
# 23. オペレーティング・システムのスケジュール法†

亀田　壽夫††

## 1. はじめに

計算機システムは，中央処理装置 (CPU)，チャネル，入出力装置 (I/O)，主記憶，補助記憶，等のハードウェア資源から構成される．このようなシステムにより，複数個のプログラム（ジョブ）が並行して処理されるとき，各資源において競合がおこり，待ちが生ずる．このような待ちから来る無駄を，適当なスケジュール法を用いて，軽減するのが，オペレーティング・システム（以下 O.S. と記す）の役割の一つである．すなわち，O.S. の目的の中に，ジョブに対する応答性を良くすること，さらに資源全体の使用効率を高くすることがある．

本稿の目的は，O.S. のスケジュール法として適当なものを紹介することにある．しかし，O.S. の最適スケジュール法が，現実を反映する十分複雑なモデルを用いて理論的に求められているとはいえない．特に，近年，積形解 (product-form solution) による待ち行列ネットワークモデルの解析方法が大いに発展したが，その方法により評価できるスケジュール法が極めて限定されることも明らかになった[1)~3)]．

したがって，取り得る次善の方策として，次のものが考えられる：すなわち，比較的単純で解析可能なモデルにおける結果から得られた直観・見通しに基づき，現実の複雑な状況から得られる経験に即して，採用すべきスケジュール法を決定する，というものである．本稿では，この線に沿って，まず2節において，単純なモデルにおける，見通しを与えるような理論的結果を紹介し，次に3節において，現実の場面で考えられたスケジュール法について論ずる．

## 2. 待ち行列スケジューリングの基礎

この節では，まず，待ち行列の諸スケジュール法を分類し特徴づける (**2.1**節)．次に，各スケジュール法の間に成立する基本的性質を示す (**2.2**節)．さらに，与えられた性能指標を最適化するスケジュール法について論ずる (**2.3**節)．

### 2.1 スケジューリング・アルゴリズム[4)~7)]

本稿で考えるスケジュール法は，**work-conserving**（処理装置にジョブが1つ以上あれば必ず処理が行われ，また各ジョブはその所要処理量を完了するまでは処理装置を離れない）とする．work-conservingであるスケジュール法は，2.2節に記すように G/G/1 の保存則を満足させる．現実に O.S. で使われるスケジュール法は，ほとんどみな work-conserving であると考えてよい．

処理が開始されたジョブが，完了まで中断なく処理されるようなスケジュール法は，**nonpreemptive** であるといい，ジョブ処理を中断して他のジョブを処理することがあるものは，**preemptive** であるという．preemptive 方式の場合，中断されたジョブの処理が，中断された時点の状態から再開されるものは，特に，**preemptive-resume** であるという．CPU は，割込み可能であるので，preemptive-resume 方式を採用できる．I/O の多くはその性質上 nonpreemptive 方式のみを採用できる．

さらに，スケジュール法は，その決定において次の条件を満たすかどうかで分類される．

**条件 I**．ジョブの処理順の決定において，各ジョブやシステムの現在の状態および過去のふるまいに関する情報のみが用いられる．

通常よく用いられるスケジュール法の多くは条件 I を満たす：たとえば，FCFS (First Come First Served ——先着順処理)，LCFS (Last Come First Served)，LCFSPR (Preemptive Resume Last Come First Served)，Round Robin [各ジョブを，一定時間 (**time quantum**) ずつ処理し，time quantum 終了ごとに待ち行列の最後尾につなぐ]，Processor Sharing (処理装置に $n$ 個のジョブがあると，各ジョブを並行して

† Scheduling in Operating Systems by Hisao KAMEDA (Department of Computer Science, The University of Electro-Communications).

