多重プログラミング方式による計算機システムの資源管理最適化に関する研究

A Study of the Optimal Resource Management in Multiprogrammed Computer Systems

１９８２年 ５月

西垣　通

Tohru Nishigaki

ABSTRACT

Multiprogramming and virtual memory are technical features commonly found in current computer systems. They provide the potentiality to achieve high system performance. To this end, however, the operating system must have complicated resource management strategies. Although considerable work has been done related to the computer system performance, many problems remain in the design of the optimal resource management strategies.

The performance of a multiprogrammed virtual memory computer system is discussed. To achieve the optimal performance, a set of resoure management strategies is developed. The major subjects investigated in this thesis are :

(1) a theoretical method for the analysis of resource management strategies ;

A mathematical model is proposed, to analyze the performance of a multiprogrammed virtual memory system. The model predicts the system performance for various resource management strategies by the asymptotic approximation.

(2) a main memory management strategy ;

The virtual memory paging control is discussed in terms of main memory management, and a new strategy is developed. The strategy controls the main memory allotment to the jobs in a multiprogrammed environment so as to attain high resource utilization and moderate paging load.

(3) a secondary memory management strategy ;

The virtual memory paging control is discussed in terms of secondary memory management. Based on the analysis of widely-used strategies, a new strategy is developed. Paging overhead is decreased by the use of the developed strategy.

(4) an integrated resource management strategy .

The scheduling algorithms in addition to the paging control are discussed for the optimal usage of computing resources. The proposed resource management strategy integrates the scheduling algorithms of various resources each of which has so far worked independently. Thus it is possible to achieve the global performance optimization.

AUSZUG

Mehrprogrammbetrieb und virtuelle Speicherung sind die technischen Merkmale, die sich in gegenwärtigen Rechnersystemen weit durchsetzen. Sie ermöglichen hohes Leistungsvermögen. Zu diesem Zweck, muß aber das Betriebssystem komplizierte Betriebsmittel-Verarbeitungsstrategien haben. Obgleich zahlreiche Untersuchungen über das Leistungsvermögen von Rechnersysteme durchgeführt wurden, bleiben viele Probleme beim Entwurf der optimalen Betriebsmittel-Verarbeitungsstrategien ungelöst.

Die Leistungsbewertung von einem Rechnersystem mit Mehrprogrammbetrieb und virtuellem Speicher wird diskutiert. Um optimale Leistung zu erreichen, wird eine Reihe von Betriebsmittel-Verarbeitungsstrategien entwickelt. Der Schwerpunkt der von dieser Dissertation vorgestellten Untersuchungen liegt auf den folgenden Gebieten :

(1) Eine theoretische Methode zur Analyse von Betriebsmittel-Verarbeitungsstrategien :

Ein mathematisches Modell wird zur Leistungsanalyse eines Rechnersystems mit Mehrprogrammbetrieb und virtuellem Speicher vorgeschlagen. Das Modell schätzt annährend die Leistung des Rechnersystems mit verschiedenen Betriebsmittel-Verarbeitungsstrategien ab.

(2) Eine Hauptspeicher-Verarbeitungsstrategie :

Im Hinblick auf eine Hauptspeicher-Verarbeitung, wird die seitenweise Informationsverarbeitung des virtuellen Speichers diskutiert und eine neue Strategie entwickelt. Diese Strategie kontrolliert die Hauptspeicher-Zuordnung für die Arbeitsabläufe bei Mehrprogrammbetrieb, so daß die Betriebsmittel optimal genutzt werden können und nur eine mäßige Belastung der seitenweisen Informationsverarbeitung anfällt.

(3) Eine Sekundärspeicher-Verarbeitungsstrategie :

Im Hinblick auf eine Sekundärspeicher-Verarbeitung, wird die seitenweise Informationsverarbeitung des virtuellen Speichers diskutiert. Auf der Basis der Analyse von weitgebrauchten Strategien, wird eine neue Strategie entwickelt. Die Strategie vermindert die Unkosten der seitenweisen Informationsverarbeitung.

(4) Eine integrierte Betriebsmittel-Verarbeitungsstrategie :

Neben der seitenweisen Informationsverarbeitung werden die Planungs-Algorithmen zur optimalen Nutzung der Rechnungshilfsquellen diskutiert. Die vorgeschlagene Betriebsmittel-Verarbeitungsstrategie integriert die Planungs-Algorithmen der verschiedenen Betriebsmittel, die bisher separat betrieben wurden. Es ist daher möglich, ein globales Leistungsoptimum zu erzielen.

目　　次

1. 序　論 …… 1

1.1 緒　言 …… 1

1.2 本研究の目的と範囲 …… 1

1.3 本研究と従来の諸研究との関係 …… 6

2. 多重プログラミング・システムの性能解析モデル …… 11

2.1 概　要 …… 11

2.2 漸近モデル（Asymptotic Model） …… 11

2.3 解析例 …… 19

2.4 シミュレーションと実測による検証 …… 24

2.5 結　言 …… 27

3. 仮想メモリ・システムのワーキングセット最適化方式 …… 29

3.1 概　要 …… 29

3.2 Space Time Product と処理能力 …… 29

3.3 ワーキングセット最適化方式（Optimum Working-set Estimator） …… 31

3.4 実験と評価 …… 34

3.5 結　言 …… 38

4. 仮想メモリ・システムの二次メモリ最適管理方式 …… 40

4.1 概　要 …… 40

4.2 二次メモリのスロット割当て …… 40

4.3 固定方式と浮動方式の比較解析 …… 43

4.4 スロット割当ての最適化と集中方式（Concentrating Technique） …… 52

4.5 実験と評価 …… 56

4.6 結　言 …… 58

5. 多重プログラミング・システムにおける一般資源管理方式 …… 59

5.1 概　要 …… 59

5.2 処理能力と応答性の均衡的向上 …… 59

5.3 一般資源管理方式（General Resources Manager） …… 61

5.4 実験と評価 …… 64

5.5 結　言 …… 71

6. 結　論 …… 73

謝　辞 …… 76

参考文献 …… 76

付　録 …… 84

第１章　序　　論

1. 序　　論

1.1 緒　　言

電子計算機システムは，科学技術をはじめ教育，産業，金融，行政，交通等，現代社会の広範囲な分野に導入されており，既にいくつかの分野においてはその活動に多大な影響を与えている。利用範囲の拡大にともない，利用形態もますます複雑化，多様化しつつあり，この傾向は今後とも続くと予想される。一方，利用者の要求の増大は，ハードウェア，ソフトウェア技術の進歩をうながし，最近の電子計算機システムは高速化，大規模化の一途をたどっている。１９７０年から１９８０年にかけ，代表的な計算機システムの演算処理速度は数マイクロ秒／ステップから数百ナノ秒／ステップ，実記憶容量は数百キロ・バイトから数メガ・バイト，端末数は数端末から数十ないし数百端末に増加した。

このような周囲条件のもとで，オペレーティング・システムに対する最大の要求のひとつは，多様な利用形態に柔軟に対処し，多重プログラミング方式で並行に処理される各種のジョブに，中央処理装置，入出力装置，実記憶装置などの諸資源を適切に配分することである。オペレーティング・システムは，各ジョブの諸資源使用状況や諸資源の利用状況などを常に把握し，大局的見地から資源配分の決定を下すことにより，システムの性能を向上させうると考えられる。ここで性能向上とは，システム全体としての単位時間あたりの仕事量の増大化すなわち処理能力を向上させることと，各ジョブの応答性の向上の問題に帰着される。

システムの処理能力と応答性の均衡的向上を明確な目標とする資源管理方式は，ＩＢＭ社のＭＶＳ[S3)]（Multiple Virtual Storage），日立のＶＯＳ３[H6)]（Virtual-storage Operating System ３）などの仮想メモリ方式の大型オペレーティング・システムにおいて，１９７０年代の半ばに実現された。筆者はＶＯＳ３の開発に参加し，資源管理方式の設計，評価を担当した。その間，仮想メモリ方式の大型オペレーティング・システムの資源管理の基本的な機能を把握した。さらに，上記目標達成のためには，なお解決すべき多くの問題が存在することを認識し，これらの問題を解決するために研究を行なってきた。ここに，これまでに得られた研究成果を報告し，批判をうけるとともに，今後この方面の研究，開発に携わる方々の参考としたい。

1.2 本研究の目的と範囲

計算機システムは，ＣＰＵ（中央処理装置），入出力装置，実メモリ等の組み合わせから構成される。ジョブは実メモリを割り当てられることにより，ＣＰＵや入出力装置を使用することが可能となる。統計的にみて大半のジョブは演算動作とファイル入出力動作を交互に実行するため，ＣＰＵと入出力装置とを同時に使用することは少ない。したがって，実メモリ内に１つのジョブしか存在しない「単一プログラミング・システム」では，ＣＰＵや入出力装置に遊休時間が発生し，処理能力，応答性ともに低下する。

複数のジョブに実メモリを割り当て，それらのジョブの間でＣＰＵや入出力装置を共用させる「多

重プログラミング・システム」では，この欠点の改善が可能である。あるジョブがＣＰＵを使用中に他のジョブが入出力装置を使用することにより，ＣＰＵや入出力装置の利用率向上を達成できる。ただし並行処理されるジョブの数は実メモリの量に依存するので，多重プログラミングの効果が発揮される条件として実メモリを効率的に使用することが必要である。ジョブの実メモリ要求量は一般にジョブごとに異なり，その参照する領域はジョブ実行中に変化する。したがって，ジョブ実行に先立って一定量の連続した実メモリ領域を与える「実メモリ方式」では，ある時点に着目すると実際には参照されない領域が生じて実メモリの使用効率が低下する。一方「仮想メモリ方式」では，プログラムが参照する論理的なアドレスと実メモリ上での実際のアドレスとは一般に異なり実行時に対応づけられるので，連続した実メモリ領域を割り当てる必要はない。プログラムは通常磁気ドラム，磁気ディスクなどの二次メモリ上に蓄積され，参照される部分のみが各時点で実メモリ内に読みこまれ，実行される。したがって仮想メモリ方式のもとで，実メモリ使用効率の向上が可能である。

以上のような理由*で，現在では商用の大型計算機システムはほとんど全て仮想メモリ方式の多重プログラミング・システムとなっている。しかし，このようなシステムにおいては資源管理が性能に与える影響が大きく，場合によっては期待した効果が得られない危険がある。資源管理の問題は，上述の二次メモリと実メモリ間の情報転送オーバヘッドに着目することにより，２つに分けることができる。[D8)S5)] 第１の問題はこの情報転送オーバヘッドを最小化することであり，第２の問題は第１の問題が解決されたとして，さらにその上での性能向上にかかわる。

第１の問題は，ジョブへの実メモリの割当て法，ジョブのプログラムの二次メモリ上における蓄積法等に関するものである。これは仮想メモリ方式のシステムに固有の問題である。仮想メモリ方式では，プログラムのアドレス空間は通常ページとよばれる固定区画**に分割される。あるジョブに実メモリが割り当てられる場合，そのプログラムの中で近い将来に参照される可能性の高いページのみが選択されて実メモリ割当ての対象となる。したがって処理の進行にともない，時々刻々，これらのページを二次メモリから実メモリ内に読みこみ，また逆に参照される可能性の低いページを実メモリから二次メモリ上に書き出す処理が必要となる。この処理をページングとよぶ。実メモリ割当て法やプログラムの二次メモリ上での蓄積法が不適切な場合にはページング処理のオーバヘッドが不必要に増大する「ページング・ネック」となり，性能が低下する。

第２の問題は各ジョブの諸資源割当て優先度の動的決定法に関するものである。これは仮想メモリ・システムに限らず一般に多重プログラミング・システムに共通の問題である。ジョブに対する資源の割当て法によっては，多重プログラミングの効果が小さく，かえって制御オーバヘッドのために性能が低下する場合がある。たとえば，ＣＰＵを多用するジョブのみが実メモリ内に収容されていると

＊　仮想メモリ方式採用の理由として，この他にプログラミングの容易さがあげられるが，本論文では性能面のみに着目する。

＊＊　可変区画の方式も考えられるが，現在の商用システムの多くは固定区画なので，本論文ではこれを仮定する。

ＣＰＵがボトルネックとなり，入出力装置には遊休時間が生ずる。オペレーティング・システムは実メモリ割当て優先度の操作により，ＣＰＵを多用するジョブ（ＣＰＵバウンド・ジョブ）と入出力装置を多用するジョブ（ＩＯバウンド・ジョブ）を適切に実メモリ内に混在させる等の方法で，このような状態の発生を防止する必要がある。

以上のような背景のもとに，本論文全体を通じての目的は，上記２つの問題を検討し，仮想メモリ方式の多重プログラミング・システムの性能向上を達成しうる，実用的でしかも理論的根拠のある資源管理方式をみちびくことである。資源管理の最適化を行なうにあたって必要なのは，まず対象を分析し，システムの内部変数と性能との関係を明らかにすることである。資源管理方式の評価手段としてシミュレーション[K4)N2)]や実測[A1)]が従来より用いられるが，これらはむしろ与えられた方式の制御効果の検証手段として有効である。より優れた方式を設計するためには，理論解析的な評価手段が望ましい。

したがって本論文の第１の目的は，仮想メモリ方式の多重プログラミング・システムの性能評価モデルをみちびき，資源割当て優先制御方式やページング処理のオーバヘッドがシステムの処理能力，応答性に与える影響を考察することである。このモデルは，以後本論文で提案する資源管理方式の効果を議論するための理論的枠組となる。

従来，計算機システムの性能解析モデルは，主として待行列理論にもとづいて検討されてきた。すなわち，ＣＰＵ，入出力装置などを各々待行列を有するサービス窓口に対応させ，これらを連結した待行列ネットワーク理論によるモデルが多用されている。待行列ネットワーク理論は，サービス時間がある種の分布にしたがう場合には，諸資源の利用率やジョブの平均応答時間の厳密解を与えるという特長をもっている。しかし，各待行列における待合せ処理方式（queueing discipline）はＦＣＦＳ（First Come First Served），ＬＣＦＳ（Last Come First Served），ＰＳ（Processor Sharing）など比較的単純なものに限られている。すなわち，性能向上のために有効なことが経験的に知られており現実に多用されている優先制御は，待行列ネットワーク理論では求解不可能である。現在，解がえられるのは，システムの状態確率が各待行列の状態変数の積の形（Product Form）となるものに限られており，優先制御の場合はこれが積形にならない。一般に優先度がある場合には，待行列内に同数のジョブが存在する場合でも優先度の組み合わせにより状態数が急激に増大するため，求解は困難となる。さらに，ひとつのジョブが同時に複数の資源を使用する「二次資源問題」についても，状態確率は積形にならず待行列ネットワーク理論により厳密解を求めることはできない。しかし実際には，ジョブは実メモリを割り当てられた上でＣＰＵや入出力装置を用いるので，実メモリは二次資源として扱う必要がある。以上の理由で，既存の理論の範囲内では現実の複雑なシステムの評価は不可能であり，たとえ近似的であっても新しい方法による解析が必要とされていた。

本論文では，優先制御のもとでのシステム性能を与える「漸近モデル（Asymptotic Model）」とよぶ理論的モデルをみちびく[N9)N12)]。本モデルにおいて，実メモリは二次資源として扱われる。本モデルの特徴は，厳密解をその漸近解により近似する点にある。呼源数を独立変数としたとき，諸資源の利用率やジョブの平均応答時間が（資源サービス時間の分布によらず）資源サービス時間平均値のみか

ら定まる或る関数に漸近する。このことは，単一ジョブ・クラスで実メモリ待ちを無視した場合については広く知られているが，本モデルは優先度をもつ複数ジョブ・クラス，二次資源としての実メモリでの待ちを含む場合について漸近解を与える。漸近解は，利用率が最も高くシステムのボトルネックとなる資源に注目し，この利用率を１００％とみなすことにより求まる。実際の性能評価の場においては，系の確率的挙動すなわち分布には再現性があるとは限らないため，むしろ性能の構造的性質（たとえば応答性が急激に低下する負荷条件など）を求めることが必要な場合が多い。本モデルはこの構造的性質を解析する手がかりを与える。本モデルの精度を検討するため，シミュレーション結果および実測結果との比較検証を行なう。なお，本モデルは日立の計算機システム性能評価ツールＩＳＣＰ[01]（Interactive tool for System Configuration Planning）の一環として実用に供されている。

既に提示した資源管理の第１の問題，すなわちページング・ネック防止の問題は，従来より各方面から検討されている。たとえば参照する情報（ページ）を局所化しページングを多発しないプログラムの作成法，あるいは作成済のプログラムをページングを多発しないよう再構成する方法など，ジョブのページ参照特性そのものを最適化する研究はその代表的なもの[F2)H1)M2)]である。しかし現実のシステムで処理されるジョブの大部分に対して最適なページ参照特性を期待することはできない。したがって本論文ではジョブのページ参照特性は与件とし，資源割当ての運用によってページング・ネックを防止する方式について論ずる。一般にページング処理のオーバヘッドは，ページング実行頻度とページング実行時間の２つの観点からとらえることができる。前者は実メモリ管理方式，後者は二次メモリ管理方式にそれぞれ依存する。

したがって本論文の第２の目的は，ページング・ネック防止の問題を実メモリ管理の面から扱うことであり，ジョブに対する実メモリ割当て法について検討を加える[N10)N11)]。多重プログラミング・システムにおいて，各ジョブに割り当てられる実メモリ量が過大の場合は多重度（実メモリ内に収容されるジョブ数）が減じ，ＣＰＵ，入出力装置などの遊休時間が増大する。一方これが過小の場合は，実メモリ内に無いページの参照にともない，二次メモリと実メモリ間のページ転送すなわちページングの実行頻度が増大してページング・ネックとなり，やはり諸資源の遊休時間が増大する。したがって何らかの基準にもとづいて実メモリ割当て量の適正値を定める必要がある。従来より，理論的検討においてはＳＴＰ（Space Time Product）を評価基準とすることが多い。ＳＴＰはジョブの占有実メモリ量と処理時間の積であり，その最小化が目標とされる。しかし，実システムにおいては，処理されるジョブのページ参照特性が通常不明なためＳＴＰを算出できず，その最小化は困難であった。したがって従来のオペレーティング・システムでは，実メモリ割当て量決定の根拠は薄弱であった。例えばワーキングセット・ポリシー（各ジョブに対して過去ウインド・サイズτステップの間に参照した全てのページを実メモリ内に保つ方式）において，実メモリ割当て量を与えるウインド・サイズτは単に経験的に与えられていたに過ぎない。

本論文では，まず各ジョブのＳＴＰとＣＰＵ利用率との関係を論じ，ＳＴＰ最小化がシステム全体

の性能向上に寄与することを明らかにする。ついで，多重プログラミング環境における各ジョブに対し，各々のＳＴＰを推定し，これを最小化するよう実メモリ割当て量を定める「ワーキングセット最適化方式ＯＷＥ（Optimum Working-set Estimator）」を提示する。ジョブ実行中にそのプログラムのページ参照特性を学習し，ワーキングセット・ポリシーのウインド・サイズτの調節によりＳＴＰを削減するのがＯＷＥの特徴である。ＯＷＥを実現し，そのページ参照特性学習能力とＳＴＰ最小化達成可能性を検討する。またＯＷＥによる処理能力向上効果，オーバヘッド等について実測を行なう。なお，ＯＷＥは日立のオペレーティング・システムＶＯＳ３において実用化が予定されている。

本論文の第３の目的は，既に提示した第１の問題（ページング・ネック防止の問題）を二次メモリ管理の面から扱うことである[N7)N8)]。すなわちアドレス空間上のページの二次メモリへの蓄積法について検討することである。ページング・ネック防止のためには，ページングの実行時間を短縮することが有効である。二次メモリ上のページ単位に区分されたメモリ・スペースを「スロット」とよぶが，アドレス空間上のページに対する二次メモリ上のスロットの割当て方式はページング実行時間に多大な影響を与える。代表的な従来方式として，ジョブの実行を通じ終始同一のスロットを割り当てる「固定方式」と，ジョブ実行中のページ書き出しの際にその書き出し時間が最小のスロットを割り当てる「浮動方式」の２つがある。しかし従来，仮想メモリ・システムの検討は専ら実メモリ管理に関するものであり，これら両方式の得失など二次メモリ管理に関する報告は知られていない。

本論文では，まず固定方式と浮動方式とのページ入出力効率を理論的に比較し，両者の優劣がプログラムのページ参照特性に依存することを示す。あるページに割り当てられるスロットの存在する範囲ないし自由度に着目すると，固定方式では二次メモリ上の一点で自由度最小であり，浮動方式では二次メモリ全域に等しく自由度最大である。一般にページ書き出し時間，ページ読みこみ時間は，自由度の増加につれて各々減少，増加の傾向をもつ。したがって両者の和に着目すれば，最適な自由度が存在する。このことから，実用的な条件のもとで最適自由度に対応する「集中方式（Concentrating Technique）」を提案する。集中方式を実現し，その効果を実測により検証する。なお本方式は日立のオペレーティング・システムＶＯＳ３において実用化されている。

最後に本論文の第４の目的は，既に提示した第２の問題，すなわち多重プログラミング環境における各ジョブの資源割当て優先度の動的決定法につき考察することである[N5)N6)]。ページング・ネック防止方策を検討するという目的については既に提示したので，ここではページング・ネックにはならないと仮定して議論を進める。実際のシステムでは極力ページング・ネックを防止する方策がとられるべきなので，この仮定は妥当である。各種の資源使用特性をもつジョブが並行処理される多重プログラミング・システムにおいては，資源利用率や各ジョブへの資源供給量のようなシステムの状態量は一般に変動する。したがって処理能力，応答性の向上のためには，この状態量を常に観測し，これが目標値域内にあるよう資源割当て優先度を調節するフィードバック制御方式が有効である。近年，実メモリの割当てに関してはこのような方式が実用化されつつあるが，ＣＰＵや入出力装置の割当てには採

用されている例はない。このため，フィードバック制御の効果は，実メモリ利用率が高くその割当てに関して競合が生ずる場合に限られていた。

本論文では，フィードバック制御の手段として実メモリのみならずＣＰＵ，入出力装置など諸資源の割当て優先度調節を用いる「一般資源管理方式ＧＲＭ（General Resources Manager）」を提案する。ＧＲＭにおいては，各時点で，利用率が高く競合が生じている資源の割当て優先度を操作する。例えば，競合度の高い資源の割当てに際し，ある資源に遊休時間が生じている場合にはその資源を多用するジョブの優先度を上げる。このようにＧＲＭは，オペレーティング・システムで割当て優先度を操作できる資源の各々について，その割当て優先度の決定を互いに関連づけるのが特徴である。第２章で述べる漸近モデルは，ＧＲＭの制御効果を裏付けるものである。ＧＲＭを実現し，資源ごとに独立の制御を行なう従来方式と制御効果を実測により比較検討する。さらにシミュレーションによる検討をも行なう。なおＧＲＭは，既に日立のオペレーティング・システムＶＯＳ３にて実用化されている。

以上，本論文の４つの目的をあげ，それぞれの概要を述べた。これらは相互に無関係なものではなく，実用的でしかも理論的根拠のある資源管理の体系をみちびくという目的に到達する過程で現われるものである。

仮想メモリ方式の多重プログラミング・システムでは諸資源の有効な使用が可能であり，すぐれた処理能力，応答性が期待できることから，今後この種のシステムが広く用いられることは確実と考えられる。しかし，その性能向上効果は資源管理の方式に強く依存しており，場合によっては期待されたよりはるかに低い性能しか得られない危険もある。時々刻々変化する複雑なシステムの状態に応じて適切な資源管理を行なうために，各方面からの研究がなされなければならない。本論文の内容は無論これらの覆う範囲をつくしているものではないが，よりよい資源管理を行なうために実用価値の高いいくつかの知見を提供しているものと信ずる。

1.3　本研究と従来の諸研究との関係

多重プログラミング・システムの基礎技術を確立したオペレーティング・システムとして，オランダのEindhoven工科大学のＴＨＥ[D11]，デンマークのRegnecetralen社のＲＣ４０００[B8]などが知られている。しかし，商用の本格的な大型多重プログラミング・システムは，ＩＢＭ社のＯＳ／３６０[M5]ＭＦＴ，ＭＶＴにおいて出現したと考えられる。ただしＯＳ／３６０は基本的にバッチ・アプリケーションを対象にしており，ＴＳＳはＭＶＴにオプションとしてつけ加えられたに過ぎない。一方，英国Manchester大学のＡＴＬＡＳ[K5]において最初に用いられた仮想メモリ方式は，米国ＭＩＴのＭＵＬＴＩＣＳ[C7]，日立のＨ５０２０ＴＳＳ[M6]，ＩＢＭ社のＴＳＳ／３６０[L1]，ＣＰ－６７[R3]などの実験システムにおいて，ＴＳＳアプリケーションを対象に研究が行なわれた。これらのシステムは１９６０年代に開発されたものである。１９７０年代に到って，商用の多重プログラミング・システムへの仮想メモリ方式の採用が日立のＯＳ７[O3]，ＩＢＭ社のＳＶＳ[W1]などにおいて開始された。ただしＳＶＳはＯＳ／３６０

にもとづいており，依然としてバッチ・アプリケーションを主対象にしている。

筆者がその資源管理方式の設計，評価に携わった日立のＶＯＳ３[H6]は，ＩＢＭ社のＭＶＳ[S3]とともに，ＴＳＳとバッチ・アプリケーションの両者に統一的な方法で対処することを目的としている。ＶＯＳ３は日立のＭシリーズ計算機の最上位機種に用いられ，現時点で世界最大規模のオペレーティング・システムのひとつである。

資源管理の最適化の準備として，各種方式のもとにおけるシステムの大局的挙動を分析することが必要である。このためには理論的アプローチが適しており，従来より待行列理論による計算機システムの性能解析が数多く行なわれてきた。当初は主としてＣＰＵのみに着目したモデルがA.L.Scherr[S2], L.Kleinrock[K7], E.G.Coffman[C4]らにより扱われたが，近年は入出力装置なども含んだモデルとして，待行列を結合した待行列ネットワーク理論が多用されつつある。待行列の任意の結合をゆるし，１ジョブ・クラス，各待行列での待合せ処理方式（queueing discipline）はＦＣＦＳ（First Come First Served），サービス時間分布は指数分布，とした場合はW.J.Gordon[G1]らにより解析された。このような系では，システムの状態確率が各待行列の状態変数の積形（Product Form）として表わせる。解がProduct Formとなる問題はJ.R.Jackson[J1]により最初に扱われ，その後M.Reiser[R1], J.P.Buzen[B9]らの研究で実用的な解法がえられている。F.Baskett[B3]らはW.J.Gordonらの結果をさらに拡張し，任意に待行列を結合したネットワークで，複数ジョブ・クラス，各待行列での待合せ処理方式はＦＣＦＳ，ＬＣＦＳ（Last Come First Served），ＰＳ（Processor Sharing：量子無限小のラウンド・ロビン），ＩＳ（Infinite Servers：待ちなし）の４種類，サービス時間分布はクラス毎に異なるCox分布の場合についても，Product Formの解を得ることができると述べた（ただしＦＣＦＳのとき全クラス共通の指数サービス）。またA.S.Noetzel[N13]は解がProduct Formとなる待合せ処理方式を数学的に整理した。現段階ではF.Baskett, A.S.Noetzelらの研究が待行列ネットワーク理論の限界を与えている。

現実の資源管理は諸資源割当ての優先制御により行なわれるが，待行列ネットワーク内で待合せ処理方式として優先制御を用いたとき，状態確率はProduct Formにならず，厳密解は得られない。優先制御のもとでの待行列問題はW.Chang[C1]らにより研究されたが，これらは単一待行列系に関するものである。すなわち大半がＭ／Ｍ／１ないしＭ／Ｇ／１を仮定しており，たかだかＭ／Ｍ／ｓがR.H.Davis[D1]により扱われた程度である。これら固定優先度の問題とは別に，ラウンド・ロビン，多段フィードバック，ＳＲＰＴ（Shortest Remaining Processing Time First）などの動的優先度の問題もL.Kleinrock[K6], L.E.Schrage[S4]らにより扱われたが，やはりＭ／Ｍ／１ないしＭ／Ｇ／１を前提としている。単一待行列系では諸資源割当てが相互に与える影響を分析することができない。待行列ネットワーク内で優先制御を行なう系の近似的な扱いはいくつかなされている。K.C.Sevcik[S6]は，ＣＰＵのみ２レベルの優先度，D.Neuse[N1]らはＣＰＵおよび入出力装置に複数レベルの優先度をもつ場合につき，実メモリの待ちなし（実メモリ量無限大）という条件のもとで近似解析手法を報告した。一方Y.Bard[B2]は実メモリに関する優先制御の効果を近似解析したが，ＣＰＵと入出力

装置は優先制御の対象外とした。

筆者は[N9)N12)]，実メモリ，ＣＰＵ，入出力装置の各待行列で優先制御が行なわれ，複数ジョブ・クラスの各々についてその優先度は待行列ごとに異なってよいという，極めて一般的な場合を扱った「漸近モデル（Asymptotic Model）」をみちびいた。本モデルの特徴は，ボトルネックとなる資源での割当て待ちに着目し，資源の利用率やジョブの平均応答時間の厳密解を，資源サービス時間平均値の関数であるその漸近解で近似する点にある。単一ジョブ・クラス，優先制御なし，実メモリ量無限大の場合についての漸近解の解析は，R.R.Muntz[M8)M7)]，L.Kleinrock[K8)]らにより行なわれており，筆者のモデルはこのボトルネック解析の概念を拡張したものである。なお，筆者のアプローチとは異なるが，実メモリでの待ちの評価法として，P.J.Courtois[C8)]により論じられたDecompositionにもとづく近似法がある。これは，ＣＰＵや入出力装置と実メモリとを別のレベルで解析するものであり，広く用いられるので付記しておく。

仮想メモリ方式に関しては各方面からの研究がなされてきた[S10)]。特に実メモリ内に保持するページの選択法すなわちページ・リプレースメント・アルゴリズムについては，L.A.Belady[B4)]，E.G.Coffman[C5)]らにより研究が開始され，その後理論，実験，両面で多くの報告がある。しかし，その大半は単一プログラミング環境に関するものである。近年，T.Masuda[M3)]は多重プログラミング環境におけるページ・リプレースメント・アルゴリズムの比較を行ない，P.J.Denning[D3)D5)]の提案した「ワーキングセット・ポリシー」の優位を論じた。これは，各ジョブに関してそれが過去ウインド・サイズτステップの間に参照したページを実メモリ内に保つ方式である。ワーキングセット・ポリシーはプログラムの局所参照性という性質に着目した点で優れた方式であるが，その有効性はウインド・サイズτの設定値に依存する。

ＳＴＰ（Space Time Product）は実メモリ割当て法の評価基準としてL.A.Belady[B5)]，W.W.Chu[C3)]，B.G.Prieve[P2)]，G.Henderson[H3)]らにより用いられている。さらにP.J.Denning[D7)D8)]らはＳＴＰ最小化による処理能力向上効果につき述べた。その他，ワーキングセット・ポリシーのウインド・サイズτの最適設定については，P.J.Denning[D3)]，B.G.Prieve[P1)]がページの平均参照間隔が与えられたとして理論的扱いを行ない，J.Rodriguez-Rosell[R2)]，G.S.Graham[G2)]らは，実際のプログラムのアドレス軌跡にもとづく実験的扱いを行なった。しかし，これらの報告は全てプログラムのページ参照特性が既知の場合の議論である。現実のオペレーティング・システムの動作環境では一般にこれは未知であるため，上記議論はＳＴＰを最小化する手法を与えるものではない。既にワーキングセット・ポリシーはＩＢＭ社のＭＶＳ[S3)C2)]，仏国Grenoble大学で改造したＣＰ－６７[R3)]などで実現されているが，ウインド・サイズτの値はＳＴＰと無関係に経験的に定められる。

筆者は[N10)N11)]，プログラム実行中にそのページ参照特性を学習することにより，オペレーティング・システムの実際の動作環境で各ジョブのＳＴＰを推定し，これを最小化するτを与える「ワーキングセット最適化方式ＯＷＥ（Optimum Working-set Estimator）」を提案し，実験を行なった。なお，筆者の提案方式と直接の関連は無いが，ワーキングセット・ポリシーの概念にもとづく修正方式もい

くつか提案されているので付記する。W.W.Chu[C3]，H.Opderbeck[O6]は，ワーキングセット・ポリシーより実現の容易なＰＦＦ（Page Fault Frequency）という方式を提案している。また，A.J.Smith[S9]は，ワーキングセット・ポリシーにおいてページ参照特性の変化に対する追従性を良くした方式を述べた。

仮想メモリ方式についての報告はほとんど実メモリ管理に関するものである。二次メモリ管理については，ＩＢＭ社のＯＳ／ＶＳ１[I1]，日立のＶＯＳ２[H5]などではジョブの実行を通じて各ページに同一のスロット（二次メモリ上のページ単位に区分されたメモリ・スペース）を割り当てる「固定方式」が用いられる。また，ＯＳ／ＶＳ２[I1]ではページ書き出し時点で書き出しオーバヘッドの小さいスロットを割り当てる「浮動方式」が用いられる。従来方式としてこの二つが代表的だが，ＭＵＬＴＩＣＳ[C7]，Ｈ５０２０ＴＳＳ[M6]などでは，ページ書き出し時点で特に入出力効率の配慮をすることなく空いたスロットを順次割り当てる方式がとられた。

筆者は[N7]，固定，浮動両方式の効率を比較し，優劣がプログラムのページ参照特性に依存することを示した。ページ参照のモデルは各種提案されているが，ここではJ.R.Spirn[S11]らにより有効性が検証されたＶＳＬＭ（Very Simple Locality Model）を用いた。筆者[N8]はさらに，実用的条件のもとでページ参照特性によらず既存の二方式より効率がよいと考えられる「集中方式（Concentrating Technique）」を提案し，実験を行なった。従来，磁気ドラム，磁気ディスクなどの入出力動作の解析はP.J.Denning[D2]，T.J.Teory[T2][T3]らによりなされているが，二次メモリとしてプログラム動作と関連づけた扱いはなされていない。

多重プログラミング・システムの性能を上げるための資源管理方式は種々提案されてきたが，大半はシステムの一部分の状態にもとづく制御であり，その効果も局所的最適化にとどまるものであった。この例として，K.D.Ryder[R5]により提案されたダイナミック・ディスパッチング制御，L.Kleinrock[K9]らにより研究された各種のタイム・スライシング制御などがある。近年，システム全体の状態にもとづいて性能の大局的な最適化を行なう総合的資源管理方式がいくつか提案された。大須賀の研究[O4]はこの種の方式として先駆的なものである。H.Kameda[K1][K2]は経済学の需要と供給の均衡にもとづく方式を提案した。A.J.Bernstein[B6]，J.C.Sharp[S7]らは，サービス目標関数にもとづくＰＤＳ（Policy Driven Scheduler）とよぶ方式を実験した。ＰＤＳにおいては，各ジョブへの資源のサービス供給量（割り当て量）を時々刻々検知し，これと目標値との差にもとづくフィードバック制御を行なう。さらにM.Ruschitzka[R4]らはＰＤＳをより一般化した方式を提案した。このような資源管理方式の中で実用化された最初のものはＩＢＭ社のＭＶＳにおけるＳＲＭ[L3]（System Resources Manager）である。ＳＲＭでは，各ジョブへの資源サービス供給量に加え諸資源の利用率もフィードバック制御の対象となっており，応答性と処理能力の均衡的向上が目標として明示された。しかし，ＳＲＭが関与するのは実メモリ割当てのみであり，ＣＰＵや入出力装置などの割当ては従来と同様に局所的管理にゆだねられている。

筆者の提案した[N5][N6]「一般資源管理方式ＧＲＭ（General Resources Manager）」は，ＳＲＭをさ

らに一般化したものである。ＧＲＭは，実メモリ，ＣＰＵ，入出力装置などを一般資源としてとらえ，処理能力と応答性の均衡的向上を目標として，これら諸資源の優先制御を行なう。大局的な性能向上のために，オペレーティング・システムで割当て優先度を操作可能な諸資源についてその割当てを相互に関連づけたことが，ＧＲＭの特徴である。なお，ＳＲＭ[L3]やダイナミック・ディスパッチング制御[R5]，タイム・スライシング制御[K9]は，ＧＲＭに概念的に含まれる。

第２章　多重プログラミング・システムの性能解析モデル

2. 多重プログラミング・システムの性能解析モデル

2.1 概　　要

多重プログラミング・システムにおいては，複数のジョブが実メモリ，ＣＰＵ，入出力装置などの諸資源を競合的に利用しつつ，並行的に処理される。これらのジョブへの資源割当ての方式がシステム性能に影響する。したがって資源割当ての優先度を適切に操作することにより，システム性能の向上が期待できる。このためには，優先制御が性能に与える影響を定量的に把握する必要があり，システムの特性をモデル化し，解析する何らかの手法が不可欠となる。しかるにこの問題の複雑さのため，従来の待行列ネットワーク理論によっては十分な解析は不可能である。またシミュレーションや実測では，システムの大局的挙動を知ることが難しいといった困難があった。

そこで本章では，優先制御のもとでのシステムの大局的挙動を簡便に求めるための解析モデルをみちびく。[N9)N12)]このモデルは，第３～第５章で提案する資源管理方式の効果に理論的根拠を与えるものである。本章では2.2節において，各種優先制御のもとで資源の利用率やジョブの平均応答時間を与えるモデルを提示する。本モデルは厳密解をその漸近解で近似するものであり，「漸近モデル（Asymptotic Model）」とよぶ。2.3節では，２ジョブ・クラスの場合について，様々な優先度のもとでの性能の解析例を示す。また2.4節では，シミュレーション結果および実測結果と漸近モデルの解析結果とを比較検討し，漸近モデルの検証を行なう。

2.2 漸近モデル（ Asymptotic Model ）

2.2.1 システム・モデル

システムの特性を理解し，システムの各部間の定量的関係をみちびくために，数学的モデル化を行なう。システムはアクティブ端末数が有限の「有限多重度の有限呼源モデル（ finite population model with limited degree of multiprogramming ）」で表わされるとし，その定常状態において解析する。資源の要求単位を「トランザクション」とよぶ。図2.1で，各端末は思考時間が終わるごとにトランザクションを発生する。具体的には，トランザクションはＴＳＳコマンドやバッチ・ジョブに対応する。ただしバッチ・ジョブについては思考時間ゼロの端末から発生するとみなす。

図2.1は仮想メモリ・システムを想定しており，入出力装置の一部はページ入出力装置（二次記憶装置）として用いられる。実メモリ容量をV，実メモリ内のトランザクションの実メモリ割当て量をw_k（$k=1,2,\cdots$），最大多重度をsとすると$\sum_{k=1}^{s} w_k \leq V$が必要であるが，いまシステムの平均的特性に着目し，トランザクションの平均実メモリ割当て量をwとすると次式がなりたつ。

$$s = V / w \qquad (2.1)$$

思考中以外の端末数がsを越えると，越えた分だけのトランザクションは実メモリに入りきれず，二次メモリに格納されて実メモリ割当て待ちとなる。簡単のため，Vはsの倍数と仮定する。

優先制御は，実メモリ，ＣＰＵ，入出力装置の割当てに関して行なわれるが，トランザクション間の優先度は時間的に不変とする。すなわち固定優先度を扱う。

2.2.2　単一クラス・モデル

トランザクションは，ＴＳＳ，バッチなどいくつかのグループに分けることができる。このグループを「クラス」とよび，ひとつのクラスの中では資源割当て優先度は等しく，資源使用量も一様と仮定する。以下，各クラスにおける平均的性能について考察するものとする。

優先度をもつ複数クラス・モデルをみちびく準備として，まず優先度のない単一クラス・モデルにつき述べる。平均応答時間，平均実メモリ占有時間，平均思考時間をそれぞれT，R，zで表わす。TはRに実メモリ割当て待ち時間を加えた値に等しい。本モデルにおいてシステムの動作解析を行なうためには，ＣＰＵと入出力装置を区別する必要はないので，これらをともに「プロセッサ」とよぶ。ひとつのトランザクションによるプロセッサjの平均使用時間をd_jとすると，トランザクション間の競合がなく，したがって待ち合わせが生じない場合の平均応答時間R_0は，

$$R_0 = \sum_j d_j \qquad (2.2)$$

で与えられる。R_0はRの下限値を表わしている（ただし同一トランザクションが複数プロセッサを同時に使用することはないとする）。d_jはトランザクションの資源使用特性から定まる定数である。ただしここで，入出力処理のうちページ入出力処理についてはＣＰＵによる情報（ページ）参照処理の延長という性格のため，ページ入出力装置の平均使用時間d_{PG}は平均実メモリ割当て量wの関数となる。この関数形は実メモリ管理方式，二次メモリ管理方式などに依存するが，一般に図2.2に示したようにwの単調減少関数となることが知られている。[D4)D6)]

プロセッサの利用率u_j（$j=1,2,\cdots$），実メモリの利用率u_mと平均応答時間T，平均実メモリ占有時間R，平均思考時間zとの関係について考える。いま，ある端末に着目すると，これは定常状態で平均的に（$T+z$）の間にd_jだけプロセッサjを用いる。したがってこの端末によるプロセッサjの利用率は$d_j/(T+z)$である。また実メモリについては，この端末は（$T+z$）の間に平均的にRだけ実メモリを占有し，実メモリは最大多重度s個までトランザクションを収容できるから，この端末による実メモリの利用率は$(R/s)/(T+z)$である。以上より，システムのアクティブ端末数すなわち呼源数をNとすると，次式が得られる。

$$\begin{cases} u_j = Nd_j/(T+z) & (j=1,2,\cdots) \\ u_m = N(R/s)/(T+z) \end{cases} \qquad (2.3)$$

式（2.3）は，諸資源における割当て要求の到着時間間隔やサービス時間の分布形によらず成立するグローバルな関係である。$\max_j d_j$をd_xとすると，式（2.3）よりNによらず次式が成り立つ。

$$\max_j u_j = u_x \qquad (2.4)$$

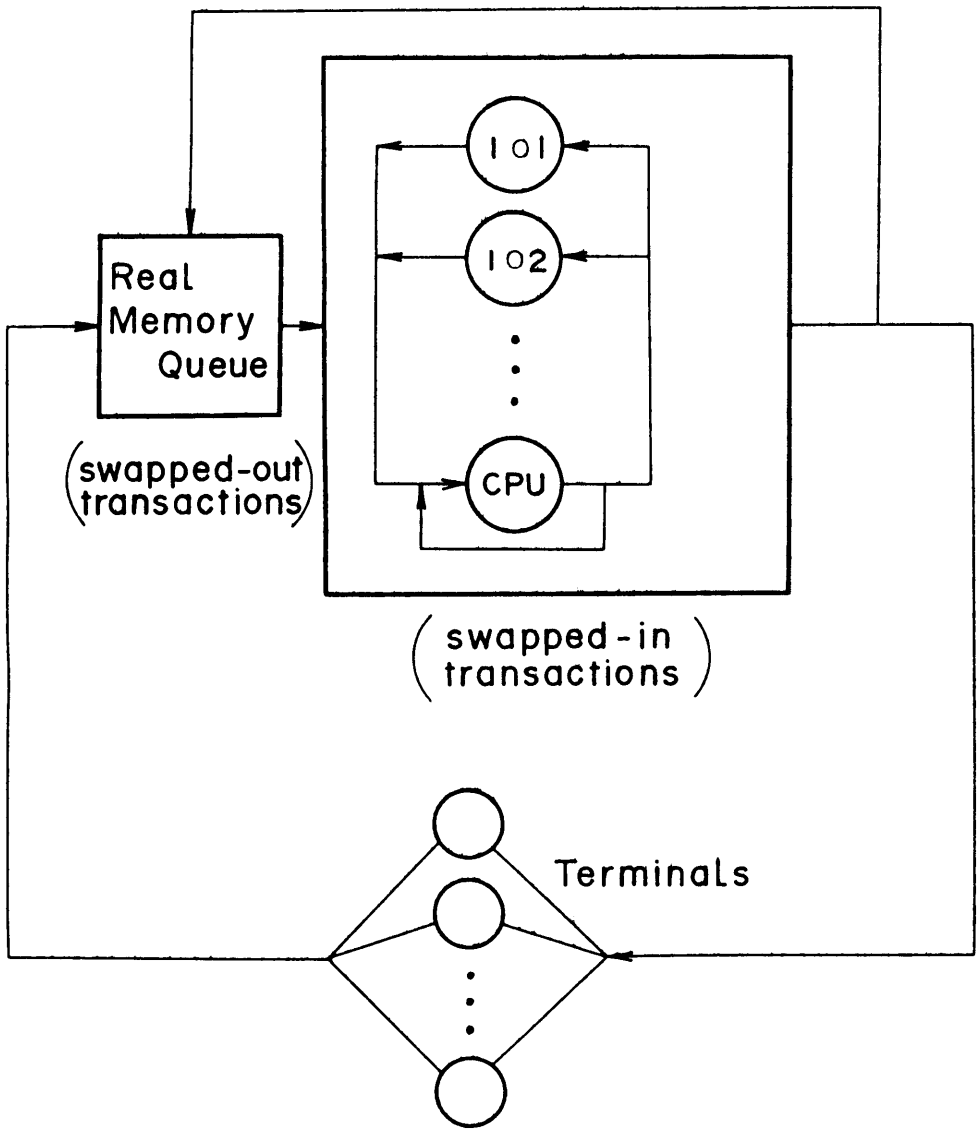


図 2.1　有限多重度の有限呼源モデル

Finite population model with limited degree of multiprogramming

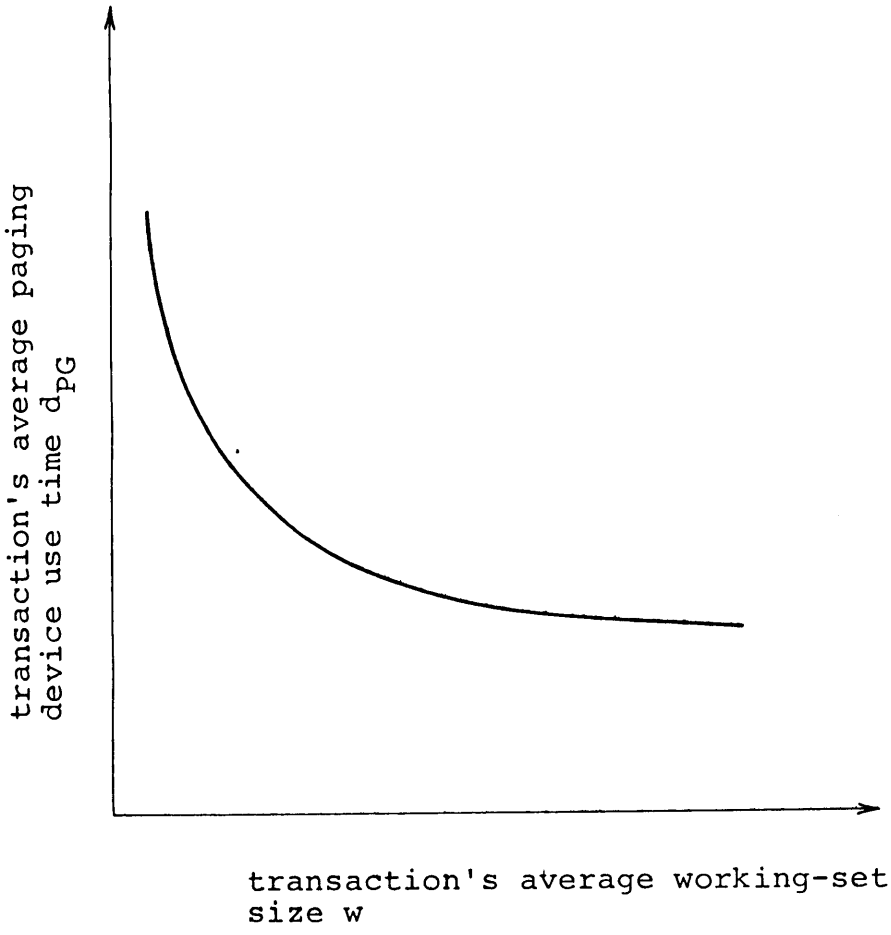


図2.2　ページ入出力装置使用時間の実メモリ割当て量による変化

Variation of paging device use time with working-set size w

ボトルネック解析は，L.Kleinrock[K8]，R.R.Muntz[M7)M8]，P.J.Denning[D10]らにより開始された。その特徴は，利用率の高いボトルネックとなる資源に着目し，このような資源の利用率を１００％に等しいとみなすことにより漸近解を得る点にある。すなわちボトルネック解析の基本的な仮定は次の通りである。

仮定１：「トランザクションからある資源へのサービス要求が出されたとき，システム全体として当資源を１００％使用するに足るだけのサービス要求がない場合は，当資源使用のための待ち時間はゼロとする。また，これ以外の場合すなわちサービス要求の総量が当資源の能力を超える場合は，当資源は１００％使用されるとする。」

本仮定から平均応答時間Tの漸近解を求めることができる。ここで，漸近解は到着時間間隔やサービス時間の分布形によらず生ずる最小限の資源割当て待ち時間を含んでいる。実際には資源利用率が１００％未満の場合にももちろん割当て待ちは発生する。また，サービス要求の総量が資源の能力を超える場合でも，資源利用率は必ずしも１００％とはならない。この差異は，後述するような誤差となって現われる。ただし，漸近解の特長は性能値の大局的な把握を可能にする点にある。現実の評価の場においては，系の確率的特性を表わす分布形自体には必ずしも再現性があるとは限らないため，性能の絶対値よりもその構造的性質（例えば応答性が急激に低下する負荷条件など）を求めることが必要な場合が多い。漸近解は性能の構造的性質を解析する手がかりを与える。

最大多重度sが無限大（実メモリ容量Vが無限大）のとき，式（2.3）と仮定１から，よく知られた次の漸近解が得られる。図2.3を参照されたい。

$$T = \begin{cases} d_x N - z & : \quad (R_0 + z) / d_x \leq N \quad \text{の場合} \\ R_0 & : \quad (R_0 + z) / d_x > N \quad \text{の場合} \end{cases} \qquad (2.5)$$

sが有限のとき，実メモリもボトルネックになりうる。このときの漸近解は，従来の議論の拡張として次のようにみちびかれる。まず仮定１より，Nが小さく$u_x < 1$なる場合にはRはR_0と一致する。また$u_x = 1$となる場合は，仮定１と式（2.3）より，

$$T = d_x N - z \qquad (2.6)$$

となる。同様に，Nが小さく$u_m < 1$なる場合にはTはRと一致し，$u_m = 1$となる場合には次式がなりたつ。

$$T = (R / s) N - z \qquad (2.7)$$

以上をまとめて，sが有限のときの漸近解は式（2.8）で与えられる。

$$T = \begin{cases} d_x N - z & : \quad u_x = 1 \text{となりうる} N \\ (R_0 / s) N - z & : \quad u_x < 1 \text{かつ} u_m = 1 \text{となりうる} N \\ R_0 & : \quad \text{上記以外の} N \end{cases} \qquad (2.8)$$

Nとsの関数としてのTを図2.4に示す。図2.4において，sの値によりグラフは2つに分かれる。いま$0 \leq s < R_0 / d_x$であれば，システムは「メモリ・ネック」になるが，決して「プロセッサx・ネックかつメモリ・ネック」にはならない（以下，$u_x = 1$のとき「プロセッサx・ネック」，$u_m = 1$のとき「メモリ・ネック」とよぶ）。これは，メモリ・ネックの状態すなわち$N \geq (R_0 + z)\, s / R_0$においては式（2.3），（2.4），（2.6），（2.7）より次式が成り立つためである。

$$u_x = s\, d_x / R_0 < 1 \qquad (2.9)$$

一方，$s \geq R_0 / d_x$ならば，Nを増すにつれてシステムはまず「プロセッサx・ネック」になりついで「プロセッサx・ネックかつメモリ・ネック」となる。これは，プロセッサx・ネックの状態すなわち$N \geq (R_0 + z) / d_x$においては，式（2.3），（2.4），（2.6），（2.7）より次式が成り立つためである。

$$u_m = \{ N - (z / d_x) \} / s \qquad (2.10)$$

Nを増していき$\{ s + (z / d_x) \}$に近づくにしたがって，式（2.10）は1に近づく。なお，図2.4において，sが十分大なるところでs＝一定の面で切った断面が図2.3である。

