論　文

タイムスライススケジューリングを持つバッチシステムの近似解析*

渡　辺　　隆**

Abstract

This paper discusses a batch system with time slice scheduling. We present a cyclic queue model with a feedback in the CPU stage, in which the CPU burst time distributions are exponential but have different avarage values. The transitions of the states in this model form a semi-Markov process.

It becomes more and more difficult to evaluate the equilibrium probability distribution as the number of states increases.

We present its approximate analysis in the case that the time slice value is relatively small. Further we define the Job Load that estimates the load of a job in the multiprogramming system and give its value under time slice scheduling.

1. はじめに

バッチシステムは，ジョブの多重度が一定であることから，普通 M 個の CPU と N 個の I/O で構成する循環待行列でモデル化される．その場合，解析を簡単にするために，それぞれのユニットにおけるサービス時間の分布は，ジョブに独立で同一の指数分布に従うことが仮定される．

この様なモデルに関しては，窓口に存在する客の数で規定される状態の平衡確率分布が一般的に定式化されており[1]，更にこれをもとにして，ユニットの利用率や行列長の値を求める簡単な計算機アルゴリズムも与えられている[2]．

しかし，普通のバッチシステムでは必ずしも各ユニットにおけるサービス時間は独立で同一の分布に従っているわけではない．確かに，I/O に関してはディスクであればサービス時間のほとんどはアームのアクセス時間である様にジョブに無関係に，ユニットの種類ごとに同一の分布になっていると考えられるが，CPU では一般にあてはまらない．たとえばコンパイラやファイルのユーティリティは I/O 指向型のジョブで，ユーザの実行プログラムやシミュレーションプログラムに比べて一回当りの連続 CPU 時間は短く，それぞれ違う分布をなしていると考えられる．

またマルチプログラミングシステムでは，一つのジョブが CPU を占有しないように種々のスケジューリングアルゴリズムを採用している．たとえば一つのジョブに一定時間以上のサービスを与えないタイムスライス方式や，いろいろな予測式を用いて最も短い時間しか CPU を使用しないと考えられるジョブに優先権を与えるダイナミックディスパッチングが代表的なものである[3]．

この様な状況で，システムに存在する個々のジョブのターンアランドタイムやジョブに対する課金等を扱わなければならない場合は，従来の均一なジョブを扱った FIFO モデルでは不充分である．

そこで本稿では，CPU の連続使用時間がジョブによって異なる平均値を持つ指数分布であるジョブをタイムスライススケジューリングを持つバッチシステムで処理する場合を，CPU ステージにフィードバックを持つ循環待行列でモデル化する．このモデルの状態推移はセミマルコフ過程をなす．しかしジョブの種類が多くなれば定義しなければならない状態の数が多く

* An approximate analysis of a batch system with time slice scheduling by Takashi WATANABE (Railway Technical Research Institute, Japanese National Railways).

** 日本国有鉄道鉄道技術研究所電子計算センタ

なって平衡確率分布の計算は事実上出来なくなる．

そこで，上記のモデルにタイムスライスの値を0に近づけた仮想的な Processor Shared model を適用すれば，平衡確率分布の計算も容易であり，タイムスライスの値が適当に小さい時にはシミュレーションによって，CPU 利用率は現実の値に十分近い値を持つことを示す．

次にマルチプログラミングシステムにおけるジョブの負荷を定義し，Processor Shared 近似によってタイムスライスを持つバッチシステムでの個々のジョブ負荷評価式を与える．

2. タイムスライスモデルと平衡確率分布

2.1 物理モデル

バッチシステムのモデルとして **Fig. 1** の様な1個の CPU と N 個の I/O で構成する循環待行列を考える．系内のジョブ多重度は常に一定 n である．ジョブの連続 CPU 使用時間の分布はジョブごとに違う平均値を持つ指数分布とする．I/O サービス時間は I/O 種類ごとに同一の指数分布をなすものとする．

CPU で処理されているジョブは，ある定められたタイムスライスの値を越えて CPU は与えられない．タイムスライス内に CPU 処理が終らなければ，CPU 行列の最後につけられ，CPU は次のジョブを処理する．CPU 処理がタイムスライス内に終了すれば，ジョブに独立に p_i の確率で i 番目の I/O（以下 i I/O とする）を選んでサービスを要求する．I/O サービスは先着順であって，サービス終了後はただちに CPU ステージにゆき，同じサイクルをくりかえす．