†† 電気通信大学電気通信学部計算機科学科

**表-1** work-conserving であるスケジュール法の分類

| | スケジューリングの決定において | |
|---|---|---|
| | 現在の状態および過去のふるまいに関する情報のみを用いるもの | 将来の状態に関する情報を用いるもの |
| preemptive (resume) 方式 | LCFSPR, Round Robin Processor Sharing, Generalized Processor Sharing SEPT, SERPT, LEPT, LERPT, Preemptive Priority 等 | SPT, SRPT LPT, LRPT 等 |
| nonpreemptive 方式 | FCFS, LCFS Nonpreemptive Priority 等 | Nonpreemptive SPT Nonpreemptive LPT 等 |

各 $1/n$ の速さで処理する——Round Robin において time quantum の長さを無限小にした極限の方式と考えられる), Generalized Processor Sharing (処理装置の能力を各ジョブに異なる比率で分配し並行処理する——比率を等しくすると Processor Sharing 方式となる), Preemptive Resume および Nonpreemptive Priority, 等. また過去のふるまいから, 各ジョブの所要処理時間の期待値等が求められるので, 次の方式も条件Ⅰを満たす: SEPT (Shortest Expected Processing Time First), SERPT (Shortest Expected Remaining Processing Time First), LEPT (Longest Expected Processing Time First), LERPT (Longest Expected Remaining Processing Time First), 等.

しかし, 各ジョブの正確な所要処理時間を前もって知ることを要するスケジュール法は条件Ⅰを満たさない: たとえば, SPT (Shortest Processing Time First), SRPT (Shortest Remaining Processing Time First), LPT (Longest Processing Time First), LRPT (Longest Remaining Processing Time First), 等がある (**表-1** 参照).

### 2.2 スケジューリングの保存則と実現可能な応答性能

一般に, 各ジョブクラスに対する平均応答時間のような性能指標の値は, スケジュール法に依存する. しかし, スケジュール法を適当に変えてすべてのジョブクラスの平均応答時間を短くするというようなことはできない. すなわち, 各ジョブクラスごとの実現可能な性能指標値の間には何らかの拘束がある. 本節では, そのような拘束あるいは関係について論ずる.

以下, 処理装置の能力は, 処理要求中のジョブ数によらず不変で, また装置台数は一台とする. 処理装置のスケジュール法はみな work-conserving とする. 各ジョブは, クラス $1,2,\cdots,N$ に分類されるとする. 集合 $(1,2,\cdots,N)$ を $\boldsymbol{N}$ と記す. 次の記法を用いる.

$\lambda_j$: クラス $j$ ジョブの到着率

$\bar{s}_j$: クラス $j$ ジョブの所要処理時間の平均

$\overline{s_j^2}$: クラス $j$ ジョブの所要処理時間の自乗の平均

$T_j$: クラス $j$ ジョブの平均応答時間 (ジョブが処理装置に到着してから, 処理完了し出ていくまでの時間の平均)

$\rho_j=\lambda_j\bar{s}_j$: クラス $j$ に対する処理装置の利用率

$\rho=\Sigma_{j\in N}\ \rho_j$: 処理装置の利用率

$\boldsymbol{T}$: 応答時間ベクトル $(T_1, T_2, \cdots, T_N)$

(A) ジョブ到着が, 処理装置の状態やスケジュール法によらない (infinite-source の) 場合: 次の3つの場合を考える.

(A-1) G/G/1——各クラスのジョブの到着過程も, 所要処理時間分布も, 任意の場合:

任意の時点において, 処理装置には到着済みであるが処理が未完了のジョブの, 残余処理時間の総和を考える. この量を, その時点における **unfinished work** と呼ぶ. work-conserving なスケジュール法のみを考えると, unfinished work の期待値は, スケジュール法によらない (これは条件Ⅰが満足されなくても成立する). これを G/G/1 の保存則という.

(A-2) クラスごと M/M/1——各ジョブクラスの到着時間間隔も所要処理時間も指数分布をなす場合:

work-conserving で条件Ⅰを満たす (preemptive 方式を含む) スケジュール法のみを考える. スケジュール法Sを用いた場合に応答時間ベクトルが $\boldsymbol{T}$ になるとき, スケジュール法Sが応答時間ベクトル $\boldsymbol{T}$ を**実現する**という. 任意の応答時間ベクトル $\boldsymbol{T}$ を実現する (上述のような) スケジュール法が存在するための必要十分条件は次のとおりである[6),8)]:

$$\Sigma_{j\in N}\ \rho_jT_j=(1-\rho)^{-1}\Sigma_{j\in N}\ \rho_j\bar{s}_j, \qquad (2.1)$$

および $\boldsymbol{N}$ の任意の空でない真部分集合 $\boldsymbol{Z}$ に対して,

$$\Sigma_{j\in Z}\ \rho_jT_j\geqq(1-\Sigma_{j\in Z}\ \rho_j)^{-1}\Sigma_{j\in Z}\ \rho_j\bar{s}_j. \qquad (2.2)$$

(2.1) は保存則をあらわしている.

