オペレーティング・システムの動的資源管理方式の評価

久保秀士，土井根敏明，大野直哉，小野隆喜
（日本電気(株) 中央研究所）

1．はじめに

オペレーティング・システムにおける資源管理はコンピュータ・システムのハードウェア的及びソフトウェア的リソースを管理し，個々のジョブに適切に割当てることにより，システムとしての性能目標の達成を図る。性能の指標はエンドユーザから見ると応答時間であり，システム管理者からはスループットである。コンピュータ・システムの形態の複雑化，特にオンライン，TSS，RJE，バッチが共存し並列処理される多次元処理の採用とシステムの大規模化の傾向から，各ジョブから発せられる応答性要求は多様化し，スループットに対する要求も厳しくなってきており，また，両者のバランスのとり方の制御も重要になってきている。時々刻々変動するジョブの特性，システムの状況に応じた制御を行うため，資源管理は動的に行われる必要がある。

過去においても，断片的な動的資源管理方式は広く研究対象になっており，インプリメントもされているが[1]，スループット，応答性を総合的にバランスよく達成する制御はIBMのOS/VSえのSRM（System Resources Manager）で初めて試みられた[2]。SRMではアドレス空間を単位とするスワッピングを制御手段として用いている。スループット向上方式は入出力系をチャネル単位で扱うためにきめが荒く，応答性に関してはサービス率の指定に基く制御をしているが，これは各ジョブの応答性要求に直接的に答えることにはならないと思われる。また，パラメタの設定が煩雑であり，物理的イメージの明確でないものが多い。

ここでは，よりきめ細い制御によるスループットの向上と，直接的な応答性指定に基く応答性の制御及びこの両者のバランスを制御可能にすることを狙いとして，ディスパッチング，スワッピングの各レベルにおいて様々な制御方式を考え，シミュレーションを用いて比較評価を行っている。

2．シミュレーション・モデル[3]

2.1 モデルの構成

モデルの基本的構成を図1に示す。バッキング・ストア，主記憶，CPU，入出力機器の各リソースの割当てに関して制御が可能であるが，ここでは主記憶の割当てをスワッピング・レベルの制御として，CPU及び入出力機器の割当てをディスパッチング・レベルの制御として，この両者を扱っている。

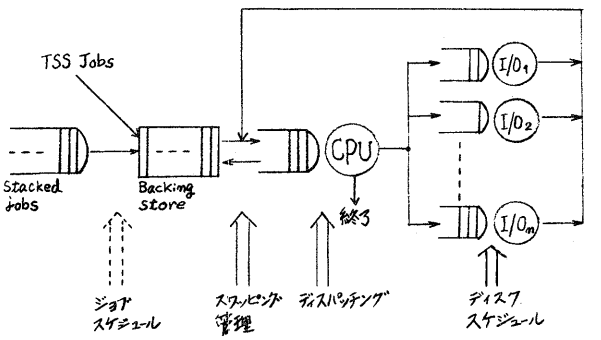

図1．モデルの基本的な構成

モデルにおけるシステム構成とハードウェアの性能を表1に示す。ディスクのシーク時間はファイルの配置の効果を考慮してハード的な平均値より短くしてある。ディスク制御装置における競合もシミュ

表1. システム構成及びハードの性能

	台数	性能
CPU	1	1 MIPS
ディスク (ファイル用)	12* (MSC 2台)	Seek(平均) 20 ms 回転待ち 8.3 ms データ転送 1MB/s
ディスク (スワップ用)	1 (MSC 1台)	同上
CR	2	1050枚/分
LP	3	1500行/分

＊応答評価は8台で行っている

レートされる。

モデルはGPSS/V-6(ACOS-6)で記述された。OSの制御機能でとにトランザクションを用意し，必要に応じてそれらが動くようにモデル化している。これにより制御アルゴリズムの切換が容易になっている。ジョブも各々が一つのトランザクションで表現され，これらが図1のモデル上を動き回る。

2.2 ジョブ・プログラム・モデル

システム負荷の特性はジョブ発生時に決定される特性，到着特性，実行時に決定される特性で定まる。到着特性はTSSではポアソン到着とし，バッチではスタックされている状態から開始されるものとする。他の2つの特性(プログラム特性)に関しては図2に示す手順で決定した。ここでは制御アルゴリズムの比較を可能にするため，次の2点に留意している：①シミュレーションのケース(制御方法の別)により負荷特性が変らないようにすること，②アルゴリズムが制御の手がかりにしようとしているジョブの性質は正確に反映すること。