2.2 平衡確率分布

このモデルは，タイムスライスの値が一定値であるために普通のマルコフ過程として平衡確率分布を求めることができない．そこで次の様な特定の時点（再生点）での各ユニットにおけるジョブの順列の組み合わせによってシステムの状態を規定する．

（1） CPU がジョブ処理中で，タイムスライスの値がつきた直後．

（2） CPU がジョブ処理を終了した直後．

（3） CPU ステージにジョブが存在しない時には，任意の I/O がジョブを終了した直後．

そうすると，状態推移は有限のマルコフ連鎖（隠れマルコフ連鎖）を形成するから，モデルの状態推移はセミマルコフ過程をなしている．

この隠れマルコフ連鎖の推移確率行列を Q とすると，この隠れマルコフ連鎖で表わされる状態の平衡確率分布 $\boldsymbol{y}$ は

$$\boldsymbol{y}Q=\boldsymbol{y} \tag{2.1}$$

で求められる[4]．これにより状態 i の平均推移時間を t_i とすれば，求めるセミマルコフ過程の状態 i の平衡確率 x_i は

$$x_i=y_it_i/\sum_j y_jt_j \tag{2.2}$$

で表わされる[4]．

この方法は一般的であって，モデルのパラメータのどの様な値に対しても平衡確率分布を求めることができる．しかし隠れマルコフ連鎖の推移確率行列を計算する際は，I/O に存在するジョブの数だけの「たたみこみ積分」が必要となるし，ジョブや I/O の種類が増えれば，定義しなければならない状態の数も膨大なものとなる．たとえば異なる5個のジョブと5個の I/O では 2,520 状態が存在し，(2.1)の連立一次方程式は普通のやり方では解くのは難かしい．

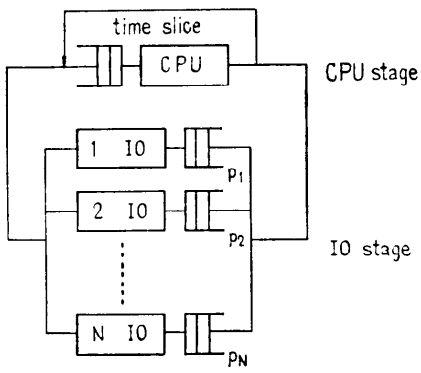

Fig. 1 Time slice model

3. Processor Shared model

3.1 Processor Shared model の平衡確率分布

2.1 の物理モデルにおいてタイムスライスの値を0に近づけた仮想的なモデルを Processor Shared model（以下 PS model）と呼ぶ，この場合は CPU ステージに存在するジョブは同時に CPU を割り当てられるが，ジョブの連続 CPU 使用時間は同時に存在しているジョブの数だけ延ばされる．CPU ステージに唯一つ存在する時の連続 CPU 使用時間の分布が $F(t,1)=1-e^{-\alpha t}$ で表わされるとすると，n 個のジョブが CPU ステージに存在する時には $F(t,n)=1-e^{-\alpha t/n}$ となる．

従って I/O に存在するジョブの順列で状態を定めると，ジョブの連続 CPU 使用時間の分布が常に指数分布をなすので，システムの状態は有限マルコフ連鎖を

なす．このマルコフ連鎖では，任意の状態から他の任意の状態へ推移が可能であるから，既約であり唯一の平衡確率分布を持つ，記号の定義を次の様におこなう．

m: I/O ステージに存在するジョブの数．

m_i: i I/O に存在するジョブの数．

M_i: i I/O のジョブの順列，この I/O は右端のジョブを処理している．$1 \leqq i \leqq N$

J_k: システムに存在するジョブ

Jl_i: M_i の左端のジョブ

a_k: J_k の平均連続 CPU 使用時間の逆数

b_i: iI/O の平均サービス率

$\boldsymbol{M}=(M_1, \cdots, M_i, \cdots, M_N)$ として

$\boldsymbol{C}(\boldsymbol{M})$: 状態 $\boldsymbol{M}$ で CPU ステージに存在するジョブ．

$\boldsymbol{M}(M_i(+J_k))$: 状態 $\boldsymbol{M}$ の M_i の右端にジョブ J_k を加えた状態．$k \in \boldsymbol{C}(\boldsymbol{M})$ である．

