プログラムおよびデータの特性指標とその仮想記憶システムの特性への影響†

大須賀 節雄††

計算機システムの特性を把握することはシステム設計者にとって極めて重要なことであることは言うまでもない．特に計算機利用技術が多様化しつつある現在，利用形態がシステム性能に及ぼす影響を正しく予測し，それに対応することが必要である．このために，計算機システムのモデル化とそれによる評価・解析に関する研究は，これまでも数多く行われてきた．しかしこの多くはシステム動作の時間的推移を，時間軸上で現在時点を中心に局所的に見た状況によって表現するため，システムの全体的な特性を把握するには必ずしも十分なモデルとなっていない．

本論文ではプログラムの動作をページ遷移系列の確率過程とし記述することにより，全体的なプログラム動特性を表わすモデルが得られること，これをシステム全体の動作を表わす待合せモデルと組合わせることにより，応用プログラムの性質によるシステムの性能変化を調べることができること，さらにプログラムにおけるデータの増大により，システム性能が低下する可能性があり，今後のシステム設計に際してはシステムアーキテクチャの再検討が必要となることなどを示す．

1. はじめに

計算機システムの性能は(1)処理される情報の物理的性質（プログラムやデータの大きさ，アクセス方法など），(2)システムのアーキテクチャ（各リソースの配置，CPU 処理速度，主メモリ量，I/O 処理速度など），(3) OS の制御アルゴリズム に依存する．このうち(1)の情報の性質は処理目的に応じてユーザが作成するプログラムにより決定される．したがって，システム設計者には，(1)に関する十分な知識に基づいて，(2)，(3)に関する決定を最適なものにする努力が要求され，それには十分なシステム解析が必要である．すなわち，システム解析や評価の一つの，そして最大の目的は設計に役立つ情報を得ることである．そのためにはシステム動作の一般的傾向を把握でき，種々の条件のもとで，(実システムで測定するよりはずっと容易に）動作特性を推定できるモデルを得ることが必要である．

プログラムの動特性を知るために，これまで多くの理論解析や実験が行われてきた．それにもかかわらず，それらによって得られた情報は上述の要求を必ずしも十分に満たしていない．従来はシステム全体を記述する代りに，処理過程内の一点を中心に，その前後の有限の時間帯での現象を数学的に表現し，これによってシステムの特性を表わす方法や[3)-5),8)]，実際のシステムの動作の測定から，システムの一般的傾向を論ずる方法[1),7),13)]などによってシステムの解析が行われていた．

一般論としては，システムが複雑で全体の記述が困難なとき，前者のような方法が効果的な場合もある．たとえば有限オートマトンの表現に対するチューリング機械の表現がこの良い例を示す．しかし，求める結果がシステム特性のように全体としての性質である場合，このような局所の傾向の表現と全体の傾向との関係が明確にされねばならず，従来の方法はこの点が不十分である．

一方，実験的に得られる情報の多くは，(必ずしも最適の保証のない) OS の制御アルゴリズムその他の影響を受けることが多い．現在，システム解析において最も必要なのは，システム特性に影響する諸要因をできるだけ純粋な形で把握することであり，OS などの影響を受けたデータからこの情報を分離することは難しい．

本論文では，このような点を考慮して，システム全体の動作モデルを作成する．この方法はシステム動作を確率過程と見なし，各プログラムのページ参照過程を遷移マトリックスで表わすものである．プログラムの進行に伴って，命令語の参照とデータ語への参照が別個に行われるので，この両方のページの参照過程を同時に表わすモデルを作る．このため遷移マトリック

† Characteristic Indices of Program and Data and their Effects upon the Behavior of Multiprogrammed Virtual Storage Systems by SETSUO OHSUGA (Institute of Space and Aeronautical Science, University of Tokyo).

†† 東京大学宇宙航究研究所

スは大型のものになるが，一定の近似のもとで解析解が得られる．以下ではユーザのプログラムの中で，プログラムのページとデータのページを区別し，それぞれのページ参照をプログラム参照，データ参照と呼ぶ．またこの全体をユーザ・プロセスまたは単にプロセスという．

本論文の主たる目的はこのようを方法により仮想メモリ・システムのモデル化の方法を与えることであり，このモデルを用いた解析により，(1)広い範囲にわたるシステムの諸特性を表現することができ，この分析によりシステムのアーキテクチャや制御アルゴリズムを最適なものにするために役立つ情報が得られることが示されるが，さらに，(2)プログラムとデータ特性が明白に相違すること，システム動特性に及ぼす影響はこの両者で大幅に異なり，特にデータの特性の影響が大であること，今後のメモリの低廉化に伴ってデータ利用の技術が急速に発達し，参照されるデータ量が飛躍的に増大することが予想されるが，これはシステム性能の大幅な低下をもたらす可能性があることなどが示される．仮想メモリ方式はユーザに大きな論理空間を与える点に意義があり，この効用は特に大型のデータ構造を扱う際に大であることが期待されるにもかかわらず，この場合のシステム性能が大幅に低下することは仮想メモリ・システムの設計理念における矛盾ともいえるもので，この解決が早急に必要になることと予想される．以下では，まずシステムの効率に関連するパラメータをシステムの平衡条件から求め，次いでこのうちの重要なパラメータ値を得るためにプログラムの動作特性を表わすモデル化を行う．

2. システム性能に影響する要素

以下では複数のユーザを擁する仮想メモリ・システムを想定する．システム内でアクティブなユーザの処理プロセスの集合を $\mathbf{P}$ とする．この各プロセス P_i について，所要ページのうち一部 (W_i) が主メモリに準備されていて，いつでも実行可能な状態にあるとき，"ready" の状態にあるという．このようなプロセス P_i の実行中，W_i 以外のページを参照したとき，ページ割込みが生じて処理が中断し，要求したページの読込み要求が出され，他のプロセスに制御が移される．ユーザ・プロセスから I/O 要求が発せられた場合も同様である．ready 状態のプロセスと I/O 待ちの状態のプロセスの集合を $\mathbf{P}_r=\{P_i/i=1,2,\cdots,r\}$ とする．

