論　文

多重プログラミング・システムの効率について*

宮崎正俊**　富田真吾***
野口正一****　大泉充郎***

Abstract

For performance evaluation of multiprogramming systems, a queuing model with a finite waiting room is considered. The model is assumed that a computer system consists of a CPU and identical channels, and service times of the both units are exponentially distributed with independent random variables. Then, the queuing model is described as $M/M/S(N)$, where S is number of channels and N is degree of multiprogramming.

Compared with measured and calculated data, an appropriateness of the model is discussed. Futhermore, some considerations of factors which affect performance of systems are given, and overhead of a monitor, throughput and cost performance are also discussed.

1. はじめに

多重プログラミングはほとんどのオペレーティング・システムで採用されている基本的な技法であり，計算機システムの性能を評価する際には重要な要素となるものである．このため，多重プログラミング・システムの解析は従来から種々行われてきた．

その手法は主として実際のシステムの計測（メータリング），シミュレーション，数学的モデルの3つに分けられる．計測による方法では目的とするシステムに関する詳細なデータを得ることができ，ある程度正確な評価が可能であるが，一般性に乏しくシステムを拡張したときの予測や結果を他のシステムに応用することは困難である．シミュレーションではモデルの作成とその記述が重要であり，モデルを正確にすればする程プログラムは複雑になり処理時間も多くなって，得られた結果の信頼性が問題となる．数学的モデルによる方法では，現実のシステムをどの程度正確に表現できるかによってその適合性が左右されるが，仮定を最小限にし，適切なモデルを作れば広く応用が可能となる．

本稿ではシステム評価の一般性を考慮し，数学的モデルによる解析を取り扱う．この方法は 1960 年代の後半から Gaver[1] 等により種々報告されているが，手法には主として確率論や待ち行列の理論が多く用いられている[2),3)]．しかし，一般にモデルが複雑であり拡張や応用が困難な場合が多いので，この点を考慮しここでは簡明な有限行列[5)] によって多重プログラミング・システムを表現した．2.ではその基本モデルの説明，3.では実際のシステムとの適合性と若干の解析を行う．さらに 4.ではシステムの効率に影響を与える要因，5.ではモニタのオーバーヘッド，6.ではコスト・パフォーマンスについて考察する．

2. 基本モデル

一般にオペレーティング・システムのもとで，計算機システムが1つのプログラムだけを処理する場合，システムの状態はプログラムの演算動作，入出力(I/O)動作，モニタの動作に分けられる．1つの演算動作とI/O 動作は交互に繰り返えされるべき性格のものであり，モニタの動作はほとんど演算動作および I/O 動作

* On Performance Evaluation of Multiprogramming Systems by Masatoshi MIYAZAKI (Tohoku University Computer Center), Singo TOMITA, Juro OIZUMI (Tohoku University Research Center for Applied Information Sciences) and Shoichi NOGUCHI (Tohoku University Research Institute of Electrical Communication)
** 東北大学大型計算機センター
*** 東北大学応用情報学研究センター
**** 東北大学電気通信研究所

の開始直前と終予直後に行われるものと考えられる．したがって，演算動作とモニタの動作を連続したものとみなすことができ，これを一括して CPU 動作と呼ぶことにすると，結局システムの状態は CPU 動作と I/O 動作の2つになり，これらが交互に繰り返えされることになる．

I/O 動作は一般に複数個の I/O 装置が対象となるが，最近のシステムでは中間の入出力媒体としてファイル（磁気ディスク等）を用いることが多いので，特別な I/O 装置（磁気テープ等）を使わない限り，I/O 装置はファイルを想定して1種類として扱っても特に一般性は失わない．以下，I/O 装置を1種類と仮定してこれをチャネルと呼ぶことにする．したがって，I/O 動作の要求はチャネルに対して出されることになる．

1回の CPU 動作が持続する時間は一般にランダムと考えられる．一方，1つの I/O 動作が完予するのに要する時間は，CPU と同程度のランダム性はないと思われるが，以後の解析を容易にするためにここでは一応ランダムと仮定する．ここで，プログラムをプロセスと呼ぶことにし，処理の対象となる複数個のプロセスをプロセス群ということにする，あるプロセス群が与えられ，これを1つずつ連続して処理するとき，このプロセス群の CPU 動作の持続時間はパラメータ λ の指数分布，I/O 動作の完了時間はパラメータ μ の指数分布に従うものとする．つまり，あるプロセス群はあるシステムで処理されるとき，パラメータ λ と μ によって特徴づけられることになる．