$\boldsymbol{M}((-Jl_i)M_i)$: 同様に M_i の左端の Jl_i を除いた状態．

$\boldsymbol{B}(\boldsymbol{M})$: $\boldsymbol{M}$ で稼動中の I/O の集合

システムが平衡状態において $\boldsymbol{M}$ に存在する確率を $P_m(\boldsymbol{M})$ として $\boldsymbol{M}$ に関する平衡方程式は

$$\sum_{k \in \boldsymbol{C}(\boldsymbol{M})} \sum_{i=1}^{N} \frac{a_k p_i}{n-m} P_m(\boldsymbol{M}) + \sum_{i \in \boldsymbol{B}(\boldsymbol{M})} b_i P_m(\boldsymbol{M}) = \sum_{k \in \boldsymbol{C}(\boldsymbol{M})} \sum_{i=1}^{N} b_i P_{m+1}(\boldsymbol{M}(M_i(+J_k))) + \sum_{i \in \boldsymbol{B}(\boldsymbol{M})} \frac{a_{l_i} p_i}{n-m+1} P_{m-1}(\boldsymbol{M}((-Jl_i)M_i)) \quad (3.1)$$

ただし a_{l_i} は Jl_i の平均連続 CPU 使用時間の逆数であり $1 \leqq m \leqq n-1$ である．

すべてのジョブが CPU ステージに存在する状態に関する平衡方程式は，その確率を P_0 として

$$\sum_{k \in \boldsymbol{C}(\boldsymbol{M})} \sum_{i=1}^{N} \frac{a_k p_i}{n} P_0 = \sum_{k \in \boldsymbol{C}(\boldsymbol{M})} \sum_{i=1}^{N} b_i P_1(\boldsymbol{M}(M_i(+J_k))) \quad (3.2)$$

すべてのジョブが I/O ステージに存在する状態に関する平衡方程式は

$$\sum_{k \in \boldsymbol{B}(\boldsymbol{M})} b_i P_n(\boldsymbol{M}) = \sum_{k \in \boldsymbol{B}(\boldsymbol{M})} a_{l_i} p_i P_{n-1}(\boldsymbol{M}((-Jl_i)M_i)) \quad (3.3)$$

ところで CPU は PS 規律をとるのでジョブは入った順序に関係なく独立に CPU ステージから I/O ステージに入る．I/O ステージではジョブに関係なく i I/O に分岐し FIFO の指数サービスを受けて CPU ステージに戻る．従って I/O ステージから出る時もジョブは独立に出てゆき，着目するジョブが i I/O に入る確率とそのジョブが I/O ステージから出る確率は等しいことが考えられる．これを証明する．

定理 PS model の平衡確率は次式を満足する．

$$\frac{a_k p_i}{n-m} P_m(\boldsymbol{M}) = b_i P_{m+1}(\boldsymbol{M}(M_i(+J_k))) \quad (3.4)$$

$0 \leqq m \leqq n-1$

証明 (3.4)において $m=0$ とすれば，P_0 に関する平衡方程式(3.2)を満足していることは明らかである．

(3.4)が成立すれば，$\boldsymbol{M}(M_i(+J_k))$ の i I/O の右端に更に J_j を加えた状態 $\boldsymbol{M}(M_i(+J_kJ_j))$ の平衡確率は，J_j の平均連続 CPU 使用時間の逆数を a_j として

$$P_{m+2}(\boldsymbol{M}(M_i(+J_kJ_j))) = \frac{a_k a_j}{(n-m)(n-m-1)} \times \left(\frac{p_i}{b_i}\right)^2 P_m(\boldsymbol{M})$$

で $\boldsymbol{M}(M_i(-J_jJ_k))$ の平衡確率と等しい．すなわち I/O における組み合わせが同じならば順序に関係なく同じ確率を持つ．従って

$$\frac{a_{l_i} p_i}{n-m+1} P_{m-1}(\boldsymbol{M}((-Jl_i)M_i)) = b_i P_m(\boldsymbol{M}((-Jl_i)M_i(+Jl_i))) = b_i P_m(\boldsymbol{M}) \quad (3.5)$$

