Vol. 23 No. 5　　情報処理学会論文誌　　Sep. 1982

# 計算機群同時停止のためのバッチ・ジョブ・スケジューリング†

浅井　清††　高橋國夫†††　藤井　実††

与えられたバッチ・ジョブ群を複数計算機で処理し，これら計算機をほとんど同時停止させるためのジョブ・スケジューリングの1手法を提案した．この手法はラウンド・ロビン法にもとづくもので，ジョブが起動されるごとにジョブ多重度が決定される．各ジョブの演算装置時間，入出力回数は既知と仮定されている．この手法による予測計算結果はシミュレーション・プログラムによるものと比較的よい一致を示す．

## 1. はじめに

筆者らが勤務する日本原子力研究所における計算機システムの運用方法では，昼間においては3システムがそれぞれ大型ジョブ処理，中小型ジョブ処理，およびタイムシェアリング処理の異なった用途に当てられている．夜間における運用はこれとはまったく異なり，3システムはバッチ・ジョブ専用となる．このときこれらの3システムの運用で望ましい条件は

（1） 3システムがほとんど同時にジョブ終了を完了し，同時に電源切断が可能なようにジョブのスケジューリングをおこなうこと，ジョブ処理が明朝9時までに完了しないときは，9時までに完了できる範囲でジョブ・スケジューリングをおこない，昼間の運用をさまたげないこと，

（2） 夜間ジョブのスケジューリングは，演算装置（以下 CPU と略す），入出力装置（以下 I/O と略す）などの計算機資源をできるだけ有効利用するようおこなうこと，

などである．

このためにはジョブ実行多重度の影響を精度よく推定する必要がある．従来このジョブ実行多重度はジョブ処理の環境を考慮した適当な値が感覚的に選ばれている．しかしこの多重度はオペレイタ・コンソールからでも，あるいは近年どの計算機システムにおいても開発されつつあるオペレイタ・コンソール自動応答機能ソフトウェアによっても変更できるので，処理すべきジョブ・パターンに応じて多重度を変化させることが可能となった．

そこで次に問題となるのはどのようなジョブ・パターンのときに多重度をどのように変化させればよいかということである．ジョブ多重度やバッチ処理の多重度による時間遅れについては，これまでもいくつかの論文[1)~6)]で議論されているが，多重度固定とかグラフで多重度と CPU 使用率の近似値を求めるものが多い．

本論文で筆者らは，一回の I/O 処理時間は常に一定，各ジョブについて CPU 時間と I/O 回数は既知と仮定したとき，ロールイン/アウトのない実記憶計算機システムのバッチ処理において，ⅰ）時々刻々のジョブ処理状況，ⅱ）ジョブ終了ごとの次の最適ジョブ多重度，ⅲ）全ジョブ処理完了時刻，を推定できる変形ラウンド・ロビン・モデルを提案した．

野中と大野[7)]はオンライン・システムのマルチタスク処理における時間おくれを近似する方法を一般的な形式で提案している．筆者らの論文の大部分はおそらく野中と大野による方法の特殊な場合となっているものと思われる．ただ野中と大野の方法では各タスクに対するトランザクションの到着率は与えられているという前提がある．しかしこれは一般には不明であるから，筆者らの方法ではこの到着率を求めることも目的のひとつとなっている．

原田と箱崎[8)]は全ジョブの CPU と I/O 時間が事前に判明しているバッチ処理なら平均膨張率という概念を導入することによって，CPU 使用率とジョブ多重度の関係を簡便な方法で精度よく推定できることを示した．ただし，この方法では個別的なジョブ処理状況を見ることはできない．われわれの方法は，任意の時点のジョブ処理状況，すなわち各ジョブが獲得した

† A Batch Job Scheduling Method to Complete Job Processing of Multiple Computers at Same Time by KIYOSHI ASAI (Japan Atomic Energy Research Institute), KUNIO TAKAHASHI (Japan Software Development, Ltd.) and MINORU FUJII (Japan Atomic Energy Research Institute).

†† 日本原子力研究所
††† 日本ソフトウェア開発

CPU 時間，I/O 回数，各ジョブの残り CPU 時間，I/O 回数および最終的には全ジョブの処理終了時刻が推定できる点が異なっている．

Brandwajn[9] は，仮想記憶システムのバッチ処理でページ・フォールトとユーザ・プログラムのファイル入出力が多いとき，多重度と CPU 使用率の関係を各種のパラメータについて数値実験をおこない，i）ページ・フォールトのオーバヘッドは CPU 使用率にさほど影響を与えないこと，ii）ページング・ディスクやユーザ・プログラムのファイル入出力の平均サービス時間の多少は CPU 使用率に大きな影響があることを示した．また Denning *et al.*[10] は Thrashing などの多い状況における仮想記憶システムでページ・イン/アウト時間間隔の3種の制御方法について数値実験をおこない，それぞれの方法がジョブ特性に応じて最高のスループット，ジョブ多重度を与える範囲を数値計算で求め，多少の多重度の変化では最高，あるいは最高に近いスループット値が変化しない制御方法の強固さ（robustness）について論じている．われわれが本論文で対象としているのは実記憶システムにおけるバッチ処理で，しかもロールイン/アウトはないと仮定した．これは対象とする夜間ジョブが CPU 寄りのものが多く，多重度をさほど上げなくてもユーザ側の CPU 使用率が1に近づくためである．もちろん一般のジョブ・パターンではこのようなことはいえないから，われわれの方法を仮想記憶システムに適用したときの問題点は今後の研究課題として残されている．

なお Brandwajn, Denning *et al.* らが数値実験に使用したモデルは，通常の central server や cyclic queue タイプのもので，本論文で筆者らが提案したモデルのような特長を備えていない．

