計算機システムのシミュレーションについて*

金沢正憲** 北川一*** 萩原宏**

Abstract

The intention of this paper is to describe the use of simulation method in the analysis of a multiprogramming computer system. In this simulation the model and input data are given on the basis of the measurements on an actual system. We describe a few techniques, taking the efficiency of simulation into consideration. The results of the simulation and the effectiveness of the presented techniques are shown.

1. まえがき

計算機システムのシミュレーションは，システム・パフォーマンスの分析および評価を行なうための有力な1つの方法である．一般に，計算機システムは，多くの確率変数やパラメータ（X）によってその動作環境が与えられる複雑な待ち行列プロセスであり，その動作状況が種々のパフォーマンス測度（P）として定まるような $P=F(X)$ の関係と考えることができる．システム・パフォーマンスの解析のためのシミュレーションは，モデル化された計算機システムの動作をシミュレートして，入力 X と出力 P の関係を求めることになる．しかし，その場合に問題となるのは，シミュレーションの精確さとそれに要する計算時間などのコストである[1]．したがって，必要な精度を保ち，しかも計算時間を短くするという工夫が必要になる．

実システムをモデル化する段階においては，システム・パフォーマンスの隘路となる部分がモデルに十分反映されていること，一方，モデル作成時の仮定を，求めようとする結果にできるだけ影響の少ない範囲内で設定することが大切である．このためには，シミュレーションの前段階として，実システムの測定による分析，シミュレーションの解析結果からのフィード・バックなどが必要となろう．また，シミュレーションと実システムの差異は，パラメータの一部である入力データと実際のデータの違いからも生じる．シミュレーションに限らず，有効な解析法が得られたとしても，実用的な解析を行なうためには，種々の確率変数に関して実測されたデータの利用を考えねばならない．

つぎに，処理技法上の問題として，まず，1回のシミュレーション・ランにおける計算時間と得られる結果の精度（あるいは揺ぎ）であるが，これは，シミュレーションの初期条件および収束判定による打切り条件により，時間と精度をバランスさせることにある．一方，計算機システムのシミュレーションでは，パラメータの種々の組合せと結果の関係，または最適な結果を与えるパラメータの組合せなどを知ろうとする場合が多い．このような時は，通常，多大の計算時間を必要とするので，シミュレーションの効率として，全シミュレーション・ランに要する時間の短縮を考えることも有効となる．

われわれは，シミュレーションの効率の向上に役立つと思われる 2，3 の手法について検討し，それを取入れて，マルチプログラミング・システムのシミュレーションを行なった．その対象として，京都大学大型計算機センターの FACOM 230-60 システム（2CPU，コア・メモリ 192 k 語，マルチプログラミングによるバッチ処理，ユーザのプログラミング言語はほとんどが FOTRAN）を取上げ，システムの動作状況を測定し，そのデータに基づいて，オペレーティング・システム（MONITOR-V システム[5]）をモデル化し，シミュレーションによってシステム・パフォーマンスの評価を行なった．

---

* Simulation of a Computer System, by Masanori Kanazawa (Faculty of Engineering, Kyoto University), Hajime Kitagawa (Data Processing Center, Kyoto University) and Hiroshi Hagiwara (Faculty of Engineering, Kyoto University).

** 京都大学工学部

*** 京都大学大型計算機センター

2. システムの分析

システムの動作状況を知るために，実システムで磁気テープに収録されるアカウント情報（ジョブ・ステップ毎の各リソースの使用量など）のなかの CPU 専有時間 (T_{CPU})，コア・メモリ専有時間 (T_{CORE})，同専有語数，ラインプリンタ出力行数 (L) などを利用した．

多重度1でジョブ処理（平常時に，このシステムで処理されるものと同等の FORTRAN ジョブ）を行ない，各ジョブ・ステップ*に関して，次のような量の統計を取り，その分布を求めてシミュレーションの入力データとした．

（1） 入出力時間 ($T_{I/O}$)

