の特定の位置を与える．

論理的属性は，そのボリュームが持つ，以下に示すようなファイルのアクセスに関する論理的特性とボリュームの論理体系（後述）中にしめるそのボリュームに固有の位置を示す属性を与える．

(1) そのボリュームのファイルは，他のボリュームから入力され得るか，あるいは他のボリュームに出力され得るか．

(2) そのボリュームのファイルは，他のボリュームにより (Instruction Sequence として) 実行され得るか，あるいはそのボリュームは他のボリュームのファイルを (Instruction Sequence として) 実行し得るか．

(3) そのボリューム中のファイルあるいはそのボリュームに入れられるファイルのうち，どの情報は価値あるとして取り，どの情報は価値なしとして捨てるか（フィルタの役目）．

(4) そのボリューム中のファイルあるいはそのボリュームに入れられるファイルの各情報をいかに結合して，そのボリュームが意図するものとするか（ファイルの組織化）．

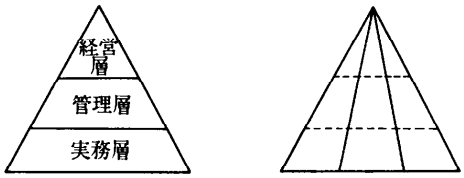
(5) そのボリューム中のファイルあるいはそのボリュームに入れられるファイルの各情報は，いかなる形式で表現されているか，またそれに伴ってその情報をいかにアクセスするべきであるか．

ボリュームの概念は MIS を例にとって，そこにおけるファイルの利用の仕方をながめるとき，その明確なイメージを描けるであろう．ひとつの企業が保持する情報の集合は，その企業のファイルを与えるが，その同一のファイルが，それがどのような目的で使われるかにより全く異なった組織化が要求されることが通常である．企業に属するメンバーはは大別して次の3種類のいずれかに分けらる．

(1)経営層，　(2)管理層，　(3)実務層

これらの3層に属するメンバーは，それぞれの仕事の遂行のために一般に上位層のものほどより概括的な

情報を，下位層のものほどより詳細な情報を要求する．一方，上位層のものといえども，ある特定の分野に関してはかなり詳細な情報を必要とする場合もある．前者はファイルの横方向の組織化であり，後者は縦方向の組織化であるといえる[4]（**第2図**）．この際，ファイルはひとつであり，それを収容するボリュームが，各



(a) 横方向の組織化　　(b) 縦方向の組織化

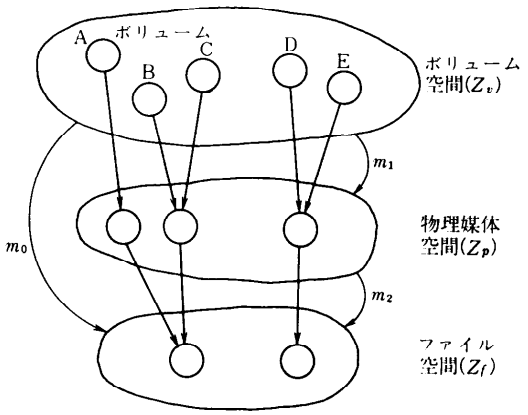
第2図　MIS におけるファイルの組織化

層に対して横方向，縦方向の2種類，計6種類用意される．そして，その特定のひとつのボリュームを選択すれば，希望するように組織化されたファイルをアクセスすることができる．

情報の集合をファイル空間（Z_f）と名付ける．それに対応して $X_a \times X_l \times Y_a$ 空間をボリューム空間（Z_v），$X_a \times X_l$ 空間を物理媒体空間（Z_p）と名付けるとすれば，ファイルがシステムにとってアクセス可能なものとして実在することは，次の写像 m_0 が定義されていることと等価であると定義する（**第3図**）．

$$m_0 : Z_v \to Z_f, \quad m_0(Z_v) = Z_f$$
$$m_0 = m_1 \cdot m_2, \quad m_1 : Z_v \to Z_p, \quad m_1(Z_v) = Z_p$$
$$m_2 : Z_p \to Z_f, \quad m_2(Z_p) = Z_f$$