多重プログラミング・システムでは，いくつかのプロセスを同時に主記憶に置き，これらをみかけ上同時に処理する．主記憶に存在するプロセスの数は多重プログラミングの多重度と呼ばれるものであり，これを $N(1\leqq N<\infty)$ で表わす．あるプロセスの処理が完了してシステムから出力されると，同じプロセス群の中から新しいプロセスが入力され，システムには常に N 個のプロセスが存在するものとする．

N 個のプロセスのうち I/O 動作が完了して CPU 動作が可能なプロセスが少なくとも1つ存在する間は，CPU 動作が常に行われることになり，I/O 動作の要求（I/O 要求）がパラメータ λ の指数分布に従って発生する．ただし，CPU は1台とし，多重度によって λ の値は変化しないものとする．一方，チャネルから見たときは，I/O 要求は到着率 λ でポアソン到着することになる．

つぎに，N 個のプロセスがすべて I/O 動作の完了待ちの状態になった場合を考える．このとき CPU はどのプロセスも実行することができないので，アイドル・プロセスを実行するものとする．アイドル・プロセスとは，(1) CPU 動作の持続時間が処理中のプロセス群と同じパラメータ λ の指数分布に従う，(2) I/O 要求を出したあとその完了を待たずに引き続き CPU 動作に移れる，(3) 出された I/O 要求はチャネルで処理されずに消滅する，(4) モニタのオーバーヘッドはゼロである，という性格をもつものである．アイドル・プロセスの実行中に I/O 動作の完了したプロセスが1つでも生じると，その実行は即座に中断されそのプロセスの CPU 動作が開始される．

以上のようなアイドル・プロセスの概念を導入することにより，チャネルへはシステムの状態に関係なく I/O 要求が到着率 λ で到着することになる．ただし，処理中のものも含めた I/O 要求の数が N になると，その後に到着する I/O 要求はアイドル・プロセスからのものになり，これは I/O 要求の待ち行列に加わらず立ち去る（消滅）ことになる．このことはチャネルを窓口と考えたとき，系に存在する客（I/O 要求）が最大 N であるような有限行列と等しくなる．チャネルは同一能力のもの（処理率は μ）が複数個あってもよく，その個数を S とするとこの待ち行列は **Fig. 1**（次頁参照）のように $M/M/S(N)$ で表わされる．ただし，S 個のチャネルは独立動作が可能であり，I/O 要求はどのチャネルで処理されてもよいものとする．以後，λ を I/O 発生率，μ を I/O 処理率と呼ぶことにする．

$M/M/S(N)$ 型の待ち行列は既に解析されており[5]，処理中のものも含めて系に存在する I/O 要求の数（系の状態）を $n(0\leqq n\leqq N)$ とすると，系の平衡状態における平衡状態確率 P_n は（1）式のようになる．（2）式は自然に成立するのでこれを正則条件と呼ぶ．

$$P_0=\left\{\sum_{n=0}^{S}\frac{\rho^n}{n!}+\frac{\rho^S}{S!}\sum_{n=1}^{N-S}\left(\frac{\rho}{S}\right)^n\right\}^{-1},$$

$$P_n=\begin{cases}\dfrac{\rho^n P_0}{n!}, & (1\leqq n\leqq S)\\[2ex] \dfrac{\rho^n P_0}{S!\,S^{n-S}}, & (S<n\leqq N)\end{cases}$$

$$\rho=\frac{\lambda}{\mu}. \qquad (1)$$

$$\sum_{n=0}^{N}P_n=1. \qquad (2)$$

（1）式の ρ はプロセス群の特性を表わすものであ

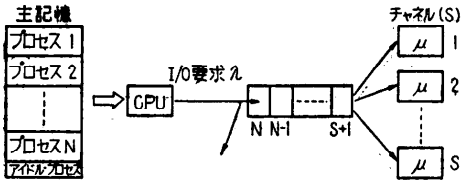

Fig. 1 Multiprogramming model with a finite waiting room

り，以後これをI/O率と呼ぶことにする．チャネルが1個の場合は，(1)式は簡単になり P_n は(3)式のようになる．

$$P_n=\begin{cases}\dfrac{1-\rho}{1-\rho^{N+1}}\rho^n, & (\rho\neq 1)\\ \dfrac{1}{N+1}, & (\rho=1)\end{cases} \quad . \tag{3}$$