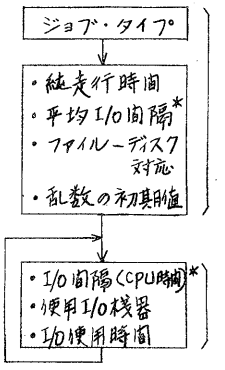

図2. ジョブ特性の決定手順

ジョブ発生時にまずタイプを決定する。これに従ってジョブ固有の性質が定められる。バッチ・ジョブには9タイプある。表2にその特性を示す。純走行時間とはシングルプログラム時の走行時間である。I/O間隔にはOSの入出力管理の時間も含んでいる。これらの特性値は稼動中のプログラムの実測値に基づいて決定した。他の性質としてアクセスするファイルとその使用比率がある。ファイルとディスクの対応はジョブ毎に決定する。従って，ジョブ毎に各ディスクの使用比率が与えられることになる。また，ジョブ毎に乱数発生器をもたせて実行時における入出力先の機器，I/O間隔の決定に使用し，各ジョブについてケースによらず全く同一の特性をもたせるようにした。[6]

表2. ジョブとジョブスタックの特性

特性 / ジョブタイプ	平均純走行時間(秒)	平均I/O間隔(Kstep)	特性⑤の発生個数	特性⑥の発生個数
FTN-CPL	20.0	6.5	6	20
COB-CPL	30.0	5.0	8	3
LINK	15.0	3.0	18	22
COB-OBJ1	240.0	2.0	13	2
COB-OBJ2	200.0	10.0	10	5
FTN-OBJ	40.0	100.0	9	22
SORT	200.0	2.5	8	3
INPUT-RDR	40.0	3.0	15	13
OUTPUT-WTR	120.0	6.5	13	12

2.3 シミュレーション条件

スループット向上方式の評価においてはジョブを100個発生させてスタックしておき，その全体を処理終了するまでシミュレートしている。ジョブ・スタック特性は事務計算指向(特性⑤)，科学技術計算指向(特性⑥)の2種類を用意した(表2)。

シミュレーションでは次の点で簡略化したモデルを使っている。

1) 主記憶とバッキング・ストアの多重度はそれぞれ固定とする。(プログラム・サイズの違いは意識しない。)
2) 各ジョブ・プログラムはCPUとI/O機器を逐次的にしか使用しない。
3) バッチ・ジョブ・プログラムの純走行時間は実際の(表2の)1/4に設定する。
4) 資源管理のためのオーバヘッドは原則として考慮しない。

3. スループット向上方式の評価[4),5)]

3.1 シミュレーション条件

まずディスパッチング・レベルの，次にこの結果を踏えてスワッピング・レベルのシミュレーションを，次の条件の下で行った。

1) ジョブの選択：ディスパッチング・レベルではジョブの発生順に主記憶へ入れ，スワッピング・レベルでも同様にしてバッキング・ストアに入れる。

2) 主記憶及びバッキング・ストア（BS）の多重度：主記憶の多重度は8 BSの多重度は16にそれぞれ固定する。但し，RDRタイプのジョブ2個，WTRタイプのジョブ3個まではこれと別枠で入りうるものとする。

3) 制御のオーバヘッド：スワッピング・レベルの評価では，ジョブ単位のスワップ・インまたはアウトにCPU時間10 ms，ディスクとの転送時間100 msを仮定している。その他のオーバヘッドは考慮していない。

3.2 ディスパッチング・アルゴリズム

シミュレーション評価の対象にした制御アルゴリズムは以下の通りである。

1) DFIFO方式：CPU及びI/O機器に対する待ち行列は先着順に処理される。

2) DDO方式：リソースの稼動率の和が常に最大になるようにディスパッチング・プライオリティを制御することを狙いとする。このため，リソースを i，主記憶上のジョブを j で表現し，単位時間にジョブに割当てられるCPU時間を t_j とした時，目的関数 F を次のように定義する。

$$F = \sum_j \left(\sum_i n_{ij}/n_i \right) t_j$$

但し，n_i：単位時間当りのリソース i の処理能力
n_{ij}：ジョブ j がCPUを単位時間使用する間に用いる他のリソース i の使用量

この F が次の制約式の下で最大になる t_j を線型計画法（LPP）で求める。

$$\begin{cases} \sum_j t_j \cdot n_{ij} \leqq n_i \\ t_j \leqq D_j \end{cases}$$

但し，D_j：ジョブ j の単位純走行時間当りのCPU時間

求めた t_j を使用して，$IOC_j \times t_j / D_j$ を計算し，この値の大きい順にCPUプライオリティをつける。IOC_j はジョブ j にCPUを単位時間割当てた時の総I/O回数である。I/Oプライオリティは CPUと同じとする。

3) DK方式：スタック・ジョブ全体としての各リソースの使用比率が予め判っているものとして，常にこの比率に近い比率で各リソースが使用されるようにすることを狙う。このため，DDO方式と同じくLPPを用いて，次の制約式の下で F が最大になる t_j を求める。

$$\begin{cases} \sum_j t_j \cdot n_{ij} \leqq a_i \cdot n_i \\ t_j \leqq b \cdot D_j \end{cases}$$