ただし，Z_f は空集合をとひとつの元として持っているとする．m_0 は Z_v から Z_f の上への写像であり，



A と B，A と C は Logical File Sharing の関係にある
B と C，D と E は Physical File Sharing の関係にある

第3図　ボリューム空間からファイル空間への写像

それは Z_v から Z_p への写像 m_1 と Z_p から Z_f への写像 m_2 の結合で与えられることを示す．これらは上への写像であるが1対1写像ではない．すなわち，各ファイルに対しては少なくともひとつのボリュームが割り当てられなければならず，かつ各ファイルにひとつ以上のボリュームを割り当てることができることを意味する．写像 m_1 はボリュームが物理媒体空間のどの領域の上に定義されるものであるかを示し，写像 m_2 はその領域にどのようなファイルが入れられているかを示す．

ここで，ふたつの相異なるボリューム V_n, $V_m (n \neq m)$ の間に $m_0(V_n) = m_0(V_m)$ の関係があるとき，両ボリュームはファイルを共有していることを表わす．さらに特に $m_1(V_n) = m_1(V_m)$ の関係があるとき，両ボリュームはファイルを物理的に共有している（Physical File Sharing）といわれ，$m_1(V_n) \neq m_1(V_m)$ であるとき，両ボリュームはファイルを論理的に共有している（Logical File Sharing）といわれる．前者の場合，一方のボリュームの Status の変更は同時に他方のボリュームの Status の変更を帰結する．ファイルを物理的に共有するか，論理的に共有するかは用途に応じて使い分けられるであろう．

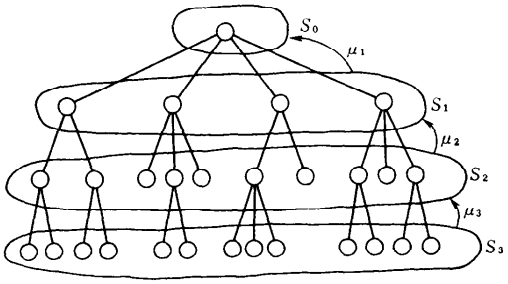
2.2　ボリュームの論理体系

情報の世界全体を表わすボリューム自身も，ボリューム空間の中に定義されるひとつの部分集合である．それを S_0 で表わす．S_0 は互に重なりあわないひとつ以上のボリュームに分割され，各ボリュームはさらに互に重なりあわない，いくつかのボリュームに分割される．ボリュームにとってそれが包含するボリュームを Inferior Volume，それが包含されるボリュームを Superior Volume と名付ける．特に他のボリュームを介することなく直接包含する，あるいは包含されるボリュームをそれぞれ Immediate Inferior Volume, Immediate Superior Volume と名付ける．

ボリューム空間においては，このようなボリュームの包含関係を定義するために各ボリュームに特定のランクが賦与されている．ボリュームのランクが i である（ランク i ボリューム）とはその Immediate Inferior Volume のランクが $i+1$ であり，Immediate Superior Volume のランクが $i-1$ であることを表わす（$i=1, 2, \cdots$）．特にランク0ボリュームは最高位のランクを持ち，ひとつの情報の世界の中に唯一存在する．上記の S_0 がそれに相当する．

ひとつの情報の世界のすべてのボリュームは，その

ランクにより類別されることができる．ランク i ボリュームの集合を S_i で表わし，S_i から S_{i-1} への写像を μ_i で表わすとすれば（$\mu_i: S_i \to S_{i-1}$），ボリュームの論理体系は，ボリュームの部分集合，$S_0, S_1, \cdots, S_N$ と写像 $\mu_1, \mu_2, \cdots, \mu_N$（$N$ はその情報の世界のボリュームの最大ランクを示す）で表現される（**第4図**）．



