処理状態を考慮した並列処理システムのための最適スケジューリング・アルゴリズム†

宮村 勲†† 榎本 肇†††

複数の処理装置からなる計算機システムにおいて，個々の処理装置に対して処理状態を定義する．このモデルにおいて，ジョブの平均待ち時間の短縮だけでなく，処理装置の状態切換え回数を抑える比較的簡単なスケジューリング・アルゴリズムを提案し，理論とシミュレーションの両方でその有効性を示している．待つことによって生じる損失と状態切換えによって生じる損失を考え，両方の和の期待値を最小にする．しかし，単なる時間平均でなく，未来の損失ほど影響が小さくなるように指数関数の重みを付ける．タスクを割り当てるには各処理装置の状態を決定する必要があるので，未来への影響も考えて先読みする．しかし，確率の導入により，動的計画法のように複数のパスをすべて調べるのでなく，平均的な到着を仮定して，状態を切り換えた場合に減少する損失と，増加する損失を計算し，その差が一定値を越えると状態を切り換える．

このような簡単化により，アルゴリズムの実行に必要な時間は，最も単純な先着順の場合に比べても，2～3倍程度である．同じカテゴリに属するタスクはまとめて処理するので，平均待ち時間は少し大きくなるが，状態切換え回数は小さくなる．このアルゴリズムを用いた時の状態切換え間隔の性質や，到着率の小さいタスクについての性質を理論的に調べ，望ましい性質を持つことを示し，さらに，シミュレーションによって処理装置の台数についての結果も示した．

1. はじめに

一般にM台の処理システムで構成された計算機システムのスケジューリングについて述べる．個々の処理システムに処理状態というものを考え，処理するタスクに応じて処理状態を切り換えるものとする．このようなシステムにタスクが順次到着する場合に，各タスクの平均待ち時間の短縮と，処理システムの状態切換え回数を抑えるスケジューリング・アルゴリズムを与える．

従来からスケジューリングについて幾多の研究がなされている．処理順序に制約が課せられ，個々の処理時間も分かっているタスクの有限集合が与えられた時に，この有限集合を最短時間で処理する問題は，Ramamoorthy[1] や Hu[2] らが解いている．これに対し，与えられたタスクの平均待ち時間を最小にする問題は，単一処理システムの場合にしか解かれていない[3)~5)]．処理システムが複数になると，効果的なアルゴリズムの存在しない NP-complete な問題になることが，Ullman[6] によって示された．

ある統計的性質を持ってタスクが順次到着する場合には，先着順やラウンド・ロビン方式等の簡単なスケジューラを用いた時の待ち時間の分布しか分からない．

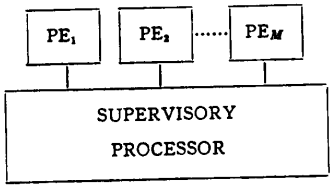
一方，TSS や仮想記憶システムにおいて，ロールイン・ロールアウトによるオーバヘッドが大きいことはよく知られており，一般にプロセスの切換えやページ記憶の内容交換時においても発生する．これらは CPU 時間と計算時間の差として現われる．処理の内容が多岐になる程この問題は重要であり，並列処理を行う時，どのようにスケジュールするかを研究する必要がある．ここでは処理が同じカテゴリに属するタスクを継続実行するならば損失はなく，異なるカテゴリに属するタスクを実行するためには処理システムの状態を変更する必要があり，その際に損失が発生すると考える．本論文は統計的定常性を利用して，状態切換えによるオーバヘッドを小さく抑えつつ，各タスクの平均待ち時間も小さくなるように，次々と到着するタスクの実行を管理するスケジューラについて論じる．

2. モデルの設定

対象とするシステムは**図1**で示すように，同じ処理能力を持つ複数の処理システムで構成され，個々の処理システムは簡単のため，多重処理しないものとする．もし多重処理する場合，その多重度をnとすれ

† An Optimal Scheduling Algorithm for the Parallel-Processing Systems Considering Loss Caused by the Change of Processing States by ISAO MIYAMURA (Department of Information Engineering, Faculty of Engineering, Niigata University) and HAJIME ENOMOTO (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 新潟大学工学部情報工学科
††† 東京工業大学工学部情報工学科



PE: PROCESSING ELEMENT

図 1 システム構成

Fig. 1 The system configuration.

ば，仮想的に n 個の処理システムがあると考えればよい．

ユーザのジョブは，タスクを節点として持つ有向グラフで表わされ，枝は実行順序に関する制約を示す．あるタスクが実行できるのは，それより先に処理しなければならないタスクがすべて終了した後に限る．

ユーザにとり，個々のタスクの終了でなく，ジョブの終了が問題となる．そこで，タスクを処理装置に割り当てるには，処理装置の状態やタスク自身の特徴だけでなく，タスクが属すジョブの特徴も考慮する必要がある．しかし，実行可能なタスクすべてに対してこれらを比較していたのではスケジュールに時間がかかり，何のためにスケジュールするかの意味がなくなる．これを避けるため，スケジューラの機能を2つに分割する．1つは，各タスクに対し，そのタスクの属すジョブの特性やタスク自身の特性から優先度を決定する．他方は与えられた優先度を基に，各処理システムの状態を考慮してタスクを割り当てる．