但し，a_i：リソース i の使用比率
b：ジョブのアクティブ時間の制約係数

求められた t_j を使用し，DDO方式と同様にしてCPUプライオリティを決定する。I/Oに関しては先着順で処理している。

4) APG方式：CPUプライオリティは平均I/O間隔の短い順につける。モデル上で測定した値をもとに一定周期毎に計算している。I/Oに関しては先着順処理である。

5) SR方式[6)]：CPUプライオリティはAPGと同じ方法でつける。I/Oに関しては各機器ごとに，その機器を使う割合の少いジョブ程高いプライオリティをつける。使用割合はジョブ発生時に，各ジョブ j が各リソース i を使用する時間割合 x_{ij} を求めておいて，この値を使っている。

6) DS方式：CPUプライオリティはAPGと同じ方法でつける。ディスクへの入出力要求についてはディスク別待行列の中から現在のアーム位置に最も近いシリンダにI/O要求を出しているものを選択し，サービスする。

3.3 スワッピング・アルゴリズム

スワッピング・レベルの評価でとり上げた制御アルゴリズムは以下の通りである。すべて，ジョブ特性データは測定値でなく設定値を用いている。

1) SFIFO方式：スワップ・インの順序はジョブ発生順であり，強制的なスワップ・アウトは行わない。ディスパッチング・レベルの制御はAPGと同じであるが，測定値でなく，ジョブ発生時に定められた平均値に基いてプライオリティを定めている。

2) SDO方式：シミュレーション開始時及びジョブ終了時に，新たにスワップインすべきジョブを決定するため，バッキング・ストア中の全ジョブを対象に，次の目的関数Fと制約式をもつLPPを解き，t_jを求める。

$$F=\sum_j\left(\sum_i n_{ij}/n_i\right)\cdot\delta_j t_j \quad , \quad \text{制約式} \begin{cases} \sum_j t_j\cdot n_{ij}\leqq n_i \\ t_j\leqq D_j \end{cases}$$

[但し，δ_jはジョブjが主記憶上にあれば1.0，そうでなければ0.85とし，スワッピングによる損失を折込んでいる。

そして，t_j/D_jの大きい順に主記憶の多重度分のジョブを選択し，スワッピングを行う。ディスパッチング・レベルの制御はDDO方式と同様に行う。

3) SK方式：SDO方式と同じ目的関数，DK方式と同様の制約式をもつLPPを解いてt_jを求め，SDO方式と同様にスワッピングを行う。ディスパッチング・レベルの制御はDK方式と同様に行う。

4) DSM方式：ジョブ終了時及び一定周期毎に，CPU及びディスク制御部の使用率が，各々に対して設定された許容使用率の範囲内に常に納まるように，スワッピングで制御する。ディスパッチング・レベルの制御はAPG方式で行う。

5) SLB方式：文献[7]の提案に改造を加えた方式である。主記憶上のジョブを対象として，SR方式で定義したx_{ij}を使用して$X_i=\sum_j x_{ij}$を定義する。ジョブ終了時には，終了したジョブを除いてX_iを算出し，BS上の全ジョブについて$E_j=\sum_i X_i x_{ij}$を求め，E_jが最小のジョブをスワップ・インする。また，一定周期毎に主記憶上のジョブについて$E_j=\sum_i(X_i-x_{ij})\cdot x_{ij}$を求め，この値の最大になるジョブ（最大値を$E_{max}$とする）を決定し，次に今求めたジョブを除く残りの主記憶上のジョブについて，再度X_iを算出する。このX_iを用いてBS上の全ジョブに対して$E_j=\sum_i X_i\cdot x_{ij}$を求め，この最小になるジョブ（最小値を$E_{min}$とする）を選択する。そして$E_{max}$が$E_{min}$より大きく，その差が一定値以上の時，これらのジョブのスワッピングを行う。ディスパッチング・レベルの制御はAPG方式で行う。

3.4 シミュレーション結果と検討

表3にディスパッチング・レベルのシミュレーション結果の一部を，図3にCPUの平均待ち時間とディスクの平均待ち時間の制御アルゴリズム間における比較を示す。表3に示すように，ジョブ・スタック処理時間は，静的な制御（DFIFO）の場合に比べ動的な制御を導入することによって約12%～20%程度短縮されている。アクティブ・リソースの平均数（CPUとディスクの使用率