図2.4より，平均実メモリ割当て量wおよびページ入出力装置の平均使用時間d_{PG}がシステムの性能に与える影響を知ることができる。wを過大に定めると，式（2.1）よりsが減少し，「メモリ・ネック」となってTが増加する。一方，図2.2に示したように，wを0に近づけるとページングが多発するページング・ネック状態[D4]となるのでd_{PG}は急激に増大する。したがってwを過小に定めると，

$$d_x = d_{PG} \qquad (2.11)$$

となり，また式（2.2）よりR_0も増大するので，やはりTは増加する。これから，wを適正に与えることが性能向上の必要条件であることがわかる。wの決定法に関する問題は第3章で扱う。また，ページ入出力実行時間の短縮により関数$d_{PG}(w)$全体を減少させることができれば，ページング・ネックに陥ることなくsを増加することが可能となり，性能向上が期待できる。この問題は第4章で論ずる。

2.2.3 複数クラス・モデル

複数クラス，優先度ありの漸近モデルを以下にみちびく。ここでクラスの分け方は与件とする。クラスiのトランザクションに関する全ての変数を，T_i，R_i，d_{ij}，R_{io}，z_i，s_i，N_iなどのように添字iであらわす。式（2.3）の拡張として次式が得られる。

$$\begin{cases} u_j = \sum_i N_i\, d_{ij} / (T_i + z_i) & (j = 1, 2, \cdots) \\ u_m = \sum_i N_i\, (R_i / s_i) / (T_i + z_i) \end{cases} \qquad (2.12)$$

優先度のある複数クラスの場合の漸近解を求めるため，仮定1に加え次の仮定を設ける。

仮定2：「優先制御は割りこみ優先権出直し規準（preemptive resume rule）にしたがうとする。」

したがってサービス実行中により高い優先度のトランザクションの割当て要求が発生すると，その時点でサービスは中断され，資源は高優先度トランザクションへ割当て変更される。仮定2より，高優

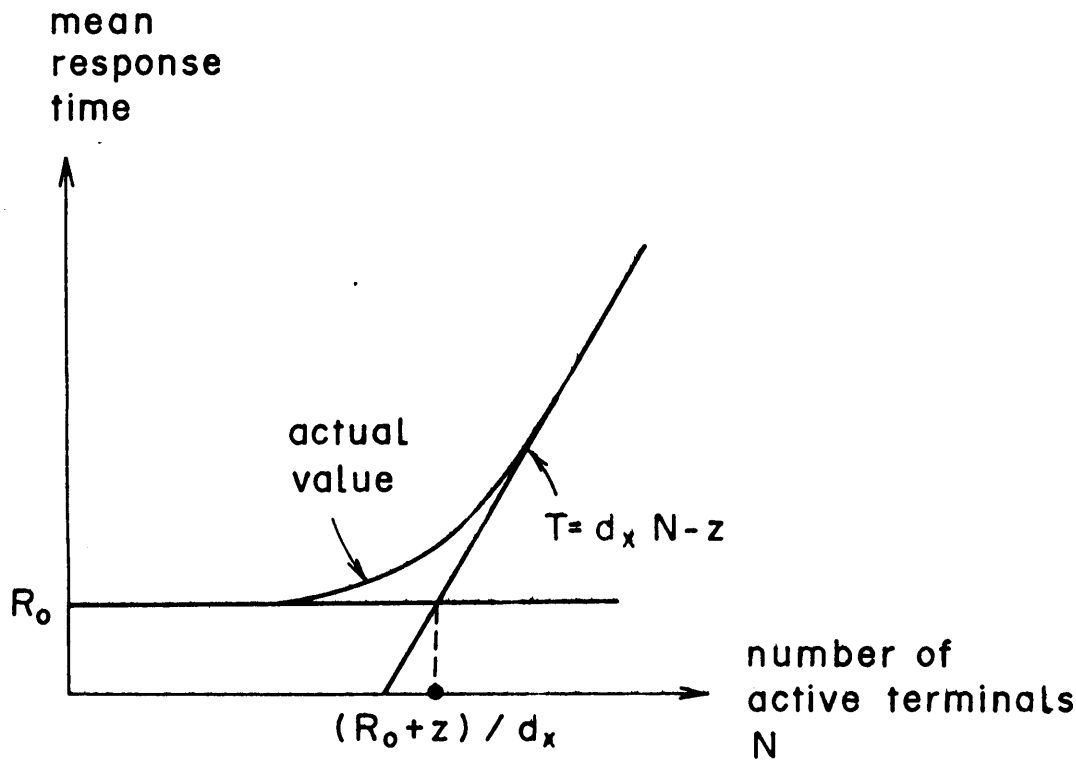


図 2.3 無限多重度の場合の平均応答時間Tの漸近解

The asymptotes of mean response time T
with unlimited degree of multiprogramming

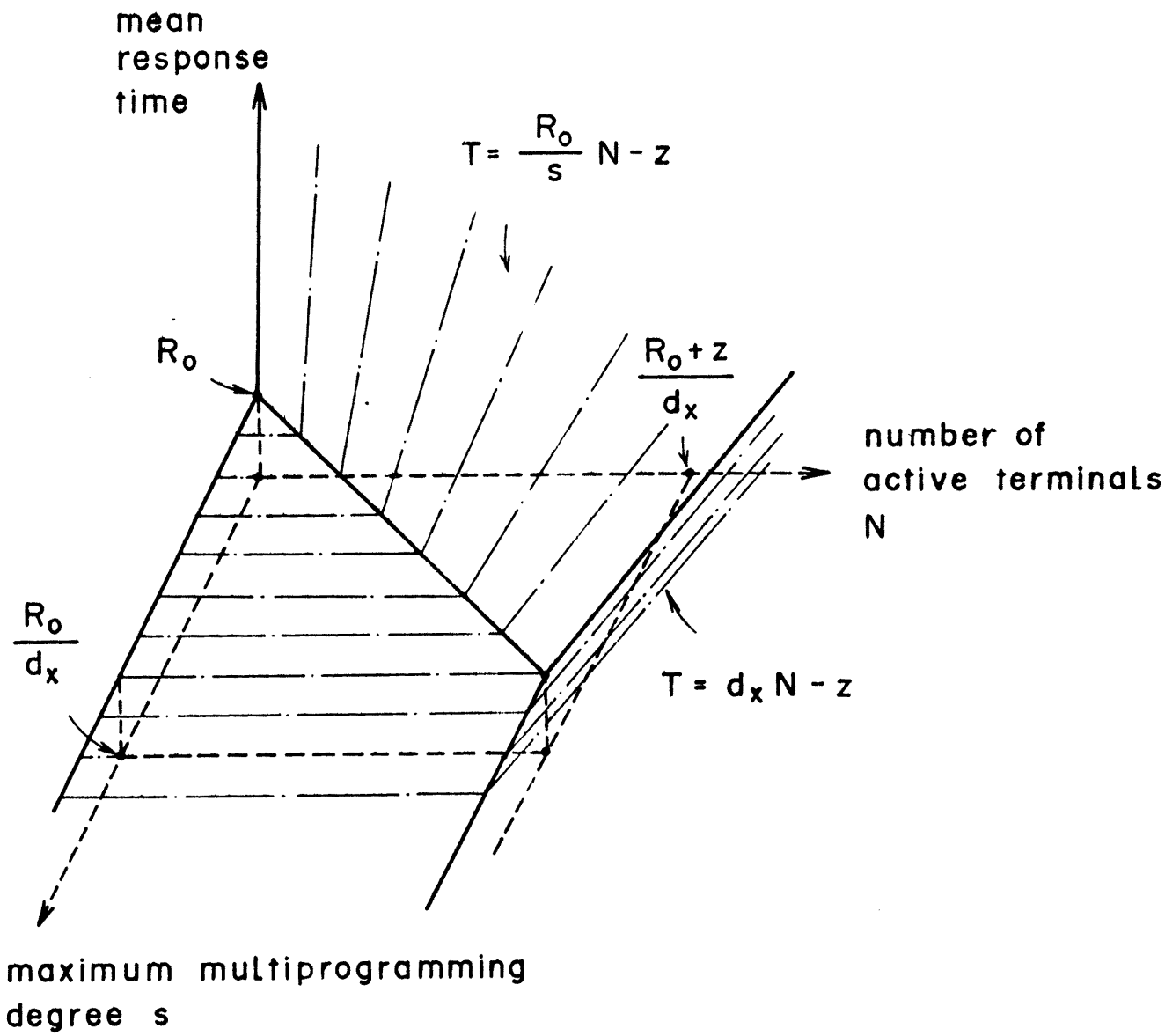


図 2.4　有限多重度の場合の平均応答時間 T の漸近解

The asymptotes of mean response time T
with limited degree of multiprogramming

先度トランザクションは低優先度トランザクションによる影響は受けない。低優先度トランザクションには高優先度トランザクションの使用した残りの資源が割り当てられる。

実際のシステムでは入出力処理はサービスを中断されないので，入出力装置はpreemptiveではない。また入出力実行中のトランザクションはその完了までスワップ・アウトできないので，実メモリもnon-preemptiveな場合がある。 したがって実際の値は得られた解から非優先制御の場合の解にやや近づく傾向をもつ。

全てのクラスについて有限のT_iを保証するため，単一クラスの場合と異なり，解析の対象とするN（$=\sum_i N_i$）の範囲は限定される。換言すると，高優先度トランザクションの発生数は，いずれかの資源をそれのみで占有するほど大ではないとする。これは，ある資源が高優先度トランザクションで占有されると，低優先度トランザクションの平均応答時間は無限大となるためである。（Nの限定により議論の一般性は失なわれない。T_iが無限大のとき，他のクラスの平均応答時間は次にのべる方程式の中で第iクラスの項を無視することにより得られる。）上記のようにNの範囲を限定したとき，仮定1と仮定2より，この場合の漸近解を求めるに際し，各資源の最低優先度クラス・トランザクションの割当て待ちのみを評価すれば十分である。

ボトルネック資源の最低優先度クラス・トランザクションに対するサービス処理方式(queueing discipline)により，漸近解は異なる（これ以外のトランザクションに対しては，いかなるサービス処理方式でも漸近解は同一である）。ここで，次の2つのモデルを仮定する。

Ⅰ. ボトルネック資源の最低優先度クラスの各トランザクションに対し「単位時間あたりの資源割当て量が等しい」とする（フェア・シェア・サービス）。

Ⅱ. ボトルネック資源の最低優先度クラスの各トランザクションに対し「資源1回使用あたりの平均待ち時間が等しい」とする。

前者，後者を仮定するモデルをそれぞれＡＭⅠ（Asymptotic Model Ⅰ），ＡＭⅡ（Asymptotic Model Ⅱ）とよぶ。これらは現実のオペレーティング・システムの異なった制御方式に対応する。すなわちオペレーティング・システムが各トランザクションの資源使用状況を刻々検知してフィードバック制御を行なう場合[N5)N6)]にはＡＭⅠ，等優先度トランザクションにＦＣＦＳ（First Come First Served）で資源を割り当てる場合にはＡＭⅡが対応する。

以下，漸近モデルＡＭⅠ，ＡＭⅡの方程式を提示する。

(1) ＡＭⅠ（Asymptotic Model Ⅰ）

まず実メモリ容量Vが無限大の場合について述べる。プロセッサjを利用するトランザクション・クラスのうち，jの割当て優先度が最低であるようなトランザクション・クラスの集合をy_jとする。また，利用率が1であるプロセッサ（ボトルネック・プロセッサ）の集合をHとする。Hが空集合のとき，すべてのiについてT_iはR_{i0}に等しい。これ以外の場合には，T_iは次式により与えられる。

$$\sum_i \{(1-e_i)N_i\, d_{ij} / (R_{io}+z_i) + e_i N_i\, d_{ij} / (T_i+z_i)\} = 1 \quad (j \in H) \qquad (2.13)$$

$$d_{i_1 j} / (T_{i_1}+z_{i_1}) = d_{i_2 j} / (T_{i_2}+z_{i_2}) = \cdots\cdots \qquad (2.14)$$

$$(i_1, i_2, \cdots\cdots \in y_j \ ; \ j \in H)$$

ただし，

$$e_i = \begin{cases} 1 & : \ {}^{\exists}{}_{k \in H} \ \text{such that} \ i \in y_k \ \text{（すなわち，利用率＝1なるプロセッサ} \ k \\ & \quad \text{を使用し優先度最低のとき）} \\ 0 & : \ \text{上記以外} \end{cases} \qquad (2.15)$$

とする。式（2.13）は，式（2.12）の第1式において，ボトルネック・プロセッサの最低優先度クラス以外のT_iをR_{io}でおきかえることにより得られる。式（2.14）は，ボトルネック資源がその最低優先度クラスのトランザクションに均等に分配されることを表わしている。未知数T_iの数はT_iの数は$|\bigcup_{j \in H} y_j|$である（以下，$|\ |$は集合の要素の数，$\cup$は和集合を示す）。式（2.13），（2.14）の数はそれぞれ$|H|$，$\sum_{j \in H}(|y_j|-1)$である。したがって解を得るためには，次式が成立しなければならない。

$$\left|\bigcup_{j \in H} y_j\right| = \sum_{j \in H} |y_j| \qquad (2.16)$$

式（2.16）はy_jが互いに排反集合であることを意味する。なお，式（2.16）が成立すれば，式（2.13），（2.14）は独立な線形方程式に帰着することができ，解を得ることができる（証明は付録7）。ただし，得られたT_iは次式をみたさねばならない。

$$\begin{cases} 0 \le u_j \le 1 & (j = 1, 2, \cdots\cdots) \\ R_{io} \le T_i & (i = 1, 2, \cdots\cdots) \end{cases} \qquad (2.17)$$

いいかえると，N_i（$i = 1, 2, \cdots\cdots$）が与えられたとき，Hは式（2.12）～（2.15），（2.17）より定まる。

次に実メモリ容量Vが有限すなわち最大多重度s_iが有限の場合を考える。このとき，実メモリもボトルネックになりうる。もしメモリ・ネックでなければ，T_iは式（2.13）～（2.15）より与えられる。したがって以下メモリ・ネックすなわち$u_m = 1$と仮定し，実メモリ割当ての最低トランザクション・クラスの集合をy_mと表わす。

実メモリ容量が有限の場合の方程式は，式（2.13）～（2.15）の拡張として次のように与えられる。

$$\sum_i [(1-e_i)(1-e_{im})N_i\, d_{ij} / (R_{io}+z_i) + \{1-(1-e_i)(1-e_{im})\}N_i\, d_{ij} / (T_i+z_i)] = 1 \qquad (2.18)$$

$$(j \in H)$$

$$d_{i_1 j} / (T_{i_1} + z_{i_1}) = d_{i_2 j} / (T_{i_2} + z_{i_2}) = \cdots\cdots \quad (i_1, i_2, \cdots\cdots \in y_j \ ; \ j \in H) \tag{2.19}$$

$$\sum_i [(1-e_i)(1-e_{im}) N_i (R_{io}/s_i)/(R_{io} + z_i) + \{e_i (1-e_{im})\} N_i (T_i / s_i)/(T_i + z_i) + \{(1-e_i) e_{im}\} N_i (R_{io}/s_i)/(T_i + z_i) + \{e_i e_{im}\} N_i (R_i / s_i)/(T_i + z_i)] = 1 \tag{2.20}$$

$$\{(1-e_{i_1})(R_{i_1 o}/s_{i_1}) + e_{i_1}(R_{i_1}/s_{i_1})\}/(T_{i_1} + z_{i_1}) = \{(1-e_{i_2})(R_{i_2 o}/s_{i_2}) + e_{i_2}(R_{i_2}/s_{i_2})\}/(T_{i_2} + z_{i_2}) = \cdots\cdots \quad (i_1, i_2, \cdots\cdots \in y_m) \tag{2.21}$$

ただし，

$$e_{im} = \begin{cases} 1 & : \ i \in y_m \\ 0 & : \ \text{上記以外} \end{cases} \tag{2.22}$$

である。i が y_m に属すとき T_i が実メモリ割当て待ち時間を含むこと以外は，式（2.18），（2.19）は式（2.13），（2.14）と同一である。方程式 $u_m = 1$ は式（2.20）に対応するが，ここで仮定1および仮定2からみちびかれる，

$$T_i = R_i \quad (i \in y_m) \tag{2.23}$$

なる関数が用いられている。式（2.21）は，実メモリ割当て優先度が最低のトランザクションに実メモリを均等に分配することを意味する。未知数 T_i，R_i の数はそれぞれ $|(\bigcup_{j \in H} y_j) \cup y_m|$，$|(\bigcup_{j \in H} y_j) \cap y_m|$ であり，その和は $\{|\bigcup_{j \in H} y_j| + |y_m|\}$ に等しい。（ここで∩は積集合を表わす。）一方，式（2.18），（2.19），（2.20），（2.21）の数はそれぞれ $|H|$，$\sum_{j \in H}(|y_j| - 1)$，1，$(|y_m| - 1)$ であり，その和は $(|y_m| + \sum_{j \in H} |y_j|)$ に等しい。したがって，上記方程式を解くためには式（2.16）が必要である。さらに，式（2.16）のもとで，式（2.18）～（2.21）も式（2.13）～（2.14）と同様に独立な線形方程式に帰着することができ，解が得られる（証明は付録7）。ただし，解は式（2.17）に加えて次の条件をみたさねばならない。

$$R_{io} \leq R_i \leq T_i \qquad (i = 1, 2, \cdots\cdots) \tag{2.24}$$

(2) AMⅡ（Asymptotic Model Ⅱ）

まず実メモリ容量 V が無限大の場合について述べる。AMⅠと同様に，ボトルネック・プロセッサの集合 H が空集合ならば，全ての i について T_i は R_{io} に等しい。これ以外の場合には，T_i は次式により与えられる。

$$\begin{cases} T_i = R_{io} + \sum_j \delta_{ij}\,\theta_{ij}\,\psi_j \\ \delta_{ij} = \begin{cases} 1 & : \quad i \in y_j \text{かつ} \ j \in H \\ 0 & : \quad \text{上記以外} \end{cases} \end{cases} \qquad (2.25)$$

$$\sum_i d_{ij} N_i \,/\, (R_{io} + \sum_k \delta_{ik}\,\theta_{ik}\,\psi_k + z_i \ ) = 1 \qquad (j \in H) \qquad (2.26)$$

ψ_j は，ボトルネック・プロセッサ j を1回使用するための平均待ち時間である。θ_{ij} は i クラスのトランザクションによる j の平均使用回数を表わす。（式（2.26）では，j のかわりに k を用いた。）ボトルネック資源の最低優先度トランザクションについて，「資源1回使用あたりの平均待ち時間が等しい」という仮定から式（2.25）が得られる。また式（2.26）は，式（2.25）を式（2.12）の第1式に用いたものである。

式（2.26）より ψ_k，さらに式（2.25）より T_i を求める。ここで，ψ_k が求まるための条件につき考察する。いま，ボトルネック・プロセッサの集合 H の要素数を $|H|$ とし，プロセッサ j（$j = 1, 2, \cdots\cdots, |H|$）がボトルネックとする。またトランザクション・クラスの数を ℓ とする。いま，第（i，j）行列要素が $\delta_{ij}\ \theta_{ij}$ であるような ℓ 行 $|H|$ 列の行列 $\boldsymbol{D}$ を考えるとき，式（2.26）より未知数 ψ_k（$k = 1, 2, \cdots\cdots, |H|$）が求まるためには，次式が必要であり十分でもある*（証明は付録8）。

$$\mathrm{rank}(\boldsymbol{D}) = |H| \qquad (2.27)$$

なお得られた解は式（2.17）をみたさねばならない。ここで式（2.17）の第2式は $\psi_k \geq 0$ に対応する。

次に実メモリ容量 V が有限すなわち最大多重度 s_i が有限の場合について述べる。実メモリがボトルネックでなければ T_i は式（2.25），（2.26）より求まるので，以下メモリ・ネックすなわち $u_m = 1$ と仮定する。このとき，式（2.25），（2.26）の拡張として次式が成立する。

$$\begin{cases} T_i = R_{io} + \sum_j \delta_{ij}\,\theta_{ij}\,\psi_j + \delta_{im}\,\theta_{im}\,\psi_m \\ \delta_{ij} = \begin{cases} 1 & : \quad i \in y_j \ \text{かつ} \ j \in H \\ 0 & : \quad \text{上記以外} \end{cases} \\ \delta_{im} = \begin{cases} 1 & : \quad i \in y_m \\ 0 & : \quad \text{上記以外} \end{cases} \end{cases} \qquad (2.28)$$

---

* ただし，式（2.27）が成立せずrank（$\boldsymbol{D}$）$= |H|' < |H|$ の場合も，（$|H| - |H|' + 1$）個のボトルネック・プロセッサにおける各 ψ の線形和を改めて ψ_j とおき，ボトルネック・プロセッサ数を $|H|'$ とみなすことにより T_i は求められる。

$$\begin{cases} \sum_i d_{ij} N_i \,/\, (R_{io} + \sum_k \delta_{ik} \theta_{ik} \psi_k + \delta_{im} \theta_{im} \psi_m + z_i) = 1 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (j \in H) \\ \sum_i \{ (R_{io} + \sum_k \delta_{ik} \theta_{ik} \psi_k) / s_i \} N_i \\ \qquad\qquad / (R_{io} + \sum_k \delta_{ik} \theta_{ik} \psi_k + \delta_{im} \theta_{im} \psi_m + z_i) = 1 \end{cases} \qquad (2.29)$$

ψ_mは実メモリを１回割り当てられる（スワップ・インされる）ための平均待ち時間，θ_{im}はiクラス・トランザクションによる実メモリの平均使用回数である。（短小なトランザクションでは$\theta_{im} = 1$だが，長大なものはしばしばスワップ・アウトされるので$\theta_{im} \geq 2$となりうる）。ボトルネック資源の最低優先度トランザクションについて，これを１回使用するための平均待ち時間が等しいという仮定から式（2.28）が得られる。また，式（2.28）を式（2.12）に用いることにより式（2.29）が得られる。

式（2.29）よりψ_k，ψ_mを求め，さらに式（2.28）よりT_iを求める。求解の条件は実メモリ容量無限大の場合と同じく式（2.27）で与えられる（証明は付録８）。得られた解は式（2.17）に加え式（2.24）をみたす必要がある。これは$\psi_m \geq 0$に対応する。

なお，ＡＭＩと異なり，式（2.26）および式（2.29）は線形化できないが，ニュートン・ラフソン法を用いて数値解を得ることができる。ここで，式（2.26）および式（2.29）は一般に複数の解をもつ。したがって妥当な解を得るためには，適切な初期値を設定する必要がある。これは，端末数（呼源数）を１から順に増加していき，端末数（$N-1$）のときの解を初期値として端末数Nのときの解を求めるという方法により実行可能である。

2.3 解析例

前節でみちびいた漸近モデルによる解析例を以下に述べる（本例ではＡＭＩとＡＭⅡは同一の結果を与える*）。実用上よく現れる２つの場合について解析する。第１はＣＰＵないし入出力装置がネックとなりメモリ・ネックとはならない場合であり，第２はＣＰＵないしメモリがネックとなりＩＯネックとはならない場合である。前者は複数プロセッサ・ネック，後者はプロセッサ＆メモリ・ネックの例である。ネックとなる入出力装置はシステム内で１個とし，以下入出力装置（ＩＯ）とはこれを意味する。なお，簡単のため，トランザクション・クラスはＡ，Ｂ２クラスとし，平均思考時間および平均実メモリ割当て量は等しいとする。すなわち，

$$\begin{cases} z \triangleq z_A = z_B \\ s \triangleq s_A = s_B \end{cases} \qquad (2.30)$$

＊　ボトルネック資源の最低優先度クラスが２つ以上存在しない限り，ＡＭⅠとＡＭⅡの結果に差異は生じない。

とする。さらに，

$$\begin{cases} N_A \triangleq \alpha_A N \\ N_B \triangleq \alpha_B N \qquad (\alpha_A + \alpha_B = 1) \end{cases} \tag{2.31}$$

とおき，α_Aとα_BはNによらないとする。この仮定により，議論の一般性は失なわれない。なお以下，クラスi（i＝A，B）のトランザクションに関するＣＰＵ，入出力装置，実メモリの割当て優先度をそれぞれP_i（ＣＰＵ），P_i（ＩＯ），P_i（ＭＥＭ）とかく。

(1) 例１：ＣＰＵ／ＩＯネック

P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＩＯ）＞P_B（ＩＯ），およびP_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＩＯ）＜P_B（ＩＯ），の２つの場合について解析する。式（2.12）～（2.15），（2.17）より，以下の結果が得られる。なお，入出力装置をページ入出力装置とみなせば，本例はページング・ネックの解析ととらえることもできる。まず，メモリ・ネックでないので次式がなりたつ。

$$\begin{cases} T_A = R_A \\ T_B = R_B \end{cases} \tag{2.32}$$

(a) P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＩＯ）＞P_B（ＩＯ）

T_A，T_Bを表2.1に示す。T_AはR_{A0}に一致し，T_BはNに関して双曲線ないし直線を描く。d_{ij}，α_iの大小により，６つの場合がある。場合分けを行なうにあたって，まずパラメータΔ_1を導入する。ＡとＢの２クラス，プロセッサとしてＣＰＵとＩＯの２つがあるという条件で，式（2.12）の第１式に式（2.30），（2.31）を用いると，Nが小さく待ち合わせがないときのＣＰＵの利用率は，

$$\left(\frac{d_{A,CPU}}{R_{A0}+z} \alpha_A + \frac{d_{B,CPU}}{R_{B0}+z} \alpha_B \right) N$$

である。またＩＯの利用率は同様に，

$$\left(\frac{d_{A,IO}}{R_{A0}+z} \alpha_A + \frac{d_{B,IO}}{R_{B0}+z} \alpha_B \right) N$$

で与えられる。いま両者の差をとってNで割り，

$$\Delta_1 \triangleq \frac{d_{A,CPU} - d_{A,IO}}{R_{A0}+z} \alpha_A + \frac{d_{B,CPU} - d_{B,IO}}{R_{B0}+z} \alpha_B \tag{2.33}$$

とおくと，Δ_1は待ち合わせがないときの１端末あたりの資源利用率の差異を表わす。したがって，Nを０から増していくと，$\Delta_1 > 0$ならＣＰＵ，$\Delta_1 < 0$ならＩＯが，それぞれまずネックとなる。ＣＰＵとＩＯの置換に関して系は対称だから，表2.1に示す３つの場合を考えればよい。各場合につき，Nを増していったときの挙動を述べる。

(i) $d_{A,CPU} \geq d_{A,IO}$ かつ $d_{B,CPU} \geq d_{B,IO}$ ：

クラスA，BともCPUバウンドなので，式（2.33）より$\Delta_1 \geq 0$であり，Nを増していくとまずCPUネックとなる。やがてCPUがAクラス・トランザクションで占有され，T_Bは∞となる。

(ii) $d_{A,CPU} < d_{A,IO}$ かつ $d_{B,CPU} \geq d_{B,IO}$ かつ $\Delta_1 \geq 0$：

$\Delta_1 \geq 0$なのでまずCPUネックとなるが，CPU割当て優先度の高いAクラス・トランザクションが優先的に処理されるにしたがい，やがてIOネックに変化する。これはAクラス・トランザクションがIOバウンド（$d_{A,CPU} < d_{A,IO}$）のためである。さらにNを増すとIOがAクラス・トランザクションで占有されてT_Bは∞となる。あるNを越えると，$u_{CPU}=1$から$u_{CPU}<1$に変化するのは興味深い。

(iii) $d_{A,CPU} < d_{A,IO}$ かつ $d_{B,CPU} \geq d_{B,IO}$ かつ $\Delta_1 < 0$：

$\Delta_1 < 0$なので，まずIOネックとなる。IO割当て優先度が高く，IOバウンド（$d_{A,CPU} < d_{A,IO}$）のAクラス・トランザクションが優先的に処理されるので，さらにNを増すとIOがAクラス・トランザクションで占有されてT_Bは∞となる。

(b) P_A（CPU）$> P_B$（CPU）かつ P_A（IO）$< P_B$（IO）：

応答性向上のためには資源ごとに相異なる優先度を与える必要は無いが，処理能力向上のためには本例のような設定が有効な場合もある。T_A，T_Bを表2.1に示す。T_A，T_BはNに関して双曲線ないし直線を描く。6つの場合があるが，系はCPUとIOおよびAとBとの置換に関し対称だから，表2.1に示す3つの場合を考えればよい。各場合につき，Nを増していったときの挙動を述べる。

(i) $d_{A,CPU} \geq d_{A,IO}$ かつ $d_{B,CPU} \geq d_{B,IO}$ ：

IOネックとはならないので，(a)(i)と同様の挙動をとる。

(ii) $d_{A,CPU} < d_{A,IO}$ かつ $d_{B,CPU} \geq d_{B,IO}$ かつ $\Delta_1 \geq 0$ ：

まずCPUネックとなるが，やがてCPUかつIOネックとなる。以後Nを増大すると，AクラスはIO，BクラスはCPUで待ちが発生し，T_AとT_Bはともに線形に増大してNの上限は存在しない。u_{CPU}，u_{IO}はともに1であるから，これはCPUでIOバウンド，IOでCPUバウンドのトランザクションを優先することの処理能力向上効果を表わしている。これは(a)(ii)と比較すると明らかである。

(iii) $d_{A,CPU} \geq d_{A,IO}$ かつ $d_{B,CPU} < d_{B,IO}$ かつ $\Delta_1 \geq 0$：

まずCPUネックとなり，やがてCPUがAクラス・トランザクションで占有されT_Bは∞となる。CPUでCPUバウンド，IOでIOバウンドのトランザクションを優先すると処理能力は低下する。

(2) 例2：ＣＰＵ／メモリ・ネック

P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＭＥＭ）＞P_B（ＭＥＭ），およびP_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＭＥＭ）＜P_B（ＭＥＭ），の２つの場合について解析する。式（2.12）（2.15），（2.17），（2.18）～（2.22），（2.24）より，以下の結果が得られる。ただし，本例はＡ，ＢいずれもＣＰＵバウンドであり，次式を仮定する。

$$d_{i,CPU} > d_{i,IO} \quad (i = A, B) \tag{2.34}$$

(a) P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＭＥＭ）＞P_B（ＭＥＭ）

T_B，R_Bを表2.2に示す。T_A，R_AはR_{A0}に一致し，T_B，R_BはNに関して双曲線ないし直線を描く。d_{ij}，α_i，sの大小により，５つの場合がある。いま，

$$\Delta_2 \triangleq \frac{d_{A,CPU} - R_{A0}/s}{R_{A0} + z}\alpha_A + \frac{d_{B,CPU} - R_{B0}/s}{R_{B0} + z}\alpha_B \tag{2.35}$$

とおく。Nを0から増していくとき，$\Delta_2 > 0$ならＣＰＵ，$\Delta_2 < 0$ならメモリが，それぞれまずネックとなる。各場合につき，Nを増していったときの挙動を述べる。

(i) $\dfrac{d_{A,CPU} - R_{A0}/s}{R_{A0} + z}\alpha_A \geq \dfrac{\alpha_B}{s}$ ：

ＣＰＵネックとなり，メモリ・ネックとなる前にＣＰＵがＡクラス・トランザクションで占有されT_Bは∞となる。

(ii) $\Delta_2 \geq 0$ かつ $\dfrac{d_{A,CPU} - R_{A0}/s}{R_{A0} + z}\alpha_A < \dfrac{\alpha_B}{s}$ かつ $d_{A,CPU} \geq \dfrac{R_{A0}}{s}$ ：

まずＣＰＵネックとなり，ついでＣＰＵネックかつメモリ・ネックとなり（このときR_BはNの増加関数），やがてＣＰＵはＡクラス・トランザクションで占有されT_Bは∞となる。

(iii) $\Delta_2 \geq 0$かつ $d_{A,CPU} < \dfrac{R_{A0}}{s}$ ：

まずＣＰＵネックとなり，ついでＣＰＵネックかつメモリ・ネックとなり（このときR_BはNの減少関数），さらにNを増すとメモリ・ネックのみとなり，やがてメモリはＡクラス・トランザクションで占有されT_Bは∞となる。あるNをこえると$u_{CPU} = 1$から$u_{CPU} < 1$に変化するのは興味深い。このNはR_BがR_{B0}と一致する点である。

(iv) $\Delta_2 < 0$ かつ $d_{A,CPU} \geq \dfrac{R_{A0}}{s}$ ：

表2.1　漸近モデルによる平均応答時間解析例1

An example of mean response time analysis
by Asymptotic Model

(a)　$P_A(CPU) > P_B(CPU)$ かつ $P_A(IO) > P_B(IO)$　（次表で，CPU⇄IOの置換を行なってよい）

条　　件	N の変域	$T_A(=R_A)$	$T_B(=R_B)$
(a)(i)　$d_{A,CPU} \geq d_{A,IO}$ & $d_{B,CPU} \geq d_{B,IO}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \hat{N}_{CPU}$	R_{A0}	$F_1(N)$
(a)(ii)　$d_{A,CPU} < d_{A,IO}$ & $d_{B,CPU} \geq d_{B,IO}$ & $\Delta_1 \geq 0$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \tilde{N}$	R_{A0}	$F_1(N)$
	$\tilde{N} \leq N < \hat{N}_{IO}$	R_{A0}	$F_2(N)$
(a)(iii)　$d_{A,CPU} < d_{A,IO}$ & $d_{B,CPU} \geq d_{B,IO}$ & $\Delta_1 < 0$	$1 \leq N < N^*_{IO}$	R_{A0}	R_{B0}
	$N^*_{IO} \leq N < \hat{N}_{IO}$	R_{A0}	$F_2(N)$

(b)　$P_A(CPU) > P_B(CPU)$ かつ $P_A(IO) < P_B(IO)$　（次表で，$\begin{pmatrix} CPU \rightleftarrows IO \\ A \rightleftarrows B \end{pmatrix}$ の置換を行なってよい）

条　　件	N の変域	$T_A(=R_A)$	$T_B(=R_B)$
(b)(i)　$d_{A,CPU} \geq d_{A,IO}$ & $d_{B,CPU} \geq d_{B,IO}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \hat{N}_{CPU}$	R_{A0}	$F_1(N)$
(b)(ii)　$d_{A,CPU} < d_{A,IO}$ & $d_{B,CPU} \geq d_{B,IO}$ & $\Delta_1 \geq 0$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \tilde{N}$	R_{A0}	$F_1(N)$
	$\tilde{N} \leq N < \infty$	$F_3(N)$	$F_4(N)$
(b)(iii)　$d_{A,CPU} \geq d_{A,IO}$ & $d_{B,CPU} < d_{B,IO}$ & $\Delta_1 \geq 0$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \hat{N}_{CPU}$	R_{A0}	$F_1(N)$

表 2.1（続き）

$$\Delta_1 = \frac{d_{A,CPU} - d_{A,IO}}{R_{A0} + z} \alpha_A + \frac{d_{B,CPU} - d_{B,IO}}{R_{B0} + z} \alpha_B$$

$$N^*_{CPU} = 1 \Big/ \left(\frac{d_{A,CPU}}{R_{A0} + z} \alpha_A + \frac{d_{B,CPU}}{R_{B0} + z} \alpha_B \right)$$

$$\hat{N}_{CPU} = \frac{R_{A0} + z}{d_{A,CPU}\, \alpha_A}$$

$$\tilde{N} = \frac{d_{B,CPU} - d_{B,IO}}{d_{A,IO}\, d_{B,CPU} - d_{A,CPU} d_{B,IO}} \cdot \frac{R_{A0} + z}{\alpha_A}$$

$$\hat{N}_{IO} = \frac{R_{A0} + z}{d_{A,IO}\, \alpha_A}$$

$$N^*_{IO} = 1 \Big/ \left(\frac{d_{A,IO}}{R_{A0} + z} \alpha_A + \frac{d_{B,IO}}{R_{B0} + z} \alpha_B \right)$$

$$F_1(N) = \left\{ d_{B,CPU}\, \alpha_B N \Big/ \left(1 - \frac{d_{A,CPU}}{R_{A0} + z} \alpha_A N \right) \right\} - z$$

$$F_2(N) = \left\{ d_{B,IO}\, \alpha_B N \Big/ \left(1 - \frac{d_{A,IO}}{R_{A0} + z} \alpha_A N \right) \right\} - z$$

$$F_3(N) = \frac{d_{A,IO}\, d_{B,CPU} - d_{A,CPU}\, d_{B,IO}}{d_{B,CPU} - d_{B,IO}} \alpha_A N - z$$

$$F_4(N) = \frac{d_{A,IO}\, d_{B,CPU} - d_{A,CPU}\, d_{B,IO}}{d_{A,IO} - d_{A,CPU}} \alpha_B N - z$$

まずメモリ・ネックとなり，ついでＣＰＵネックかつメモリ・ネックとなり（このときR_BはNの増加関数），やがてＣＰＵはＡクラス・トランザクションで占有されT_Bは∞となる。

(v)　$\Delta_2 < 0$ かつ $d_{A,CPU} < \dfrac{R_{A0}}{s}$ ：

まずメモリ・ネックとなり，ＣＰＵネックとなる前にメモリがＡクラス・トランザクションで占有されT_Bは∞となる。

(b)　P_A（ＣＰＵ）$> P_B$（ＣＰＵ）かつ P_A（ＭＥＭ）$< P_B$（ＭＥＭ）

T_A，T_Bを表2.2に示す。R_AはR_{A0}に，R_BはT_Bに，それぞれ一致する。T_A，T_BはNに関して双曲線ないし直線を描く。d_{ij}，α_i，sの大小により５つの場合がある。各々につき，Nを増していったときの挙動を述べる。

(i)　$\dfrac{d_{A,CPU} - R_{A0}/s}{R_{A0} + z} \quad \alpha_A \geq \dfrac{\alpha_B}{s}$ ：

ＣＰＵネックとなり，(a)(i)と同様に，メモリ・ネックとなる前にＣＰＵがＡクラス・トランザクションで占有されT_Bは∞となる。

(ii)　$\Delta_2 \geq 0$ かつ $\dfrac{d_{A,CPU} - R_{A0}/s}{R_{A0} + z} \quad \alpha_A < \dfrac{\alpha_B}{s}$ かつ $d_{B,CPU} \geq \dfrac{R_{B0}}{s}$ ：

まずＣＰＵネックとなり，ついでＣＰＵネックかつメモリ・ネックとなり（このときT_AはNの増加関数，T_Bの増減は一般には不明），やがてメモリはＢクラス・トランザクションで占有されT_Aは∞となる。(b)(iii)でも同様であるが，Nの増大にともなってT_Bが減少する場合がみられるのは興味深い。ただし，$d_{B,CPU} \geq R_{B0}/s$であれば，T_Bが減少関数でも値がR_{B0}と一致する以前にT_Aが∞となる。また(b)(iii)，(b)(v)でも同様であるが，ＣＰＵネックかつメモリ・ネックの場合，Nを十分大きくするとT_Aは∞に発散するが，T_Bは有限値$R_{A0}\, d_{B,CPU}/d_{A,CPU}$に収束する。すなわち，一般にプロセッサと実メモリがともにネックとなる環境では，実メモリの割当て優先度の高いトランザクションの方が応答性の点で有利な場合が多い。

(iii)　$\Delta_2 \geq 0$ かつ $\dfrac{d_{A,CPU} - R_{A0}/s}{R_{A0} + z} \quad \alpha_A < \dfrac{\alpha_B}{s}$ かつ $d_{B,CPU} < \dfrac{R_{B0}}{s}$ ：

まずＣＰＵネックとなり，ついでＣＰＵネックかつメモリ・ネックとなり（このときT_AはNの増加関数，T_BはNの減少関数），さらにNを増すとメモリ・ネックのみとなる。あるNをこえると$u_{CPU} = 1$から$u_{CPU} < 1$に変化するのは興味深い。このNは，T_BがR_{B0}と一致する点である。

(v)　$\Delta_2 < 0$ かつ $d_{B,CPU} \geq \dfrac{R_{B0}}{s}$ ：

まずメモリ・ネックとなり，ついでＣＰＵネックかつメモリ・ネックとなり（このときT_A，T_BはともにNの増加関数），やがてメモリがＢクラス・トランザクションで占有されT_Aは∞となる。

(v)　$\Delta_2 < 0$かつ $d_{B,CPU} < \dfrac{R_{B0}}{s}$ ：

メモリ・ネックとなり，ＣＰＵネックとなる前にメモリがＢクラス・トランザクションで占有されT_Aは∞となる。

2.4　シミュレーションと実測による検証

2.4.1　シミュレーションによる検証

漸近モデル（Asymptotic Model）をシミュレーションにより検証した。検証したシステムの周囲条件は，前節の例２と同一である*。ただし，トランザクション・クラスＡ，Ｂにおいて，式（2.30）に加え各プロセッサの使用量d_{ij}（i＝Ａ，Ｂ）は全て等しいと仮定し，以後これらの変数における添字iは省略する。また，式（2.31）において，N個の端末は２クラスに等分割されるとする。トランザクションは入出力装置ＩＯ１，ＩＯ２を使用するが，例２と同じく式（2.34）を仮定し，ＩＯネックにはならないとする。

優先度系(1) P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＭＥＭ）＞P_B（ＭＥＭ），(2) P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＭＥＭ）＜P_B（ＭＥＭ），のそれぞれについての漸近モデルによる解析結果は，前節で示した表2.2において次式を用いることにより得られる。

$$\begin{cases} d_j \triangleq d_{Aj} = d_{Bj} \\ R_0 \triangleq R_{A0} = R_{B0} \\ \alpha_A = \alpha_B = 0.5 \end{cases} \qquad (2.36)$$

式（2.36）のもとでは式（2.35）のΔ_2の正負は（$d_{CPU} - R_0/s$）と一致するので，表2.2における(a)(iii)，(a)(v)，(b)(iii)，(b)(v)は存在しない。すなわち，優先度系(1)(2)のもとでのシステムの挙動は，それ

＊　したがってＡＭⅠとＡＭⅡは同一の結果を与える。

表 2.2　漸近モデルによる平均応答時間解析例 2

An example of mean response time analysis by Asymptotic Model

(a)　$P_A(MEM) > P_B(MEM)$ かつ $P_A(CPU) > P_B(CPU)$

条　　件	N　の　変　域	$T_A(=R_A)$	T_B	R_B
(a)(i) $\frac{d_{A,CPU}-R_{A0}/s}{R_{A0}+z}\alpha_A \geq \frac{\alpha_B}{s}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}	R_{B0}
	$N^*_{CPU} \leq N < \hat{N}_{CPU}$	R_{A0}	$G_1(N)$	$G_1(N)$
(a)(ii) $\Delta_2 \geq 0$ & $\frac{d_{A,CPU}-R_{A0}/s}{R_{A0}+z}\alpha_A < \frac{\alpha_B}{s}$ & $d_{A,CPU} \geq \frac{R_{A0}}{s}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}	R_{B0}
	$N^*_{CPU} \leq N < \tilde{N}$	R_{A0}	$G_1(N)$	$G_1(N)$
	$\tilde{N} \leq N < \hat{N}_{CPU}$	R_{A0}	$G_1(N)$	$G_2(N)$
(a)(iii) $\Delta_2 \geq 0$ & $d_{A,CPU} < \frac{R_{A0}}{s}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{B0}	R_{B0}
	$N^*_{CPU} \leq N < \tilde{N}$	R_{A0}	$G_1(N)$	$G_1(N)$
	$\tilde{N} \leq N < \tilde{\tilde{N}}$	R_{A0}	$G_1(N)$	$G_2(N)$
	$\tilde{\tilde{N}} \leq N < \hat{N}_{MEM}$	R_{A0}	$G_3(N)$	R_{B0}
(a)(iv) $\Delta_2 < 0$ & $d_{A,CPU} \geq \frac{R_{A0}}{s}$	$1 \leq N < N^*_{MEM}$	R_{A0}	R_{B0}	R_{B0}
	$N^*_{MEM} \leq N < \tilde{\tilde{N}}$	R_{A0}	$G_3(N)$	R_{B0}
	$\tilde{\tilde{N}} \leq N < \hat{N}_{CPU}$	R_{A0}	$G_1(N)$	$G_2(N)$
(a)(v) $\Delta_2 < 0$ & $d_{A,CPU} < \frac{R_{A0}}{s}$	$1 \leq N < N^*_{MEM}$	R_{A0}	R_{B0}	R_{B0}
	$N^*_{MEM} \leq N < \hat{N}_{MEM}$	R_{A0}	$G_3(N)$	R_{B0}

表 2.2（続き）

(b) $P_A(MEM) < P_B(MEM)$ かつ $P_A(CPU) > P_B(CPU)$

条　　件	N の変域	T_A	R_A	$T_B(=R_B)$
(b)(i) $\frac{d_{A,CPU}-R_{A0}/s}{R_{A0}+z}\alpha_A \geq \frac{\alpha_B}{s}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \hat{N}_{CPU}$	R_{A0}	R_{A0}	$G_1(N)$
(b)(ii) $\Delta_2 \geq 0$ & $\frac{d_{A,CPU}-R_{A0}/s}{R_{A0}+z}\alpha_A < \frac{\alpha_B}{s}$ & $d_{B,CPU} \geq \frac{R_{B0}}{s}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \tilde{N}$	R_{A0}	R_{A0}	$G_1(N)$
	$\tilde{N} \leq N < \hat{\hat{N}}_{MEM}$	$G_4(N)$	R_{A0}	$G_5(N)$
(b)(iii) $\Delta_2 \geq 0$ & $\frac{d_{A,CPU}-R_{A0}/s}{R_{A0}+z}\alpha_A < \frac{\alpha_B}{s}$ & $d_{B,CPU} < \frac{R_{B0}}{s}$	$1 \leq N < N^*_{CPU}$	R_{A0}	R_{A0}	R_{B0}
	$N^*_{CPU} \leq N < \tilde{N}$	R_{A0}	R_{A0}	$G_1(N)$
	$\tilde{N} \leq N < \tilde{\tilde{N}}'$	$G_4(N)$	R_{A0}	$G_5(N)$
	$\tilde{\tilde{N}}' \leq N < \hat{N}'_{MEM}$	$G_6(N)$	R_{A0}	R_{B0}
(b)(iv) $\Delta_2 < 0$ & $d_{B,CPU} \geq \frac{R_{B0}}{s}$	$1 \leq N < N^*_{MEM}$	R_{A0}	R_{A0}	R_{B0}
	$N^*_{MEM} \leq N < \tilde{\tilde{N}}'$	$G_6(N)$	R_{A0}	R_{B0}
	$\tilde{\tilde{N}}' \leq N < \hat{\hat{N}}_{MEM}$	$G_4(N)$	R_{A0}	$G_5(N)$
(b)(v) $\Delta_2 < 0$ & $d_{B,CPU} < \frac{R_{B0}}{s}$	$1 \leq N < N^*_{MEM}$	R_{A0}	R_{A0}	R_{B0}
	$N^*_{MEM} \leq N < \hat{N}'_{MEM}$	$G_6(N)$	R_{A0}	R_{B0}

$$\Delta_2 = \frac{d_{A,CPU}-R_{A0}/s}{R_{A0}+z}\alpha_A + \frac{d_{B,CPU}-R_{B0}/s}{R_{B0}+z}\alpha_B$$

表2.2（続き）

$$N^{*}_{CPU} = 1 \Big/ \left(\frac{d_{A,CPU}}{R_{A0}+z}\alpha_A + \frac{d_{B,CPU}}{R_{B0}+z}\alpha_B \right)$$

$$N^{*}_{MEM} = s \Big/ \left(\frac{R_{A0}}{R_{A0}+z}\alpha_A + \frac{R_{B0}}{R_{B0}+z}\alpha_B \right)$$

$$\hat{N}_{CPU} = \frac{R_{A0}+z}{d_{A,CPU}\,\alpha_A}$$

$$\hat{N}_{MEM} = \frac{s\,(R_{A0}+z)}{R_{A0}\,\alpha_A}$$

$$\hat{N}'_{MEM} = \frac{s\,(R_{B0}+z)}{R_{B0}\,\alpha_B}$$

$$\hat{\hat{N}}_{MEM} = \frac{z + s\,d_{B,CPU}}{d_{B,CPU}\alpha_B}$$

$$\tilde{N} = (z + s\,d_{B,CPU}) \Big/ \left\{ d_{B,CPU}\,\alpha_B + \frac{\alpha_A}{R_{A0}+z}\,(d_{A,CPU}\,z + R_{A0}\,d_{B,CPU}) \right\}$$

$$\tilde{\tilde{N}} = \frac{(z+R_{A0})(s\,d_{B,CPU} - R_{B0})}{\alpha_A\,(d_{B,CPU}R_{A0} - d_{A,CPU}R_{B0})}$$

$$\tilde{\tilde{N}}' = \frac{(z+R_{B0})(s\,d_{A,CPU} - R_{A0})}{\alpha_B\,(d_{A,CPU}R_{B0} - d_{B,CPU}R_{A0})}$$

$$G_1(N) = \left\{ d_{B,CPU}\,\alpha_B N \Big/ \left(1 - \frac{d_{A,CPU}}{R_{A0}+z}\alpha_A N \right) \right\} - z$$

$$G_2(N) = \frac{\{ s - R_{A0}\alpha_A N/(R_{A0}+z) \}\, d_{B,CPU}}{\{ 1 - d_{A,CPU}\,\alpha_A N/(R_{A0}+z) \}}$$

$$G_3(N) = \frac{R_{B0}\,\alpha_B\,N}{\{ s - R_{A0}\,\alpha_A\,N/(R_{A0}+z) \}} - z$$

$$G_4(N) = \frac{\alpha_A N(d_{B,CPU}R_{A0} + z\,d_{A,CPU})}{(z + s\,d_{B,CPU}) - d_{B,CPU}\,\alpha_B\,N} - z$$

$$G_5(N) = \frac{\alpha_B N(d_{B,CPU}\,R_{A0} + z\,d_{A,CPU})}{\alpha_B d_{A,CPU}\,N - (s\,d_{A,CPU} - R_{A0})} - z$$

$$G_6(N) = \frac{R_{A0}\,\alpha_A\,N}{\{ s - R_{B0}\,\alpha_B N/(R_{B0}+z) \}} - z$$

ぞれ(a)(i)(ii)(v)，(b)(i)(ii)(v)の各場合に分けられる。

漸近モデルの誤差評価を目的として，詳細なシミュレーションを行なった。現実のシステムにできるだけ近づけるため，入出力動作はnon-preemptiveとし，ＨＩＴＡＣ　Ｈ－８５８９－１ディスクのシーク時間分布[H4]にもとづいてシミュレートした。思考時間とＣＰＵサービス時間については，その分布は一般に不明なので，指数分布と一点分布の２つの場合についてシミュレーションを行なった。実際の分布がアーラン分布であれば，結果は上記２つの値の間に入る。シミュレーションで用いた諸元の値は次の通りである。d_{CPU} ＝ 0.285秒，d_{IO1} ＝ 0.086 秒，d_{IO2} ＝ 0.129 秒，R_0 ＝ 0.5秒，　z ＝ 200秒，　s ＝１～２０。