一方，従来の平均待ち時間最小問題は，処理すべきタスクの数が有限なので，待ち時間の総和を最小にする問題と等価である．ただし，単なる和でなく，重み付き和を最小にするのが普通である．

$$\sum_i k_i \cdot e_i \longrightarrow \mathrm{Min} \qquad (2.1)$$

ここで，k_i, e_i はそれぞれ i 番目のタスクの重み係数と，処理完了までの待ち時間を表わす．$k_i \cdot e_i$ を待ちによる損失と考えれば，重み係数 k_i は単位時間の待ちによって生じる損失量を表わすので，重み係数を損失係数と呼ぶ．(2.1)式の評価関数における最適スケジュールは，単一処理システムの場合，損失係数 k と処理時間 T との比 k/T の値の大きい順に処理することである[7]．すなわち，k/T がタスクの優先度を表わす．

タスクに対して優先度を決定する方法については，従来からさまざまな方法が研究されており，計算機システムの置かれる環境やタスクの種別によって大いに影響を受ける．そこで，本論文では優先度を決定する部分については議論せず，実行可能なタスクに対して，損失係数が与えられているものとして議論する．

スケジューラはタスクの平均待ち時間を短縮するものであるが，処理システムの状態切換えに要する時間を状態切換えによる損失と考え，待ちによる損失と状態切換えによる損失との和を最小にする．そこで，1台の処理システムにおける1回の状態切換えに際して一定時間に対応した損失を考え，これを Ls で表わす．

割り当てるタスクをスケジューラが決定する際に用いる評価関数を決める．時刻 t において待たされているすべてのタスクの損失係数の和を $f(t)$ とし，状態切換えの起きる時刻を $t_1, t_2, \cdots, t_i, \cdots (t_i>0)$ とする時，時刻0における評価関数 $\boldsymbol{P}$ を次のように定義する．

$$\boldsymbol{P}=\int_0^\infty [f(t)+Ls\cdot\sum_i \delta(t-t_i)]e^{-\alpha t}\,dt \qquad (2.2)$$

これは割引付きの決定問題で，α は割引係数と呼ばれ，$\alpha>0$ である．このような割引を付けたのは，未来で生じる損失の影響を小さくするのが主目的である．また，被積分関数の値がある程度先で0に近づくので，先読みを途中で打ち切っても誤差が小さくてすむ．あるいは，システムが過負荷になり，システム内の未処理タスクの数が増加すると $f(t)$ も大きくなるが，積分値は有限に確定する等の利点があるからである．

処理装置のとり得る状態は，$S_1, S_2, \cdots, S_N$ の N 種類とする．各処理システムはタスクを処理するために，そのタスクが要求する状態になる必要がある．タスクの集合は，それが要求する処理システムの状態によって，N 個のカテゴリに分類でき，状態 S_i において処理できるカテゴリをカテゴリ i と呼ぶ．

3. 最適スケジューリング・アルゴリズム

システムに到着したジョブは複数のタスクに分割される．実行可能になったタスクは，それが属すジョブの特徴その他から損失係数が決められ，カテゴリごとに優先度順の待ち行列を作る．処理システムは，自分の処理していたタスクが終了すると，自分の状態に対応したカテゴリの待ち行列の先頭のタスクを処理すれば良い．そこで，スケジューラは各時点での評価関数の値が最小となるように，各処理システムの状態を決定すれば良い．しかし，ある決定を下す時の評価関数の値はその時点では決まらず，無限の未来までの情報が必要となる．スケジューラは実際の評価関数の値でなく，その期待値が最小となるように処理システムの

状態切換えを決定する．

まず，M台の処理システムの中で，1台だけが状態を切り換える場合を考える．現在を時刻0とし，時刻 t ($t\geqq 0$) において状態 S_i の処理システムを1台，状態 S_j に切り換えると決定した時の評価関数の期待値$\boldsymbol{P}(t)$を，現在までに得られている情報より計算する．時刻 τ ($\tau\geqq 0$) におけるカテゴリ l ($1\leqq l\leqq N$) に属すタスクの損失係数の和の期待値を $f_l(\tau,t)$ とすれば，$\boldsymbol{P}(t)$ は次式のようになる．

$$\boldsymbol{P}(t)=\int_0^\infty \{f_i(\tau,t)+f_j(\tau,t)\}e^{-\alpha\tau}d\tau + L_i e^{-\alpha t}+B \tag{3.1}$$

ここで，Bはカテゴリ i, j 以外のタスクによる損失を表わす．この式を最小にする t が最適な状態切換え時刻である．これを求めるため，(3.1)式を t で微分する．$0<\tau<t$ の範囲では，$f_i(\tau,t)$, $f_j(\tau,t)$ は切換えの影響を受けないので，

$$\frac{\partial f_i(\tau,t)}{\partial t}=\frac{\partial f_j(\tau,t)}{\partial t}=0 \quad \text{for} \quad 0<\tau<t \tag{3.2}$$