表3. ディスパッチング・レベルのシミュレーション結果

負荷	項目＼アルゴリズム	DFIFO	DDO	DK	SR	DS	APG
事務計算指向	全処理時間 (sec)	537.3	464.1	477.1	480.5	470.0	476.2
	CPU使用率 (%)	51.2	59.3	57.7	57.3	58.6	57.8
	CPU平均待ち時間 (ms)	14.27	2.41	3.35	2.41	2.38	2.32
	ディスク平均待ち時間 (ms)	17.06	18.16	16.81	16.90	17.00	17.95
	同時にアクティブなリソースの平均数*	5.29	6.02	6.02	6.13	6.11	6.08
技術計算指向	全処理時間 (sec)	349.9	296.9	289.5	292.2	280.6	290.3
	CPU使用率 (%)	68.3	80.5	82.6	81.8	85.2	82.3
	CPU平均待ち時間 (ms)	49.95	16.6	16.0	13.97	14.80	14.45
	ディスク平均待ち時間 (ms)	23.81	23.07	20.24	22.44	20.66	25.78
	同時にアクティブなリソースの平均数*	4.33	5.17	5.43	5.43	5.17	5.35

＊ 主記憶多重度 ≧ 8 の間の値（CPUとディスクが対象）

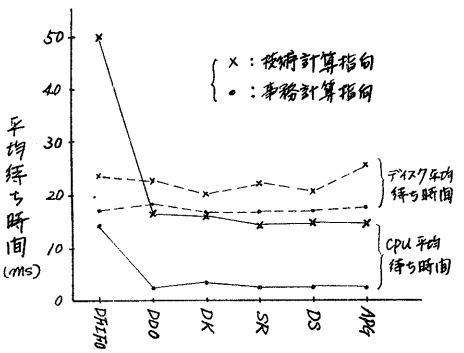

図3. ディスパッチング・レベル制御の待ち時間

の和）で見ると，12%～25%程度向上している。しかし，動的な制御アルゴリズム相互の間ではそれ程大きな差は見られない。スループット向上の原因は，図3から明らかなように動的な制御によってCPU待ち時間を大幅に短縮できたことが大きい。一方，図3に示されているように，ディスクの平均待ち時間がいくつかの制御アルゴリズムにおいてDFIFO方式の場合より長くなっている。特に個々のディスクを識別していないAPG方式では，識別して制御しているDK方式などに比べ長くなっている。また，別に行ったやや小規模なシステムに関する同様なシミュレーション結果では，どの動的な制御もDFIFOよりディスクの平均待ち時間は短くなっている[4]。これらのことから，大型のシステムでは個々のI/O機器を意識し，I/Oプライオリティの制御も行うアルゴリズムが望ましいと考えられる。DDO方式において線型計画法で求めた最大スループットと実際に実現されたものとを比較すると，ジョブ特性⑤で81%，⑥で89%の達成度が得られていた。このことから，各ジョブの使うリソースの数が多くシステム規模も大きいような場合が特に制御が困難であり，逆に言うとまだスループット改善の余地が残されていると言えそうである。

次に，スワッピング・レベルのシミュレーション結果の一部を表4に，平均

表4. スワッピング・レベルのシミュレーション結果

負荷	項目＼アルゴリズム	SFIFO	SDO	SK	DSM	SLB
事務計算指向	全処理時間 (sec)	438.3	437.5	411.1	424.1	420.4
	CPU使用率 (%)	67.0	67.3	71.7	69.4	70.0
	1I/O出すまでのCPU平均待ち時間 (ms)*	2.86	3.85	3.68	3.89	2.78
	1I/O要求当りのディスク平均待ち時間 (ms)*	18.93	14.64	13.12	17.14	13.04
	スワップイン/アウト回数	72	110	168	106	112
	同時にアクティブなリソースの平均数*	4.92	5.02	5.23	4.95	5.23
技術計算指向	全処理時間 (sec)	280.0	279.1	270.0	270.7	266.6
	CPU使用率 (%)	93.4	93.7	97.0	96.6	98.2
	1I/O出すまでのCPU平均待ち時間 (ms)*	33.79	30.24	29.81	28.05	25.89
	1I/O要求当りのディスク平均待ち時間 (ms)*	26.14	22.02	19.48	27.24	21.42
	スワップイン/アウト回数	75	87	119	93	113
	同時にアクティブなリソースの平均数*	3.59	3.73	3.91	3.81	3.92

＊ 主記憶多重度 ≧ 8 の間の値

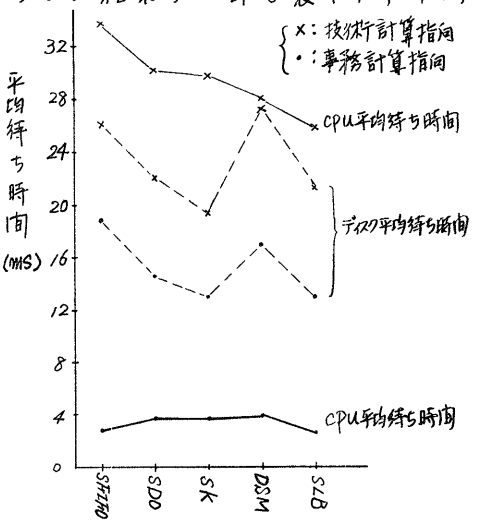

図4. スワッピング・レベルの待ち時間

CPU待ち時間と平均ディスク待ち時間を図4に示す。SLBとSKの二方式が他の方式に比べて効果が大きく，ディスパッチング・レベルの制御による向上の上に更に6%～9%のスループット向上が得られている。個々のディスクまで考慮する方式は考慮しない方式に比べて向上度が大きいが，このことは図4における待ち時間で裏づけられており，個々のI/O機器まで意識するのが望ましいことが示されている。SFIFO方式はジョブ・スケジュールの順序によってスループットが大きな影響を受けるが，SKやSLB方式を導入すると影響をほとんど受けず，安定した高スループットが得られることも確認されている。スケジュール順序が悪い場合はSFIFOより17%の向上が得られた例がある[5]。