優先系(1) P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつ P_A（ＭＥＭ）＞P_B（ＭＥＭ）についての，シミュレーションと漸近モデルによる解析との比較結果を図2.5に示す。表2.2(a)(ii)のＣＰＵネックないしＣＰＵネックかつメモリ・ネックの状況は，図2.5のs＝１０，s＝２０に示されている。(a)(i)の場合については，表2.2より明らかなように，T_Bは(a)(ii)のそれと同一なので省略した。s＝１０のとき，s＝２０の場合に比べてより早くメモリ・ネックになる。しかし，ＣＰＵネックである限り，メモリ・ネックであるか否かによらずT_Bは同一値をとるので，s＝１０とs＝２０との間に差異はみとめられない。図2.5において，いかなるNに対しても$T_A \leq T_B$であり，解析結果は低負荷（Nが小）ないし高負荷（Nが大）のときシミュレーション結果と一致している。T_Bの解析にともなう誤差は，負荷が中程度のときにみられ，s＝１，１０，２０いずれの場合においてもNのクリティカルな値において最大となる。ここでクリティカルな値とは，その値以上のとき資源を１００％使用可能となる点を意味する。前述のように，漸近モデルはシステム性能の構造的性質を表現するモデルであるから，構造が変化するクリティカル・ポイントにおいて競合が発生し誤差を生ずるのは予想された結果である。さらに誤差は，思考時間とＣＰＵサービス時間が指数分布にしたがう場合の方が，一点分布にしたがう場合より大である。一般に，系の確率的なランダムネスの程度が増すほど，すなわち資源サービス時間分布の分散が増すほど，誤差は増大する。このような誤差にもかかわらず，図2.5は，システム性能の構造的性質が漸近モデルにより正しく表現されることを示している。

優先度系(2) P_A（ＣＰＵ）＞P_B（ＣＰＵ）かつP_A（ＭＥＭ）＜P_B（ＭＥＭ）についての，シミュレーションと漸近モデルによる解析との比較結果を図2.6に示す。前節の表2.2の(b)(ii)の場合が，s＝１０，s＝２０について図2.6に例示されている。(b)(i)は(a)(i)と等しく，(b)(v)は(a)(v)においてＡとＢを交換した場合にあたるので省略した。既に前節で述べたように，ＣＰＵネックかつメモリ・ネックの場合，Nの増加につれてT_B が減少するという解析結果が得られたが，これはシミュレーションによっても確かめられた。この現象は次のようにして説明できる。ＣＰＵネックのときT_BはＣＰＵ待ちのため増大するが，いったんＣＰＵネックかつメモリ・ネックになるとメモリがＢクラス・トランザクションで占有され始める。この結果，T_Aがメモリ待ちのため増大し，Ａクラス・トランザクションによるＣＰＵ利用率が減ずるので，T_BにおけるＣＰＵ待ちが減少する。したがって，Nの増加にともなってT_Aは増加し，T_Bは減少する。なお，図2.5においてT_Bの過大評価は，T_Aの過小評価に

ともなうＣＰＵ待ちの過大評価によりもたらされたものである。

図2.5と同じく，図2.6においても，漸近モデルによる解析結果はシミュレーション結果が示すシステムの大局的挙動を正しく予想している。ただし，その誤差は性能の構造が遷移する点において顕著となる。

2.4.2　実測による検証

実システムの測定データと漸近モデル（Asymptotic Model）による解析結果を比較することにより，漸近モデルの精度を検証した。実システムでは等優先度のトランザクションに対しＦＣＦＳ（First Come First Served）で資源を割り当てるので，ここではＡＭⅡ（Asymptotic Model Ⅱ）を用いた。

実システムはＨＩＴＡＣ Ｍ－２００Ｈ，実メモリ容量８～１２ＭBytes，入出力装置としてＨ－８５８９－１ディスク４台とＨ－８５９５－１ディスク１１台，端末３３～５８台からなる。アプリケーションは端末から起動されるＴＳＳトランザクションとジョブ・キューからイニシエータにより起動されるバッチ・トランザクションとからなる。ＴＳＳトランザクションはＩ（Interactive）型すなわちテキスト編集などの短小なコマンド処理と，Ｅ（Executive）型すなわちプログラム翻訳実行処理が約３０対１の比率で混在する。また，バッチ・トランザクションは５０種類のＦＯＲＴＲＡＮプログラムの翻訳実行処理であり，バッチ・イニシエータ数は４～５本である。したがって，漸近モデルでは，Ｉ型ＴＳＳ，Ｅ型ＴＳＳ，バッチの３つのトランザクション・クラスを設定する。

ボトルネックとなった資源はＣＰＵと入出力スプール用ディスクであった。実システムではＣＰＵに関し，Ｉ型ＴＳＳトランザクションの優先度が最も高く，Ｅ型ＴＳＳトランザクションに対する単位時間あたりのＣＰＵ割当て量がバッチ・トランザクションに対するそれをやや上回るようスケージュールが行なわれる。[N5] しかし，これを漸近モデルに直接反映させることは難しいので，モデルにおいては２レベルのＣＰＵ割当て優先度を設定し，Ｉ型ＴＳＳトランザクションを高，Ｅ型ＴＳＳおよびバッチ・トランザクションを低とした。また入出力スプール用ディスクについては，優先度は無く，ＴＳＳトランザクションはＩ型Ｅ型ともにこれをほとんど使用しない。

なお実測データによれば実メモリはボトルネックでなく，トランザクションごとのページングの挙動も明確ではなかったので，スワッピングを含めページ入出力処理はモデルから除外した（実システムでは，実メモリに余裕がある限り思考中端末のワーキングセット[D3]も実メモリ内に保つというデマンド・スワッピング方式を採用しているため，トランザクション到着ごとにスワッピングが発生するとは限らない）。

端末数，バッチ・イニシエータ数，実メモリ容量をパラメータとして，ケース１～３の３つの場合について平均応答時間（バッチの場合は平均処理時間）の実測データと漸近モデルの解析データを比較した結果を表2.3に示す。Ｉ型ＴＳＳトランザクションについては，負荷を増すにしたがって（前述のクリティカルな端末数に近づくにしたがって）誤差の増大がみられる。これは，低優先度トラン

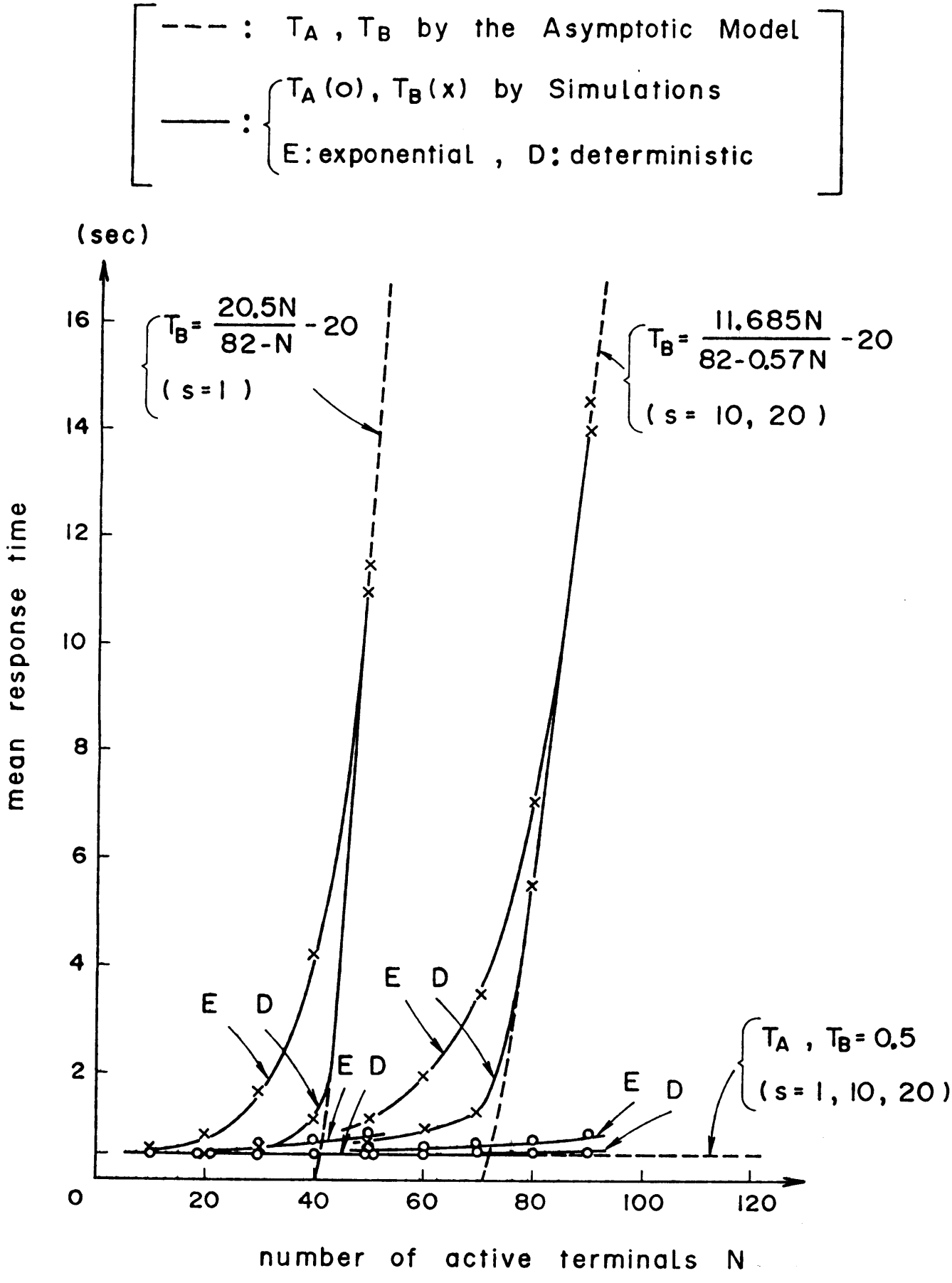


図 2.5 シミュレーションによる漸近モデルの検証
(P_A(CPU) > P_B(CPU) & P_A(MEM) > P_B(MEM)の場合)
The validation of Asymptotic Model by simulations

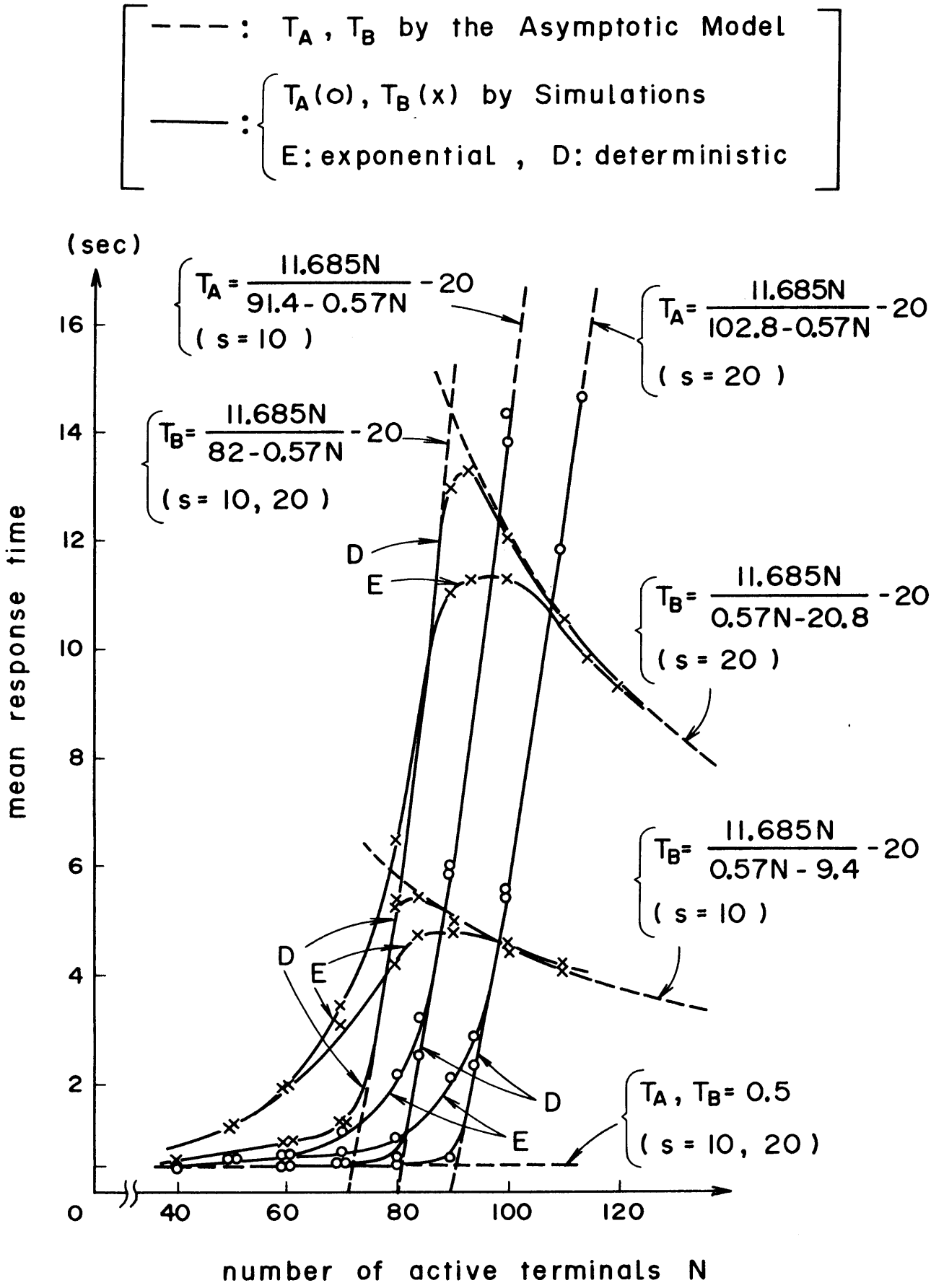


図 2.6 シミュレーションによる漸近モデルの検証

(P_A(CPU)>P_B(CPU) & P_A(MEM)<P_B(MEM) の場合)

The validation of Asymptotic Model by simulations

表2.3　実測による漸近モデルの検証

（上段：解析値，下段：実測値）

Validation of the Asymptotic Model

case no.	number of terminals	number of batch initi-ators	memory capacity (M Bytes)	I-type terminal-initiated	E-type terminal-initiated	Batch
1	33	4	8	{0.21 0.36	{4.70 3.36	{21.6 22.9
2	52	4	8	{0.21 0.62	{5.05 4.53	{23.6 25.5
3	58	5	12	{0.21 0.43	{6.40 4.99	{30.3 30.2

(sec)

ザクションが有限の平均応答時間をもつとき高優先度トランザクションどうしの待ちを評価できないというボトルネック解析の限界をあらわしている。さらにケース2とケース3の実測データを比べると，負荷増大にもかかわらず後者の方が応答性が向上しているが，これは実メモリ増加にともないデマンド・スワッピング方式によるスワッピング省略の確率が増したためである。ページ入出力処理を考慮すれば，これに関連した誤差短縮は可能である。

E型ＴＳＳトランザクションについては，誤差は１１～４０％である。実測データの方が解析データより全般に値が小さいが，この理由は，ＣＰＵ割当てにおいてE型ＴＳＳトランザクションをバッチ・トランザクションより優遇するスケジュールがモデルに反映されていないためである。一方，長大なバッチ・トランザクションについては，誤差は０～７％にとどまった。一般に優先制御を行なうシステムの性能解析で問題となるのは，低優先度トランザクションがより高い優先度（同一優先度を含む）のトランザクションから受ける影響の評価である。表2.3は，漸近モデルによりこの影響が評価でき，低優先度トランザクションの挙動を分析可能なことを示している。

2.5 結　言

多重プログラミング・システムの資源管理において，資源割当て優先度の操作は最も重要な部分を占める。本章では，資源割当て優先制御のもとでのシステム性能の近似解析手法「漸近モデル（Asymptotic Model）」を提案した。漸近モデルは日立の計算機システム性能評価ツールＩＳＣＰ[01]（Interactive tool for System Configuration Planning）の一環として実用に供されている。本手法の範囲や特徴，本手法により得られた知見，本手法の検証結果などを以下にまとめる。

(1) 有限多重度のもとで，各種資源（実メモリ，ＣＰＵ，入出力装置）の割当てに関し各トランザクション（＝ＴＳＳコマンドないしバッチ・ジョブ）に任意の優先度を与えたときの，トランザクションの平均応答時間（バッチの場合は平均処理時間）および資源利用率の近似解を与える方程式をみちびいた。有限呼源の待行列ネットワーク・モデルにおいて，これらの値の厳密解は呼源数の関数として漸近解をもつ。本手法の特徴は，厳密解をその漸近解で近似した点にある。漸近解は，利用率が高くボトルネックとなっている資源の割当て待ち時間に着目することにより得られ，システム性能の構造的性質を表わす。従来，優先制御を扱う近似解析手法は限られた条件のもとで性能をできるだけ詳細に分析するものが多く[B2)K3)N1)S6)]，このように一般的条件のもとでシステムの大局的挙動を簡便に与える手法は報告されていない。

(2) 漸近モデルによる解析例として，2トランザクション・クラスの場合につき，方程式を解いて具体的に各種優先度のもとでの平均応答時間につき検討した。この結果，呼源数を１から増していったときのシステムの挙動が明らかになり，

(a) 処理能力向上のためには，入出力装置割当てに関しＣＰＵバウンド，ＣＰＵ割当てに関しＩＯバウンドのトランザクションに高優先度を与えることが有効である。

(b) ページ入出力装置の使用量が他の資源使用量に比して大のとき，呼源数を増すにしたがって

ページング・ネックとなる。このとき，平均応答時間はページ入出力装置における待ちでほぼ決定され，ＣＰＵなど他資源の割当てにおける優先制御は無効となる。

など，興味深い結果がえられた。これらは従来より直観的には認められていたが，理論的に示されたことに意味がある。

(3) 漸近モデルの解析能力を検証するため，シミュレーションおよび実測を行なった。これから，

(a) 漸近モデルにより，優先度をもつシステムの大局的挙動を分析し，性能の構造的性質を解析できる。特に長大な低優先度トランザクションの挙動分析に有効である。

(b) 漸近モデルの誤差は，システム性能の構造が遷移する点において大である。また，トランザクションの資源使用時間の分布における分散とともに増大する。

などの結論を得た。

本章では優先度が時間的に不変な固定優先度を扱った。実際のオペレーティング・システムではこれが動的に変更される場合も多い。動的優先度をもつシステムの解析は，今後理論的検討が待たれる分野である。

本章の議論より，仮想メモリ方式のもとでは，ページング処理のオーバヘッドが性能に多大な影響を与えることが示されたが，この問題は第３章，第４章で扱う。また，利用率の高いボトルネック資源の割当て優先度がシステム性能の決定要因であることもわかった。これは第５章で提案する資源管理方式を示唆するものである。

第3章　仮想メモリ・システムの
ワーキングセット最適化方式

3. 仮想メモリ・システムのワーキングセット最適化方式

3.1 概　要

仮想メモリ方式の多重プログラミング・システムでは，各ジョブへ実メモリを適切に割り当てることが処理能力や応答性の向上の必要条件である。第2章の議論より，この条件がみたされないとページング処理が頻発し，これがボトルネックとなって性能が低下することがわかった。本章では，ページング・ネック防止のための実メモリ管理の最適化に関する問題を扱う。

一般に実メモリ割当て量を増せばページング処理の頻度を削減できるので，ジョブの処理時間は実メモリ割当て量の減少関数となる。したがって実メモリ割当て量と処理時間との積であるＳＴＰ(Space Time Product)を考えると，これを実メモリ管理方式の評価基準として用いることができ，従来より理論的検討においてはその最小化が目標とされてきた[B5)C3)G3)P2)S9)]。しかしＳＴＰは本来ジョブの実行完了時点でそのページ参照特性から算定される事後的な量である。このため，現実システムでは実行中のジョブのＳＴＰは不明であり，その最小化を達成できないという問題点があった。

本章では，現実システムの多重プログラミング環境のもとで，実行中の各ジョブのＳＴＰの推定値を最小化する「ワーキングセット最適化方式ＯＷＥ(Optimum Working-set Estimator)」を提案する[N10)N11)]。ページ参照特性はプログラムのフェイズ進行とともに変化しうるが，ひとつのフェイズの中では定常的とみなせる場合が多い[H1)]。ページ参照特性の定常性を仮定し，ジョブ実行中にこれを学習するのがＯＷＥの特徴である。ＳＴＰの推定値最小化は，具体的にはワーキングセット・ポリシー[D3)]のウインド・サイズτの調節により遂行される。有効な学習のためにはある程度の時間が必要なので，ＯＷＥの主対象は比較的長大なジョブである。これら長大なジョブはシステム性能に与える影響が大きいので，これらを適切に制御することにより，性能向上が期待できる。

本章では3.2節でＳＴＰを定義し，第2章の議論を用いて，システム内の各ジョブのＳＴＰの最小化とシステム全体の性能との関連につき述べる。ついで3.3節では，ＯＷＥを提案する。さらに3.4節では，ＯＷＥを日立のオペレーティング・システムＶＯＳ３において実現し，実験を行なった結果について述べる。まず定常的なページ参照動作をする人工的ジョブ(synthetic job)を用いて，ＯＷＥのページ参照特性学習能力，ＳＴＰ最小化達成能力などを検討する。ついで実際のベンチマーク・ジョブを用いて，ＯＷＥによる性能向上効果を実測する。

なお本章では，ページング処理のオーバヘッドを決定する第2の要因であるページ入出力実行時間は与件として議論を進めた。実際にはページ入出力実行時間は二次メモリ管理方式に依存するが，この問題については第4章で述べる。

3.2 Space Time Productと処理能力

まずＳＴＰ(Space Time Product)を定義し，これと処理能力との関係について述べる。いま，長さVのページ参照列を考える。第k番目の参照時点における実メモリ割当て量をw_kとする。参照

ページに実メモリが割り当てられていなければページ・フォールトが発生し，当該ページは実メモリ内に読みこまれる。ここでＳＴＰは次式で定義される。

$$XV \triangleq \sum_{k=1}^{V} w_k + \sum_{k=1}^{V} e_k w_k S \qquad (3.1)$$

ただし，

$$e_k = \begin{cases} 1：k\text{番目の参照でページ・フォールト発生} \\ 0：\text{上記以外} \end{cases} \qquad (3.2)$$

Sはページ・フォールト処理に要する時間を実メモリ参照回数に換算した値であり，一定値と仮定する。すなわち式（３.１）において，時間の単位は単位参照あたりの実メモリ・アクセス時間とする。w_k，e_kの平均値$\sum_{k=1}^{V} w_k / V$，$\sum_{k=1}^{V} e_k / V$がそれぞれ平均ワーキングセット・サイズ$\bar{w}$，ページ・フォールト率fである。このとき，式（３.１）は$\bar{w}$とfを用いて次のように表わされる。

$$XV = V(\bar{w} + S f \bar{w}) \qquad (3.3)$$

評価基準である単位参照あたりのＳＴＰは次式で与えられる。

$$X = \bar{w} + S f \bar{w} \qquad (3.4)$$

実メモリ割当て量の評価基準としてX[G3)]は代表的なものであり，これを最小化する方法を本論文ではＳＴＰルールとよぶ。

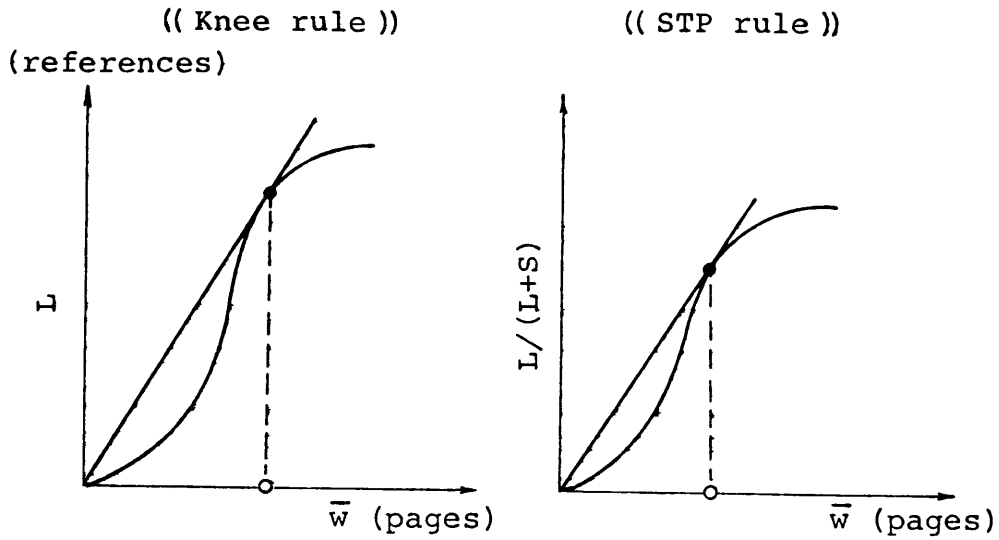
類似の評価基準として，ライフタイム（平均ページ・フォールト発生間隔）Lと$\bar{w}$との比がよく知られているので付記する。Lは$1/f$に等しく，$\bar{w}$の関数として表わすことができる。多くのプログラムにおいて単調増加関数$L(\bar{w})$は，$\bar{w}$を増すにつれて下に凸から上に凸に変化することが知られている[D7)]。$L/\bar{w}$を最大化する点は曲線$L(\bar{w})$の「Knee」に対応する（図３.１参照）。この点に実メモリ割当て量を一致させる方法はKneeルールとよばれる[D7)D8)]。なお，Kneeルールは式（３.４）の第２項の最小化に等しいことから，これを式（３.４）の第１項を無視したＳＴＰルールと考えることもできる。

実メモリ容量は有限であるからＳＴＰ削減は一般に処理能力を向上させる[D8)]。以下これについて考察する。いま，$\bar{w}$の関数$L/(L+S)$について考える。Lを$1/f$でおきかえ式（３.４）を用いると，

$$L/(L+S) = \bar{w}/X \qquad (3.5)$$

であるから，ＳＴＰルールは$\{L/(L+S)\}/\bar{w}$の最大化に等しい。すなわち図３.１に示したように，ＳＴＰルールは曲線$L/(L+S)$の「Knee」に対応する$\bar{w}$を与える方法に他ならない。（$L/(L+S)$はProcessor Efficiency[B5)]，LはLifetimeであるから，ＳＴＰルール，Kneeルールをそれぞれ ＰＥ－Kneeルール，ＬＴ－Kneeルールともよぶ[N11)]）。なお，$L/(L+S)$のKneeに対応する$\bar{w}$は，Lのそれより通常小さい。詳細は付録１を参照されたい。

以上より，ＳＴＰルールは投資（＝実メモリ量）に対する効果（$=L/(L+S)$）の比の最大化を意味することがわかる。ここで，$L/(L+S)$と処理能力との関係について考える。これまでひとつのトランザクション*に着目し，その仮想時間（ＣＰＵ実行時間）の軸上で議論を進めてきたが，



$\bar{w}$: Average Working-set Size
L: Lifetime
S: Page Fault Processing Delay

図3.1 **STP**ルールおよび**Knee**ルールによる最適な実メモリ割当て量

Optimum working-set sizes by STP rule and Knee rule

以下本節では多重プログラミング環境における複数のトランザクションを考え，実時間の軸上で議論を行なう。いま，トランザクションのページ参照動作のみに着目しているので，処理能力の尺度としてＣＰＵ利用率を採用する。トランザクション i が実行完了までに発生する平均ページ・フォールト回数を κ_i，i のライフタイムを L_i とし，$d_{i,\mathrm{CPU}}$，$d_{i,\mathrm{PG}}$ によってそれぞれ i の平均ＣＰＵ使用時間，i の平均ページ入出力装置使用時間を表わすと，

$$\begin{cases} d_{i,\mathrm{CPU}} = \kappa_i L_i \\ d_{i,\mathrm{PG}} = \kappa_i S \end{cases} \qquad (3.6)$$

が成り立つ。式（3.6）より，

$$d_{i,\mathrm{CPU}} / d_{i,\mathrm{PG}} = L_i / S \qquad (3.7)$$

がえられる。トランザクション i によるＣＰＵ利用率を $u_{\mathrm{CPU}}^{(i)}$ とする。ここで第２章の式（2.2），（2.3），（2.12），（2.17）より次式が成り立つ。

$$u_{\mathrm{CPU}}^{(i)} = \frac{d_{i,\mathrm{CPU}}}{T_i + z_i} \leq \frac{d_{i,\mathrm{CPU}}}{R_{i0}} \leq \frac{d_{i,\mathrm{CPU}}}{d_{i,\mathrm{CPU}} + d_{i,\mathrm{PG}}} \qquad (3.8)$$

したがって，これに式（3.7）を用いることにより次式をえる。

$$u_{\mathrm{CPU}}^{(i)} \leq L_i / (L_i + S) \qquad (3.9)$$

結局ＣＰＵ利用率 u_{CPU} について，

$$u_{\mathrm{CPU}} \leq \min\{1, \sum_i L_i / (L_i + S)\} \qquad (3.10)$$

が成り立つ。したがって，式（3.5）を考えると，各トランザクションについてＸすなわち単位参照あたりのＳＴＰを削減し，$\sum_i L_i / (L_i + S)$ を増大させることが，ＣＰＵ利用率向上のための必要条件であることがわかる。

3.3　ワーキングセット最適化方式（Optimum Working−set Estimator）

多重プログラミング・システムにおいてＳＴＰルールを実現するのがワーキングセット最適化方式ＯＷＥ（Optimum Working−set Estimator）である。ＯＷＥは，ワーキングセット・ポリシーにおけるパラメータ最適化によりＳＴＰを最小化する。ここでワーキングセット・ポリシーとは，仮想時間で過去ウインド・サイズ τ 時間の間に参照したページの集合（ワーキングセット）に実メモリを割り当てるページ・リプレースメント方式である[D3]。ワーキングセット・ポリシーはプログラムの局所参照原理を直接反映できる点が優れており，既にいくつかのシステムで実現されている[C2)R3]。しかし現実には多重プログラミング環境にある各トランザクションごとの最適なウインド・サイズを見出すことが困難なため，これらの実現例ではいずれもシステムに唯一つ設定されたウインド・サイズを各トランザクションが共有する単一ウインド・サイズ方式を採用している。いいかえると，これらの実現例においてＳＴＰ最小化は達成されない。

前頁＊　第２章と同じく，本章でも処理の単位を「ジョブ」でなく「トランザクション」とよぶ。

ＳＴＰは，式（３.４）に示したように，平均ワーキングセット・サイズ$\bar{w}$とページ・フォールト率fとから与えられる。$\bar{w}$，fはともにウインド・サイズτの関数であるから，Ｘすなわち単位参照あたりのＳＴＰもτの関数である。多重プログラミング環境にあるトランザクションiの平均ワーキングセット・サイズ，ページ・フォールト率，ＳＴＰをそれぞれ$\bar{w}_i$，f_i，X_iとするとき，X_i（$i=1,2,\cdots$）を最小化するようそのウインド・サイズτ_i（$i=1,2,\cdots$）を与えるのがＯＷＥの目的である。実際には，$\bar{w}_i$やf_i，したがってX_iはトランザクション実行完了時に定まる事後的な量であり，実行中のトランザクションについてX_iは不明である。現実システムで測定可能なのは実行中のトランザクションの過去のワーキングセット・サイズであるから，その実測値を用いてX_iを推定するという方法が考えられる。いま，$\bar{w}_i$，f_i，X_iの推定量をそれぞれ$\hat{w}_i$，$\hat{f}_i$，$\hat{X}_i$と表わすとき，過去のワーキングセット・サイズの実測値からこれらの値を求めることがＯＷＥの特徴である。一般にプログラムは複数のフェイズから構成され，ひとつのフェイズの中ではページ参照動作は多くの場合定常的なことが知られている[H1)]。ＯＷＥはこのページ参照特性の定常性を仮定し，各トランザクションi（$i=1,2,\cdots$）について任意時点で$\hat{X}_i$の値を最小化するτ_iを設定する。次に具体的にその方法について述べる（なお以下，多重プログラミング環境における或るトランザクションに着目し，簡単のため添字iは省略する）。

仮想時間の軸上で，トランザクション処理実行開始時点から任意の間隔Δごとにサンプルした時点をt_n（$n=1,2,\cdots$）とする。ウインド・サイズτのもとで，t_nにおけるワーキングセット，ワーキングセット・サイズをそれぞれ$W(t_n,\tau)$，$w(t_n,\tau)$とする。厳密なワーキングセット・ポリシーにおいては，任意の時点でページ・リプレースメントが行なわれるが，これは過大なオーバヘッドを招く。本実験ではページ・リプレースメントはサンプル時点のみで行なうとし，最後の参照時刻が（$t_n-\tau$）以前のページはt_nにおいてリプレースされる。この近似は次式で示される（簡単のためτはΔの倍数と仮定する）。

$$W(t_n+\nu,\tau)\cong W(t_n+\nu,\tau+\nu) \qquad (0\leq\nu<\Delta)\ (n=1,2,\cdots) \tag{3.11}$$

式（３.１１）のもとでＳＴＰ最小化を達成するため，まず平均ワーキングセット・サイズ$\bar{w}(\tau)$のt_nにおける推定量$\hat{w}(t_n,\tau)$を，過去のワーキングセット・サイズの実測値を用いて次式で定義する。

$$\begin{cases}\hat{w}(t_n,\tau)\triangleq(1-\alpha)w(t_n,\tau)+\alpha\hat{w}(t_{n-1},\tau)\\ \hat{w}(t_0,\tau)=w(t_1,\tau)\end{cases} \qquad (0<\alpha<1)\quad(n=1,2,\cdots) \tag{3.12}$$

式（３.１２）は次のように書ける。

$$\hat{w}(t_n,\tau)=(1-\alpha)\sum_{j=0}^{n-1}w(t_{n-j},\tau)\alpha^j+\alpha^n w(t_1,\tau) \tag{3.13}$$

すなわち$\hat{w}(t_n,\tau)$は過去Δごとに測定したワーキングセット・サイズの重みつき平均であり，$\bar{w}(\tau)$の推定量である。算術平均をとらなかった理由は，実環境で生ずるページ参照特性の変化に追従し易

くするためである。

次にページ・フォールト率$f(\tau)$のt_nにおける推定量$\hat{f}(t_n,\tau)$を次式で定義する。

$$\hat{f}(t_n,\tau) \triangleq \{\hat{w}(t_n,\tau+\Delta)-\hat{w}(t_{n-1},\tau)\}/\Delta \qquad (n=1,2,\cdots) \tag{3.14}$$

いま，ウインド・サイズτのもとで$[t_{n-1}+1,\ t_n]$の間に生ずるページ・フォールト回数を$g(t_n,\tau)$とすると，式(3.11)のもとで，これは次のように与えられる。

$$g(t_n,\tau) = w(t_n,\tau+\Delta) - w(t_{n-1},\tau) \tag{3.15}$$

式(3.13)～(3.15)より次式が成り立つ。

$$\begin{aligned}
\hat{f}(t_n,\tau) &\triangleq \Big[(1-\alpha)\sum_{j=0}^{n-1} w(t_{n-j},\tau+\Delta)\alpha^j + \alpha^n w(t_1,\tau+\Delta) \\
&\quad -(1-\alpha)\sum_{j=0}^{n-2} w(t_{n-j-1},\tau)\alpha^j - \alpha^{n-1} w(t_1,\tau)\Big]/\Delta \\
&= \Big[(1-\alpha)\sum_{j=0}^{n-2}\{w(t_{n-j},\tau+\Delta)-w(t_{n-j-1},\tau)\}\alpha^j \\
&\quad + (1-\alpha)w(t_1,\tau+\Delta)\alpha^{n-1} + \alpha^n w(t_1,\tau+\Delta) \\
&\quad - \alpha^{n-1} w(t_1,\tau)\Big]/\Delta \\
&= (1-\alpha)\sum_{j=0}^{n-2}\{g(t_{n-j},\tau)/\Delta\}\alpha^j \\
&\quad + \alpha^{n-1}\{w(t_1,\tau+\Delta)-w(t_1,\tau)\}/\Delta
\end{aligned} \tag{3.16}$$

すなわち，$\hat{f}(t_n,\tau)$はウインド・サイズτのもとで過去Δごとに測定したページ・フォールト率（単位時間あたりのページ・フォールト発生回数）の重みつき平均である。なお，上記議論と関連して一般に，

$$f(\tau) = \frac{d\bar{w}(\tau)}{d\tau} \tag{3.17}$$

が成立することを付記する。[D5)]

以上より結局，$X(\tau)$のt_nにおける推定量$\hat{X}(t_n,\tau)$は次式で定義される。

$$\hat{X}(t_n,\tau) \triangleq \hat{w}(t_n,\tau) + S\hat{f}(t_n,\tau)\hat{w}(t_n,\tau) \qquad (n=1,2,\cdots) \tag{3.18}$$

式(3.12)，(3.14)，(3.18)より$\hat{X}(t_n,\tau)$を求めるためには，サンプル時点t_n（$n=1,2,\cdots$）においてτの関数$w(t_n,\tau)$が測定できればよい。このため，各ページに対し，それが最後に参照されてから経過した仮想時間を示す「非参照カウンタ」を設ける。$w(t_n,\tau)$のτについての最小測定単位はΔとし，測定点を，

$$\tau_m = m\Delta \qquad (m=1,2,\cdots) \tag{3.19}$$

と書く。いま，t_nにおいて，各ページが$[t_{n-1}+1,\ t_n]$の間に参照されたか否かを調べ，参照されたページについてはその非参照カウンタをゼロクリアし，参照されなかったページについてはこれ

に１を加える（参照，非参照の区別は，参照ビットを備えた計算機では容易である）。このとき，$w(t_n, \tau_m)$は，その非参照カウンタの値がmより小である相異なるページ数として求められる。

以上より，$\min_m \hat{X}(t_n, \tau_m)$に対応する$\tau$を$t_n$におけるウインド・サイズとすることができる。ただしここで，求められたウインド・サイズは限られた範囲における（局所的な）最適値である。t_{n-1}において設定したウインド・サイズをτ_Mとすると，t_{n-1}の時点で非参照カウンタがM以上のページはリプレースされる。したがってt_nで測定可能なのは$w(t_n, \tau_m)$（$m = 1, 2, \cdots, M+1$）であり，$\hat{X}(t_n, \tau_m)$は$0 \leq \tau \leq \tau_M$の範囲でしか求められない。最適値がこれ以外の範囲にある場合の対策として，次の２条件が成立するときウインド・サイズをτ_{M+1}に増加し，t_nでページ・リプレースメントは行なわない。

$$\min_{1 \leq m \leq M} \hat{X}(t_n, \tau_m) = \hat{X}(t_n, \tau_M) \qquad (3.20)$$

$$\hat{X}(t_n, \tau_{M-1}) \gg \hat{X}(t_n, \tau_M) \qquad (3.21)$$

多くの場合，Xのτによる変化は比較的単純なので[C3)G2)G3)]，このような処理で最適値に到達する可能性は大である。

なお，多重プログラミング環境における各トランザクションに対し仮想時間でΔごとに$w(t_n, \tau_m)$を測定することは，次のようにして近似的に実現できる。ＯＷＥは実時間で周期的に起動されて割り込み，各トランザクションのＣＰＵ実行時間累計を用いて，前回測定以来のＣＰＵ実行時間を算定する。これがΔ以上であるトランザクションについて，そのワーキングセット*内の各ページを調べ，$w(t_n, \tau_m)$を測定する。このような処理によれば，Δのサンプリングにおける最大誤差はＯＷＥの起動周期に等しくなる。

以上がＳＴＰルールを実現するためのＯＷＥの概要である。ただし学習を行なうためにはある程度の時間が必要なので，短小なトランザクションに対してはＯＷＥの効果は小さい。すなわちＯＷＥの主対象は，処理能力に多大な影響を与える長大なトランザクションである。

3.4 実験と評価

3.4.1 Space Time Productの最小化

ワーキングセット最適化方式ＯＷＥをオペレーティング・システムＶＯＳ３において実現し，大型計算機ＨＩＴＡＣ Ｍ－１８０を用いて実験を行なった。ＯＷＥの最終目標はシステム全体の性能向上にあるが，まず直接の制御目標であるＳＴＰ最小化の達成可能性を検証する必要がある。したがって，処理能力の実測評価については次節で述べることにし，本節ではＯＷＥによるページ参照特性の学習能力，ＳＴＰ最小化の達成能力などを検討する。このため，理論的にページ参照特性が求められる人工的なプログラムを作成し，これを用いた。

---

* 本論文では，トランザクション間で共用するページは考察の対象外とする。

本実験で用いたトランザクションは，SLRUM(Simple Least Recently Used Model)[S11]にしたがってページを参照する。すなわちΩページからなるプログラムにおいて，スタック距離がjの位置にあるページは$q(j)$ ($j=1,2,\cdots,\Omega$)の確率で参照される。$q(j)$は時間によらず，その和$\sum_{j=1}^{\Omega} q(j)$は1に等しい。$q(j)$に様々の値を設定することにより，各種のページ参照特性をもつ定常的なプログラムが作成できる。

R.Turner[T4]らは，$q(j)$が与えられたとき$\bar{w}(\tau)$を求める次の方法をみちびいた。いま，τの間に参照される相異なるページ数すなわちワーキングセット・サイズがh ($h=1,2,\cdots,\Omega$)である確率を$P(h,\tau)$とすると，

$$P(h,\tau)=P(h-1,\tau-1)\sum_{j=h}^{\Omega} q(j)+P(h,\tau-1)\sum_{j=1}^{h} q(j) \qquad (3.22)$$

が成り立つ。また，

$$P(h,1)=\begin{cases} 1 : h=1 \\ 0 : h=2,3,\cdots,\Omega \end{cases} \qquad (3.23)$$

であるから，式(3.22)，(3.23)より$P(h,\tau)$ ($h=1,2,\cdots,\Omega$) を求めることができる。$\bar{w}(\tau)$は次式で与えられる。

$$\bar{w}(\tau)=\sum_{h=1}^{\Omega} h\,P(h,\tau) \qquad (3.24)$$

さらに式(3.17)を用いると$f(\tau)$が求まるので，結局式(3.4)を算定することができる。

3種類のスタック距離分布を設定し，これにしたがってページを参照するプログラム，ＪＯＢⅠ，ＪＯＢⅡ，ＪＯＢⅢを作成した。上記のようにしてこれらの$\bar{w}(\tau)$，$f(\tau)$，$X(\tau)$を求めた結果を図3.2に示す。ＪＯＢⅠは$X(\tau)$が極小値をもつ場合，ＪＯＢⅡ，ＪＯＢⅢはそれぞれ$X(\tau)$が単調減少，単調増大の場合であり，プログラムの大きさΩはいずれも50ページである。

ＪＯＢⅠ～Ⅲを多重プログラミング環境で実行し，ＯＷＥのもとでのワーキングセット・サイズを観測した。ＯＷＥの起動周期は実時間で約100,000ステップ，サンプリング間隔Δは仮想時間で200,000ステップとした。また，式(3.12)において学習の重み係数αは0.5，式(3.4)，(3.18)のページ・フォールト処理時間Sは80,000ステップとした。

ワーキングセット・サイズの測定はＯＷＥによる正の効果の検証であるが，その負の効果としてＯＷＥ実行にともなうＣＰＵオーバヘッドを測定した。いま，プログラムのページ参照特性は図3.2に示したように既知である。したがってプログラムの実行を通じ，$X(\tau)$を最小化する最適なウインド・サイズを学習によらず固定値として与えることが可能である。この様な理想的な場合について実測を行ない，ＯＷＥのもとで学習により最適値を求めた場合のＣＰＵオーバヘッドを測定した。

ＪＯＢⅠ～Ⅲの各々について，ＯＷＥによりウインド・サイズτを与えたときの実測値と，理論的に求めた最適値との比較結果を表3.1に示す。表3.1は，8分間の測定を通じ，仮想時間200,000ステップごとに測定したウインド・サイズτの平均値，ワーキングセット・サイズの平均値$\bar{w}$，これ

に対応する単位参照あたりのSpace Time Product Xと，それぞれの理論的最適値との比較である。ウインド・サイズの実測値と理論的最適値とではやや差があるが，一般に最適点付近では$X(\tau)$の変動巾は小さいことから[C3)G2)G3)]，一応許容できる範囲にある。実際に問題となるのは，制御パラメータτよりむしろワーキングセット・サイズである。$\overline{w}$，Xについての実測値と理論的最適値との差異は，それぞれ２～５％，１～１０％であり，各種のページ参照特性をもつプログラムに対してＯＷＥの制御効果を確認した。

なお，これらの結果は，３３節で述べた方法によりプログラムのページ参照動作をマクロ・レベルで解析できることを示している。従来，ページ参照動作は専らアドレス軌跡を用いてミクロ・レベルで解析されてきたが[G2)M1)R2)S8)]，この方法は処理するデータ量が多大なため，実際には数秒間程度の特性しか得られない。また多重プログラミング環境でのページ参照動作の解析にも不適当である。本論文で述べた方法は，多重プログラミング環境における数十秒～数十分間のページ参照動作の平均的特性の解析に適用可能である。

表３１には，ＯＷＥのＣＰＵオーバヘッドの測定結果をも示した。ＯＷＥの処理においてはページごとに行なう処理の占める部分が大なので，ＣＰＵオーバヘッドは一般にプログラムのワーキングセット・サイズの増加関数となる。したがってワーキングセット・サイズ最大のＪＯＢⅡと最小のJOBⅢについて，それぞれ測定を行なった。ＪＯＢⅡのみを５本，５多重で８分間実行した場合について，ＯＷＥのもとでのオペレーティング・システムのオーバヘッドと，予め最適なウインド・サイズを与えた理想的な場合のそれとを比較した。両者の差がＯＷＥ実行にともなうＣＰＵオーバヘッドであり，これは，1.16％であった。ＪＯＢⅢについて同様に比較すると0.35％であった。ＪＯＢⅠについては両者の間の値をとる。

前述のように，$X(\tau)$は最小値付近ではτの変化に対して余り敏感ではないので，ページ参照動作が定常的であればひんぱんにτを調節する必要はない。すなわち，制御系としての安定性を増すとともにＣＰＵオーバヘッドの削減が可能である。$\hat{X}(t_n, \tau_m)$の最小値を求めてτを変える際，適当に閾値を設けることにより，τの過度の変動を防止することができる。

3.4.2　処理能力に関する実測

第３２節で述べたように，ＳＴＰの削減は一般に処理能力向上に寄与する。本節では，ＯＷＥのもとでの処理能力向上効果を検討するためにＨＩＴＡＣ　Ｍ－１８０システムの上でベンチマーク・ワークロードを用いて行なった実測評価結果について述べる。

ベンチマーク・ワークロードは，翻訳，連係，実行フェイズを持つ４種類のＦＯＲＴＲＡＮプログラムＢＪ１～ＢＪ４からなり（ただしＢＪ４は実行フェイズのみ），いずれも処理完了までに２５秒以上のＣＰＵ実行時間を必要とするかなり長大なトランザクションである。ＢＪ１～ＢＪ４のページ参照特性は必ずしも定常的とは限らないが，近似的にもとめた平均ワーキングセット・サイズを図３３に示す。ここで図３３には線形の平均ワーキングセット・サイズ曲線を有する「キング・サイズド・プロ

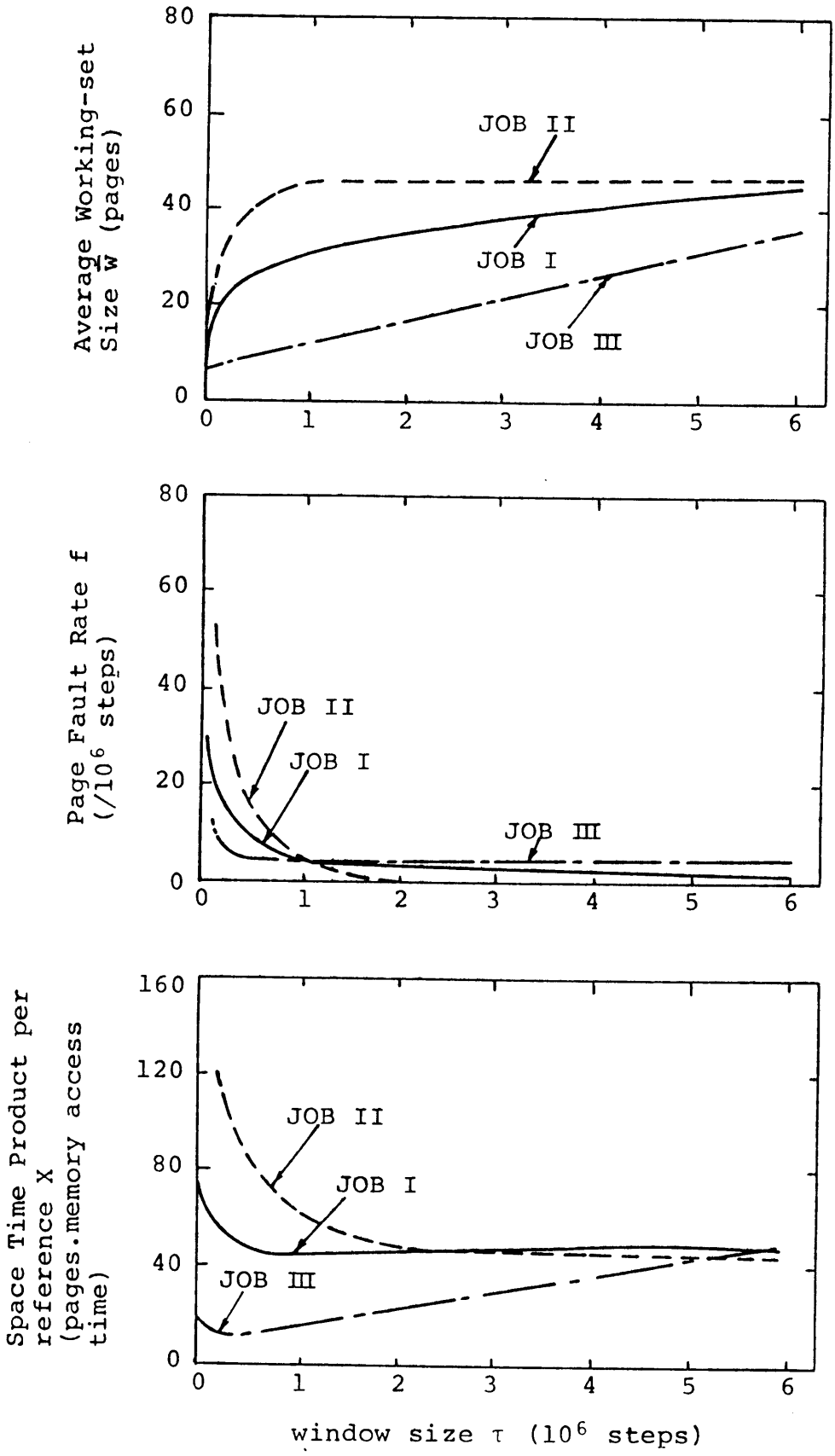


図3.2　実験で用いた定常的プログラムの諸特性

The characteristics of the synthetic programs investigated in the experiment

表3.1　**OWE**の制御効果および制御オーバヘッドに関する実験結果（上段：実測値，下段：理論的最適値）

Experimental results of the OWE's control effects and overhead (upper row: observed value, lower row: theoretically optimum value)

PROGRAM	JOB I	JOB II	JOB III
AVERAGE WINDOW SIZE (10^6 steps)	0.82 1.20	1.76 2.60	0.56 0.30
AVERAGE WORKING-SET SIZE (pages)	32.5 34.2	49.0 50.0	11.1 10.7
SPACE TIME PRODUCT PER REFERENCE (pages·memory access time)	48.5 48.1	55.1 50.0	15.8 15.2
CPU OVERHEAD BY OWE (%)	——	1.16 ——	0.35 ——