## 2. ジョブの計算機資源使用量の予測可能性

現在，筆者らの計算センタではジョブは利用者が入力時に指定した計算機資源使用の設定値（上限値）にもとづいてスケジューリングされ，処理されている．しかし，利用者の設定した上限値と実際にジョブが使用する資源量とにはかなりの差がある．

筆者らは，計算機に利用者の計算機使用情報に関する記憶をもたせれば，ジョブ処理前に実際にジョブが使用する資源量を，精度よく推定できることを示した[11]．利用者の計算機使用情報に関する記憶には，過去のジョブ処理情報数件と平均値のデータが使用されている．

表1 ジョブの計算機資源使用量の予測精度
Table 1 Accuracy of prediction of computer resource use for jobs.

| | 予測精度1 | | | 予測精度2 | | |
|---|---|---|---|---|---|---|
| | CPU 時間 | メモリ量 | I/O 回数 | CPU 時間 | メモリ量 | I/O 回数 |
| 予測しない場合 | 42% | 78% | 25% | 42% | 78% | 25% |
| 予測した場合 | 84% | 95% | 94% | 98.4% | 99.9% | 99.9% |

（注） 予測しない場合は，予測値に利用者が設定した上限値を用いた．予測した場合は，過去のデータから資源使用量を予測した値を用いた．

表1にその計算機資源使用量の予測結果を示す．以下の式で，$n$ は対象としたジョブ件数で，筆者らの計算センタの大口利用者30人の昭和55年12月分4062件である．

予測精度1

$$=\left\{1-\left(\sum_{i=1}^{n}\frac{|\text{予測値}-\text{実際使用値}|}{\text{設定値}}\right)\Big/n\right\}\times 100\%$$

予測精度2

$$=\left\{1-\left(\sum_{i=1}^{n}\frac{\text{予測値}-\text{実際使用値}}{\text{設定値}}\right)\Big/n\right\}\times 100\%$$

とする．

1夜分の夜間ジョブについて行った予測の精度は，CPU 時間について予測精度 1=84±12%，予測精度 2=98.4+0.6% から 98.4−3.4% 程度となる．

昭和 54 年にも同様の調査を6か月間おこないほぼ同じ結果を得ている．

## 3. スケジューラ

### 3.1 モデルと仮定

前章でも述べたように大型計算機システム群に入力される個々のジョブの特性（ここでは CPU 時間 $a_j$，I/O 回数 $b_j$）は1か月程度の短期間では変わらない．したがって過去数回のジョブ処理から $a_j, b_j$ を精度よく予測することができる．しかし入力されるジョブの組合せは毎夜変化するので計算機システムにとっては毎夜異なったジョブ処理となる．ここでのべるスケジューラの手法は Round Robin（以下 RR と略す）法として知られている古典的な方法を少し修正したものである．RR 法（図1の CPU 系の破線部分）においてはジョブは到着率 $\lambda$（単位時間当りのジョブ到着個数）で CPU サービスを要求する．サービスはひとつのまとまった CPU 時間 $Q$（これをタイム・クワンタムという）単位でおこなわれ，時間 $Q$ で終わらないジョブは CPU サービスを待っている行列の最後に

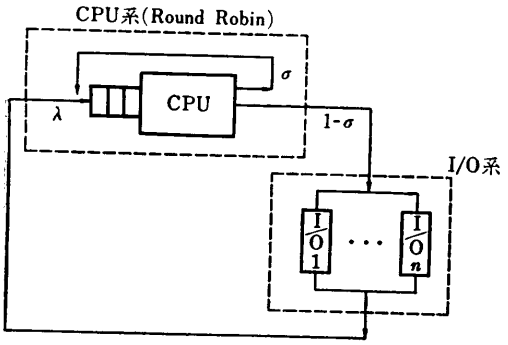



図 1 ラウンド・ロビン形式モデル
Fig. 1 Round-Robin type model.

ついて再びサービスを待つ．ただしこのフィードバック・ループは新しいジョブの到着とはみなさない．すなわち到着率 $\lambda$ に寄与するとはみなさない．

この方法を考えるにあたって次の仮定をした．

(a) 入力されるジョブの CPU 時間と I/O 回数は既知であり，どのジョブも同一の優先順位をもち，ジョブ間の順序関係はない．

(b) 起動されたジョブは主記憶から追い出されることはない．

(c) オペレイティング・システムのオーバヘッドはない．CPU は1台である．

(d) 使用可能な実記憶量を超えない範囲でジョブ多重度を決定する．

(e) I/O 1回当りに要する時間は固定であり，I/O 待ちは発生しない．

### 3.2 計算の方法

(0) ジョブ $J_i$ の CPU 時間，I/O 回数，1回の I/O 時間をそれぞれ $a_i, b_i, c$ としたとき，$J_i$ の実行時間を $a_i+b_i\cdot c$ とし，与えられたジョブの集合を LPT (largest processing time job first) ルールで3システムに配分する．以下(1)～(9)は1システムについての議論である．