4. 応答性向上方式の評価

応答性については，直接的に応答時間目標を設定しこれを達成するように制御することが望ましい。スループットを低下させない配慮も重要である。以下ではこのためにここで採用した方式の説明とその評価結果について述べる。

4.1 方式の概要

応答性目標：　個々のジョブ，インタラクション，メッセージ等は何らかの応答要求タイプに属すものとする。このタイプ毎に応答性目標が，目標応答時間とその目標達成率により規定される。

GP (General Priority)：　応答時間管理は各ジョブごとに，目標の達成の厳しさを表わす指標としてGPを与える（時間的に可変）。これをディスパッチング制御，スワッピング制御で参照し，各ジョブのGPに対応して処理の優先度を考慮することにより応答時間管理が行われる。

4.2 GPの制御

応答時間管理は与えられた応答性目標を満すために，各ジョブに適切なGPを与える必要がある。GPの与え方として次のような方式を導入する。

a) デッドライン制御[8]：各ジョブの投入時には比較的低いGPを与え，目標応答時刻が迫るにつれて高いGPを与えるようにする。

b) 個別制御：　同一の応答性要求タイプに属するジョブでも，個々に目標応答時間の達成の難易が予測できる場合，要求の厳しいジョブには高いGPを，達成の容易なジョブには低いGPを割当てるようにする。

c) フィードバック制御：　各時点における全体の負荷，各タイプ毎の負荷状況，目標応答時間の達成状況に応じて，各タイプに与えるGPを制御する。

これにより，各負荷レベルに応って一定の応答時間を達成できることになる。

4.3 制御アルゴリズム

a) スワッピング管理

GPに3つのレベル（L1, L2, L3）を設け，各ジョブが属すレベルに応じてスワッピングを行う。すなわち，L1は最も高応答が要求されるレベルで，L1のジョブが発生すると主記憶に空きがない場合でも，これより低いレベルのジョブを追出して空きを作った上でスワップインが行われる。L2はある程度応答時間を要求するレベルである。L2にあるジョブは，他のジョブの終了等で主記憶に空きができた時は優先的にスワップインされるが，他のジョブを追出してまではスワップインされない。また，スワップインに際しては，このレベルの中でもGPの高いジョブが優先される。L3は最も優先度の低いレベルで，このレベルのジョブは応答時間の観点からはスワップインの対象にはせ

ず，L1，L2のジョブが無い時にスループットをよくするという立場から選択されてスワップインされる。

このように3つのレベルを設けたのは，スワッピングが頻繁に発生するような事態を防ぐことを狙ったものである。また，バッチジョブの主記憶上での多重度に制限を設け，主記憶に空きがある場合でも，この多重度以上はバッチジョブをスワップインしないようにしている。これはバッチジョブのスワップアウトされる回数を抑えるためである。

(2) ディスパッチング管理

GPの値により2つのレベルに分け，ディスパッチングの行い方を区別する。即ち，下のレベルにあるジョブにはAPG方式で優先順位を与え，上のレベルにあるジョブにはこれより高い優先順位を与える。

4.4 シミュレーションの方法

固定優先順位方式，スワッピング・レベルだけにデッドライン制御を導入した場合，ディスパッチング・レベルでも考慮した場合，個別制御を導入した場合，の各場合についてシミュレーションを行った。モデルは2.に示した通りである。

応答性向上方式の評価のシミュレーション条件を次に示す。システムではバッチ・ジョブ及び4タイプのTSSジョブが実行される。表5にTSSのジョブ特性，応答目標を示す。バッチ・ジョブは特性⑤（事務計算指向）に従って最初60個発生させてスタックしておく。各バッチ・ジョブにも応答目標（3タイプ）が規定されている。主記憶多重度は8に，バッチのBS多重度は10に固定するが，TSSジョブのBS多重度に制限は設けない。スワップ・インまたはアウトのためのCPUオーバヘッドは50msとする。シミュレーションはTSS負荷（TSSジョブの発生個数／秒）が1.0，1.5，1.67，2.0の各場合について行った。シミュレーションは残りバッチ・ジョブ数が（主記憶多重度－1）になると終了するものとする。

