

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

亀田 壽夫 (電気通信大学)

要旨: I/O-bound ジョブとCPU-bound ジョブとから成るベンチマーク・ジョブストリームを処理する多重プログラミングシステムのマルコフモデルを解析した。そのモデルにおいて, I/O-bound ジョブにより高い先取優先権 (pre-emptive priority) を与える CPU スケジューリング方式が, CPU 利用率を最大にする (スループットを最大にする) ことを示した。

1. はじめに

多重プログラミングシステムにおいて, 中央処理装置 (CPU) が性能上のボトルネックになることが多い。一方, 各ジョブが用いる CPU 量に大きな違いがあることが知られている: つまり, あるジョブは, 相対く2つの入出力要求の間に CPU を少ししか用いない (I/O-bound) のに対し, あるジョブは, CPU を多く用いる (CPU-bound)。上述のボトルネックを除去するために, 次のような CPU スケジューリング方式が既に提案されている: つまり, CPU において, I/O-bound ジョブに, より高い preemptive priority を与える方式である。この方式を 'H-schedule' と呼ぶことにしよう。

数多くの実験において, この方式が, 他の簡単に実現できる方式よりも, より多くのスループットをもたらすことが示されてきた [1, 6, 8, 9, 11, 14]。その中でも特に実験的条件をはっきり示しているものは, 次のように記述していきましょう, いろいろな種類のジョブの系列であるジョブストリームが, 幾つか用意される。(これを, ベンチマークジョブストリーム, あるいは単に ジョブストリーム と呼ぶことにしよう; 更に, 主メモリに共有し, ほぼ同時に並行処理されるジョブの集まりを, ジョブミックス と呼ぶことにしよう。) 次に, 多重プログラミングシステムに, 同じジョブストリームを, CPU スケジューリング方式をいろいろ変えて処理させる。そして, 性能指標として, ジョブストリーム処理終了までの経過時間や, 全体にわたる CPU 利用率の値等を得る。そして次のような結論が得られている:

- (1) CPU 利用率は, CPU において H-schedule が用いられるとき最大になる。
- (2) またそのとき, 全経過時間が最小となる。

多重プログラミングシステムの性能解析をするとき, central-server model や finite-source queueing model (機械修理モデル) が用いられる。前者の方がより現実的に思われるが, 後者が多重プログラミングシステムの良しいモデルになるとの報告もある [2]。いずれにせよ, 入出力装置における待ち合わせが無視できるとき, finite-source queueing model はよいモデルになり得る (このことは, 必ずしも, 入出力装置の利用率が低くならなければならないことを意味

しないことに注意されたい。例えば、各 I/O が、それをよく用いる一つのジョブに専用される場合を考えることができる。しかし、前述のような待ち行列モデルを用いて CPU スケジューリング方式の効果を解析した結果は殆ど変わったように思われる。

最近、多重プログラミングシステムの finite-source queueing model に関して次のような結果が得られている [5]。モデル内の各ジョブのサービス時間分布が、CPU においてジョブ毎に異なる平均を持つ指数分布で、I/O において同一平均を持つ指数分布であると仮定しよう。また、CPU スケジューリング方式が、FCFS, preemptive あるいは nonpreemptive priority, generalized processor sharing 等を含む諸方式の一つであるとしよう。そうすると、次の性質が成立つ：

- I. CPU の利用率は、CPU スケジューリング方式によらず、一定である。
 - II. CPU と I/O の全ての装置の利用率の総和（つまり、実効多重度）が最大になるためには、H-schedule が CPU で用いられることが必要十分である。
- 一見すると、性質 I は、前述の結論 (1) と相反するように思われる（性質 II は、結論 (2) と調和するように思われるが）。

上の理論的結果と実験による結論との間の隔りは、次のような直観的議論で埋められるように思われる：つまり、H-schedule の下では、I/O-bound ジョブの優先度が最も高いので、ジョブミックス中に滞在する時間が最も短くなる。同様に、CPU-bound ジョブの滞在時間は最も長くなる。CPU-bound ジョブをより多く含むジョブミックスは、より高い CPU 利用率を示すと考えられるので、H-schedule の下で CPU 利用率は最大になる、というのである。本論文では、H-schedule が最適であることと、上述の直観的議論が正当であることを、ジョブストリーム処理のマルコフモデルを用いて示す。