第4図　ボリュームの論理体系（$N=3$）

従来のファイル・システムでは，ボリュームの概念が存在しないため，写像 m_0, m_1, m_2，ならびに写像 $\mu_1, \mu_2, \cdots, \mu_N$ を不可分で混融したものであるファイル・ダイレクトリによりファイルの体系が記述されてきた．そのためにファイルの共有は，たとえばLink[5]という特殊な道で表わされなければならない．このようなからみあったファイル体系は，ファルの管理を非常に困難にする．また，たとえばファイル・ダイレクトリのエントリの誤った消去は，ファイルをたどる道に出口のないループを作り出す恐れもあるであろう[5]．これらは，すべてファイルとボリュームという明確に分離されたふたつの概念を使って体系を記述すべきであるところを，ファイルというひとつの概念だけで無理に組みたてようとするためであるとわれわれは考える．

上述のボリューム体系にあっては，あるボリュームが任意のファイルを共有する際には，そのファイルのためのボリュームをそのボリュームの中に新しく取るという形がとられる．そしてファイルの共有状態はボリューム体系（具体的には写像，$\mu_1, \mu_2, \cdots, \mu_N$）には何ら表われず，写像 m_0, m_1, m_2 においてそれが表示されるのである．写像 m_0, m_1, m_2 にまで通常ユーザが関与することはない．

2.3 物理媒体空間

ボリュームは物理媒体空間のある領域が与えられてはじめて物理的に存在するものとなる．そこで以下に物理媒体空間の定義づけをおこなう．

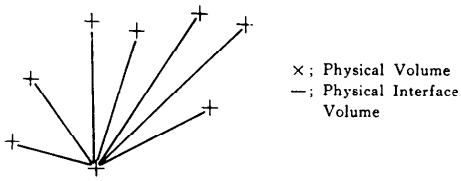
物理媒体空間（Z_p）は，物理的に連続した点の部分集合から構成される．これらの部分集合は，システムのボリュームの論理体系とは無関係にシステムが構成されるハードウェアから定められるものであり，その意味でこれらを Physical Volume (P) と名付ける（それに対して，ボリューム空間（Z_v）内に定義されるボリュームは Logical Volume と名付けられるべきものであるが，特にことわらない限り，単にボリュームと呼ぶときは，常に Logical Volume を意味する）．Physical Volume は任意に細かく定義されることができるが，それをどれだけ細かく定めるかは Physical Volume を管理するシステムの機能（物理的に連続な点の解釈の細かさ）に依存する．たとえば，計算機システムを例にとれば，ディスク・パック，磁気テープ・リール，コア・メモリ，ドラム，タイプライタ，チャネル，CPU などの物理的に閉じた単位のそれぞれが通常ひとつの Physical Volume として扱われる．しかし，たとえばコア・メモリをひとつ例にとってみても，それをメモリ・バンクの集合としてとらえ，メモリ・バンク単位の管理をおこなうシステムがあるとすればコア・メモリ全体でなくメモリ・バンクのひとつひとつが Physical Volume とみなされる．

システムが M 個の Physical Volume $P_i (i=1, 2, \cdots, M)$ から構成されるとき，その情報の世界の任意のボリューム V と P_i とは次式で結合される．

$$m_1(V) = \bigcup_{i=1}^{M} p_i, \quad p_i \subset P_i, \quad P_n \cap P_m = \phi (n \neq m)$$

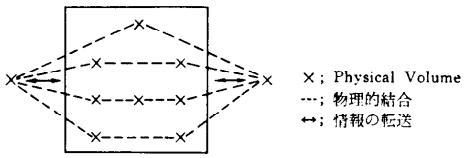
ただし，p_i は空集合も含めた P_i のひとつの部分集合である．

ふたつの Physical Volume を結合する物理的な道は一般に複数個存在する．しかし両 Physical Volume の間に論理的なひとつのボリュームを仮想し，そのボリュームにより Physical Volume が結合されているとみなすことができる．このボリュームを Physical Interface Volume と名付ける．すべての Physical Volume の間にはそれぞれの組に固有の Physical Interface Volume が定義される．そしてシステム中の Physical Volume の物理的結合体系は，ひとつの Physical Volume を選び（たとえばコア・メモリ），それと他の Physical Volume を結ぶすべての Physical Interface Volume により記述される（**第5図**）．情報の転送の立場から同一 Physical Volume 間にも Physical Interface Volume を定義するとすれば，情報の転送は常に一方の Physical Volume から Physical Interface Volume に転送されてのち，その Physical



第5図 Physical Volume の結合体系

Interface Volume から他方の Physical Volume に転送されるとして一般化される（**第6図**）.