上述した平衡状態確率を用いて多重プログラミング・システムの効率を考えると次のようになる．系の状態 n が $n=N$ のとき CPU はアイドル・プロセスを実行するので有効な仕事はできない．平衡状態におけるこの確率は P_N である．一方，$0\leqq n\leqq N-1$ のときは CPU はいずれかのプロセスの実行が可能なので，常に有効な仕事ができることになる．この確率は(2)式の正則条件から $1-P_N$ であり，この値はシステムの処理能力に影響を与えるので，これをシステム効率の1つの尺度と考えることができる．以後これを CPU 効率と呼ぶことにし $\alpha(0\leqq\alpha\leqq 1)$ で表わすと(4)式のようになる．

$$\alpha=\begin{cases}1-\dfrac{\rho^N P_0}{S!\,S^{N-S}}, & (S<N)\\ 1-\dfrac{\rho^N P_0}{N!}, & (S\geqq N),\end{cases}$$

$$P_0=\begin{cases}\left\{\displaystyle\sum_{n=0}^{S}\frac{\rho^n}{n!}+\frac{\rho^S}{S!}\sum_{n=1}^{N-S}\left(\frac{\rho}{S}\right)^n\right\}^{-1}, & (S<N)\\ \left\{\displaystyle\sum_{n=0}^{N}\frac{\rho^N}{n!}\right\}^{-1}, & (S\geqq N),\end{cases}$$

$$\rho=\frac{\lambda}{\mu}. \tag{4}$$

いま，1つの CPU，チャネルおよびモニタで構成されるシステムがあり，このシステムで処理すべきプロセス群が与えられた場合，これを多重度1で処理することにより ρ の値が求まる．この ρ は多重度やチャネル数によって変化しないものとすると，(4)式から明らかなように α は多重度とチャネル数の関数となる．

* NEAC シリーズ 2200 モデル 700. オペレーティング・システムは MOD7.

3. 基本モデルの解析

最初に，前章で考察した基本モデルと実際のシステムとの適合性を検討する．**Fig. 2** は東北大学大型計算機センターのシステム*の実測値と基本モデルの計算値の比較である．実測には特定のプロセス群を用い，計測システム[6]によって CPU 効率を求めた．計算値を求める際に用いる ρ の値は多重度が1の場合の実測値から求めた．図から明らかなように，多重度が1のときは実測値と計算値は等しく，多重度が2以上では実測値の方がいくらか大きくなっている．

この原因としては次のことが考えられる．使用したチャネルは磁気ディスクであるが，この装置の I/O 動作は制御装置の回転待ちも含む転送動作と，ユニットのシーク動作に分けられ，しかもこの2つは並列動作が可能である．したがって，ユニットの数が制御装置の数より多い場合は，多重度が大きくなる程（ただしユニット数以下）チャネルの平均的な処理能力が向上することになるので，実測値の方が CPU 効率は大きくなる．多重度が2以上の場合の各 μ の平均的な値を実測して計算値を求めれば，各多重度における計算値は実測値にほぼ近い値になるものと予想される．

以上の **Fig. 2** に示された結果の考察から，有限行列を用いた多重プログラミングの基本モデルは，実際のシステムの解析に適用できると結論できる．

次に，計算例をいくつか示し若干の考察を行う．**Fig. 3** (次頁参照)はチャネルが1個の場合における多重度および ρ と CPU 効率の関係を示したものである．N の増加にともなって α も増加するが，(3)式から $\rho\leqq 1$ の場合は $\alpha\to 1$，$\rho>1$ では $\alpha\to 1/\rho$ となることが分かる．しかし，N をあまり大きくすることは，主記憶をより多く必要としモニタのオーバー

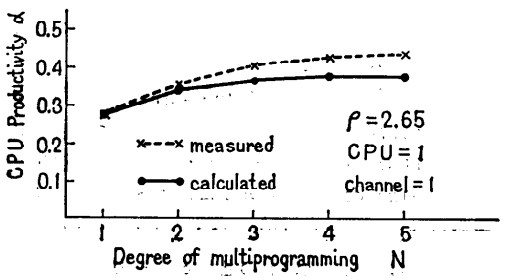

Fig. 2 A comparison between measured and calculated data of CPU productivity.

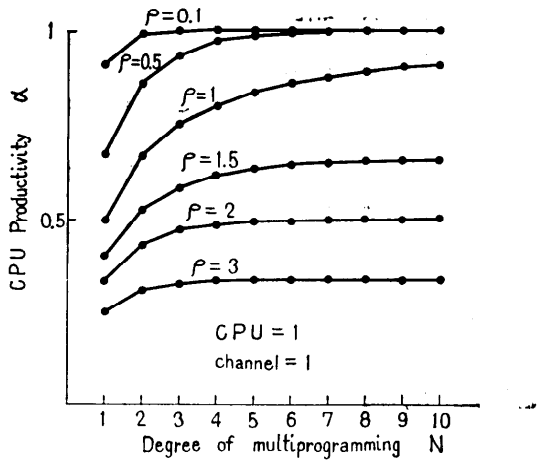

Fig. 3 Effect of degree of multiprogramming and ρ upen CPU productivity.