が成り立つ．この関係を用いると，$\boldsymbol{P}(t)$ の導関数は次のようになる．

$$\frac{d\boldsymbol{P}(t)}{dt}=\left[\int_t^\infty\left\{\frac{\partial f_i(\tau,t)}{\partial t}+\frac{\partial f_j(\tau,t)}{\partial t}\right\}e^{-\alpha(\tau-t)}d\tau - \alpha\cdot Ls\right]e^{-\alpha t} \tag{3.3}$$

ここで，$\beta_i^{(-)}(t)$, $\beta_j^{(+)}(t)$ を次のように置く．

$$\left.\begin{aligned}\beta_i^{(-)}(t)&\equiv-\int_t^\infty\frac{\partial f_i(\tau,t)}{\partial t}e^{-\alpha(\tau-t)}d\tau\\ \beta_j^{(+)}(t)&\equiv\int_t^\infty\frac{\partial f_j(\tau,t)}{\partial t}e^{-\alpha(\tau-t)}d\tau\end{aligned}\right\} \tag{3.4}$$

最適な状態切換え時刻 t は，(3.3)式を0にする値なので，次式の解として与えられる．

$$\beta_j^{(+)}(t)-\alpha\cdot Ls=\beta_i^{(-)}(t) \tag{3.5}$$

$\beta_j^{(+)}(t)$ は，状態切換えを単位時間遅らせた時に，カテゴリ j に属するタスクの待ちによる損失の和の増加割合を示す．逆に $\beta_i^{(-)}(t)$ は，状態切換えを単位時間遅らせた時に，カテゴリ i に属するタスクの待ちによる損失の減少割合を示す．

$\beta_i^{(-)}(t)$ や $\beta_j^{(+)}(t)$ は現在までに得られている情報から，その期待値を推定する．そのためには，定義より $f_i(\tau,t)$ と $f_j(\tau,t)$ の導関数が分からなくてはならない．しかし，$f_i(\tau,t)$ と $f_j(\tau,t)$ の導関数を直接求めることは不可能なので，実際に計算できる関数によってどのように表現されるかを求める．

まずカテゴリ j の場合を計算する．$f_j(\tau,t)$ は時刻 τ において待ち行列内にあるカテゴリ j のタスクの損失係数の和である．$\partial f_j(\tau,t)/\partial t$ は状態切換えを単位時間遅らせた時の損失係数の和の変化を表わしている．時刻τにおいて待ち行列に並ぶカテゴリ j のタスクで，優先度がpと $p+dp$ の間にあるものの処理時間の和を $h_j(\tau,t,p)dp$ とおく．優先度と処理時間を掛けると損失係数になるので，$f_j(\tau,t)$ は次式で表わされる．

$$f_j(\tau,t)=\int_0^\infty h_j(\tau,t,p)p\,dp \tag{3.6}$$

$$\frac{\partial f_j(\tau,t)}{\partial t}=\int_0^\infty\frac{\partial h_j(\tau,t,p)}{\partial t}p\,dp \tag{3.7}$$

$\partial h_j(\tau,t,p)/\partial t$ は状態切換えを単位時間遅らせた時の，残っているタスクの変化を表わしている．状態切換えを単位時間遅らせると，カテゴリ j のタスクを処理している延べ時間は単位時間減少する．タスクの到着などが状態切換えの影響を受けないものとすれば，残っているタスクの処理時間の和は単位時間増加する．

$$\int_0^\infty\frac{\partial h_j(\tau,t,p)}{\partial t}dp=1 \tag{3.8}$$

同じカテゴリに属すタスクは優先度順に処理されるので，処理している時間が減った時に直接影響を受けるのは一番優先度が高いものである．そこで，時刻 t で状態を切り換えた時に，時刻τで残っているカテゴリ j のタスクの中で優先度が最大であるものに着目し，この最大値の期待値を $c_j(\tau,t)$ と置く．状態切換えが遅れると，$c_j(\tau,t)$ の値が大きくなる．ゆえに，$\partial h_j(\tau,t,p)/\partial t$ は次のようになる．

$$\frac{\partial h_j(\tau,t,p)}{\partial t}=\delta(p-c_j(\tau,t)) \tag{3.9}$$

この関係を(3.7)式に代入すると，$\partial f_j(\tau,t)/\partial t$ は次式で計算できる．

$$\frac{\partial f_j(\tau,t)}{\partial t}=c_j(\tau,t) \tag{3.10}$$

この関係を $\beta_j^{(+)}(t)$ の定義式(3.4)に代入すると，$\beta_j^{(+)}(t)$ は次式で求まる．

$$\beta_j^{(+)}(t)=\int_t^\infty c_j(\tau,t)e^{-\alpha(\tau-t)}d\tau \tag{3.11}$$

$\beta_i^{(-)}(t)$ の場合，カテゴリ i は状態切換えを単位時間遅らせると，カテゴリ i を処理している時間の総和が単位時間増加するので，待っているタスクの処理時間の和は単位時間減少する．

$$\int_0^\infty\frac{\partial h_i(\tau,t,p)}{\partial t}dp=-1 \tag{3.12}$$

カテゴリ i も優先度順に処理されるので，処理してい

る時間が増加すると，最大優先度を持ったタスクが待ち行列からなくなるだけなので，$\partial h_i(\tau, t, p)/\partial t$ は次のようになる．

$$\frac{\partial h_i(\tau, t, p)}{\partial t} = -\delta(p - c_i(\tau, t)) \qquad (3.13)$$