第6図 Physical Volume 間の情報の転送

たとえば，ひとつの入出力ユニットが複数のチャネルに結合されている場合，コア・メモリとその入出力ユニットとの間の情報の転送の物理的な道はチャネルの数だけ存在する．この場合，両者の間に定義される論理的なチャネル (Logical Channel[6]) が Physical Interface Volume である．また，ディスク・パックや磁気テープ・リールのようにユニットから取りはずしたり，別のユニットにつけ換えることができる Physical Volume (Removable Volume) を考えるとき，Removable Volume とコア・メモリの間にはその Removable Volume を取りつけることができるユニットと，おのおののユニットに結合されているチャネルの数の組み合わせだけの物理的な道が定められているといえる．この場合，これらのチャネルとユニットを包含してひとつの Physical Interface Volume が定義されるであろう．この節は説明の便宜上，たとえそのものはすべて計算機システムから引用しているが，物理的世界をその物理的要素のいかんにかかわらず，Physical Volume と Physical Interface Volume という一般的要素で組みたてることを試みたものである．

3. 情報処理作業

情報を処理する作業の手順を Procedure と名付けるとすれば，情報処理作業はどのような Procedure が，どのようなファイルに作用されるかが決められて一意的に定義される．しかし，現実にその作業が実行されるためには，Procedure 自身がひとつのファイルとして存在し，Procedure を実行するものがあって，対象とするファイルならびに Procedure が作用されたのちに得られるファイルが存在すべき場が用意されていなければならない．すなわち，情報処理作業はひとつのボリュームの世界の中でおこなわれる．このボリュームを Information Processing ボリューム (IP ボリュームと略記) と名付ける．

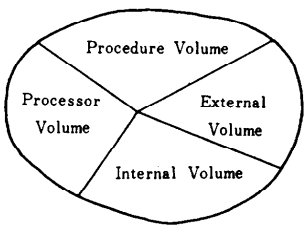
ひとつの情報処理システム（たとえば，計算機システム）に着目する場合，その世界全体がひとつの IP ボリュームである．また，その管理の下でおこなわれる任意の情報処理作業に対しても，ひとつの IP ボリュームが割り当てられる．換言すれば，情報処理作業とは，それに対してひとつの IP ボリュームを割りつけ，その Status を求める結果となる Status に変化させる作業である，と定義することができる．一方，またひとつの情報処理システム中の任意のボリュームは，常にそのシステム全体を表わす IP ボリュームを構成するひとつのボリュームとして，IP ボリューム内に位置づけられる．

IP ボリュームの概念は，いわゆる Logical Processor と Address Space の概念を包含するものであると同時に，Processor もデータ・ファイルも区別せず，すべてを均質な回帰的ボリューム体系の中の一構成要素として扱うところに，われわれが定義するシステムの特徴があり，システムの記述をより一般性に富むものとしている．

3.1 IP ボリュームの構成

Procedure を具体的に表現する情報の集合を Procedure File と名付ける．また，Procedure が処理の対象とするファイルを Object File と名付ける．実際は，しばしば Procedure File 自身が Object File の中に含まれるであろう．

IP ボリュームは**第7図**に示すように4種類のボリュームから構成される．当然のことながら，各ボリュームは一般にはまた何層もの，何種類ものボリュームから構成される．これらのボリュームの情報処理作業



第7図 IP ボリュームの構成

における役割は以下のとおりである.

(1) Processor Volume; Procedure を実行する.

(2) Procedure Volume; Procedure File を保持する.

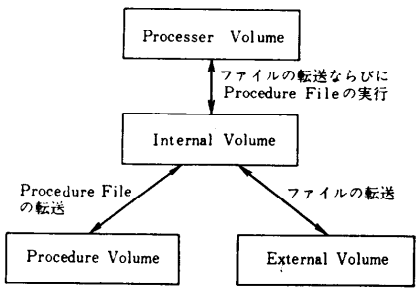
(3) Internal Volume; Processor Volume がファイルを処理するための「まないた」の役割を果す.