ヘッドの影響も考慮しなければならないので現実的ではない．これらの問題については後章で考察する．

あるシステムに対して ρ はプロセス群に固有な値である．つまり，ρ は CPU とチャネルの能力およびモニタの性格によって決まる．CPU 動作の多いプロセス群は ρ が小さく，I/O 動作の多いものは ρ が大きいので，ρ の小さいプロセス群を処理する程 CPU 効率がよくなるのは当然である．CPU とチャネルの能力が一定な場合，プロセス群の ρ を小さくする方法として I/O 要求のブロッキングがあるが，これについても後で考察する．

Fig. 4 は ρ が一定の場合におけるチャネル数および多重度と CPU 効率の関係を示したものである．チ

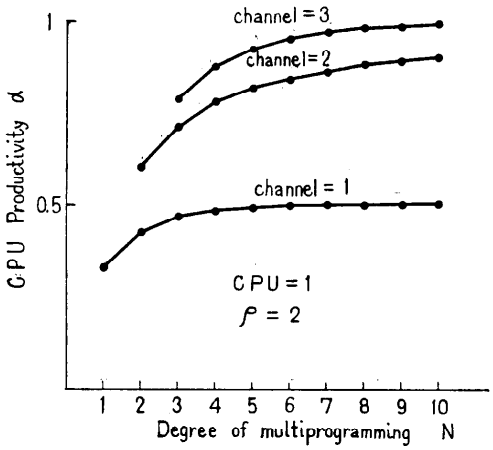

Fig. 4 Effect of number of channels upon CPU productivity.

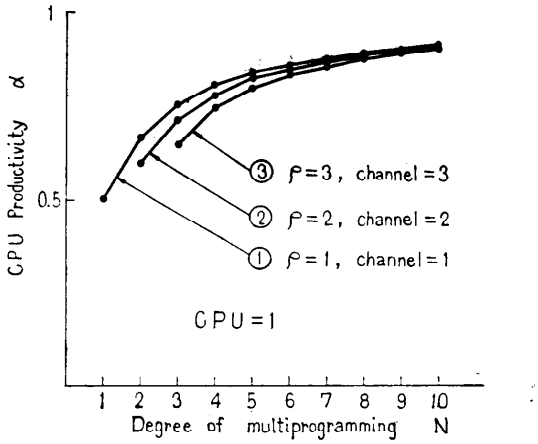

Fig. 5 A relation between a number and transfer rate of channels.

ャネルが1個の場合，多重度が5程度で CPU 効率は飽和に近くなるが，チャネルを2個にすると大幅に向上する．しかも多重度が5以上でも CPU 効率は更にいくらか向上する．チャネルを3個にしてもこの場合はあまり効果はない．しかし ρ が比較的大きい場合は有効である．したがって，どのようなプロセス群を処理するかによってチャネルの数を決める必要がある．

Fig. 5 はチャネルの処理能力と個数の関係を示したものである．あるプロセス群があり，そのλは一定であるとすると，図の①〜③のチャネルの各総能力はそれぞれ等しい．つまり，①を基準にすると同一プロセス群に対して，②は能力が 1/2 のチャネルを2個，③は 1/3 のものを3個使用する場合に相当する．図から明らかなように，処理能力の小さいチャネルを多く使うよりは，数は少なくても処理能力の大きいチャネルを使う方が CPU 効率はよい．特に，多重度が小さいところではこの傾向が強い．

4. CPU 効率とその要因

CPU 効率は多重度やチャネル数あるいは ρ の値（多重度が1の場合の）によって変化するが，この他にも CPU 効率を変動させる要因がいくつかある．この章ではこの問題を考察する．

4.1 多重度の変動

現実のシステムにおいては，あるプロセスの処理が終了して次の新しいプロセスを1つ選んで入力するとき，そのスケジューリングの対象となるプロセスの数は一般に有限である．プロセスが処理されるためには，主記憶，入力ファイル，作業用ファイル，出力フ

ァイル，専用ファイル等の各種のリソースが必要である．したがって，必要なリソースをすべて確保できるプロセスが1つも存在しないとき，同時に処理できるプロセスの数が減ることになる．このように，一般のバッチ処理においてはリソースの競合により，処理中の多重度が常に設定した多重度 N になっているとは限らない．