この関係を $\beta_i^{(-)}(t)$ の定義式(3.4)に代入すると，$\beta_i^{(-)}(t)$ は次式で計算できる．

$$\beta_i^{(-)}(t) = \int_t^{\infty} c_i(\tau, t) e^{-\alpha(\tau - t)}\, d\tau \qquad (3.14)$$

任意の時刻で状態の切換えが可能である場合，理論的には(3.5)式を満たす時刻 t において，状態 S_i の処理システムを状態 S_j に切り換えた方が良い．しかし，あらゆる時刻 t に対して $\beta_i^{(-)}(t)$, $\beta_j^{(+)}(t)$ を計算するのは得策でなく，また，実行中のタスクの処理を途中で中断するのも望ましくない．したがって，以後では切換え可能時点を設定し，この時点においてのみ状態は切換え可能であるものとする．切換え可能時点は等間隔であっても，そうでなくても良い．これはある期間，タスクの処理を中断しないことを意味する．TSS の場合，タイマ等で規定される区切りでしか状態を切り換えないことに対応する．

状態切換え可能な時刻 t に対して，

$$\beta_j^{(+)}(t) - \alpha \cdot L_s - \beta_i^{(-)}(t) \geqq 0 \qquad (3.15)$$

を満たすと，$d\boldsymbol{P}(t)/dt \geqq 0$ となるので，時刻 t において状態 S_i の処理システムを1個，状態 S_j に切り換えた方が評価関数の値は小さくなる．$\beta_j^{(+)}(t)$ は t の増加関数であり，$\beta_i^{(-)}(t)$ は減少関数であることが分かっているので[10]，ある t で(3.15)式が成り立つと，その後も成り立つ．そこで，(3.15)式を成り立たせ，かつ中断可能な処理システムが存在する最初の時点で切り換える．また，未来の時点での状態切換えについて決定を下すのは大変なので，常に現時点での状態切換えについて決する．すなわち，各処理システムがタスクを処理してゆき，中断可能な時点になると，スケジューラはその処理システムをその時点でほかの状態に切り換えた方が良いかを判断する．もし切り換えた方が良ければ，その処理システムを指定された状態に切り換える．もし，切り換えないほうが良い場合はそのままの状態で，再び中断可能な時点に達するまで処理を続ける．

以上の議論は一度に1個の処理システムが状態を切り換える場合であった．タスクの処理時間が連続的に分布しているので，2台以上が同時に中断可能になることは稀である．また，無関係な2組の状態間での切換えは互いに影響を及ぼさない．1つの状態に関連して2個以上の処理システムが同時に切り換わる場合でも，1個目を切換えた後でもまだ(3.15)式が成り立つので，直ちに2個目を切り換えれば良い．

(2.2)式を最小化する方法として，動的計画法を直接用いると複雑になり，必要な計算量も多くなる．これに対して，本方法は各処理システムの状態切換えを現在までの情報から決定するものである．そのため，統計的パラメータその他の急激な変化に対して対応が遅れる可能性もある．

4. スケジューラの性質

前章では損失係数の与えられたタスクが次々と到着する場合に，いつ処理システムの状態を切り換えたら良いかを決定するアルゴリズムを与えた．

本アルゴリズムでは損失係数の決め方については問題にせず，各タスクの現在の損失係数がその後も不変であるとして先読みする．損失係数あるいは優先度の決定については幾多の提案がなされているが，まだ決定的方法は見つかっていない．一度優先度を決めた後でその値を固定するもの，随時更新するもの等が考えられ，後者の方が一般的である．しかし，本論文で先読みを行う時に，現在の損失係数をそのまま用いているのは次の理由による．1) 損失係数の決め方に決定的なものがないので，決め方が変わってもアルゴリズムに影響を及ぼさないようにする．2) 本アルゴリズムの中で損失係数の変化まで予測させると，損失係数の決定と本アルゴリズムを分離した意味がなくなり，アルゴリズムが複雑になりすぎる．たとえ損失係数が変化する場合でも，急激に変化することはなく，ゆっくり変化する．また，変動の影響を小さくするため，指数関数の重みを付けている．

本章では，前章で求めたアルゴリズムを用いた時に，システムの動作として状態切換え間隔や到着率の小さなカテゴリの影響などを解析する．簡単のため，すべてのタスクの処理時間は等しく T とする．処理システムの個数 $M=1$ とし，カテゴリ数 $N=2$ とする．タスクの到着は平均 λ のポアソン分布に従うものとし，システムの利用率 $\rho \equiv \lambda \cdot T$ と置く．利用率 ρ をカテゴリ1と2に分け，それぞれ，ρ_1, ρ_2 と置く．($\rho = \rho_1 + \rho_2$)

適当な損失係数の決め方を仮定しないと解析できないので，タスクが実行可能になってからの経過時間を

損失係数として採用する．ジョブその他の特徴については考えない．

4.1 状態切換え間隔

本スケジューラを用いた時に，処理システムの状態切換えがどうなるかを調べる．処理システムは状態 S_1 と S_2 を交互にとるが，連続して1つの状態をとる間隔はタスクの到着によってばらつく．そこで，間隔の平均値を調べる．連続して状態 S_i である間隔の平均値を T_i とし，この間をセグメント i と呼ぶ．