(4) External Volume; Object File を保持する.

すなわち, ひとつの情報処理作業に対応する IP ボリュームを生成するためには, そのProcedureを与える Procedure File と, その Procedure が処理の対象とする Object File をそれぞれ保持する Procedure Volume ならびに External Volume がまず選ばれなければならない. 次に, その Procedure File が与える Procedure を実行することができる能力を持つ Processor Volume を割りつけなければならない. 各 Processor Volume は, それが直接ファイルをアクセス (読み, 書き, 実行) できる世界を持つ. それが Internal Volume である. Procedure Volume ならびにExternal Volume 中のファイルは Internal Volume を介してのみ実行され処理されることができる. これらのボリューム間の機能結合を**第8図**に示す.

Processor Volume と Internal Volume の関係はたとえば, 通常の計算機のCPUを Processor Volume とすれば, その Internal Volume はコア・メモリ中の領域であろう. しかしセグメントによる二次元アドレス空間を対象とする Processor Volume においては, コア・メモリからドラム, ディスクにわたって定義される Virtual Memory 内のひとつの領域が Internal Volume となる. また, ひとつの計算機システム自身が Procedure Volume の場合, その計算機システムがアクセスできるすべてのボリュームがその Internal Volume を与える.

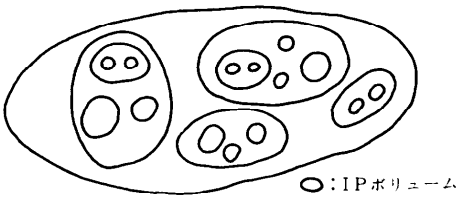
IP ボリュームは, その情報処理の過程において,



第8図　IP ボリュームの内部結合

何から何まですべてをそれ自身が処理する必要はない. その仕事を論理的に閉じたいくつかの単位に分解することができる場合, それぞれの単位の情報処理作業に対して IP ボリュームを割りつけ, 自らはこれらの IP ボリュームが独立におこなう作業を監視するだけで良い.

たとえば, 計算機システムという IP ボリュームがあって, そのシステムを多くのユーザ (Principal[12]) が共有する場合には, 各ユーザごとに IP ボリュームが与えられる. また, ひとつのユーザの IP ボリュームに着目するとき, それは Job の IP ボリュームに分解され, Job は Job Step に, Job Step は Task に, Task はさらに下位の Task (Sub-Task) に, と繰り返えし小単位の IP ボリュームに分解されていくであろう. すなわち, IP ボリュームはそれ自身が IP ボリュームの回帰構造で構成されるものであり, この回帰構造が情報処理システムの論理体系を与える (**第9図**).



第9図　情報処理システムの論理体系

IP ボリュームの直接下位の IP ボリュームは, その External Volume を構成するひとつのボリュームである. したがって, それは通常の External Volume と同様, これらの下位の IP ボリュームを任意に生成, 消去することができる.

IP ボリュームの構造に関して, 特に注意しなければならないことは, IP ボリュームといえども前章で説明がなされたボリュームのひとつにすぎないことである. これを補足すれば IP ボリュームを構成するボリューム同志は論理的に全く異質であるとしても, そのしめる物理媒体空間上の領域が完全に同一であることもあり (たとえば, Procedure Volume と Internal Volume が Physical File Sharing の関係にある場合. この場合 Procedure を実行するに際して Procedure Volume から Internal Volume に Procedure File を転送することは全く論理的なものであって, 物理的には何らの転送もされない), 異なる IP ボリュームが数多く存在するとしても, 実際上はその大部分がハード

ウェアを共有し合っているということである.