いま，処理中の多重度が $M(1\leqq M\leqq N)$ である確率を Q_M，このときの CPU 効率を α_M で表わすことにする．Q_M については

$$\sum_{M=1}^{N} Q_M=1 \qquad (5)$$

が成立する．前述の計測システムによる実際の運用において，$N=5$ のときに $Q_1=0.03$，$Q_2=0.23$，$Q_3=0.40$，$Q_4=0.27$，$Q_5=0.07$ というデータが一部得られている．

ここで，多重度が変動する場合の CPU 効率を α_V とすると，これは(6)式のように α_M の期待値として表わせる．

$$\alpha_V=\sum_{M=1}^{N} Q_M\alpha_M. \qquad (6)$$

ただし，α_M は(7)式で求める．

$$\alpha_M=\begin{cases}1-\dfrac{\rho^M P_0}{S!\,S^{M-S}}, & (S<M)\\ 1-\dfrac{\rho^M P_0}{M!}, & (S\geqq M)\end{cases},$$

$$P_0=\begin{cases}\left\{\displaystyle\sum_{n=0}^{S}\frac{\rho^n}{n!}+\frac{\rho^S}{S!}\sum_{n=1}^{M-S}\left(\frac{\rho}{S}\right)^n\right\}^{-1}, & (S<M)\\ \left\{\displaystyle\sum_{n=0}^{M}\frac{\rho^n}{n!}\right\}^{-1}, & (S\geqq M)\end{cases}. \qquad (7)$$

4.2 入出力のブロッキング

1回の I/O 動作の時間が転送するデータ量に依存する比例部分と，データ量に無関係な固定部分に分けられるようなチャネル（たとえば磁気ディスクや磁気ドラム）の場合は，I/O 要求をいくつかまとめて処理する（ブロッキング）ことにより，I/O 率が小さくなり CPU 効率は向上する．比例部分だけからなるようなチャネルでは I/O 率は不変である．

いま，K を I/O 要求のブロッキングの個数としたとき，I/O 発生率を λ_K，I/O 処理率を μ_K で表わし，このときの I/O 率を $\rho_K=\lambda_K/\mu_K$ とする．ブロッキングをしないとき $(K=1)$ の1回の I/O 動作のうち，固定部分の比率を $a(0<a<1)$，比例部分を $1-a$ とすると，λ_K,μ_K,ρ_K は λ_1 と μ_1 を基準にして(8)式のように表わせる．

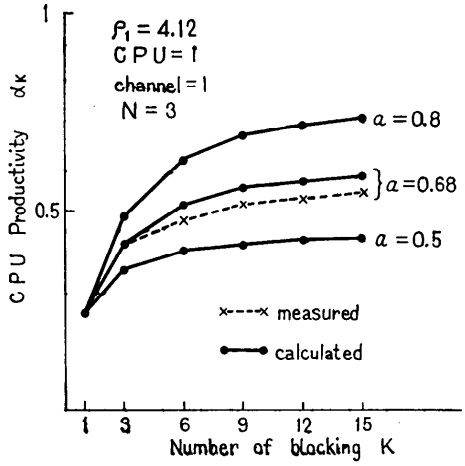

Fig. 6 Effect of number of blocking upon CPU productivity.

$$\lambda_K=\frac{\lambda_1}{K},$$

$$\mu_K=\frac{\mu_1}{a+(1-a)K},$$

$$\rho_K=\rho_1\left(\frac{a+(1-a)K}{K}\right). \qquad (8)$$

したがって，ブロッキング個数が K のときの CPU 効率を α_K とすると，これは(4)式において $\rho=\rho_K$ と置いたものになる．**Fig. 6** は文献 7) で報告されている実測データ(一部補正)と計算値を比較したものであり，さらに a を変化させたときの α_K も示してある．実測値はラインプリンタへの出力要求のみをブロッキングしたものであるが，この点を補正して考えれば実測値と計算値はほぼ等しくなるものと思われる．K を大きくすることによりいずれも CPU 効率は向上するが，その上限は $\rho_1(1-a)$ によって決まる．なお，a が大きい程ブロッキングの効果が大きいことは明らかである．

4.3 ページング・システム[4]

多重度の増加にともなって I/O 発生率が変化する例としてページング・システムがある．限られた主記憶(ページ)にプロセスを多数置く程，各プロセスのページ要求の頻度が多くなる．いま，I/O 要求がページ要求のみであるとすると，I/O 発生率は多重度 N の関数となる．どのような関数になるかは処理するプロセス群のアドレス参照のパターンに依存する．