(A-3) クラスごと M/G/1——各ジョブクラスの到着時間間隔は指数分布をなすが, 所要処理時間が任意の場合:

work-conserving で条件Ⅰを満たし nonpreemptive なスケジュール法のみを考える. 任意の応答時間ベクトル $\boldsymbol{T}$ を実現する (上述のような) スケジュール法が存在するための必要十分条件は次のとおりである[6)]:

$$\Sigma_{j\in N}\ \rho_jT_j=\omega_0\rho/(1-\rho)+\Sigma_{j\in N}\ \rho_j\bar{s}_j, \qquad (2.3)$$

および，$N$の任意の空でない真部分集合$Z$に対して，

$$\Sigma_{j\in Z}\ \rho_j T_j \geqq \omega_0(\Sigma_{j\in Z}\ \rho_j)/(1-\Sigma_{j\in Z}\ \rho_j)^{-1} + \Sigma_{j\in Z}\ \rho_j s_j \qquad (2.4)$$

ただし，$\omega_0=\Sigma_{j\in N}\ (1/2)\lambda_j \overline{s_j^2}$.

(2.3) は保存則をあらわしている．

(A-2)，(A-3) の場合，実現可能な応答時間ベクトル$T$の領域は，超多面体をなし，凸領域であることに注意されたい．

(A)において，処理装置へのジョブ到着がスケジュール法によらないと仮定した．タイムシェアリングシステムにおける計算機や，多重プログラミングシステムにおける CPU を，処理装置とみなしたとき，この仮定は現実的ではない．したがって次のような場合を考えることにする．

(B) ジョブ到着が，処理装置の状態やスケジュール法に依存する場合：

この場合の性質を一般的に求めることは非常に困難であるが，次のように単純化された (finite-source の) 場合には，性質が得られている：クラス $j$ のジョブが処理装置にあるときは，そのクラスの到着はない (すなわち，処理装置にある各クラスのジョブ数は高々1つである)．クラス $j$ ジョブの，所要処理時間は，平均 $\mu_j^{-1}$，処理装置を出てから次に到着があるまでの時間は，平均 $\lambda^{-1}$ で，いずれも指数分布をなすとする．(A-2)と同様，処理装置において，work-conserving で条件を満たす (preemptive 方式を含む) スケジュール法が用いられるとする．このとき，任意の応答時間ベクトル$T$を実現するスケジュール法が存在するための必要十分条件は，次のとおりである[7),9),10)]：

$$\Sigma_{j\in N}\ U_j = U(N) \qquad (2.5)$$

および，$N$の任意の空でない真部分集合$Z$に対して，

$$\Sigma_{j\in Z}\ U_j \leqq U(Z) \qquad (2.6)$$

ただし，$U_j=1/[\mu_j(T_j+1/\lambda)]$ および，$U(M)=1-(\Sigma_{n=0}^{|M|} n!\ \Sigma_{I\subseteq M,\,|I|=n}\ \Pi_{j\in I}\ \lambda/\mu_j)^{-1}$.