このマルチプログラミング・システムでは，カード入力やラインプリンタ出力は，ファイル装置（ドラム，ディスク・パック）にスタック（システム入出力）されるので，実行中のジョブからの入出力動作はすべてファイル装置との間で行なわれる．多重度1の時，ファイル装置と CPU の使用のための待ち時間は0とみなせるから，次式が成立つ．

$$T_{I/O}=T_{CORE}-T_{CPU}-T_P-T_F. \qquad (1)$$

ただし，T_P：プログラム・ロード時間

T_F：ファイルのオープン，クローズ時間．

上式によりアカウント情報に収集されていないジョブ・ステップ毎の入出力時間の分布が求められる．入出力時間は一般に，そのジョブ・ステップでの処理プログラムの入出力要求の回数，入出力量などからなる複雑さに依存する．この複雑さを示すものとして，コンパイル，結合編集ステップでは CPU 専有時間を，実行ステップではラインプリンタ出力行数を取上げ，それと入出力時間との関係を一次式 ($T_{I/O}=ax+b$，$x=T_{CPU}$ または L) で表わしたときの傾き (a) の分布を **Fig. 1** に示す．なお，$b=T_P+T_F$ とし，この値の測定に関しては，簡単で短い FORTRAN ジョブ (STOP, END の2つのステートメントからなるジョブ) をいくつかランさせることによって，平均値を求め，それを用いた．(**Table 1** 参照.)

（2） スケジュラ**時間

多重度1の場合，スケジュラどおしの待ちおよびスケジュラ内でのリソース使用のための待ちは起こらないと考えられるので，各ジョブおよびジョブ・ステップにおけるスケジュラの開始時刻と終了時刻をコンソール・タイプライタに打出し，スケジュラに要する時間の分布を求めた．これを **Fig. 2** に示す．

* FORTRAN ジョブでは，コンパイル，結合編集，実行の最大3つのジョブ・ステップに分けられる．

** 制御プログラムの一部であるジョブおよびジョブ・ステップのイニシエータ/ターミネータ．

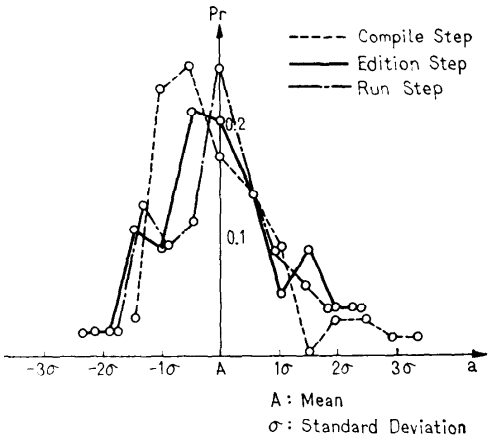

Fig. 1 Probability Density of 'a'

Table 1 CPU/CORE Time of STOP-END Job.

	CPU Time (sec)	Core Time (sec)
Compile Step	0.15	7.75
Edition Step	2.60	13.40
Run Step	0.03	2.30

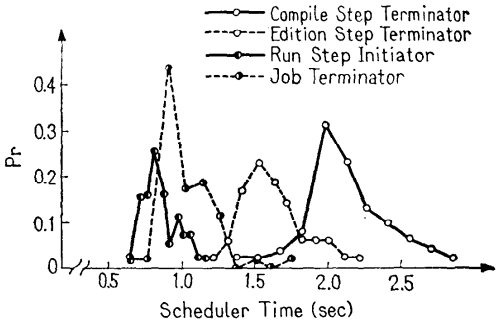

Fig. 1 Probability Density of Scheduler Time

3. シミュレーション・モデル

次のような仮定のもとに，マルチプログラミング処理の効率の大きな因子となっているスケジュラの走行

とブロック化された入出力動作に重点を置いてシミュレーション・モデルを作成した.

(i) 処理プログラムにおける入出力の間の連続したCPU専有時間は，そのジョブ・ステップで一定である.

(ii) 制御プログラムが走行する時のCPUの使用時間，ならびにタスク・スイッチに伴なうオーバーヘッド・タイムは考慮しない.

(iii) 異なる処理プログラムからの入出力が，たとえ同じファイル装置を使用していても，同時にチャネルの台数まで可能である.

(iv) 低速入出力装置とファイル装置間のシステム入出力は考慮しない.

モデル化されたシステムの構成は，**Fig. 3** に示されているように，CPU 2台，コア・メモリ（ユーザ・プログラム領域 Mk 語），チャネル3台（入出力用2台，プログラム・ロード用1台）となっている.

ジョブは，ジョブ・イニシエータにより開始され，ジョブ・ステップ・イニシエータ，処理プログラム(ジョブ・ステップ)，ジョブ・ステップ・ターミネータからなるサイクルをジョブ・ステップの回数だけ繰返した後，ジョブ・ターミネータにより終了する．このようなジョブの流れが設定された多重度の数だけでき