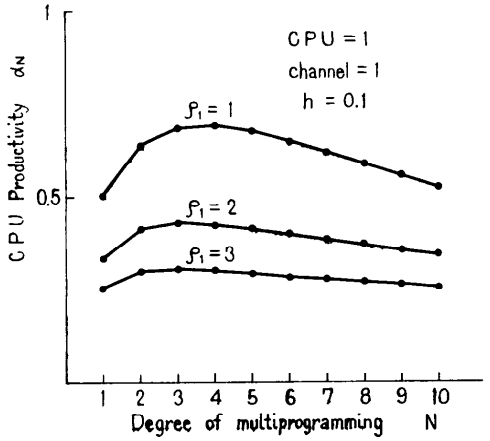

Fig. 7 CPU productivity of paging system.

多重度が N のときの I/O 発生率を λ_N とし，I/O 処理率は不変で μ，I/O 率を $\rho_N=\lambda_N/\mu$ とする．λ_N と ρ_N を一般的に表わせば(9)式のようになる．ただし，$f(N)$ は正の値をとる非減少関数で $f(1)=1$ とする．

$$\lambda_N=\lambda_1 f(N),$$
$$\rho_N=\rho_1 f(N),\ (1\leqq N<\infty). \qquad (9)$$

多重度が N のときの CPU 効率を α_N とすると，これは(4)式において $\rho=\rho_N$ と置いたものになる．$f(N)$ は非減少関数であるから，N の増加にともなって ρ_N は大きくなるので，α_N は逆に小さくなる．そして $f(N)$ の性質によっては，α_N が最大値をもつような場合がある．

実際のシステムにおける $f(N)$ の関数型は現在求まっていないが，全体のページ数が多く各プロセスが必要とするページが少ない場合は，多重度が小さいところでは線型の関数となることが予想される．そこで，例として $f(N)$ が次式で表わせる場合を考える．

$$f(N)=1+h(N-1). \qquad (10)$$

Fig. 7 は $h=0.1$ の場合における α_N をいくつかの ρ_1 について求めたものである．ρ_1 が大きい場合は N を大きくしてもあまり影響はないが，ρ_1 が小さいと α_N は減少する．実際には，N をあまり大きくすると λ_N が急激に大きくなり，スラッシングの状態がおこり α_N が極端に小さくなる現象が生じる．

5. モニタのオーバーヘッド

いままで議論した CPU 効率にはモニタのオーバーヘッドが含まれていた．CPU 効率がいくらよくてもオーバーヘッドが大きければ，実際のプロセスに対する CPU の割当ては少なくシステムの処理能力は低下する．この章では CPU 効率のうちプロセスに使える比率をプロセス効率と定義し二，三の考察を行う．

あるプロセス群を多重度1で処理したときの I/O 発生率を λ_1 とすると，CPU 動作時間の平均は $1/\lambda_1$ であり，この中にモニタの時間 δ が含まれているものとすると，$1/\lambda_1-\delta$ がプロセスに使える時間となる．$1/\lambda_1-\delta$ と $1/\lambda_1$ の比を (11) 式のように η_1 で表わすと，η_1 は平衡状態においても CPU 効率のうちプロセスに使える部分の比率を表わすことになる．したがって，多重度によって δ は変化しないものとすると，プロセス効率を β で表わせばこれは(12)式のようになる．

$$\eta_1=\frac{1/\lambda_1-\delta}{1/\lambda_1}=1-\lambda_1\delta. \qquad (11)$$
$$\beta=\alpha\,\eta_1. \qquad (12)$$

δ は一般にシステムに固有の値になるものと思われるが，λ_1 はプロセス群に固有の値であるから η_1 も同様である．**Fig. 8** は Fig. 2 と同一のプロセス群についてプロセス効率の実測値と計算値を示したものであり，多重度がこの範囲では両者はほぼ近い値となる．

しかし，一般に多重度が増加するとプロセス効率は低下することが考えられるので，以下この点を考慮したプロセス効率を検討する．多重度が N のときの I/O 発生率を λ_N，I/O 処理率を μ (一定)，I/O 率を $\rho_N=\lambda_N/\mu$ とする．オーバーヘッドは多重度に比例して増加するものと仮定すると，$1/\lambda_N$ は