表5. TSSの特性　(*) 分布は位相2のアーラン分布

ジョブ・タイプ	ジョブ特性			応答目標	
	(*) 平均純走行時間(ms)	(*) 平均I/O間隔(ms)	発生率	目標応答時間(sec)	目標未達成率(%)
TSS 1	520	16	0.53	5	5
TSS 2	2000	5	0.15	10	10
TSS 3	3500	17	0.07	15	10
TSS 4	650	50	0.25	5	10

固定優先順位方式ではバッチ・ジョブに対してCPU時間1000msのタイムスライシングを行う。終了時に自分と同等か高優先度のジョブがあれば，それらの間でスワッピングが行われる。TSSジョブに関してはタイムスライシングなしの方がよい応答性が得られたので，なしとした。優先度はバッチ・ジョブよりもTSSジョブを高くし，バッチ・ジョブ内には応答目標に応じて3ランク設けた。

デッドライン方式ではGP1～3をL3，4～5をL2，6～8をL1とし，GP7以上はディスパッチングでAPG用よりも高い優先度を与えている。デッドライン制御では目標応答時間の1/4経過するごとにGPを1つ上げていく。バッチ・ジョブには1～5の間のGPを与え，L3ジョブのスワップ・インまたはアウト対象の決定はSK方式に基づいて行っている。

なお，両方式とも，バッチ・ジョブの主記憶多重度に制限を設け，その制限を越えてはバッチ・ジョブはスワップインできないとする。また，今回のシミュレーションではフィードバック制御は，シミュレーション結果を見てパラメタ値を修正することにより，オフラインで行われている。

4.5 シミュレーション結果および検討

図5～7に，TSS負荷（TSSジョブの発生個数／秒）を1.5，1.67，2.0とした時の各アルゴリズムにおける応答時間目標の未達成率を示す。また，表6に，これらの各ケースにおける応答時間，スループット等を示す。

(1) スワッピング・レベルでデッドライン制御を導入する効果

ディスパッチングはすべてAPGを適用し，スワッピング・レベルだけにデッドライン制御を導入した場合について見る。この場合にはプライオリティの与え方により制御できるのは，主記憶の待ち時間だけである。固定優先順位方式で各タイプに対して等しいプライオリティを与えると（FIFO，表6の①），表6に示すように，主記憶の待ち時間は各タイプに同様に分配される。このため，図5に示すように，目標応答時間の短いTSS1，TSS4は目標応答が達成できていない。これを改善するために，TSS1，TSS4の主記憶待ちを小さくするようにプライオリティを調整したのが②のケースである。固定優先順位方式の場合には，あるタイプの待ちは各タイプのプライオリティの順序関係により直接影響される。このため，②のケースでTSS3の未達成を改善しようとしても，TSS4の待ちが増えすぎてこれ以上調整できない。

ケース③は，デッドライン方式でTSSの全タイプにGP5～7（初期値5で最高7まで行きうる）の等しいGPを割当てた場合であり，TSS4は未達成，TSS1は過達成である。④はこれを改善するためにGPを調整した結果である。デッドライン方式では，GPの範囲を変えることによりきめ細い調整ができるので，目標に近い応答性が実現し易いということが言える。

なお，デッドライン方式の場合には，全タイプに等しいGPを与えても，目標応答時間の長さによりGPの上昇速度が異るため，目標応答時間の短いタイプのジョブの主記憶待ち時間を自動的に小さくするという効果がある。

(2) ディスパッチング・レベルにもデッドライン制御を導入する効果

ディスパッチング・レベルにもデッドライン制御を導入することにより，負荷レベル1.5個，1.67個／秒で，固定，スワッピングのみデッドライン導入，の各場合ではできなかった応答目標の達成が可能となっている（ケース⑤，⑦）。これは，ジョブ特性がCPUバウンドであり，かつ目標応答が短いために僅かな主記憶待ち時間しか許されなかったTSS4が，ディスパッチ・レベルで優遇されるようになったことにより，主記憶待ち時間に余裕が生じたことが大きく貢献している。なお，固定優先順位方式で，TSS4に対してディスパッチング・レベルでも高いプライオリティを与えてしまう（ケース⑧）と，TSSは過達成になり，スループットは低下する。

この時の応答特性を図8に示す。デッドラインを導入すると，固定優先順位方式の場合に比べて不必要に早く終了するジョブが少なく，分散が小さくかつ目標達成度の高い，かなり良好な応答特性が得られることがわかる（ケース⑦）。

負荷が比較的低い場合には（1.5個／秒），スワッピング・レベルだけでデッドラインを導入する場合と比べてGPの初期値を低くしても，目標応答が達成できている。これにより，スワッピング回数は減少しスループットも向上している。なお，負荷レベル1.5／秒におけるCPUの使用率はTSS：44～46％，バッチ：32～34％，スワッピングOH：9～11％，ディスク使用率は44～47％，主記憶多重度はTSSが3.0～3.1，バッチが4.0～4.3であった。