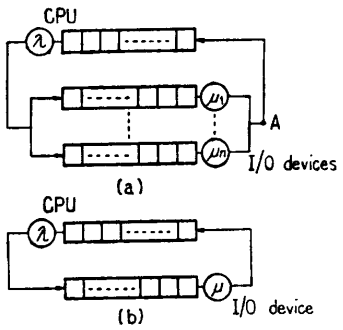

図 1 システムの待合せモデル
Fig. 1 Queuing model of the system.

あるプロセス P_i の実行開始後，ページまたは I/O 要求が生じるまでを 1 パス，またこの時間長を寿命と呼び，L_i で表わす．要求ページが読込まれた時点または I/O 終了時点でこのプロセスは再び ready になる．以下では割込みに伴う OS 動作時間は I/O 時間に含めて考える．

以上の状況は，**図 1**(a)のように循環型の待合せ系モデルで表わされるが，紙数の都合で本稿では I/O 待合せ系を同図(b)のような一つの等価待合せ列および窓口に置きかえてシステムの大局的な特性を調べるにとどめ，複数 I/O を含む系については別稿で扱うことにする．この等価待合せ系のサービス窓口の処理時間分布は図 1(a)の A 点で観察したとき，いずれかの I/O 装置での処理を終了し，CPU の待合せに戻るプロセス間の相互の時間間隔である．この時間は確率変数であり，その分布は入出力装置の性質や数，全プロセス数などにより変化し，厳密に定めることは困難である．しかしこの分布形状を厳密に求めることはあまり重要ではない．後に述べるように，一定の条件のもとで分布形状の相違によるシステム特性の変化は比較的狭い範囲に入り，平均処理率以外は結論に影響を与える程のものではないからである．このことは CPU 待合せ系についても同様である．

この仮想的な I/O 装置の単位時間あたりの平均処理率を μ とする．同様に，CPU 系の平均処理率を λ とする．ここで粗い近似として，CPU および仮想 I/O の処理時間分布をそれぞれパラメータ λ および μ の指数分布としてみる．このとき CPU 系の待合せ数を処理中のものも含めて数えて i の確率を p_i とする．p_0 が CPU 遊休確率を表わす．p_i は待合せ計算から求まり

$$p_i=(1-\rho)\rho^i/(1-\rho^{r+1}) \qquad (2.1)$$

である．ここで $\rho=\mu/\lambda$，r はこの循環系内に存在す

るプロセスの数である．r は各プロセスごとに主メモリに準備されるページ数を $|W_i|$ とし，主メモリのユーザ用領域の容量を M とするとき，$\sum_{i=1}^{r}|W_i|\leq M$ なる最大の整数 r とする．

各プロセスはそれぞれ固有の resource 利用要求を有している．この際，CPU 以外の resource を要求しても必ずその後に CPU 処理を受け，CPU 要求がすべて満たされた時点で処理終了となる．したがって CPU の効率はシステムの through put および平均応答時間の両方に関わる．このことから CPU の稼動率 $\eta_r=1-p_0$ をシステムの性能指標とする．

$$\eta_r=\begin{cases}\rho(1-\rho^r)/(1-\rho^{r+1}), & \rho<1\\ r/(r+1), & \rho=1\\ \rho(\rho^r-1)/(\rho^{r+1}-1), & \rho>1\end{cases}\qquad(2.2)$$

r にたいして η_r を求めたのが**図2**である．$r\to\infty$ における値を η_∞ とすると

$$\eta_\infty=\begin{cases}\rho \text{ if } \rho<1\\ 1 \text{ if } \rho\geq 1\end{cases}\qquad(2.3)$$

である．r の増大に伴う $\eta\to\eta_\infty$ の漸近特性は CPU および I/O の処理時間分布に依存する．しかし，ここで次の諸点に留意する．

(i) η_1 ($r=1$) は分布によらず ρ のみに依存する．(ii) ρ 固定したとき，η_r は r の単調増加関数であり，(iii) この増加率は各分布の(標準偏差/平均値) の値が小なる程，大である（たとえば両方の分布の分散が0で，かつ $\rho\geq 1$ のとき，$\eta_2=1$ である)．

上記(iii)は可能なすべての分布について理論的に証明されている訳ではないので，ここでは経験則として用いる．処理時間が非負であるから，処理時間の，可能なすべての分布関数の集合を考えたとき，その大部分は偏差/平均値<1 である．指数分布はこの値が1であり，この意味で図2の実線は物理的に可能な分布の中で最も効率の低いものもクラスに入る*．このことから，たとえば $\rho=1.2$ の場合，実際上生ずるのは図2で斜線領域に入るものと考えられる．この領域は比較的狭く，これらのことから，任意の分布について以下の結論が導かれる．すなわち，システム効率 η_r を大にするためには，(1) ρ を大にすること，および (2) r を大にすること が必要である．(1)に関しては $\rho=\mu/\lambda$ より (1-1) μ と大にすること (1-2) λ を小にすること が導かれる．μ^{-1}, λ^{-1} がそれぞれ平均I/O 処理時間，平均寿命に相当するから，効率の向上には処率速度の早いI/O装置を用い，かつ各プロセスごとに平均寿命 L_i を大にする努力が必要である．

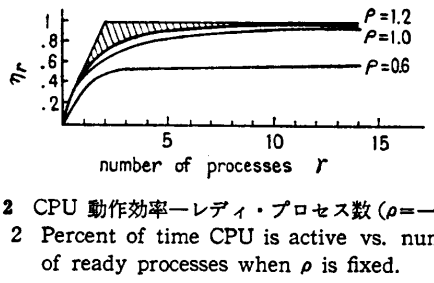

図 2 CPU 動作効率—レディ・プロセス数 (ρ=一定)
Fig. 2 Percent of time CPU is active vs. number of ready processes when ρ is fixed.