(3.4)は(3.1)の左右の第一項が等しいことを示し，(3.5)は(3.1)の左右の第二項が等しいことを示している．また(3.5)は $m=n$ とすれば(3.3)と同じとなる．従って(3.4)は(3.1)，(3.2)，(3.3)の平衡方程式を満足する． (証明終り)

系 平衡確率分布の一般形は次式で与えられる．

$$P_m(\boldsymbol{M}) = P_0 \frac{\prod_{i \in \boldsymbol{B}(\boldsymbol{M})} \prod_{k \in M_i} \left(\frac{a_k p_i}{b_i}\right)}{\frac{n!}{(n-m)!}} \quad (3.6)$$

証明 (3.4)から明らかである． (証明終り)

以上のことから任意のジョブの組み合わせについての平衡確率分布を求める．

c: ジョブの組み合わせ

S_m: 異なる n 個のジョブの中から m 個のジョブを選ぶ組み合わせでつくる集合

d: c の N 個の I/O への分割

d_i: d の i I/O のジョブの組み合わせ

$\boldsymbol{D}(c)$: d の集合

分割 d であらわされる状態に平衡状態においてシステムが存在する確率は

$$P_m(d)=P_0\prod_{i=1}^{N}(m_i!)\cdot\prod_{i=1}^{N}\prod_{j\in d_i}\left(\frac{a_j p_i}{b_i}\right) \tag{3.7}$$

m 個のジョブが I/O ステージに存在する確率は

$$P_m=\sum_{c\in S_m}\sum_{d\in D(c)}P_m(d) \tag{3.8}$$

P_0 は $\sum_{m=0}^{n}P_m=1$ より求められる．このシステムの CPU 利用率は $1-P_n$ である．

次に CPU ステージから特定の I/O へゆく確率が I/O によらず等しく，I/O のサービス時間がすべての I/O について同一の分布に従うときは，$b_i=b$, $p_i=1/N$ として(3.8)は，

$$\begin{aligned}P_m&=\sum_{c\in S_m}\sum_{d\in D(c)}P_m(d)\\&=\frac{\alpha(N,m)P_0}{\dfrac{n!}{(n-m)!}}\sum_{c\in S_m}\left(\prod_{j\in c}\left(\frac{a_j}{bN}\right)\right)\end{aligned} \tag{3.9}$$

$\alpha(N,m)$ は異なる m 個のジョブを異なる N 個の I/O にならべるやり方の数で次式で与えられる．

$$\alpha(N,m)=(N+m-1)!/(N-1)!$$

(3.9)の P_m 算出のためには $\binom{n}{m}$ のくりかえし計算が必要であるから，平衡確率分布を求めるためには $\sum_{m=0}^{n}\binom{n}{m}=2^n$ 回のくりかえし計算が必要である．

3.2 バランスジョブ

ここですべてのジョブの連続 CPU 使用時間も同一の分布に従う場合は

$a_j=a$, $a/bN=x$ として，(3.9)は

$$\begin{aligned}P_m&=P_0\frac{(N+m-1)!}{(N-1)!}\frac{(n-m)!}{n!}\binom{n}{m}x^m\\&=P_0\binom{N+m-1}{m}x^m\end{aligned} \tag{3.10}$$

一つのジョブの CPU 利用率 CPU (N,n,x) は，

$$\mathrm{CPU}(N,n,x)=\sum_{m=0}^{n-1}P_m\Big/\left(n\cdot\sum_{m=0}^{n}P_m\right) \tag{3.11}$$

ジョブの CPU 利用率と I/O 利用率との和を**実質システム利用率** RSUT(N,n,x) と定義すると，

$$\mathrm{RSUT}(N,n,x)=\mathrm{CPU}(N,n,x)(1+a/b)$$

この値が大きい程，システムはより効率良くジョブを処理出来る．言いかえればシステムにとって負担の少ないジョブであることを意味している．この値は $x=1$，即ち $a=bN$ の時最大値を持つことを示す．