優先度が待ち時間に比例するので，同じカテゴリに属すタスクは到着順に処理される．そこで，セグメント1の始めに，カテゴリ1,2それぞれにおいて X_1, X_2 時間前以降に到着したタスクがすべて未処理で残っているものとする。同様に，セグメント2の始めにおいて，それぞれ Y_1, Y_2 時間前以降に到着したタスクが未処理とする．状態切換えを決定する場合，常にその時点が切換え時点か判断するので，$\beta_i^{(-)}(t), \beta_j^{(+)}(t)$ を計算する場合，$t=0$ に固定されているので，t は書かない．簡単のため，任意の時刻で切換え可能とする．

セグメント1の始めで，状態 S_2 から状態 S_1 へ切り換わるので，この時点で(3.5)式が成り立つ．そこで，この時点を時刻0とし，この時の $\beta_1^{(+)}$ と $\beta_2^{(-)}$ を計算する．これ以後はセグメント1になり，時刻 τ において未処理であるカテゴリ1のタスクは，時刻0よりも $X_1-\tau/\rho_1$ 時間前以降に到着したタスクである．したがって，その最大優先度 $C_1(\tau)$ は次のようになる．

$$C_1(\tau)=\max\left\{\frac{X_1-\tau/\rho_1}{T},0\right\} \tag{4.1}$$

これを(3.11)式に代入すると，$\beta_1^{(+)}$ は次式となる．

$$\beta_1^{(+)}=\frac{1}{\alpha T}\left\{X_1-\frac{1}{\alpha\rho_1}(1-e^{-\alpha\rho_1 X_1})\right\} \tag{4.2}$$

時刻0以後のカテゴリ2の最大優先度は X_2/T である．これを(3.14)式に代入すると，次式が得られる．

$$\beta_2^{(-)}=\int_0^\infty \frac{X_2}{T}e^{-\alpha\tau}d\tau=\frac{X_2}{\alpha T} \tag{4.3}$$

この2式を(3.5)式に代入すると，次の関数が得られる．

$$X_1-\frac{1}{\alpha\rho_1}(1-e^{-\alpha\rho_1 X_1})=X_2+\alpha^2\cdot T\cdot L_S \tag{4.4}$$

セグメント2の始まりでは，状態 S_1 から S_2 へ切り換わるので，(4.4)式と同様に Y_1 と Y_2 に次の関係が成り立つ．

$$Y_2-\frac{1}{\alpha\rho_2}(1-e^{-\alpha\rho_2 Y_2})=Y_1+\alpha^2\cdot T\cdot L_S \tag{4.5}$$

次にセグメント1,2の間に到着するタスクや処理されるタスクについて考える．セグメント1で到着するカテゴリ1のタスクの処理時間の和は $\rho_1 T_1$ である．また，処理システムが休みなく処理した場合，セグメント1の終わりに残っているカテゴリ1のタスクの処理時間の和は $\rho_1(X_1+T_1)-T_1$ である．カテゴリ2の場合は到着だけで処理されないので，$\rho_2(X_2+T_1)$ だけ残る．そこで，Y_1, Y_2 は次のように表わされる．

$$\left.\begin{aligned} Y_1&=\max\left\{X_1-\frac{1-\rho_1}{\rho_1}T_1,0\right\}\\ Y_2&=X_2+T_1\end{aligned}\right\} \tag{4.6}$$

セグメント2での処理を考えると，X_1, X_2 は次式で表わされる．

$$\left.\begin{aligned} X_1&=Y_1+T_2\\ X_2&=\max\left\{Y_2-\frac{1-\rho_2}{\rho_2}T_2,0\right\}\end{aligned}\right\} \tag{4.7}$$

以上の6式を連立して得られる T_1, T_2 が状態切換え間隔を示す．解析的には解けないので，利用率 $\rho=0.8, 0.9$ に対する数値例を図2に示す．横軸は到着率の小さい方のカテゴリ1の到着率 ρ_1 を示し，縦軸はカテゴリ1を処理している間隔 T_1 と，全体の周期

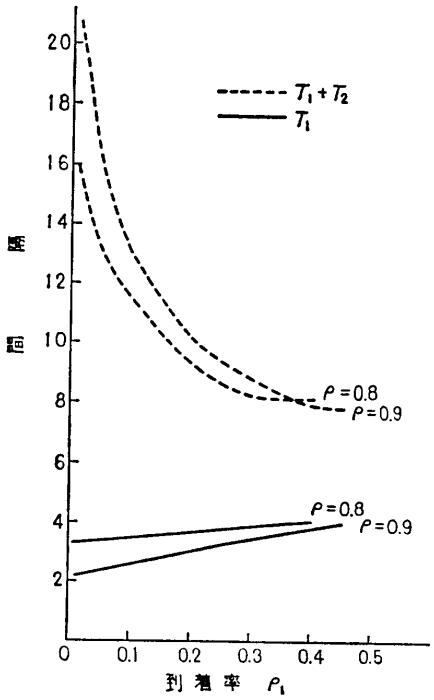


図2 到着率と状態切換え間隔の関係

Fig. 2 Relation between the arrival rate and the state change interval.