* 偏差/平均値=1 となるもう一つの例として，時間0および $2a$ をそれぞれ確率 1/2 でとる分布がある．

多重度 r を大にするためメモリ容量 M は十分大でなければならない．しかし，η_r の漸近特性から，r の増大による効果は急速に減少する．各プロセスごとに，$\rho_i\gg 1$ の条件が満たされていれば r はたかだか5〜6でも十分である（図2参照)．M が十分大でないときに r を大にすると，$\rho<1$ となり，スラッシングと呼ばれる η_r の急落の現象が生ずる．

以上，システム効率に影響する要因のうち，M と μ^{-1} はシステムの構成や装置の特性から求まるが，寿命 L_i は多くの要因の複合であり，以下，これをモデルにより求める．

3. プログラムの状態遷移モデル

3.1 ページ遷移に基づく状態遷移モデル

以下では CPU で実行中のプロセス P_a に注目する．P_a に関する諸量に添字 a を付して表わす．P_a の CPU による実行により主メモリ内のページが順次参照されるが，このページ参照過程をマルコフ過程と考える．なお2章のモデルに対応し，I/O 要求，ページ要求をまとめ，すべてを要求として扱う．

n ページのプロセス P_a で，ページ i から j への遷移確率を p_{ij} とする．$p_{ij}\geq 0$，$\sum_{j=1}^{n}p_{ij}=1$, ($i=1,\cdots,n$) である．p_{ii} は同一ページ内の語に二回続けてアクセスする確率である．このとき，次の遷移マトリックス，T_a を定義する．主メモリ内にあるページ集合を $W_a=\{P_1,P_2,\cdots,P_k\}$ とする．

$$T_a=\begin{bmatrix}p_{11}\cdots p_{1k} & p_{1k+1}\cdots p_{1n}\\ \vdots & \vdots\\ p_{k1}\cdots p_{kk} & p_{kk+1}\cdots p_{kn}\\ \vdots & \vdots\\ p_{n1} & p_{nk} \quad p_{nk+1}\cdots p_{nn}\end{bmatrix}\qquad(3.1)$$

N 回の遷移後，ページ i にある確率を $\pi_i(N)$，これを i 要素とするベクトルを $\pi(N)=(\pi_1(N),\pi_2(N),\cdots,\pi_n(N))$ とする．同様に初期状態ベクトルを $\pi(0)$ で定義する．すると，

$$\pi(N)=\pi(0)\cdot T_a^N \tag{3.2}$$

である.

実行開始後, N 回の遷移がすべて W_a 内のページで行われる確率を求めるには, 新しい遷移マトリックス T_a^* を定義する.

$$T_a^*=\left[\begin{array}{c|c} \begin{matrix} p_{11}\cdots p_{1k} \\ \vdots \quad \vdots \\ p_{k1}\cdots p_{kk} \end{matrix} & \begin{matrix} p_{1k+1}\cdots p_{1n} \\ \vdots \\ p_{kk+1}\cdots p_{kn} \end{matrix} \\ \hline 0 & I \end{array}\right] \tag{3.3}$$

0; 全要素0のマトリックス

I; 単位マトリックス

これに対応して, N 回遷移後の状態確率ベクトルを $\pi^*(N)$ とすると,

$$\pi^*(N)=\pi(0)\cdot T_a^{*N} \tag{3.4}$$

が得られる.

さらに, N 回の遷移後, W_a 内のどれかのページにある確率を $\hat{P}_a(N)$ とすると

$$\hat{P}_a(N)=\pi^*(N)\cdot K=\pi(0)\cdot T_a^{*N}\cdot K \tag{3.5}$$

となる. ここで K は $1\sim k$ 要素が1, 他は0なる縦ベクトルである. これより, 1ステップの実行時間 h を用いると $Nh=t$ であるから, 時間 t の後に W_a にある確率も (3.5) 式から求まる. 以下ではすべて時間をステップ数を1単位として表わし, h を省略する. 丁度 N 回目までは W_a 内で遷移を続け, $N+1$ 回目で W_a の外に出る確率は

$$\hat{p}_a(N)=\hat{P}_a(N)-\hat{P}_a(N+1) \tag{3.6}$$

である.

以上の結果, L_a が次のように得られる.

$$\begin{aligned} L_a=\bar{N}&=\sum_{N=1}^{\infty}N\hat{p}_a(N)=\sum_{N=1}^{\infty}\hat{P}_a(N) \\ &=\sum_{N=1}^{\infty}\pi^*(N)\cdot K=\sum_{N=1}^{\infty}\pi(0)\cdot\hat{T}_a^{*N}\cdot K' \\ &=\pi(0)\cdot\hat{T}_a^*\cdot(I-\hat{T}_a^*)^{-1}\cdot K' \end{aligned} \tag{3.7}$$

I は単位マトリックス, K' はすべての要素が1の k 次の列ベクトルである. また, $\hat{T}_a^*$ は T_a^* のうち, W_a に入る要素のみから成る小行列である.

3.2 プログラムとデータの特性

遷移マトリックスは情報参照シーケンスを表わしており, これによりプログラム動作の特性が定まるから特性指標として用いることができる. 以下ではこれがプログラムとデータで非常に異なることを示す.