$\sum_{i=0}^{b}\binom{a+i}{i}=\binom{a+1+b}{b}$ を用いると，

$$\mathrm{RSUT}(N,n,1)=\frac{N+1}{N+n} \tag{3.12}$$

である．従って

$$\begin{aligned}&\mathrm{RSUT}(N,n,1)-\mathrm{RSUT}(N,n,x)\\&=\frac{(1+N)n\sum_{m=0}^{n}\binom{N+m-1}{m}x^m-(1+Nx)}{(n+N)n\sum_{m=0}^{n}\binom{N+m-1}{m}}\\&\quad*\frac{(N+n)\sum_{m=0}^{n-1}\binom{N+m-1}{m}x^m}{x^m}\\&=\frac{\sum_{m=0}^{n-1}\frac{(N+m)!}{(N-1)!\,m!}(n-1-m)x^m}{(n+N)n\sum_{m=0}^{n}\binom{N+m-1}{m}x^m}(x-1)^2\geqq 0\end{aligned}$$

確かに $x=1$ で最大値をとる．

付録に示す様に，この関係を持つジョブは CPU と I/O ステージにおけるユニット使用時間と待ち時間の比が常に等しく，ジョブがそれぞれのステージにかたまってたまることがない．これ故にこの様なジョブを**バランスジョブ**と呼ぶことにする．バランスジョブが n 個存在するとき，一つのジョブの CPU 利用率は(3.11)より

$$\mathrm{CPU}(N,n,1)=\frac{1}{N+n} \tag{3.13}$$

である．

3.3 Processor Shared model に関する補題

ユニットでジョブがサービスを受けている間に，同じステージに存在する他のジョブを待たせる時間の和を「**待たせ時間**」と定義する．

平衡状態にある PS model の性質として次の補題1が成り立つ．

補題1 PS model の各ステージでのジョブの「待ち時間」と「待たせ時間」は等しい．

証明 PS 規律の CPU は平等にジョブをサービスするので，t 時間に n 個のジョブが存在すればサービス時間は t/n，「待たせ時間」は $t/n\times(n-1)$ で，これは「待ち時間」に等しい．I/O に関しても **3.1** で明らかな様に同じ組み合わせならばジョブの順序が違っても同じ平衡確率を持つので，同じ組み合わせのジョブは平等にサービスを受けることになり CPU ステージと同じことがいえる． (証明終り)

以上から PS model では各ステージの「待ち時間」の和と「待たせ時間」の和は等しい．

ここでジョブの CPU 使用時間と I/O 使用時間の

和を「**実質システム使用時間**」と定義する.

これによって補題1よりPS modelの「待たせ時間」は経過時間から「実質システム使用時間」を除いたものとして表わすことができる.

次にCPUからI/Oへの推移確率がすべて等しく,I/Oのサービス時間がI/Oに関係なく同一の分布に従うならば,次の補題2,補題3が成り立つ.

補題2 PS modelに1個のバランスジョブと$n-1$個の任意のジョブが存在するとき,バランスジョブのCPU利用率は,他のジョブに関係なく$1/(N+n)$である.

証明記号を次の様に定める.

S_m': S_mの中でバランスジョブを含む組み合わせを除いた集合

S_m'': S_mの中でバランスジョブを含む組み合わせの集合 $S_m'+S_m''=S_m$

$P(S_m')$: 平衡状態において,S_m'で表わされる状態にシステムが存在する確率

$P(S_m'')$: 同様にS_m'で表わされる状態にシステムが存在する確率.

(3.9)より

$$P(S_m')=\frac{\alpha(N,m)P_0}{\dfrac{n!}{(n-m)!}(bN)^m}\sum_{c\in S_m'}\prod_{j\in c}a_j$$

バランスジョブに関して$a=bN$が成立するから

$$P(S_{m+1}'')=\frac{\alpha(N,m+1)P_0}{\dfrac{n!}{(n-m-1)!}(bN)^{m+1}}\sum_{c\in S_{m+1}''}\prod_{j\in c}a_j$$

$$=\frac{\alpha(N,m+1)P_0}{\dfrac{n!}{(n-m-1)!}(bN)^m}\sum_{c\in S_m'}\prod_{j\in c}a_j$$

ここで$\alpha(N,m+1)=\alpha(N,m)(N+m)$の関係から

$$=\frac{N+m}{n-m}P(S_m')$$

バランスジョブがCPUステージに存在するとき,実際にバランスジョブにCPUが与えられるのは$P(S_m')/(n-m)$であるから,このバランスジョブのCPU利用率は

$$\frac{\sum_{m=0}^{n-1}\dfrac{P(S_m')}{n-m}}{\sum_{m=0}^{n-1}P(S_m')+\sum_{m=1}^{n}P(S_m'')}=\frac{1}{N+n}$$

である. (証明終り)

この値は(3.13)よりn個がすべてバランスジョブであるときの一個当りのCPU利用率である.