(1) RR 法の特長は CPU サービス時間分布を陽に計算できることである．図1において，CPU サービスを受けたジョブがフィードバックされる確率を $\sigma$ とすると，あるジョブが $i$ 回サービスを受ける確率 $g_i$ は

$$g_i=\sigma^{i-1}(1-\sigma), \quad i=1,2,\cdots \tag{1}$$

で与えられる．

サービス時間分布の期待値 $E$ は

$$E=\sum_{i=1}^{\infty}(iQ)g_i=Q/(1-\sigma). \tag{2}$$

また CPU 系への到着率を $\lambda$ とすると，CPU 使用率 $\rho$ は

$$\rho=\lambda E \tag{3}$$

となる．

(2) このσをシステムに入力されているジョブの特性から割り出すことができればシステムのジョブ処理性能を推定することができる．$c$ を1回の I/O 時間とし，いま $M$ 個のジョブ

$$J_i=(a_i, b_i\cdot c), \quad i=1,2,\cdots,M \tag{4}$$

が1多重で図1のシステムで実行されるとするとCPUへの到着率 $\lambda_i$ は近似的に

$$\lambda_i=(b_i+1)/(a_i+b_i\cdot c) \tag{5}$$

とかける．$(b_i+1)$ の $+1$ は系への最初のジョブ到着を示す．ここで各ジョブ $J_i$ は到着率がそれぞれ $\lambda_i$ のポアソン分布にしたがって CPU 要求をおこなうと仮定しよう．このひとつの CPU 要求を擬ジョブ $J_i$ と呼ぶことにする．

$$Q=\min(a_i/(b_i+1)), \quad i=1,\cdots,M \tag{6}$$

$$N_i\cong(a_i/(b_i+1))/Q, \quad i=1,\cdots,M \tag{7}$$

ただし $N_i$ は $(a_i/(b_i+1))/Q$ を四捨五入した整数値とする．上記 $M$ 個のジョブが図1のシステムにおいて滞在しているとき，CPU サービスを要求する全ジョブの到着率 $\lambda$ の近似値は

$$\lambda=\sum_{i=1}^{M}\lambda_i \tag{8}$$

となる．また $\lambda^{\nu}=\sum_{i=1}^{M}N_i\lambda_i$ とすると，到着率が $\lambda$ であるときこの $\lambda^{\nu}$ はフィードバックまで勘定に入れた CPU サービス要求の率を示している．したがって図1でジョブが CPU 系から離れていく確率 $1-\sigma$ は

$$1-\sigma=\lambda/\lambda^{\nu} \tag{9}$$

で近似できる．

(3) $k$ クワンタを要求するジョブの CPU 系における滞在時間 $W_k$ の計算法は Kleinrock[12], Coffman & Denning[13] らによって与えられている．

以下(6)までは $\lambda$ についての収束計算である．$k$ クワンタを要求する擬ジョブの第1回目の到着率 $\lambda_k=\lambda_k^{(1)}$ は，実際の到着率よりも大きく出ているはずである．なぜなら各擬ジョブは多重度1で処理されるとして $\lambda_k$ を計算しているからである．

そこでこの $\lambda_k$ を使って $W_k$ を計算すると，$W_k$ の値は実際の $W_k$ よりも大きくなる．この $W_k$ で $\lambda_k$ を再計算する．この操作を繰り返す．$W_k$ の計算は Coffman & Denning[13] の導出とほとんど同じであるが，ここではシステム内に滞在する擬ジョブ数が有限であるという点が一般の RR 方式と異なる．各ジ

ョブは並行タスクを発生しないものとする．したがって現在実行中のジョブが終了しない限りはジョブ実行多重度に変化はない．

（4） $M$ をジョブ実行多重度，$k$ クワンタのサービスを要求する擬ジョブがCPU系に到着したときのシステムに滞在する擬ジョブ数が $j$ 個のときの条件つき待ち時間 $W_k(j)$ とすると，待ち時間 $W_k$ は

$$W_k=\sum_{j=0}^{M} p_j W_k(j) \tag{10}$$

ここで $\{p_j\}_{j=0}^{M}$ はCPU系に $j$ 個の擬ジョブが存在する定常確率を示す．これはCPU系に $j$ 個のジョブが存在するときの確率と同じである．図1のCPU系のみを考えると，$W_k$ はCoffman & Denning[13]によって

$$W_k=W_1+\frac{(k-1)Q}{1-\rho}+Q\left[\lambda W_1+\sigma\bar{n}-\frac{\rho}{1-\rho}\right]\frac{1-(\lambda Q+\sigma)^{k-1}}{1-(\lambda Q+\sigma)}, \tag{11}$$

ここで $\bar{n}=\sum_{j=0}^{M} p_j\cdot j$ はCPU系における平均待ち行列長である．この $\bar{n}$ は次の有限待ち行列の平均待ち行列長で近似する[14),15)]．

$$\bar{n}=\frac{1-\rho}{1-\rho^{M+1}}\sum_{n=0}^{M} n\rho^n, \quad (\rho\neq 1) \tag{12}$$

$$\bar{n}=M/2, \quad (\rho=1). \tag{13}$$

また，ある擬ジョブが待ち行列のうしろに到着したときのCPUでのサービス残り時間の期待値を $E(Q_r)$ とすると，

$$W_1=E(Q_r)+\bar{n}_q Q+Q, \tag{14}$$

ここで $\bar{n}_q$ は平均待ち行列長（サービス中の擬ジョブは含まない）で，この $\bar{n}_q$ は $\bar{n}_q=\bar{n}-\rho$ なる関係をみたす[13]．$\rho$ はCPUの使用率であって $\rho=\lambda E$ で定義される．$E(Q_r)$ は $E(Q_r)=\rho Q/2$[13]となる．

（5） このようにして $k$ クワンタを要求する擬ジョブのCPU系での滞在時間 $W_k$ が計算される．しかし，いままで展開した議論は，システム内のジョブが幾何分布にしたがってサービスを受けると仮定しているが，有限の多重度 $M$ で処理されている現実のジョブ群に適用すると合わないことが多い．それを例で示そう．いまシステム内に**表2**のジョブが滞在しているとする．(9)式の定義によって $\sigma$ の値は $1-\sigma=4/(8+4+11+1)=4/24=0.167$ となり，$g_i$ の値は図2の斜線の部分になる．**図2**において斜線以外のジョブが存在しないことから明らかなように，このままでは $W_k$ の値は正しくない．なぜなら，この例では $i=2$, 3, 5, 6, 7, 9, 10 および 12 以降のクワンタを要求する擬ジョブが存在しない．この難点を解決するためにつぎのように擬ジョブ群を細分して，$g_i$ 曲線に比例した新しい擬ジョブ群を発生させる．この細分された新しい擬ジョブ群を細分割擬ジョブ群と呼ぶことにし，その発生の方法は以下のようにする．