3.2 情報処理の作業形態

すでに述べたように，情報処理作業は IP ボリュームの回帰的実行によりおこなわれる．ひとつの IP ボリュームは，それ単独でひとつの仕事をおこなうことができるが，それらの IP ボリュームが集ってより大きな情報処理作業を実現できるためには，IP ボリューム間の情報伝達の手段とその上下（支配，被支配）関係が確立されていなければならない．後者はボリュームの包含関係の論理体系から規定される．前者の手段にはファイルの共有による方式と System Interrupt による方式の2種類がある.

ファイルの共有による情報伝達は，情報を伝達しあうIPボリュームがそれぞれ互にPhysical File Sharing の関係にあるボリュームを保持することにより実現される．この場合，一方の IP ボリュームが自らの情報伝達用のボリュームに伝達したいファイルを生成すれば，他方の IP ボリュームは自らの情報伝達用のボリュームからそのファイルを得ることができる.

System Interrupt による方式では，Communication は一方の IP ボリュームから他方の IP ボリュームの Processor Volume に対して，特定の割り込みをかけることによりおこなわれる．割り込み時に同時にある種の情報を伝達したい場合には，上述の場合と同様 Physical File Sharing のボリュームが使われる.

情報処理作業にとって，次に考慮しなければならない問題は IP ボリュームの State である．IP ボリュームの State は，その Procedure の実行の状態から定められるもので，IP ボリュームは常に次の2種類のいずれかの State にある.

（1） Running State

（2） Blocked State

Running State は，現在その IP ボリュームの Procedure が実行されつつある状態である．Blocked State は何らかの理由により，そのIPボリュームがRunning State となり得ない状態である．Blocked State は次の2種類に分けられる.

（1） Time Blocked

（2） Space Blocked

Time Blocked State とは，独立にプロセスされている他の IP ボリュームと何らかの時間的同期をとる必要が生じた場合である．たとえば，入出力動作の完了を待つ場合，Sub-Task の完了を待つ場合，特定の時間の経過を待つ場合，あるいは他の IP ボリュームにより排他的に使用されている Physical File Sharing のボリュームを使用しようとした場合，などである.

Space Blocked State とは，IP ボリュームが Running State を維持するために必要とする物理媒体空間内の領域が，Time Blocked 以外の原因により得られない場合である．Space Blocked は物理媒体空間が十分大きければ起り得ないものであるのに対し，Time Blocked は避けえない本質的原因に基づいている.

一方，任意の時点をとるとき，その時点に存在する IP ボリュームのすべてにわたって何らかの形の優先順位（Priority）が定められていると仮定する（この Priority が，どのような Algorithm で決められるものであるかは，ここでは問題としない[7]）．それは，すなわち IP ボリューム間で物理媒体空間内の特定の領域の競合使用要求が起るとき，その領域がどちらに優先的に与えられるかを定める．しかし，互に要求するボリュームが Physical File Sharing の関係にあって，かつ排他的な使用要求の場合には Priority のいかんにかかわらず，先に出された要求が優先される．この例外の場合を除けば，低い Priority のものがすでに他の高い Priority の IP ボリュームにより使用されている領域を使えるのは，現在使用中の IP ボリュームが Blocked State に陥るか，それがシステムから消去（Quit）されてしまった場合に限られる．また，低い Priority のものがすでに占有している領域の使用を高い Priority のものが要求するとき，Priority が低い方の IP ボリュームは Running State にあるときは Blocked State に，すでに Blocked State にあるときはより強い Blocked State に陥り，その領域は Priority が高い方の IP ボリュームに使用が委ねられる．そしてその領域を保持していた Priority が低い方の IP ボリューム中のボリュームの Status は，それより低レベルの物理媒体空間にはき出される．これを Roll-Out と名付ける．一方，低レベルの物理媒体空間にはき出されたボリュームの Status は，そのボリュームが属する IP ボリュームが Running State に戻るときに，高レベルの物理媒体空間中に戻される．これを Roll-In と名付ける．このような Roll-Out, Roll-in の機能により，IP ボリュームは高い Priority の IP ボリュームにより，その一部の領域が奪われても必らずもとの状態に復元されることができる.

3.3 System Interrupt

一方の IP ボリュームから他方の IP ボリュームへの一切の割り込みを System Interrupt と名付ける.