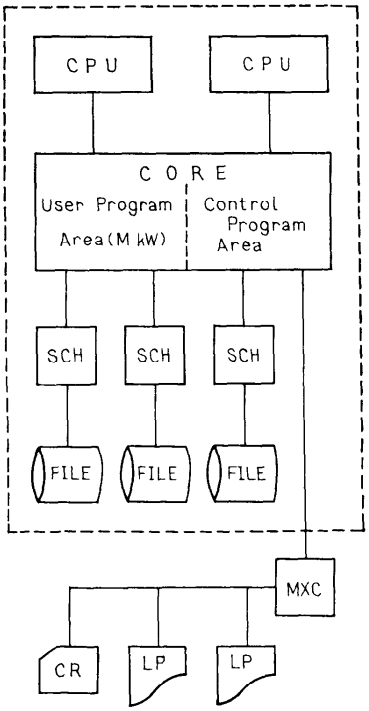

Fig. 3 System Configuration

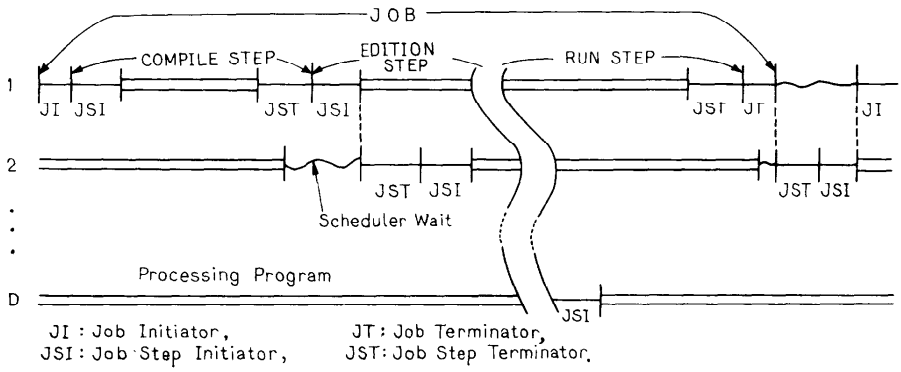

Fig. 4 Flow of D Jobs Processed in Multiprogramming Environment

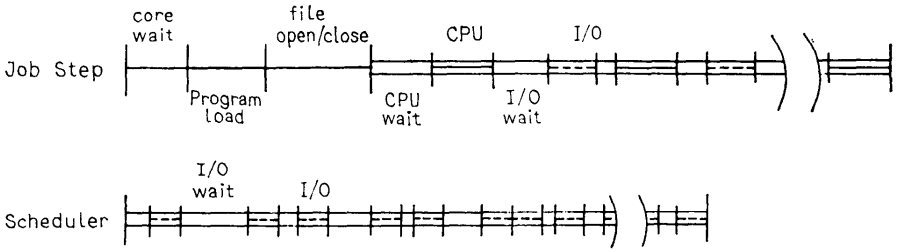

Fig. 5 Structure of Job Step and Scheduler Time

る．スケジュラは同時に2つ以上のジョブにサービスを行なうことはなく，ジョブ・ステップ・ターミネータに続くジョブ・ステップ・イニシエータおよびジョブ・ターミネータは他のスケジュラよりも優先権が高い．多重度Dで処理されるジョブの流れを **Fig. 4** に示す．

実際のシステムにおける各スケジュラの走行は，入出力動作が大部分を占め，CPU の使用時間は短いので，スケジュラは何回かの入出力動作のみからなり，1回の入出力量は一定とした*．(**Fig. 5** 参照．)

一方，ジョブ・ステップの処理プログラム部分の構成は **Fig. 5** に示されている．最初に，処理プログラムが必要とするコア・メモリの領域を確保し**，処理プログラムをロードし，必要なファイルのオープンを行ない，処理プログラムの実質的な実行が開始される．何回かの CPU の専有，入出力動作の繰返し後，ファイルがクローズされて，そのジョブ・ステップの実行は終了する．モデルでは，ファイルのクローズに要する時間は，オープンの時間とまとめてオープンのところにとった．処理プログラムの1回の入出力量も一定と考え，さらに前述（i）の仮定のもとに，CPU 専有時間は入出力回数*** によって等分割し，1回の CPU 専有時間とした．

4. シミュレータの構成と入力データ

シミュレータは，**Fig. 6** に示すように，3つの部分，すなわち，戦略プログラム，シミュレーション・プログラム，監視プログラムからなっている．戦略プログラムは，監視プログラムへ収束判定，区間推定などのために必要な値を与えるとともに，シミュレーション・プログラムのパラメータの値を決定するルーチンである．監視プログラムは，シミュレーション・プログラムからの結果により，区間推定，収束判定を行ない，シミュレーションの続行・打切りを決定するルーチンである．シミュレーション・プログラムは，さらに，タスク管理，スケジュラ管理，多重度管理などのシステムをシミュレートしているルーチンと，パフォーマンス測度に関する情報の採集を行なうパフォーマンス・ルーチンからなっている．