**表2** ジョブの例
Table 2 Example of jobs.

| ジョブ番号 | 要求クワンタ数 | 到着率 | $g_i$ 値 |
|---|---|---|---|
| 1 | 8 | 1 | 0.046 |
| 2 | 1 | 1 | 0.167 |
| 3 | 4 | 1 | 0.097 |
| 4 | 11 | 1 | 0.027 |

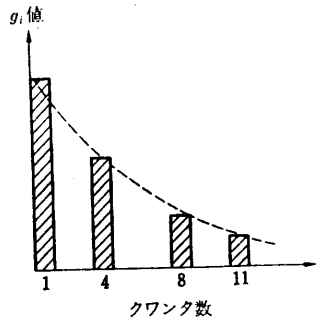



**図2** 有限個の $g_i$ 値と要求クワンタ数
Fig. 2 Finitely distributed $g_i$ values and corresponding quanta.

$$l=\sum_{i=1}^{M} N_i, \quad M \text{ はジョブ多重度} \tag{15}$$

$$S=\sum_{i=1}^{L} i\cdot g_i, \quad L=\max\{N_i \mid i=1,\cdots,M\} \tag{16}$$

$$\beta=S/l \tag{17}$$

$$m_i=g_i/\beta, \quad i=1,\cdots,L \tag{18}$$

とすると，$l$ は擬ジョブが要求する全クワンタ数，$m_i$ は $g_i$ に比例して発生させる細分割擬ジョブの個数である．この $m_i$ の定め方は次のようにする．

$m_i=\alpha g_i$ とおき，両辺に $i$ をかけて和を求めると，

$$\sum_{i=1}^{L} i\cdot m_i=\alpha\sum_{i=1}^{L} i\cdot g_i$$

であり，左辺は全クワンタ数 $l$，右辺の $\sum_{i=1}^{L} i\cdot g_i$ はジョブが平均してサービスを受ける割合 $\sum_{i=1}^{\infty} i\cdot g_i$ の近似値（$S\cdot Q$ は期待値 $E$ の近似値である）とおく．このように両辺を定めると

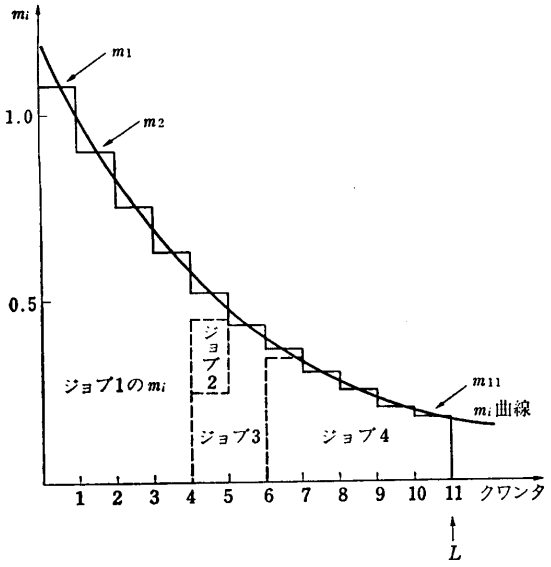



**図 3　$m_i$ 値の構成**
Fig. 3　Construction of $m_i$ values.

**表 3　許容される擬ジョブの組合せ**
Table 3　Example of allowable pseudo job combinations.

| クワンタ数 | $g_i$ 値 | $i \cdot g_i$ 値 | $m_i$ 値 | $i \cdot m_i$ 値 |
|---|---|---|---|---|
| 1 | 0.167 | 0.167 | 1.080 | 1.080 |
| 2 | 0.139 | 0.278 | 0.899 | 1.798 |
| 3 | 0.116 | 0.348 | 0.750 | 2.251 |
| 4 | 0.097 | 0.388 | 0.627 | 2.509 |
| 5 | 0.080 | 0.400 | 0.517 | 2.587 |
| 6 | 0.067 | 0.402 | 0.433 | 2.600 |
| 7 | 0.056 | 0.392 | 0.362 | 2.535 |
| 8 | 0.046 | 0.368 | 0.297 | 2.380 |
| 9 | 0.039 | 0.351 | 0.252 | 2.270 |
| 10 | 0.032 | 0.320 | 0.207 | 2.070 |
| 11 | 0.027 | 0.297 | 0.175 | 1.921 |
| 合　計 | 0.866 | 3.711 | 5.599 | 24.001 |

$$\alpha=\left(\sum_{i=1}^{L} i\cdot m_i\right)\Big/\left(\sum_{i=1}^{L} i\cdot g_i\right)=l/S=1/\beta$$

となってサービス分布 $g_i$ から，これに比例して発生させる細分割擬ジョブの個数 $m_i$ への変換係数が定まる．$g_i$ 曲線に比例する $m_i$ 曲線は横軸 $L$ の点で打ち切られていて $L+1$ 以後の $m_i$ は無視する（**図3**）．表2の例では

$S=3.71$，$\beta=3.71/24=0.155$

であるから，細分割擬ジョブの個数は**表3**の $m_i$ のようになる．ここで $m_i$ 個の細分割擬ジョブは $i \cdot m_i$ 回のCPUサービスを受けるから，$i \cdot m_i$ 値は1クワンタのみを要求する細分割擬ジョブの個数を示している．$S$ は $1/(1-\sigma)=24/4$ に近いことが望ましいから，この例では近似による誤差は非常に大きいことになる．このとき $M$ 個のジョブから発生する細分割擬ジョブの要求するクワンタを振り分ける方法は数多くあり，任意性がある．

