The University of Osaka
Institutional Knowledge Archive

Title	トランザクション制御方式と性能評価に関する研究
Author(s)	松田, 晃一
Citation	大阪大学, 2000, 博士論文
Version Type	VoR
URL	https://doi.org/10.11501/3169458
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

トランザクション制御方式と性能評価に関する研究

2000年1月

松田晃一

IF 7530

謝辞

本論文は、筆者が日本電信電話株式会社（ＮＴＴ）電気通信研究所在職中および、大阪大学大学院在学中に、大阪大学大学院工学研究科通信工学専攻 教授 池田 博昌博士の御指導の下に行った研究成果をまとめたものである。

本論文執筆に関して親切丁寧なご指導、ご鞭撻を賜った大阪大学大学院工学研究科通信工学専攻 教授 池田 博昌博士に心から感謝の意を表します。

また、その成果をとりまとめるにあたり、大阪大学産業科学研究所教授 元田 浩博士には、終始懇切丁寧なご指導とご鞭撻を賜り謹んで感謝の意を表します。

終始ご教授、ご指導戴くとともに、本論文に対して有益なご討論、ご助言を賜った大阪大学大学院工学研究科通信工学専攻教授 森永 規彦博士、同教授小牧 省三博士、同教授 前田 肇博士、同教授 塩澤 俊之博士、大阪大学大学院工学研究科電子情報エネルギー工学専攻 北山 研一博士、オハイオ州立大学教授 児玉 裕治 博士（前 大阪大学大学院工学研究科通信工学専攻教授）に謹んで感謝の意を表します。

本論文に関し懇切丁寧なご指導、ご鞭撻を賜った大阪大学大学院工学研究科通信工学専攻助教授 山本 幹博士に感謝の意を表します。

終始ご教授、ご指導戴くとともに、本論文に関して有益なご討論、ご助言を賜った大阪大学大学院工学研究科通信工学専攻助教授 三瓶 政一博士、同助教授 松本 正行博士、同助教授 飯國 洋二博士、同助教授 塚本 勝俊博士、大阪大学大学院工学研究科電子情報エネルギー工学専攻助教授原 晋介博士、大阪大学産業科学研究所助教授 鷲尾 隆博士、大阪大学大学院工学研究科通信工学専攻講師 戸田 裕之博士に謹んで感謝の意を表します。

本論文の執筆の機会を与えていただいたＮＴＴエレクトロニクス株式会社社長伊澤 達夫博士ならびにＮＴＴ取締役第三部門長鈴木 滋彦博士に心から感謝いたします。

日本電信電話株式会社入社以来、本研究の遂行には多数の方々から直接、間接に御指導や御協力を頂きました。特に富士通研究所フェロー戸田 巌博士には本研究全般にわたって多大な御指導と御助言を頂き、心から感謝致します。また、東海大学理事教授 吉田 庄司博士、東京工科大学教授 伊吹 公夫博士、ＮＴＴソフトウェア株式会社 高村 眞司相談役、同社社長 鶴保 征城博士、国際短期大学 新井 克彦教授、大阪大学大学院基礎工学研究科教授 橋本昭洋博士お

よび一橋大学教授 石野 福彌博士には、研究の初期の段階からまとめの段階にいたるまで適切な問題提起と種々の御指導を頂きました。それぞれの方々に深く感謝致します。

本研究は、ＮＴＴ通信研究所の多くの方々の御協力により進めることができました。特に、本研究における性能評価の方法について御指導頂き、様々な御助言を頂いた、図書館情報大学教授 徳山 五郎博士、芝浦工業大学教授 村尾 洋博士、ＮＴＴアドバンステクノロジ株式会社トラヒックリサーチセンタ所長 川島 幸之助博士に深く感謝致します。