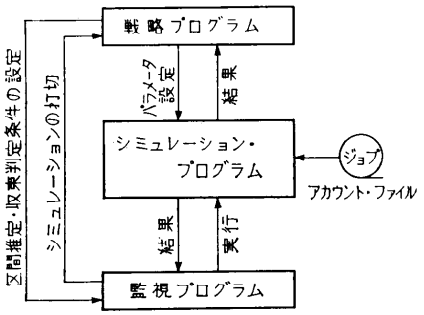

Fig. 6 Simulator

入力データのうち，ジョブの各ジョブ・ステップにおける入出力時間，スケジュラ時間は，システムの分析の結果に従って，近似式の傾き，時間の分布を正規分布で近似し，乱数を発生させて求めた．一方，CPU 専有時間，コア・メモリ専有語数，ライン・プリンタ出力行数については通常のジョブ処理時に収録されたアカウント情報の一日分のデータをそのまま用いて，システムへの実際の負荷状態を再現した．

5. システム・パフォーマンスの評価

システム・パフォーマンスを評価する上で重要と考えられるパラメータとして，CPU の速度比*，コア・メモリのユーザ・プログラム領域，設定多重度を選び**，一方，パフォーマンス測度として，各リソースの使用率，スケジュラ走行率***，実効多重度などを取上げた．

また，システム全体のパフォーマンス/コストを評価する量として，

$$P_c=\frac{T}{\sum_i C_i}\times\frac{\sum_i C_iU_i}{\sum_i C_i} \qquad (2)$$

を考えた．T は単位時間当りのジョブ・ステップ処理件数，C_i はリソース i (i=CPU, SCH, CORE) の価格，U_i はリソース i の1台当りの使用率である．CPU の速度比 r による価格の比率は Grosch の法則に従うものとした．各リソースの台数を N_i とすると，

$$\sum_i C_i=\frac{1}{\sqrt{r}}C_{CPU}N_{CPU}+C_{SCH}N_{SCH}+C_{CORE}N_{CORE} \qquad (3)$$

* ブロック化された入出力を想定し，1回の入出力量およびアクセス時間を一定とした．2.（2）で与えられた分布に従うスケジュラ時間をこの単位入出力時間で割って入出力回数を求める．

** コア・メモリ専有量は，そのジョブ・ステップでは一定であり，また，空き領域は常に連続になっているとする．

*** 処理プログラムのタスクの切換えは，自から入出力要求を発した時にのみ行なわれるとし，2.（1）で与えられた分布に従うジョブ・ステップ入出力時間を単位入出力時間で割って回数を求める．

* 実際のシステムを1とし，CPU 速度比が r であるというのは，CPU による処理時間が r 倍になることである．

** 他のパラメータは，実システムを想定し，スケジュラ，処理プログラムでのそれぞれの単位入出力量を 256, 512 語，アクセス時間を 17 ミリ秒，転送速度を 39 語/ミリ秒とした．

*** スケジュラ時間の総和をシステム稼動時間で割ったもの．

となる.

6. シミュレーションの効率と方法

(1) 収束判定と打切り

平均的状態を求めるシミュレーションの場合，所要の精度を得るために必要十分な試行回数を決めることが問題となる.

シミュレーション $\boldsymbol{P}=\boldsymbol{F}(\boldsymbol{x})$ において，入力乱数を示すパラメータを $\boldsymbol{y}$ とすれば，シミュレーションは，

$$\boldsymbol{P}=\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}_i), \qquad (4)$$

ただし，$\boldsymbol{x}$ はパラメータ $\boldsymbol{X}$ から入力乱数のパラメータ $\boldsymbol{y}$ を除いた部分

と表わせる．すなわち，$P_j{}'$ を第 i 回目の試行による P_j の値とすれば，$P_j{}'$ はパラメータ $\boldsymbol{x}$ で試行した時の入力乱数 $\boldsymbol{y}_i$ によって生じる揺ぎを含んでいる．この揺ぎをいかに判定するかということが問題となる.

それには，つぎのような区間推定法を利用することができる．すなわち，正規母集団 $\boldsymbol{N}(\mu,\sigma^2)$ からの標本 $(x_1, x_2, \cdots, x_n)$ による平均値の最尤推定量の信頼区間幅は，危険率 ε のもとで，

$$2t_\phi(\varepsilon/2)S/\sqrt{n-1} \qquad (5)$$

となる．ただし，$t_\phi(\varepsilon/2)$ は自由度 $\phi=n-1$ の t 分布の上側 $(\varepsilon/2)$ 点で，S は分散の最尤推定量の平方根(正)である.