2 節において、ジョブストリームと、多重プログラミングシステムの、モデルと仮定を述べる。

3 節において、モデルを解析し、H-schedule の最適性を示す。モデル全体が、2 段階 nearly completely decomposable であることを示す：下の段階はジョブミックス処理段階と呼ばれ、finite-source queueing model を用いて解析される；その結果を用いて、ジョブミックス遷移段階と呼ばれる上の段階で、H-schedule の最適性が証明される。

2. モデル

モデル全体は、ジョブストリームとそれを処理する多重プログラミングシステムとから成る。それらに関する仮定をこの節で述べる。

2.1 ベンチマーク・ジョブストリーム

次のようなジョブストリームを考える。ジョブストリーム内の全てのジョブは、クラス1 (I/O-bound ジョブ) と、クラス2 (CPU-bound ジョブ) の2つのクラスに分かれる。クラス j ($j=1, 2$) のジョブは、CPUにおいて平均 μ_j^{-1} 、I/Oにおいて平均 λ_j^{-1} の指数サービス時間分布を持つとする。他に仮定しない限り $\mu_1 > \mu_2$ としよう。 λ/μ_j ($j=1, 2$) を ρ_j とあらわす。仮想記憶システムの場合には、ページング装置もI/Oの一つと考え、ページフォールト回数はジョブの性質だけに依存するとする（つまり、ページング方式として、ワーキングセット方式[4]のようなもの考える）。従って、 μ_j ($j=1, 2$) は定数パラメータと考えられる。クラス j ($j=1, 2$) のジョブが要求する総CPU時間は、平均 s_j^{-1} の指数分布であるとする。

各ジョブが要求する主メモリ量は一定であるとする；すなわち、多重度 (N で表わす) は、ジョブミックスに入るので待つジョブがある限り一定である。

ジョブストリーム上のジョブの系列は、次に述べるように、有限マルコフ連鎖の一見本過程とみなされる：ジョブストリーム上の各位置に、クラス j ($j=1, 2$) のジョブが出現する確率を、 p_j としよう。すなわち、1つのジョブが処理を完了しジョブミックスを出ていくと、クラス j のジョブがジョブミックスに入ると確率は、過去にどのような順序でジョブが出ていったかによらず、 p_j に等しい。 $p_1 + p_2 = 1$ となる。 $1 - p_j$ ($j=1, 2$) を q_j とあらわす。

次の仮定は、モデル全体が、2段階 *nearly completely decomposable* になるためのものである。 $q_1 s_1$, $q_2 s_2$ よりも十分大きく、同時に、 $\mu_2/N!$, $(\lambda/N!)(\mu_2/\lambda)^N$ よりも十分小さい値 F が存在すると仮定する。つまり、

$$\max [q_1 s_1, q_2 s_2] \ll F \quad \text{および} \\ (\mu_2/N!) \min [1, (\mu_2/\lambda)^{N-1}] \gg F.$$

これは、分解可能のための十分条件に過ぎないことに注意されたい。

2.2 多重プログラミングシステムのモデル

上で定義したベンチマークジョブストリームを処理するコンピュータシステムのモデルをここで述べる。そのコンピュータシステムは、ジョブを N 個収容する主メモリと、CPUとI/Oとの2つのサービスセンターからなる：CPUにおいては、ジョブは待たされることがあるが、I/Oにおいては、台数が十分あり待ちが無視できるとする。従って、finite-source queueing model を多重プログラミングシステムのモデルとして用いることができる。

現在および過去の情報のみに基づいてスケジューリングの決定をする方式が、CPUにおいて使われると仮定する。従って、残余サービス時間の正確な情報に基づくスケジューリング方式（これは現実には実現不可能である）は考慮しない。含まれるのは、FCFS, preemptive および nonpreemptive priority, shortest および longest expected remaining processing time first, preemptive および nonpreemptive LCFS, processor sharing, generalized

processor sharing, 等である。[より正確には, 採用されるスケジューリング方式は, 次のような特徴を持つものである: 同一ジョブの集合 (T) が, CPU にいる状態を, すべてまとめて1つの状態 (S_T とあらわす) としよう。状態 S_T の確率を P_T とあらわす ($\sum_{j \in T} P_T = P_T$)。次のような条件を満たすスケジューリング方式のみを考えよ: 任意の状態 S_T からの, ジョブ j が CPU に入はれることによる, 状態遷移の割合は, $\mu_j P_T$ である ($j \in T$); 任意の状態 S_T からの, ジョブ k ($k \notin T$) が I/O に入はれることによる, 状態遷移の割合は, λP_T である。] オペレーティングシステムによる, スケジューリングやジョブミックス遷移の際のオーバーヘッドは, 無視できるとする。