(2.5)は保存則をあらわしている．また $U_j$ は，クラス $j$ ジョブによる処理装置利用率をあらわしている．(この場合，$U_j$ は，一般に，スケジュール法に依存する．)

実現可能なベクトル $U=(U_1, U_2, \cdots, U_N)$ の領域も超多面体をなし，凸領域であることに注意されたい．

### 2.3 最適スケジュール法[4)~6)]

性能指標のうち，応答時間ベクトル$T$や，ベクトル$U$から直ちに算出されるものが数多くある．前節の(A-2)，(A-3)，(B)で述べた場合においては，このような性能指標を最適化する問題は，凸制約条件を持つ数理計画法に帰着されることがわかる[10)]．たとえば，全応答時間の平均$T$を考えてみよう．(A-2)，(A-3)の場合には，$T=(\Sigma_{j\in N}\ \lambda_j T_j)/(\Sigma_{j\in N}\ \lambda_j)$．(B)の場合には，$T=(\Sigma_{j\in N}\ \mu_j U_j T_j)/(\Sigma_{j\in N}\ \mu_j U_j)$ となる．これらの場合，平均所要処理時間 ($\bar{s}_j$ あるいは $\mu_j^{-1}$) の短いジョブクラスほど高い優先順位を与えて処理する方式 (SEPT) によって，$T$が最小になることが容易にわかる．

平均応答時間$T$に限っていえば，(A-1)の場合，条件Iをはずすと，SRPT が $T$ を最小にすることが知られている[11)]．さらに，いくつかのスケジュール法について，それぞれの与える$T$の値の間の大小関係が知られている[5),12)] (たとえば，SPT よりも FCFS の方が，FCFS より LPT の方が，大きい $T$ の値を与える)．

## 3. オペレーティング・システムのスケジューリング

ジョブが計算機システムによって処理されるには，まずそのジョブが主記憶にロードされることが必要で，それからそのジョブが CPU や I/O を使用する．したがって，計算機システムにおける O.S. によるスケジューリングの決定には，次の2段階が含まれる．

(i) 主記憶に置くジョブの集合 (job mix と呼ぶ) を決定すること，

(ii) 各処理装置 (CPU，周辺装置) における待ち行列のスケジュール．

本節では，まず，各種処理装置のスケジューリングについて論じ (**3.1**, **3.2** 節)，次に job mix の決定にかかわるシステム資源管理について述べる (**3.3** 節)．

### 3.1 CPU スケジューリング

CPU における応答性に対する，各ジョブからの要求は様々である：TSS ジョブは速い応答を要するが，バッチジョブは応答性の要求が低い．CPU は割込み可能であるので，preemptive スケジュール法を適当に用いることにより，各ジョブからの応答性要求に応えることができる．

各ジョブの CPU 使用特性も様々である：あるジョブは，1回の CPU 処理の所要時間が短い (I/O bound) が，あるものは，入出力要求による中断があるまでの，1回の CPU 処理の所要時間が長い (CPU

bound). CPU において，I/O bound ジョブに高い優先順位を与えると，計算機システム全体の処理能力（スループット）が高くなることが，経験的に示されてきた[13)~17)]．また，そのようなスケジュール法が，数多くの計算機システムにおいて実際に用いられている（このような方式を **dynamic dispatching** 等と呼ぶ）．理論的にも，限定された場合であるが，その最適性が示されている[18),19)]．この方式を実際に用いる場合に問題になるのが，各ジョブの，I/O bound/CPU bound の程度を予測する方法である．1つは，time quantum を用いて実現する方式であり，Round Robin はその一例である．また，各ジョブの過去のふるまいに基づいて予測する方式もある．いずれにせよ，I/O bound ジョブに高い優先順位を与える方式は，処理能力を向上させることにより全体として応答性を良くすることにねらいがある．

割込みは，CPU におけるオーバヘッドを必然的に伴うので，preemptive スケジュール法については，オーバヘッドを，状況に応じて，考慮する必要がある．

### 3.3 ディスク，ドラムのスケジューリング[20)~22)]

磁気ディスクや磁気ドラムのような補助記憶装置は，機械的動作を伴うので，アクセス時間が主記憶装置に比べて非常に長いことが，大きな問題となる．

ドラムやディスクにおいて，待ち時間も含めた平均アクセス時間を最小にするスケジュール法として，各時点のアームやヘッドの位置からのアクセス時間の，最も小さい場所への，アクセス要求を，最優先処理する方式がある．この方式は，ドラムやディスクの回転については，**SATF** (Shortest Access Time First) 等と呼ばれ，FCFS 法の数倍の処理能力をもたらすことが解析的に示されている．また，この方式は，ディスクのアームのシーク操作については，**SSTF** (Shortest Seek Time First) と呼ばれる．SSTF 法によれば，アームが，両端の近くにあるシリンダに行くことが少なくなり，アクセスするシリンダが両端に近くなるにつれ，応答時間が長くなるという問題がある．これに対処するために SCAN（アームを動かす方向を，端に至るまで変えない）法，等が考えられている．