われわれの場合は先行するジョブ（先に起動されたジョブ）は要求クワンタのより小さい値をもつ細分割擬ジョブから成っているとした（図3）．表2のジョブの待ち時間 $W_k{}^{\blacktriangledown}$ は，$W_8{}^{\blacktriangledown}, W_1{}^{\blacktriangledown}, W_4{}^{\blacktriangledown}, W_{11}{}^{\blacktriangledown}$ が先行順序とし，また1クワンタの細分割擬ジョブの待ち時間を $(W_i/i)$ で近似すると，

$$W_8{}^{\blacktriangledown}=\sum_{i=1}^{5}\gamma_i(W_i/i)$$
$$=\sum_{i=1}^{4}(m_i\cdot i)(W_i/i)+(0.362)(W_5/5)$$

となる．ここで $\gamma_5$ は $\sum_{i=1}^{4}(m_i\cdot i)+\gamma_5=8$（クワンタ），すなわち $\gamma_5=0.362$ として，表3の $i\cdot m_i=5\cdot m_5=2.587$ を分割して使用する．同様にして $W_1{}^{\blacktriangledown}=1.00(W_5/5)$ として近似する．係数1.00は $(2.587-0.362)=2.225$ を分割使用したもので，残り 1.225 は $W_4{}^{\blacktriangledown}$ の近似値の一部として使用する．

このようにして待ち時間 $W_8{}^{\blacktriangledown}, W_1{}^{\blacktriangledown}, W_4{}^{\blacktriangledown}, W_{11}{}^{\blacktriangledown}$ は

$$W_8{}^{\blacktriangledown}=1.08W_1+1.798\frac{W_2}{2}+2.251\frac{W_3}{3}$$
$$+2.509\frac{W_4}{4}+0.362\frac{W_5}{5}$$

$$W_1{}^{\blacktriangledown}=1.00\frac{W_5}{5}$$

$$W_4{}^{\blacktriangledown}=1.225\frac{W_5}{5}+2.600\frac{W_6}{6}+0.175\frac{W_7}{7}$$

$$W_{11}{}^{\blacktriangledown}=2.360\frac{W_7}{7}+2.380\frac{W_8}{8}+2.270\frac{W_9}{9}$$
$$+2.070\frac{W_{10}}{10}+1.921\frac{W_{11}}{11}$$

として求められる．この例からわかるように $k$ クワンタを要求する擬ジョブの CPU 系における滞在時間 $W_k{}^{\blacktriangledown}$ は

$$W_k{}^{\blacktriangledown}=\sum_i\gamma_i(W_i/i),\quad \sum_i\gamma_i=k \qquad (19)$$

として近似する．ただし，$i$ と $\gamma_i$ 値は上述の例と図3のように定める．

（6）　以上のようにして第1近似 $\lambda_j$ を使ってジョブ $J_j$ に対応する擬ジョブの CPU 滞在時間 $W_k{}^j$ の第1近似が得られた．この $W_k{}^j$ を使ってジョブ $J_j$ の滞在時間 $E_k{}^j$ は

$E_k{}^j=$(I/O 回数)$_j\times$($W_k{}^j$+1回の I/O 時間)

$$=b_j\cdot(W_k{}^j+c). \tag{20}$$

新しい $\lambda_j{}^{(2)}$ は，1回目の $E_k{}^j$ を $E_j{}^{(1)}$ と表すと

$$\lambda_j{}^{(2)}=(\text{I/O 回数})_j/E_j{}^{(1)}=b_j/\{b_j(W_k{}^j+c)\}$$
$$=1/(W_k{}^j+c) \tag{21}$$

として求める．$\lambda^{(i+1)}=\lambda_1{}^{(i+1)}+\cdots+\lambda_M{}^{(i+1)}$ として

$$|\lambda^{(i+1)}-\lambda^{(i)}|/\lambda^{(i)}<\varepsilon_\lambda,\ \varepsilon_\lambda:\text{ given} \tag{22}$$

となるまで計算を繰り返す（$\lambda$-iteration）．上の式をみたす $\lambda^{(i)}$ を $\lambda$ とするとCPU使用率 $\rho_M$ は $\rho_M=\lambda E$，ここで $M$ は系内でのジョブ滞在数．

（7） ジョブ多重度 $M$ を $M=1,2,\cdots$ と動かして $\rho_M$ が最大となる $M$，あるいは

$$|\rho_{M+1}-\rho_M|/\rho_M<\varepsilon_M,\ \varepsilon_M:\text{ given} \tag{23}$$

となる $M$ を求めるべき多重度とする．

（8） 多重度 $M$ によってジョブ処理がおこなわれたとする．最初に終了するジョブ $J_j$ の経過時間を $E_j$ とすると，ポアソン入力の仮定から，この間にジョブ $J_j$ を含め各ジョブ $J_k$ が発信した入出力回数 $b_k{}^\nu$ は

$$b_k{}^\nu=\sum_{i=0}^{\infty} i\frac{(\lambda_kE_j)^i}{i!}e^{-\lambda_kE_j}$$
$$=\lambda_kE_j\sum_{i=1}^{\infty}\frac{(\lambda_kE_j)^{i-1}}{(i-1)!}e^{-\lambda_kE_j}$$
$$=\lambda_kE_j. \tag{24}$$