補題3 PS modelで$n-1$個のバランスジョブと任意のジョブが1つ存在するとき,全てのジョブの実質システム利用率は等しく$(N+1)/(N+n)$である.

証明 バランスジョブに関しては補題2より実質システム利用率は$(N+1)/(N+n)$で明らか,バランスジョブ以外のジョブをJ_1として記号を次の様に定める.

S_m': S_mの中でJ_1を含む組み合わせを除いた集合.

S_m'': S_mの中でJ_1を含む組み合わせの集合.

$P(S_m')$, $p(S_m'')$は補題2と同じである.

$J_2\sim J_n$がバランスジョブだから

$$P(S_m')=\frac{\alpha(N,m)P_0}{\dfrac{n!}{(n-m)!}}\binom{n}{m}\left(1-\frac{m}{n}\right)$$

$$=P_0\binom{N+m-1}{m}\left(1-\frac{m}{n}\right)$$

$$P(S_m'')=\frac{\alpha(N,m)P_0}{\dfrac{n!}{(n-m)!}}\binom{n}{m}\frac{m}{n}\frac{a_1}{bN}$$

$$=P_0\binom{N+m-1}{m-1}\frac{a_1}{bn}$$

従ってJ_1の実質システム使用率は

$$\frac{\sum_{m=0}^{n-1}\dfrac{P(S_m')}{n-m}\left(1+\dfrac{a_1}{b}\right)}{\sum_{m=0}^{n-1}P(S_m')+\sum_{m=1}^{n}P(S_m'')}$$

$$=\frac{\dfrac{(N+n-1)!}{N!\,n!}\left(1+\dfrac{a_1}{b}\right)}{\dfrac{(N+n)!}{(N+1)!\,n!}+\dfrac{(N+n)!}{(N+1)!\,n!}\dfrac{a_1}{b}}=\frac{N+1}{N+n}$$

これはバランスジョブの実質システム利用率であるからすべてのジョブの実質システム利用率は等しく,$(N+1)/(N+n)$である. (証明終り)

3.4 シミュレーションとの比較

タイムスライススケジューリングを含むモンテカルロシミュレーションと(3.7)で得られるPS modelの平衡確率分布による個々のジョブのCPU利用率の比較をおこなう.両モデルは**Fig. 1**において$N=2$, $p_1=p_2=0.5$としジョブの多重度は3で,J_1, J_2, J_3とする.I/Oサービス分布はI/Oに関係なく同一で平均100 msecの指数分布である.連続CPU使用時間も指数分布である.シミュレーションモデルにおけるタイムスライスの値は50 msecとした.FIFOの場合も参考のために載せた.(**Table 1** 次頁参照)

平均連続CPU使用時間の違いが大きくなる程タイ

Table1 The CPU Utilization Comparison
(TS=time slice, FIFO=first-in first-out, PS=processor share)

CPU burst av. (msec)			J_1 CPU utili.			J_2 CPU utili.			J_3 CPU utili.		
J_1	J_2	J_3	TS	FIFO	PS	TS	FIFO	PS	TS	FIFO	PS
50	25	25	0.203	0.217	0.200	0.117	0.111	0.118	0.117	0.111	0.118
50	25	50	0.202	0.209	0.200	0.117	0.107	0.120	0.202	0.209	0.200
50	25	100	0.200	0.187	0.200	0.118	0.096	0.123	0.312	0.361	0.307
50	25	200	0.200	0.148	0.200	0.118	0.076	0.127	0.438	0.551	0.420
50	25	500	0.200	0.086	0.200	0.119	0.044	0.130	0.565	0.772	0.540
50	50	50	0.200	0.200	0.200	0.200	0.200	0.200	0.200	0.200	0.200
50	50	100	0.199	0.179	0.200	0.199	0.179	0.200	0.307	0.347	0.300
50	50	200	0.195	0.142	0.200	0.195	0.142	0.200	0.410	0.532	0.400
50	50	500	0.194	0.083	0.200	0.194	0.083	0.200	0.515	0.753	0.500
50	100	100	0.195	0.161	0.200	0.295	0.312	0.291	0.295	0.312	0.291
50	100	200	0.192	0.129	0.200	0.281	0.252	0.282	0.383	0.487	0.376
50	100	500	0.189	0.078	0.200	0.275	0.153	0.274	0.470	0.712	0.456
50	200	200	0.189	0.107	0.200	0.359	0.405	0.355	0.359	0.405	0.355
50	200	500	0.184	0.069	0.200	0.338	0.262	0.337	0.435	0.633	0.420
50	500	500	0.181	0.050	0.200	0.400	0.467	0.390	0.400	0.467	0.390