T_1+T_2 を示す．ρ_1 による T_1 の変化は少ない．すなわち，到着率にあまり関係なく，一定数のタスクが連続して処理される．一方，到着率の大きいカテゴリでは，相手の到着率が少し大きくなっても処理時間 T_2 は急激に減少する．カテゴリ1に一定のタスクが集まるまでに要する時間は ρ_1 に反比例するからである．カテゴリ1のタスクがまとめて処理される数は，システムの利用率 ρ，状態切換え損失と優先度の比の2つの要因で決まる．

4.2 到着率の小さなカテゴリの影響

稀にしか利用されない状態の存在がシステム全体に及ぼす影響を調べる．前節の議論で，カテゴリ1の利用率 ρ_1 が極端に小さいものとする．この場合，セグメント1の終わりでは，カテゴリ1に属すタスクは何も残っていない．カテゴリ1のタスクの平均待ち時間を求めるため，セグメント1と2それぞれの時に到着したタスクの待ち時間を計算する．

セグメント1の始めから t 時間後 $(0\leqq t\leqq T_1)$ に到着したカテゴリ1のタスクの待ち時間 $W^{(1)}(t)$ を求める．この時，待ち行列内のカテゴリ1のタスクの処理時間の和の期待値は $\{\rho_1 T_2-(1-\rho_1)t\}$ である．また，その時に実行中であるタスクの残りの処理時間の期待値は，待ち行列理論より $(T_1+T_2)\rho_1 T/2T_1$ である．これらの和が待ち時間となる．

$$W^{(1)}(t)=\frac{1}{2}\frac{T_1+T_2}{T_1}\rho_1 T+\rho_1 T_2-(1-\rho_1)t \tag{4.8}$$

次にセグメント2の始めから t 時間後 $(0\leqq t\leqq T_2)$ に到着したカテゴリ1のタスクの待ち時間の期待値 $W^{(2)}(t)$ を求める．この時，待ち行列内のカテゴリ1のタスクの処理時間の和の期待値は $\rho_1 t$ である．これらのタスクはセグメント2の間には処理されず，次のセグメント1で処理される．ゆえに，待ち時間はセグメント2が終わるまでの時間と $\rho_1 t$ との和である．

$$W^{(2)}(t)=(T_2-t)+\rho_1 t \tag{4.9}$$

カテゴリ1の平均待ち時間 $\tilde{W}$ は $W^{(1)}(t)$ と $W^{(2)}(t)$ の平均である．

$$\tilde{W}=\left[\int_0^{T_1}W^{(1)}(t)\,dt+\int_0^{T_2}W^{(2)}(t)\,dt\right]\Big/(T_1+T_2)$$

$$=\frac{1}{2}\rho_1(T+T_1+T_2)+\frac{1}{2}(T_2-T_1) \tag{4.10}$$

システムの利用率 ρ が1に近いと，カテゴリ1のタスクの処理で済むとすぐ，その処理装置はほかの状態に切り換わる．つまり，T_1 と T_2 に次の関係が成り立つ．

$$\rho_1(T_1+T_2)=T_1 \qquad \therefore\ T_2=\frac{1-\rho_1}{\rho_1}T_1 \tag{4.11}$$

この関係は図2でも明らかであり，これを(4.10)式に代入する．

$$\tilde{W}=\frac{1}{2}\rho_1 T+\frac{1}{2}\frac{1-\rho_1}{\rho_1}T_1$$

$$=\frac{1}{2}\rho_1 T+\frac{1}{2}T_2 \tag{4.12}$$

T_1 は ρ_1 によってほとんど変化しないので，待ち時間は到着率にほぼ反比例する．

単純な先着順で処理した場合，平均待ち時間は個々のカテゴリの到着率と関係なく，$\rho T/2(1-\rho)$ となる．これを(4.12)式と比較すると，状態を考慮した場合の方が待ち時間が増加する．しかし，その分だけカテゴリ2の平均待ち時間は小さくなっており，状態切換えによる損失はシステムのオーバヘッドとして，処理システムの時間を消費するので，全体としては本アルゴリズムの方が有利である．

次に状態切換え回数を調べる．本スケジューラの場合，状態切換えは (T_1+T_2) 時間ごとに2回起きるので，単位時間当りの切換え回数 N_c は次のようになる．

$$N_c=\frac{2}{T_1+T_2}=\frac{2\rho_1}{T_1} \tag{4.13}$$

先着順の場合，このカテゴリに属すタスクは稀にしか到着しないので，連続して2以上のタスクが処理されることはない．1つのタスクが到着すると前後2回の切換えが起きるので，単位時間当りでは $2\rho_1/T$ となる．本スケジューラを用いると，先着順に比べ，状態切換え回数は T/T_1 に減少する．この比は諸条件によって変化するが，図2の場合，1/2〜1/3 程度である．すなわち，どんなに到着率が小さくなろうとも，2〜3個のタスクをまとめて処理する．

5. シミュレーション

本アルゴリズムの性能を調べるため，シミュレーションを行う．ジョブは複数のタスクからなる構造を持つので，前もって幾つかのパターンを用意しておき，乱数によってその中から選択する．ジョブの到着はポアソン分布に，タスクの処理時間は指数分布に従って決める．カテゴリの決定も乱数による．