また、研究内容について熱心な御討論を頂いた、ＮＴＴソフトウェア株式会社大橋 楷一郎情報システム部長、同大南 正人モバイルネットワーク事業部長、ＮＴＴコムウェア豊嶋 祥司技術開発部長、ＮＴＴプラットフォーム研究所中島 寿生主幹研究員、ＮＴＴサイバーコミュニケーション総合研究所川嶋 伸夫主幹研究員、ＮＴＴデータ株式会社佐藤 栄システム技術部長、同十倉 建二担当部長、ＮＴＴソフトウェア株式会社小菊 一三担当部長、同小松 俊雄担当課長、ＮＴＴ情報流通プラットフォーム研究所杉村 康主任研究員に心から感謝致します。

本研究は、以上の方々のほかに数多くの方々の御指導、御協力のもとに達成されたものであり、ここに謹んで感謝の意を表します。

内容梗概

本論文は、著者がＮＴＴ電気通信研究所に在職中、および大阪大学大学院工学研究科（通信工学専攻）在学中に行った、トランザクション制御方式と性能評価に関する研究の成果をまとめたものであり、序論、結論を含め全体としては以下の６章から成っている。

第1章では、本研究の背景について述べるとともに、研究の対象であるトランザクション処理システムを概観し、本論文の目的と位置づけを明確にする。

第２章では、トランザクション処理システムの電文制御の方式設計について論ずる。トランザクション処理システムが多種多様な業務に適用できるためには、業務の骨格を形成する電文の流れに関する設計に大きな自由度が必要である。特に、トランザクション処理システムにおいては、トランザクション処理モニタがオペレーティングシステムと業務プログラムとの間に介在し、電文の送受を制御する。従って、業務の多様性に答えられるためには、このトランザクション処理モニタにおける電文の制御機能の多様なサービスへの適用性の確保が重要である。

本章ではこの課題に対して、まずトランザクション処理システムにおける電文制御の要求条件を分析・体系化し、それらを満足する電文制御方式を明らかにした。従来はシステム毎に専用のプログラム開発が行われていた電文制御について、共通的なプログラムを用いて多様なサービスが実現できる汎用的な方式として考案したゲート制御方式を提案する。さらに、電文制御に対する要求条件の大きく異なる３つの実サービスへこのゲート制御方式の適用結果を評価し、目標とした汎用性が達成されていることを明らかにしている。

第３章では、トランザクション処理システムにおいてシステム全体の性能に大きな影響を与えるジャーナル処理の性能を評価し、高性能化のための新しいジャーナル処理方式を提案する。

トランザクション処理システムに故障が生じた時には、データベースの内容や端末との間で送受された電文の内容を矛盾の無い状態に復元した上でシステムの正常な運転を迅速に再開することが必要である。この目的のために、運転中に再開処理に必要

な情報をトランザクション毎に時系列順にジャーナルとして記録する。このためジャーナル処理の性能はシステム全体の性能に大きく影響を及ぼし、その高性能化は重要な課題である。

本章では、シミュレーションによってジャーナル処理方式の性能評価を行い、性能上の隘路を詳細に分析する。そして、排他制御によるジャーナル処理ルーチンの保留時間を短縮するために、排他制御を入出力割込み処理のレベルで実現する新たな方式を提案する。さらに、提案の方式は、システム全体の性能を約２０％改善する効果をもたらすことをシミュレーションを用いて明らかにする。

第４章では、大容量記憶装置（Mass Storage System : ＭＳＳ）をオンラインシステムに適用し効率的に制御するためのトラヒック設計の方法を提案し、設計結果の妥当性、制御方式の有効性を定量的に実証する。

トランザクション処理システムの運転においては、処理の履歴情報を格納するジャーナルやオンラインファイルのバックアップ情報などのように、オンラインでの直接のアクセスは無いが、大容量で且つ長期間保存すべきファイルが必要である。これらには、通常磁気テープ装置が用いられてきたが、システムの規模の増大に伴い、量が急増し、操作コストの増大、操作誤りなどテープ操作の煩雑さに起因する各種の問題が大きくなってきた。また、必ずしも短い応答時間は必要無いが、低コストのオンラインで用いるファイルに対するニーズも顕在化してきた。