ムスライスシミュレーションと PS model での値との差が大きくなることがわかるが，通常の範囲ではFIFOと比較してかなり良い近似となっている．

4. マルチプログラミングにおけるジョブ負荷評価への適用

複数のジョブが，マルチプログラミングシステムで同時に処理されているとき，個々のジョブがシステムに対して持つ負荷を考えてみよう．ジョブの負荷の基本として個々のジョブが計算機を使用した時間を単位にとることに異論はないと考えられる．

従って，ある経過時間 t において，システムが n 個のジョブを同時に処理するとき，ジョブの負荷を代表する適当な評価子によって，t を n 個のジョブに配分できればよい．

本稿では，先に定義した「実質システム使用時間」と「待たせ時間」の和をジョブ負荷評価子とする．ジョブ J_i のそれぞれのステージでの待たせ時間を CPUWD$_i$, IOWD$_i$ 及び，CPU, I/O 使用時間を CPU$_i$, IO$_i$ とすると J_i．負荷評価子 ld_i は，

$$ld_i=\mathrm{CPU}_i+\mathrm{IO}_i+\mathrm{CPUWD}'+\mathrm{IOWD}_i. \quad (4.1)$$

となる．

つまり実際にユニットを使用した時間が長い程，及び連続的にユニットを使用する時間が長い程，ジョブは大きい負荷を持つと考える．明らかに

$$nt=\sum_{i=1}^{n} ld_i$$

が成立し，J_i に関するジョブ負荷 JL_i を次式と定義する．

$$JL_i=\frac{ld_i}{\sum_i ld_i}\times t \quad (4.2)$$

たとえば多重度が1のときには，CPUWD$_i$=IOWD$_i$=0 であるから，ld_i は実質システム使用時間に等しくなり，現実と一致する．

そこで，タイムスライススケジューリングを持つシステムを PS model で近似して，(4.1)からジョブ負荷評価子を求める．この値は一緒に処理されているジョブに依存するので，負荷評価子を求めるためには，相手のジョブを特定しなければならない．しかし計算センタのような処理形態をとるところでは，ジョブはランダムに入力され，同時に走っているジョブを知ることは難かしいし，たとえ可能でも，負荷が状況によって変わることになり，ジョブの負荷評価としてはあまり意味を持たないと考えられる．従って，現実的には，一緒に走るジョブをあらかじめ決めておく必要があり，それを個々のセンタで平均的な CPU 比率を持つジョブとするのが適当であろう．

そこで J_2〜J_n のジョブをすべて同じバックグランドジョブとして J_1 の CPU 使用時間 CPU$_1$ と I/O 使用時間 IO$_1$ を与えて J_1 の負荷を求める．

補題1により PS model の「待ち時間」と「待たせ時間」は等しいので，すべてのジョブの負荷評価子は経過時間となり等しくなる．故に経過時間 t におけるジョブの負荷は(4.2)より

$$JL_1=\frac{t}{n} \quad (4.3)$$

この値は(3.8)により I/O に関する任意のシステム構成と任意のバックグランドジョブに対して J_1 の CPU 利用率を求めることができるから，J_1 の「実質システム使用時間」を実現する経過時間を求められる．

ここでは I/O への推移確率が等しく I/O サービス時間の分布がすべて同一の分布に従う場合を考える．

J_1 の「実質システム使用時間」を実現する経過時間は補題3と同じ様に求められる．バックグランドジョブに関して $a_j/bN=x$, $(j=2,\cdots,n)$ とする．