上述のスケジュール法の解析は，通常，アクセス要求到着がポアソン過程をなすと仮定して行われている．しかし，SSTF 法については，いろいろな人により解析結果が提出されているが，まだ確実なものが得られていないともいわれている[23)]．いずれにせよ，実測によると，通常，ディスクにおける待ちは小さく，あるいは，小さくなるようにシステム構成がなされるので，アームのスケジューリングの効果は少ないともいわれている[22)]．

### 3.3 システム資源管理

システム全体の処理能力の向上と，各ジョブの応答性要求の充足とを目的とした，計算機システムのジョブ・スケジューリングは，まず job mix の決定により行われる．このような job mix の決定は，CPU や I/O におけるスケジュールの効果の考慮も要する，非常に複雑な問題である．これまでの，わずかに得られている解析的見通しからも，その困難さが示唆される[24)]．

一方，その目的の実現に対して，実際的使用環境からの要求は大きく，そのため，現実の O.S. において，次のような自動制御方式が実現された：その方式では，O.S. が，システム全体の利用状態や，各ジョブに対する応答性の状態を，絶えずモニタし，その測定値に基づいて，各時点における適当な job mix の選択が，(O.S. によって) 自動的に行われる[25)]．

この方式において，job mix の制御の手段として，**swapping**（ジョブのプログラムを主記憶から補助記憶へ，あるいはその逆方向へ，転送すること）が用いられた．しかし，swapping には，非常に多くの資源が費やされ，その消費量は，CPU における割込みの場合の比ではない．この方式の下で，制御のための swapping が多発し性能が低下することが，しばしば観測された[26)]．その結果，後に，job mix の制御を，swapping のみによらずに，**domain** という方式を通して間接的にも行う方式が実現された[26)]．domain は，ジョブクラスに対応し，前もって定められた多重度のわくと重みとを持つ．domain の多重度は，システムの混雑度に応じて，O.S. により調整される．この方式は，job mix の自動制御という観点からは，少し後退したものと考えることができよう．[他の改良方式としては，例えば 27) 参照．]

システムの処理能力が，スケジュール法により変わらないと考えられる場合には，事情がはるかに簡単になる．2.2 節の (A) の場合の見通しが使えるし，また応答性要求が **policy function** の形に表現される場合の理論的結果も得られている[28)]．

## 4. おわりに

この分野においても，他の多くの分野と同様に，理論的に厳密に分析される範囲と，現実のシステムの行

動範囲との間には大きな違いがある．しかし，現実のシステムを記述するパラメータの数は非常に多く，その値の十分な数の組合せについて，直接に実験を行い，スケジューリングの最適性を正当に理解することは，不可能に近い．一方，理論的解析の方は，直接その結果が現実に応用できない場合が多いとしても，現実のシステムの複雑な行動を理解するための直観や，性能向上のための見通しを与える点においては，少しずつ進歩している．今後もその進展が注目されることが期待される．