クラス1のジョブが i 個, クラス2のジョブが $N-i$ 個あるジョブミックスは, ジョブミックス $(i, N-i)$ と呼ぶ。ジョブミックスには, $(0, N), (1, N-1), \dots, (N, 0)$ の $N+1$ 種類あることになる。各ジョブは, ジョブストリームの中のジョブ系列にある順に, ジョブミックスに入るとする; そして, ジョブミックスに入ると, そのジョブの全ての計算が終了まで, とどまるものとする。

3. 解析

任意の時間区間 $t \sim t + \Delta t$ を考えよう; t はジョブストリーム処理中の任意の時点である; (i) Δt は非常に小さく, この時間区間内に, ジョブミックス遷移が2回以上おこる確率は無視できるものとする; 同様に, (ii) Δt は十分大きく, もしこの時間区間内にジョブミックス遷移がおこらないとすると, クラス j ($j=1, 2$) のCPU利用率は, 統計的平衡状態の値に, 十分な精度で, 等しいとみなすことができるとする。つまり, モデルは, 2段階 nearly completely decomposable であり, その下の段階は, ジョブミックス処理段階で, finite-source queueing model を用いて解析され, 上の段階は, ジョブミックス遷移段階で, 平衡状態のマルコフモデルを用いて解析される。

まず, 3.1節において, 上述のような Δt があると仮定して, ジョブミックス処理段階の諸性質を求める。次に, 3.2節において, 上述のような Δt が存在し, モデルが nearly completely decomposable であることを示す。最後に, 3.3節において, ジョブミックス遷移段階の性質を求め, H-scheduleの最適性を示す。

3.1 ジョブミックス処理段階における性質

任意のジョブミックス $(i, N-i)$ を考えよう。このときのCPUの利用率を $\rho^{(i)}$ とあらわそう。このとき, クラス j ($j=1, 2$) のジョブに対するCPU利用率を $\rho_j^{(i)}$ とあらわそう ($\rho_1^{(i)} + \rho_2^{(i)} = \rho^{(i)}$)。2.2節における仮定により, finite-source model をこの段階の解析に用いることができるので, 次のような3つの性質が得られる[5]。

A1. $\rho^{(i)}$ は, CPU スケジューリング方式によらず一定で, 次のように求められる.

$$\rho^{(i)} = 1 - \left[\sum_{n_1=0}^i \sum_{n_2=0}^{N-i} (n_1 + n_2)! \binom{i}{n_1} \binom{N-i}{n_2} r_1^{n_1} r_2^{n_2} \right]^{-1} \quad (3.1)$$

ただし r_j ($j=1, 2$) は λ/μ_j に等しいと定義した.

A2. $\rho_j^{(i)}$ ($j=1, 2$) が最大であるための必要十分条件は, クラス j のジョブは他のジョブよりも高い preemptive priority が与えられることである. このとき $\rho_j^{(i)}$ は, クラス内の優先順位付けには依らない.

A3. ジョブミックスが $(i, N-i)$ のときの, CPU と I/O の利用率の総和を $U^{(i)}$ とあらわそう. $U^{(i)}$ が最大になるための必要十分条件は, CPU において H-schedule が用いられることである.

$U^{(i)} = \sum_{j=1}^2 (1 + r_j^{-1}) \rho_j^{(i)}$ となることに注意されたい. 性質 A1 は, 1 節で述べた性質 I と同じで, 性質 A3 は, 性質 II と同じである. これらの性質の導出については, 文献 [5] を参照されたい.

性質 A1 より, 次の性質が導き出される.

A4. $\mu_1 \geq \mu_2$ に対し, $\rho^{(i)} \geq \rho^{(i+1)}$ ($i=0, 1, \dots, N-1$).