そこで，シミュレーションによって得られる系列 $\{P_j{}'\}$ から，中心極限定理に基づいて，近似的に正規分布に従う系列 $\{\bar{p}_k\}$ を作り，この標本 $(\bar{p}_1, \bar{p}_2, \cdots, \bar{p}_n)$ によって上のような区間推定を行なうことができる．このような方法により，前もって信頼区間幅を決定しておけば，余分の試行を行なうことなどに必要な精度の結果を求めることができる.

(2) 初期設定とパラメータの変更

平均的状態を求めるシミュレーションでは，その初期状態の設定によって，区間推定による収束の度合いが変化する．たとえば，待ち行列の長さが0で，リソースがすべてアイドルである状態（0状態 [empty & idle] と呼ぶ）からシミュレーションを開始すれば，立上がりの状態による揺ぎが区間推定に影響をおよぼし，収束が遅くなる．このようなシミュレーションの試行の最初に現われる過渡的な値による揺ぎを少なくするために，つぎのような処理技法が考えられる.

(i) 最初の何個かの結果を無視することによって，過渡的な値を取除く.

(ii) パラメータの値を変更する場合に，毎回0状態にするのではなくて，シミュレータをそのままの状態にしておき，パラメータの値が少し変化するようなパラメータの組を続いてシミュレートする．これにより，過渡的な状態を少なくすることができる．(Slide 法と呼ぶ.)

このような方法により，揺ぎを小さくして，収束までの試行回数を減少させることができる.

つぎに，過渡的な状態の値（異常値が入ってくることによる収束の状態の変化について検討してみよう.

正規母集団 $N(\mu,\sigma^2)$ の標本を $(x_1, x_2, \cdots, x_n)$ とし，平均値および分散の最尤推定量を H, S^2 とする時，信頼区間幅は，$2t_{n-1}(\varepsilon/2)\sqrt{n-1}$ である．いま x_1 の前に $x_0=H+kS$ なる異常値（正規母集団からの標本と考えられない値）があったとする時，標本 $(x_0, x_1, \cdots, x_n)$ の平均値および分散 $\hat{H}$, $\hat{S}^2$ として

$$\left.\begin{aligned}\hat{H}&\equiv\frac{1}{n+1}\sum_{i=0}^{n}x_i=H+\frac{kS}{n+1},\\ \hat{S}^2&\equiv\frac{1}{n+1}\sum_{i=0}^{n}(x_i-\hat{H})^2=\frac{n(n+1+k^2)S^2}{(n+1)^2}\end{aligned}\right\} \qquad (6)$$

をとると，信頼区間幅は，

$$2t_n(\varepsilon/2)S\sqrt{(n+1+k^2)/(n+1)^2} \qquad (7)$$

と変わる．ここで，第1回目の試行の結果が異常値で，第2回目以後の試行の結果は，$\boldsymbol{N}(\mu,\sigma^2)$ に従っているとし，0状態から開始した場合および第1回目の結果を無視した場合の収束判定による打切り回数をそれぞれ l, m 回（分散を $S_l{}^2$, $S_m{}^2$）とすると，同じ信頼区間幅を得るためには次式を満足せねばならない.

$$t_{m-1}(\varepsilon/2)S_m/\sqrt{m-1}=t_{l-1}(\varepsilon/2)S_l\sqrt{(l+k^2)/l^2}. \qquad (8)$$

そして，0状態から開始する場合の方が $(l-m-1)$ 回多く試行せねばならない．l と m の平均的関係を

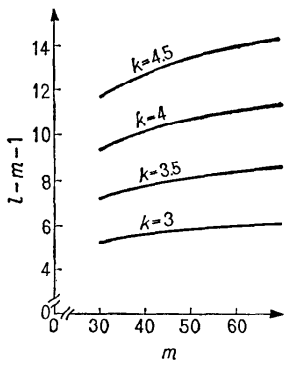

Fig. 7 Relation between m and $l-m-1$

求めるために $S_1=S_m$ とすれば，（8）式はつぎのようになる．

$$l^2(t_{m-1}(\varepsilon/2)/t_{1-1}(\varepsilon/2))^2-(m-1)(l+k^2)=0. \quad (9)$$

Fig. 7 にその場合の $(l-m-1)$ と m の関係を示す．

（3） Direct Search 法

あらゆるパラメータの組合せに対するシミュレーションを試みる必要がなく，最適（最大または最小）の結果を出力するパラメータの値を見出せば良い場合がある．このような場合に考えられる一つの方法が，Direct Search 法である．Direct Search 法とは，試みた結果を順番に吟味していくことを基本にした方法で，単純な比較という作業手続きによって，それ以後の試みるパラメータの値を決めて行く．