この様な要請に応えるために自動操作機構をもった磁気テープ装置と磁気ディスク装置とを組み合わせた大容量記憶装置(Mass Storage System :ＭＳＳ）が開発された。

本章では、まずＭＳＳの評価モデルとして、系内客数制限をもつ開放形の待ち行列ネットワークを提案し、その解析方法および平衡状態における各種特性値を得る方法を示す。また、系が平衡状態にあるための条件について考察し、理論的な限界スループットを明らかにする。さらに、モデルの解析結果に基づく数値計算例を実測値と比較し、良く一致することを示し、モデルの妥当性を実証する。

第５章では、ソフトウェア資源とハードウェア資源の関係に代表されるような、資源と資源の間に階層的な関係が存在するシステムの性能評価を行うための一般的なモデルを提案する。本論文の第２章の電文制御方式、第３章ジャーナル処理方式や第４章大容量記憶装置のそれぞれにおいて個別的に示してきたトラヒック設計法をさらに高度化し、より広い設計問題に適用可能なトラヒック設計法に一般化を試みるものである。

トランザクション処理システムにおいては、タスクはシステムの重要な資源であり、

その最適構成を得ることは重要であるが、評価にあたってはタスク単体ではなく、タスクの実態を担っているハードウェア資源との相互作用を含めて表現できるモデルが必要である。これまでの研究はハードウェア資源のみ、あるいはタスクなどのソフトウェア資源のみを単独で取り扱うモデルであり、このような要求に十分応えられるとは言えない。

ここでは、まずタスクをハードウェア資源と関係づけて評価する手法として、ハードウェアを窓口とする待ち行列モデルの一部に存在し得る客数を一定数に制限することによってタスクの存在を表現する系内客数制限モデルを提案する。外からの客の到着が無い閉じた系内客数制限モデルについて解析の方法を提案し、それを用いてタスク構成を評価し、本モデルが当初の目的に応える有効なものであることを明らかにする。さらに、より一般的なモデルとして外部から客の到着がある開放型の系内客数制限モデルについて、その解析の手法を考案し、それに基づいて系が安定であるための必要十分条件を明らかにし、理論的な限界値や系の各種特性値を解析的に得る方法を提案する。また、考案した解析手法をもとに実際的な数値計算の手順を明らかにする。

第６章では、本研究で得られた成果を総括すると共に、その意義および今後の課題を述べる。

目次

謝辞

梗概

第 1 章 序論 1
1.1 研究の背景 1
1.2 研究の対象 3
1.2.1 トランザクション処理とその研究課題 3
1.2.2 トランザクション処理システムのソフトウェア構成 7
1.2.3 タスクの概念 8
1.3 本論文の概要 9

第 2 章 トランザクション処理システムにおける電文制御方式の設計 13
2.1 緒言 13
2.2 トランザクション処理における電文制御の要求条件 14
2.2.1 電文の流れの多様性 14
2.2.2 端末および端末制御機能の多様化 16
2.2.3 端末通信処理の効率化 16
2.3 電文制御機能の方式設計 17
2.3.1 端末交信処理と業務処理の分離 17
2.3.2 電文待ち行列と電文スケジュール機能 19
2.4 電文スケジュール機能の設計 21
2.4.1 汎用化の考え方 22
2.4.2 電文スケジュール機能の抽象化 22
2.4.3 ゲート制御の機能と構成 24
2.4.4 ゲート制御のサービス適用性 26
2.5 結言 29

第 3 章 ジャーナル処理方式の設計 31
3.1 緒言 31
3.2 障害復旧処理とジャーナル処理方式 32
3.2.1 障害復旧処理 32
3.2.2 ジャーナル処理の方式条件 33
3.2.3 ジャーナル処理方式 36
3.3 方式評価 39
3.3.1 評価の条件 39
3.3.2 シミュレーション結果と考察 43
3.4 高性能ジャーナル処理方式の設計 46
3.4.1 高性能ジャーナル処理方式 46
3.4.2 高性能ジャーナル方式の評価 49
3.5 総合評価 51
3.6 結言 52