(1 page = 4 K bytes)

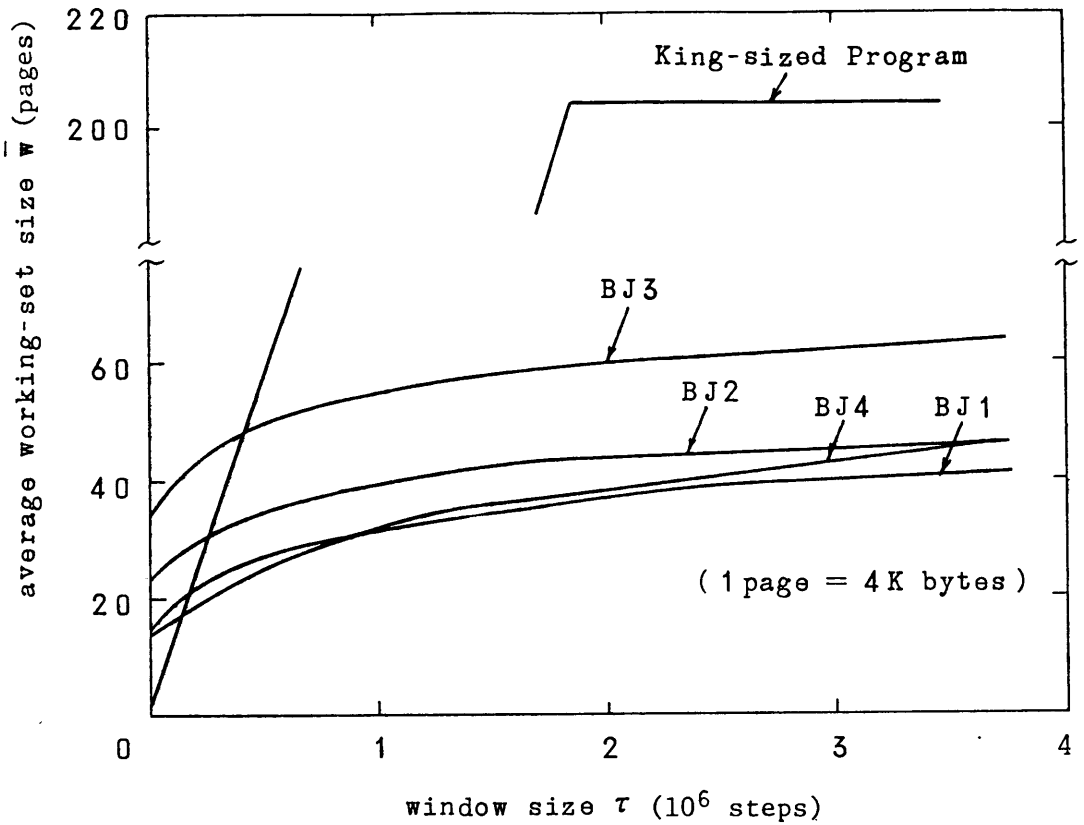


図3.3 OWEの実測評価に用いたベンチマーク・ワークロード

The benchmark workload used in the measurement of OWE

グラム」をも図示したが，これは実験結果の解析を容易にするためにベンチマーク・ワークロードにつけ加えた定常的プログラムである。キング・サイズド・プログラムは一定の割合で逐次新しいページを参照していき，最後のページに達すると再び最初のページに戻る。ＳＴＰの観点からは，この種のトランザクションには最小限の実メモリを割り当てることが望ましい。これは，ページ・フォールトの発生率がワーキングセット・サイズによらず一定のためである*。したがってＯＷＥのもとで，キング・サイズド・プログラムのワーキングセット・サイズは減少する筈である。一方従来方式では，多重プログラミング環境における全てのトランザクションが単一のウインド・サイズを共用するので，キング・サイズド・プログラムに対するこのような制御は不可能である。以上より，ＯＷＥによる処理能力の向上が期待できる。

しかし，処理能力向上の程度は実メモリの利用状況に依存する。すなわち，第２章の議論が示すように，実メモリ管理が性能に多大な影響を与えるのは実メモリがボトルネックとなる場合に限られる。メモリ・ネック状況は，大規模な行列演算など多量のメモリを用いるトランザクションを処理する場合などにしばしば発生する。キング・サイズド・プログラムは多量のメモリを用いるトランザクションの例であり，これを用いてメモリ・ネック状況における実測データを得ることができる。

実測に際し，ジョブ・イニシエータ数（仮想空間数）は６とし，そのうちの１つはキング・サイズド・プログラムが専有する。ベンチマーク・プログラムＢＪ１～ＢＪ４は必要に応じて繰返し実行し，約１０分間の測定を通じて６空間は常にアクティブに保った。

ＯＷＥのもとでの処理能力を，従来の単一ウインド・サイズ方式と比較した結果を図３.４に示す。ここで従来方式におけるウインド・サイズは，この方式のもとで最高の処理能力を達成できるよう配慮を加えて設定した。すなわち，従来方式における単一のウインド・サイズを，ページ入出力装置の負荷に応じて動的に調節した。一般にこのような調節により処理能力は向上することが知られている。[D7)D8)L2)]

しかし，従来方式におけるこのようなウインド・サイズの調節にもかかわらず，ＯＷＥのもとではより高い処理能力が得られることが図３.４に示されている。ここで実メモリ容量２Ｍバイト，３Ｍバイトはそれぞれメモリ・ネック，非メモリ・ネックの場合である。メモリ・ネックの場合，従来方式と比較して，ＯＷＥのもとでＣＰＵ利用率の顕著な向上がみとめられる。一方，非メモリ・ネックの場合は両者の差異は明らかではない。非メモリ・ネック状況では，処理能力は実メモリ管理以外の要因で決定されるためである。

図３.４に示した総資源サービス量（total service amount）は，ファイル入出力を含み，オペレーティング・システムのＣＰＵオーバヘッドを除外した値であり，したがって処理能力のより正確な表現である。総資源サービス量は，ＣＰＵの10,000命令，ファイル入出力１回をそれぞれ10service units（su）にあたるとして，システムの供給した資源サービスの総量を与えるものである。（より詳細には，第５章の式（5.2）で定義する資源サービス量累計を各トランザクションについて

---

* 巨大なプログラムなので，全参照ページを実メモリ内に収容することはないとする。

加えることにより求まる。)ＯＷＥにより，従来方式と比べ，メモリ・ネック（２Ｍバイト）の場合は約１９％，非メモリ・ネック（３Ｍバイト）の場合は約１％の総資源サービス量の増加，すなわち処理能力向上がみとめられる。

ＯＷＥによる処理能力向上を裏づける諸測定データが図３５に示されている。ＯＷＥのもとでのキング・サイズド・プログラムのワーキングセット・サイズは，従来方式のもとでのそれより大巾に削減されている。この結果，メモリ・ネック（２Ｍバイト）の場合は，多重度（実メモリ内にスワップ・インされているトランザクション数）およびＣＰＵ実行可能なトランザクション数が増大している。このことが，処理能力向上の原因と考えられる。一方，非メモリ・ネック（３Ｍバイト）の場合には，多重度，ＣＰＵ実行可能なトランザクション数とも差異は小さく，ＯＷＥの効果は顕著ではない。

３５　結　　言

仮想メモリ・システムにおいて，ページング・ネックを防止するための実メモリ管理につき論じ，「ワーキングセット最適化方式ＯＷＥ（Optimum Working－set Estimator）」を新しく提案するとともに，実験を行なった。本方式ＯＷＥは，日立のオペレーティング・システムＶＯＳ３において実用化の予定である。本章で得られた結論，提案した方式の特徴などを以下にまとめる。

(1) 実メモリ割当て量の評価基準として従来より用いられるＳＴＰ（Space Time Product）とシステム全体の性能との関係を論じた。ＳＴＰの削減が，ページング・ネックを防ぐとともに処理能力向上に寄与することを明らかにした。

(2) 本方式ＯＷＥは，多重プログラミング環境のもとで，各トランザクション(＝ＴＳＳコマンドないしバッチ・ジョブ）のＳＴＰを推定し，ワーキングセットのウインド・サイズの調節により，その最小化を達成する。このため，ページ参照動作の定常性を仮定し，トランザクション実行中にそのページ参照特性を学習する。なお，ＳＴＰ最小化に関する従来の扱いはほぼ理論的なものに限られ，実システムでの実験報告は知られていない。

(3) 本方式ＯＷＥをＶＯＳ３において実現し，大型計算機ＨＩＴＡＣ Ｍ－１８０ を用いて実測を行なった。まず，最小のＳＴＰを与える最適実メモリ割当て量が理論的に求められる定常的なトランザクションを作成し，ＯＷＥのもとでの実測値と理論的最適値とを比較した。実メモリ割当て量について，実測値の理論的最適値に対する比は0.95～1.04であった。またＳＴＰについて，その比は1.01～1.10であり，ＯＷＥのページ参照特性の学習能力とＳＴＰ最小化達成能力とを確認した。ＯＷＥ実行によるＣＰＵオーバヘッドは0.35～1.16％であった。

(4) さらに，必ずしもページ参照動作が定常的とは限らないベンチマーク・ワークロードを用いて，処理能力についての実測を行なった。従来方式に比べ，ＯＷＥのもとでは処理能力が１～１９％向上することを確かめた。特に実メモリがボトルネックの場合，顕著な処理能力向上効果が観察された。

本章の議論は，基本的にプログラムのページ参照動作における局所参照原理（locality princi-

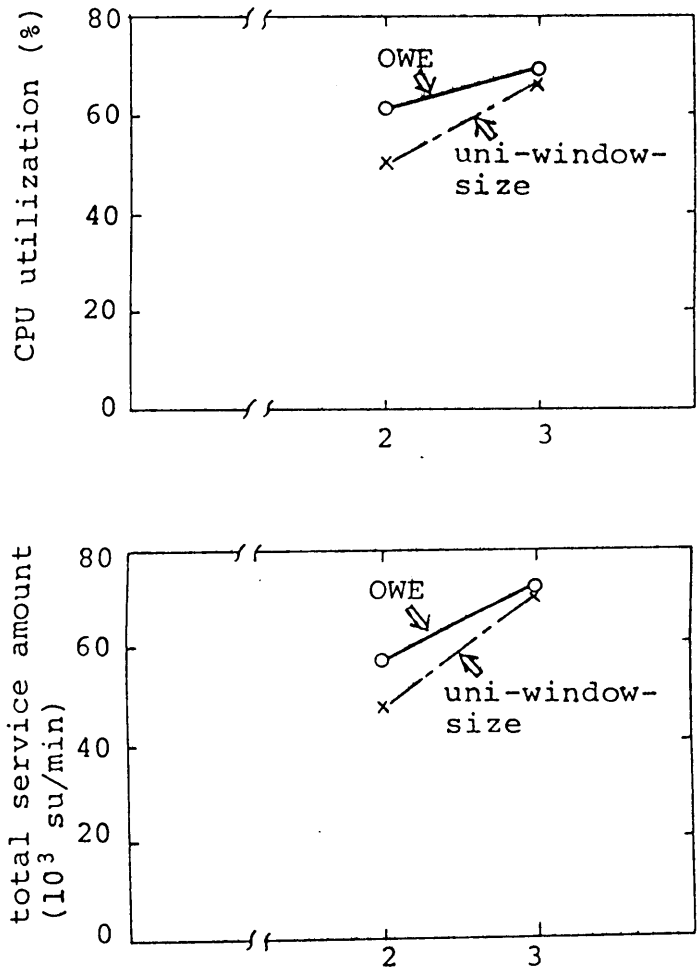


図 3.4 **OWE** と従来方式との処理能力実測比較

Throughput comparison between the OWE and conventional uni-window-size scheduler.

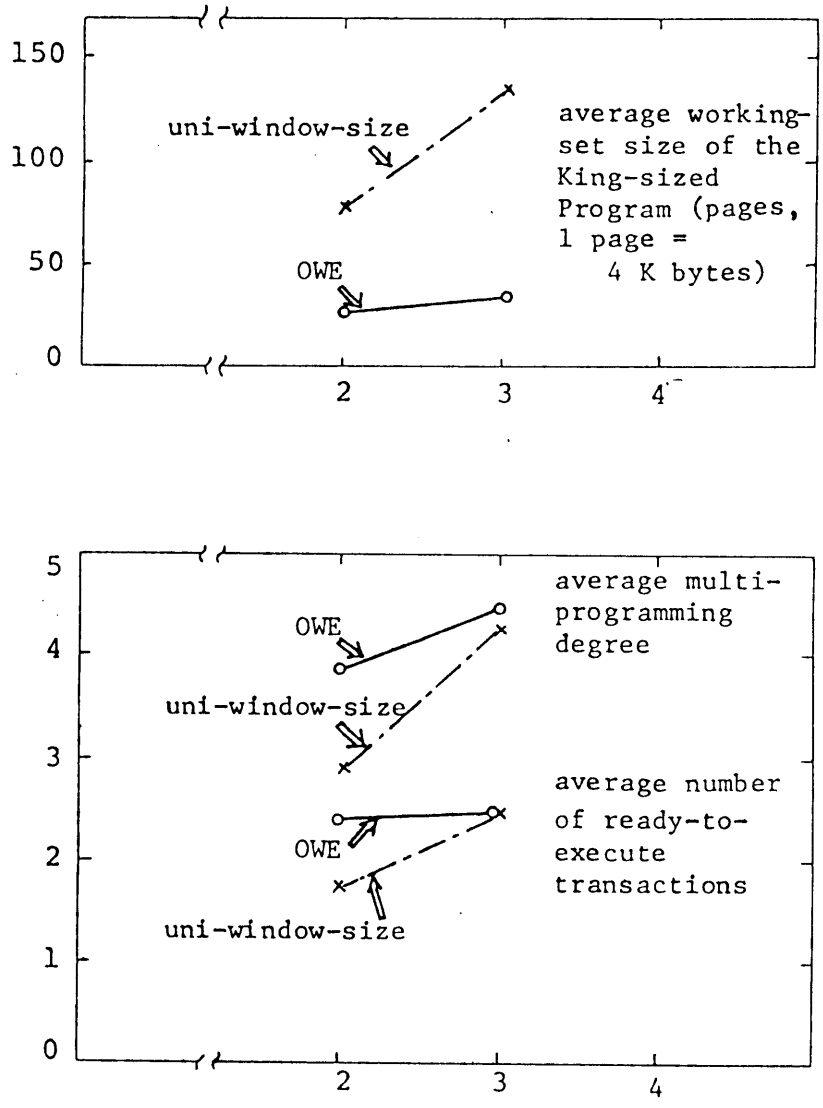


図 3.5　OWEと従来方式の処理能力に関連する諸実測データ

Some performance factors affecting the throughput; under the OWE and conventional uni-window-size scheduler

ple)にもとづいている。この性質はプログラムのプロシジャ部分については成立することが知られているが，データ部分については必ずしも成立しない[05]。データ部分に関する実メモリ管理の研究は，今後検討さるべき分野である。

第4章　仮想メモリ・システムの二次メモリ最適管理方式

4. 仮想メモリ・システムの二次メモリ最適管理方式

4.1 概　　要

前章では，仮想メモリ・システムの性能を実メモリ管理の観点から扱った。本章ではこれを二次メモリ管理の観点から論ずる。

仮想メモリ・システムにおいては，アドレス空間上のページは磁気ディスク，磁気ドラムなどの二次メモリ媒体に蓄積されており，実メモリ内にはジョブ実行のため当面必要なページのみが収容される。処理の進行にともない，新たに必要となったページの読みこみ，不要になったページで内容が変更されたものの書き出しなどのページ入出力が実行される。仮想メモリ・システムの性能上の問題点はこのページング処理のオーバヘッドにあり，これが過大な場合，第2章の議論により示されるようにページング・ネックとなり性能が低下する。ページング処理のオーバヘッドは，ページ入出力実行時間の短縮により削減が可能である。

二次メモリ管理方式は，アドレス空間上のページを二次メモリ媒体上のどこに格納するかのアルゴリズムを与えるものであり，したがってページ入出力実行時間は二次メモリ管理方式により多大な影響を受ける。しかし従来，仮想メモリ・システムの性能は専ら実メモリ管理のみの観点から扱われ，二次メモリ管理に関する報告は知られていない。

以上より本章では，ページ入出力実行時間短縮を目的として，従来方式である「固定方式」と「浮動方式」の効率を比較解析するとともに，「集中方式（Concentrating Technique）」とよぶ新しい二次メモリ管理方式を提案する[N7)N8)]。固定方式とは，アドレス空間上のページの二次メモリ上の格納場所（スロット）がジョブ実行を通じて常に不変の方式であり，ページ書き出しの際には元のスロット上に重ねて書き出す。一方浮動方式では，ページ書き出しの際に必ずしも元のスロットではなく，ページ書き出し時間が最小となるようなスロットが新たに選ばれてこの上に書き出される。換言すれば，或るページが書き出されるスロットの選択の自由度に着目すると，固定方式，浮動方式はそれぞれ自由度最小，最大の方式に対応する。一般に自由度増大とともにページ書き出し時間は減少するが，ページ読みこみ時間は増大の傾向を持つ。したがって総合的な観点からページ入出力実行時間を最小化する最適な自由度が存在する。この最適自由度を実現するのが，提案する集中方式である。

本章では4.2節で，アドレス空間上のページを二次メモリ上のスロットに写像するスロット割当ての問題を定義する。4.3節では固定方式と浮動方式のページ入出力実行時間を解析的に比較する。4.4節ではページ入出力実行時間を最小化する最適自由度を求め，これを与える集中方式を提案する。4.5節では，集中方式をオペレーティング・システムＶＯＳ３において実現することにより得られた実測評価結果について述べる。

4.2 二次メモリのスロット割当て

用語および前提条件を整理し，本章で扱う問題を呈示する。二次メモリ媒体上のページ単位に区分

された情報格納場所（メモリ・スペース）を「スロット」とよぶ。アドレス空間上の各ページについて，それが蓄積される二次メモリ上のスロットを決定するのが「スロット割当てアルゴリズム」であり，本章の目的はページ入出力実行時間を最小化するようなスロット割当てアルゴリズムを与えることにある。

スロット割当てには，ジョブ実行に先立って行なわれる初期割当てと，ジョブ実行中に内容の変わったページを書き出す実行時の割当て，の2種類がある。（内容の変わらないページは，ページ転送オーバヘッドの不必要な増加を防ぐため書き出さないとする。）書き出し時に常に元のスロット上に書き出す固定方式では当然実行時の割当ては影響しないが，相異なるスロット上への書き出しをゆるす浮動方式では実行時の割当てがページ入出力効率に与える影響を無視できない。本章では理論的にこれを解析し，浮動方式の効率を固定方式と比較する。

実メモリ管理方式としてワーキングセット・ポリシー[D3]を仮定する。このとき，ジョブ実行中のページ書き出しの契機は，(1)ワーキングセットから外れ，内容が変わったページのページ・アウト，(2)ジョブ実行を一時中断するために，ワーキングセットの中で内容の変わったページ全てを一括して書き出すスワップ・アウト，の2つに分類できる。複数ページを同時にまとめて書き出す際，これらを二次メモリ上の連続したスロット群の上に書き出した方が効率がよい。このひとまとまりのスロット群を「ブロック」とよぶ。もちろん，既に他のページに割り当てられているスロットは割当て不可能だが，現実システムではページ入出力効率を上げるため二次メモリのスペースは余裕をもって与えることが多い。したがって本論文のモデルでは，二次メモリ上に十分な未使用スペースがあり，書き出し時に常に連続したスロット群の割当てが可能と仮定する。

浮動方式のページ入出力効率を解析するにあたり，ひとつのジョブのプログラムに割り当てられたスロット群が定常状態でいくつのブロックに分かれるかを求め，それらブロック群の二次メモリ上での散在度を評価する必要がある。4.3.2でこの散在度を算定し，浮動方式のページ入出力実行時間を求める。また4.4で最適な散在度を与える集中方式について述べる。

なお，二次メモリ媒体としては，スロット割当てアルゴリズムが入出力効率に与える影響が大きく，かつ経済性の点から現実システムで多用されている可動ヘッドの磁気ディスクを仮定する。アプリケーションとしては，インタラクティブなＴＳＳジョブを想定する。この理由は，この種のジョブ実行にともなうスワッピングが二次メモリ媒体の負荷の大半を占める[B10]ためである。また本章では，「ジョブ」は複数のＴＳＳコマンドの連鎖よりなる処理単位を表わし，各々のＴＳＳコマンドは第2章で定義したように「トランザクション」とよぶ。トランザクションのワーキングセットは，トランザクション到着時に一括してスワップ・インされ，内容が変更された部分が実行終了時に一括してスワップ・アウトされる。

また，システムの動作環境は，多端末が同時に稼動中の多重プログラミング環境を想定し，それらのジョブの用いる二次メモリは磁気ディスク上の連続したL個のシリンダを占めるとする。ただし，ジョブ間で共用するページの影響は無視し，1つのジョブで参照するページは1シリンダに収納可能

とする。通常，インタラクティブなＴＳＳコマンドのプログラム・サイズは２０～３０ページ程度（１ページ＝４Ｋバイト）であり，現実の磁気ディスクは１シリンダに６０ページ以上格納可能[H4]なので，この仮定は妥当である。また，ページ入出力のためのシーク動作開始時点でのディスク・ヘッドの位置は一様分布にしたがうと仮定する。複数ページを入出力実行するとき，これらをシリンダ番号についてソートし，シリンダ番号の若い順からディスク・ヘッドのスキャンによりアクセス[D2)T2)T3)]するスキャン方式を仮定する。ただし，シリンダ内でのソートは行なわず，ランダム・サーチを実行する。

次にスロット割当てアルゴリズムの効率を比較するための評価関数を設定する。トランザクションはスワップ・インされて実行され，しかるのちスワップ・アウトされる。この間にページ・フォールト（ワーキングセットへ加わるページの読みこみ）とページ・アウト（ワーキングセットから外れ，内容が変化したページの書き出し）とが行なわれる。本論文では，１つのトランザクション実行にともなうページ入出力実行時間を評価関数とし，式（4.1）で定義する。

$$C \triangleq n_{SO} C_{SO} + n_{SI} C_{SI} + n_{PO} C_{PO} + n_{PF} C_{PF} \qquad (4.1)$$

C_{SO}，C_{SI}，C_{PO}，C_{PF} はそれぞれスワップ・アウト・コスト，スワップ・イン・コスト，ページ・アウト・コスト，ページ・フォールト・コストであり，各行為１回あたりの平均アクセス時間を表わす（ページ入出力実行時間は転送時間とアクセス時間との和であるが，前者はスロット割当てアルゴリズムにより影響されないので，アクセス時間分布のみに着目すれば十分である）。n_{SO}～n_{PF} は１トランザクションあたりの各行為の頻度を表わす。いま，トランザクションは実行の途中でスワップ・アウトされることはないと仮定すると，$n_{SO} = n_{SI} = 1$，である。また，実際のオペレーティング・システムではページ・アウトはまとめて行なわれる。すなわち，ワーキングセットから外れた直後でなく，効率を上げるためにある程度一括してページ・アウトを行なう。したがっていま，短小なトランザクションを想定しているので，$n_{PO} = 1$，と仮定する（この仮定については，4.4.3でふれる）。さらに式（4.1）の第４項は，スロット割当てアルゴリズムにより影響されない。これは第１に，多重プログラミングのもとでページ・フォールトが発生したとき，当ページに対応するスロットの位置ならびにディスク・ヘッドの位置は，ともにランダムと考えられること，第２にページ・フォールトの発生率はプログラムのページ参照特性と実メモリ管理方式とから決定されるためである。以上より，式（4.1）をさらに次のように再定義する。

$$\begin{cases} C \triangleq C_{SO} + C_{SI} + C_{PO} \\ C_{SO} \triangleq F_{SO} + G_{SO} \\ C_{SI} \triangleq F_{SI} + G_{SI} \\ C_{PO} \triangleq F_{PO} + G_{PO} \end{cases} \qquad (4.2)$$

F，Gはそれぞれシーク時間，サーチ時間の期待値を表わす。Cをトータル・コストとよび，これを最小化するスロット割当てアルゴリズムを最適なアルゴリズムとする。

トータル・コストCとシステムの処理能力，応答性との関係は，第２章，第３章の議論から次のように得られる。Cの削減は，ページ入出力装置すなわち二次メモリ媒体の使用時間 d_{PG} の削減に対応

する。その効果は2つに分けることができる。第1に，本章で対象とした短小なＴＳＳトランザクションについては，d_{PG}の削減は平均応答時間Tの短縮に直接つながる（第2章参照）。第2に，d_{PG}の削減によりページ入出力装置の負荷は減少するので，ページング・ネックに陥ることなくワーキングセット・サイズwを削減し多重度を増加することが可能となる（第3章参照）。これは，長大なトランザクションを含めた処理能力の向上を意味する。

4.3 固定方式と浮動方式の比較解析

本節では，代表的な従来方式である固定方式（Fixed Technique）と浮動方式（Floating Technique）との比較について述べる。

4.3.1 固定方式の性能解析

固定方式とは，既述のように各ページに対し，ジョブ実行開始から終了まで常に同一のスロットを割り当てる方式である。商用のオペレーティング・システムでは，日立のＶＯＳ２[H5]，ＩＢＭ社のＶＳ１[I1]などがこれを採用している。ジョブ実行中に内容が変更されたページを書き出すときは，以前当該ページに割り当てられていたスロットに書き出す（内容変更のないページは書き出さない）。１つのジョブで参照するページ群に対する割当ては，初期割当て時に１シリンダ上になされる。

固定方式におけるトータル・コストC^{FX}を求めるための諸元を下記に導く（以下，C，F，Gなどの右肩の「ＦＸ」は固定方式を表わす）。

(1) スワップ・アウト・コスト C_{SO}^{FX}

本論文ではディスクのシーク時間を図4.1に示す一次関数（$=F_0+K\cdot$シーク・シリンダ数）で近似する。F_0は最小シーク時間，Kは比例定数，Lは二次メモリの占める総シリンダ数である。このとき，期待シーク時間 F_{SO}^{FX}は次式で与えられる。

$$F_{SO}^{FX} = F_0 + KL/3 \qquad (4.3)$$

ジョブが参照するページの蓄積されたシリンダの位置，およびスワップ・アウト発生時のディスク・ヘッドの位置はともに$1\sim L$の一様分布にしたがうとの仮定より，式（4.3）が得られる。[D2] また期待サーチ時間 G_{SO}^{FX}は次式で与えられる。

$$G_{SO}^{FX} = w\,p\cdot D/2 \qquad (4.4)$$

wは，ワーキングセット・サイズである。pは，１トランザクションの間にワーキングセット内のあるページがチェンジする（書きこみで内容が変わる）確率である。Dはディスクの回転時間であり，$D/2$はランダム・サーチの期待サーチ時間を表わす。

(2) スワップ・イン・コスト C_{SI}^{FX}

スワップ・アウトの場合と同様にして，期待シーク時間 F_{SI}^{FX}，期待サーチ時間 G_{SI}^{FX}は，それぞれ式（4.5），（4.6）で与えられる。

$$F_{SI}^{FX} = F_0 + KL/3 \qquad (4.5)$$

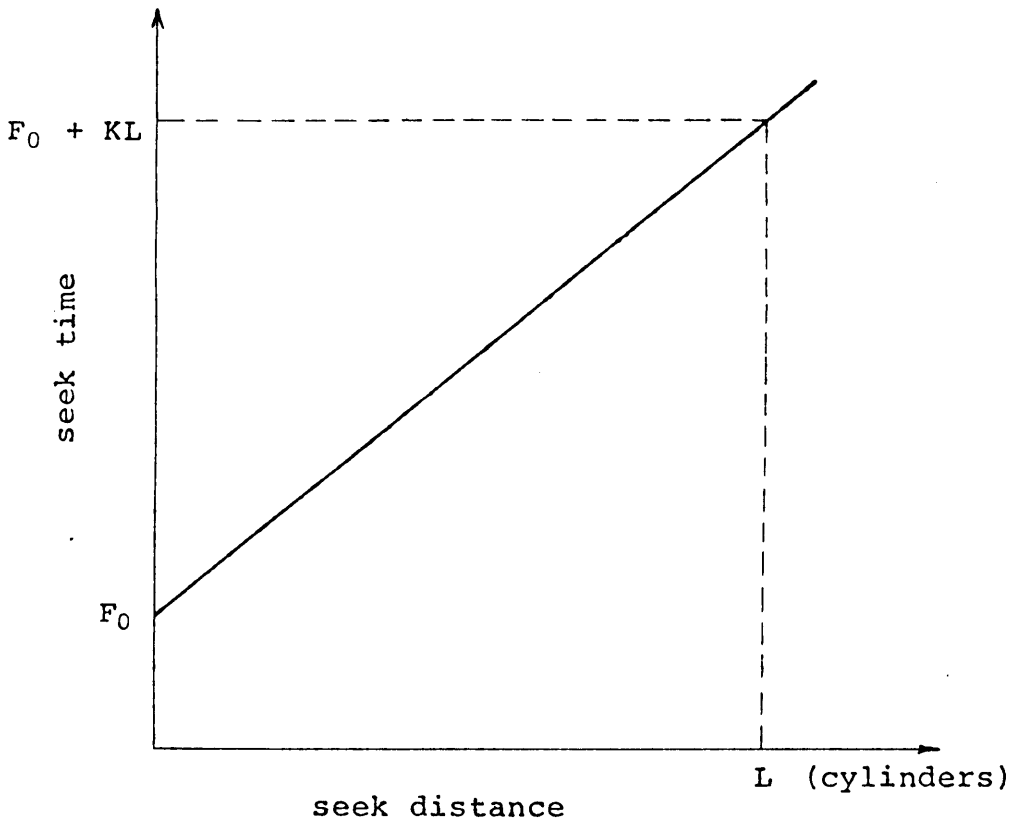


図4.1　ディスク・シーク時間の一次関数による近似

A continuous approximation for disk seek time

$$G_{SI}^{FX} = w \cdot D/2 \qquad (4.6)$$

(3) ページ・アウト・コスト C_{PO}^{FX}

既に述べたように1トランザクションについてのページ・アウトを一括して行なうと仮定したので，期待シーク時間 F_{PO}^{FX}，期待サーチ時間 G_{PO}^{FX} はそれぞれ式（4.7），（4.8）で与えられる。

$$F_{PO}^{FX} = F_0 + KL/3 \qquad (4.7)$$

$$G_{PO}^{FX} = wpr \cdot D/2 \qquad (4.8)$$

ただし式（4.8）において，r は1トランザクション，1ページあたりのページ・フォールト発生率である（式（4.17）参照）。

4.3.2 浮動方式の性能解析

4.3.2.1 トータル・コスト C^{FL}

浮動方式においては，ジョブ実行中に内容が変更されたページをスワップ・アウトないしページ・アウトする際，書き出すスロットは必ずしも初期割当て時のスロットとは限らない。どのようなアルゴリズムで新たなスロットを選択するかにより各種の方式が考えられる。ここでは書き出し時間が最小となるようなスロットを選ぶものを浮動方式と定義する。したがって浮動方式のもとで，新たなスロットは現在ディスク・ヘッドが停止しているシリンダの中から選ばれる。商用のオペレーティング・システムでは，ＩＢＭ社のＶＳ２[11]が浮動方式を採用している。

浮動方式におけるトータル・コストを式（4.2）より求めるとき，問題となるのはスワップ・イン・コストである。スワップ・アウトないしページ・アウトの際，内容が変更されたページはたまたまディスク・ヘッドが停止しているシリンダ上の新たなスロット上に書き出されるが，内容が変更されなかったページの格納されているスロットは元のままである。したがってジョブ実行開始後，スワップ・アウト／インを繰り返す間に，ひとつのジョブのワーキングセットに対応するスロット群が複数のシリンダにまたがることになる。スワップ・イン・コストを求める際には，このスロット群の散在度を評価する必要がある。

ここでは，まずスロット群の散在度がわかったとして，浮動方式のトータル・コスト C^{FL} を求めるための諸元を下記に導く（以下，C，F，G などの右肩の「ＦＬ」は浮動方式を表わす）。次に，4.3.2.2において，プログラムのページ参照動作モデルを仮定し，定常状態におけるスロット群の散在度を理論的に求める。

(1) スワップ・アウト・コスト C_{SO}^{FL}

スワップ・アウト時の期待シーク時間 F_{SO}^{FL}，期待サーチ時間 G_{SO}^{FL} は，それぞれ式（4.9），（4.10）で与えられる。

$$F_{SO}^{FL} = 0 \qquad (4.9)$$

$$G_{SO}^{FL} = D/2 \qquad (4.10)$$

ここで既に述べたように，スワップ・アウト時点でディスク・ヘッドが停止しているシリンダ内に

連続したwp個の未使用スロット群が常に存在すると仮定した。

(2) スワップ・イン・コスト C_{SI}^{FL}

F_{SI}^{FL}, G_{SI}^{FL} はそれぞれ式（4.11），（4.12）で与えられる。

$$F_{SI}^{FL} = \{F_0 + KL/2\} + \{F_0 + KL/(y+1)\}y \qquad (4.11)$$

$$G_{SI}^{FL} = x \cdot D/2 \qquad (4.12)$$

スワップ・アウト，ページ・アウトの際，ページはその時点でディスク・ヘッドが停止していたシリンダ内にまとめて書き出される。既に述べたように，このときの連続したスロット群が「ブロック」である。複数のトランザクションが実行されていく過程で，1つのジョブで参照されるページ群に対応するスロット群はディスク上に散在する。xは，定常状態で，ワーキングセット内のページ群に割り当てられたスロット群の属する相異なるブロック数の期待値である。また，yは，ワーキングセットに対応するスロット群のまたがる相異なるシリンダ数の期待値である。x個のブロックを形成する時点でのディスク・ヘッド位置が一様分布にしたがうとの仮定のもとで，yはxの関数として次のように与えられる（証明はF1）を参照されたい）。

$$y = \sum_{m=\max(0,L-x)}^{L-1} (L-m)\, T_m(x, L) \qquad (4.13)$$

ただし，

$$T_m(x, L) = \binom{L}{m} \sum_{j=0}^{L-m} (-1)^j \binom{L-m}{j} \left(1 - \frac{m+j}{L}\right)^x \qquad (4.14)$$

であり，$T_m(x, L)$は，ちょうど$(L-m)$個のシリンダにx個のブロックが存在し，m個のシリンダが空である確率である。$L \gg x$のときは$y \cong x$であるが，xの増大につれて$y < x$となる。図4.2に$L = 50$，100におけるxとyの関係を示す。yの算出法については，付録2.を参照されたい。

xをw，p，rから求める議論については4.3.2.2で述べるが，ここではx，yが与えられたとする。既に仮定したように，アクセスすべきy個のシリンダがディスク上に散在しているとき，読みこむスロット群をシリンダ番号についてソートし，図4.3のようにスキャンする。式（4.11）の第1項はLシリンダの二次メモリ領域の端までのシーク時間期待値であり，第2項はy個のシリンダをスキャンするシーク時間の期待値である。[D2)T2)] また，サーチするブロックの数はxであるから，式（4.12）が得られる。

(3) ページ・アウト・コスト C_{PO}^{FL}

ページ・アウトは既述のようにトランザクションごとに一括して行なわれるので，スワップ・アウトと同様にして次式が得られる。

$$F_{PO}^{FL} = 0 \qquad (4.15)$$

$$G_{PO}^{FL} = D/2 \qquad (4.16)$$

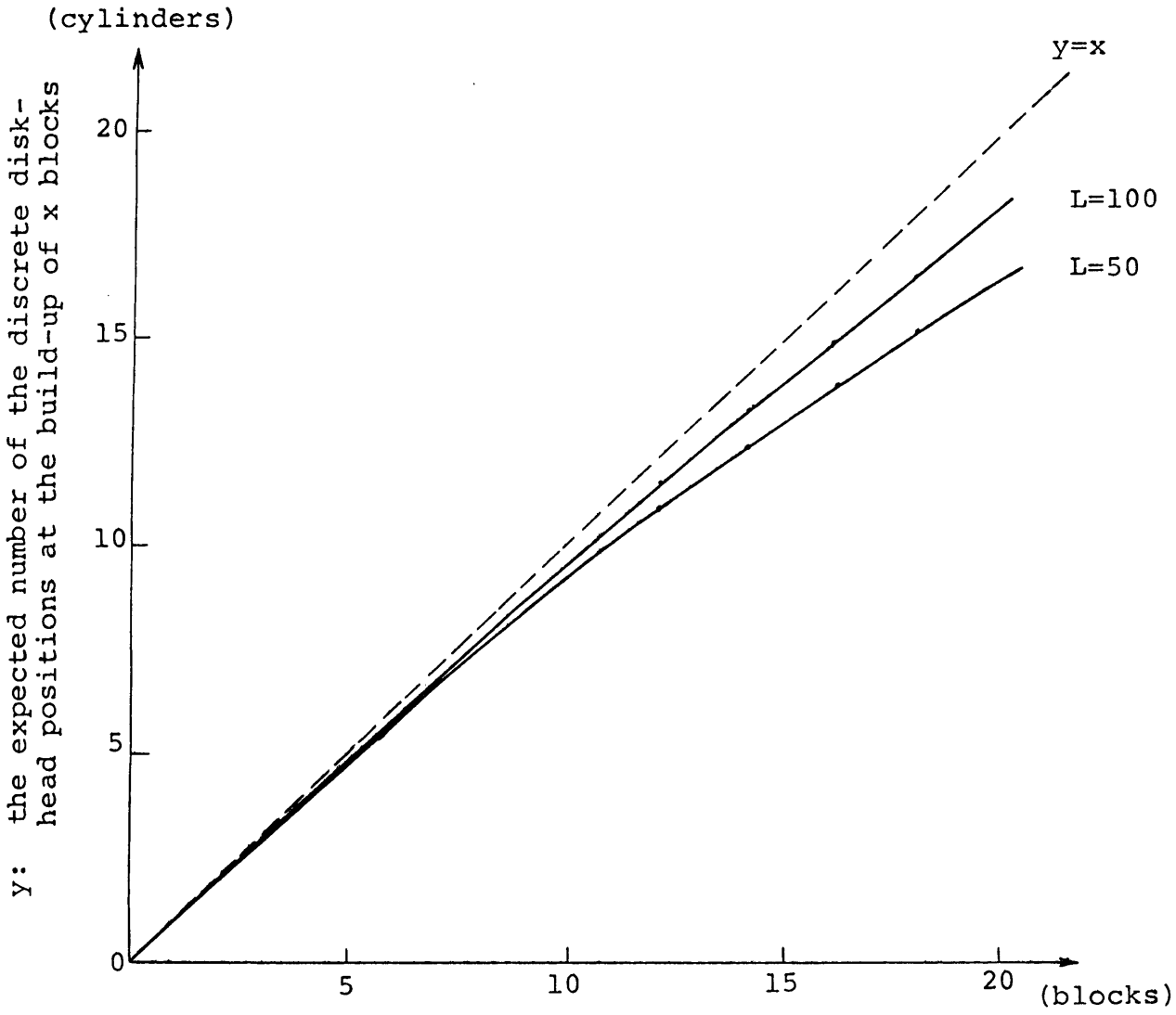


図4.2　確率モデルにより求めたxとyの関係

x versus y, calculated by the probabilistic model

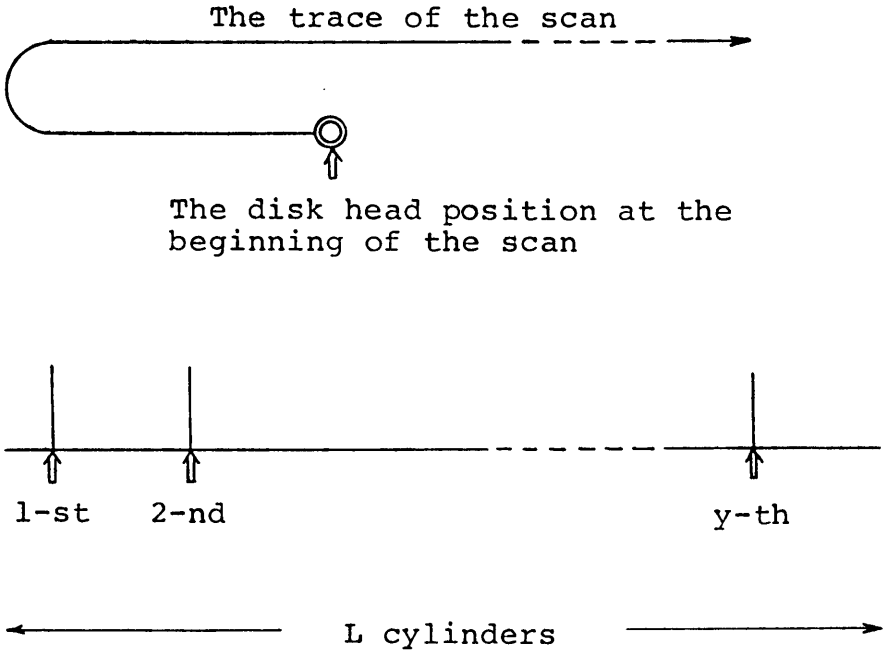


図4.3 浮動方式におけるスワップ・イン時の，y個のシリンダのアクセス法

The scan of y cylinders to swap-in the working-set under Floating technique

4.3.2.2　ブロック数 x の算定

x の算定にあたり，ページ参照の動作はＶＳＬＭ（Very Simple Locality Model）[S11)]にしたがうとする。ＶＳＬＭにおいて，局所参照集合（locality）外のページが参照される確率を λ，局所参照集合内の相異なるページ数（locality size）を b，プログラム・サイズを v（$b \leq v$）とする。局所参照集合内のひとつのページに着目するとき，これが参照される確率は（$1-\lambda$）/ b である。また，局所参照集合外の或るページが参照される確率は λ /（$v-b$）である。局所参照集合外の或るページが参照された場合，そのページは新たな局所参照集合に含まれ，古い局所参照集合の中から１ページがランダムに選ばれて局所参照集合から除外される。

ワーキングセット・ポリシーのもとで適当にウインド・サイズを選ぶと，ワーキングセットが局所参照集合の良い近似となることが知られている[S11)]。したがって以下，両者が一致すると仮定し，ワーキングセット・サイズ w は b と等しいとする。

トランザクションのページ参照回数を θ とすれば，この間のページ・フォールト発生回数の期待値は $\theta\lambda$ である。したがって，１トランザクション，１ページあたりのページ・フォールト発生率は，

$$r = \theta\lambda / w \qquad (4.17)$$

で与えられる。あるページが１トランザクションを通じてワーキング・セット内にとどまる確率は，r が比較的小さいとき，次式で近似できる。

$$(1-\lambda/w)^{\theta} = 1-\theta\lambda/w+\frac{1}{2}(\lambda/w)^2\theta(\theta-1)-\cdots$$
$$\cong 1-r \qquad (4.18)$$

誤差は $r^2/2$ のオーダである。なお，いったんワーキングセットから外れたページは，そのトランザクション中に再びワーキングセットに加わることはないと仮定する。

以上の準備のもとに x を求める。ワーキングセットはトランザクション開始時にスワップ・インされ，終了時にスワップ・アウトされる。x はスワップ・イン・コストに関与するから，トランザクション開始直前，いいかえればスワップ・アウト完了後の値を求めればよい。

(1)　ワーキングセット内のページのスロット位置（その１）

一般にワーキングセット内のページは(a)それがワーキングセットに加わって以来少なくとも１回書きこみが行なわれ内容が変化した（チェンジした）か否か，(b)そのスロットがページ・アウトにより割り当てられたかスワップ・アウトにより割り当てられたか，の２つの要因により４つに分類できる。（$p \equiv 0$ の場合の x は明らかなので，ここで $p > 0$ と仮定する。）ただし，ページ・アウトはワーキング・セットから外れたとき行なわれるので，ワーキングセットに加わってからチェンジし，なおワーキングセット内にとどまっているページは，全てスワップ・アウトによりスロットを割り当てられたものである。したがってワーキングセット内のページは実際には３つに分類できる。ます，チェンジし，かつスワップ・アウトによりスロットを割り当てられたページ群につき考える。

トランザクション開始直前において，前回のトランザクション中にチェンジした wp ページのス

ロット群は1ブロックを形成している。一般に第（$i+1$）回前のトランザクション中にチェンジし，その後チェンジせずワーキングセットにとどまったページのスロット群はひとつのブロックを形成する。その大きさを E_i とする。1トランザクションの間に，或るページがチェンジせずにワーキングセット内にとどまる確率は（$1-p$）（$1-r$）だから，

$$E_i = wp(1-p)^i(1-r)^i \qquad (4.19)$$

$$(i=0,1,2,\cdots)$$

となり，和は次式で与えられる（ただし以下，「ブロックの大きさ」は全て期待値を表わす）。

$$\sum_{i=0}^{\infty} E_i = wp/(p+r-pr) \qquad (4.20)$$

ここで次のようにしてブロック数を求める。$E_i \geq 1$ なら，当ブロックの存在する確率を1と考える。$E_i < 1$ なら，当ブロックの存在する確率を E_i に等しいと考える（この近似については付録3で考察を加えた）。E_i は i に関し単調減少であるから，$E_i \geq 1$ のなりたつ最大の i を I とかくと，ブロック数 Γ は次式で与えられる。

$$\Gamma = I + \sum_{i=I+1}^{\infty} E_i \qquad (4.21)$$

次に，ワーキングセットに加わって以来，一度もチェンジしていないページ群につき考える。このページ群は，そのスロットがページ・アウトで割り当てられたものと，スワップ・アウトで割り当てられたものとからなる。

前回のトランザクション中にワーキングセットに加わり，チェンジしなかったページ数は $wr(1-p)$ である。一般に，第（$i+1$）回前のトランザクション中にワーキングセットに加わり，その後チェンジせずにワーキングセット内にとどまったページ数を H_i とすると，

$$H_i = wr(1-p)^{i+1}(1-r)^i \qquad (4.22)$$

$$(i=0,1,2,\cdots)$$

となり，和は次式で与えられる。

$$\sum_{i=0}^{\infty} H_i = wr(1-p)/(p+r-pr) \qquad (4.23)$$

式（4.20）と（4.23）を加えると，その和は w に一致する。

E_i と異なり，H_i に対応するスロット群は必ずしも同一ブロックに属するわけではない。H_i に対応するスロット群の位置を解析するためには，ワーキングセット外のページのスロット位置に着目する必要がある。

(2) ワーキングセット外のページのスロット位置

トランザクション開始時にワーキングセット外にあった或るページが，トランザクション終了時まで引き続きワーキングセット外にとどまる確率は，

$$\{1-\lambda/(v-w)\}^{\theta} \cong 1-wr/(v-w) \qquad (4.24)$$

である（誤差は $r^2/2$ のオーダ）。いま，

$$q \triangleq wr/(v-w) \tag{4.25}$$

とすれば，q はワーキングセット外のページが1トランザクションの間にワーキングセットに加わる確率である。なお，いったんワーキングセットに加わったページは，そのトランザクション中は必ずワーキングセット内にとどまると仮定する。

ワーキングセット外のページは，そのスロットがページ・アウトにより割り当てられたかスワップ・アウトにより割り当てられたかにより，2つに分類できる。まず前者について考える。

前回のトランザクションでページ・アウトされたページ群の形成するブロックの大きさ ζ_0 は，

$$\zeta_0 = wpr \tag{4.26}$$

である。次回，および次々回のトランザクション終了時の，当該ブロックの大きさをそれぞれ ζ_1，ζ_2 とすると，それらは次のように与えられる。

$$\zeta_1 = \zeta_0(1-q) \tag{4.27}$$

$$\zeta_2 = \zeta_1(1-q) + \zeta_0 q(1-p)^2 r \tag{4.28}$$

式（4.28）の第2項は，ワーキングセットに入っていた $\zeta_0 q$ ページの中の $\zeta_0 q(1-p)^2 r$ ページは，チェンジせずに再びワーキングセットから外れたことを表わす。一般に，n 回トランザクション終了時の当該ブロックの大きさ ζ_n は，

$$\zeta_n = \zeta_{n-1}(1-q) + \sum_{i=0}^{n-2} q(1-p)^{n-i}(1-r)^{n-i-2} r\zeta_i \tag{4.29}$$

$$(n = 2, 3, \cdots)$$

で与えられる。ζ_n は，いいかえると，$(n+1)$ 回前のトランザクションでページ・アウトされたページのうち，現在ワーキングセットに含まれないもののスロット群からなるブロックの大きさである。なお，ここで次式が成り立つ（証明は付録4）。

$$\sum_{n=0}^{\infty} \zeta_n = (v-w)(p+r-pr)/(1+r-pr) \tag{4.30}$$

次に後者すなわちスワップ・アウトによりスロットを割り当てられたページにつき考える。いま，前々回のトランザクション終了時にスワップ・アウトされ，前回のトランザクションではチェンジせずにワーキングセットから外れたページのスロット群の形成するブロックの大きさ η_0 は，

$$\eta_0 = wp(1-p)r \tag{4.31}$$

である。次回，および次々回のトランザクション終了時の当該ブロックの大きさを η_1，η_2 とすると，

$$\eta_1 = \eta_0(1-q) + wp(1-p)^2(1-r)r \tag{4.32}$$

$$\eta_2 = \eta_1(1-q) + \eta_0 q(1-p)^2 r + wp(1-p)^3(1-r)^2 r \tag{4.33}$$

となる。式（4.32）の第2項，式（4.33）の第3項は，前々回のトランザクション終了時にスワップ・アウトされ，チェンジせずにワーキング・セット内にとどまっていたが，各々次回ないし次々

回のトランザクションでワーキングセットから外れるページに対応する。一般に，n 回トランザクション終了時の当該ブロックの大きさを η_n とすれば，次式が成り立つ。

$$\eta_n = \eta_{n-1}(1-q) + \sum_{i=0}^{n-2} q(1-p)^{n-i}(1-r)^{n-i-2} r \eta_i + wp(1-p)^{n+1}(1-r)^n r \qquad (4.34)$$

$$(n = 2, 3, \cdots)$$

η_n は，いいかえると，（$n+2$）回前のトランザクション終了時にスワップ・アウトされ，（$n+1$）回前以降のトランザクションで，チェンジせずにワーキングセットから外れたページのスロット群からなるブロックの大きさである。なお，ここで次式が成り立つ（証明は付録4）。

$$\sum_{n=0}^{\infty} \eta_n = (v-w)(1-p)/(1+r-pr) \qquad (4.35)$$

式（4.30）と（4.35）の和は（$v-w$）に一致する。

以上より，定常状態で，ワーキングセット外のページのスロットは，大きさ ζ_n ないし η_n（$n=0, 1, 2, \cdots$）の相異なるブロックのいずれかに属していることがわかる。