3.2.1 プログラム・ページ参照における寿命

プログラムにおける命令語の参照の順序により, プログラムの動的な性質が表わされる. 命令語の参照のみに注目した場合, 通常, アクセスが連続的に行われるという連続性あるいは局所参照性により, $p_{ii}(i=1,\cdots,n)$ が一般の $p_{ij}(i,j=1,\cdots,n,i\neq j)$ より大なることが予想される. p_{ij} を直接的に示すデータが十分でないので, 以下では既存のデータを手がかりに p_{ii} の値を推定してみる.

一つは, 少し古いが, Gibson Mix のもとになっている命令語の使用頻度からの推定である. このデータは全命令のうち無条件ジャンプ命令が 0.175, 条件付きジャンプ命令が 0.065 の割合を占めていることを示している. 条件付きジャンプのうち, 実際にジャンプ条件が成立するのがこの半分とみると, ジャンプの生ずるのが全体の 20% であり, 残りが逐次的に実行される命令である. 各ページで最後の命令はジャンプ命令以外でもページ遷移が生ずるが, 頻度としては少ないので, ジャンプによる遷移に含めて扱う.

ジャンプ命令でも同一ページ内遷移の場合が多いのでこの割合を f とすると, $p_{ii}\fallingdotseq 0.8+0.2f$ である.

仮想メモリ方式は, 事実, このプログラムの局所性, すなわち, 「ページ参照確率が平均的でなく, 参照される部分が限定されかつプログラム実行に伴って移動する」という性質のもとで成り立っている. この性質が存在するとき, 総ページのある一部を主メモリに準備することにより十分大きな寿命を得ることができ, Working Set の概念が有効になる.

今あるページ集合 W_a^+ を考え, この範囲内ですべての遷移が生ずるものとする. この集合の大きさを n_p とする. ここで次のように遷移確率分布を近似する.

$$p_{ii}=\alpha$$

$$p_{ij}=\begin{cases}(1-\alpha)/(n_p-1) & \text{if } j\in W_a^+, j\neq i \\ 0 & \text{if } j\notin W_a^+, (j\neq i)\end{cases} \tag{3.8}$$

すると(3.7)式より

$$L_a=\{1/(1-\alpha)\}\cdot\{\alpha+(z_p-1)/(n_p-z_p)\}, \tag{3.9}$$

が得られる. z_p は集合 W_a の大きさであり, W_a は W_a^+ の部分集合とする.

比較的小さなプロセスで, 全プログラム・ページ m_p が W_a^+ に入ると考えよう ($n_p=m_p$). 総ページ50ページ程度 ($m_p=50$) のプログラムで, $z_p=25$ としたとき, 平均的に寿命が100ステップ以上という仮定は決して無理なものではない. このとき, $\alpha=p_{ii}\geq 0.98$ となり, $f\geq 0.9$ が得られる.

3.2.2 データ・ページ参照における寿命

データについては線型配列, ポインタによるリスト型構造, ランダム配置のハッシュ構造など各種の形式

があり，プログラムの場合に比して参照の仕方が多様である．特定の言語内で許されるデータ形式には一定の制約があるが，データの配列に関する約束と参照順序とは原理的には無関係である．この意味でデータはアクセス順序の観点からは規則性の乏しいのが特徴である．したがって，以下ではアクセスに関する規則性の全くない場合をまず解析し，しかる後に規則化について考察する．

m_D ページのデータを考える．遷移確率などに関する諸概念はプログラムの場合と同様とする．アクセス順序に規則性のないデータの最大の特徴は，遷移確率 q_{ij} が状態 i に依らず，総ページ数を m_D とすると一定値 $1/m_D$ となることである．したがって W_a 内でデータ参照の行われる確率は，z_D のみに比例する．以下，この場合の解析を行うと，$q_{ij}=1/m_D$ として

$$T_a^{*N}=\frac{z_D{}^{N-1}}{m_D{}^N}\left[\begin{array}{c|c}\begin{matrix}1\cdots1\\ \vdots\ \ \vdots\\ 1\cdots1\end{matrix} & \begin{matrix}1\cdots1\\ \vdots\ \ \vdots\\ 1\cdots1\end{matrix}\\ \hline 0 & I\end{array}\right] \tag{3.10}$$

$$\hat{T}_a^{*N}\cdot K'=(z_D/m_D)^N\cdot K' \tag{3.11}$$

$$\hat{P}_a(N)=\pi(0)\cdot\hat{T}^{*N}\cdot K'=(z_D/m_D)^N\cdot\pi(0)\cdot K'$$
$$=(z_D/m_D)^N \tag{3.12}$$

$$L_a=\sum_{N=1}^{\infty}\hat{P}_a(N)=\sum_{N=1}^{\infty}(z_D/m_D)^N=z_D/(m_D-z_D) \tag{3.13}$$

が得られる．

例として，プログラムの場合と同じ数値，$m_D=50$, $z_D=25$ を代入すると $L_a=1$ である．

実際にはデータもある程度の構造をもち，それによって参照時の局所性が生じてくる．ただしデータに与えられる構造は主として論理的な構造であり，必ずしもそれが寿命という．物理的な関係から定まる量に反映されるとは限らない．寿命に関係するのは参照の順序と記憶配列との一致の程度である．これが共に線形的で良い一致を示すプログラムと異なり，データの場合にはこの一致が保証されない．データの場合配列方法は一通りであるのに，利用する側からのデータの見方（すなわち参照順序）は多様であり，常にこの一致を図ること自体が不可能である．以下では代表的なデータの形式について特性を調べてみる．

［1］ 線形配列：線形配列ではデータ要素のアドレスが論理的データ順序 i と線形関係 $a+bi$ で対応づけられる．一般にはデータ要素はメモリ内に密に詰められる．アクセスが i の順序で行われる場合には q_{ii} の値が十分大になる．