関数 $F(\boldsymbol{x})$ が領域 $\boldsymbol{D}$ で唯一つの極大（または極小）しか持たない時，最大（最小）値は，Direct Search によって求められる．ここでは，その一つの方法として，Hooke & Jeeves 法[2] を使用した．なお，以下では最大値を求めることにする．

Hooke & Jeeves 法のアルゴリズム

（I） 始点 x^S での値を $V=F(x^S)$ 求める．
　x, x^B, $\bar{x}^B \leftarrow x^S$.
　但し，x はパラメータ $(x_1, x_2, \cdots, x_d)$ を示す．

（II） Exploratory Move
　（i） $i \leftarrow 1$.
　（ii） $i>d$ ならば，（III）へ飛ぶ．
　（iii） $x_i \leftarrow x_i+\Delta x_i$.
　（iv） $F(x)>V$ ならば，$V \leftarrow F(x)$，（viii）へ飛ぶ．
　（v） $x_i \leftarrow x_i-2\Delta x_i$.
　（vi） $F(x)>V$ ならば，$V \leftarrow F(x)$，（viii）へ飛ぶ．
　（viii） $i \leftarrow i+1$，（ii）へ飛ぶ．

（III） $x=x^B$ ならば，（V）へ飛ぶ．

（IV） Pattern Move
　（ix） $x^B \leftarrow x^B$, $x^B \leftarrow x$, $x \leftarrow 2x^B-\bar{x}^B$.
　（x） （II）.（i）へ飛ぶ．

（V） ステップ幅が十分小さければ，終る．

（VI） ステップ幅を小さくし，$x \leftarrow x^B$，（II）.（i）へ飛ぶ．

7. シミュレーションの結果

（1） システム・パフォーマンス

ここでは，6.（2）で述べた Slide 法により，パラメータの値を少しずつ変化させて，シミュレーションを連続的に実行させた．信頼区間（$\varepsilon=0.05$）の打切り条件は，結果の値と信頼区間の相対的割合（R）および，信頼区間の絶対的な値（A）の2つの場合を考え，今回のシミュレーションでは，$R=\pm 0.05$, $A=\pm 1$（結果の値の単位は%）とし，チャネル使用率で判定した．また，タスク・スイッチ8000回毎の平均を求め，それを1つの標本として各リソースの使用率を求め