(証明) $\binom{N-i}{n_2} = \binom{N-i-1}{n_2} + \binom{N-i-1}{n_2-1}$. これより,

$$\begin{aligned} & \sum_{n_2=0}^{N-i} (n_1 + n_2)! \binom{N-i}{n_2} r_2^{n_2} \\ &= n_1! + \sum_{n_2=1}^{N-i-1} (n_1 + n_2)! \binom{N-i-1}{n_2} r_2^{n_2} \\ & \quad + \sum_{n_2=1}^{N-i-1} (n_1 + n_2)! \binom{N-i-1}{n_2-1} r_2^{n_2} + (n_1 + N-i)! r_2^{N-i} \\ &= \sum_{n_2=0}^{N-i-1} (n_1 + n_2)! \binom{N-i-1}{n_2} r_2^{n_2} \\ & \quad + \sum_{n_2'=0}^{N-i-1} (n_1 + n_2' + 1)! \binom{N-i-1}{n_2'} r_2^{n_2'+1} \\ &= \sum_{n_2=0}^{N-i-1} \sum_{n_3=0}^1 (n_1 + n_2 + n_3)! \binom{N-i-1}{n_2} r_2^{n_2+n_3}. \end{aligned}$$

従って, (3.1) 式より

$$(1 - \rho^{(i)})^{-1} = \sum_{n_1=0}^i \sum_{n_2=0}^{N-i-1} \sum_{n_3=0}^1 (n_1 + n_2 + n_3)! \binom{i}{n_1} \binom{N-i-1}{n_2} r_1^{n_1} r_2^{n_2+n_3}.$$

同様に

$$(1 - \rho^{(i+1)})^{-1} = \sum_{n_1=0}^i \sum_{n_2=0}^{N-i-1} \sum_{n_3=0}^1 (n_1 + n_2 + n_3)! \binom{i}{n_1} \binom{N-i-1}{n_2} r_1^{n_1+n_3} r_2^{n_2}.$$

従って

$$(1 - \rho^{(i)})^{-1} - (1 - \rho^{(i+1)})^{-1} \\ = \sum_{n_1=0}^i \sum_{n_2=0}^{N-i-1} (n_1 + n_2 + 1)! \binom{i}{n_1} \binom{N-i-1}{n_2} r_1^{n_1} r_2^{n_2} (r_2 - r_1).$$

$r_j = \lambda / \mu_j$ ($j=1, 2$) に注意して, 上から

$$\mu_1 \geq \mu_2 \quad \text{に対して} \quad \rho^{(i)} \geq \rho^{(i+1)}.$$

証明終

性質 A4 により, ジョブミックス中の CPU-bound ジョブの数が多いほど, そのジョブミックスによる CPU 利用率が高くなるという, 1 節で述べた直観がこのモデルにおいて正しいことがわかった。

3.2 モデルの near complete decomposability

種々の確率過程における, 状態遷移行列に基づいた, near complete decomposability の詳細な議論が積み重ねられてきた [3, 10]. 本論文では, 状態遷移行列を用いず, より直接的, 直観的に取扱いを行なう。

CPU は, idle であり続ける idle period と, busy であり続ける busy period のサイクルの繰り返しを経由する. idle period 1 つと busy period 1 つとから成る 1 つのサイクルを考えよう. そのサイクルにおける, クラス j ($j=1, 2$) のジョブに対する, CPU の利用率は, CPU がクラス j のジョブを処理していた時間長を, サイクルの時間長で割ったものである。

ジョブミックスが $(i, N-i)$ のとき, 1 サイクルの平均時間長 ($t_c^{(i)}$ であらわす) は, idle period が $(N\lambda)^{-1}$ であるので,

$$t_c^{(i)} = [N\lambda(1 - \rho^{(i)})]^{-1}.$$

$t_c^{(i)}$ は, 性質 A4 より $\rho^{(0)} > \rho^{(i)}$ $i \neq 0$ であるので, ジョブミックス $(0, N)$ のとき最大長 (これを t_c であらわす) となり,

$$t_c = [N\lambda(1 - \rho^{(0)})]^{-1}.$$

従って, もし $\Delta t \gg t_c$ ならば, 区間 $t \sim t + \Delta t$ の間に生ずるサイクル数は, 非常に大きくなることがわかる. 各サイクルの開始点は, regeneration point であることに注意されたい. 従って, 中心極限定理より, この時間区間についての, クラス j ($j=1, 2$) のジョブによる CPU 利用率は, ジョブミックスが変わらなければ, 定常状態の値に, 十分な精度で, 等しいということができた. 従って, もし $\Delta t \gg t_c$ ならば, Δt が 3 節のはじめに述べた性質 (j) を持つことがわかる。