多次元のアレイ，たとえば (i, j) 要素が $a+(g(i-1)+j-1)b$ の形で与えられる二次元アレイでは，b は1と定めても i に関してデータ要素を参照すると，g 間隔の離散的参照となり，q_{ii} は著しく低い．特にページ・サイズを e としたとき $g>e$ では q_{ii} はほとんど0にまで低下する．このことは多次元アレイ方式は本質的にシステム効率を低下させる作用を含むことを意味する．

［2］ ハッシング：ハッシュ関数も関数を用いてアドレスを算出するという点では［1］と同じタイプのアドレス法といえるが，この性質は線型配列と正反対でランダム化を目的としており，q_{ij} が均一の典型的な場合である．主メモリ以外の記憶を用いる場合，ハッシング法はマルチ・ユーザのもとでシステム全体としての効率を著しく低下させることになる*．

［3］ ポインタ結合によるデータ構造：ポインタによりデータ要素がリンクされるリスト形データ構造は論理的なデータ要素間の関係を物理的な記憶アドレスの関係から分離するものとして用いられている．したがってこの型のデータ構造も基本的には前述のランダム型に近い．ただしデータのアクセス範囲が限定されることは，関係の深いデータを近い位置におくという操作が限定的ではあるが可能であり，q_{ii} を多少増加し得る．これにはクラスタリングなどの手法が有効であろう．

3.3 特性値による情報の型の分類

以上の結果から，情報の各種形式について，W_a の大きさと寿命の長さの関係の一般的な傾向を示したのが**図3**である．プログラム・ページと線形配列型のデ

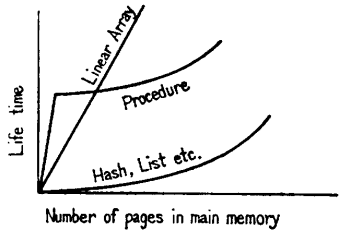

図3 手続きとデータの動特性
Fig. 3 Behavioral characteristics of procedure and data.

* ただしハッシングは内容によるアクセスであり，これと同じ結果を得る方法，たとえば線形ファイル探索によって求めるデータに達する場合との総合的な比較が必要である．線形ファイル探索は局所参照性を有するが目的データに達するのに，ハッシングではハッシュ関数とコリジョンの処理で済む．これまで含めた評価を行うには寿命の概念のみでは不十分で，処理内容に立ち入ったより高度の評価基準が必要になる．

ータ・ページは局所性が大であり，仮想メモリ方式の採用による性能低下は小さい．これに対しハッシュ・コード型やポインタ型のデータ構造は局所性が著しく低く，4章に述べるように寿命が著しく小さくなる．この型を D 型情報と呼び，これに対し前者を P 型情報と呼ぶ．D 型でも，データ・ページの総数が少ないものは全ページ W_a をに含めることができるから，(3.13)式で $m_D=z_D$ となり L_a が大になる．したがって問題となるのは m_D の大きな D 型情報である．データベースなどが仮想メモリ・システム上で実現されるような場合がこれに相当する．

4. プログラムの動特性

4.1 プログラム・シーケンス・モデル

命令の実行は(1)命令の Fetch とその解釈，(2)オペランド Fetch とその操作 という順序で行われる．この(1)と(2)に対応して，プログラム・ページの遷移とデータ・ページの遷移がそれぞれ生ずる．すなわち現実のプログラムの動作は前章のプログラム特性とデータ特性の両方を含むものとして表わされねばならない．

ページ遷移過程において，各状態をプログラム・ページ i とデータ・ページ ξ の複合で (i,ξ) と表わし，$p_{(i\xi)(j\eta)}$ により，状態 (i,ξ) から (j,η) への遷移確率を表わす．以下ではプログラム・ページの遷移とデータ・ページの遷移間の関連は少ないと仮定し，$p_{(i\xi)(j\eta)}$ をそれぞれプログラムおよびデータの遷移確率の積で表わす近似を用いる．すなわちプログラム・ページのみの場合の遷移確率を p_{ij}，データ・ページのみのものを $q_{\xi\eta}$ とすると

$$p_{(i\xi)(j\eta)}=p_{ij}\cdot q_{\xi\eta} \tag{4.1}$$

と表わす．遷移マトリックスを T_a で表わし，要素の順序は各 i の中で ξ の順序を付ける辞書的順序とする．

一方，N 回遷移後の状態確率を $\pi_{(i\xi)}(N)$ とし，これを要素とするベクトルを $\pi(N)$ とする．これにたいして，(3.2)式が成立する．

p_{ij} は3.2.1項で用いた特性のものを用いる．$q_{\xi\eta}$ は3.2.2項のものを基本とするが，より実際の状況に近いものとするため，データ参照の対象をファイルのみでなく，ユーザ作業領域やシステムのコモン領域など，主メモリ内に存在する確率の高いもの (CDP) と，ユーザ用のファイル領域 (FDP) に分ける．CDP 内の遷移は一律に確率 a，CDP から FDP へは b，FDP から CDP へは c とする．FDP 内ではデータ形式による相違を反映できるように，対角要素は d，他は e とする．これより以下のように遷移確率が得られる．

$$P_{(i\xi)(j\eta)}=\begin{cases}\alpha a, & i=j,\ \xi\in\text{CDP},\ \eta\in\text{CDP}\\ \alpha b, & i=j,\ \xi\in\text{CDP},\ \eta\in\text{FDP}\\ \alpha c, & i=j,\ \xi\in\text{FDP},\ \eta\in\text{CDP}\\ \alpha d, & i=j,\ \xi\in\text{FDP},\ \eta\in\text{FDP},\ \xi=\eta\\ \alpha e, & i=j,\ \xi\in\text{FDP},\ \eta\in\text{FDP},\ \xi\neq\eta\\ \beta a, & i\neq j,\ \xi\in\text{CDP},\ \eta\in\text{CDP}\\ \beta b, & i\neq j,\ \xi\in\text{CDP},\ \eta\in\text{FDP}\\ \beta c, & i\neq j,\ \xi\in\text{FDP},\ \eta\in\text{CDP}\\ \beta d, & i\neq j,\ \xi\in\text{FDP},\ \eta\in\text{FDP},\ \xi=\eta\\ \beta e, & i\neq j,\ \xi\in\text{FDP},\ \eta\in\text{FDP},\ \xi\neq\eta\end{cases} \tag{4.2}$$