したがって，Hardware Interrupt もその中に含まれることは当然であるが，ここではあくまで Interrupt をその持つ論理的意味の面からとらえる．

System Interrupt は次の2種類に大別される[2]．

（1） Block Interrupt

（2） Wake-Up Interrupt

Block Interrupt は，Running IP ボリュームに対してそれを Blocked State に変え，Blocked IP ボリュームに対してはそれをさらに強い Blocked State にならしめる．これは上位の IP ボリュームから実行を強制的に停止させるために下位の IP ボリュームに対して与えられる (Suspend) か，高い Priority の IP ボリュームが領域の解放を要求して低い Priority の IP ボリュームに与える (Roll-Out)．

Wake-Up Interrupt は，Running IP ボリュームに対しては何の効果も及ぼさないが，Blocked IP ボリュームに対してはそれを Running State に変えるか，それができなければそれをより弱い Blocked State に変わらせしめる．これは上位の IP ボリュームから Suspend されている IP ボリュームの実行を再開させる (Restart) 目的で与えられるか，特定の IP ボリュームから Time Blocked State にある IP ボリュームに対して同期信号として与えられる (Synchronization)．

3.4 ボリュームの生成と消去

ボリュームの生成ならびにその消去は，それが包含される IP ボリュームによりおこなわれるものであり，その IP ボリューム自身もひとつのボリュームであって，その上位の IP ボリュームにより生成，消去されるとすれば，ボリュームの生成と消去の問題は結局その根元となる IP ボリュームがいかにして生成，消去されるかという問題にいきつくであろう．計算機システムではその根元のものは一般にオペレーティング・システムの名で呼ばれているものである．

われわれは，この機構を記述するために IP ボリュームに核の概念を導入する．これはすべての IP ボリュームに対して常にひとつあって，IP Kernel Volume と名付けられる．IP Kernel Volume は，Processor Volume の一部分であって，System Interrupt の受けつけと自分自身の IP ボリュームを生成する（厳密には IP Kernel Volume を除く）機能並びに Roll-In 機能を保持する．IP Kernel Volume 自身は，その上位の IP ボリュームにより生成，消去され，かつ起動される．したがって IP Kernel Volume が生成されるためには，順に上位のレベルの IP ボリュームにさか昇らなければならないが，最終的にオペレーティング・システムの IP Kernel Volume 自身は，たとえば人間自身が IP Kernel Volume となったより大きな IP ボリュームにより，生成，消去，起動されると考えられる．

ここでボリュームの生成を Open，消去を Close と名付ける．特に IP Kernel Volume に対しては，その Open は同時にその起動をおこなう．ボリュームの Open には次の3種類の方式がある．

（1） 既存のボリュームと Physical File Sharing の関係にあるボリュームを生成する．

（2） 既存のボリュームと Logical File Sharing の関係にあるボリュームを生成する．

（3） 既存のボリュームと関係なく全く新しいボリュームを生成する．

同様にボリュームの Close には次の3種類の方式がある．

（1） ボリュームの消去をおこなうと同時に，上位の任意のランクの IP ボリュームに対して，そのボリュームを登録する (Catalog) ことを要求する．実際は，そのボリュームと Physical File Sharing あるいは Logical File Sharing の関係にあるボリュームを指定された IP ボリュームが Open する．

（2） ボリュームの消去をおこなうと同時に，それが Catalog されている場合，その除去を要求する．実際はそのボリュームと Physical File Sharing あるいは Logical File Sharing の関係で Catalog されているボリュームを，それを保持する IP ボリュームが Close する．

（3） 単にボリュームの消去のみをおこなう．

特に Close 時にオペレーティング・システムの IP ボリュームに対して Catalog を要求するとき，それが一般に使われている Cataloging[6] である．

IP Kernel Volume には，その生成時にこれを生成した上位の IP ボリュームから，その IP ボリュームに許された物理的ならびに論理的世界の大きさに関する情報 (Capability[1]) が与えられている．したがって Processor Volume は任意にその External Volume ならびに IP ボリューム自身を Open することができるが，決して与えられている Capability の範囲を越えたボリュームの Open は許されない．Capability の変更は，その IP ボリュームを生成した上位の IP ボリュームに対して要求し，それが許可されたときのみ