さらに, $q_j s_j \rho_j^{(i)} < q_j s_j \rho^{(i)} \leq q_j s_j \rho^{(0)}$ $j=1, 2, i=0, 1, \dots, N$ であることに注意された。従って, もし $q_j s_j \rho^{(0)} \Delta t \ll 1$ ならば, クラス j のジョブの

全CPU時間分布が、平均 s_j^{-1} の指数分布であるので、クラス j のジョブがその区間内に2つ以上減る確率は無視できる。つまり、その区間内に、ジョブミックス遷移が2回以上おこる確率は無視できる。これから、もし $q_j s_j p^{(0)} \Delta t \ll 1$, $j=1, 2$ ならば、 Δt が、3節のはじめに述べた性質(ii)を持つことがわかる。

ここで、次のような Δt が存在することを示そう。

$$t_c \ll \Delta t \ll \{\max [q_1 s_1, q_2 s_2] p^{(0)}\}^{-1}.$$

性質 A1 より、

$$p^{(0)} = 1 - \left\{ \sum_{n=0}^N [N! / (N-n)!] r_2^n \right\}^{-1}.$$

従って

$$\begin{aligned} t_c &= [N\lambda(1-p^{(0)})]^{-1} = (N\lambda)^{-1} \left\{ \sum_{n=1}^N [N! / (N-n)!] r_2^n \right\} / p^{(0)} \\ &\leq \mu_2^{-1} N! \max [1, (\lambda/\mu_2)^{N-1}] / p^{(0)}. \end{aligned}$$

2.1 節の終りの仮定より、次のような F が存在する。

$$\max [q_1 s_1, q_2 s_2] \ll F \ll (\mu_2/N!) \min [1, (\mu_2/\lambda)^{N-1}].$$

従って、 $\Delta t = [p^{(0)} F]^{-1}$ とおくと、

$$\Delta t \gg \mu_2^{-1} N! \max [1, (\lambda/\mu_2)^{N-1}] / p^{(0)} \geq t_c$$

$$\Delta t \ll \{\max [q_1 s_1, q_2 s_2] p^{(0)}\}^{-1}$$

これより、near complete decomposability が証明された。

3.3 ジョブミックス遷移段階の性質と、最適CPUスケジューリング方式

ジョブストリーム全体を処理した後のCPU利用率を ρ であらわそう。モデルの仮定の下で、 ρ が最大になるためには、CPUにおいてH-scheduleを用いることが必要十分であることを示す。3.2節の議論より、時間区間 $t \sim t + \Delta t$ の間において、ジョブミックス $(i, N-i)$ が変わらないとすると、クラス j ($j=1, 2$) のジョブにより用いられたCPU時間長は、十分に精度で、 $\rho_j^{(i)} \Delta t$ に等しいといえる。更に、3.2節において、 $q_j s_j \rho_j^{(i)} \Delta t \ll 1$, $j=1, 2$, $i=0, 1, \dots, N$ となり得ることをみた。

とて、 $q_j s_j \rho_j^{(i)}$ ($j=1, 2$, $i=0, 1, \dots, N$) の中の最大のものと、同程度の大きさの量としよう ($\varepsilon \ll 1$)。クラス j のジョブの全CPU時間が、平均 μ_j^{-1} の指数分布であるので、クラス j のジョブの数が減る確率は、 $q_j s_j \rho_j^{(i)} \Delta t + O(\varepsilon)$ となる。ただし、 $O(\varepsilon)$ は ε と同程度の量をあらわす。クラス j のジョブが2個以上減る確率は、 $O(\varepsilon)$ となり無視できる。