以上のことから，デッドライン制御に関して次のことが言える。デッドライ

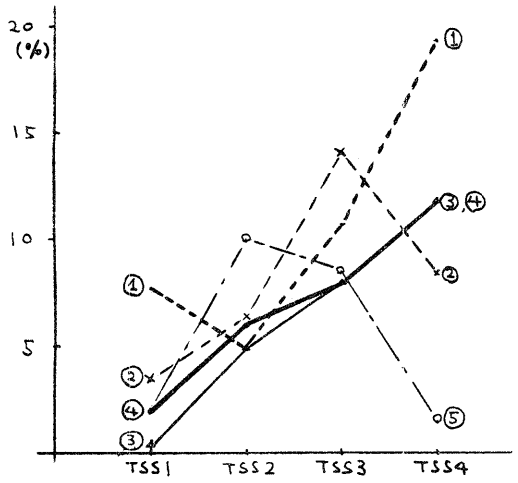

図5 タイプ別の応答目標の未達成率（負荷1.5）

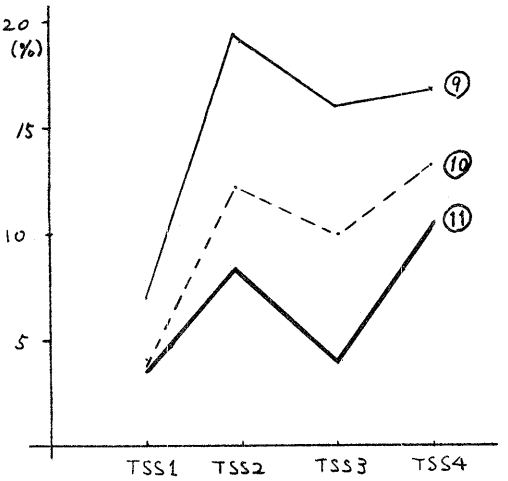

図7 タイプ別の応答目標の未達成率（負荷2.0）

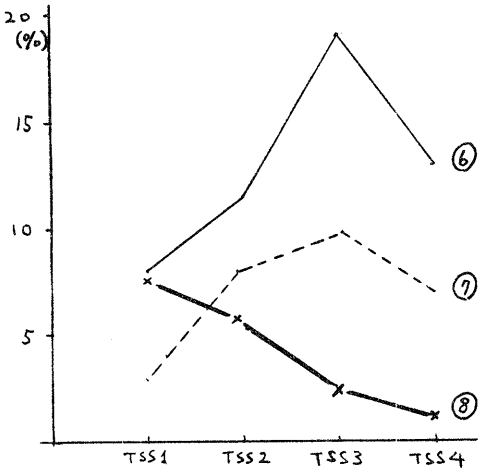

図6 タイプ別の応答目標の未達成率（負荷1.67）

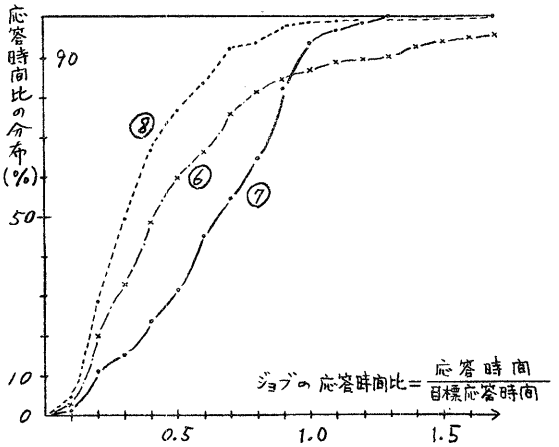

図8 TSS4の応答特性（負荷1.67）

表6. シミュレーション結果

方式	TSS1～4のGPまたは優先度	TSS負荷	タイプ別(TSS1～4)の平均応答時間(sec)				主記憶待ち時間の平均(sec) タイプ別(TSS1～4)				TSS全体	スワップ・アウトの数/sec	スループット(1)
① FIFO(APG)	10, 10, 10, 10	1.5	2.3	4.6	8.3	3.7	1.3	1.2	1.3	1.1	1.23	0.17	3.87
② FIX(APG)	9, 8, 8, 10	1.5	2.2	4.8	8.8	2.7	1.2	1.4	1.9	0.6	1.08	0.17	3.92
③ DL(APG)	5~7, 5~7, 5~7, 5~7	1.5	1.7	4.5	8.4	3.2	0.7	1.0	1.3	0.8	0.85	0.20	3.94
④ DL(APG)	4~6, 4~7, 4~7, 5~7	1.5	2.1	4.9	8.3	3.0	1.1	1.4	1.5	0.7	1.04	0.12	4.09
⑤ DL(APG+DRB)(2)	4~6, 4~7, 4~7, 4~7	1.5	2.2	5.0	8.9	2.9	1.0	1.4	1.7	1.2	1.18	0.09	4.11
⑥ FIX(APG)	9, 8, 8, 10	1.67	2.5	5.8	9.5	3.2	1.5	2.4	2.4	0.6	1.44	0.16	3.94
⑦ DL(APG+DRB)	4~7, 5~7, 5~7, 4~7	1.67	2.7	5.3	9.3	3.2	1.4	1.2	1.3	1.5	1.40	0.13	3.85
⑧ FIX(FIX)(3)	9, 8, 8, 10	1.67	2.1	5.2	8.2	1.9	1.4	1.9	2.0	0.6	1.27	0.12	3.49
⑨ FIX(APG)(4)	9, 8, 8, 10	2.0	2.5	6.9	9.4	3.9	1.4	2.9	2.7	0.6	1.50	0.20	3.63
⑩ DL(APG+DRB)	5~7, 5~8, 5~8, 5~7	2.0	2.5	6.5	10.0	3.2	1.1	1.9	1.9	1.3	1.34	0.24	3.44
⑪ DL(APG+DRB) 個別制御	5~7, 5~8, 5~8, 5~7	2.0	2.7	6.2	9.7	3.2	1.2	1.7	1.7	1.2	1.33	0.26	3.38

(1) RDR/WTRは除く (2) GPが7以上のジョブはディスパッチングでAPG用よりも高い優先度を与える (3) TSSにはバッチよりも高いディスパッチング優先度を与える（TSSの中ではI/O間隔の短いタイプほど優先） (4) バッチのタイムスライス 600ms

ン制御の導入により期待できる最も大きな効果は，固定優先順位方式に比べて応答目標達成度の配分がきめ細かくできるので，フィードバック制御が比較的容易に行えることと，達成率のアンバランスを小さくできるので，固定優先順位方式の場合よりも高い負荷レベルまで対応できることである。