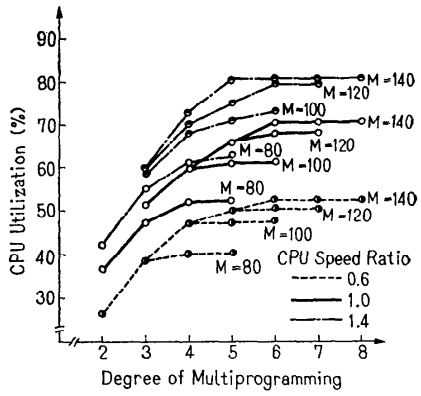

Fig. 8 CPU Utilization

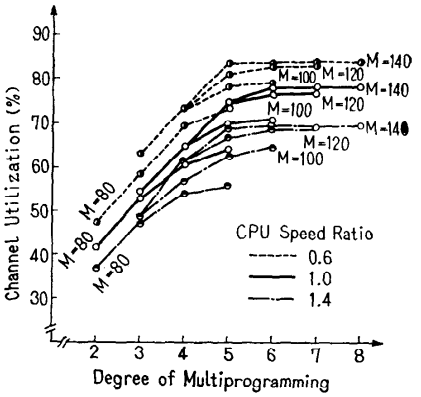

Fig. 9 Channel Utilization

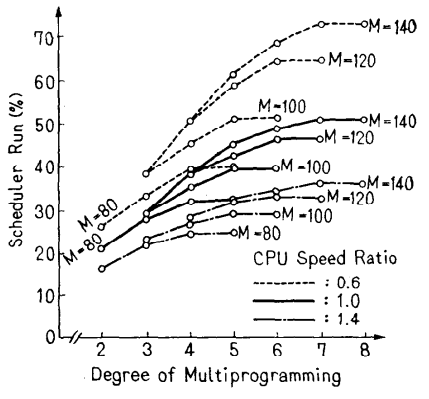

Fig. 10 Scheduler Run

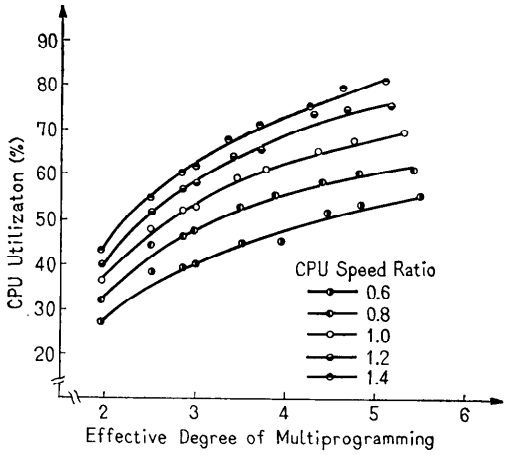

Fig. 11 CPU Utilization

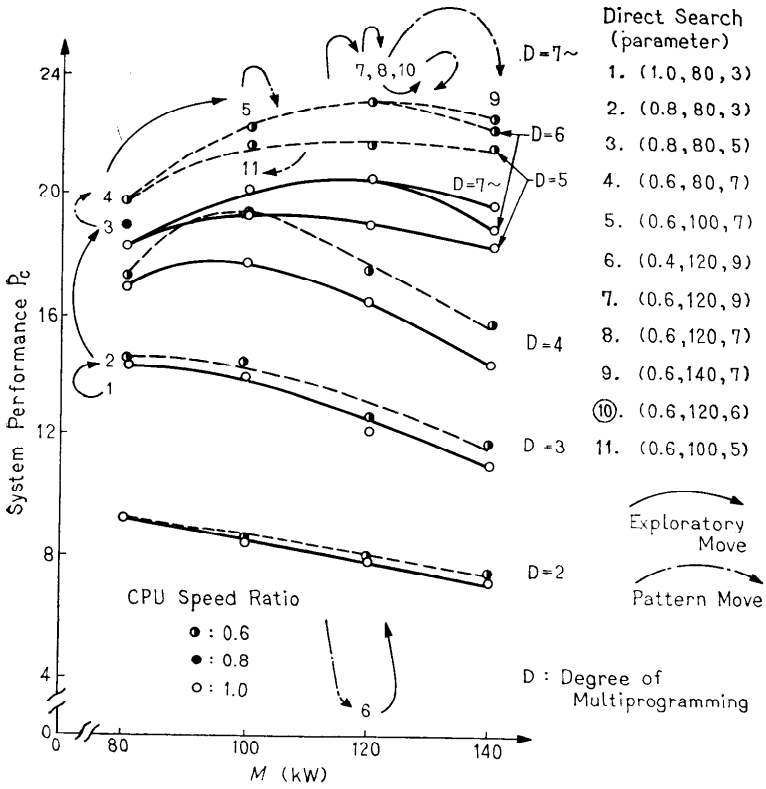

Fig. 12 System Performance and Maximum Value by Direct Search Method

た．その時の CPU，チャネル（入出力用）の使用率，スケジュラ走行率と多重度の関係を **Fig. 8, 9, 10** に，CPU 使用率と実効多重度*の関係を **Fig. 11** に，（2）式のシステム・パフォーマンス P_C を **Fig. 12** に示す．

パラメータ (1.0, 120, 5)**または (1.0, 120, 6) の場合のモデルは，対象としたシステムにほぼ相当している．CPU をさらに有効に使用するため，シミュレーションに用いたジョブ特性*** を考慮し，システムへの負荷を長時間計算が含まれるジョブ・ミックスに変えて，現在システムを動かしている．その結果は，シミュレーションの結果よりも CPU の使用率は約 10% 高く，チャネル使用率は約 20% 低くなることが測定された．[5]

（2） シミュレーションの効率

6.（2）で述べた収束を速める方法の有効性を示す一例として，0状態から始めた場合および Slide 法による場合の平均値を **Fig. 13** に示す．0状態から始めた場合に，第1回目の結果を過渡的状態による異常値とみなし，第2回目以後の結果によって，区間推定を行なえば，第47回目で十分の精度が取れている．（9）式に，$k=3.3$，$l=56$ を代入すれば，$m \fallingdotseq 48$ となるが，この例で Slide 法の方がシミュレーションの収束がさらに速くなっているのは，第2回目のデータも過渡的状態にあるためと考えられる．

システム・パフォーマンスの測度として，（2）式の P_C を取り上げて，その値が最大になるパラメータの値を求めるために，Direct Search 法を利用した．P_C の値を決定する関数が，あらかじめ与えられた変域で高高1つの極大しか持たないこと