I/O ステージに m 個のジョブが存在するとき，J_1 が CPU ステージに存在する確率 $P(S_m')$ は(3.9)より

$$P(S_m')=P_0\binom{N+m-1}{m}\left(1-\frac{m}{n}\right)x^m$$

同様に J_1 が I/O ステージに存在する確率 $P(S_m'')$ は

$$P(S_m'')=P_0\binom{N+m-1}{m-1}\frac{a_1}{bn}x^{m-1}$$

a_1/b は IO_1/CPU_1 で与えられる．

以上から J_1 の実質システム利用率は

$$RSUT_{J_1}=\frac{\sum_{m=0}^{n-1}\frac{P(S_m')}{n-m}\left(1+\frac{a_1}{b}\right)}{\sum_{m=0}^{n-1}P(S_m')+\sum_{m=1}^{n}P(S_m'')} \qquad (4.4)$$

これを使って(4.3)より J_1 のジョブ負荷は

$$JL_1=\frac{t}{n}=\frac{CPU_1+IO_1}{RSUT_{J_1}}\times\frac{1}{n} \qquad (4.5)$$

である．

特別な場合としてバックグランドジョブがバランスジョブならば (4.5) はいちじるしく簡単になる．補題3によれば $RSUT_{J_1}=(N+1)/(N+n)$ であるから

$$JL_1=(CPU_1+IO_1)\frac{N+n}{(N+1)n} \qquad (4.6)$$

このとき JL_1 は J_1 の CPU 比率に関係なくなる．

実際のジョブ負荷評価に(4.3)を適用するときは，一緒に存在しているジョブの数 n は例えばプログラムサイズに依存するものとしてシミュレーション等によって平均的な値を与えれば良い．

5. おわりに

1個の CPU と N 個の I/O で構成する循環待行列において，タイムスライスの値を0に近づけた PS model で，連続 CPU 使用時間が同一の分布に従わない不均一なジョブを処理する場合の平衡確率分布を定式化した．同時に PS model に関する2,3の性質も明らかにした．

従来の解析的手法ではタイムスライスモデルはモデルが少し大きくなると事実上平衡確率分布の計算が不可能となるが，この PS model はタイムスライススケジューリングのシミュレーションと比較しても，タイムスライススケジューリングを持つバッチシステムを十分近似しており，計算も容易であることを示した．次にマルチプログラミングシステムにおけるジョブ負荷評価式を定義し，タイムスライススケジューリングをとる場合について PS model で近似してその計算式を与えた．この式はタイムスライススケジューリングを持つマルチプログラミングシステムでのジョブに対する課金や負荷に応じて動的に優先権を変更する様なスケジューリングに適用が可能である．

謝辞 終始御指導をいただいた鉄道技研山本一郎システム研室長，森英一計算センタ室長，ならびに適切な助言をいただいた三浦武，野末尚次主任研究員に深謝します．

参考文献

1) W.J. Gordon and G.F. Newell: Closed queueing systems with exponential servers, Opr. Res., Vol. 15, No. 2, pp. 254～265 (1967).
2) J.P. Buzen: Computational Algorithms for Closed Queueing Networks with Exponential Servers, C. ACM, Vol. 16, No. 9, pp. 527～531 (1973).
3) S. Sherman, etc: Trace-Driven Modeling and Analysis of CPU scheduling in a Multiprogramming, C. ACM, Vol. 15, No. 12, pp. 1063～1069 (1972).
4) James D. Foley: A Markovian Model of the University of Michigan Executive System, C. ACM, Vol. 10, No. 9, pp. 584～588 (1967).

(昭和50年7月8日受付)
(昭和50年10月15日再受付)

付 録

CPU ステージで待っているジョブが着目するジョブである確率は $(n-m-1)/n$ であるから，CPU でのジョブの待ち確率は，(3.10)において $x=1$ として，

$$W(CPU)=\sum_{m=0}^{n-1}P_m\frac{n-m-1}{n}=\frac{1}{N+1}\frac{n-1}{N+n}$$

I/O ステージでの待ち確率は，全体から CPU 利用率，CPU 待ち確率，I/O 利用率を除いたものであるから，(3.13)より

$$W(IO)=\frac{N}{N+1}\frac{n-1}{N+n}$$

故にいずれの場合も使用時間と待ち時間の比は，$(n-1)/(N+1)$ となり，常に等しい．