(3) ワーキングセット内のページのスロット位置（その2）

再び H_i に対応するページにつき考える。既述のように，これらのページ群はそのスロット割当ての契機がページ・アウトによるものと，スワップ・アウトによるものとに分類できる。まず，ページ・アウトでスロットを割り当てられたページ群に着目する。H_i は，（$i+1$）回前のトランザクション開始時にワーキングセット外にあったページの中からランダムに選ばれたページ群の大きさである。したがって，大きさ ζ_n のページ群の中で H_i に含まれるページ数は $H_i \zeta_n/(v-w)$ である。ここで，ある時点における大きさ ζ_n のブロックは，その前回のトランザクション開始直前には大きさ ζ_{n-1} のブロックであったわけだから，$H_i \zeta_n/(v-w)$ ページと $H_{i+1}\zeta_{n-1}/(v-w)$ ページとは，そのスロットが同一のブロックに属する。一般に大きさ $H_{m-j-1}\zeta_j/(v-w)$（$j=0, 1, 2, \cdots, m-1$）のスロット群は同一のブロックに属する。結局ブロックの大きさ ϕ_m は次式で与えられる。

$$\phi_m = \sum_{j=0}^{m-1} H_{m-j-1}\zeta_j/(v-w) = wr(1-p)(v-w)^{-1}\sum_{j=0}^{m-1}(1-p)^{m-j-1}(1-r)^{m-j-1}\zeta_j \qquad (4.36)$$

$$(m = 1, 2, \cdots)$$

なお ϕ_m の和は次式で与えられる（証明は付録5）。

$$\sum_{m=1}^{\infty} \phi_m = wr(1-p)/(1+r-pr) \qquad (4.37)$$

$\phi_m \geq 1$ の成り立つ m の添字集合を $\{M\}$ とすると，スロットの属する相異なるブロック数 Π は，

$$\Pi = \sum_{m=1}^{\infty} \{ \delta_m + (1 - \delta_m) \phi_m \} \qquad (4.38)$$

$$\delta_m = \begin{cases} 1 : m \in \{M\} \\ 0 : m \notin \{M\} \end{cases}$$

で与えられる（付録3参照）。$\{M\}$ を求めるには，

$$\sum_{m=1}^{\bar{m}} \phi_m > \sum_{m=1}^{\infty} \phi_m - 1 \qquad (4.39)$$

が成立する最小の $\bar{m}$ を $\hat{m}$ とかくとき，$\phi_1 \rightarrow \phi_2 \rightarrow \cdots \rightarrow \phi_{\hat{m}}$ を順次計算し，その中で $\phi_m \geq 1$ なるものを選べばよい。更に式（4.38）の第2項は，

$$\sum_{m=1}^{\infty} (1 - \delta_m) \phi_m = \sum_{m=1}^{\infty} \phi_m - \sum_{m=1}^{\infty} \delta_m \phi_m \qquad (4.40)$$

より求まるので，式（4.38）が計算できる。なお，ϕ_m は次のように表わせるので，ζ_n を求めることなく $\phi_1 \rightarrow \phi_2 \rightarrow \phi_3 \rightarrow \cdots$ と順次計算することができる（証明は付録6）。

$$\phi_1 = w^2 r^2 p (1 - p) / (v - w) \qquad (4.41)$$

$$\phi_2 = w^2 r^2 p (1 - p) \{ (1 - p) (1 - r) + (1 - q) \} / (v - w) \qquad (4.42)$$

$$\phi_m = q r \sum_{j=1}^{m-2} (1 - p)^{m-j} (1 - r)^{m-j-2} \phi_j + (1 - q) \phi_{m-1} + w^2 r^2 p (1 - p)^m (1 - r)^{m-1} / (v - w) \qquad (4.43)$$

$$(m = 3 , 4 , \cdots)$$

次に，H_t に対応するページの中で，スワップ・アウトでスロットを割り当てられたページ群に着目する。このとき，ζ_n を η_n で置きかえると全く同様な議論が成り立ち，ブロックの大きさ ψ_m は，

$$\psi_m = \sum_{j=0}^{m-1} H_{m-j-1} \eta_j / (v - w)$$

$$= w r (1 - p) (v - w)^{-1} \sum_{j=0}^{m-1} (1 - p)^{m-j-1} (1 - r)^{m-j-1} \eta_j \qquad (4.44)$$

$$(m = 1 , 2 , \cdots)$$

となり，その和は次式で与えられる（証明は付録5）。

$$\sum_{m=1}^{\infty} \psi_m = w r (1 - p)^2 / \{ (p + r - p r) (1 + r - p r) \} \qquad (4.45)$$

式（4.37）と（4.45）の和は，確かに $\sum_{t=0}^{\infty} H_t$ に一致する。$\psi_m \geq 1$ の成り立つ m の添字集合を $\{M'\}$ とすると，スロットの属する相異なるブロック数 Π' は，

$$\Pi' = \sum_{m=1}^{\infty} \{ \delta'_m + (1 - \delta'_m) \psi_m \} \qquad (4.46)$$

$$\delta'_m = \begin{cases} 1 : m \in \{M'\} \\ 0 : m \notin \{M'\} \end{cases}$$

で与えられる。式(4.46)は式(4.38)と全く同様にして計算できる。なお，ψ_m は次のように表わせるので，η_n を求めることなく $\psi_1 \rightarrow \psi_2 \rightarrow \psi_3 \rightarrow \cdots$ と順次計算することができる(証明は付録6)。

$$\psi_1 = w^2 r^2 p (1-p)^2 / (v-w) \qquad (4.47)$$

$$\psi_2 = w^2 r^2 p (1-p)^2 \{2(1-p)(1-r) + (1-q)\} / (v-w) \qquad (4.48)$$

$$\psi_m = q r \sum_{j=1}^{m-2} (1-p)^{m-j} (1-r)^{m-j-2} \psi_j + (1-q)\psi_{m-1} + m w^2 r^2 p (1-p)^{m+1} (1-r)^{m-1} / (v-w) \qquad (4.49)$$

$$(m = 3, 4, \cdots)$$

(4) x の算出

(1)～(3)の議論から，結局ワーキングセット内のページ群に割り当てられたスロット群の属する相異なるブロック数の期待値 x は，式(4.21)，(4.38)，(4.46)より次のように算出される。

$$x = \Gamma + \Pi + \Pi' \qquad (4.50)$$

ただし前述のように，Γ はワーキングセットに加わって以来少なくとも1回チェンジしたページ群，$(\Pi + \Pi')$ は全くチェンジしなかったページ群に対応する。さらに $(\Pi + \Pi')$ の中で，Π はページ・アウトによりスロットを割り当てられたページ群，Π' はスワップ・アウトによりスロットを割り当てられたページ群に対応する。

4.3.3 数値例

パラメータの値を次のように定めて，固定方式と浮動方式の性能比較を行なった。$w = 20$ ページ，$v = 40$ ページ，$p = 0.2, 0.4, 0.6, 0.8$。$r = 0.1, 0.2$。$L = 100$ シリンダ，$F_0 = 10$ ms*，$K = 0.1125$ ms*/シリンダ，$D = 16.7$ ms。

式(4.50)により求めた x と，この x と式(4.13)より求めた y とを表4.1に示す(ただし式(4.13)で，y は x が整数値のときのみ計算できる。したがって表4.1の y は内挿した値である)。さらに表4.1には，解析結果を検証するために実行したシミュレーションより求めた，ブロック数の期待値 x' とその標準偏差 σ' をも示した。

なお，シミュレーションの概略は次の通りである。v ページの各々には，ワーキングセット内か外かを示すエントリと，その割り当てられたスロットの属するブロック番号が与えられる。初期状態では，全ページのブロック番号は0であり，ランダムに選ばれた w ページがワーキングセット内，残りの $(v-w)$ ページがワーキングセット外とする。シミュレーションはトランザクションをくり返すことにより行なわれるが，ここでトランザクションは，スワップ・イン→ページ・イン→ページ・アウ

---

* HITAC H-8589-1 ディスク[H4)]を仮定する。

表 4.1　二次メモリ(ディスク)上における，ワーキングセットに割り当てられたスロット群の散在度

The dispersion of slots over a paging device

x　：the expected number of distinct blocks in which the slots are

y　：the expected number of distinct cylinders in which the slots are

x', σ'：x obtained by the simulation, and the standard deviation

p	r	x	y	x' ± σ'
0.2	0.1	13.5	12.7	11.7 ± 2.3
0.2	0.2	14.8	13.8	12.5 ± 2.2
0.4	0.1	8.1	7.8	7.3 ± 2.0
0.4	0.2	9.3	8.9	8.2 ± 1.8
0.6	0.1	5.1	5.0	5.0 ± 1.5
0.6	0.2	5.9	5.8	5.6 ± 1.4
0.8	0.1	3.1	3.1	3.1 ± 1.0
0.8	0.2	3.4	3.4	3.3 ± 1.1

p: the probability for a page to change during a transaction

r: page fault rate (per transaction, page)

ト→スワップ・アウトの4段階から構成される。スワップ・インでは，各ページのエントリは変化しない。ページ・インでは，ワーキング・セット外の各ページをqの確率でワーキングセット内に含める。次にワーキングセット内の各ページをpの確率でチェンジする。ページ・アウトでは，新しくページ・インされたページ以外のワーキングセット内の各ページをrの確率でワーキングセット外に出す。このとき出されるページ群の中でチェンジしたページ群に新たなブロック番号を与える。スワップ・アウトでは，ワーキングセット内のページ群の中でチェンジしたページ群に新たなブロック番号を与える。

トランザクションを多数回くり返した後，ワーキングセット内のページ群に対応する相異なるブロック番号の数を算定した。x'とσ'は，100回以上トランザクションをくり返して定常状態になった時点でのブロック数の，100サンプルについて求めた。xはほぼ$x' \pm \sigma'$の範囲に含まれており，xとx'との差異は0～16%である。xはx'よりやや大きいが，これは付録3で考察したように，ブロックの存在確率が直接求められず，ブロックの大きさを用いて近似計算をせざるを得ないためである。

両方式のスワップ・アウト・コスト，スワップ・イン・コスト，トータル・コストを比較した結果を図4.4～4.6に示す。スワップ・アウト・コストは浮動方式がすぐれている。スワップ・イン・コストは，チェンジする確率pが小さいとき固定方式がすぐれている。さらにトータル・コストを比べると，総合的な効率としてはpが小さいとき固定方式，大きいとき浮動方式がすぐれていることがわかる。これは，pが増大するにつれxやyが減少し，浮動方式が有利になるためである。

4.4 スロット割当ての最適化と集中方式（Concentrating Technique）

4.4.1 スロット割当てのモデル

スロット割当ての最適化を行なうために，まず従来方式を包含したスロット割当ての一般モデルを提示する。本モデルにおいて，スワップ・アウト，ページ・アウトの際，ディスク・ヘッドが第（a＋1）～第（$a+h$）シリンダ（図4.7の斜線部）にあれば，これが現在停止しているシリンダ内の連続したスロット群（ブロック）に書き出す*。ディスク・ヘッドが第1～aシリンダにあれば第（a＋1）シリンダ内（ただし$a \neq 0$のとき），第（$a+h+1$）～Lシリンダにあれば第（$a+h$）シリンダ内（ただし$a+h \neq L$のとき）の，それぞれ連続したスロット群（ブロック）に書き出す*。ここでhは全ジョブで共通だが，aはジョブ毎に異なる値をとる。すなわち本モデルでは，hはスロット割当ての自由度を表わし，各ジョブに割り当てられるスロット群は，ジョブ毎に連続したhシリンダ内にまとまる。hの値域は式（4.51）で与えられ，hが定められたときaの値域は式（4.52）で与えられる。

$$1 \leq h \leq L \qquad (4.51)$$

---

* 4.2で述べたように，本論文では連続した未使用スロット群が常にあると仮定する。

$$0 \leq a \leq L - h \qquad (4.52)$$

hの増大にともない，スワップ・アウトの期待シーク時間F_{SO}，ページ・アウトの期待シーク時間F_{PO}は減少するが，スワップ・インの期待シーク時間F_{SI}は増大するため，トータル・コストCを最小にするhの最適値が存在する。なお，期待サーチ時間G_{SO}，G_{SI}，G_{PO}はhやaの値によらない。

本モデルで$h = L$とおくと，浮動方式と一致する。また本モデルで$h = 1$とおくと，これは固定方式に近いが，より自由度が大であり，以下の理由により固定方式よりトータル・コストCが小さい。まず，両者ともシリンダは固定なので期待シーク時間F_{SO}，F_{SI}，F_{PO}は同一である。次に本モデル（$h = 1$）ではシリンダ内のスロット割当ては浮動でありスワップ・アウト，ページ・アウトの際当該シリンダ上の連続したスロット群に書き出すが，固定方式ではこれも固定であり書き出すスロット群は一般に非連続なので，期待サーチ時間G_{SO}，G_{PO}は前者の方が小さい。さらにワーキングセット内のページが属する相異なるブロック数xは当然ワーキングセット・サイズw以下であり，前者，後者のG_{SI}はそれぞれx，wにランダム・サーチ時間期待値$D/2$を乗じた値であるから，G_{SI}も前者の方が小さい。（なお，一般に$h = 1$より自由度を減ずるとCは増大することが，同様にして言える。）

以上より，本モデルでhの最適解を求めると，それは浮動方式，固定方式のいずれよりも効率がよい。

4.4.2 最適な自由度

本モデルにおいて式（4.2）の諸元を以下にみちびく。

$$\begin{aligned} F_{SO}(h, a) &= (a/L)(F_0 + Ka/2) \\ &\quad + \{(L-a-h)/L\}\{F_0 + K(L-a-h)/2\} \\ &\quad (= F_{PO}(h, a)) \end{aligned} \qquad (4.53)$$

式（4.53）で，第1項の(a/L)はディスク・ヘッドがシーク動作開始時点で第1～aシリンダにある確率であり，$(F_0 + Ka/2)$はそのときの期待シーク時間を表わす。第2項はディスク・ヘッドが第$(a+h+1)$～Lシリンダにある場合について同様に求めたものである。

次に$F_{SI}(h, a)$をみちびく。既にのべたように，ワーキングセット内のページに対応するスロット群は，x個の相異なるブロックに属する。これらのブロックをスワップ・アウトないしページ・アウトで形成するためのシーク動作開始時点における，相異なるディスク・ヘッド位置の数の期待値をyとする（このyの定義は，4.3.2で与えた定義すなわち「ワーキングセット内のページに対応するスロットの存在する相異なるシリンダ数の期待値」を含むより広いものである）。ただし前述のように，ディスク・ヘッドが図4.7の斜線部にあれば，実際にはシーク動作は行なわれない。$h = L$のとき，x個のブロックが第1～Lシリンダの範囲のy個のシリンダにそのまま散在する。一般の場合には，x個のブロックが存在するシリンダは第$(a+1)$～$(a+h)$シリンダの範囲に限られ，これらをアクセスするための期待シーク時間は次式で与えられる。

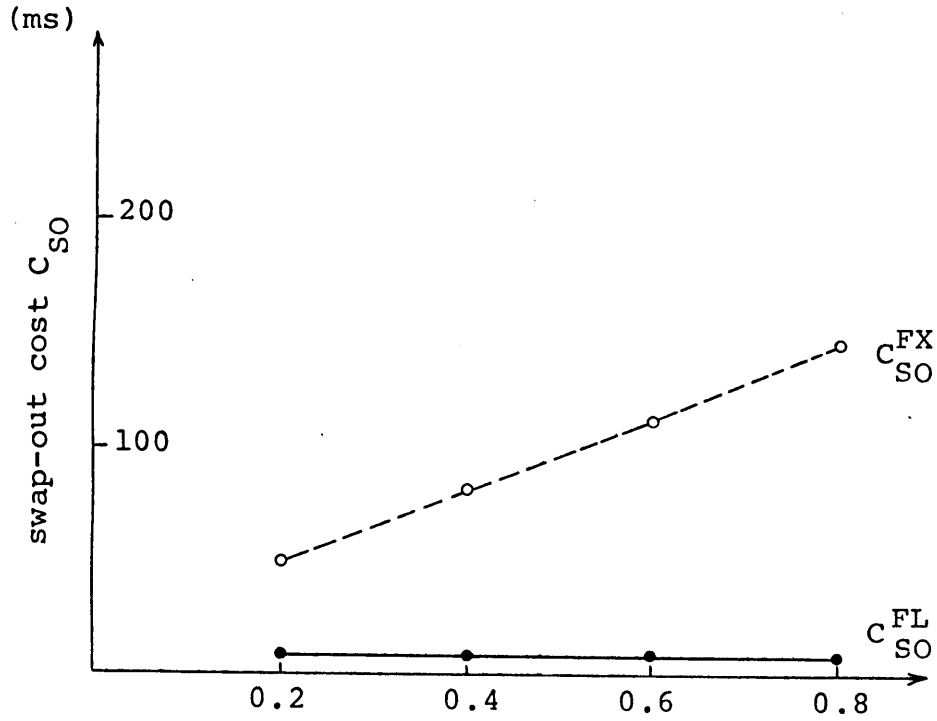


図4.4 スワップ・アウト・コストに関する，浮動方式と固定方式の比較

Performance comparison of the two techniques (Fixed: o--o, Floating: ●—●) with respect to the swap-out cost

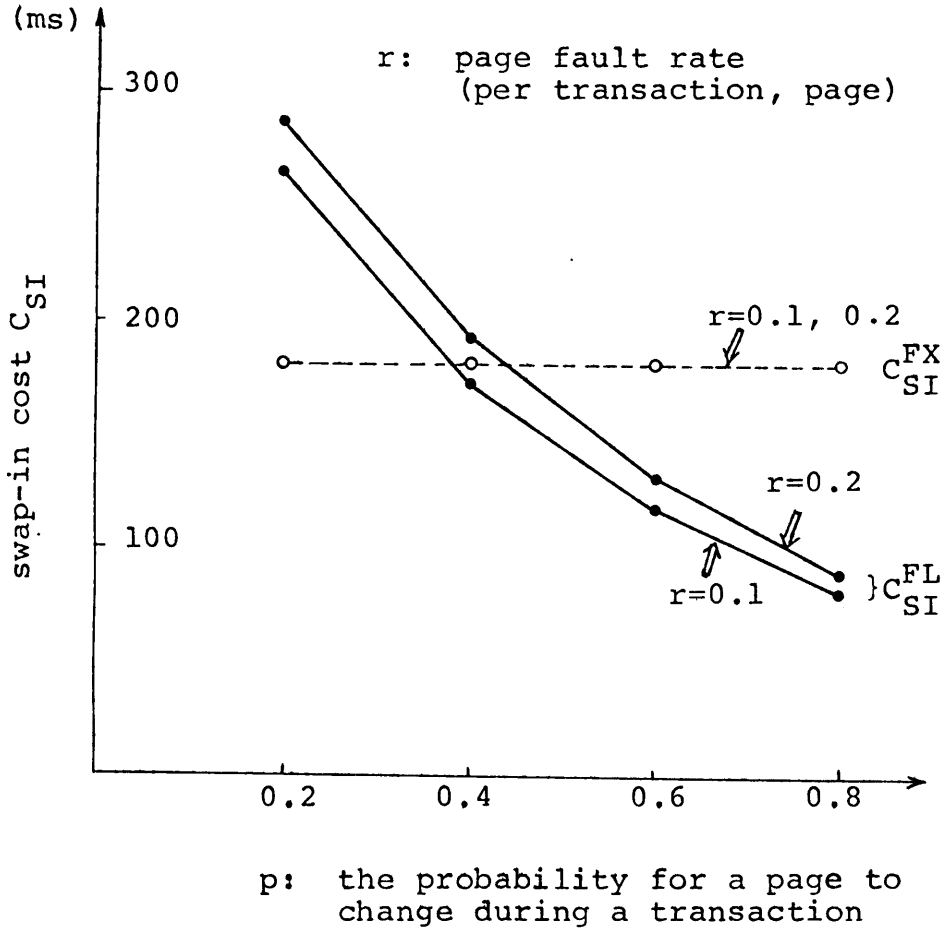


図4.5 スワップ・イン・コストに関する，浮動方式と固定方式の比較

Performance comparison of the two techniques (Fixed: o---o , Floating: ●—●) with respect to the swap-in cost

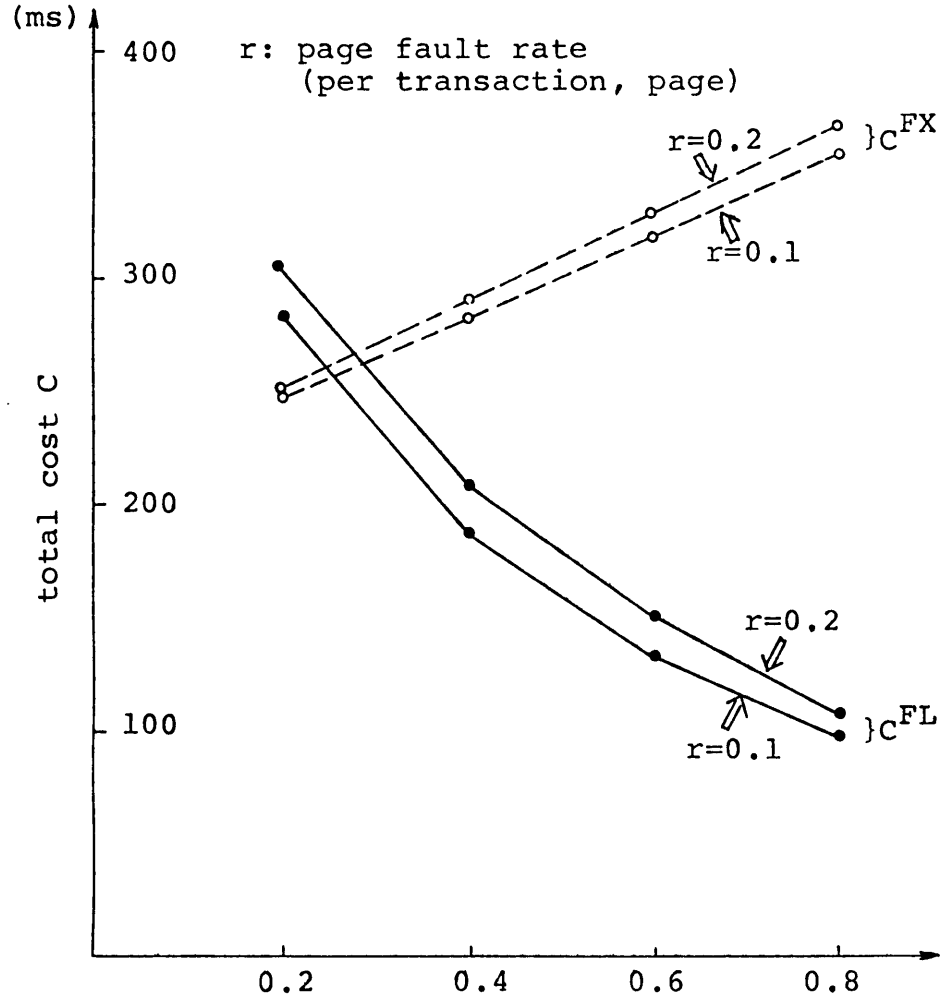


図4.6 トータル・コストに関する，浮動方式と固定方式の比較

Performance comparison of the two techniques (Fixed: o---o, Floating: ●——●) with respect to the total cost

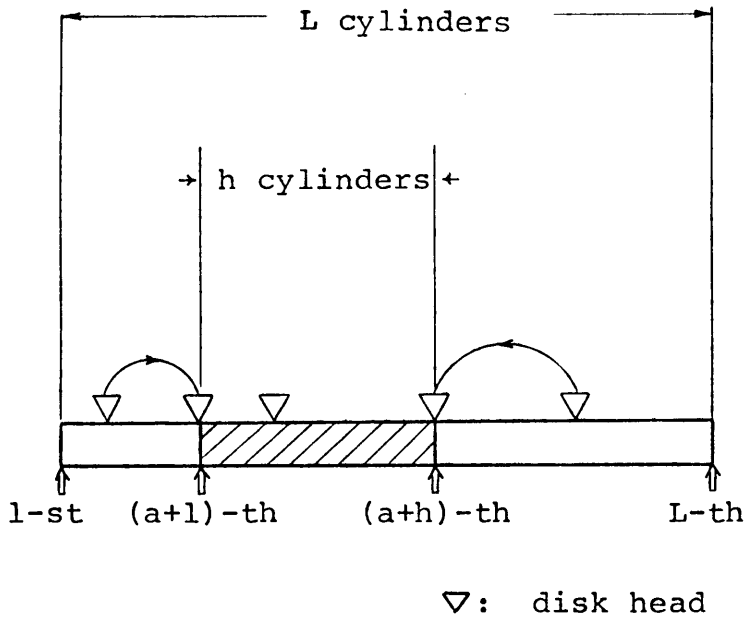


図 4.7 スロット割当てモデルにおける，スワップ・アウト／ページ・アウト時のシーク動作

The seek operation at swap-out/page-out in the slot allocation model

$$F_{SI}(h,a) = [\{(L-1)/L\}F_0 + (a/L)(Ka/2) + \{(L-a-1)/L\}\{K(L-a-1)/2\}] + \{y(h-1)/L\}\{F_0 + KL/(y+1)\} \quad (4.54)$$

式(4.54)の第1項は，スキャンの準備動作としてディスク・ヘッドが第$(a+1)$シリンダに到着するための期待シーク時間である。a/Lはシーク動作開始時点でディスク・ヘッドが第1～aシリンダ，$(L-a-1)/L$はこれが第$(a+2)$～Lシリンダにある確率であり，$a/2$，$(L-a-1)/2$はそれぞれの場合の期待シーク距離を表わす。第2項は第$(a+2)$～$(a+h)$シリンダをスキャンするための期待シーク時間であり，$y(h-1)/L$はこの範囲でアクセスすべきシリンダ数の期待値である。ここで，$L/(y+1)$は1シリンダあたりの期待シーク距離を表わす。[D2)T2)]

期待サーチ時間G_{SO}，G_{SI}，G_{PO}は次式で与えられる。

$$G_{SO} = D/2 \quad (= G_{PO}) \quad (4.55)$$

$$G_{SI} = xD/2 \quad (4.56)$$

式(4.53)～(4.56)を式(4.2)に代入すると，Cがaとhの関数として求まる。ここで，多重プログラミング方式で処理されるジョブの数が多ければ，aは式(4.52)の範囲で一様分布にしたがうとみなせる。したがって，平均としてのトータル・コスト$\bar{C}(h)$を次式で定義する。*

$$\bar{C}(h) \triangleq \{1/(L-h)\}\int_0^{L-h} C(h,a)\,da$$
$$= K(L-h)^2/L + \{2F_0 - K(L-1)/2\}(L-h)/L + \{F_0 + KL/(y+1)\}y(h-1)/L + K(L-1)^2/2L + (L-1)F_0/L + (x+2)D/2 \quad (4.57)$$

式(4.57)をhで微分すると次式を得る。

$$\frac{d\bar{C}}{dh} = -2K(L-h)L - \{2F_0 - K(L-1)/2\}/L + \{F_0 + KL/(y+1)\}y/L \quad (4.58)$$

これより，

$$\frac{d\bar{C}}{dh} = 0 \quad (4.59)$$

をみたす$\hat{h}$は，次式で与えられる。

$$\hat{h} = -Ly/\{2(y+1)\} - F_0(y-2)/(2K) + (1/4 + 3L/4) \quad (4.60)$$

$\bar{C}(h)$は，hの下に凸な放物線であるから，式(4.51)の条件より，$\hat{h} \leq 1$なら$h=1$，$\hat{h} \geq L$なら$h=L$，それ以外のとき$\hat{h}$が最適解を与える。

---

* hやaは整数値だが，連続値をとると仮定して議論を進める。

4.4.3　集中方式（Concentrating Technique）

実際には，実用的条件下で$\hat{h} \leq 1$が成立し，$h=1$を近似的な最適解とみなすことができる。以下$h=1$に対応する方式を「集中方式（Concentrating Technique）」とよぶ。集中方式では，シリンダ割当ては固定（常に第（$a+1$）シリンダに書き出す），シリンダ内のスロット割当ては浮動であり，その実現は比較的容易である。

式（4.60）より，$\hat{h} \leq 1$の条件は次式で与えられる。

$$y \geq (1/F_0)\left[(F_0/2 + KL/4 - 3K/4) + \sqrt{\{F_0/2 + KL/4 - 3K/4\}^2 + 2F_0\{F_0 + 3KL/4 - 3K/4\}}\right] \quad (4.61)$$

HITAC H-8589-1*（$F_0=10$ ms，$K=0.1125$ ms/シリンダ）およびH-8589-11*（$F_0=10$ ms，$K=0.05625$ ms/シリンダ）の2種のディスク[H4]について，式（4.61）の条件を図示すると図4.8の斜線部のようになる（なお，Lは余り小さい値とはなりえないので，ここでは$L \geq 10$として示した）。実際にはyが斜線部内にある場合が多い（表4.1参照）。また，

$$\frac{d\bar{C}}{dy} = (h-1)F_0/L + K(h-1)/(y+1)^2 + \frac{dy}{dx}(D/2) > 0 \quad (4.62)$$

であり（付録2より$dy/dx > 0$），yの減少にともなって$\bar{C}$も減少する。したがってyが小さい場合（斜線部以外）には$\bar{C}$自体が小さいので，実用上余り問題とならない。

次に集中方式と従来方式との比較について論ずる。集中方式は，最適とならない条件のもとでも，多くの場合従来方式よりは優れている。まず固定方式と比較すると，既に4.4.1で示したように，集中方式のトータル・コストはL，x，yの値によらず常に固定方式のそれより小である。したがって本論文では以下，浮動方式との比較を中心に述べる。

以下，集中方式（Concentrating：CN）と浮動方式（Floating：FL）のコストを右肩の記号で区別し，$C^{CN} < C^{FL}$の成立する条件を求める。C^{CN}，C^{FL}はそれぞれ式（4.57）より次式で与えられる。

$$C^{CN} = \bar{C}(1) = K(L-1)^2/L + 3F_0(L-1)/L + (x+2)D/2 \quad (4.63)$$

$$C^{FL} = \bar{C}(L) = \{F_0 + KL/(y+1)\}y(L-1)/L + K(L-1)^2/2L + F_0(L-1)/L + (x+2)D/2 \quad (4.64)$$

式（4.63），（4.64）より$C^{CN} < C^{FL}$の成立する条件は，

$$y > (1/F_0)\left[(F_0/2 - KL/4 - K/4) + \sqrt{\{F_0/2 - KL/4 - K/4\}^2 + 2F_0\{F_0 + KL/4 - K/4\}}\right] \quad (4.65)$$

で与えられ，$F_0=10$ms，$K=0.1125$ms/シリンダと$F_0=10$ms，$K=0.05625$ms/シリンダ

---

* H-8589-1，H-8589-11は，IBM社の3330-1，3330-11に相当し，この種のディスクは広く用いられている。

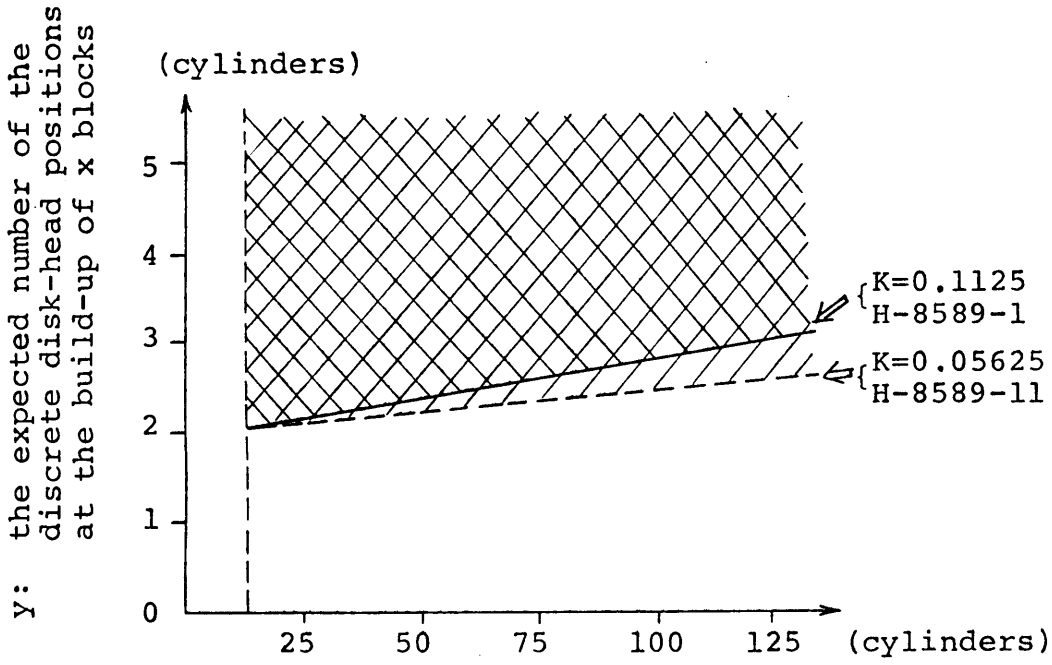


図4.8 集中方式が最適となる範囲（斜線部）

The condition yielding $\hat{h} \leq 1$ (shaded portion) where the Concentrating technique is optimum

の２つの場合につき，この範囲を斜線部で示したのが図4.9である（図4.8と同じく$L \geq 10$とした）。図4.9の斜線部は，現実に生じうる場合をほぼつくしている（表4.1参照）。

さらに，$F_0 = 10$ms，$K = 0.05625$ ms／シリンダ（H−8589−11）のディスクで$L = 50$，100シリンダの条件のもとで，両方式のコスト比$(C_{SO}^{CN} + C_{SI}^{CN}) / (C_{SO}^{FL} + C_{SI}^{FL})$，$C^{CN} / C^{FL}$をブロック数$x$の関数として示したのが図4.10である。ただし，スワップ・アウト・コストC_{SO}^{CN}，およびスワップ・イン・コストC_{SI}^{CN}はそれぞれ次式で与えられる（C_{SO}^{FL}，C_{SI}^{FL}は既に4.3.2.1で求めた）。

$$C_{SO}^{CN} = \{1/(L-1)\} \cdot \int_0^{L-1} C_{SO}(1, a)\, da$$

$$= K(L-1)^2/(3L) + F_0(L-1)/L + D/2 \qquad (4.66)$$

$$C_{SI}^{CN} = \{1/(L-1)\} \cdot \int_0^{L-1} C_{SI}(1, a)\, da$$

$$= K(L-1)^2/(3L) + F_0(L-1)/L + xD/2 \qquad (4.67)$$

前節でxの理論的算出法について論じたが，さらに次節では実測データにもとづくxの推定値を提示する。

図4.10より，集中方式を浮動方式と比較すると，トータル・コストとして約0～40%の効率向上が期待される。実際の効率向上効果は，ページ・アウトの頻度がスワップ・アウト／インの頻度と同程度ならばC^{CN} / C^{FL}で評価でき，これが減少するにしたがって$(C_{SO}^{CN} + C_{SI}^{CN}) / (C_{SO}^{FL} + C_{SI}^{FL})$に近づく。

なお，図4.10で$L = 50$と100とを比べると，$L = 50$がやや効率向上効果が小さいが，両者の差はわずかである。また，$K = 0.1125$ ms／シリンダ（H−8589−1）の場合は，識別できない程度に図4.10と一致した。

4.5 実験と評価

4.5.1 測定条件と測定方法

集中方式を実験的に実現し，その効果を浮動方式との比較において実測により検証した結果を以下に示す。初めに測定の条件と方法を述べる。

実測評価の機器構成としては，大型計算機ＨＩＴＡＣ　Ｍ−１８０を使用し，二次メモリ媒体はＨＩＴＡＣ H−8589−11[H4)] ディスクを用いた。負荷としては，５０端末から短小な入力編集処理が実行されると仮定した。５０人の測定人員を確保することは困難なので，端末の回線入出力をディスクへの入出力により模擬する「端末シミュレータ」を用いて測定を行なった。端末シミュレータは，各端末ごとに，平均２０秒の指数分布にしたがう思考時間が終了すると入力編集コマンドを発生してトランザクションを実行する。回線入出力模擬用のディスクと二次メモリ用ディスクは別の媒体を使用し，また回線入出力以外の動作は全く実システムの動作と一致している。したがって，二次メモリ管理の方式評価に関する限り，実際に５０人が端末使用中と同様の周囲条件にあるとみなすことができる。なお，バ

ッチ・ジョブは実行しなかった。

測定は2つの方法により行なった。第1は実メモリのダンプである。これによりyの推定値を求めた。既述のように，浮動方式のもとでは，yはワーキングセット内のページに割り当てられたスロット群が存在する相異なるシリンダ数に等しい。実メモリ内の管理テーブルのダンプ情報から，このシリンダ数を算出した。第2は本実験のために開発したページ入出力効率評価用のソフトウェア・モニタである。本モニタはページ入出力が実行される度に，その実行に要した時間を記録し，逐次磁気テープに出力するものである。

なお，測定時間は各方式につき約30分間とし，全50端末がlog－onを完了し入力編集コマンドを発生中の定常状態約16分間について測定データを整理した。

4.5.2 実測結果

浮動方式のもとで定常状態になってから約10分間経過した時点で，yの推定値を測定した結果が図4.11である。図4.11は，この時点でたまたまスワップ・アウトされていた49端末のジョブについて*，各々スワップ・アウト動作直前におけるワーキングセット内のページに割り当てられたスロット群の存在する相異なるシリンダ数の度数分布を表わす。平均値は4.61シリンダ，標準偏差は0.94シリンダである。図4.12はこのときのワーキングセット・サイズの度数分布を示す。平均値は9.18ページ，標準偏差は1.24ページである。したがって約2ページずつ各シリンダに散在していることがわかる。

集中方式および浮動方式のもとで，定常状態の約16分間にわたって測定したページ入出力実行時間の度数分布を図4.13に示す。集中方式では，ページ入出力実行時間平均値が浮動方式より約38%（18.88 msから11.63 ms）短縮された。また，同標準偏差，同90%累積値においても顕著な短縮がみられた。なお，入出力ページ数の総計は31,973ページ（集中方式），32,008ページ（浮動方式）であり，その契機の内訳はスワップ・アウト，スワップ・イン，ページ・アウト，ページ・インがそれぞれ43.97%，52.93%，0.10%，3.00%（集中方式），44.04%，52.92%，0.13%，2.91%（浮動方式）であった。

図4.10～4.13により，集中方式によるページ入出力効率向上効果の検証を行なうことができる。図4.11に示したシリンダ数平均値は一時点におけるポイント・データにすぎず，測定全体を通じての統計値ではない。しかし，図4.14に示すように，1分毎のページ入出力実行時間平均値の時間変化が比較的小さいことから，一応yの推定値として採用可能である。ここで，yが約4.6のときxは約4.8である（図4.2参照）。ページ入出力実行時間平均値とトータル・コストCとは直接対応はしないが，集中方式と浮動方式とのページ入出力実行時間平均値の比率0.62は図4.10の$x=4.8$における$(C_{SO}^{CN}+C_{SI}^{CN})/(C_{SO}^{FL}+C_{SI}^{FL})$の値に比較的よく一致している。いまページ・アウトの頻度が

---

* スワップ・イン中の端末は，スロット割当て実行中なので除外した。

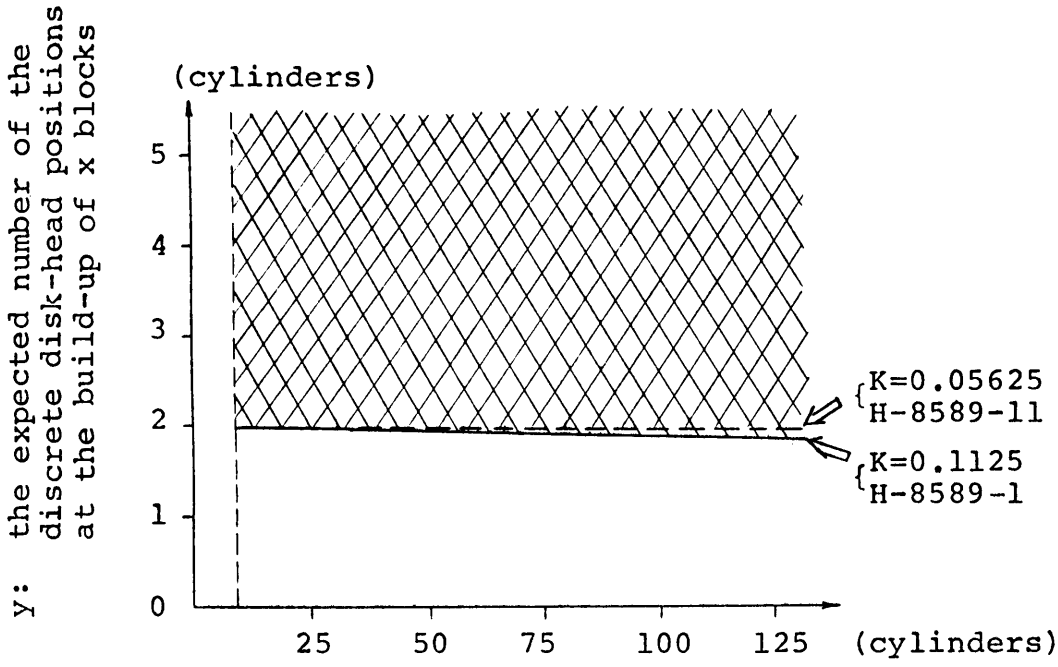
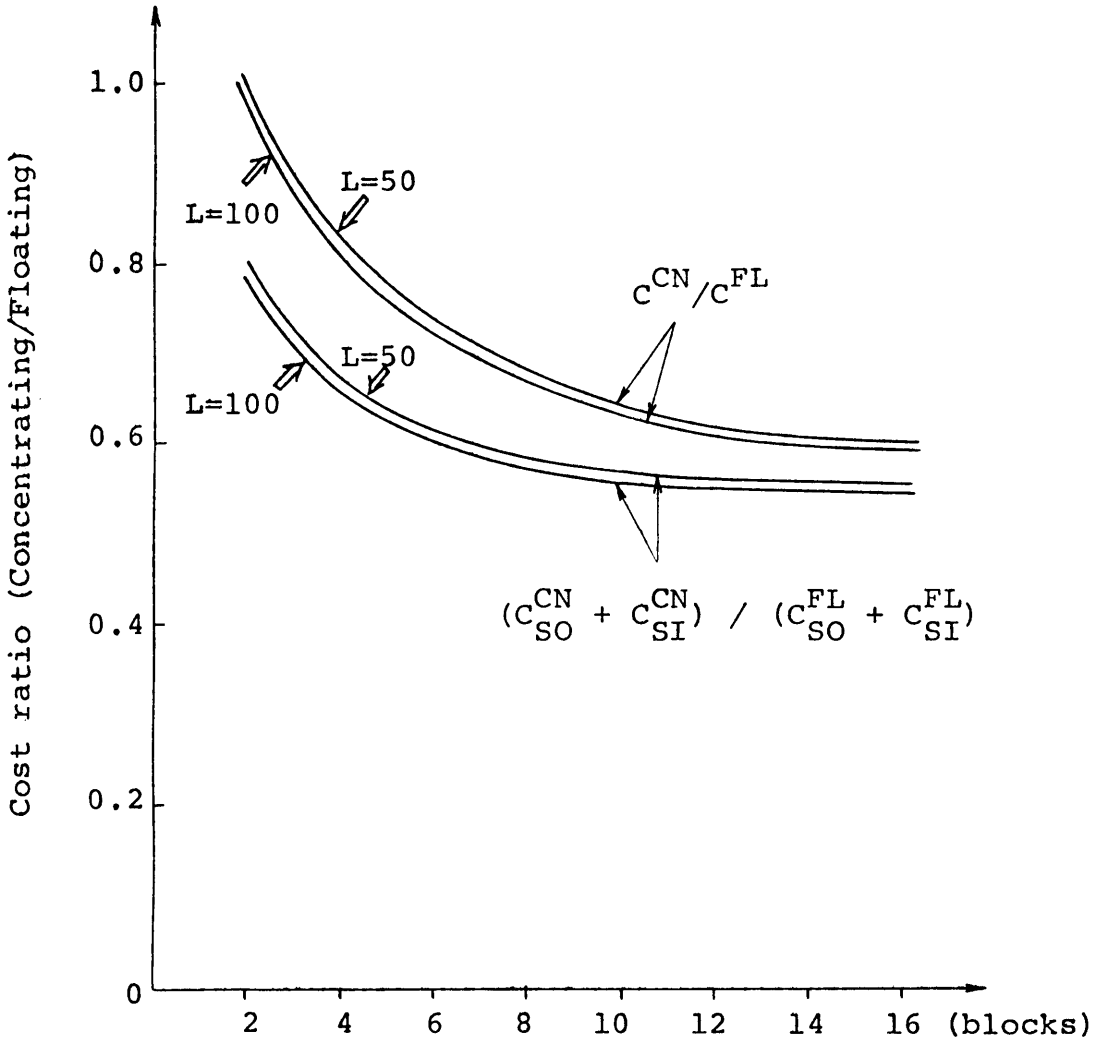


図4.9　集中方式が浮動方式より優れる範囲(斜線部)

The condition yielding $C^{CN} < C^{FL}$ (shaded portion), where the Concentrating technique is superior to the Floating technique



x: the number of the blocks corresponding to program's working-set

図4.10　集中方式と浮動方式とのコスト比較

The paging cost of the Concentrating technique in comparison with the Floating technique

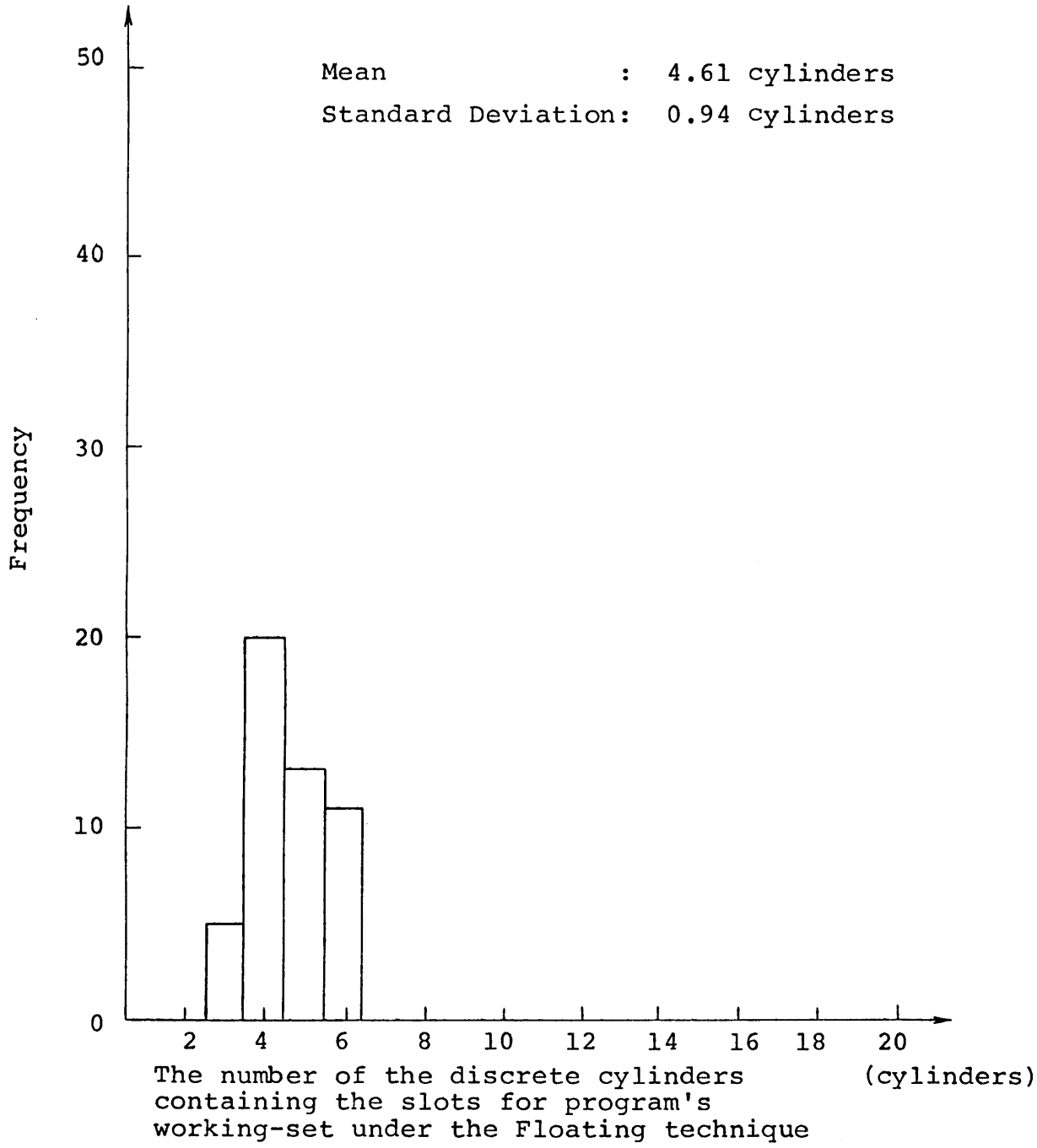


図4.11 浮動方式のもとで，実測中の一時点（定常状態）における，プログラムの置かれたシリンダ数の度数分布

The distribution used for the estimation of y, observed at a steady state under the Floating technique

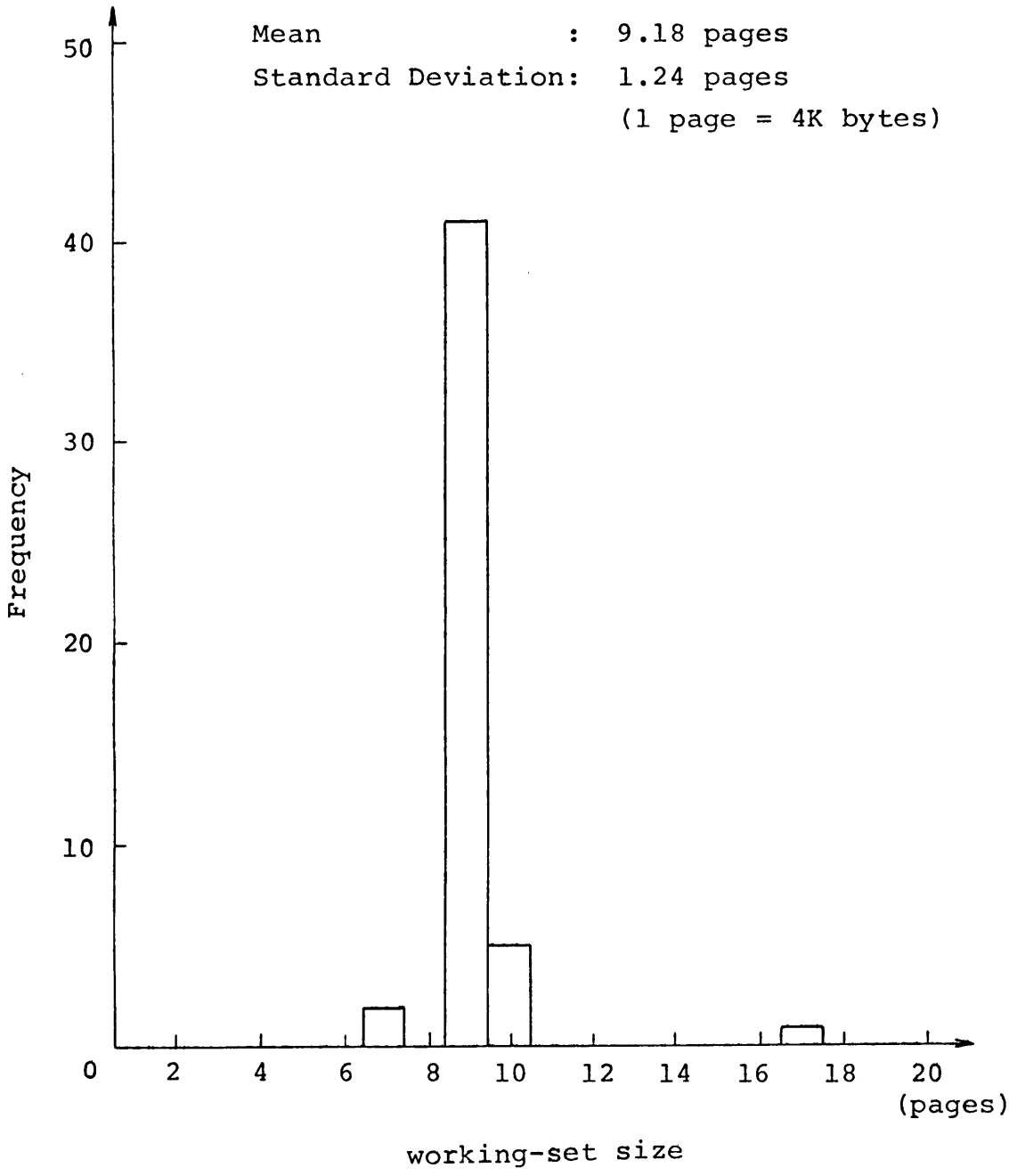


図 4.12 浮動方式のもとで，実測中の一時点（定常状態）における，プログラムのワーキングセット・サイズの度数分布

The distribution of the observed working-set size at a steady state under the Floating technique