シミュレーションを行うには何らかの形で損失係数を決めなくてはならない．決定的な方法はないが，ここでは，タスクの属すジョブの全処理時間，ジョブ内

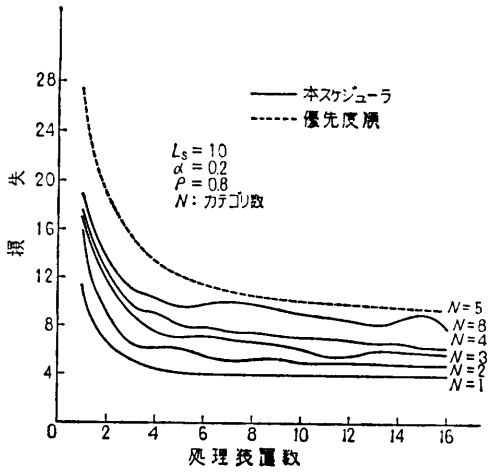


図 3 処理装置数と平均損失の関係

Fig. 3 Average loss as a function of the number of processors.

でのタスクの位置，タスクが実行可能になってからの経過時間という3要因に基づいて決める．損失係数を一度決めたまま固定すると，優先度の低いタスクが長く待たされる危険があるので，タスクの経過時間を反映するように，一定の間隔で損失係数を更新する．

状態切換え時点と，切り換えるべき状態 i, j の決定は次のようにする．状態切換え可能な時点で，$\beta_i^{(+)}(0)$ が最大となる j，および $\beta_i^{(-)}(0)$ が最小となる i を選び，これらが，(3.15)式の関係を満たすかを調べる．もし成立すればそのカテゴリ間で1台の処理システムを切り換える．さらに残ったものについて，(3.15)式が成立しなくなるまで繰り返す．

状態切換えによる損失と，ジョブが投入されてから終了するまでの待ち時間による損失との和の時間平均としてシステムの性能を表わす．

まず，カテゴリ数Nをパラメータにとり，処理システムの数Mと損失の関係を表わしたのが**図3**である．各カテゴリに属すタスクの割合は等しく，$1/N$とした．タスクの処理時間の平均値を1とし，1回の状態切換えによる損失 $L_S=10$，割引き係数 $\alpha=0.2$，システムの利用率 $\rho=0.8$ とした．損失係数は40～100位の値である．カテゴリ数が増加する程，状態切換え回数が増えるので損失も増大する．処理システムの数が増加すると，待ち時間と状態切換え回数の両方が減少する．図3において，損失の減少が一様でないのは状態切換えによる影響と考えられる．処理システムの数とカテゴリの数により，各状態の処理システムの数が不安定になるからである．カテゴリ数の整数倍よりも少し多い数の処理システムが望ましい．

図3には本スケジューラを用いた場合と，単純な優先度順による場合の差も示している．実線が本スケジューラであり，点線が優先度順である．単なる優先度順といっても，余分な切換えはせず，割り当てるべきタスクの要求する状態の処理システムが空いていれば，必ずその処理システムに割り当てる．2つのスケジューラの性能を比較するとその差は歴然としている．処理システムが少なく，状態切換え回数が多い時

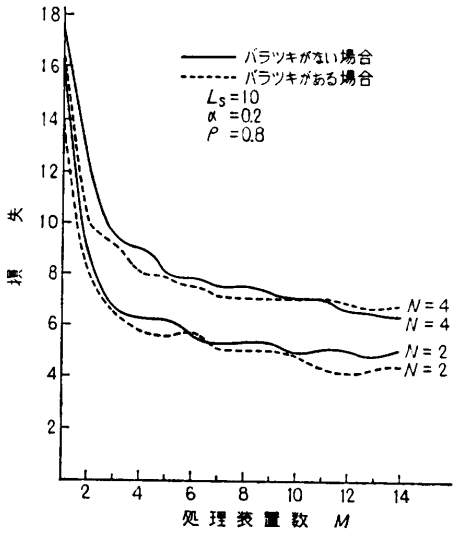


図 4 到着率のバラツキの影響

Fig. 4 The effect of non-uniform arrival rates.

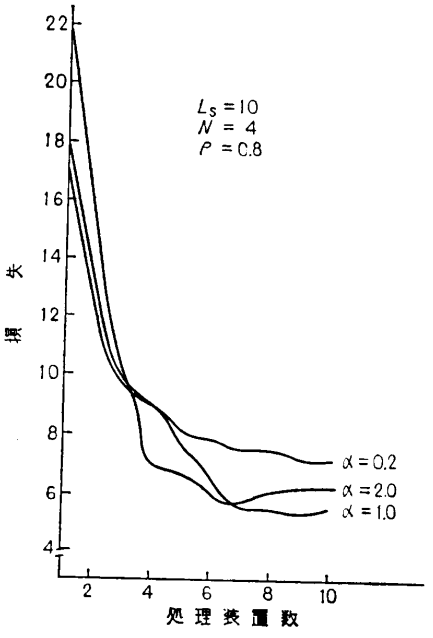


図 5 割引き係数 α の影響

Fig. 5 The effect of the discounting factor α.