$P_j(t)$ が、時刻 t におけるジョブミックスが $(i, N-i)$ である確率をあらわすしよう。そうすると、

$$\begin{aligned}
P_i(t+\Delta t) &= [1 - (q_1 s_1 \rho_1^{(i)} + q_2 s_2 \rho_2^{(i)}) \Delta t + O(\epsilon)] P_i(t) \\
&\quad + (q_1 s_1 \rho_1^{(i+1)} \Delta t + O(\epsilon)) P_{i+1}(t) \\
&\quad + (q_2 s_2 \rho_2^{(i-1)} \Delta t + O(\epsilon)) P_{i-1}(t) + O(\epsilon) \\
&\quad i=1, 2, \dots, N-1, \\
P_N(t+\Delta t) &= (1 - q_1 s_1 \rho_1^{(N)} \Delta t + O(\epsilon)) P_N(t) \\
&\quad + (q_2 s_2 \rho_2^{(N-1)} \Delta t + O(\epsilon)) P_{N-1}(t) + O(\epsilon), \\
P_0(t+\Delta t) &= (1 - q_2 s_2 \rho_2^{(0)} \Delta t + O(\epsilon)) P_0(t) \\
&\quad + (q_1 s_1 \rho_1^{(1)} \Delta t + O(\epsilon)) P_1(t) + O(\epsilon).
\end{aligned}$$

モデルは 統計的平衡にあると考えられるので,

$$P_i(t+\Delta t) = P_i(t) = P_i \quad i=0, 1, \dots, N.$$

さらに, $O(\epsilon)$ は無視できるので, 上から, 次の方程式群が得られる.

$$\begin{aligned}
q_1 s_1 \rho_1^{(i+1)} P_{i+1} - q_2 s_2 \rho_2^{(i)} P_i &= q_1 s_1 \rho_1^{(i)} P_i - q_2 s_2 \rho_2^{(i-1)} P_{i-1} \\
&\quad i=1, 2, \dots, N-1, \\
q_1 s_1 \rho_1^{(N)} P_N - q_2 s_2 \rho_2^{(N-1)} P_{N-1} &= 0, \\
q_1 s_1 \rho_1^{(1)} P_1 - q_2 s_2 \rho_2^{(0)} P_0 &= 0.
\end{aligned}$$

これから,

$$P_i / P_{i-1} = (q_2 s_2 \rho_2^{(i)}) / (q_1 s_1 \rho_1^{(i-1)}) \quad i=1, 2, \dots, N. \quad (3.1)$$

α_i ($i=1, 2, \dots, N$) が $(q_2 s_2 \rho_2^{(i)}) / (q_1 s_1 \rho_1^{(i-1)})$ とおくとしよう.
 $\alpha_0 = 1$ とする. 従って, $0 < \alpha_i < \infty$, $i=0, 1, \dots, N$. 3.1節の性質 A2 より, α_i が最小になるためには, ジョブミックスが $(i, N-i)$ および $(i-1, N-i+1)$ のとき, クラス1のジョブがクラス2のジョブよりも高い pre-emptive priority を持つことが必要十分である. 従って

$$P_i = P_0 \prod_{k=0}^i \alpha_k \quad i=0, 1, \dots, N \quad (3.2)$$

$$\text{したがって } P_0 = \left(\sum_{k=0}^N \prod_{k=0}^i \alpha_k \right)^{-1}.$$

全体的なCPU利用率は,

$$\rho = \sum_{i=0}^N \rho^{(i)} P_i$$

$\rho^{(i-1)} - \rho^{(i)}$ を d_i とおく. 性質 A4 と $\mu_1 > \mu_2$ から $\rho^{(i-1)} > \rho^{(i)}$ であるので, $d_i > 0$ $i=1, 2, \dots, N$, である. 従って

$$\rho = \sum_{i=0}^N d_i \sum_{j=0}^i P_j. \quad (3.3)$$

$\sum_{j=0}^i P_j$ は, ジョブミックスが $N-i$ 個以上のCPU-boundジョブを含む確率であることを注意したい. (3.2) より,

$$\sum_{j=0}^i P_j = \left(\sum_{j=0}^i \prod_{k=0}^j \alpha_k \right) / \left(\sum_{k=0}^N \prod_{k=0}^i \alpha_k \right). \quad (3.4)$$

従って,

$$\sum_{j=0}^i P_j = \begin{cases} (\sum_{k=0}^i \pi_{k=0}^i \alpha_k) / (\sum_{k=0}^{m-1} \pi_{k=0}^j \alpha_k + \alpha_m \pi_{j=m}^N \pi_{k=0}^j \alpha_k) & m > i \\ 1 - (\alpha_m \sum_{j=i+1}^N \pi_{k=0}^j \alpha_k) / (\sum_{k=0}^{m-1} \pi_{k=0}^j \alpha_k + \alpha_m \sum_{j=m}^N \pi_{k=0}^j \alpha_k) & m \leq i \end{cases} \quad (3.5)$$