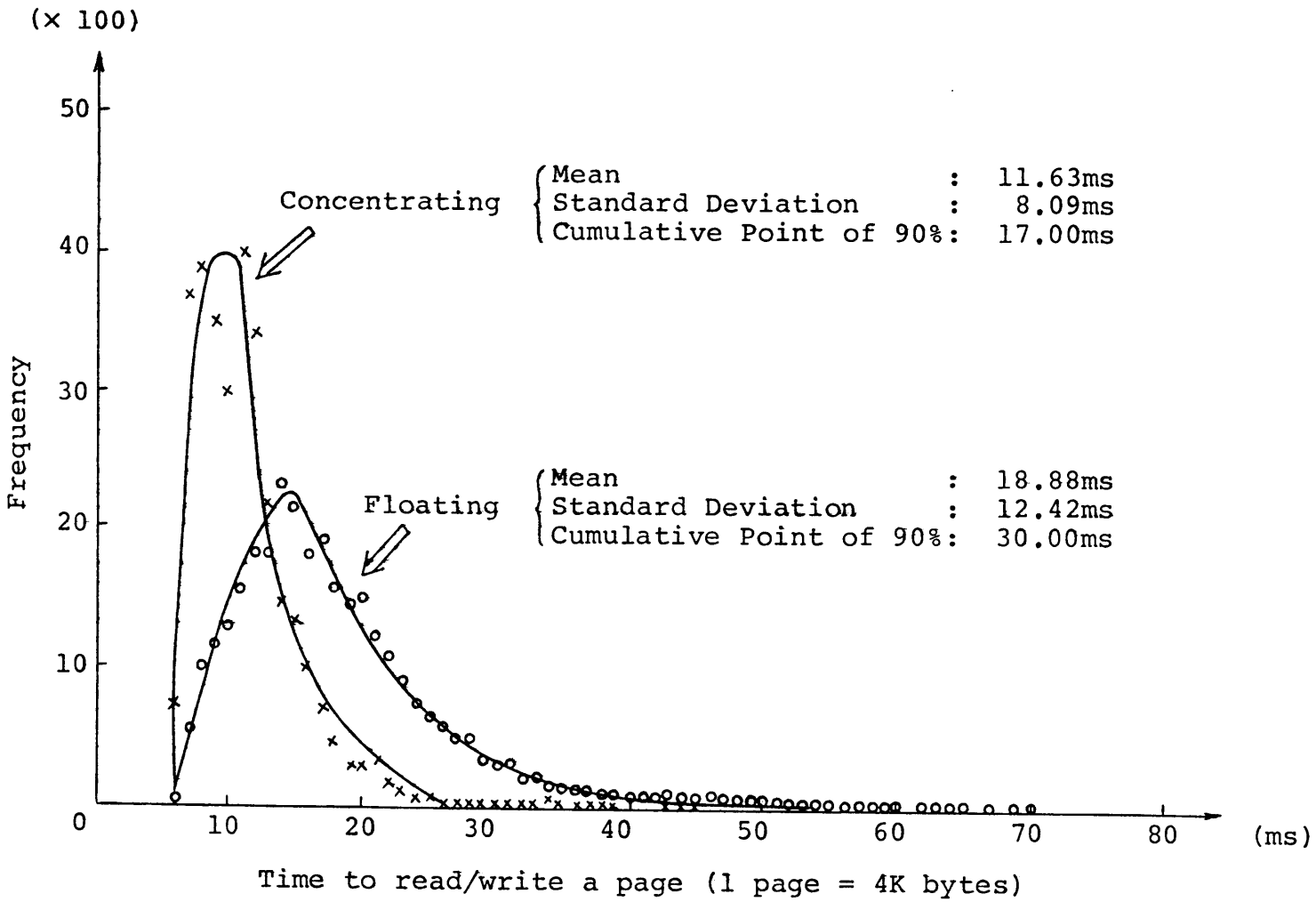


図4.13 集中方式と浮動方式のページ入出力実行時間実測値の比較

The distributions of the observed page I/O time for the two techniques, Concentrating (x) and Floating (O)

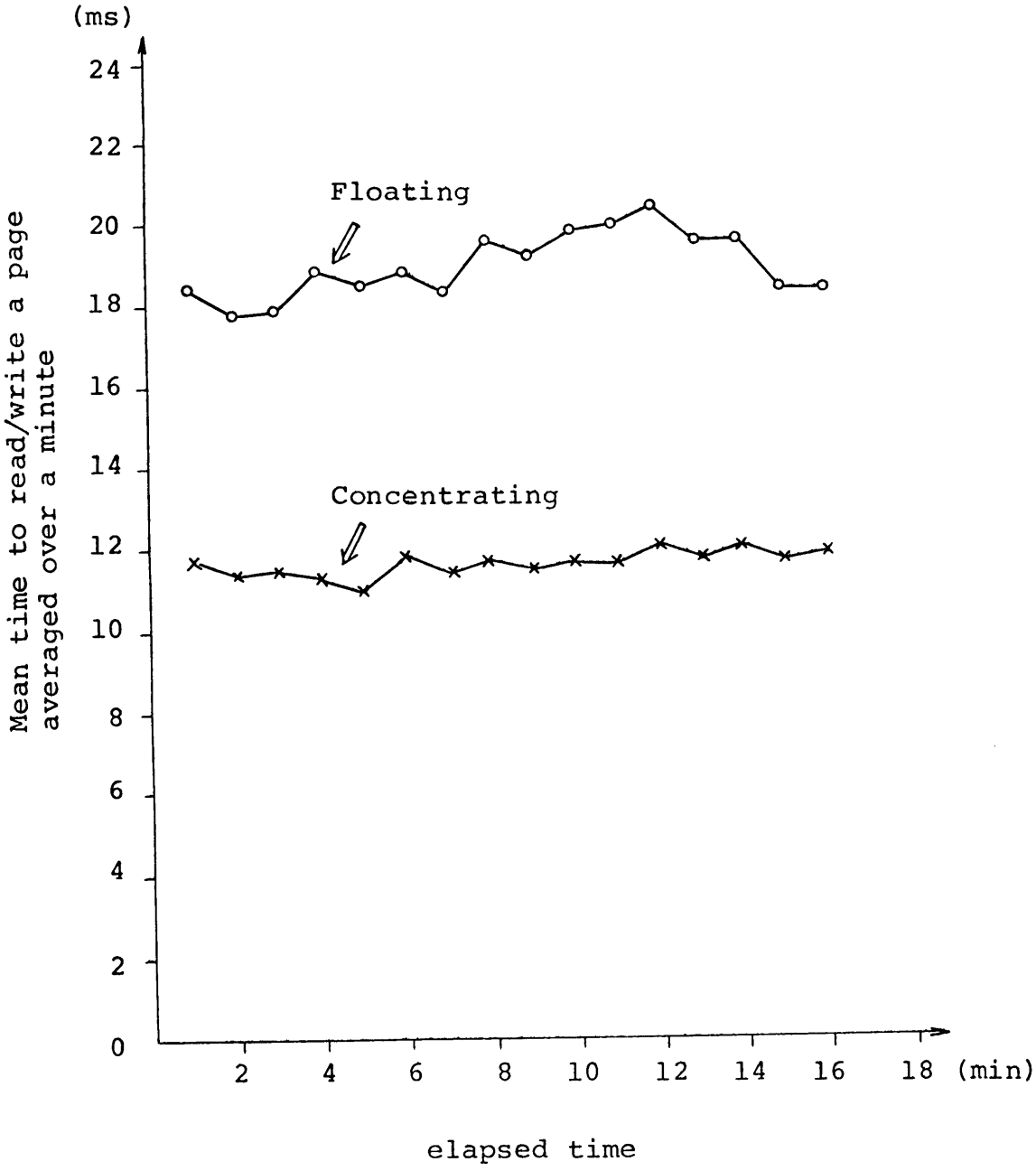


図4.14　平均ページ入出力実行時間の時間変化

The mean page I/O time versus
elapsed time

非常に低く，既述のようにページ入出力効率はスワップ・アウト・コストとスワップ・イン・コストの和で評価できることから，ほぼ期待された効果を実現できたとみなせる。

4.6 結　　言

仮想メモリ・システムにおけるページ入出力実行時間の短縮を目的として，二次メモリ管理方式につき論じた。まず従来方式の優劣を論じ，ついでより優れた効率が期待できる「集中方式（Concentrating Technique）」を提案した。本方式は既に日立のオペレーティング・システムＶＯＳ３において実用化され，稼動中である。以下に本章で得られた結果をまとめる。

(1) 代表的な従来方式である浮動方式と固定方式のページ入出力効率を解析的に比較した。このため，従来ほとんど知見が得られていない，プログラムのページ参照動作と磁気ディスクの入出力効率の相互関係を解析した。評価関数としてはインタラクティブなＴＳＳコマンドの実行にともなうページ入出力実行時間を用いた。

浮動方式，固定方式は各々ページ出力効率，ページ入力効率において優れている。浮動方式においてはワーキングセット内のページに対応するスロットが磁気ディスク・ボリウム上で散在するが，この散在度が増大するとスワップ・イン時の効率が低下する。両方式の優劣は散在度の大小により定まる。解析の結果，散在度はページに書きこみが行なわれる確率に依存すること，ページ・フォールト発生率の影響は比較的小さいこと，などが明らかになった。以上より，両方式の優劣はプログラムのページ参照特性に依存するという結論が得られた。

(2) アドレス空間上のページを二次メモリ上のスロットに写像する際の自由度に着目すると，浮動方式は自由度最大，固定方式は自由度最小である。この自由度をパラメータとして，両従来方式を包含したモデルを設定し，評価関数であるページ入出力実行時間を最小化する最適自由度を求めた。さらに実際の磁気ディスクの特性値を考慮して，実用的な範囲で最適自由度を与える「集中方式（Concentrating Technique）」を提案した。集中方式は，シリンダの割当ては固定，シリンダ内のスロット割当ては浮動である。集中方式をオペレーティング・システムＶＯＳ３において実現し，大型計算機ＨＩＴＡＣ　Ｍ－１８０を用いて実測評価を行なった。従来方式と比較して，ページ入出力実行時間の平均値，標準偏差につきそれぞれ約３８％，３５％の削減が観察された。

本章では，ページ入出力効率と二次メモリのスペース利用率の関係は論じなかったが，これは理論的検討が必要な問題である。また，本章では実メモリと磁気ディスク間の情報転送効率のみに着目したが，一般に広くメモリ階層という観点から各階層間の総合的情報転送効率について検討することは，今後重要な分野と考えられる。

第5章　多重プログラミング・システムにおける一般資源管理方式

5. 多重プログラミング・システムにおける一般資源管理方式

5.1 概　　要

第２章の議論において，利用率の高いボトルネック資源の割当てにおける優先度の決定がシステム性能に与える影響を求めた。また第３，４章でシステムがページング・ネックに陥ることを防止する方法を述べた。システム全体の性能向上をはかるための残された問題として，本章ではシステムが非ページング・ネックであるという仮定のもとに，資源割当て優先度の最適な決定法について述べる。

多重プログラミング・システムにおいて，資源管理の目的は処理能力と応答性の向上にある。一方の向上が必ずしも他方の向上につながらないため，両者の均衡的向上が要求される。これを実現するためには，システム全体の情報の流れを最適化するような大局的な制御方式もしくは資源管理方式が必要である。しかし，従来の資源管理方式においては各資源ごとに管理系が独立していたため，性能向上は局所的最適化にとどまり，大局的観点からこの目的を達成することはできなかった。

本章では，処理能力と応答性の均衡的向上を目的とした資源管理方式である「一般資源管理方式ＧＲＭ（General Resources Manager）」を提案している[N5)N6)]。ＧＲＭでは，オペレーティング・システムで割当て優先度を操作できる諸資源について，各々の管理系がこの目的達成のために連携してフィードバック制御を行なう。本章では5.2節において，フィードバック制御のための基本概念を整理する。5.3節ではＧＲＭの制御方策を提示し，さらに具体的に実メモリ，ＣＰＵ，入出力装置の割当て優先度決定法を述べる。また5.4節ではＧＲＭを日立のオペレーティング・システムＶＯＳ３において実現することにより得られた実測評価結果を，従来方式との比較において述べる。また，各種の条件のもとで行なったシミュレーション評価結果をも示す。さらに，第２章の議論を用いて，ＧＲＭのもとでのシステム性能の理論的な予測法についても述べる。

5.2 処理能力と応答性の均衡的向上

まず用語を整理する。「トランザクション」は第２章で定義したように資源の割当て要求単位であり，ＴＳＳコマンドやバッチ・ジョブに対応する。「資源」は論理的な単位を意味するので，ＣＰＵはシングル・プロセッサ，マルチ・プロセッサによらず１つの資源である。実メモリは個々のページでなく，ワーキングセット[D3)]を割当ての単位とする。

ＧＲＭの制御方策と，これを具体化する資源割当て優先度の決定法を提示する準備として，「状態」と「制御目標」を以下に定義する。システムの「状態」とは，資源割当て待ち行列長，未割当て実メモリ量などのような各種の状態量から定められるものであり，これを用いて時々刻々変化するシステムの動作特性を数学的に扱うことができる。ただしここでは次にのべる制御目標を達成するために，システムの「状態」は諸資源の利用率およびシステム内の各トランザクションに供給される資源サービス量という２種類の状態量から定められるとする。

(1) 処理能力向上のための資源利用率制御

処理能力とは単位時間あたりのシステムの仕事量であり，システム内の各トランザクションに供給される資源サービス量の総和として与えられる。資源の供給しうる最大処理量がハードウェアの制約のため一定という条件のもとでの処理能力の向上は，ソフトウェアの制御を用いて諸資源の利用率を高めることにより達成できる。資源 j の利用率を u_j，目標値域下限値を ρ_j とする。$u_j \geq \rho_j$ のときビジー，$u_j < \rho_j$ のときアイドルと2値で表現する。J 個の資源が存在するとき，2^J 個の相異なる状態が存在する。制御目標は次式で与えられる。

$$u_j \geq \rho_j \qquad (j = 1, 2, \cdots) \qquad (5.1)$$

ただし処理能力向上のためにはＣＰＵと入出力装置がビジーであればよいので，j はＣＰＵないし入出力装置を表わす。

上記目標を達成するための制御を資源利用率制御とよぶ。資源利用率制御を行なうためには，各トランザクションの資源使用上の特性を識別する必要がある。トランザクション i の資源 j 使用特性値 v_{ij} は，i と他のトランザクションとの競合がないという前提のもとでの i による j の利用率として定義される。v_{ij} が大きいとき，i を j のヘビイ・ユーザとよぶ。

(2) 応答性向上のためのサービス分配制御

トランザクションには緊急度の高いものと低いものがある。したがって応答性とは，各トランザクションの要求する資源サービス量と実際の供給量との合致度により表わされる。応答性の向上は次のようにして達成できる。

トランザクション i がシステムに到着してから現在までの間にこれに供給された資源サービス量の累計（resource service count）$\Lambda_i(\tau)$ は次式で与えられる。

$$\Lambda_i(\tau) = \sum_j a_j \lambda_{ij}(\tau) \qquad (5.2)$$

τ は i の到着以来の経過時間（transaction age）である。$\lambda_{ij}(\tau)$ は，τ の間に i に与えられた資源 j のサービス量であり，a_j は重み係数である。$\lambda_{ij}(\tau)$（$j = 1, 2, \cdots$）の算定の際，各種の資源のサービスはservice unit（su）という共通単位に換算される。

$\Lambda_i(\tau)$ ないしその傾き $d\Lambda_i(\tau)/d\tau$ を予め与えた目標関数に合致させる制御をサービス分配制御とよぶ。$\Lambda_i(\tau)$ を目標関数であるPolicy Curve $f_i(\tau)$ 以上に保つという制御方式が提案されているが[B6]，高負荷状態に対応するため $f_i(\tau)$ をかなり低く設定する必要があり，中ないし低負荷状態で制御効果が低下する[S7]。この解決策は，負荷のレベル WL（Workload Level）に応じて f_i を変化させる方式である（図5.1参照）。$d\Lambda_i(\tau)/d\tau$ を資源サービス率（resource service rate）とよび r_i で表わすとき，r_i の目標関数としてPerformance Objective $g_i(WL)$ を設定する方式はＳＲＭ（System Resources Manager[L3]）において実現された。$g_i(WL)$ は $df_i/d\tau$ に対応し，負荷レベルの単調減少関数である。

本論文では $g_i(WL)$ を目標関数とする。いま，g_i の逆関数を g_i^{-1} とするとき，次式で与えられる値を i のNormalized Workload Level とよび NWL_i とかく（図5.1参照）。

$$NWL_i = g_i^{-1}(r_i) \qquad (5.3)$$

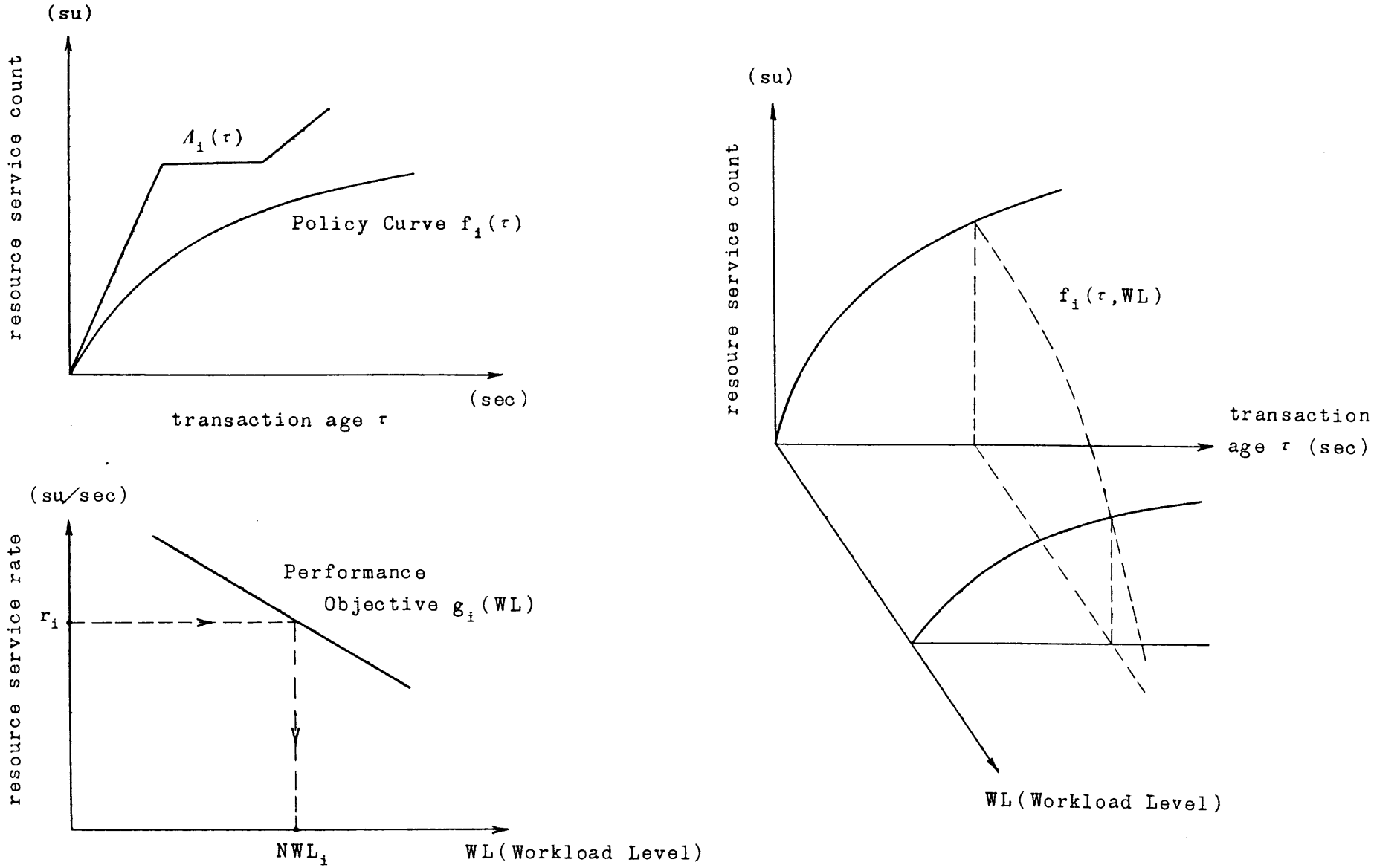


図5.1　サービス目標関数

Service Objective Functions

なお r_i は，実際には $\Lambda_i(\tau)$ の差分として近似的に与えられる。

状態はシステム内の各トランザクションの NWL より定められる。制御目標は各トランザクションの NWL を一致させることであり，次式で与えられる。ただしいま，システム内に I 個のトランザクションが存在するとする。

$$| NWL_i - \overline{NWL} | \leq \xi_i \quad (i = 1, 2, \cdots, I) \qquad (5.4)$$

ここで $\overline{NWL}$ は NWL_i の平均値 $\left(= \sum_{i=1}^{I} NWL_i / I\right)$ を表わす。ξ_i は偏差の上限値であり，式(5.4)をみたすか否かにより 2^I 個の状態が存在する。* なお，上記目標は資源サービス率の絶対値に関するものではない。むしろ，システム内の他のトランザクションとのサービス分配の比率を g_i により規定するものである。

5.3 一般資源管理方式（General Resources Manager）

5.3.1 資源割当ての依存関係にもとづく制御方策

式（5.1），（5.4）を達成するための資源管理方式につき述べる。これを「一般資源管理方式(General Resources Manager：GRM）」とよぶ。本方式の特徴は，資源割当ての依存関係にもとづいて諸資源の割当てを相互に関連づけた点にある（ただし本方式で対象とする「資源」は，優先度を操作できる割当て待行列がオペレーティング・システムに存在するものに限られる）。資源 j を割り当てられない限り資源 j' に対する割当て要求を発生できないトランザクションが存在するとき，j を j' の predecessor，j' を j の successor とよび，次のように表わす。

諸資源に関する上記のような関係を「資源割当ての依存関係」とよぶ。いま，j がビジー，j' がアイドルであるとき，j' のヘビイ・ユーザに対する j の割当て優先度を増加すれば j' の利用率は上がる。また，サービス分配はビジーな資源の割当てにより決定されるので，j の割当て優先度を変更することにより NWL のバラツキを低減できる。

ＣＰＵ，入出力装置，実メモリ，仮想メモリについての割当ての依存関係を図5.2に示す。たとえば実メモリがビジーでＣＰＵがアイドルのとき，ＣＰＵの利用率はＣＰＵバウンドなトランザクションをスワップ・イン（実メモリ内に収容）することにより向上する。スワッピングは応答時間を制御するためにも有効である。一方，ＣＰＵがビジーで入出力装置がアイドルのとき，入出力装置の利用率はＩＯバウンド・トランザクションのＣＰＵ割当て優先度を上げることにより向上する。このとき，ＣＰＵ割当て優先度の変更はサービス分配制御の手段としても有効である。

上記議論より，式（5.1），（5.4）を達成するための制御方策を以下に与える。

(1) 資源利用率制御方策

---

* 資源利用率，サービス分配の両制御系を考えると，システムの相異なる状態数は 2^{I+J} となる。

アイドルな資源のpredecessorであるビジーな資源の割当てに関し，アイドルな資源のヘビイ・ユーザの優先度を上げる。

(2) サービス分配制御方策

ビジーな資源の割当てに関し，NWLが相対的に大なるトランザクションの優先度を上げ，これが小なるトランザクションの優先度を下げる。

（例）

システム内にI個のトランザクションが存在するとき，ＣＰＵ，入出力装置＃１，入出力装置＃２，実メモリの４個の資源について，上記制御方策(1)(2)の実行される状態を例示する。資源がビジーのとき１，アイドルのとき０とし，2^3，2^2，2^1，2^0の各ビットをそれぞれ実メモリ，ＣＰＵ，入出力装置＃１，入出力装置＃２に対応させる。また，NWL_iが式（5.4）をみたすとき１，みたさないとき０とし，2^{I+3}～2^4の各ビットをそれぞれ各トランザクションに対応させる。実メモリの割当てについて制御方策(1)を実行する状態は下位４ビットが1000～1110の７通り，ＣＰＵについては0100～0110，1100～1110の６通り，入出力装置＃１については0010，0011，1010，1011の４通り，入出力装置＃２については0001，0011，1001，1011の４通りである（ただし，トランザクションは入出力装置＃１か＃２のいずれか一方のみを用いると仮定する）。制御方策(2)を実行するのは，いずれの資源についても，それがビジーでかつ上位のIビットが全て１以外の場合であるから，$8(2^I-1)$通りの状態においてである。

5.3.2 資源割当て優先度の決定法

制御方策を実現するため，周期的に資源割当て優先度を決定する。なお以下，実メモリ，ＣＰＵ，入出力装置の３種の資源について考察する（図5.2の破線内参照）。

(1) 実メモリ割当て優先度 $P_i(\mathrm{MEM})$

トランザクションiに対する実メモリ割当て優先度 $P_i(\mathrm{MEM})$は式（5.5）により与えられる。ただしシステム内のトランザクションのワーキングセット・サイズの総和が実メモリ容量以下（実メモリがアイドル）であれば，全トランザクションに実メモリが割り当てられ，スワッピングは行なわれない。

$$P_i(\mathrm{MEM}) = \alpha_{\mathrm{M}}^{(1)} CP_i(\mathrm{MEM}) + \alpha_{\mathrm{M}}^{(2)} IO_i(\mathrm{MEM}) + \alpha_{\mathrm{M}}^{(3)} WL_i(\mathrm{MEM}) \qquad (5.5)$$

$\alpha_{\mathrm{M}}^{(1)}$，$\alpha_{\mathrm{M}}^{(2)}$，$\alpha_{\mathrm{M}}^{(3)}$は，処理能力と応答性の均衡をとるための重み係数である。ここでＣＰＵと入出力装置は実メモリのsuccessorである。なお簡単のため，以下，トランザクションiは入出力装置＃mのみを使用すると仮定する。このとき式（5.5）の各項は次式で与えられる。

$$\begin{cases} CP_i(\mathrm{MEM}) = v_{i,\mathrm{CPU}}\ \delta_{\mathrm{CPU}}(\rho_{\mathrm{CPU}} - u_{\mathrm{CPU}})^2 \\ IO_i(\mathrm{MEM}) = v_{i,\mathrm{IO}m}\ \delta_{\mathrm{IO}m}(\rho_{\mathrm{IO}m} - u_{\mathrm{IO}m})^2 \\ WL_i(\mathrm{MEM}) = \nu_i(NWL_i - \overline{NWL})\,|\,NWL_i - \overline{NWL}\,| \end{cases} \qquad (5.6)$$

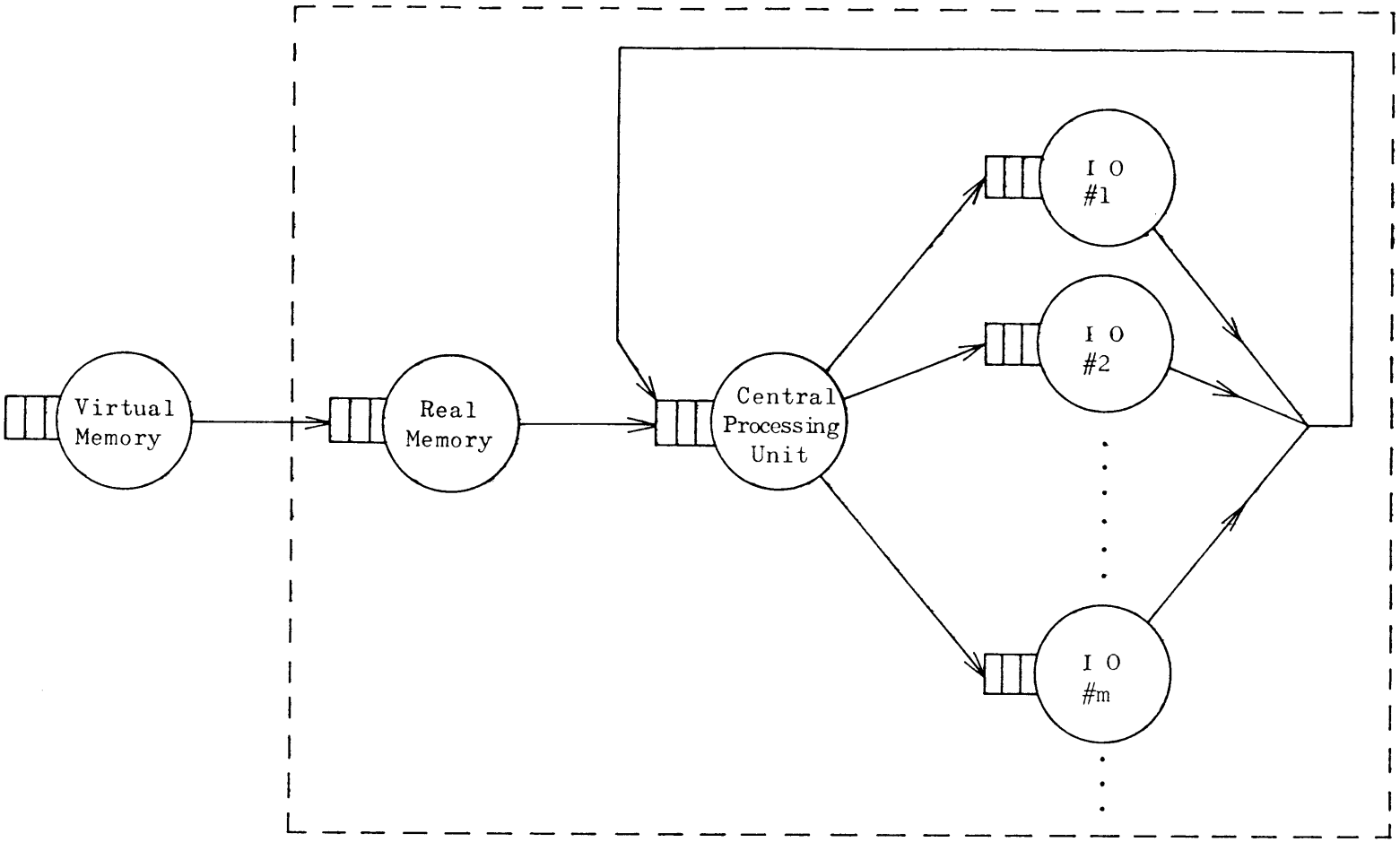


図 5.2　資源割当ての依存関係の一例

An example of resource allocation constraints

$$\delta_j = \begin{cases} 1 : u_j < \rho_j \\ 0 : u_j \geq \rho_j \end{cases} \quad (j = \mathrm{CPU}, \mathrm{IO}m) \qquad (5.7)$$

$$\nu_i = \begin{cases} 1 : |NWL_i - \overline{NWL}| > \xi_i \\ 0 : |NWL_i - \overline{NWL}| \leq \xi_i \end{cases} \qquad (5.8)$$

u_j, v_{ij}, NWL_i は周期的に測定する。v_{ij} は，たとえば i が Δt 演算処理を行なうごとに入出力命令を発行して入出力装置＃m を $\Delta t'$ 用いるとき，次のようにして求める。

$$\begin{cases} v_{i,\mathrm{CPU}} = \Delta t / (\Delta t + \Delta t') \\ v_{i,\mathrm{IO}m} = \Delta t' / (\Delta t + \Delta t') \end{cases} \qquad (5.9)$$

P_i（MEM）が大なるトランザクションから順に実メモリが与えられる。ただし過度のスワッピング・オーバヘッドを防ぐため，スワッピングを実行するのは次式がなりたつ場合のみとする。

$$P_{k'}(\mathrm{MEM}) - \sum_{k \in \Xi} P_k(\mathrm{MEM}) \geq SIT + |\Xi| SOT \qquad (5.10)$$

k' は実メモリを新たに割り当てられるトランザクション，Ξ はかわりに実メモリを剝奪されるトランザクションの集合，$|\Xi|$ はその要素数を表わす。SIT, SOT はそれぞれスワップ・イン，スワップ・アウトの閾値である。式（5.6）で，各項が目標値と実現値の差の2乗の形をとる理由は，この閾値を設定したためである。

(2) CPU割当て優先度 P_i（CPU）

トランザクション i に対するCPU割当て優先度 P_i（CPU）は式（5.11）により与えられる。ただしCPU割当て優先度の変更が行なわれるのはCPUがビジー（$u_{\mathrm{CPU}} \geq \rho_{\mathrm{CPU}}$）の場合に限られる。

$$P_i(\mathrm{CPU}) = \alpha_{\mathrm{P}}^{(1)} IO_i(\mathrm{CPU}) + \alpha_{\mathrm{P}}^{(2)} WL_i(\mathrm{CPU}) \qquad (5.11)$$

$$\begin{cases} IO_i(\mathrm{CPU}) = (1 - \delta_{\mathrm{IO}m}) OLDIOm \\ \qquad\qquad + \delta_{\mathrm{IO}m} v_{i,\mathrm{IO}m} (\rho_{\mathrm{IO}m} - u_{\mathrm{IO}m}) \\ WL_i(\mathrm{CPU}) = \nu_i (NWL_i - \overline{NWL}) \end{cases} \qquad (5.12)$$

$\alpha_{\mathrm{P}}^{(1)}$, $\alpha_{\mathrm{P}}^{(2)}$ は重み係数である。入出力装置＃m はCPUの successor なので式（5.11）第1項が得られる。式（5.12）で $OLDIOm$ は前回 P_i（CPU）を算出した際の IO_i（CPU）を表わす。入出力装置＃m の利用率が目標値域内にあれば，その過度の変動を防止し現状を保つことが望ましいので，本項を設ける。CPUは P_i（CPU）が大なるトランザクションから割り当てられる。ここで優先制御は preemptive 方式とし，トランザクション k にCPUが割り当てられているとき，$P_{k'}$（CPU）$> P_k$（CPU）なる k' が割当て要求を発生すると，その時点でCPUは k' に割当て変更される。

上記方式を，従来のダイナミック・ディスパッチング制御[R5]と比較すると，入出力装置利用率向上という目的については共通しているが，個々の入出力装置を識別している点，及びサービス分配制御をも行なっている点が異なる。また，従来のタイム・スライシング制御[K9]は，上記方式のサービス分配制御に概念的に含まれる。

(3) 入出力装置＃m 割当て優先度 P_i（IOm）

トランザクション i に対する入出力装置＃mの割当て優先度 P_i（IOm）は式（5.13）により与えられる。ただし P_i（IOm）の変更が行なわれるのは，入出力装置＃mがビジー（$u_{IOm} \geq \rho_{IOm}$）の場合に限られる。

$$P_i(IOm) = \alpha_C^{(1)} CP_i(IOm) + \alpha_C^{(2)} WL_i(IOm) \qquad (5.13)$$

$$\begin{cases} CP_i(IOm) = (1-\delta_{CPU})\,OLDCP \\ \qquad\qquad + \delta_{CPU}\, v_{i,CPU}(\rho_{CPU} - u_{CPU}) \\ WL_i(IOm) = \nu_i(NWL_i - \overline{NWL}) \quad (= WL_i(CPU)) \end{cases} \qquad (5.14)$$

$\alpha_C^{(1)}$, $\alpha_C^{(2)}$は重み係数である。ＣＰＵは入出力装置＃mのpredecessorであるばかりでなく，successorでもあることから式（5.13）第１項がえられる。また，式（5.14）の$OLDCP$は前回P_i（IOm）を算出した際のCP_i（IOm）であり，その設定理由は式（5.12）の$OLDIOm$と同様である。

P_i（IOm）が大なるトランザクションの発生した入出力処理から実行される。ただし，入出力装置割当てにおける優先制御はnon-preemptive方式とする。すなわちトランザクションkの発生した入出力処理実行中に，$P_{k'}$（IOm）＞P_k（IOm）なるk'が入出力処理を発生しても，kの入出力処理完了まで待たされる。

5.4　実験と評価

5.4.1　実測評価

一般資源管理方式ＧＲＭ（General Resources Manager）をオペレーティング・システムＶＯＳ３の上で実現し，ＨＩＴＡＣ　Ｍ－１８０を用いて実測評価を行なった。本実験においては，実メモリとＣＰＵの割当てのみに着目し，入出力装置のスケジューリングは対象外とした。負荷としてＣＰＵバウンド・トランザクションを設定したので，入出力装置のスケジューリングは実測結果に影響しない。なお，P_i（ＭＥＭ）とP_i（ＣＰＵ）の算出周期はそれぞれ約４秒，約２秒とした。また，資源サービス量の単位として，ＣＰＵについては１０,０００命令の実行を１ｓｕ，入出力装置については入出力処理１回を１ｓｕとし，式（5.2）の重み係数はＣＰＵ，入出力装置とも１０とした。

実メモリ割当てのみを制御手段とするＳＲＭ[13]（System Resources Manager）はＧＲＭの一部として実現できる。以下，ＧＲＭの性能をＳＲＭとの比較において考察する*。ＳＲＭのもとでのＣＰＵ割当ては，バッチ・トランザクションはダイナミック・ディスパッチング制御にしたがい，ＴＳＳトランザクション（terminal-initiated transactions）の優先度は同一で固定とする。バッチ・トランザクションによるＴＳＳトランザクションの応答性低下を防ぐため，ＴＳＳトランザクションの優先度はバッチ・トランザクションの優先度より常に高く設定する。

実測はPartⅠとPartⅡとからなる。PartⅠはＴＳＳトランザクションの応答性，PartⅡはバ

---

* 本実験では，ＳＲＭの実メモリ割当て方式はＧＲＭと完全に同一と仮定した。したがって比較したＳＲＭはＩＢＭ社のＭＶＳのＳＲＭと正確には一致しない。

ッチ・トランザクションに対する資源サービスの分配に関する測定である。実測で用いた機器構成は，図5.3に示すＨＩＴＡＣ　Ｍ－１８０システムであり，２端末が結合されている。実メモリは大容量（６Ｍバイト）なので，本実測において実メモリはほとんど常にアイドルであり，ボトルネックにはならない。

(1)　実測Ｐａｒｔ Ⅰ

端末は，測定時間を通じて常に稼動中（アクティブ）とした。端末＃１は，約２０秒の思考時間が終了するごとにＥＤＩＴコマンドを入力し，短かいインタラクティブなトランザクションを実行する。一方，端末＃２は長いエグゼキュティブなトランザクションであるＦＯＲＴＲＡＮジョブのコンパイル＆実行を行なう。ＦＯＲＴＲＡＮジョブは入出力命令を発行することなく約１０秒間演算を実行するものである。このトランザクションは，測定時間を通じて，端末＃２より繰り返し実行される。

上記ＴＳＳトランザクションと並行して，２本のバッチ・トランザクションを実行した。これらはともにプログラム・サイズ２５０Ｋバイトの ＦＯＲＴＲＡＮ ジョブである。

設定したサービス目標関数を図5.4に示す。図5.4において，Aは資源サービス量の累計（resource service count）を表わす。MAは過度のスワッピング・オーバヘッド防止用パラメータであり，いったんスワップ・インされたトランザクションはMA以上サービスを受けるまでは，原則としてスワップ・アウトされない。図5.4は，ＴＳＳをバッチよりやや優先し，短かいトランザクションを長いトランザクションより優先したもので，標準的な設定である。

８分間の測定より観察された，インタラクティブなＴＳＳトランザクションの応答時間の度数分布を図5.5に示す。ＧＲＭのもとでの応答性は，ＳＲＭの場合に比べ，顕著な向上を示している。両者の差異の原因は，エグゼキュティブなＴＳＳトランザクションに対する資源割当て方式にある。ＧＲＭにおいては，エグゼキュティブなＴＳＳトランザクションのＣＰＵ割当て優先度は，それに供給されたＣＰＵサービス量にもとづいて式（5.11）により調節される。一方，ＳＲＭにおいてはこれが固定なので，エグゼクティブなＴＳＳトランザクションがインタラクティブなＴＳＳトランザクションの応答性を低下させている。ＳＲＭにおいて，エグゼキュティブなＴＳＳトランザクションをダイナミック・ディスパッチング制御の対象としても，この問題は本質的に解決されない。その理由は，ダイナミック・ディスパッチング制御は処理能力向上のみを目的とするものであり，サービス分配能力の著しい低下をひき起こす可能性があるためである。

(2)　実測 Ｐａｒｔ Ⅱ

本実測においては，ＴＳＳトランザクションは実行せず，６本の同一なバッチ・トランザクションを並行に実行した。これらは約３０Ｋステップ演算を行なうごとに入出力命令を発行するＦＯＲＴＲＡＮ ジョブであり，プログラム・サイズは約２５Ｋバイトである。６本のトランザクションを，緊急度の高いものと低いものと２グループに分類する。図5.6に示すサービス目標関数において，３本（＃１～＃３）は g_1 に，残りの３本（＃４～＃６）は g_3 に，それぞれ対応させる。すなわ

ち，トランザクション＃１～＃３には＃４～＃６より多くの資源サービスを供給することが必要であり，そのサービス分配制御の効果はNWLの標準偏差により測定することができる。

GRMのサービス分配制御効果を，式（5.11）の重み係数の比$\alpha_P^{(2)}/\alpha_P^{(1)}$の関数として表わしたのが図5.7である（ただし，$\alpha_M^{(2)}/\alpha_M^{(1)}$は1.0とした）。ここで，従来のダイナミック・ディスパッチング制御は常にＩＯバウンド・トランザクションをＣＰＵバウンド・トランザクションより優先する方式であり，$\alpha_P^{(2)}/\alpha_P^{(1)}=0$の場合に対応する。いま各トランザクションの入出力使用特性は等しいので，ダイナミック・ディスパッチング制御のもとで各トランザクションのＣＰＵ割当て優先度は等しくなる。

図5.7において，$\alpha_P^{(2)}/\alpha_P^{(1)}$を増加するにしたがって$NWL$の標準偏差が減少しており，サービス分配制御効果の向上を示している。図5.7には，測定時間中に各トランザクションに与えられた資源サービス量の総和すなわち処理能力をも示した。$\alpha_P^{(2)}/\alpha_P^{(1)}$を増加しても処理能力が低下していないが，これはトランザクションの入出力使用特性が同一のためである。一般に，$\alpha_P^{(2)}/\alpha_P^{(1)}$を増加すると処理能力は低下する可能性がある。たとえば，サービス目標関数の設定において，ＣＰＵバウンド・トランザクションをＩＯバウンド・トランザクションより優先すれば，処理能力低下の傾向が現れると予想される。すなわち，処理能力と応答性とは相矛盾する場合がある。このような場合には，制御系の優劣は性能（処理能力，応答時間）の絶対値でなく，性能を制御できる能力にもとづいて判断することが適切である。図5.7はＧＲＭの性能制御能力を示している。

5.4.2　シミュレーション評価

実測評価は，実メモリに余裕があり端末数も少ない状況で，ＣＰＵバウンド・トランザクションを用いて行なった。ここでは，より広い周囲条件のもとでのＧＲＭの効果を測定するために行なったシミュレーション評価につき述べる。シミュレーションにおいてはＣＰＵバウンド・トランザクションのみならずＩＯバウンド・トランザクションも想定したので，実メモリやＣＰＵに加え，入出力装置割当ても式（5.13）による優先制御にしたがうとする。なお，ＧＲＭと比較するためにシミュレートしたＳＲＭにおいては，実メモリ割当てはＧＲＭと同一，入出力装置割当てはFirst Come First Servedと仮定する。ＣＰＵについては，実測評価と同様に，バッチ・トランザクションに対しダイナミック・ディスパッチング制御[R5]を行ない，ＴＳＳトランザクションにはバッチ・トランザクションより高い固定の割当て優先度を与える。

両方式の比較は，ＣＳＳ（Computer System Simulator）言語の約２Ｋステップのモデルを開発し，５分間の状況をシミュレートすることにより行なった。

(1)　周 囲 条 件

シミュレーション評価に際し設定した周囲条件をまとめる。図5.8に想定したシステムの構成を示す。実メモリ容量は1.6Ｍバイト，端末数は４０とする。ＣＰＵの平均処理速度は0.5μs/stepとする。チャネル＃１（ＣＨ１）と＃２（ＣＨ２）にはトランザクションの用いるファイルが格納され

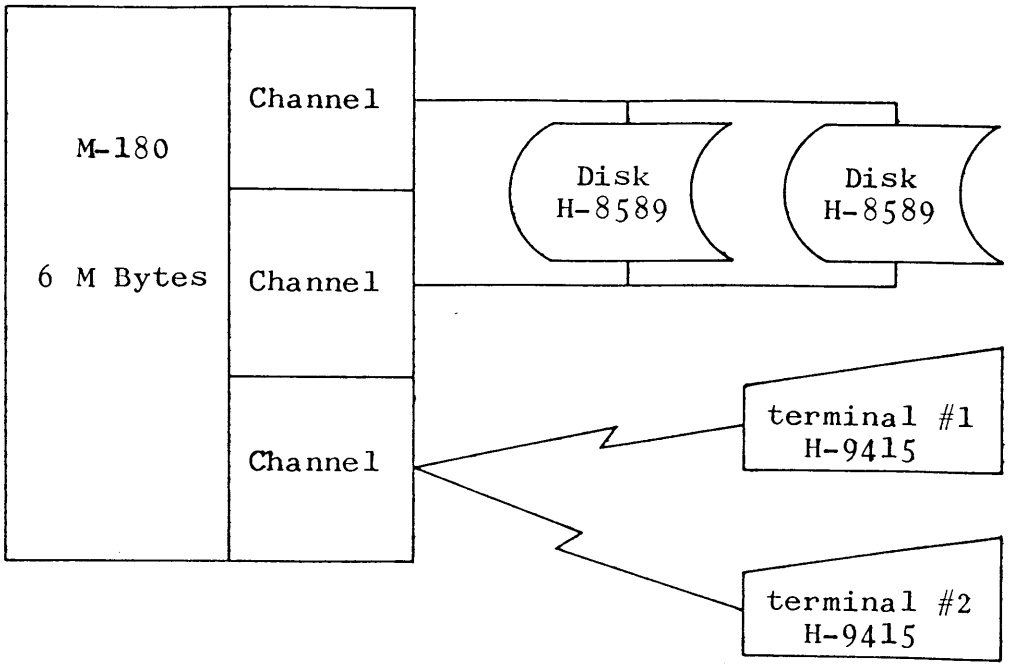


図 5.3 実測評価で用いた機器構成

The hardware configuration used in the experiment

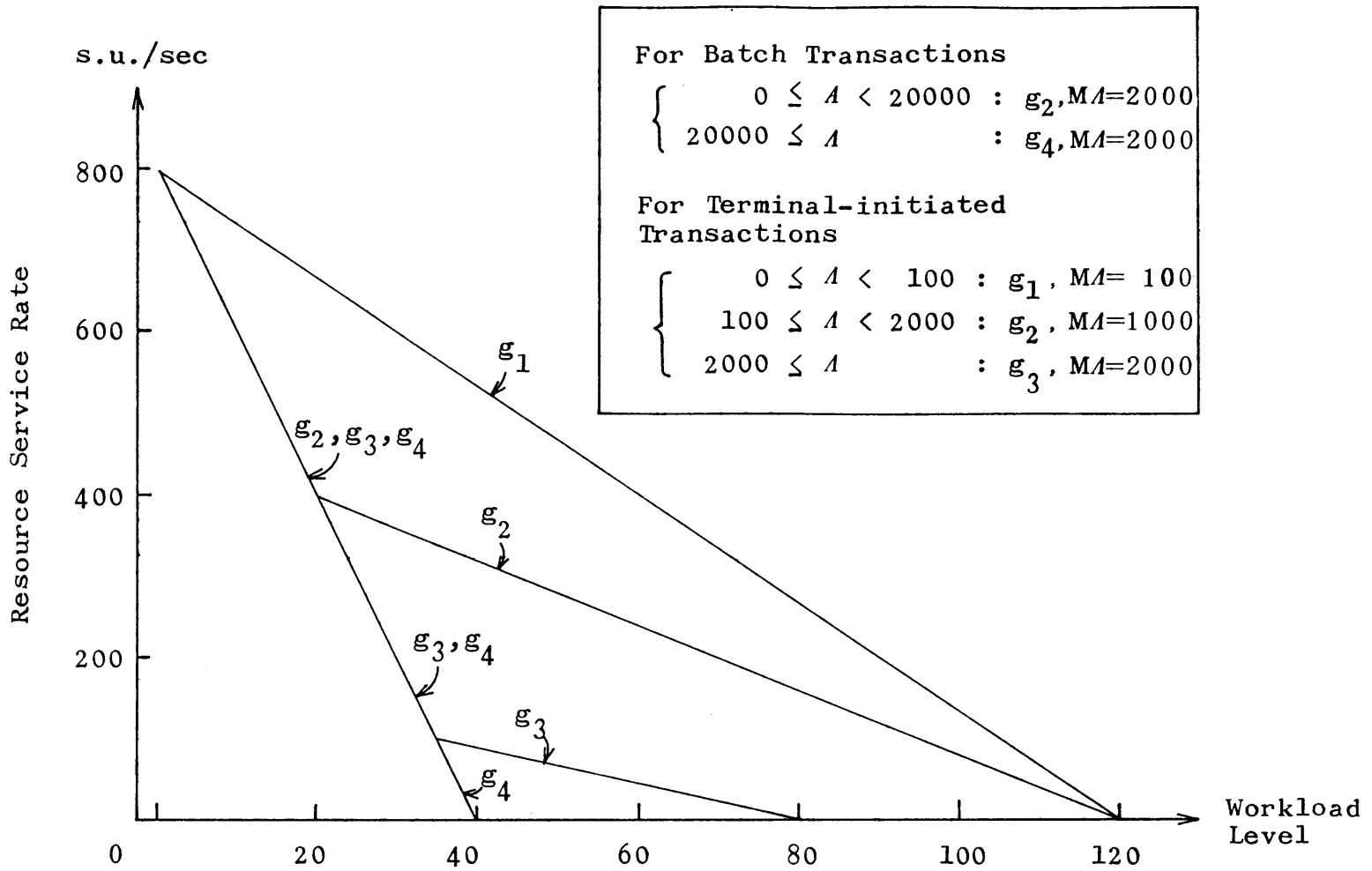


図 5.4　実測パートⅠで設定したサービス目標関数

The service objective functions specified in the Measurement Part I

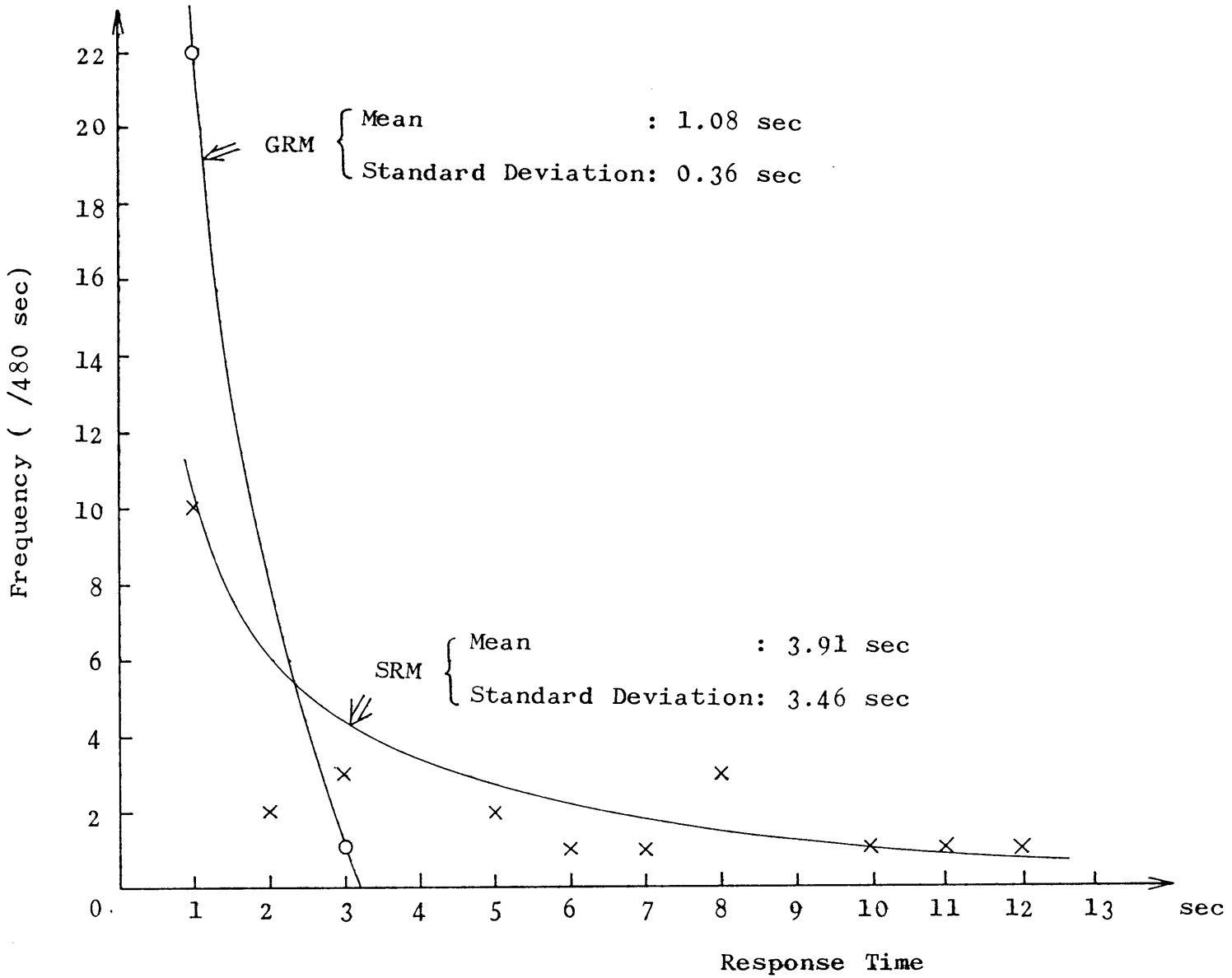


図 5.5　インタラクティブなTSSトランザクションの応答性の比較

Performance comparison of the two schedulers with respect to the responsiveness of interactive transactions initiated by terminal #1