$b_j{}^\nu$ はジョブ $J_j$ の現実の入出力回数 $b_j$ と一致することが望ましいが実際はジョブ・パターンによって差がでてくる．

（9） $b_k-b_k{}^\nu>0$ を新しい $b_k$ とする．$b_k-b_k{}^\nu\leq 0$ のときは計算を打ち切る．$a_k-a_k{}^\nu$（ここで $a_k{}^\nu=b_k{}^\nu\cdot Q_k$ である）を新しい $a_k$ とし，$T_k=a_k+b_k\cdot c$ によって新しい $T_k$ を求め，$b_k/T_k$ を新しい $\lambda_k$ とする．前回と同様の操作で新しい $M$ を求める．

（10） 上記操作を1システムに割り当てられた $N$ ジョブについておこない該当システムの終了時間を求める．他システムの終了時間と比較し，最短時間で終了するシステムへ他システムの未処理ジョブをまわす．

以上の操作を流れ図で表現したものが**図4**である．

上記（6）では $\lambda_i\leftarrow(\lambda_i{}^{NEW}-\lambda_i{}^{OLD})\cdot\theta+\lambda_i{}^{OLD}$ として収束計算をおこなう．$\theta=0.5$ を使用して5～20回で収束している．

筆者らのデータではLPTルールによる第1回の負荷配分によって各システムで終了時間予測計算をおこなったところ，それら時間の差はたかだか数百秒であったので，未処理ジョブの再配分は必要なかった．スケジューラはFORTRANで約700ステートメントである．

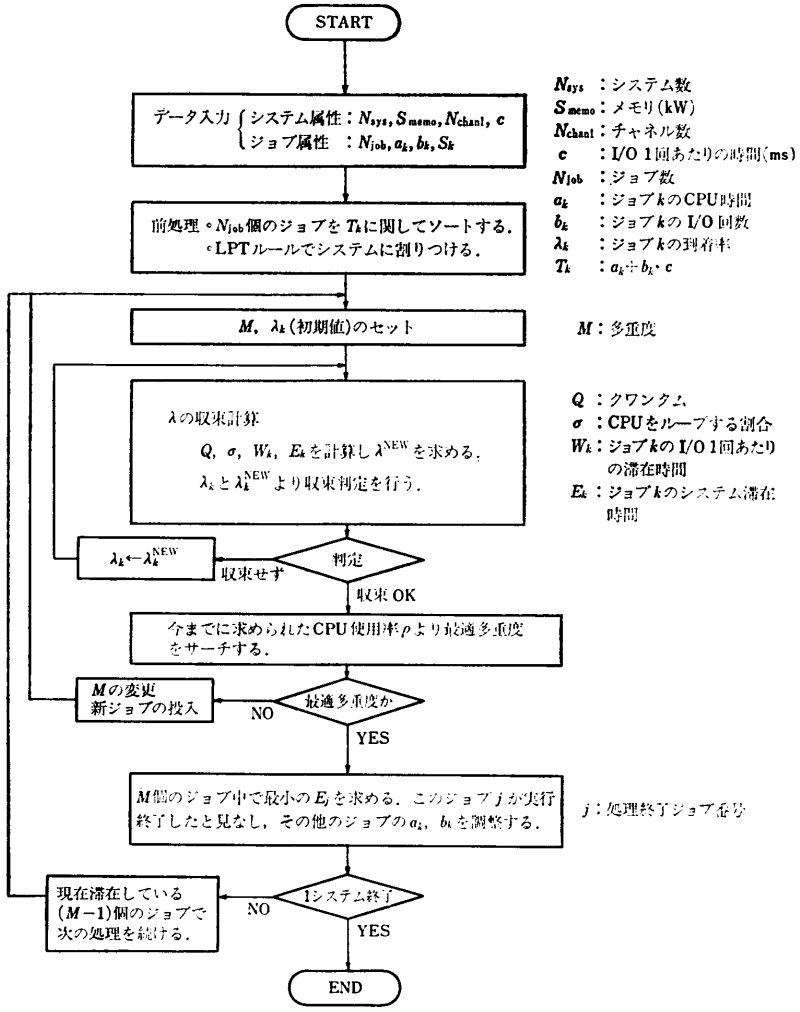



**図4** スケジューラ概略図

Fig. 4 General flow of scheduler.

表 4 処理対象ジョブ群
Table 4 A set of jobs processed.

| ジョブ個数 | CPU 時間 | I/O 時間 | I/O 回数 |
|---|---|---|---|
| 37 | 16,400秒 | 5,600秒 | 186,474 |

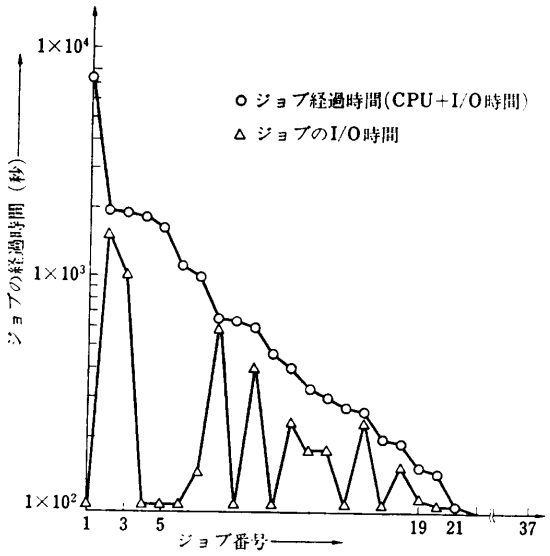



図 5 計算例のジョブ・パターン
Fig. 5 Job pattern of computed example.