にその差は顕著である．処理システムの数が多くなると，優先度順でも状態切換え回数は少なくなるので，状態を考慮した効果は少なくなる．

各カテゴリの割合にばらつきを持たせた時の影響を調べたのが**図4**である．1つのカテゴリの全体に対する割合を小さくし，ほかのカテゴリは同じ割合とする．この場合，同じカテゴリ数で割合が均等な場合よりも状態切換え回数が減少するので損失は少なくなる．

スケジューラの評価関数で使う割引き係数の影響を調べたのが**図5**である．αが大きい程，将来に生じるはずの損失の影響が小さくなり，αによって処理システムを増加した時の損失の減少方法が異なる．

最後に，本スケジューラ自身のオーバヘッドについて述べる．アルゴリズムが複雑で，スケジュールするごとに余分な時間を費やしては何もならない．(2.2)式を最小化するのに(3.5)式を直接用いるのでなく，3章で述べたようにすると比較的簡単であり，本スケジューラと優先度順スケジューラの所要時間をシミュレーションで比較したところ，約3倍であった．この数値は $\beta^{(+)}(0)$, $\beta^{(-)}(0)$ の計算に何の工夫もしない時である．これらの計算を簡素化すれば，もっと少ない時間でスケジュールできる．

6. 結　論

複数の処理システムで構成される計算機システムにおいて，処理システムに対して状態という概念を導入し，タスクを処理する時にはそれが要求する状態になる．処理システムが状態を切り換えるにはオーバヘッドが伴うので，これを一定の損失として表わす．このようなシステムに逐次ジョブが到着する場合の最適なスケジューリング・アルゴリズムを与え，その性質を論じた．

このスケジューラはタスクの平均待ち時間の増加を抑えつつ，処理システムの状態切換え回数を減少させる．この効果は処理システムの数に比べ，状態数が大きい時に顕著である．また，到着率の小さなカテゴリに対し，タスクが到着する度にその状態に切り換えることはせず，幾つかまとめて一括処理する．そのため，状態切換えによるオーバヘッドが抑えられ，到着率の小さなカテゴリが存在してもシステムの性能低下は避けられ，良好な結果が得られる．

従来の先着順などの簡単なスケジューラに比べ，アルゴリズムはやや複雑になったが，2～3倍程度で抑えられることがシミュレーションにより判明した．

本スケジューラは損失係数の決定には関与せず，与えられた損失係数がその後も変化しないものとして先読みを行う．そのため，損失係数をどのように決定するかにより，システム全体の挙動は大きな影響を受ける．損失係数が変化しても，変化がそれ程大きくなければ指数関数の重みで弱められるので，影響は少ない．しかし，タスクの損失係数が極端に変化する場合には先読みの効果が薄れ，必ずしも最適性は保障されない．先読みを行う場合，未来のことは統計的にしか分からないので，平均的事象が起こる場合を想定して行わざるを得ない．

今後とも大型計算機システムは複数の処理システムで構成される傾向にある．また，処理対象の複雑化に伴い，処理システムに処理状態という概念を導入する必要も高まりつつある．1つの処理システム内で多重処理する場合にもこのアルゴリズムは適用可能であり，今後ますますこのようなアルゴリズムの必要性は増加するであろう．

謝辞　本研究を進めるに当たり，貴重なご意見を賜わった東京大学穂坂衛教授，ならびに，東京工業大学片山卓也助教授に深く感謝いたします．

参考文献

1) Ramamorthy, C. V., Chandy, K. M. and Gonzalez, Jr. M. J.: Optimal Scheduling Strategy in a Multiprocessor System, IEEE tras. on Comp., Vol. C-21, No. 2 (Feb. 1972).
2) Hu, T. C.: Parallel Sequencing and Assembly Line Problems, Operations Research, Vol. 9, pp. 841-848 (Nov. 1961).
3) Sevcik, K. C.: Scheduling for Minimum Total Loss Using Service Time Distribution, Journal of ACM, Vol. 21, pp. 66-75.
4) Schrage, L.: Optimal Scheduling Disciplines for a Single Machine under Various Degrees of Information, 44-th ORSA Conf., BOSTON (Apr. 1974).
5) Bruno, J. and Hofri, M.: On Scheduling Chains of Jobs on One Processor with Limited Preemptions, SIAM Journal Computing, Vol. 4, No. 4 (Dec. 1975).
6) Ullman, J. D.: NP-complete Scheduling Problems, JCSS, Vol. 10, pp. 384-399 (1975).
7) Coffman, E. G. Jr. and Denning, P. J.: Operating Systems Theory, Prentice-Hall (1973).
8) Hansen, P. B.: Operating System Principles, Prentice-Hall (1973).
9) Bruno, J., Coffman, E. G. Jr. and Sethi, R.:

Scheduling Independent Tasks To Reduce Mean Finishing Time, Com. of ACM, Vol. 17, No. 7 (July 1974).
10) 榎本，片山，宮村，熊西：スケジューリングにおける切換え時点の最適決定法，信学会資料 EC 76-82 (1977-3).

(昭和54年11月16日受付)
(昭和55年12月18日採録)