## 参 考 文 献


1) Baskett, F. et al.: Open, Closed, and Mixed Networks of Queues with Different Classes of Customers, J. ACM, Vol. 22, No. 2, pp. 248-260 (1975).
2) Chandy, K. M. et al.: Product Form and Local Balance in Queueing Networks, J. ACM, Vol. 24, No. 2, pp. 250-263 (1977).
3) Kelly, F. P.: Reversibility and Stochastic Processes, Wiley, N. Y. (1979).
4) Conway, R. W. et al.: Theory of Scheduling, Addison-Wesley, Mass. (1967).
5) Kleinrock, L.: Queueing Systems, Volume II: Computer Applications, Wiley, N. Y. (1976).
6) Gelenbe, E. and Mitrani, I.: Analysis and Synthesis of Computer Systems, Academic Press, London (1980).
7) 益田，亀田：オペレーティングシステムの性能解析，情報処理叢書 9，情報処理学会 (1982).
8) Coffman, E. G., Jr. and Mitrani, I.: A Characterization of Waiting Time Performance Realizable by Single-Server Queues, Oper. Res., Vol. 28, No. 3, Part II, pp. 810-821 (1980).
9) Kameda, H.: A Finite-Source Queue with Different Customers, J. ACM, Vol. 29, No. 2, pp. 478-491 (1982).
10) 亀田壽夫：応答性能の実現可能性，情報処理学会「計算機システムの制御と評価」研究会資料 17，pp. 17-2-1～17-2-6 (1982).
11) Schrage, L. E.: A Proof of the Optimality of the SRPT Discipline, Oper. Res., Vol. 16, No. 3, pp. 687-690 (1968).
12) Suzuki, T. and Hayashi, K.: On Queue Disciplines, J. Operations Research Society of Japan, Vol. 13, No. 2, pp. 43-58 (1970).
13) Marshall, B. S.: Dynamic Calculation of Dispatching Priorities under OS/360 MVT, Datamation, Vol. 15, pp. 93-97 (1969).
14) Ryder, K. D.: A Heuristic Approach to Task Dispatching, IBM Syst. J., Vol. 9, No. 3, pp. 189-198 (1970).
15) Scherman, S. et al.: Trace-Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System, Comm. ACM, Vol. 15, No. 12, pp. 1063-1069 (1972).
16) Stevens, D. F.: On Overcoming High Priority Paralysis in Multiprogramming Systems: A Case History, Comm. ACM, Vol. 11, No. 8, pp. 539-541 (1968).
17) Wulf, W. A.: Performance Monitors for Multi-programming Systems, Proc. 2nd Symp. on Operating System Principles, ACM, N. Y., pp. 175-185 (1969).
18) 亀田壽夫：ジョブストリーム処理時間を最短にする CPU スケジューリング，情報処理学会「計算機システムの解析と制御」研究会資料 14，pp. 14-3-1～14-3-10 (1981).
19) Kameda, H.: Optimality of a Central Processor Scheduling Strategy in Multiprogrammed Computer Systems, Trans. IECEJ, Vol. E 64, No. 3, pp. 120-125 (1981).
20) Coffman, E. G., Jr. and Denning, P. J.: Operating Systems Theory, Prentice-Hall, N. J. (1973).
21) Fuller, S. H.: Analysis of Drum and Disk Storage Units, Springer-Verlag, Berlin (1975).
22) Smith, A. J.: Input/Output Optimization and Disk Architectures: A Survey, Performance Evaluation, Vol. 1, No. 2, pp. 104-117 (1981).
23) Hofri, M.: Disk Scheduling: FCFS vs. SSTF Revisited, Comm. ACM, Vol. 23, No. 11, pp. 645-653 (1980).
24) Kameda, H.: Appraisals of Scheduling Decisions in Computing Systems Using a Finite-Source Queueing Model, Proc. 8th Int. Symp. on Computer Performance Modelling, Measurement and Evaluation, North-Holland, Amsterdam, pp. 455-468 (1981).
25) Lynch, H. W. and Page, J. B.: The OS/VS 2 Release 2 System Resources Manager, IBM Syst. J., Vol. 13, No. 4, pp. 274-291 (1974).
26) Beretvas, T.: Performance Tuning in OS/VS 2 MVS, IBM Syst. J., Vol. 17, No. 3, pp. 290-313 (1978).
27) Nishigaki, T. et al.: An Experiment on the General Resources Manager in Multiprogrammed Computer Systems, J. Information Processing, Vol. 1, No. 4, pp. 187-192 (1979).
28) Ruschitzka, M.: The Performance of Job Classes with Distinct Policy Functions, J. ACM, Vol. 29, No. 2, pp. 514-526 (1982).


(昭和 57 年 11 月 29 日受付)