* コア・メモリ専有時間とスケジュラ時間の総和をシステム稼動時間で割ったものであって，コア・メモリの空き待ちによる実質的な多重度の低下を考慮に入れたものである．

** (CPU 速度比，ユーザ・プログラム領域，設定多重度)

*** コンパイル・ステップでエラーのため打切られるジョブが 60% 近く，一方，実行ステップでの CPU 専有時間は平均 66 秒で長時間計算が入っていない．

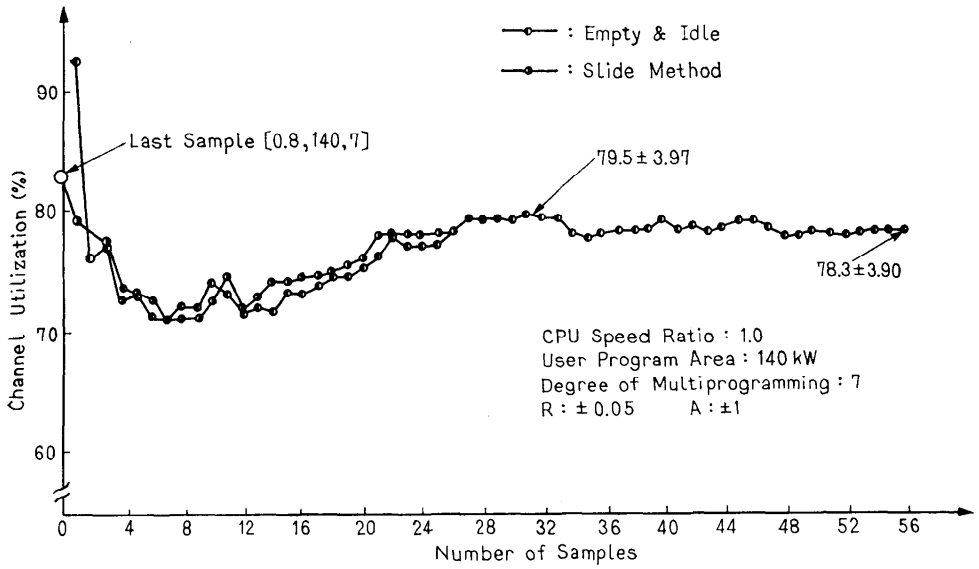

Fig. 13 Comparison of Convergence (Channel Utilization)

は，保証されているものとした*．Pc に関する Direct Search の動きを **Fig. 12** に番号で示す．ただし，Exploratory move は，成功したものだけを示した．

8. あとがき

このシミュレーションでは，システムの平均的な動作状態を求め，その結果によってシステムのパフォーマンスを評価しようと試み，巨視的に把握することができたと考えられる．一方，微視的な観点からシステムをとらえることは，シミュレーションの近似度から考えて困難な場合が多く，モデルに相当の精度が要求される．しかし，システムのパフォーマンスを評価して，その隘路となる部分を知ろうとするときには，平均的な状態だけでなく，パフォーマンスの動的な変化を把握することも必要になってくる．

Slide 法や Direct Search 法は，シミュレーションに要する時間（計算時間だけでなく，パラメータの設定などに関して人間が考えたりする時間も含めて）の短縮に有効であると考えられる．

計算機システムの解析を行なう場合，シミュレーションの他に，実システムの測定や数学的解析などの手法がある．これらのうちの1つの方法だけで計算機システムの解析を進めていくのではなくて，各々の手法の長所を生かして，その機能的役割を分担するとともに，得られた結果を相互にフィード・バックさせることが必要と考えられる．

なお，ここに述べたマルチプログラミング・システムのシミュレーションは，京都大学大型計算機センターの開発計画の1つとして行なわれたものである．

* 実際のシステムでは，多重度が大きくなれば，タスク・スイッチやロールイン/アウトによるオーバーヘッドが増加するため，ここでは，Pc が同じ場合は多重度の小さいパラメータの組を良とした．

参考文献

1) Calingaert, P.: "System Performance Evaluation: Survey and Appraisal," Comm. ACM, Vol. 10, No. 1, pp. 12-18, 1967.
2) Fine, G. H. and P. V. McIsaac: "Simulation of a Time-Sharing System," Management Science, Vol. 12, No. 6, pp. 180-194, 1966.
3) Kowalik, J. and M. R. Osborne: "Methods for Unconstrained Optimization Problems," Elservier, 1968.
4) 富士通："FACOM 230-60 MONITOR-V システム解説篇 Ⅰ/Ⅱ"，マニュアル，EX-041-2, EX-043-2.
5) 北川他："オペレーティング・システムのパフォーマンスモニタリング"，情報処理，11 月号，1972（予定）．

（昭和 47 年 2 月 24 日 受付）
（昭和 47 年 4 月 24 日 再受付）