$$1/\lambda_N=1/\lambda_1+\delta\xi(N-1) \qquad (13)$$

のように表わせる．ここで ξ は比例定数である．ρ_N は(11)式と(13)式を用いてつぎのようになる．

$$\rho_N=\frac{\rho_1}{1+(1-\eta_1)\xi(N-1)}, \qquad (14)$$

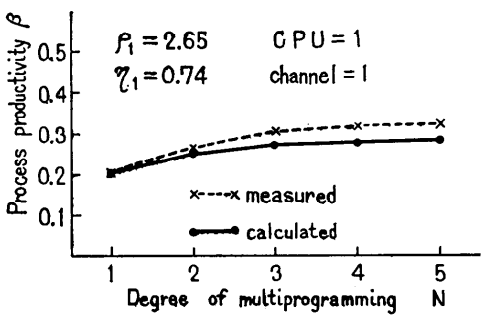

Fig. 8 A comparison between measured and calculated data of process productivity

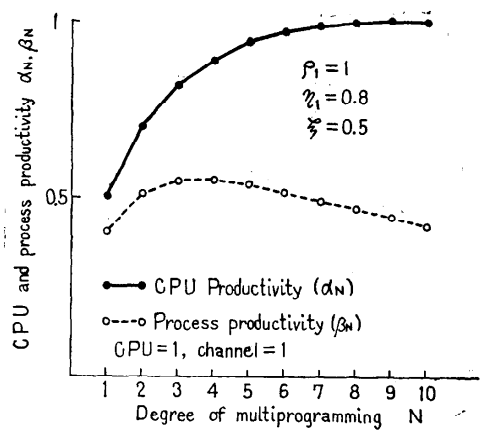

Fig. 9 An example of CPU and process productivity.

一方，多重度が N のときの η_N は

$$\eta_N=\frac{\eta_1}{1+(1-\eta_1)\xi(N-1)} \tag{15}$$

と表わせるので，結局，多重度が N のときの CPU 効率を α_N，プロセス効率を β_N とすると，β_N は(16)式となる．ここで，α_N は(14)式と(4)式で求める．

$$\beta_N=\alpha_N\eta_N. \tag{16}$$

α_N および β_N は ρ_1, η_1, ξ および N の関数となるが，ρ_1 と η_1 はプロセス群によって決まる固有の値であるから，結局 ξ と N のみの関数となる．**Fig. 9** は $\rho_1=1$, $\eta_1=0.8$, $\xi=0.5$ の場合の α_N と β_N の計算例である．N の増加にともなって β_N は最初のうちは増加するが，α_N が1に近づくにつれて逆に減少する．このケースでは多重度が3あるいは4のところで β_N が最大となる．

6. コスト・パフォーマンス

CPU とチャネルの能力が一定の場合，プロセス効率が大きくなればシステムの処理能力は向上する．このためには，ブロッキングの活用，多重度の増加，チャネルの増設等の手段が考えられる．しかし，多重度の増加あるいはチャネルの増設には相当の費用がかかるはずである．したがって，プロセス効率が向上してもプロセス当りの処理費用が増大し，コスト・パフォーマンスは逆に悪くなる．この章ではこの種の問題を考察する．

あるプロセス群において，1つのプロセスが完了するのに要する CPU 時間の平均を τ とする．多重度が N，チャネル数が S のときの単位時間当りのプロセスの処理件数（スループット）を $H_{N,S}$ で表わすと，これは β_N と τ によりつぎのようになる．

$$H_{N,S}=\frac{\beta_N}{\tau}, \tag{17}$$

一方，システムの費用も N と S により $R_{N,S}$ で表わす．これよりプロセス1個当りの費用 $W_{N,S}$ は

$$W_{N,S}=\frac{R_{N,S}}{H_{N,S}}=\frac{\tau R_{N,S}}{\beta_N} \tag{18}$$

となる．

ここで，多重度を1つ増やすのに必要な主記憶の費用を v_M，チャネル1個当りの費用を v_c とすると，$R_{N,S}$ はつぎのように表わせる．

$$R_{N,S}=R_{1,1}+v_M(N-1)+v_c(S-1). \tag{19}$$

したがって，(18)式は

$$W_{N,S}=\frac{\tau}{\beta_N}\{R_{1,1}+v_M(N-1)+v_c(S-1)\} \tag{20}$$

となる．ここで，$W_{1,1}$ を基準にすることにし(20)式に

$$\tau=\beta_1 W_{1,1}/R_{1,1},\quad v_M=AR_{1,1},\quad v_c=BR_{1,1}$$

を代入すると(21)式が得られる．ただし，A と B は $R_{1,1}$ を基準にしたときの比例定数である．

$$W_{N,S}=\frac{\beta_1}{\beta_N}W_{1,1}(1+A(N-1)+B(S-1)). \tag{21}$$

あるシステムとプロセス群が与えられたとき，$W_{1,1}$ と β_1 は定数となり，A と B も定数として与えられるので，$W_{N,S}$ が最小となるような N と S を求めることができる．**Fig. 10** は，$\rho_1=2$, $\eta_1=0.8$, $\xi=0$, $A=0.1$, $B=0.3$ の場合の $W_{N,S}$ と $W_{1,1}$ 比を求め