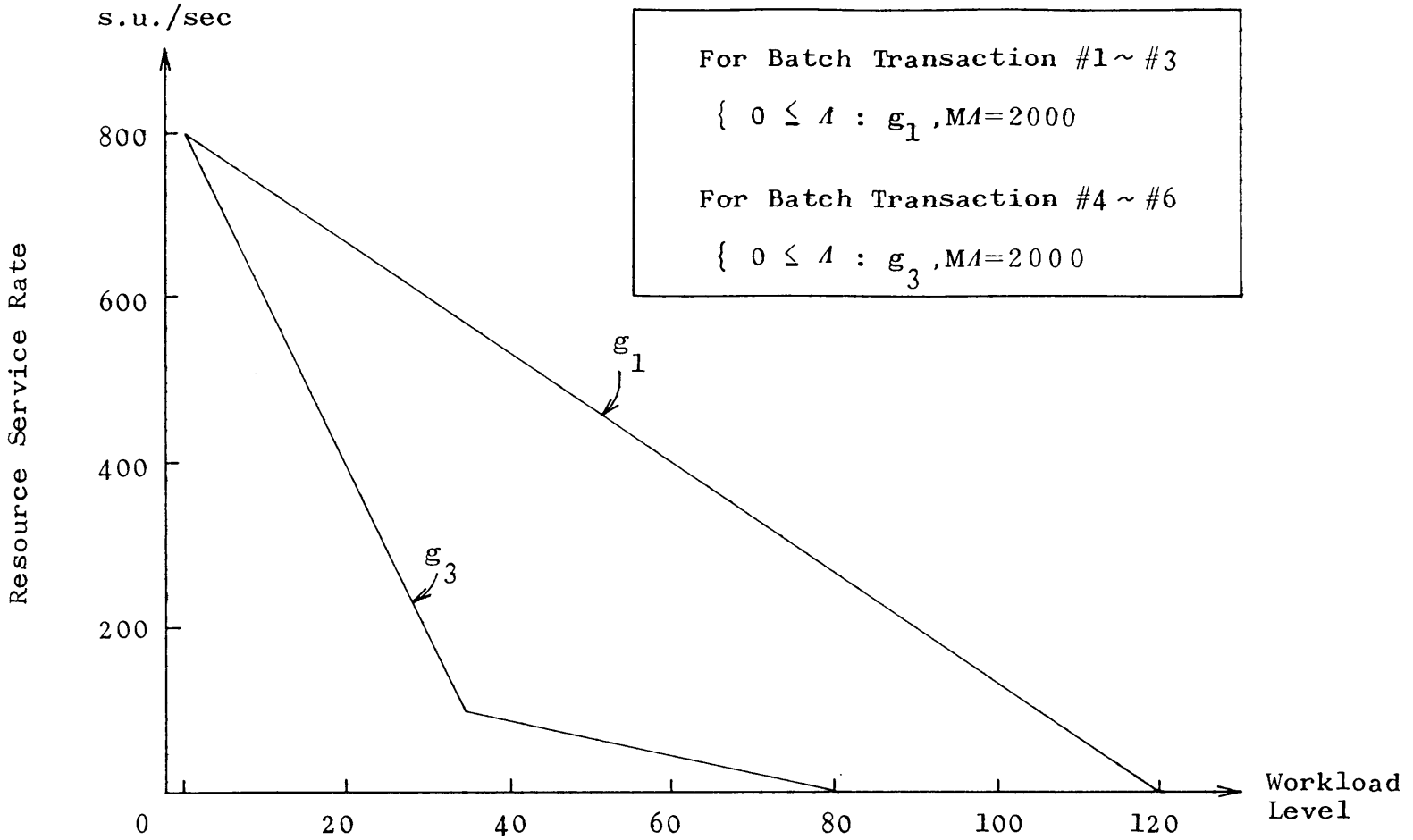


図 5.6　実測パートⅡで設定したサービス目標関数

The service objective functions specified in the Measurement Part II

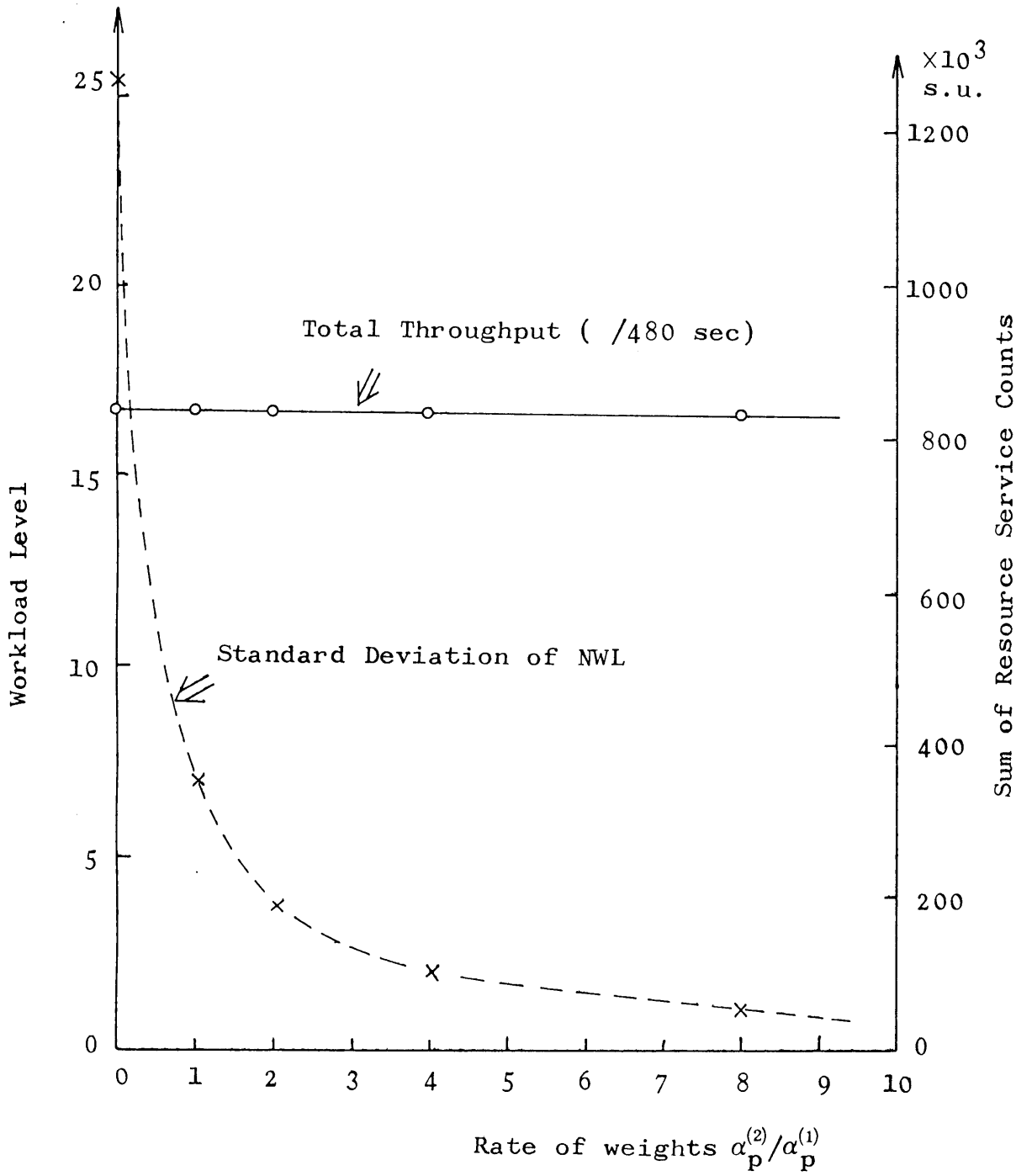


図 5.7　GRMのサービス分配制御効果

Effect of the GRM performance control on service distribution/throughput

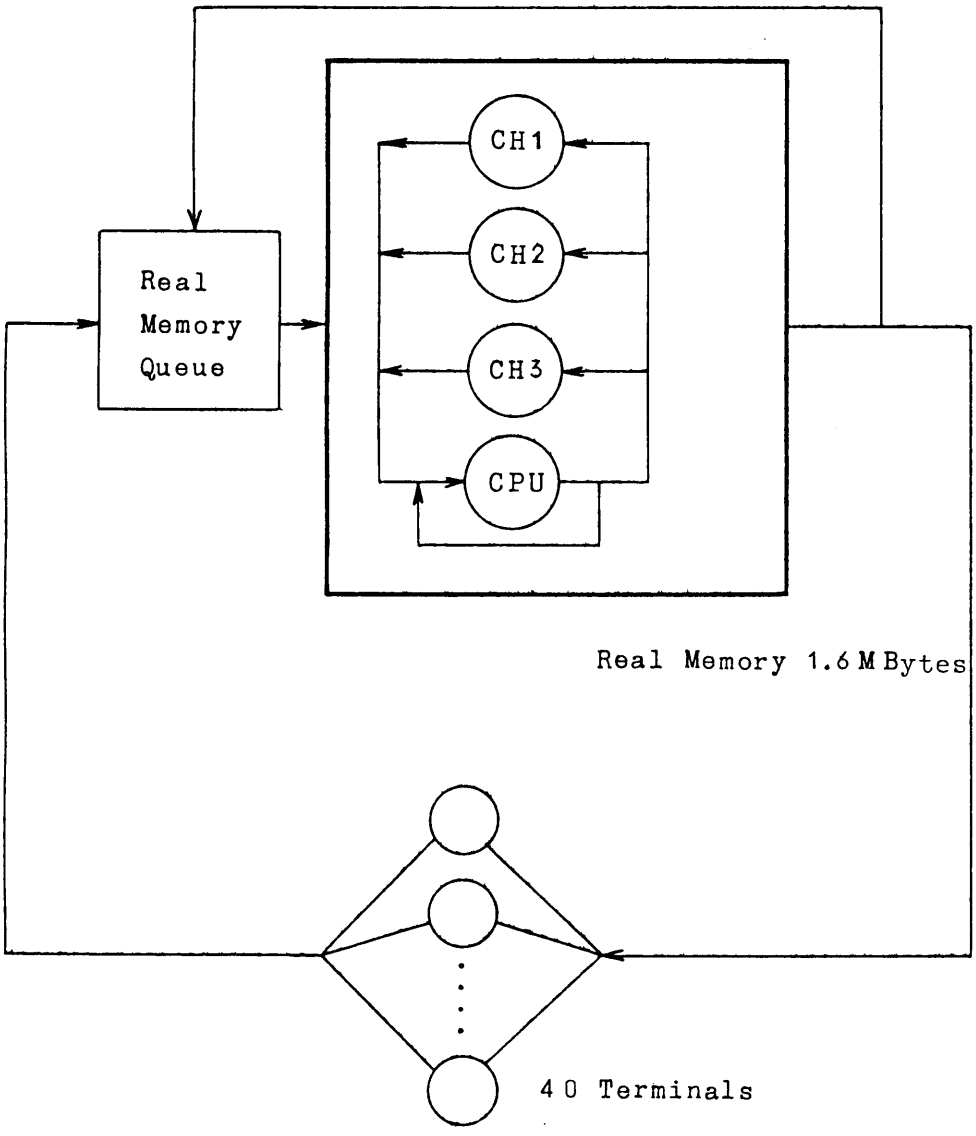


図 5.8　シミュレーション評価で設定したモデル

The system model simulated

たディスク群＃１と＃２がそれぞれ接続され，チャネル＃３（ＣＨ３）にはページング用のドラムが接続されるとする。すなわち本モデルにおいて，チャネルで入出力装置(群)を代表させる。

資源サービス量の単位は，実測評価と同じく，ＣＰＵの 10000 命令の実行，１回の入出力処理を各々１service unit（su）とした。また，式（5.2）の重み係数は，ＣＰＵ，入出力装置とも１０とした。また，サービス目標関数としては，図5.4と同様に，ＴＳＳをバッチよりやや優先し，短かいトランザクションを長いトランザクションより優先した標準的なものを設定した。これを図5.9に示す。

P_i（ＭＥＭ），P_i（ＣＰＵ），P_i（ＩＯm）（m＝１，２）の算出周期は，それぞれ約５秒，２秒，５秒とした。制御間隔を余りに短かくすることはオーバヘッドの増大を招く。したがって，中ないし長大なトランザクションを主な制御対象とする。トランザクションの到着直後には高い資源割当て優先度を与えることにより，制御間隔の間で実行完了してしまう短小なトランザクションに対しても応答性を保証できる。

想定した負荷はバッチとＴＳＳが混在したものであり，表5.1にその諸元を示す。表5.1において，入出力処理はディスクに対する８Ｋバイトの読みこみ／書き出しとする。

バッチ・トランザクションはシミュレーション開始時点で既にジョブ・キューに並んでおり，あるトランザクションが実行完了するとただちに同一の新たなトランザクションが再開されるとする。また，並行処理されるバッチ・トランザクションの本数（イニシエータ数）は８とする。なお，資源利用率制御の効果を確かめるために，ＣＰＵバウンドとＩＯバウンドの２種のバッチ・トランザクションを設定した。

端末は，シミュレーション実行中は常に稼動中（アクティブ）と仮定する。ＴＳＳトランザクションとして，Ｉ型（インタラクティブ）とＥ型（エグゼキュティブ）の２種のものを設定した。前者はテキスト編集などの処理を想定しており，モデルはH.A.Anderson[A1)]，A.L.Scherr[S2)]等の実測結果を参考にして定めた。後者はＦＯＲＴＲＡＮプログラムなどの実行処理を想定している。なお，表5.1において，トランザクションが実行完了までに供給される資源サービス量は，いずれも13200 suである（ただしＩ型ＴＳＳトランザクションは除く）。

(2) 性能比較結果

３種のパラメータを変えて測定を行なった。第１はトランザクションのワーキングセット・サイズ（実メモリ割当て量）wである。第２は端末特性であり，Ｉ／Ｅは４０端末のうちでＩ型，Ｅ型ＴＳＳトランザクションを発生する端末数の比を示す。第３はバッチ・トランザクションのチャネル（ディスク）使用特性であり，ＣＨ１／ＣＨ２はチャネル＃１(ディスク群＃１）を使用するトランザクション数とチャネル＃２(ディスク群＃２）を使用するトランザクション数との比である。この値が６／２のとき，チャネル＃１を使用するトランザクションの８３％，チャネル＃２を使用するトランザクションの５０％をＩＯバウンドとし，残りをＣＰＵバウンドとする。また，この値が4/4のとき，両者とも５０％をＩＯバウンドとし，残りをＣＰＵバウンドとする。

(a) サービス分配能力（応答性）

Ｉ型ＴＳＳ，Ｅ型ＴＳＳ，バッチの各トランザクションのprocessing time をそれぞれ表5.2，5.3，5.4に示す。processing timeは，トランザクションが到着してから実行完了までの経過時間を表わし，ＴＳＳトランザクションについては応答時間からメッセージ送信時間を除いた値に対応する。９０％accumulationは，processing timeがその値以下のトランザクション数が完了した総数の９０％を占める点である。number of completionにおいて，第１項は３００秒間に実行完了したトランザクション数である。第２項は，３００秒経過時点でまだ実行中のすべてのトランザクションについての資源サービス量累計の和を13200 su で除算し，完了した数に換算した値である（ただし表5.2のＩ型ＴＳＳトランザクションについては，第２項は小さいので無視する）。表5.4には，上記に加え定期的にシステム内の全トランザクションについて測定したNWLの標準偏差をも示す。

表5.2に示されたように，ＧＲＭにおいては測定パラメータが変化しても応答性はほぼ安定している。ＳＲＭでは，Ｉ／Ｅにおいて Ｅが増大すると応答性が低下する。低下の程度は，w＝４０ページ，すなわち実メモリ・ネックでないとき著しい。この理由は，ＧＲＭではＣＰＵ割当てに関しサービス分配制御を行なっているのに対し，ＳＲＭではこれを行なっていないためである。ＧＲＭでは，Ｅ型ＴＳＳトランザクションのＣＰＵ割当て優先度は，供給された資源サービス率にしたがって動的に調整される。しかしＳＲＭでは，常にＩ型ＴＳＳトランザクションと同一であり，したがって後から到着するＩ型ＴＳＳトランザクションにはただちにＣＰＵが割り当てられず，その応答性が低下する。

なお，バッチ・トランザクションのチャネル（入出力装置）使用特性の変化が応答性に与える影響は，ＧＲＭ，ＳＲＭいずれにおいても小さい。表5.3，5.4に示すように，w＝４０ページの場合，ＳＲＭではＥ型ＴＳＳトランザクションにサービスが偏る。Ｉ／Ｅ＝２８／１２，w=40ページのとき，バッチ・トランザクションの沈みこみが生じている。実メモリ・ネックでないとき，トランザクションの大半はスワップ・イン状態にあるので，スワッピングのみでサービス分配を制御することはできない。ＧＲＭとＳＲＭとのサービス分配制御能力の差異は，表5.4に示したNWLの標準偏差からも明らかである。

上記結果は，実測評価の結果と一致するものである。ただし，ＧＲＭとＳＲＭとのサービス分配制御能力の差異が，実メモリ・ネックでない場合（w＝４０ページ）に限らず，実メモリがネックの場合（w＝６０ページ）にも顕著であることがシミュレーションにより確認された。

(b) 処理能力

処理能力測定結果を表5.5に示す。表5.5において，throughput は３００秒間に各トランザクションに与えられた資源サービス量の総和を表わす。ＣＰＵ utilizationは全トランザクションによる利用率であり，かっこ内は特にバッチ・トランザクションによる値である。

ＧＲＭにおけるthroughput は，ＳＲＭより４～１１％向上している。向上効果は，チャネル

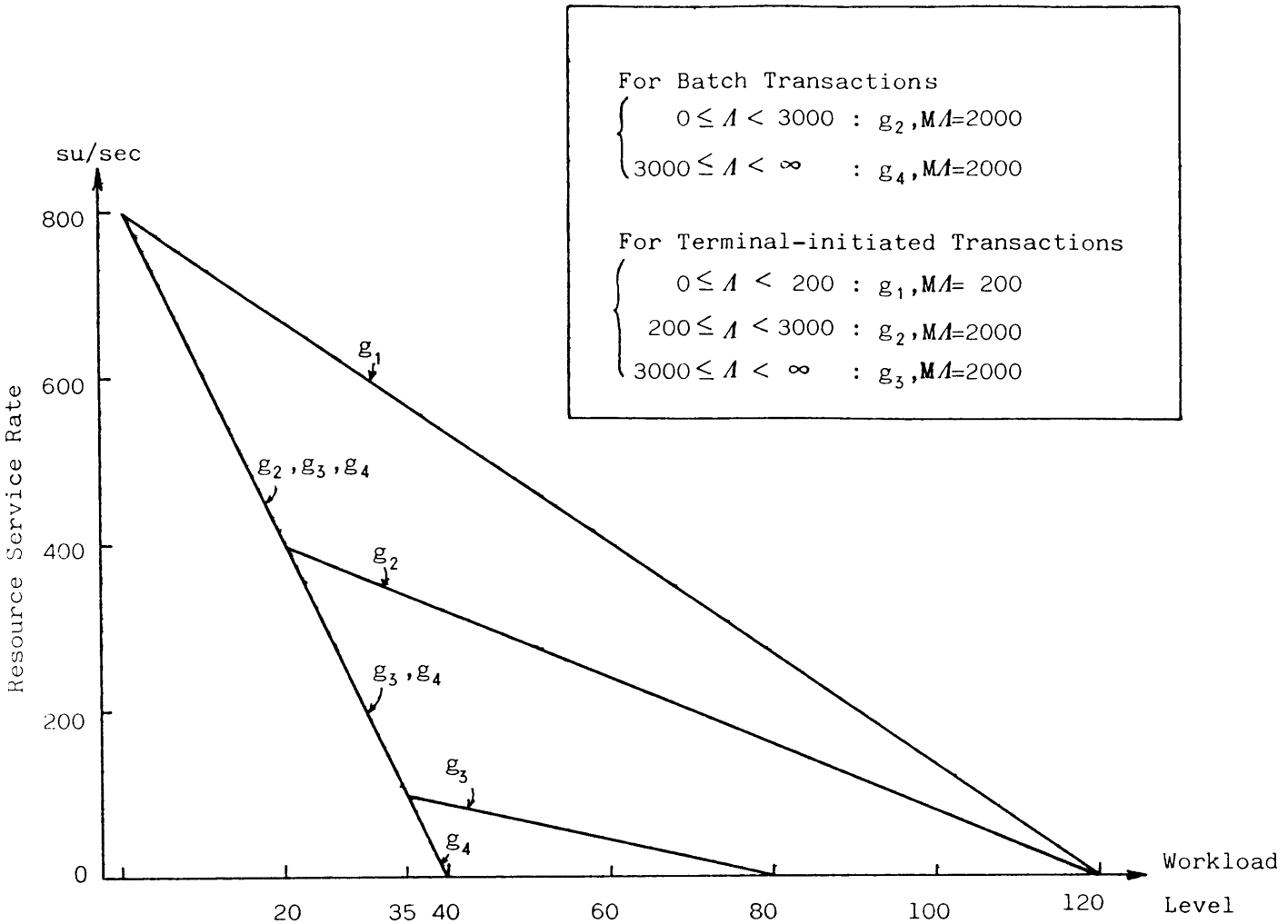


図 5.9 シミュレーションで用いたサービス目標関数

The service objective functions specified in the simulations

表 5.1　シミュレーション評価で想定した負荷

The characteristics of the simulated workload model

Transaction / Characteristics	Batch Transaction		Terminal-initiated Transaction	
	CPU-bound	I/O-bound	Interactive(I)	Executive (E)
Execution Requirements (steps)	12 M	6.6 M	100 K	12 M
Working-set Size (pages *)	40 ～ 60	40 ～ 60	20	40 ～ 60
I/O Operation Frequency (/100 K steps)	1.0	10.0	1.0	1.0
Page Fault Rate (/100 K steps)	0.5	0.5	4.0	0.5
Think Time (sec)			20	20

* 1 page = 4 K Bytes

表5.2 インタラクティブなTSSトランザクションについてのシミュレーション比較（上段GRM，下段SRM）

Performance comparison of the two schedulers (GRM: upper row, SRM: lower row) with respect to the processing time of terminal-initiated interactive(I) transactions

CH1/CH2		4/4			6/2
I/E	w	36/4	32/ 8	28/12	36/4
Processing Time, Mean (sec)	60	0.47 0.55	0.51 0.83	0.48 1.98	0.43 0.62
	40	0.42 0.70	0.63 1.35	0.61 2.48	0.42 0.77
Processing Time, Standard Deviation (sec)	60	0.30 0.31	0.41 0.64	0.34 1.35	0.27 0.37
	40	0.20 0.60	0.59 1.07	0.64 2.17	0.21 0.52
Processing Time, 90% Accumulation (sec)	60	1.00 1.25	1.00 2.00	1.00 4.00	0.75 1.25
	40	0.75 1.50	1.25 3.00	1.25 5.50	0.75 1.75
Number of Completion (/300 sec)	60	523 524	466 463	413 379	528 525
	40	532 519	462 446	408 374	530 519

表 5.3　エグゼキュティブなTSSトランザクションについてのシミュレーション比較（上段GRM，下段SRM）

Performance comparison of the two schedulers (GRM: upper row, SRM: lower row) with respect to the processing time of terminal-initiated executive (E) transactions

CH1/CH2		4/4			6/2
I/E		36/4	32/8	28/12	36/4
Processing Time, Mean (sec)	60	94.3 57.8	94.6 86.7	121.7 90.9	60.9 62.4
	40	86.4 39.7	87.7 63.1	112.2 74.9	60.4 37.9
Processing Time, Standard Deviation (sec)	60	30.6 34.2	30.0 35.3	32.2 48.8	25.6 37.0
	40	34.4 29.7	35.2 34.0	32.6 51.9	27.8 28.6
Processing Time, 90% Accumulation (sec)	60	140.0 110.0	150.0 130.0	160.0 150.0	100.0 130.0
	40	130.0 80.0	160.0 100.0	160.0 150.0	80.0 80.0
Number of Completion (/300 sec)	60	8+2.1 13+1.6	17+3.6 20+2.4	21+4.5 27+4.6	12+1.5 14+0.9
	40	10+1.5 17+2.6	18+4.1 25+3.3	22+4.5 29+5.4	12+2.4 20+0.7

表 5.4　バッチ・トランザクションについてのシミュレーション比較
（上段GRM，下段SRM）

Performance comparison of the two schedulers (GRM: upper row, SRM: lower row) with respect to the processing time of batch transactions

CH1/CH2 / I/E / w		4/4			6/2
		36/4	32/8	28/12	36/4
Processing Time, Mean (sec)	60	79.9 100.9	122.7 137.6	140.9 214.4	90.7 109.6
	40	68.3 90.0	109.6 177.5	158.4 —	76.5 102.2
Processing Time, Standard Deviation (sec)	60	28.3 52.0	39.8 75.9	27.0 12.9	40.2 57.1
	40	31.7 55.8	44.6 49.8	50.8 —	52.8 56.4
Processing Time, 90% Accumulation (sec)	60	120.0 190.0	170.0 270.0	170.0 230.0	150.0 190.0
	40	100.0 140.0	200.0 260.0	200.0 —	190.0 210.0
Number of Completion (/300sec)	60	25+4.4 19+4.5	14+5.2 11+4.6	11+4.3 2+5.0	20+5.3 15+5.0
	40	30+3.2 18+4.0	17+4.0 9+3.2	12+3.8 0+5.9	25+4.8 14+4.6
NWL, Standard Deviation	60	19.7 19.6	21.6 24.4	19.2 26.4	18.3 23.1
	40	18.6 22.8	23.9 30.4	22.8 29.1	17.0 26.1

表 5.5　処理能力，オーバヘッドについてのシミュレーション比較（上段GRM，下段SRM）

Performance comparison of the two schedulers (GRM: upper row, SRM: lower row) with respect to throughput and overhead

CH1/CH2		4/4			6/2
I/E		36/4	32/8	28/12	36/4
W					
Throughput (su/300 sec)	60	582,006 561,972	579,429 555,117	586,136 551,134	572,415 520,586
	40	650,774 609,289	623,433 585,321	605,879 576,812	643,425 578,694
CPU Utilization, in the parenthesis : by batch (%)	60	73.9(44.8) 75.3(36.9)	77.6(27.6) 77.2(24.1)	81.1(23.1) 81.0(10.7)	71.1(34.8) 66.5(28.1)
	40	81.9(49.8) 82.5(34.0)	82.9(30.1) 83.6(19.7)	84.6(24.0) 85.3(9.0)	79.9(41.9) 76.5(25.9)
Total CPU Overhead, in the parenthesis : by GRM/SRM (%)	60	16.9(0.21) 16.5(0.17)	15.7(0.22) 15.5(0.18)	14.9(0.23) 13.7(0.18)	16.8(0.21) 16.4(0.17)
	40	17.0(0.21) 16.3(0.16)	15.8(0.22) 15.1(0.18)	14.7(0.24) 13.6(0.18)	17.0(0.21) 16.1(0.16)
Channel #1 Utilization, by user I/O (%)	60	24.9 19.1	19.1 17.3	17.1 11.5	33.8 27.4
	40	28.5 18.8	22.6 15.4	17.9 12.4	37.6 28.9
Channel #2 Utilization, by user I/O (%)	60	23.9 20.3	21.9 16.4	18.8 12.2	18.2 15.0
	40	29.0 22.8	23.6 14.4	18.5 12.1	21.0 14.3
Channel #3 Utilization, by paging (%)	60	26.5 25.8	25.8 26.2	26.3 24.6	25.1 25.8
	40	22.9 22.4	22.9 22.6	22.4 21.0	22.9 22.3

（ディスク）利用が不均衡な場合（CH1/CH2＝6/2）に顕著である。このとき，CPUバウンドのE型TSSトランザクションがSRMにおいてGRMより多く実行されているにもかかわらず，CPU utilizationはGRMがSRMより大きい点は興味深い。これはディスク群＃1（チャネル＃1）の割当てにおけるGRMの資源利用率制御効果を示している。

(c) オーバヘッド

表5.5には，オーバヘッドをも示す。Total CPU overheadは，GRM/SRM実行に加えスワッピング，ページ・フォールト等によるオーバヘッドである。この中で，GRM/SRM実行のみによるオーバヘッドについては別に示した。GRMにおけるオーバヘッドは，SRMにおけるそれと比較して，全体で1％程度，GRM/SRM自体については0.04～0.06％の増加にとどまる。なお，ページング，スワッピングのページ転送のオーバヘッドは，ページング・ドラムが結合されているチャネル＃3のutilizationから明らかなように，両方式ともほぼ同程度である。

5.4.3 漸近モデルによる評価

第2章で，優先制御のもとでのシステム性能を解析する漸近モデル（Asymptotic Model）につき述べた。本節では，漸近モデルを用いてGRMの制御方策を検討し，またGRMのもとでの性能の予測法につき述べる。ただし，漸近モデルは，固定優先度系すなわち優先度が時間的に変動しない系に関するものである。GRMは周期的に優先度が変更される動的優先度系であり，その性能解析は固定優先度系の解析結果の単純な組み合わせにより行なうことはできない。これは優先度の変化が過渡的制御効果（transient effect）を生ずるためである。しかし，過渡的制御効果の解析は極めて困難であり，その手法も知られていない。したがって本論文では，この影響は無視し，GRMの制御効果について第2章の議論より得られたいくつかの知見を示すにとどめる。

(1) GRMの制御方策

GRMにおいては式（5.1）と式（5.4）で示される2つの制御目標があり，それぞれ(1)資源利用率制御方策，(2)サービス分配制御方策，がこれを達成するためのものである。まず前者について考察する。

いまプロセッサjの利用率が著しく低い（$u_j \ll \rho_j$）とする。プロセッサjのヘビイ・ユーザ，すなわちjの使用特性値v_{ij}が大なるトランザクションをiとする。いま，v_{ij}と，iによるjの使用時間d_{ij}，待ちなしの平均応答時間*R_{i0}との間には次の関係がある。

$$d_{ij} = v_{ij} R_{i0} \qquad (5.15)$$

（なお，第2章の議論では同一クラス内の個々のトランザクションは識別されていない。したがって以後，「トランザクション・クラス」は唯一のトランザクションのみから構成され$N_i \equiv 1$とする。）5.3で述べた資源利用率制御方策により，jのpredecessorであるビジーな資源の割当てに

---

* v_{ij}の算出においては思考時間は考慮されない。

関し，iの優先度は上がる。ビジーな資源は近似的にボトルネック資源とみなせるので，第2章の議論より，この結果iの平均応答時間T_iは減少しR_{io}に近づく。すなわち，iによるjの利用率$u_j^{(i)}$は式（2.3）より，

$$u_j^{(i)} = d_{ij} / (T_i + z_i) \qquad (5.16)$$

で与えられるが，これは$v_{ij} / (1 + z_i / R_{io})$に近づく。既に述べたように制御の対象となるトランザクションは中ないし長大なものであり，多くの場合R_{io}は思考時間z_iと同程度ないしそれ以上のオーダである。したがってプロセッサjの利用率u_jに対する顕著な寄与が期待できる。以上の議論は，制御目標式（5.1）達成のためのGRMの制御方策の有効性を示している。

次に後者すなわちサービス分配制御につき考察する。トランザクションiが実行完了までに必要とする資源サービス量を$\bar{A}_i$とする。iの資源サービス率r_i，平均応答時間T_iに関して次式が成り立つ。

$$T_i = \bar{A}_i / r_i \qquad (5.17)$$

いま，iに対するサービス分配量が過少であり，$NWL_i \gg \overline{NWL}$と仮定する。このとき，5.3節で述べたサービス分配制御方策により，ボトルネック資源の割当てに関しiの優先度は上がる。この結果，T_iが減少することは第2章の議論より明らかである。ここで式（5.3）においてNWL_iがr_iの単調減少関数であることを考えると，式（5.3）と式（5.17）よりNWL_iの減少が期待できる。

この議論は$NWL_i \ll \overline{NWL}$の場合についても同様に成り立つ。以上より，GRMのサービス分配制御方策は制御目標式（5.4）達成のために有効である。

(2) GRMのもとでの性能の予測

GRMの制御下での，トランザクションの平均応答時間の予測法につき述べる。優先度は相矛盾する可能性のある2つの制御目標から定められるため，平均応答時間の正確な予測は困難である。本論文では応答性が重視される場合について考察する。いいかえると，制御目標として式（5.4）を式（5.1）より優先し，次式が成立すると仮定する。

$$NWL_i \cong \overline{NWL} \qquad (i = 1, 2, \cdots) \qquad (5.18)$$

トランザクションに対して設定したサービス目標関数によっては，式（5.18）の仮定のもとでも平均応答時間予測は困難な場合がある。ここでは，次の2つのタイプのサービス目標関数を考える。

- タイプ1：固定優先度を与えるもの。
- タイプ2：資源サービス率r_iの固定比率を与えるもの。

タイプ1とタイプ2の例を図5.10に示す。図5.10でg_Aとg_BはそれぞれトランザクションA，Bに対応するサービス目標関数である。タイプ1については，次の関係がなりたつ。いま，

$$r_i \leq \bar{A}_i / R_{io} \qquad (i = A, B) \qquad (5.19)$$

だから，図5.10において，

$$WL_1 \leq NWL_B \leq WL_2 \leq NWL_A \leq WL_3 \qquad (5.20)$$

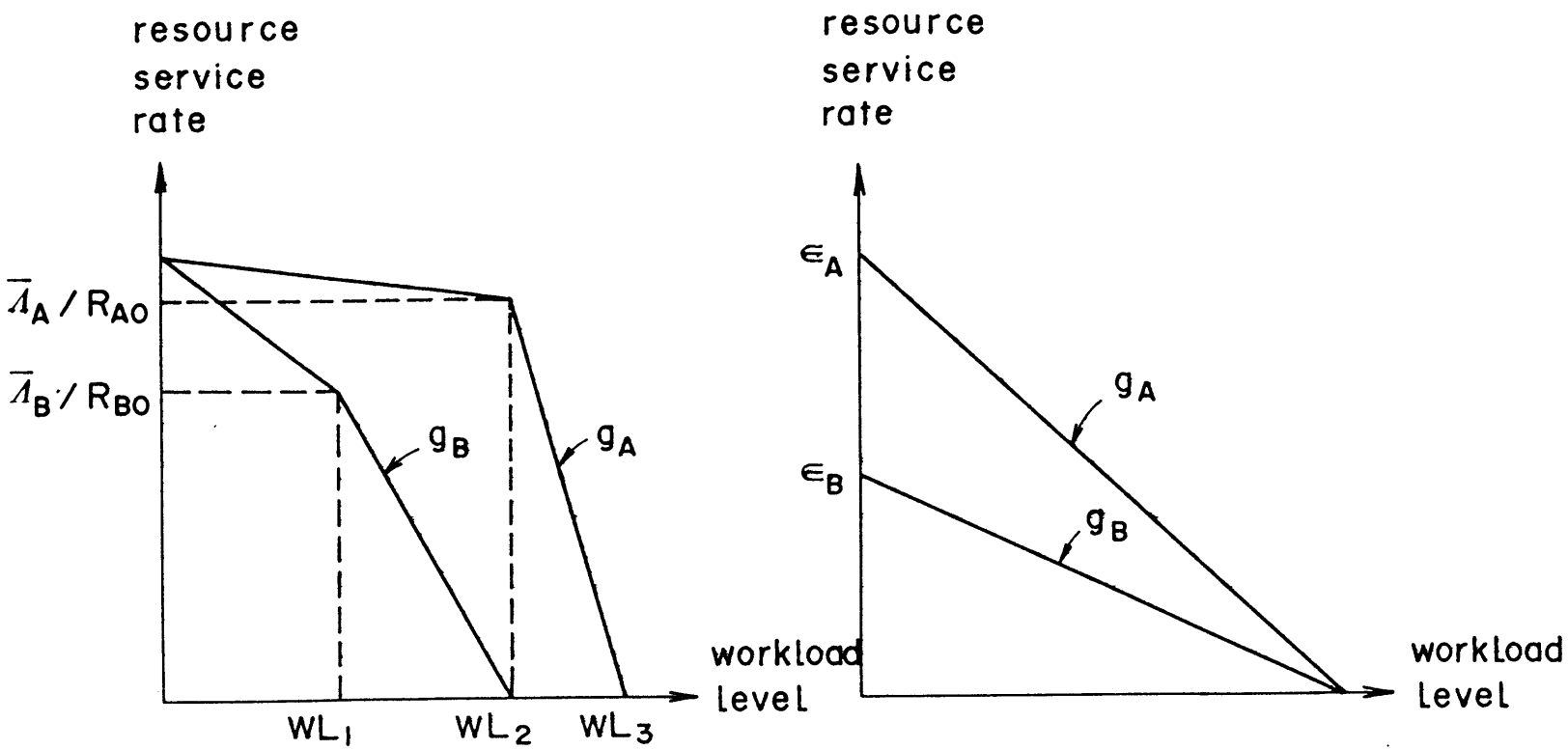


図5.10　性能予測の可能なサービス目標関数の設定例

Examples of the service objective functions which enable performance prediction

である。したがって，いかなる資源についても，Aの割当て優先度は常にBのそれより大きい。一方，タイプ2については，$\overline{NWL}$ によらず次式が成り立つ（図5.10参照）。

$$r_A / r_B = \varepsilon_A / \varepsilon_B \qquad (5.21)$$

実際には，ユーザのサービス分配に関する要求は，上記2つのタイプでほとんどの場合みたすことができる。タイプ1を指定した場合の平均応答時間の予測法は，既に2.2節で述べた。したがって，以下タイプ2を指定した場合につき述べる。

次の関係を与えるサービス目標関数を指定したとする。

$$r_1 : r_2 : r_3 : \cdots = \varepsilon_1 : \varepsilon_2 : \varepsilon_3 : \cdots \qquad (5.22)$$

式（5.17）と式（5.22）より，次式をうる。

$$\frac{\bar{\Lambda}_1}{T_1 \varepsilon_1} = \frac{\bar{\Lambda}_2}{T_2 \varepsilon_2} = \frac{\bar{\Lambda}_3}{T_3 \varepsilon_3} = \cdots \qquad (5.23)$$

諸資源の利用率は，式（2.12）よりえられる。したがって，

$$\begin{cases} \sum_i N_i d_{ij} / (R_{io} + z_i) < 1 \quad (j = 1, 2, \cdots) \\ \text{かつ} \\ \sum_i N_i (R_{io} / s_i) / (R_{io} + z_i) < 1 \end{cases} \qquad (5.24)$$

ならば，

$$T_i = R_{io} \qquad (i = 1, 2, \cdots) \qquad (5.25)$$

である。すなわち，ボトルネック資源がなく，待ちが無視できる場合には，式（5.22）は一般には成立しない。一方，式（5.24）が成立しないとき，いずれかのプロセッサ，ないし実メモリがボトルネックとなる。いまプロセッサ x がネックと仮定すると，T_i は式（5.23）と次式を連立することによりえられる。

$$1 = \sum_i N_i d_{ix} / (T_i + z_i) \qquad (5.26)$$

一方，ボトルネック資源が実メモリのみのとき，式（5.23）と，

$$1 = \sum_i N_i (R_{io} / s_i) / (T_i + z_i) \qquad (5.27)$$

を連立することにより解をえることができる。

5.5 結　言

多重プログラミング・システムの処理能力，応答性の均衡的向上を目的として，「一般資源管理方式GRM（General Resources Manager）」を新しく提案し，実験を行なった。GRMは，実メモリ，CPU，入出力装置などオペレーティング・システムで優先度を操作できる割当て待行列を有する諸資源につき，その資源割当て優先度を周期的に調節する体系であり，既に日立のオペレーティング・システムVOS3において実用に供されている。その特徴，評価結果などを以下にまとめる。

(1) GRMは，諸資源の利用率や各トランザクション（＝TSSコマンドないしバッチ・ジョブ）への資源供給量を周期的に測定し，この測定値と予め設定した目標値との差を減ずるよう制御するフ

ィードバック系である。制御は，利用率の高いボトルネック資源の割当て優先度変更により行なう。ＧＲＭの制御方策を提示するとともに，実メモリ，ＣＰＵ，入出力装置の割当てに関する優先度の決定法を具体的に示した。実メモリ割当てのみを制御手段とするＩＢＭ社のＳＲＭ[L3]はＧＲＭに含まれる。また，従来のタイム・スライシング制御[K9]，ダイナミック・ディスパッチング制御[R5]は，いずれもＧＲＭに概念的に包含される。

(2) ＧＲＭの特徴は，資源割当ての依存関係に着目し，従来資源ごとに独立していた諸資源割当て優先度の決定法を相互に関連づけた点にある。ここで依存関係とは，トランザクションが或る資源を割り当てられない限り別の資源に対する割当て要求を発生できないという関係を意味する。例えば実メモリとＣＰＵはこの関係をみたす。ＧＲＭは，各種の資源割当て優先度を，他資源の利用状況をも考慮して操作する。

(3) ＧＲＭの制御効果について検討を行なうため，詳細なシミュレーション・モデルを開発し，評価を実行した。資源ごとに管理系が独立している従来方式と比較したところ，明らかな制御能力の向上がみられた。例えば，応答時間の平均値，標準偏差においてそれぞれ１５～７６％，３～７５％の短縮，処理能力において４～１１％の向上などが観察された。また，制御のためのＣＰＵオーバヘッドは従来方式より0.04～0.06％の増加にとどまった。

(4) ＧＲＭをオペレーティング・システムＶＯＳ３の上で実現し，実測評価を行なった。資源ごとに管理系が独立している従来方式と比較して，ＧＲＭの制御効果を実測により確認した。たとえばインタラクティブ型の短小なＴＳＳトランザクションの応答性は，従来方式のもとではエグゼキュティブ型の長大なＴＳＳトランザクションの影響で低下する場合がある。しかしＧＲＭのもとでは応答性は保証され，応答時間の平均値，標準偏差においてそれぞれ７２％，９０％の短縮効果が観察された。

ＧＲＭの制御効果は，第２章の議論からある程度理論的解析が可能であり，本章では２種類の制御パラメータ設定例のもとでの性能予測法につき述べた。しかし，この議論は優先度変更にともなう過渡的影響は無視しており，今後なお理論の精緻化が必要である。特に過大な制御オーバヘッドによる性能低下を防止するためには，制御オーバヘッド自体を理論の中に含める必要がある。制御系としての安定性に関する議論は今後の課題である。

第 6 章　結　　論

6. 結　　論

現在ほとんどの商用大型計算機システムは仮想メモリ方式の多重プログラミング・システムであるといっても過言ではない。したがってその資源管理に関しては多くの理論的検討がなされてきた。しかし，それらの理論的成果は余りにも対象を単純化したり，実環境では未知の状態量を用いたりしているため，現実のシステムに適用されたものは少ない。この結果，現在稼動中のオペレーティング・システムにおける資源管理方式には，理論的根拠の薄弱な制御や，システム全体の性能と関りなく局所的目標を追求する制御などが散見される。すなわち，未解決の問題は実用面ではなお数多く残されている。

本論文は，仮想メモリ方式の多重プログラミング・システムの性能向上を目的として，その資源管理方式を理論，実用の両面から検討した研究成果をまとめたものである。システム性能に関する問題は，資源を効率的に用いるようジョブの資源使用法を改善する設計の問題[F2)M2)]と，ジョブ自体には手を加えずジョブに対する資源割当て法を改善する運用の問題とに分けられる。本論文では後者を研究の対象とした。

資源管理方式の提案に先立ち，まずシステムを数学的にモデル化し，このモデルを用いて各種の資源割当てアルゴリズムがシステム性能に与える効果を解析した。仮想メモリ・システムにおける性能上の最大の問題点のひとつは，実メモリと二次メモリとの間の情報転送処理すなわちページング処理のオーバヘッドが不必要に増大し，システム性能が低下するページング・ネック現象である。これを防止するため，ページング処理の頻度を調節し，ＣＰＵなど諸資源のページングによる遊休状態発生を防ぐ実メモリ管理方式を提案した。また，ページング処理の効率を上げる二次メモリ管理方式を提案した。さらにページング・ネック防止以外の残された問題として，諸資源の割当て優先度を動的に調節し，処理能力と応答性の均衡的向上を達成する資源管理方式を提案した。以下，得られた結果を各章ごとにまとめる。

第１章の序論に続き，第２章では，資源管理の効果を解析するため，各種の資源割当て優先度系のもとでのシステム性能を与える理論的な手法を提案した。本手法の特徴は，トランザクション（＝ＴＳＳコマンドないしバッチ・ジョブ）の平均応答時間や諸資源（実メモリ，ＣＰＵ，入出力装置など）の利用率が，資源サービス時間平均値の或る関数に漸近する点に着目し，厳密解を漸近解で近似した点にある。したがって本手法を「漸近モデル（Asymptotic Model）」とよぶ。漸近モデルはシステム性能の構造的性質を表わすモデルであり，これを用いて(1)利用率の高い資源の割当て優先度をいかに与えるかによりシステム性能が左右されること，(2)不必要にページング処理が多発するページング・ネックのとき，システム性能はページング処理のオーバヘッドにより低下し，諸資源の割当て優先度調節による性能向上は困難となること，などが明らかになった。これらは従来より経験的には認められていたが，理論的に示されたことに意味がある。漸近モデルの精度を，実測およびシミュレーションにより検証した。システム性能の構造が変化する点において誤差は増大するが，システム

の大局的挙動が簡便に解析でき，特に低優先度トランザクションの挙動の分析に有効なことが確認された。

第３章では，ページング・ネック防止を目的として実メモリ管理について論じ，「ワーキングセット最適化方式ＯＷＥ（Optimum Working － set Estimator）」とよぶ方式を提案した。まず，実メモリ割当て量の評価基準として従来より用いられるSpace Time Productとシステム全体の性能との関係を論じ，その最小化が処理能力向上に寄与することを示した。ＯＷＥは，Space Time Productの推定値を求め，ワーキングセットのウインド・サイズの調節により，これを最小化する方式である。ＯＷＥの特徴は，プログラム実行中にそのページ参照特性を学習して最適値を求める点にあり，現実の多重プログラミングの環境下で用いることができる。ＯＷＥをオペレーティング・システムＶＯＳ３において実現し，大型計算機ＨＩＴＡＣ　Ｍ－１８０を用いて実測を行なった。ＯＷＥのもとでのSpace Time Productの実測値は，理論的最小値の１.０１～１.１０倍であり，Space Time Product最小化達成能力を確認した。また，処理能力については，従来方式に比べ１～１９％の向上が観察された。

第４章では，ページング・ネック防止を目的として二次メモリ管理について論じ，「集中方式（Concentrating Technique）」とよぶ方式を提案した。ページ入出力実行時間の短縮により，システムのページング処理のオーバヘッドは削減できる。ページ入出力実行時間は，アドレス空間上のページを二次メモリ上のスロットに蓄積する二次メモリ管理方式に依存する。効率差異は二次メモリ媒体として磁気ディスクを用いたとき特に著しいので，本論文ではこれを仮定した。まず，代表的な従来方式である浮動方式と固定方式との効率を理論的に比較した。前者はページ出力効率，後者はページ入力効率において優れ，総合的なページ入出力効率の優劣はプログラムのページ参照特性に依存することが明らかになった。なおここで，ページを蓄積するスロットを選択する際の自由度に着目すると，浮動方式，固定方式はそれぞれ自由度最大，最小である。一般に自由度を増すにつれて，ページ出力効率は向上し，ページ入力効率は低下する。したがって，総合的なページ入出力効率を最大化する最適自由度を求めることができる。提案した集中方式の特徴は，実用的条件のもとでこの最適自由度を実現する点にある。集中方式をオペレーティング・システムＶＯＳ３において実現し，大型計算機ＨＩＴＡＣ　Ｍ－１８０を用いて実測を行なったところ，従来方式に比べページ入出力実行時間の平均値，標準偏差においてそれぞれ３８％，３５％の短縮が観察された。

第５章では，システム性能向上のための残された問題として，ページング・ネック防止以外の問題を扱った。すなわち，処理能力と応答性の均衡的向上を目標として諸資源割当て優先度を周期的に調節する方式を提案した。本方式を「一般資源管理方式ＧＲＭ（General Resources Manager）」とよぶ。ＧＲＭは，諸資源の利用率や各トランザクション（＝ＴＳＳコマンドないしバッチ・ジョブ）への資源供給量を測定し，これを目標に近づけるよう制御を行なうフィードバック・スケジューラである。第２章の結論にもとづき，制御は利用率が高くボトルネックとなっている資源の割当て優先度の変更により実行する。ＧＲＭの特徴は，オペレーティング・システムで割当て優先度を操作可能な

諸資源につき，従来資源ごとに独立していた各管理系を，資源割当ての依存関係に着目して相互に関連づけた点にある。ＧＲＭの有効性は第２章の議論により裏づけられるが，これを検証するためシミュレーションを実行し，性能の向上効果を確認した。また，ＧＲＭをオペレーティング・システムＶＯＳ３において実現し，大型計算機ＨＩＴＡＣ　Ｍ－１８０を用いて実測評価を行なった。従来の資源ごとに独立した管理方式に比べ，明らかな制御能力の向上がみられた。例えば端末応答時間の平均値，標準偏差が，或る場合にはそれぞれ従来方式の０．２８倍，０．１０倍に短縮されるなどの効果が観察された。

以上が本論文のまとめである。本論文においては，できる限り理論的根拠を保ちつつ，実用に供する資源管理方式の検討を心がけた。提案した３つの資源管理方式は，いずれも日立の大型オペレーティング・システムＶＯＳ３において実稼動中ないし実用化が予定されている。もちろん現実のシステムは非常に複雑であり，本論文においても解析を進める上で仮定を設けることは不可避であったが，本論文により得られた知見が実用的な価値を有することを信ずる。

しかし，現実のオペレーティング・システムの動作環境において実行される資源管理を理論化，体系化し，大局的見地からの性能向上をはかるという目的にてらすと，なお今後解決すべき問題は少なくない。これらの中で本研究の内容と関連のあるものをいくつか述べておく。

今後，大型オペレーティング・システムにおける資源管理方式は一層複雑化，大規模化すると予想される。しかしその制御効果に関する理論的検討は十分なされてきたとは言えない。第２章では資源割当てに関し固定優先度のある系の性能解析手法を述べたが，動的優先度のある系における性能の解析は残された問題である。また，制御オーバヘッドを含めた制御効果解析モデルの検討も今後の重要な課題である。第５章で提案した資源管理方式においても，制御オーバヘッドによる負の効果は付随的に考慮されているに過ぎない。制御オーバヘッドによる性能低下は，過制御状態すなわちシステムの状態が不安定な場合に生ずる。計算機システムの制御における安定性の議論[W2)W3)]は実用面から不可欠のものであり，今後理論的成果が期待される分野である。

仮想メモリ・システムの実メモリ管理に関する研究は既に数多くなされているが，それらはほとんど全てプログラムの局所参照性にもとづくものである。第３章の議論もまた例外ではない。しかし，局所参照性はプログラムのプロシジャ部分については成立するが，データ部分については必ずしも成立しない。今後，大規模なデータ・ベースの出現にともない，データ部分に関する実メモリ割当ての問題は重大化すると予想される。大須賀[05)]はこの問題を検討し，将来の計算機システム構成の変革の可能性を示唆した。データの入出力効率を上げるためには，オペレーティング・システムの実メモリ管理方式[D9)]のみならず，データ・ベースそのものの構造の改善[N3)]や，二次メモリ媒体の高速化，並列化が必要である。一般に仮想メモリ・システムにおいて，二次メモリと実メモリとの間の情報転送の効率化に関する研究は従来不足している。第４章の議論はこの分野に関するひとつの試みであったが，なお各方面からの検討が望まれる。特に，アクセス速度とコストの異なる複数のメモリ媒体の複合体としてのメモリ階層システムの効率的な設計，運用についての研究は，今後最も重要なもののひとつである。

謝　　辞

本研究をまとめるにあたり，東京大学和田英一教授に御指導を賜わった。また本研究の遂行を通じ，東京大学大須賀節雄教授に終始御指導，御助言をいただいた。ここに衷心より感謝の意を表する。さらに貴重な御助言をいただいた東京大学穂坂衛名誉教授，筑波大学益田隆司助教授，電気通信大学亀田寿夫助教授，米国Stanford 大学 G. Wiederhold助教授，Purdue 大学 P.J.Denning 教授に心から御礼を申し上げる。

本研究の機会を与えて下さった日立製作所システム開発研究所三浦武雄所長，川崎淳副所長，著者の上長として終始種々の御指導をいただいた三巻達夫主管研究員，石原孝一郎部長，大町一彦主任研究員，提案した方式をオペレーティング・システムＶＯＳ３に適用する機会を与えて下さった同ソフトウェア工場服部陽一部長，大西勛主任技師，野口健一郎主任技師，の諸氏に深く感謝の意を表する。

また，提案方式の適用に際してご協力をいただいた同ソフトウェア工場緒方慎八氏，旭寛治氏，原田晃氏，篠崎俊春氏，同小田原工場馬場正俊氏，さらに提案方式の実現および実測評価に際して多大なご協力をいただいた同システム開発研究所池田智明氏，山本彰氏に深く感謝する。

参　考　文　献

A１）　Anderson,H.A., Reiser,M. and Galati, G.L. Tuning a Virtual Storage System, *IBM Syst. J.*, **14**, 3 (1975), 246−263.