全データのうち FDP の一部と CDP が W_a を形成する．そこで，FDP のうち主メモリ内のページ数を l，CDP のページ数を k，総データ・ページ数を m_D とする．$z_D=k+l$，

$$ka+(m_D-k)b=1,\quad kc+d+(m_D-k-1)e=1 \tag{4.3}$$

である．

W_a 以外のページ（状態）は吸収状態であり，(3.7)式より明らかなように，状態 (i,ξ) のうち，i,ξ 共に W_a に入るもの以外を除去したマトリックス $\hat{T}_a^*$ を用いれば十分である．$\hat{T}_a^*$ は相当大型のマトリックスとなるが，上記の近似の場合，解析解が求まる．以下，結果のみ示す．なお，出発時の ξ は $\xi\in$CDP とする．

$$L_a=(L_1/\Delta_1)+(L_2/\Delta_1\cdot\Delta_2) \tag{4.4}$$

$$L_1=(ka+lb)\alpha_1+kl\alpha_1^2\delta' \tag{4.5}$$

$$L_2=\beta z_p[(ka+lb)-kl\delta'\{(\alpha_1+\alpha_2)-l\alpha_1\alpha_2(d+(l-1)e-lb)\}] \tag{4.6}$$

$$\Delta_1=1-(ka+l\hat{d})\alpha_1+kl\alpha_1^2\delta' \tag{4.7}$$

$$\Delta_2=1-(ka+l\hat{d})\alpha_2+kl\alpha_2^2\delta' \tag{4.8}$$

$$\alpha_1=\alpha-\beta,\quad \alpha_2=\alpha+(z_p-1)\beta \tag{4.9}$$

$$\hat{d}=[d+(l-1)e]/l,\quad \delta'=a\hat{d}-bc \tag{4.10}$$

これは上で述べた近似の下での一般的な解を示すが，さらにいくつかの仮定を加え，典型的な場合のシステムの特性を求めることにより，システム全体の傾向を探ることができる．**図4**(a)～(d)はこれを示すもので，あるプロセス P_a の実行開始時にプログラム・ページを z_p，データ・ページを z_D だけ与えてランに入ったときの寿命 L を求め，これを三次元座標で表わしたものである．Ⅰ～Ⅳの各ケースにたいす

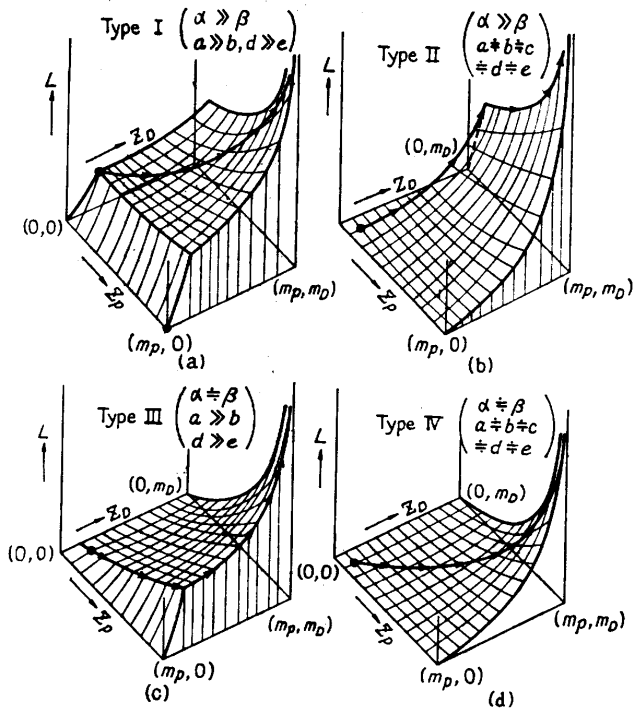

図 4 プログラムの寿命—手続きページ数およびデータ・ページ数
Fig. 4 Life time vs. number of program pages and number of data pages.

るパラメータの値から明らかなごとく，Ⅰはプログラム，データ共に強い局所性を有するもの，Ⅱはプログラムは強い局所性を有するが，データには強い局所性がない場合，Ⅲは逆にプログラムには強い局所性がないが，データにはある場合，Ⅳはどちらにも局所性のない場合である．

次にある点 (z_p, z_D) から出発し，割込みが生じたら要求されたページをそれに加えるという操作を続けるとしよう．(z_p, z_D) 点からプログラム・ページが要求された場合は (z_{p+1}, z_D) 点へ，データ・ページが要求されたら (z_p, z_D+1) 点へ移るが，主メモリ内にないページのうち，現在ページからの遷移確率の大きい順にこの参照が生ずる．結果として，この参照の系列は平均として図4の各形状の最大傾斜に沿って行われる．図4の各図で実線矢印で示されているのがこの参照系列を示す．そこでこの実線に沿って，z_p+z_D を横軸に，寿命 L を縦軸にとって示したのが**図5**である．この図はプロセス P_a のうち，L を最大にするという意味で最適なページを（プログラム，データの区別を付けずに）選ぶという前提のもとで，主メモリ内

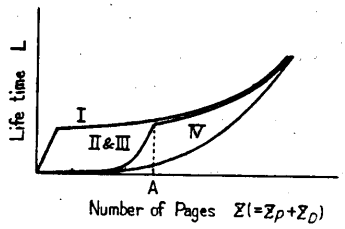

図 5 寿命—ページ数
Fig. 5 Life time vs. number of pages.