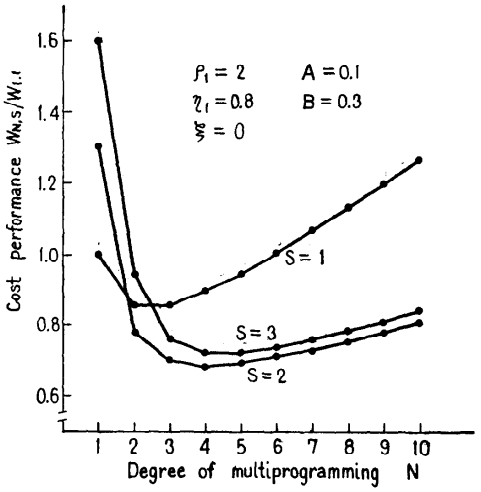

Fig. 10 An example of cost performance.

たものである．図より明らかなように，多重度が 4，チャネルが2個のときにコスト・パフォーマンスが最良となる．

7. むすび

多重プログラミング・システムを有限行列のモデル $M/M/S(N)$ で表現すると，実際のシステムと比較的よく適合することが分かり，しかもモデルが簡単であるためこれを応用してシステムの効率に関する種々の検討を行うことができた．

1つのシステムに関する基本的なデータ，つまり多重度が1のときの I/O 発生率，I/O 処理率，モニタのオーバーヘッド，基本費用等が分かれば，そのシステムの拡張（多重度やチャネル数に関して）や改造（ブロッキング数等）に関してその効果を容易に予測することができる．

特に，システムの拡張とコスト・パフォーマンスの関係は従来あまり議論されていないが，工学的にもこの問題は重要である．(19)式で多重度と主記憶の費用の関係を線型とみなしているが，実際には主記憶はある容量の単位で増設されるので，むしろステップ関数で与えた方がより正確な評価が得られるものと予想される．

一方，限られた主記憶の容量に対する多重度の設定は，プロセスが必要とする主記憶に依存するが，多重度をいくら大きくしても **4.1** で述べた処理中の多重度の変動が大きければ，コスト・パフォーマンスの低下を招く場合があり得る．

この基本モデルは，システムのチャネル・ネックという現象を前提にしている．I/O 動作が少なく大部分が CPU 動作となるようなプロセス群（ρ が極めて小さい）では，チャネル・ネックはほとんど起こらないが，このモデルは CPU 効率のみを問題にしているので，このようなプロセス群に対して適用した場合でも，ほぼ同じようにシステム効率を解析できるものと思われる．

このモデルでは I/O 動作を指数分布に従うものと仮定しているが，チャネルやシステムあるいはプロセス群の性質によっては他の分布を仮定する必要もあろう．また，チャネルが磁気ディスクのような場合は直列型の有限行列を用いる方が望ましい．

謝辞 実際のシステムの計測と種々の議論をしていただいた東北大学大型計算機センターの松沢茂，小畑征二郎の両氏に深謝する．

参考文献

1) D. P. Gaver: Probability Methods for Multiprogramming Computer Systems, J. of ACM, Vol. 14, No. 3, pp. 423～438 (1967).
2) I. Adiri, M. Hofri, M. Yadin: A Mutiprogramming Queue, J. of ACM, Vol. 20, No. 4, pp. 589～603 (1973).
3) 田中穂積：循環待ち行列モデルを用いた多重プログラミング・システムの解析，電子通信学会論文誌，Vol. 53-C, No. 2, pp. 57～64 (1970).
4) P. J. Denning: Resource Allocation in Multiprocess Computer Systems, MIT, MAC-TR-50 (1968).
5) 待ち行列研究会編：応用待ち行列事典，広川書店（昭 46).
6) 松沢，小畑，宮崎，青山，菊池：ソフトウェアによる評価システムのための計測について，情報処理学会第 14 回大会 (1973).
7) 岡部，高橋，坂田，高橋：スループットの向上について，電気関係学会東北支部連合大会(1973).
8) 宮崎，富田，野口，大泉：多重プロセッサのシステム効率に関する一考察，情報処理学会第15回大会 (1974).

（昭和50年1月17日受付）
（昭和50年6月18日再受付）