B１）　Badel, M., Gelenbe, E., Leroudier, J. and Potier, D. Adaptive Optimization of a Time − Sharing System's Performance, *Proc. IEEE*, **63**, 6 (1975), 958−965.

B２）　Bard,Y. The Modeling of Some Scheduling Strategies for an Interactive Computer System, in *Computer Performance*, K.M. Chandy and M.Reiser (Eds.), Elsevier North − Holland Inc., New York (1977), 113−138.

B３）　Baskett,F., Chandy, K.M., Muntz, R.R. and Palacious, F.G .Open, closed and mixed networks of queues with different classes of customers, *J.ACM*, **22**, 2 (1975), 248−260.

B４）　Belady, L.A. A study of replacement algorithms for a virtual − storage computer , *IBM Syst. J.*, **5**, 2 (1966), 78−101.

B５）　Belady, L.A. and Kuehner, C.J. Dynamic Space−Sharing in Computer Systems, *Commun. ACM*, **12**, 5 (1969), 282−288

B６）　Bernstein, A.J. and Sharp, J.C. A Policy Driven Scheduler for a

Time-Sharing System, *Commun. ACM*, **14**, 2 (1971), 74-78.

B7) Blevins, P.R. and Ramamoorthy, C.V. Aspects of a Dynamically Adaptive Operating System, *IEEE Trans. Comput.*, **C-25**, 7 (1976), 713-725.

B8) Brinch Hansen, P. The Nucleus of a Multiprogramming System, *Commun. ACM*, **13**, 4 (1970), 238-251.

B9) Buzen, J.P. Computational Algorithms for Closed Queueing Networks with Exponential Servers, *Commun. ACM*, **16**, 9 (1973), 527-531.

B10) Buzen, J.P. A Queueing Network Model of MVS, *Comp. Surv.*, **10**, 3 (1978), 319-331.

C1) Chang, W. Preemptive priority queues, *Oper. Res.*, **13** (1965), 820-827.

C2) Chiu, W. W. and Chow, W.-M. A performance of MVS, *IBM Syst.J.*, **17**, 4 (1978), 444-462.

C3) Chu, W.W. and Opderbeck, H. The page fault frequency replacement algorithms, *Proc. FJCC* (1972), 597-609.

C4) Coffman, E.G. and Kleinrock, L. Computer Scheduling Methods and Their Computer Measures, *Proc. SJCC* (1968), 11-21.

C5) Coffman, E.G. and Varian, L.C. Further experimental data on the behavior of programs in a paging environment, *Commun. ACM*, **11**, 7 (1968), 471-474.

C6) Coffman, E.G. and Ryan, T.A. A Study of Storage Partitioning Using a Mathematical Model of Locality, *Commun. ACM*, **15**, 3 (1972), 185-190.

C7) Corbató, F.J. and Vyssotsky, V.A. Introduction and overview of the MULTICS system, *Proc. FJCC* (1965), 185-196.

C8) Courtois, P.J. Decomposability, Instabilities and Saturation in Multiprogramming systems, *Commun. ACM*, **18**, 7 (1975) 371-377.

D1) Davis, R.H. Waiting time distribution of a multi-server, priority queueing system, *Oper. Res.*, **14** (1966), 133-136.

D2) Denning, P.J. Effects of scheduling on file memory operations, *Proc. SJCC* (1967), 9-21,

D3) Denning, P.J. The Working - Set Model for Program Behavior, *Commun. ACM*, **11**, 5 (1968), 323-333.

D 4) Denning,P.J. Thrashing : Its Causes and Prevention, *Proc. FJCC* (1968), 915—922.

D 5) Denning,P.J. and Schwartz, S.C. Properties of the Working — Set Model, *Commun. ACM*, **15**, 3 (1972) 191—198.

D 6) Denning,P.J. and Graham, G.S. Multiprogrammed Memory Management, *Proc. IEEE*, **63**, 6 (1975), 924—939.

D 7) Denning,P.J. and Kahn, K.C. An L=S criterion for optimal multiprogramming, *Proc. ACM SIGMETRICS Int. Symp. Computer Performance Modeling, Measurement and Evaluation* (1976), 219—229.

D 8) Denning,P.J., Kahn, K.C., Leroudier,J., Potier, D. and Suri,R. Optimal Multiprogramming, *Acta Inf.*, **7** (1976), 197—216.

D 9) Denning, P.J. Generalized Working Sets for Segment Reference Strings, *Commun. ACM*, **21**, 9 (1978), 750—759.

D10) Denning, P.J. and Buzen, J.P. The Operational Analysis of Queueing Network Models, *Comp. Surv.*, **10**, 3 (1978), 225—261.

D11) Dijkstra, E. W. The structure of THE multiprogramming system, *Commun. ACM*, **11**, 5 (1968), 341—346.

D12) Doherty, W.J. Scheduling TSS/360 for Responsiveness, *Proc. FJCC* (1970), 97—111.

F 1) フェラー, W. 確率論とその応用, 河田龍夫監訳, 紀伊国屋書店 (1969).

F 2) Ferrari,D. Improving Locality by Critical Working Sets, *Commun. ACM*, **17**, 11 (1974), 614—620.

F 3) Fin,T.—H. and Kameda, H. An Extension of a Model for Memory Partitioning in Multiprogrammed Virtual Memory Computer Systems, *JIP*, **2**, 2 (1979), 89—96.

F 4) Franklin, M. A., Graham, G.S. and Gupta, R.K. Anomalies with Variable Partition Paging Algorithms, *Commun. ACM*, **21**, 3 (1978), 232—236.

G 1) Gordon,W.J. and Newell, G.F. Closed queueing networks with exponential servers, *Oper. Res.*, **15** (1967), 254—265.

G 2) Graham, G.S. and Denning,P.J. On the relative controllability of memory policies, *Proc. Int. Symp. Computer Performance Modeling, Measurement and Evaluation* 1977, North Holland,Amsterdam (1977), 411—428.

G３） Gupta,R.K。and Franklin, M。A. Working Set and Page Fault Freqυency Paging Algorithms : A Performance Comparison, *IEEE Trans. Comput.*, **C−27**, 8 （1978）, 706−712。

H１） Hatfield, D.J. and Gerald,J. Program restructuring for virtual memory, *IBM Syst。J.*, **10**, 3 （1971）, 168−192。

H２） Hellerman,H. Some Principles of Time−Sharing Scheduler Strategies, *IBM Syst.J.*, **8**, 2 （1969）, 94−117。

H３） Henderson, G. and Rodriguez−Rosell,J。The optimal choice of window sizes for working set dispatching, *Proc。ACM SIGMETRICS Symp。Measurement and Evaluation* （1974）, 10−33。

H４） 日立製作所 H−8549−2ディスク制御装置，H−8589ディスク駆動装置，ハードウェア・マニュアル，8080−2−007（1974）。

H５） 日立製作所 VOS2システム・プログラマの手引，手引書，8080−3−002（1975）。

H６） 日立製作所 VOS3概説マニュアル，概説書，8090−3−001 （1975）。

H７） 日立製作所 VOS3システム・プログラマの手引，手引書，8090−3−002（1976）。

I１） IBM Introduction to Virtual Storage in System／370, *Student Text* （1972）。

J１） Jackson, J.R。Jobshop−like Queueing Systems, *Management Science*, **10**, 1 （1963）, 131−142。

K１） Kameda, H. The Analysis of Adaptive Workload Balancing Strategy in Computer System Resource Management, *Int'l.J. Comput. Inf. Sci.*, **4**, 4 （1975）, 295−306。

K２） Kameda, H。and Gotlieb, C.C. A Feedback−Coupled Resource Allocation Policy for MultiProgrammed Computer Systems, *Acta Inf.*, **8** （1977）, 341−357。

K３） 亀田 諸種のスケジューリング方式の下で多種類のジョブを扱う Finite Population Model の性質について，情報処理学会計算機システムの解析と制御研究会資料，5−1（1979）。

K４） 金田 タイムシェアリング・システムのシミュレーション，電子通信学会論文誌，**53−C**, 2 （1970）, 119−125。

K５） Kilburn,T., Edwards, D.B.G., Lanigan, M.J. and Sumner, F.H. One−level storage system, *IRE Trans.*, **EC−11**, 2 （1962）, 223−235。

K６） Kleinrock,L。and Finhelstein,R.P。Time dependent priority queue *Oper. Res.*, **15** （1967）, 104−116。

K 7) Kleinrock,L。Time shared system, a theoretical treatment, *J .ACM*, **14**,2 (1967),242−261。

K 8) Kleinrock,L. Certain analytic results for time−shared processors, *Proc .IFIP*68 (1968),838−845。

K 9) Kleinrock,L. A Continuum of Time−Sharing Scheduling Algorithms, *Proc .SJCC* (1970),453−458。

K10) Krzesinski,A。and Teunissen, P。A Multiclass Network Model of a Demand Paging Computer System, *Acta Inf.*, **9** (1978),331−343。

L 1) Lett, A.S. and Konigsford, W.L. TSS/360: A time−shared operating system, *Proc. FJCC* (1968), 15−28。

L 2) Leroudier, J. and Potier, D. Principles of optimality for multiprogramming, *Proc。ACM SIGMETRICS Int'l Symp. Computer Performance Modeling, Measurement and Evaluation* (1976), 211−218.

L 3) Linch, H.W。and Page,J.B。The OS/VS2 Release 2 System Resources Manager, *IBM Syst.J.*, **13**,4 (1974),274−291.

M 1) 益田，高橋，吉沢 ページング・マシンにおけるスワッピング・アルゴリズムの比較とプログラムの動作解析，情報処理，**13** ，2 (1972),81−88。

M 2) 益田，塩田 仮想メモリ・システム向きの最適プログラム構成と実験，情報処理，**15**，9 (1974),662−669。

M 3) Masuda,T. Analysis of Memory Management Strategies for MultiProgrammed Virtual Storage Systems, *JIP*, **1** ,1 (1978),14−24.

M 4) Mattson,R.L., Gecsei,J., Slutz,D.R。and Traiger.I.L. Evaluation techniques for storage hierarchies, *IBM Syst。J.*, **9**,2 (1970), 78−117.

M 5) Mealy,G.H., Witt, B.J。and Clark。W.A。The functional structure of OS/360 Part Ⅰ〜Ⅱ, *IBM Syst.J.*, **5**,1 (1966),3−51.

M 6) 本林，益田，勝枝，高橋 2次元番地付方式によるＨＩＴＡＣ５０２０ＴＳＳの特徴，情報処理，**9**，6 (1968),317−325.

M 7) Muntz,R.R。and Wong,J。Asymptotic properties of closed queueing network models, *Proc。8 −th Annu. Princeton Conf。Inf.Sci.Syst.* (1974), 348−353。

M 8) Muntz,R.R。Analytic Modeling of Interactive Systems, *Proc. IEEE*, **63**,6 (1975),946−953.

N１） Neuse,D。and Chandy, K.M. A Method for APproximate Analysis of General Queueing Networks, *Technical Report of University of Texas at Austine*, TR153(1980).

N２） Nielsen,N.R. The Simulation of Time Sharing Systems, *Commun.ACM*, **10**,7 (1967),397−412。

N３） 西垣，野木，宮本 クラスタ分析によるセグメント編成，情報処理，**17**，6(1976)，494−500。

N４） 西垣 計算機リソース・スケジューリングへのフィードバック概念の適用，慶応義塾大学第８１回情報科学研究会資料(１９７８)．

N５） 西垣 多重プログラミング・システムにおけるフィードバック概念にもとづく一般資源管理方式，情報処理，**19**，11 (1978)，1026−1033．

N６） Nishigaki,T., Ikeda,C., Ohmachi,K. and Noguchi,K. An Experiment on the General Resources Manager in Multiprogrammed Computer Systems, *JIP*,**1**,4 (1979), 187−192.

N７） 西垣，緒方 仮想メモリ・システムの二次記憶管理方式の比較解析，情報処理学会論文誌，**20**，4 (1979)，290−298。

N８） 西垣，緒方，大町，池田 仮想メモリ・システムの二次記憶管理の最適化，情報処理学会論文誌，**21**，5 (1980)，366−374。

N９） Nishigaki, T., Noguchi,K. and Ohmachi,K. An APproach to the GRM Performance Analysis by Asymptotic Approximation, *JIP*,**3**,2 (1980),59−67。

N10） 西垣，池田 仮想メモリ・システムのワーキングセット最適化に関する考察と実験，情報処理学会論文誌，**21**，4 (1980)，325−331。

N11） Nishigaki, T。ExPeriments on the Knee Criterion in a MultiProgrammed Computer System, *Stanford University Computer Science Report*, STAN−CS−81−849, CSL TR−205 (1981)。

N12） 西垣，山本 資源割当て優先度のある多重プログラミング・システムのボトルネック解析，情報処理学会論文誌，**23**，5(1982)．

N13） Noetzel, A.S。A Generalized Queueing Discipline for Product Form Network Solutions, *J.ACM*, **26**,4 (1979),779−793。

N14） 野口，元岡 ＴＳＳにおけるトラヒック処理の解析，信学資，ＥＣ６７−１５(１９６７)．

０１） 大町，本山，松岡，池田 計算機システムの性能評価技法"ＩＳＣＰ"の開発，日立評論，**61**，12 (1979)，65−68．

０２） Ohmachi,K., Nishigaki, T. and Takasaki, S。Analysis of PAWP／VMS: Paging Algorithm to Prevent Double Paging Anomaly in Virtual Machine

Systems, *JIP*, **4**, 2 (1981), 55−60.

03) 大西，秋田，堂免，金子 HITAC8700／8800 オペレーティング・システム (OS7)，情報処理，**14**，10 (1973)，769−777.

04) 大須賀 計算機システムの最適化制御，情報処理，**9**，2 (1968)，88−97.

05) 大須賀 プログラムおよびデータの特性指標とその仮想記憶システムの特性への影響，情報処理学会論文誌，**20**，3 (1979)，256−264.

06) Opderbeck, H. and Chu, W. W. Performance of Page Fault Frequency Replacement Algorithms in a multiprogramming environment, *Proc. IFIP* 74 (1974), 236−241.

P1) Prieve, B. G. Using Page Residency To Select the Working Set Parameter, *Commun. ACM*, **16**, 10 (1973), 619−620.

P2) Prieve, B.G. VMIN − An Optimal Variable-Space Page Replacement Algorithms, *Commun. ACM*, **19**, 5 (1976), 295−297.

R1) Reiser, M. and Kobayashi, H. Queueing Networks with Multiple Closed Chains : Theory and Computational Algorithms, *IBM J. Res. Develop.*, **19**, 3 (1975), 282−294.

R2) Rodriguez−Rosell, J. Experimental data on how program behavior affects the choice of scheduler parameters, *Proc. 3rd ACM Symp. Operating System Principles* (1971), 156−163.

R3) Rodriguez − Rosell, J. and DuPuy, J.−P. The Design, Implementation, and Evaluation of a Working Set Dispatcher, *Commun. ACM*, **16**, 4 (1973), 247−253.

R4) Ruschitzka, M. and Fabry, R.S. A Unifying Approach to Scheduling, *Commun, ACM*, **20**, 7 (1977), 469−477.

R5) Ryder, K.D. A Heuristic Approach to Task Dispatching, *IBM Syst.J.*, **9**, 3 (1970), 187−198.

S1) Saltzer, J.H. A Simple Linear Model of Demand Paging Performance, *Commun. ACM*, **17**, 4 (1974), 181−186.

S2) Scherr, A.L. An Analysis of Time Shared Computer Systems, Cambridge, Mass. MIT Press (1967).

S3) Scherr, A.L. Functional structure of IBM virtual storage operating systems Part II : OS／VS2−2 concepts and philosophies, *IBM Syst.J.*, **12**, 4 (1973), 382−400.

S4) Schrage, L.E. and Miller, L.W. The queue M／G／1 with the shortest

remaining processing time, *Oper. Res.*, **14** (1966), 670–684.

S5) Sekino, A. Performance Evaluation of MultiProgrammed Time-Shared Computer Systems, *MIT Project MAC Report*, TR-103, Cambridge, Mass. (1972).

S6) Sevcik, K.C. Priority scheduling disciplines in queueing network models of computer systems, *Proc. IFIP 77* (1977), 565–570.

S7) Sharp, J.C. and Roberts, J.N. An Adaptive Policy Driven Scheduler, *Proc. ACM SIGMETRICS Symp. Measurement and Evaluation* (1974), 199–208.

S8) Slutz, D.R. and Traiger, I.L. A Note on the Calculation of Average Working Set Size, *Commun. ACM*, **17**, 10 (1974), 563–565.

S9) Smith, A.J. A Modified Working Set Paging Algorithm, *IEEE Trans. Comput.*, **C-25**, 9 (1976), 907–914.

S10) Smith, A.J. Bibliography on paging and related topics, *Operating Syst. Review*, **12**, 4 (1978), 39–56.

S11) Spirn, J.R. and Denning, P.J. Experiments with program locality, *Proc. FJCC* (1972), 611–621.

S12) Stimuler, S. Some Criteria for Time-Sharing System Performance. *Commun. ACM*, **12**, 1 (1969), 47–53.

T1) 田中 循環待ち行列を用いた多重プログラミング・システムの解析, 電子通信学会論文誌, **53-C**, 2 (1970), 57–64.

T2) Teorey, T.J. and Pinkerton, T.B. A Comparative Analysis of Disk Scheduling Policies, *Commun. ACM*, **15**, 3 (1972), 177–184.

T3) Teorey, T.J. Properties of disk scheduling policies in multiprogrammed computer systems, *Proc. FJCC* (1972), 2–11.

T4) Turner, R. and Strecker, B. Use of the LRU Stack Depth Distribution for Simulation of Paging Behavior, *Commun. ACM*, **20**, 11 (1977), 795–798.

W1) Wheeler, T.F. OS/VS1 concepts and philosophies. *IBM Syst. J.*, **13**, 3 (1974), 213–229.

W2) Wilkes, M.V. Automatic Load Adjustment in Time-Sharing Systems, *Proc. ACM SIGOPS Workshop Syst. Performance Evaluation* (1971), 308–317.

W3) Wilkes, M.V. The dynamics of paging, *Comput. J.*, **16**, 1 (1973), 4–9.

付　　録

付録１ ： ＳＴＰルールと Knee ルールによる最適値

$X(\bar{w})$, $Sf(\bar{w})\bar{w}$ がともに１個の極小値をもつと仮定し，これを与える $\bar{w}$ をそれぞれ $\bar{w}_S$, $\bar{w}_K$ とする。以下，各関数は連続で微分可能とする。式(3.4)を微分して，

$$\frac{dX}{d\bar{w}} = 1 + \frac{d}{d\bar{w}}(Sf\bar{w}) \qquad (A\cdot 1)$$

を得る。与件より，

$$\frac{dX}{d\bar{w}}\Big|_{\bar{w}=w_S} = 0 \qquad (A\cdot 2)$$

$$\frac{d}{d\bar{w}}(Sf\bar{w})\Big|_{\bar{w}=w_K} = 0 \qquad (A\cdot 3)$$

である。したがって次式が成りたつ。

$$\frac{dX}{d\bar{w}}\Big|_{\bar{w}=w_K} = 1 > 0 \qquad (A\cdot 4)$$

$\bar{w}_S$ はＸの唯一の極小値を与える点であり，式（Ａ・４）よりＸは $\bar{w}_K$ において増加しているので，

$$\bar{w}_S \leq \bar{w}_K \qquad (A\cdot 5)$$

を得る。

なお，$\bar{w}$ の関数 $L/(L+S)$ の形状に関する知見は，次のようにこれを微分することにより得られる。

$$\frac{d}{d\bar{w}}\left(\frac{L}{L+S}\right) = \frac{S}{(L+S)^2}\left(\frac{dL}{d\bar{w}}\right) \qquad (A\cdot 6)$$

$$\frac{d^2}{d\bar{w}^2}\left(\frac{L}{L+S}\right) = -\frac{2S}{(L+S)^3}\left(\frac{dL}{d\bar{w}}\right)^2 + \frac{S}{(L+S)^2}\left(\frac{d^2L}{d\bar{w}^2}\right) \qquad (A\cdot 7)$$

L は $\bar{w}$ の広義の単調増加関数であるから，$L/(L+S)$ も $\bar{w}$ の広義の単調増加関数であり，また L が上に凸となるような $\bar{w}$ の範囲では $L/(L+S)$ も上に凸であることが式（Ａ・６），（Ａ・７）よりわかる。

付録２ ： x と y の関係

x 個のブロックを次々と蓄積するとき，各試行でディスク・ヘッドの位置は１～L の一様分布にしたがう。y は，x 回の試行で少なくとも１回ディスク・ヘッドが停止していた相異なるシリンダ数の期待値である。ここで，全くディスク・ヘッドが停止しなかった相異なるシリンダの数が m である確率 $T_m(x, L)$ は式(4.14)で与えられ，これから y を式(4.13)により算出できる。しかし実際の計算には，直接式(4.14)を用いず，x と m を変数とする次の漸化式を用いる方が簡便である。[F1)]

$$\begin{cases} T_m(x+1, L) = T_m(x, L)\cdot(L-m)/L + T_{m+1}(x, L)\cdot(m+1)/L \\ \qquad (\max(0, L-x) \leq m \leq L-2) \end{cases} \quad (A\cdot 8)$$

$$\begin{cases} T_m(x+1, L) = T_{m+1}(x, L)\cdot(m+1)/L \\ \qquad (m = L-x-1) \end{cases} \quad (A\cdot 9)$$

式(A・9)はディスク・ヘッド位置が全く重複しない場合にあたる。また$m = L-1$の場合は直接式(4.14)より,

$$T_{L-1}(x, L) = L^{1-x} \quad (A\cdot 10)$$

である。式(A・8),(A・9),(A・10)より, x, mを1から順次まして$T_m(x, L)$を求め, さらに式(4.13)よりyを求められる。

付録3 : ブロック数算定の近似について

第iブロックにnページ存在する確率を$\chi_i(n)$とする($n = 0, 1, 2, \cdots$)。当ブロックが存在するためには1ページ以上あればよいから, 真に求める値B_iは,

$$B_i \triangleq \sum_{n=1} \chi_i(n) = 1 - \chi_i(0) \quad (A\cdot 11)$$

で与えられる。しかし, 一般に$\chi_i(n)$は不明であり, 与えられているのは, ブロック・サイズの期待値E_iすなわち$\sum_{n=0} n\,\chi_i(n)$のみである。ここで,

・$E_i \geq 1$のとき, $\chi_i(0) = 0$と仮定すると$B_i = 1$

・$E_i < 1$のとき, $\chi_i(n) = 0$ ($n \geq 2$)と仮定すると$B_i = E_i$

となる。なお, 一般には$\chi_i(n) \geq 0$ ($n \geq 0$)なので, 上記の近似はB_iをやや過大評価することになる。(この議論は, E_iのみならず, ϕ_m, ψ_mについても同様である。)

付録4 : 式(4.30),(4.35)の証明

ζ_nの定義より各項は非負であり, その和$\sum_{n=0}^{\infty} \zeta_n$が0以上で$v$をこえないことは明らかである。式(4.26)～(4.29)をnについて加える。

$$\begin{aligned} \sum_{n=0}^{\infty} \zeta_n &= wpr + (1-q)\sum_{n=0}^{\infty} \zeta_n \\ &\quad + \zeta_0\, q(1-p)^2 r\{1-(1-p)(1-r)\}^{-1} \\ &\quad + \zeta_1\, q(1-p)^2 r\{1-(1-p)(1-r)\}^{-1} \\ &\quad + \cdots \end{aligned} \quad (A\cdot 12)$$

$\rightleftarrows$

$$\begin{aligned} \sum_{n=0}^{\infty} \zeta_n &= wpr + (1-q)\sum_{n=0}^{\infty} \zeta_n \\ &\quad + q(1-p)^2 r(p+r-pr)^{-1} \sum_{n=0}^{\infty} \zeta_n \end{aligned} \quad (A\cdot 13)$$

これを$\sum_{n=0}^{\infty} \zeta_n$についてとくと,

$$\sum_{n=0}^{\infty} \zeta_n = wr(p+r-pr)q^{-1}(1+r-pr)^{-1} \quad (A\cdot 14)$$

となる。式(A・14)に式(4.25)を用いると式(4.30)が得られる。

上記議論において ζ_n を η_n でおきかえると，同様にして次式が得られる。

$$\sum_{n=0}^{\infty}\eta_n = wp(1-p)r+(1-q)\sum_{n=0}^{\infty}\eta_n + q(1-p)^2 r(p+r-pr)^{-1}\sum_{n=0}^{\infty}\eta_n + wp(1-p)^2(1-r)r(p+r-pr)^{-1} \quad (A\cdot 15)$$

これを $\sum_{n=0}^{\infty}\eta_n$ について とくと，

$$\sum_{n=0}^{\infty}\eta_n = wr(1-p)q^{-1}(1+r-pr)^{-1} \quad (A\cdot 16)$$

となる。式（A・16）に式（4.25）を用いると式（4.35）が得られる。

付録5 ： 式（4.37），（4.45）の証明

ϕ_m の定義より各項は非負であり，その和 $\sum_{m=1}^{\infty}\phi_m$ が0以上で v をこえないことは明らかである。式（4.36）を m について加える。

$$\begin{aligned}\sum_{m=1}^{\infty}\phi_m &= wr(1-p)(v-w)^{-1}[\zeta_0\{1-(1-p)(1-r)\}^{-1} \\ &\quad + \zeta_1\{1-(1-p)(1-r)\}^{-1}+\cdots] \\ &= wr(1-p)(v-w)^{-1}(p+r-pr)^{-1}\sum_{i=0}^{\infty}\zeta_i\end{aligned} \quad (A\cdot 17)$$

式（A・17）に式（4.30）を用いると式（4.37）が得られる。

上記議論において，ζ を η でおきかえると，同様にして次式が得られる。

$$\sum_{m=1}^{\infty}\psi_m = wr(1-p)(v-w)^{-1}(p+r-pr)^{-1}\sum_{i=0}^{\infty}\eta_i \quad (A\cdot 18)$$

式（A・18）に式（4.35）を用いると式（4.45）が得られる。

付録6 ： 式（4.41）～（4.43），式（4.47）～（4.49）の証明

式（4.26），（4.27）を式（4.36）に用いれば，式（4.41），（4.42）はただちに得られる。さらに ϕ_m（$m\geq 3$）は，式（4.29）を式（4.36）に用いることにより，次のようにかける。

$$\begin{aligned}\phi_m &= wr(1-p)(v-w)^{-1}\sum_{j=2}^{m-1}[(1-p)^{m-j-1}(1-r)^{m-j-1} \\ &\quad \times\{\sum_{i=0}^{j-2}q(1-p)^{j-i}(1-r)^{j-i-2}r\zeta_i+\zeta_{j-1}(1-q)\}] \\ &\quad + wr(1-p)(v-w)^{-1}\{(1-p)^{m-2}(1-r)^{m-2}\zeta_1 \\ &\quad + (1-p)^{m-1}(1-r)^{m-1}\zeta_0\}\end{aligned} \quad (A\cdot 19)$$

式（A・19）の第1項に着目し，これを $\Delta\phi$ とおく。

$$\begin{aligned}\Delta\phi &= \sum_{j=2}^{m-1}[(1-p)^{m-j-1}(1-r)^{m-j-1}\{wr(1-p)(v-w)^{-1}qr \\ &\quad \times\sum_{i=0}^{j-2}(1-p)^{j-i}(1-r)^{j-i-2}\zeta_i\}] \\ &\quad + wr(1-p)(v-w)^{-1}(1-q)\end{aligned}$$

$$\times \sum_{j=2}^{m-1} \{ (1-p)^{m-j-1} (1-r)^{m-j-1} \zeta_{j-1} \} \qquad (A \cdot 20)$$

式(A・20)で $j' = j-1$ とおき，式(4.27)，(4.36) を用いて変形する。

$$\begin{aligned}
\Delta\phi = & \sum_{j'=1}^{m-2} [(1-p)^{m-j'-2} (1-r)^{m-j'-2} \{ wr(1-p)^3 (v-w)^{-1} qr \\
& \times \sum_{i=0}^{j'-1} (1-p)^{j'-i-1} (1-r)^{j'-i-1} \zeta_i \}] \\
& + wr(1-p)(v-w)^{-1}(1-q) \\
& \times \sum_{j'=0}^{(m-1)-1} \{ (1-p)^{(m-1)-j'-1} (1-r)^{(m-1)-j'-1} \zeta_{j'} \\
& - (1-p)^{m-2} (1-r)^{m-2} \zeta_0 \} \\
= & \sum_{j'=1}^{m-2} \{ (1-p)^{m-j'-2} (1-r)^{m-j'-2} (1-p)^2 qr \phi_{j'} \} \\
& + (1-q)\phi_{m-1} \\
& - wr(1-p)(v-w)^{-1} (1-p)^{m-2} (1-r)^{m-2} (1-q) \zeta_0 \\
= & qr \sum_{j'=1}^{m-2} \{ (1-p)^{m-j'} (1-r)^{m-j'-2} \phi_{j'} \} + (1-q)\phi_{m-1} \\
& - wr(1-p)(v-w)^{-1} (1-p)^{m-2} (1-r)^{m-2} \zeta_1
\end{aligned} \qquad (A \cdot 21)$$

式(A・19)に式(A・21)を代入すると，

$$\begin{aligned}
\phi_m = & qr \sum_{j'=1}^{m-2} \{ (1-p)^{m-j'} (1-r)^{m-j'-2} \phi_{j'} \} + (1-q)\phi_{m-1} \\
& + wr(1-p)^m (1-r)^{m-1} (v-w)^{-1} \zeta_0
\end{aligned} \qquad (A \cdot 22)$$

となる。式(A・22)に式(4.26)を用いると式(4.43)が得られる。

次に式(4.47)～(4.49)については，式(4.31)，(4.32)を式(4.44)に用いれば式(4.47)，(4.48)はただちに得られる。さらに，ψ_m ($m \geq 3$)は，式(4.34)を式(4.44)に用いることにより，次のようにかける。

$$\begin{aligned}
\psi_m = & wr(1-p)(v-w)^{-1} \sum_{j=2}^{m-1} [(1-p)^{m-j-1} (1-r)^{m-j-1} \\
& \times \{ \sum_{i=0}^{j-2} q(1-p)^{j-i} (1-r)^{j-i-2} r \eta_i \\
& + \eta_{j-1}(1-q) + \eta_0 (1-p)^j (1-r)^j \}] \\
& + wr(1-p)(v-w)^{-1} \{ (1-p)^{m-2} (1-r)^{m-2} \eta_1 \\
& + (1-p)^{m-1} (1-r)^{m-1} \eta_0 \}
\end{aligned} \qquad (A \cdot 23)$$

式(A・23)の第1項に着目し，これを $\Delta\psi$ とおく。上記議論で，ζ を η でおきかえ，式(4.32)，(4.44)を用いると，次式が得られる。

$$\begin{aligned}
\Delta\psi = & qr \sum_{j'=1}^{m-2} \{ (1-p)^{m-j'} (1-r)^{m-j'-2} \psi_{j'} \} + (1-q)\psi_{m-1} \\
& - wr(1-p)(v-w)^{-1} (1-p)^{m-2} (1-r)^{m-2} (1-q) \eta_0
\end{aligned}$$

$$+ w r (1-p)(v-w)^{-1} \sum_{j'=1}^{m-2} \{ (1-p)^{m-1} (1-r)^{m-1} \eta_0 \}$$

$$= q r \sum_{j'=1}^{m-2} \{ (1-p)^{m-j'} (1-r)^{m-j'-2} \psi_{j'} \} + (1-q) \psi_{m-1}$$

$$- \{ w r (v-w)^{-1} (1-p)^{m-1} (1-r)^{m-2} \eta_1$$

$$- w^2 r^2 p (1-p)^{m+1} (1-r)^{m-1} (v-w)^{-1} \}$$

$$+ (m-2) w r (1-p)^m (1-r)^{m-1} (v-w)^{-1} \eta_0 \qquad (A \cdot 24)$$

式（A・24）を式（A・23）に代入し，式（4.31）を用いれば，ただちに式（4.49）が得られる。

付録7 ： 式（2.13），（2.14）および式（2.18）～（2.21）の線形独立性

ボトルネック・プロセッサ k（$\in H$）について，式（2.14）の値を A_k とおく（証明の都合で j のかわりに k を用いる）。

$$A_k \triangleq \frac{d_{i_1 k}}{T_{i_1} + z_{i_1}} = \frac{d_{i_2 k}}{T_{i_2} + z_{i_2}} = \cdots \qquad (A \cdot 25)$$

$$(i_1, i_2, \cdots \in y_k ; k \in H)$$

よって $\forall i$（$\in y_k$）について，

$$\frac{1}{T_i + z_i} = \frac{A_k}{d_{ik}} \qquad (A \cdot 26)$$

とかける。式（A・26）を式（2.13）に用いて次式を得る。

$$\sum_{k \in H} \left(\sum_{i \in y_k} N_i \frac{d_{ij}}{d_{ik}} \right) A_k = 1 - \sum_i (1 - e_i) \frac{N_i d_{ij}}{R_{i0} + z_i} \qquad (A \cdot 27)$$

$$(j \in H)$$

いま，Hの要素数を $|H|$ とし，プロセッサ1～ $|H|$ がボトルネックであるとしても一般性を失なわない。このとき，$A_1, A_2, \cdots, A_{|H|}$ を変数とした次のベクトル形式の方程式が得られる。

$$\begin{bmatrix} \sum_{i \in y_1} N_i \frac{d_{i1}}{d_{i1}} & \sum_{i \in y_2} N_i \frac{d_{i1}}{d_{i2}} & \cdots & \sum_{i \in y_j} N_i \frac{d_{i1}}{d_{ij}} & \cdots & \sum_{i \in y_{|H|}} N_i \frac{d_{i1}}{d_{i|H|}} \\ \sum_{i \in y_1} N_i \frac{d_{i2}}{d_{i1}} & \sum_{i \in y_2} N_i \frac{d_{i2}}{d_{i2}} & \cdots & \sum_{i \in y_j} N_i \frac{d_{i2}}{d_{ij}} & \cdots & \sum_{i \in y_{|H|}} N_i \frac{d_{i2}}{d_{i|H|}} \\ \vdots & \vdots & & \vdots & & \vdots \\ \sum_{i \in y_1} N_i \frac{d_{ij}}{d_{i1}} & \sum_{i \in y_2} N_i \frac{d_{ij}}{d_{i2}} & \cdots & \sum_{i \in y_j} N_i \frac{d_{ij}}{d_{ij}} & \cdots & \sum_{i \in y_{|H|}} N_i \frac{d_{ij}}{d_{i|H|}} \\ \vdots & \vdots & & \vdots & & \vdots \\ \sum_{i \in y_1} N_i \frac{d_{i|H|}}{d_{i1}} & \sum_{i \in y_2} N_i \frac{d_{i|H|}}{d_{i2}} & \cdots & \sum_{i \in y_j} N_i \frac{d_{i|H|}}{d_{ij}} & \cdots & \sum_{i \in y_{|H|}} N_i \frac{d_{i|H|}}{d_{i|H|}} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_j \\ \vdots \\ A_{|H|} \end{bmatrix} = \begin{bmatrix} 1 - \sum_i (1-e_i) \frac{N_i d_{i1}}{R_{i0} + z_i} \\ 1 - \sum_i (1-e_i) \frac{N_i d_{i2}}{R_{i0} + z_i} \\ \vdots \\ 1 - \sum_i (1-e_i) \frac{N_i d_{ij}}{R_{i0} + z_i} \\ \vdots \\ 1 - \sum_i (1-e_i) \frac{N_i d_{i|H|}}{R_{i0} + z_i} \end{bmatrix}$$

$$(A \cdot 28)$$

式（A・28）の左辺の $|H|$ 行 $|H|$ 列の行列に着目すると，そのrankが $|H|$ ならば A_k（$k=1,2,\cdots,|H|$）は求まり，したがって式（A・26）より T_i が得られる。

次にこのrankが $|H|$ でない条件について考える。このとき，ある行が他の行の線形和になるので，次式が成り立つ。

$$\sum_{i\in y_k} N_i \frac{d_{ij}}{d_{ik}} = \sum_{p\neq j} c_p \left(\sum_{i\in y_k} N_i \frac{d_{ip}}{d_{ik}} \right) \qquad (k=1,2,\cdots,|H|) \qquad (A\cdot29)$$

ここで c_p は定数である。式（A・29）をかきなおして，

$$\sum_{i\in y_k} \left(d_{ij} - \sum_{p\neq j} c_p d_{ip} \right) \frac{N_i}{d_{ik}} = 0 \qquad (k=1,2,\cdots,|H|) \qquad (A\cdot30)$$

を得る。一般にクラスが異なるとトランザクションによる資源使用法は異なり，d_{ij} と d_{ip}（$p\neq j$）の比もクラスごとに異なる値をとる。 y_k は排反集合であり，これに属するクラスは k ごとに全く異なる。一方， c_p は各 k について共通である。したがって，式（A・30）は成立せず，式（2.13），（2.14）は独立である。

次に式（2.18）～（2.21）の線形独立性につき考える。まず，式（2.19）の値を A_k とおくと各 k（$\in H$）につき式（A・26）が得られる。また，式（2.21）の値を B とおくと次式が得られる。

$$\frac{1}{T_i+z_i} = \frac{B}{R_{i0}/s_i} \qquad (i\in y_m \text{かつ} i\notin \bigcup_{k\in H} y_k) \qquad (A\cdot31)$$

$$\frac{1}{T_i+z_i} = \frac{B}{R_i/s_i} \qquad (i\in y_m \text{かつ} i\in \bigcup_{k\in H} y_k) \qquad (A\cdot32)$$

式（A・32）と式（A・26）より，次式が得られる。

$$R_i = \frac{B}{A_k} d_{ik} s_i \qquad (i\in y_m \text{かつ} i\in y_k,\ k\in H) \qquad (A\cdot33)$$

次に式（A・26）と式（A・31）を式（2.18）に用いて，

$$\sum_{k\in H} \left(\sum_{i\in y_k} N_i \frac{d_{ij}}{d_{ik}} \right) A_k + \left(\sum_i (1-e_i) e_{im} \frac{N_i d_{ij}}{R_{i0}/s_i} \right) B = 1 - \sum_i (1-e_i)(1-e_{im}) \frac{N_i d_{ij}}{R_{i0}+z_i} \qquad (A\cdot34)$$

$$(j\in H)$$

が得られる。ここで式（A・26）より次式がなりたつ。

$$\frac{T_i}{T_i+z_i} = 1 - \frac{z_i}{T_i+z_i} = 1 - \frac{z_i}{d_{ik}} A_k \qquad (A\cdot35)$$

$$(i\in y_k,\ k\in H)$$

式（A・31），（A・32），（A・35）を式（2.20）に用いて

$$\sum_{k\in H} \left\{ -\sum_{i\in y_k} (1-e_{im}) \frac{N_i z_i}{s_i d_{ik}} \right\} A_k + \left(\sum_i e_{im} N_i \right) B$$

$$= 1 - \sum_i (1-e_i)(1-e_{im}) \frac{N_i R_{i0}/s_i}{R_{i0}+z_i} - \sum_{k \in H} \left\{ \sum_{i \in y_k} (1-e_{im}) \frac{N_i}{s_i} \right\} \qquad (A \cdot 36)$$

を得る。

A_1，A_2，…，$A_{|H|}$，Bを未知数とする線形方程式（A・34），（A・36）を，式（A・28）と同様なベクトル形式で表わすことができる。このとき，（$|H|+1$）行（$|H|+1$）列の行列の rank は（$|H|+1$）であり，解が一意的に求まることが式（A・28）と同様にして言える。得られた A_1，A_2，…，$A_{|H|}$，Bを式（A・26），（A・31），（A・33）に用いて，T_i と R_i が求まる。

付録8 ： 式（2.26）および式（2.29）の求解の条件

式（2.26）は式（2.29）において $\psi_m = 0$ の場合に対応するので，式（2.29）のみについて考えればよい。クラス i トランザクションのボトルネック・プロセッサにおける平均待ち時間の和を b_i とすると，これは，

$$b_i = \sum_{j=1}^{|H|} \delta_{ij} \theta_{ij} \psi_j \qquad (i = 1, 2, \cdots, \ell) \qquad (A \cdot 37)$$

で与えられる。式（A・37）はベクトル形式で次のように表わせる。

$$\boldsymbol{b} = \boldsymbol{D} \cdot \boldsymbol{\psi} \qquad (A \cdot 38)$$

ただし，

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_\ell \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} \delta_{11}\theta_{11} & \vdots & \\ & \ddots & \\ \cdots\cdots & \delta_{ij}\theta_{ij} & \cdots\cdots \\ & \vdots & \ddots \\ & & \delta_{\ell|H|}\theta_{\ell|H|} \end{bmatrix}, \quad \boldsymbol{\psi} = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{|H|} \end{bmatrix}$$

b_i を用いて式（2.29）をかき直すと，

$$\begin{cases} \sum_i \dfrac{d_{ij} N_i}{R_{i0} + b_i + \delta_{im}\theta_{im}\psi_m + z_i} = 1 \\ \qquad\qquad (j = 1, 2, \cdots, |H|) \\ \sum_i \dfrac{\{(R_{i0}+b_i)/s_i\} N_i}{R_{i0} + b_i + \delta_{im}\theta_{im}\psi_m + z_i} = 1 \end{cases} \qquad (A \cdot 39)$$

となる。

まず，式（2.27）の必要性を証明する。明らかに，

$$\mathrm{rank}(\boldsymbol{D}) \leq |H|$$

である。いま，

$$\mathrm{rank}(\boldsymbol{D}) \leq |H| - 1$$

とすると，独立変数 b_i の数は（$|H|-1$）個以下となり，式（A・39）の解は一般に存在しない。したがって式（2.27）は求解の必要条件である。

次に十分性を証明する。いま，式(2.27)が成立するとき，式(A・39)は(｜H｜＋1)個の独立変数 b_{i1}，b_{i2}，…，$b_{i|H|}$，ψ_m により表わせる。この方程式は解をもつ。得られた解と式(A・37)とから ψ_1，ψ_2，…，$\psi_{|H|}$ が求まる。

付録9 ： 記号・略号定義一覧

以下に，本論文で用いる主な記号，略号を示す。なお，同一記号が章ごとに相異なる量を表わす場合もあるので，使用する章番号を参照されたい。

記　　号	内　　容	使用章番号
A	トランザクション・クラス	2
AMI／II	Asymptotic Model：漸近モデルI／II	2
a	スロット割当てのパラメータ	4
B	トランザクション・クラス	2
C	トータル・コスト，ページ入出力実行時間	4
C_j^i	方式 i，ページ入出力動作 j のトータル・コスト	4
CH	Channel	5
CPU	Central Processing Unit	2，3，4，5
CN	Concentrating Technique：集中方式	4
D	ディスク回転時間	4
$d_j(d_{ij})$	(クラス i トランザクションの)資源 j 使用時間	2
d_x	$\max_j d_j$	2
E	ブロック・サイズ	4
E型	Executive型	2，5
F_0	最小シーク時間	4
F_j^i	方式 i，ページ入出力動作 j の期待シーク時間	4
FL	Floating Technique：浮動方式	4
FX	Fixed Technique：固定方式	4
f	サービス目標関数	5
$f(\hat{f})$	ページ・フォールト率(の推定量)	3
G_j^i	方式 i，ページ入出力動作 j の期待サーチ時間	4
GRM	General Resources Manager：一般資源管理方式	5
$g(g_i)$	(トランザクション i の)サービス目標関数	5
g	ページ・フォールト回数	3
H	ボトルネック・プロセッサの集合	2

記　　号	内　　容	使用章番号
H	スロット数	4
$h(\hat{h})$	スロット割当ての自由度（最適自由度）	4
I型	Interactive型	2，5
IO	入出力装置	2，5
i	トランザクション/トランザクション・クラスの添字	2
j	資源の添字	2
K	1シリンダあたりシーク時間	4
L	ライフタイム	3
L	二次メモリのしめるシリンダ数	4
MEM	実メモリ	2，5
$N(N_i)$	（クラスiトランザクションの）呼源数	2
NWL	Normalized Workload Level	5
OWE	Optimum Working−set Estimator：ワーキングセット最適化方式	3
$P_i(j)$	トランザクション（クラス）iの資源j割当て優先度	2，5
PF	ページ・フォールト	4
PG	ページ入出力装置	2
PO	ページ・アウト	4
p	ページに書きこみの行なわれる確率	4
q	ページがワーキングセットに加わる確率	4
$R(R_i)$	（クラスiトランザクションの）平均実メモリ占有時間	2
$R_0(R_{i0})$	（クラスiトランザクションの）待ちなしの応答時間	2
$r(r_i)$	（トランザクションiの）資源サービス率	5
r	ページ，トランザクションあたりのページ・フォールト発生率	4
S	ページ・フォールト処理時間	3
SI	スワップ・イン	4
SO	スワップ・アウト	4
STP	Space Time Product	3
$s(s_i)$	（クラスiトランザクションの場合の）最大多重度	2
su	service unit	3，5
$T(T_i)$	（クラスiトランザクションの）平均応答時間	2
t_n	仮想時間軸上のサンプル時点	3
u_j/u_m	資源（プロセッサ）j/実メモリの利用率	2，5
u_x	$\max_j u_j$	2

記　　号	内　　容	使用章番号
V	実メモリ容量	2
V	ページ参照列の長さ	3
VOS3	Virtual－storage Operating System 3	2，3，4，5
v_{ij}	トランザクションiの資源j使用特性値	5
v	プログラム・サイズ	4
W	ワーキングセット	3
WL	Workload Level: 負荷レベル	5
w	ワーキングセット・サイズ	2，3，4
$\bar{w}$（$\hat{w}$）	平均ワーキングセット・サイズ（の推定量）	3
X（$\hat{X}$）	単位参照あたりのSpace Time Product（の推定量）	3
x	ボトルネックとなりうる（$\max_j u_j$の）プロセッサ	2
x	ワーキングセットの蓄積されるスロット群の属する相異なるブロック数	4
y	前項xブロックを形成するためのシーク動作開始時点における相異なるディスク・ヘッド位置	4
y_j / y_m	プロセッサj／実メモリの最低優先度トランザクションの集合	2
z（z_i）	（クラスiトランザクションの）平均思考時間	2
α	重み係数	2，3，5
Γ	ブロック数	4
Δ	サンプリング間隔	3
ζ	ブロック・サイズ（スロット数）	4
η	ブロック・サイズ（スロット数）	4
θ	トランザクションのページ参照回数	4
θ_{ij} / θ_{im}	クラスiトランザクションによるプロセッサj／実メモリの平均使用回数	2
Λ（Λ_i）	（トランザクションiの）資源サービス量累計	5
λ	ワーキングセット外のページが参照される確率	4
λ_{ij}	トランザクションiの資源jサービス量累計	5
ξ_i	トランザクションiのNWLの，NWL平均値からの偏差上限値	5
Π，Π'	ブロック数	4
ρ_j	資源jの利用率下限値	5
τ	トランザクション到着以来の経過時間	5
τ	ウインド・サイズ	3

記　　号	内　　容	使用章番号
ϕ	ブロック・サイズ（スロット数）	4
ψ	ブロック・サイズ（スロット数）	4
ψ_j / ψ_m	プロセッサ j ／実メモリを１回使用するための平均待ち時間	2