ページを増加したときの寿命の変化を示している．同図のⅠ～Ⅳは図4のものに対応する．

Ⅰ～Ⅳの4つのケースのうち，Ⅲ，Ⅳは実際的ではないことは3章から明らかである．Ⅰのケースはシステム効率上最も有利な場合であるが，今後データ量が増大したときデータにプログラムと同程度の局所性を期待することは線形配列など一部の特別の場合を除き，難しい．したがって以下では今後増大すると予想されるⅡのケースについて考察してみよう．

4.2 プロセスの累積処理時間

大きなプロセスでは全体がいくつかのサブプロセスに分けられていることが多い．このようなプロセスは，各サブプロセス内の遷移マトリックスをサブマトリックスとして含み，これとサブプロセスへのエントリ部を表わす非零の要素以外は零要素を有する**図6**のような遷移マトリックスで表わされる．個々のサブプロセスが図5のⅡの場合の特性を有するものとしよう．このプロセス全体について4.1節と同様に横軸にページ数を，縦軸に寿命を取って表わすと**図7**(a)のようになるであろう（ただしこれはあくまで平均としての議論である）．

このように得られた曲線の累積値を作ってみよう．横軸が x 点までの累積値は，一度主メモリに読込まれたページは Swap out されることがないという仮定のもとでこのプロセスの最初の実行開始以後，CPU

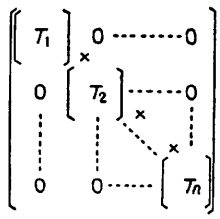

図 6 一個以上のサブ・プロセスを含むプロセスの遷移マトリックス
Fig. 6 Transition matrix of the process consisted of more than one subprocesses.

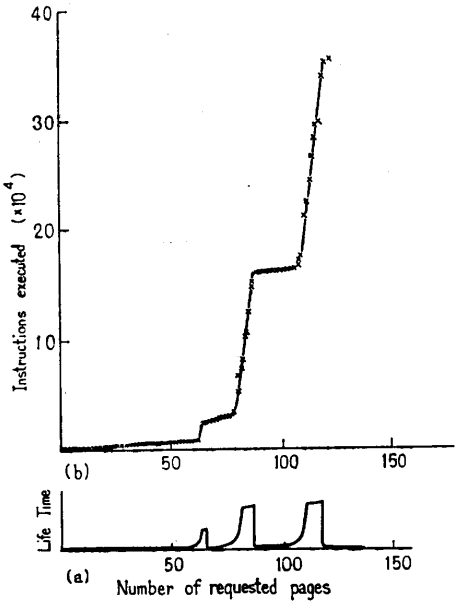

図 7 寿命および総実行命令数一要求ページ数
Fig. 7 Life time and total instructions executed vs. number of requested pages (ref 8)).

によって処理された総時間である．この累積値は図7(b)のようになるであろう．実は図7(b)の曲線は実際のプロセスについて，動的なトレーサを用いて益田[8]により得られた結果である．この実験に際して，当該プロセスの動特性を本モデルにより表わすのに必要な諸変数またはパラメータの値が記録されていればそれを用いたモデルと実験との定量的な比較ができるのであるが，その記録が残されておらず，一般的な傾向の比較に留まっている．

4.3 スラッシング

記憶容量 M を固定し，プロセスの個数 r と寿命との関係を調べるのは興味深い，同種のプロセスが r 個だけ ready になっているものとする．各プロセスには M/r 以下でこれに最も近い整数のページを与えるものとする．また各ページは最適系列で選ばれるものとする．プログラム・ページを P，データ・ページを D で表わすと，図4のケースⅡの場合，この系列は

$$P_1, D_1, D_2, \cdots, D_{m_D}, P_2, \cdots$$

となる（図4参照）．

したがってページ数 z が与えられたとき

$$\begin{cases} z_p=1, & z_D=z-1, \text{ if } k+1\leq z\leq m_D+1 \\ z_p=z-m_D, & z_D=m_D, \text{ if } m_D+1\leq z\leq m_p+m_D \end{cases}$$

とし，$z_D=k+l$ として(4.4)式ないし(4.9)式を用いることにより寿命 L が求まる．ここで $z\leq k$ および $m_p+m_D<z$ の範囲は複数 I/O 系など，より実際に近いモデルを用いないと扱えないので別稿に譲る．

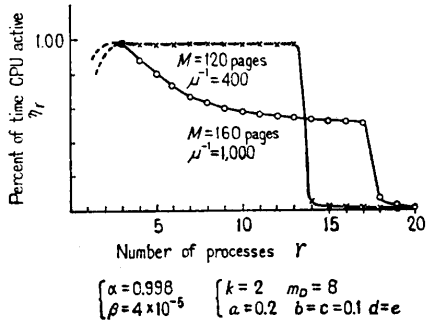

図 8 メモリ容量一定のときの CPU 効率一レディ状態のプロセス数
Fig. 8 Percent of time CPU is active vs. number of processes in ready state when memory capacity is fixed.