スループットについて見ると，デッドライン制御の導入により高応答を要求するジョブ（L1，L2）とそうでないジョブ（L3）の区別を明確化でき，L3のジョブに関してはスループット向上に専念してスワッピングを行える。これにより，負荷が比較的軽い状況（1～1.5個／秒）では，固定優先順位の場合に比べて2.5～5％のスループット向上が確認されている。

（3）個別制御の効果

個別制御を導入することにより，固定優先順位やデッドライン制御のみでは対応できなかったような高い負荷レベル（2.0個／秒）においても，目標応答を達成できる（図7，ケース⑪）。これは，個別制御により，必要以上に早く終了するジョブが減少し，一方，元来達成が厳しかったジョブが投入時から優遇されることの効果が表われたものである。

TSSにおけるコマンドの種類，バッチ・ジョブの投入からスケジュールされるまでの時間，あるいは前回のランの時に測定した情報などで目標応答の達成の難易が予測できる場合には，極力この情報を利用すべきであることがわかる。

5. むすび

コンピュータ・システムのスループット向上，応答性向上のために，ディスパッチング，スワッピングの両レベルで制御を行う動的資源管理に関して幾つかの方式を考え，シミュレーションにより評価した。最適な方式を採用すると制御しない場合に比べ18～34％程度のスループット向上を期待できることが示された。応答性に関しては，要求応答時間の達成の直接的保証が重要であるとの認識の下に，デッドライン制御とフィードバック制御，個別制御を組合せる方式を提案し，この有効性とスループット向上方式との適合性を示した。

実際に実現するためには，オーバヘッドも考慮した現実的なアルゴリズムの決定，簡明なユーザインタフェースの設定を行う必要がある。更に，動的資源管理はジョブ・スケジュールも含めて統一的に行わないと効果が小さい。今後はこれらの問題も含めて，更に検討を続けていきたいと考えている。

終りに，日頃御指導頂いている当研究所藤野部長，栃津課長，有益な討論をして頂いた電気通信大学亀田助教授，当研究所箱崎主任に感謝いたします。

参考文献

[1] R.B. Bunt, Scheduling Techniques for Operating Systems, Computer, Oct. 1976, 10-16.

[2] H.W. Linch, J.B. Page, The OS/VS2 Release 2 System Resources Manager, IBM Syst. J., 13, 4 (1974), 274-291.

[3] 久保，他，オペレーティング・システムにおける動的資源管理方式のGPSSを用いたシミュレーション，シミュレーション技術研究会論文集，6,5（昭和52年2月），31-36.

[4],[5] 小野，土井稔，他，動的資源管理方式のシミュレーションによる評価(I)，(II)。情処18回大会，321-324.

[6] 亀田，嶋田，処理装置型資源の一適応制御方式のシミュレーションによる評価，情処17回大会，191-192.

[7] 亀田，Load Balanceのための Swappingの効果，52年信学情報部門全大，340.

[8] D.D. Chamberlin 他，Experimental Study of Deadline Scheduling for Interactive Systems, IBM J. Res. Develop., May 1973, 263-269.