ここで、一時的に、 $\sum_{j=0}^i P_j$ が、独立変数 $\alpha_1, \alpha_2, \dots, \alpha_N$ の関数であるとみなそう。そうすると、(3.5) より、他の α_k ($k \neq m$) が固定されたとすると、 $\sum_{j=0}^i P_j$ が最大になるには、 α_m が最小になるのが、必要十分である。従って、 $\sum_{j=0}^i P_j$, $i=1, 2, \dots, N$ のすべてが最大になるには、 α_j $j=1, 2, \dots, N$ のすべてが最小になるのが、必要十分である。従って、(3.3) 式より、CPU 利用率 ρ が最大になるためには、CPU で H-schedule を用いることが、必要十分であることがわかる。また、ジョブストリーム中のすべてのジョブが必要とする総 CPU 時間量は、スケジューリングによらず一定であると考えられるから、H-schedule が使われると、ジョブストリーム処理完了までの総経過時間が最小になる（スループットが最大になる）ことがわかる。

4. 結論

ベンチマークジョブストリームを処理する多重プログラミングシステムのマルチコフモデルを解析した；ジョブストリームは、I/O-bound ジョブと CPU-bound ジョブとから成る。このモデルにおいて、I/O-bound ジョブに、CPU-bound ジョブよりも高い preemptive priority を与える CPU スケジューリング方式（H-schedule）が、ベンチマークジョブストリームを処理する際に、CPU 利用率を最大にする（最大のスループットをもたす）ことが示された。

本論文で示したのと類似の議論が、より一般的な場合に近似的に適用できる可能性があると考えられる；つまり、H-schedule により、一般に I/O-bound ジョブがジョブミックスに滞在する時間が、最小になる；このため、ジョブミックスは、平均して、最も CPU-bound よりになる；同じジョブミックスの CPU 利用率は、CPU スケジューリング方式に余りよらないと考えられる；従って、全体的 CPU 利用率（そしてスループット）は、H-schedule が用いられると、改善されることが考えられるのである。

5. 謝辞

本研究で取扱った問題の始原は、日立製作所、大野一彦氏、本山博司氏、等の人々との議論に刺激されて出てきたものである。ここに記して謝意を表す。

参考文献

1. Blevins, P. R., and Ramamoorthy, C. V. Aspects of a dynamically adaptive operating system. IEEE Trans. on Computers C-25, 7(July 1976), 713-725.
2. Boyse, J. H., and Warn, D. R. A straightforward model for computer performance prediction. ACM Computing Surveys 7, 2(1975), 73-93.
3. Courtois, P. J. Decomposability: Queueing and Computer System Applications. Academic Press, N. Y. (1977).
4. Denning, P. J. The working set model of program behavior. Comm. ACM 11, 5(May 1968), 323-333.
5. Kameda, H. A finite-source queue with different customers. J. ACM (to appear).
6. Marshall, B. S. Dynamic calculation of dispatching priorities under OS/360 MVT. Datamation 15 (August 1969), 93-97.
7. Price, T. G. A note on the effect of the central processor service time distribution on processor utilization in multiprogrammed computer systems. J. ACM 23, 2(April 1976), 342-346.
8. Ryder, K. D. A heuristic approach to task dispatching. IBM Systems J. 9, 3(1970), 189-198.
9. Sherman, S., Baskett, F., and Browne, J. C. Trance-driven modeling and analysis of CPU scheduling in a multiprogramming system. Comm. ACM 15, 12(December 1972), 1063-1069.
10. Simon, H. A., and Ando, A. Aggregation of variables in dynamical systems. Econometrica 29 (April 1961), 111-138.
11. Stevens, D. F. On overcoming high priority paralysis in multiprogramming systems: a case history. Comm. ACM 11, 8(August 1968), 539-541.
12. Strauss, J. C. An analytic model of the Hasp Execution Task Monitor. Comm. ACM 17, 12(December 1974), 679-685.
13. ———, Dynamic dispatching in job class scheduled systems. Proc. AFIPS 1975 NCC, Vol.44, AFIPS Press, Montvale N. J., (1975), 343-350.
14. Wulf, W. A. Performance monitors for multi-programming systems. Proc. 2nd Symposium on Operating System Principles, ACM, New York, (October 1969), 175-185.