## 4. 予測計算結果の検討

### 4.1 スケジューラへの入力

スケジューラへの入力は原研計算センタの会計情報ファイルから一夜の夜間ジョブを抽出して作成した．使用計算機は FACOM M-200 である．

113 個のジョブを LPT ルールで 3 システムに分配し，そのうちのひとつのジョブ系列（**表 4**）をスケジューラへの入力とした．そのジョブ・パターンを**図 5** に示す．

表 4，図 5 において I/O 時間とあるのは各ジョブの I/O 回数に 30 ミリ秒の時間を乗じて得られたものである．1 回の I/O 時間を 30 ミリ秒と仮定する限りは，このジョブ系列は CPU 寄り（CPU バウンド）である．図 5 において 100 秒以下の I/O 時間は 100 秒で図示している．

### 4.2 計算にあたっての制限条件

多重度を抑制する要素は CPU 使用率，主記憶容量，入出力チャネル数である．抑制の方法は次のようにした．

（1） CPU 使用率：多重度の計算において CPU 使用率が 1 を超える場合は，1 を超えて収束したときの多重度を求めるべき多重度とした．CPU は 1 台である．

表 5 経 過 時 間
Table 5 Elapsed time.

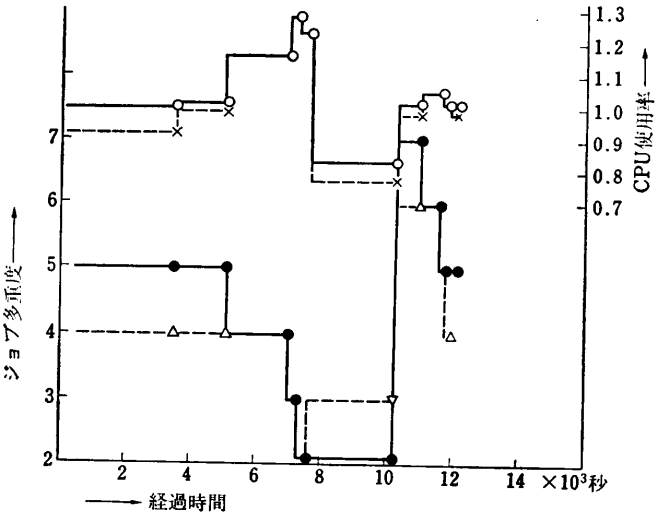
| 推 定 値 | スケジューラ | シミュレータ |
|---|---|---|
| 17,200秒 | 17,000秒 | 17,000秒 |

（2） 主記憶容量：10MB（メガバイト）までとし，多重度はこの容量を超えない範囲とした．

（3） チャネル台数：多重度はこの台数を超えない範囲とした．この章で述べる計算では 7 台としている．これらチャネルは，この台数までは任意に多重化でき，その 1 回の I/O 時間は 30 ミリ秒としている．したがって表 4 のジョブ処理の経過時間はおよそ 16,400+5,600/7=17,200 秒になるはずである．

### 4.3 計算結果の検討

（1） 表 4 のジョブ系列をスケジューラでスケジュールし，その経過時間を求めた．またスケジュールした結果を出力しそれをラウンド・ロビンのジョブ・スケジューリングを模擬するシミュレータ（FORTRAN で 600 ステートメント）への入力とした．その結果を




○ 選ばれたジョブ多重度に対応する CPU の使用率
× 次善のジョブ多重度に対応する CPU の使用率（記載してないときは○に同じ）
● 選ばれたジョブ多重度
△ 次善のジョブ多重度
▽ 次善のジョブ多重度（記載してないときは●に同じ）（多重度をあげすぎたとき）

図 6 最適多重度と CPU 使用率
Fig. 6 Optimal degree of multiprogramming and CPU utilization.

表 6 到着率 $\lambda_i$ の比較
Table 6 Comparison of arrival rate $\lambda_i$.

| 時 刻 | ジョブ番号 | $\lambda_i$(スケジューラ) | $\lambda_i$(シミュレータ) |
|---|---|---|---|
| 1 | 1, 2, 3, 4, 5 | 0.069, 15., 6.8, 0.38, 0.11 | 0.057, 15., 7.6, 0.37, 0.12 |
| 2 | 1, 3, 4, 5, 6 | 0.067, 7.8, 0.38, 0.12, 0.041 | 0.053, 8.1, 0.34, 0.11, 0.046 |
| 3 | 1, 4, 5, 6 | 0.094, 0.51, 0.16, 0.055 | 0.065, 0.40, 0.14, 0.057 |
| 4 | 1, 7 | 0.13, 1.8 | 0.074, 1.7 |
| 5 | 1,8, 9, 10, 11, 12, 13 | 0.056, 24., 0.36, 9.0, 0.022, 4.9, 3.5 | 0.046, 23., 0.31, 7.9, 0.021, 6.6, 5.2 |
| 6 | 1, 9, 10, 11, 12, 13 | 0.065, 0.41, 10., 0.024, 6.2, 3.1 | 0.045, 0.30, 8.8, 0.022, 5.5, 5.5 |
| 7 | 1, 9, 14, 15, 16 | 0.070, 0.43, 0.026, 6.6, 0.10 | 0.053, 0.35, 0.024, 8.4, 0.13 |

**表5**に示す.

(2) スケジューラによる予測計算を開始してから順次 10 個のジョブについて，その処理終了となった時刻ごとに，それまでの最適多重度，次善の多重度，それぞれに対応する CPU 使用率を図で表したものが**図6**である．これはスケジューラによる計算結果である.

(3) 上記のジョブについて同時刻にスケジューラ，シミュレータで動いている同一のジョブの CPU 系への到着率 $\lambda_i$ を，スケジューラによってジョブ処理終了とされた時刻ごとに比較すると**表6**のようになる．表4のジョブ系列のスケジューラによる処理状況を詳細に追跡した結果，図6の多重度5，4，3のとき $\rho>1$ となっているのは次のような理由によることがわかった.