寿命 L が求まると $\lambda=L^{-1}$ であり，I/O 処理速度は使用する I/O 装置によって定まるので(2.2)式よりシステムの効率が求まる．図8はこの一例を示したものである．ここで r を増したとき，ある r の値で η が急降下することに注意されたい．これはスラッシングと呼ばれる現象で現実に生ずる可能性のあるものである．スラッシングはプロセスが図4のⅡのケースの場合に生ずる．これは r が増大し，1プロセスあたりのページ数 z が，図5の A 点以下になるため，寿命が急速に減少し，これが η_r を下げるためである．スラッシングは興味ある現象であるが，紙数の都合もあり，より詳しい検討は別稿で扱うこととし，本稿では，システムの全体的なモデル化により，このような現象の分析が可能となることを指摘するにとどめる．

4.4 データ量の増加に伴う性能の低下とその改善の可能性

図4の示す最も重要な点は，Ⅱの型のプロセスにおいてはデータ・ページがほとんどすべて主メモリに準備されない限り，寿命を長くすることが困難なことである．このことは一プロセスあたりの平均データ量が増すとシステム効率が著しく減少することを意味する．2章で示したように，システム効率を増すためにはメモリ容量を増すか，I/O 処理時間を短縮するか，寿命を長くする必要がある．I/O 装置に関しては，大容量であり，かつアクセスの著しく高度のものを期待することは現状では困難であり，データ量の増大は主メモリ容量の増加で補う他ない．しかし大型のデータ利用の傾向が近年著しく，主メモリ容量の増大によってシステム効率の維持を図ることには限界がある．このことは，データの局所性を動的に増加するような新

しいシステムのアーキテクチャの必要性を示している．この問題の検討は別の機会に譲る．

5. 結　び

本論文では計算機システムのひとつのモデル化の方法を与え，それがシステムの特性の把握や効率の評価，その改善方法の検討などに有用であることを示した．この方法が従来行われてきた評価や解析手法と最も異なるのは，システムの全体的な動作を比較的簡単な数学的モデルで表わすことにより，システム動特性にたいする見通しを良くし，システム効率への諸要因の影響やその対策を明らかにし得ることである．反面，このモデルの正当性の検討はまだ十分ではない点は批判の対象になり得る．現実にはモデルの正当性を直接的に示す実験的データが少ないことがこの最大の原因であり，これは今後の課題に残されている．

それにもかかわらず，このモデルの正当性が期待できる一つの根拠がある．それはデータの遷移特性に関して相当幅広く変化させても，システム特性は図4のⅡの型のものになるという事実である．これが図4のⅠのようにまで変化するにはデータはプログラムと同程度の局所性を必要とし，データ構造として線型配列など一部のものに限られる．

さらにプログラムに関する局所性はほとんど例外なく存在することも事実である．各プログラマの恣意に任せられたプログラムが，局所参照性という点ではかなり似かよったものになる．

この二つの事実から，システム傾向の，個々の数値にたいする不感性が結論される．

最後に本稿は紙数の都合でかなりの省略を行っている．特に swap out については何も触れていない．これらについては別の機会に述べたい．

最後に本論文の検討に参加して頂いた筑波大　益田隆司，電通大　亀田寿夫，日立製作所システム開発研究所　大町一彦，西垣　通，米田　茂，東京都臨床医学研究所　刈谷文治，東大宇宙航空研究所　山内平行の諸氏に感謝の意を表する．

参考文献

1) Coffman, E.G. and Varian, L.C.: Further Experimental Data on the Behavior of Programes in a Paging Environment, Commun. ACM, Vol. 11, No. 7, pp. 471-474 (1968).
2) Coffman, E.G. and Ryan, T.J.: A Study of Storage Partitioning Using Mathematical Model of Locality, Commun. ACM, Vol. 15, No. 3, pp. 185-190 (1972).
3) Denning, P.J.: The Working Set Model for Program Behavior, Commun. ACM, Vol. 11, No. 5, pp. 323-333 (1968).
4) Denning, P.J. and Schwartz, S.C.: Properties of the Working Set Model, Commun. ACM, Vol. 15, No. 3, pp. 191-198 (1972).
5) Denning, P.J. and Graham, G.S.: Multiprogrammed Memory Management, Proceedings of the IEEE, Vol. 63, No. 6, pp. 924-939 (1975).
6) 益田，高橋，吉沢：ページング・マシンにおけるスワッピング・アルゴリズムの比較とプログラムの動作解析，情報処理，Vol. 13, No. 2, pp. 81-88 (1972).
7) 益田，高橋：アドレス軌跡を利用した計算機システムの解析法，情報処理，Vol. 13, No. 11, pp. 765-770 (1972).
8) 益田：仮想メモリ方式による計算機システムの解析と評価に関する研究，東京大学工学系学位請求論文，1977年2月．
9) Moler, C.B.: Matrix Computations with Fortran and Paging, Commun. ACM, Vol. 15, No. 4, pp. 268-270 (1972).
10) 村岡：オンデマンド・ページング・システム動作解析のためのアドレス・パターン・ジュネレータ，情報処理，Vol. 16, No. 8, pp. 671-677 (1975).
11) 村田，堀越：対称帯行列を三重対角化するための新アルゴリズム，情報処理，Vol. 16, No. 2, pp. 93-101 (1975).
12) Saltzer, J.H.: A Simple Linear Model of Demand Paging Performance, Commun, ACM, Vol. 17, No. 4, pp. 181-186 (1974).
13) Spirn, J.R. and Denning, P.J.: Experiments with Program Locality, AFIPS Conference Proceedings, FJCC, Vol. 47, pp. 611-621 (1972).
14) Wesley, W.C. and Opderbeck, H.: The Page Fault Frequency Replacement Algorithm, Proceedings, FJCC, pp. 597-609 (1972).

(昭和53年5月18日受付)
(昭和54年1月18日採録)