可能である．また IP Kernel Volume は，一度生成されたあとは決して Roll-Out されることはない．そして Roll-Out されたその IP ボリューム中のボリュームの Status をあとに Roll-In して，IP ボリュームを中断状態から再開できるための情報も保持する．最終的に IP ボリュームが Close される段になって，IP Kernel Volume 自身も消去される．

3.5 Processor Volume

Processor Volume の持つべき機能は，Procedure を実行することにあるが，その機能はより詳細には以下のように列挙されるものである．

（1） Procedure の実行の流れを管理する (Procedure File を Procedure Volume から Internal Volume に転送することをスケジュールする)．

（2） IP ボリュームの Status を変化させる (Procedure の Instruction を逐一解読し実行することにより，IP ボリューム内のファイルの情報を転送あるいは更新する)

（3） 自分自身の IP ボリュームを Open あるいは Close する．また Roll-In をおこなう．

（4） 他の IP ボリュームに対して System Interrupt を発生する．

（5） 他の IP ボリュームからその IP ボリュームに対して出された System Interrupt を受けつける．

（6） 自分自身の IP ボリュームを Blocked State に落す(ボリュームの Roll-Out が要求される場合はその仕事を受け持つ)．

（7） 下位の IP ボリュームあるいは任意の External Volume を Open あるいは Close する（下位の IP ボリュームの Open は厳密には IP Kernel Volume の Open である)．

Processor Volume の最も基礎となる形態は，ハードウェアそのものである．しかし一般には Processor Volume それ自身がひとつの IP ボリュームである．これはすなわち，既存の IP ボリュームを Processor Volume としてより高度な機能を持つ IP ボリュームを生成し，それをまた Processor Volume として，さらに高度な機能を持つ IP ボリュームを作り出すというようにして，機能を次から次に拡張していくことができることを意味している．たとえば，Processor Volume としては以下に示すものがある．

（1） 計算機の Instruction を実行する CPU

（2） 入出力コマンドを実行するチャネル

（3） 計算機の Instruction の実行とその流れを管理する Supervisor (CPU も含める)

（4） Job Control Statement のシーケンスを解読，実行する Job Scheduler

（5） 異なる計算機の言語で書かれた Prccedure を実行する Machine Simulator

4. むすび

従来，計算機システムを設計するに際しては Device Independent，さらには Configuration Free ということが強調されてきた．しかし，狭い計算機システムの概念を脱皮し，ひとつの大きな情報処理システムの一要素として調和した計算機システムを志向しようとする場合には，それでは不十分であって，さらに進んで Information System Free でなければならないと考える．われわれは，この論文における考察が，そのような計算機システムの設計のための有効な武器となることを信ずるものである．

最後に，この考察に対して適切なる御助言をいただいた当所嶋村課長，首藤，中島，黒田の各氏および計算機システム技術部の諸氏に謝意を表します．

参考文献

1) J. B. Dennis and E. C. Van Horn: Programming Semantics for Multiprogrammed Computations, Comm. ACM **9**, 3 (Mar. 1966).
2) J. H. Saltzer: Traffic Control in a Multiplexed Computer System, MAC-TR-30 (Thesis), M. I. T., Cambridge, Mass., July 1966.
3) B. W. Lampson: A Scheduling Philosophy for Multiprocessing Systems, Comm. ACM **11**, 5 (May 1968).
4) R. V. Head: Management Information Systems: A Critical Appraisal, Datamation **13**, 5 (May 1967).
5) R. C. Daley and P. G. Neumann: A General Purpose File System for Secondary Storage, Pro. FJCC, 1965.
6) W. A. Clark: Functional Structure of OS/360 Part III, Data Management, IBM Systems Journal **5**, 1 (1966).
7) P. J. Denning: Resource Allocation in Multiprocess Computer Systems, MAC-TR-50 (Thesis), M. I. T., Cambridge, Mass., May 1968.

（昭和 44 年 11 月 13 日受付）