(a) 実際の CPU 残りサービス時間は $Q/2$ ではなくQに近い.

(b) 実際の平均待ち行列長 $\pi$ は理論値よりも大きい.

(c) $Q$ の値が 3.9 秒と大きい．すなわち $Q \gg c$ である.

(d) 上記(a)，(b)を理論値で計算しているために待ち時間 $W_k$ の値が小さくなり，したがって $\lambda_k$ の値は大きくなって $\rho>1$ となる.

### 4.4 スケジューラの長短

[長所]

(1) 短時間で終わるスケジューリング

短時間でジョブのスケジューリングが可能である．表4に挙げた夜間ジョブのスケジューリングは1システムについて，FACOM M-200 計算機の CPU 時間で3.6秒であった．このことから現状の夜間ジョブを3システムに振り分けてスケジュールし，その結果生じる多少の処理時間の凹凸を調整したとしても15秒あればよい.

(2) 最適 CPU 使用率の推定

ジョブが入力されるたびに，そのジョブの特性まで含めたジョブ処理環境における最適な CPU 使用率を推定することができる.

(3) 計算機性能の内外挿

スケジューラに入力するジョブの特性，すなわち CPU 時間，I/O 回数などを変更することによって各種ジョブ・パターンに対する各種計算機の性能を予測することができる.

(4) OS オーバヘッドの考慮

クワンタム切換え，1回の I/O に要する OS の平均時間がわかっていれば予測計算に OS のオーバヘッドを考慮することができる.

[短所]

(1) 実際処理時間との誤差

入力されるデータは，到着率がポアソン分布，サービス時間が幾何分布，擬ジョブ1回の処理時間は正確にタイム・クワンタム量 $Q$ と等しいと仮定されている．ジョブの組合せによっては，この仮定は必ずしも正しくないので誤差を生じる．また近似計算式によっても誤差を生じる.

## 5. おわりに

入力されるジョブの使用 CPU 時間，I/O 回数が推定できるという前提で，バッチ処理における最適多重度決定の一手法を提案した.

この手法は，計算法が単純である，直観的でわかりやすい，計算時間はスケジューリングの対象となるジョブ数に依存し個々のジョブの特性にさほど依存しない，I/O およびタスク切換えの OS のオーバヘッド量を推定できるよう，また OS の複雑なスケジューリング・アルゴリズムを模擬するよう拡張することができる，などの特長がある．CPU バウンドのジョブが多くなると誤差が大きくなる，I/O 時間設定が単純すぎる，などの難点の解消が今後の課題である.

本論の方法は，投入される全ジョブ，個々のジョブ

の CPU 時間，I/O 回数があらかじめわかっている夜間ジョブのスケジューリングに有効である．しかし，最初にスケジューリングの対象になっていないジョブが次々と入ってくる昼間のジョブ処理に適用することはかなりむずかしい．

**謝辞** 富士通 OS 部の田淵治樹，日立システム開発研究所の渡辺坦，大町一彦の諸氏は，本報告の草稿を一読し種々有益なコメントを与えられた．深く感謝します．また査読者から有益なコメントをいただき本論文を読みやすくすることができた．感謝します．

## 参 考 文 献

1) Brandwajn, A.: A Queuing Model of Multiprogrammed Computer Systems under Full Load Conditions, *J. ACM*, Vol. 24, No. 2, pp. 222-240 (1977).
2) Price, T. G.: A Note on the Effect of the Central Processor Service Time Distribution on Processor Utilization in Multiprogrammed Computer Systems, *J. ACM*, Vol. 23, No. 2, pp. 342-346 (1976).
3) Mitrani, I.: Nonpriority Multiprogramming Systems under Heavy Demand Conditions-Customer's Viewpoint, *J. ACM*, Vol. 19, No. 3, pp. 445-452 (1972).
4) Wallace, V. L.: Degree of Multiprogramming in Page-on-Demand Systems, *C. ACM*, Vol. 12, No. 6, pp. 305-308 (1969).
5) Towsley, D. et al.: Models for Parallel Processing within Programs: Application to CPU: I/O and I/O: I/O Overlap, *C. ACM*, Vol. 21, No. 10, pp. 821-830 (1978).
6) Ishiguro, M.: Running Time Delays in Processor-Sharing Systems, *JIP*, Vol. 3, No. 1, pp. 38-44 (1980).
7) 野中，大野：専用オンラインシステムにおけるマルチタスクシステム性能解析，情報処理学会論文誌，Vol. 21, No. 4, pp. 297-305 (1980).
8) 原田，箱崎：計算機システムモデルの実用性の検討，電子通信学会論文誌，Vol. J61-D, No. 2, pp. 127-134 (1978. 2).
9) Brandwajn, A.: A Model of a Virtual Memory System, *Acta Informatica*, Vol. 6, pp. 365-386 (1976).
10) Denning, P. et al.: Optimal Multiprogramming, *Acta Informatica*, Vol. 7, pp. 197-216 (1976).
11) 浅井，高橋，藤井：計算機ジョブ処理最適多重度の決定，JAERI-M 9501，日本原子力研究所 (1981. 5).
12) Kleinrock, L.: Analysis of a Time-Shared Processor, *Nav. Res. Log. Quart.*, Vol. 11, No. 10, pp. 59-73 (1964).
13) Coffman, E. G. Jr. and Denning, P. J.: *Operating Systems Theory*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1973).
14) 西田俊夫：待ち行列の理論と応用，朝倉書店 (1971. 11).
15) Allen, A. O.: *Probability, Statistics, and Queueing Theory*, Academic Press, New York (1978).

(昭和 56 年 6 月 11 日受付)
(昭和 57 年 4 月 19 日採録)