第 4 章 大容量記憶装置制御方式のトラヒック設計 55
4.1 緒言 55
4.2 大容量記憶装置の動作概要と待ち行列モデル 56
4.2.1 大容量記憶装置の概要 56
4.2.2 大容量記憶装置の構成と動作 58
4.2.3 大容量記憶装置の待ち行列モデル 61
4.3 待ち行列モデルの解析 63
4.3.1 系の状態定義 63
4.3.2 状態方程式の導出 64
4.3.3 母関数の導入 65
4.3.4 特性値 67
4.4 系の平衡条件に関する考察 68
4.4.1 データ記録装置数が 1 の場合の平衡条件 68
4.4.2 データ記録装置数が 2 の場合の平衡条件 68
4.5 数値計算例および実測との比較 70
4.5.1 数値計算の前提 70
4.5.2 スループットおよびアクセス時間 70
4.5.3 フォールバックの影響 72

4.5.4 アクセッサとデータ記録装置の使用率 72
4.5.5 実測値との比較 73
4.6 結言 75

第5章 トランザクション制御方式のトラヒック設計 77
5.1 緒言 77
5.2 トランザクション処理システムのタスク構造 78
5.3 タスク構造のモデル化 80
5.3.1 タスクのモデル化 80
5.3.2 トランザクション処理システムのタスク構成のモデル化 81
5.3.3 モデルの解析 82
5.4 タスク構造の性能解析 82
5.4.1 タスク数と応答時間の関係 82
5.4.2 ＣＰＵ台数とスループットの関係 84
5.4.3 シミュレーションとの比較 85
5.5 タスクモデルの一般化 87
5.6 モデルの解析 88
5.6.1 状態の定義 88
5.6.2 平衡方程式の導出 89
5.6.3 母関数に関する解析 92
5.7 系の安定性に関する考察 97
5.8 数値計算手順 105
5.8.1 状態確率 105
5.8.2 特性値 107
5.8.3 数値計算手順 108
5.9 結言 112

第6章 結論 113

参考文献 119

関連発表一覧 125

第1章

序論

1.1 研究の背景

電電公社（現ＮＴＴ、以下ＮＴＴ）のデータ通信用標準情報処理システムＤＩＰＳ（Denden-kosya Information Processing System）は、1969年に実用化を開始し、1973年に最初の商用サービスを開始した[44]。この商用サービスは、公衆データ通信サービスと呼ばれるもので、大型コンピュータを多数のユーザが共同で利用するタイムシェアリング方式によって、科学技術計算などのサービスを安価に提供するものであった。ＤＩＰＳの最初の版であるＤＩＰＳ－１[45],[43],[21]は当時の国産最大機種の約3倍の性能を目標とし、マルチプロセッサ方式、ローカルメモリ方式、仮想メモリ方式などの先進的なアーキテクチャを採用したものであり、ハードウェア開発と共に、ＯＳの開発についても世界の先端的なものであった。しかし、タイムシェアリング方式という意味では、ＭＩＴにおけるＭＵＬＴＩＣＳの開発[17]などを通して数多くの研究成果が積み上げられていた。

一方、これに対して企業や官公庁などがその活動に計算機を導入し、オンラインシステム化を推進するようになると、むしろトランザクション処理といわれる処理分野がシステム開発の中心になってきた。ＮＴＴのデータ通信サービスにおいても例えば、全国銀行協会為替交換システム[75]、運輸省車検登録システム[73]、郵政省貯金システム[36] など全国規模の社会基盤的システムを初めとして、開発予定の多くがトランザクション処理システムであった。このトランザクション処理システムは銀行システムや座席予約システムに代表されるように、処理すべき業務はあらかじめ決まった定

型的なものであるため、計算機では処理を行うために必要な準備を事前に行い待機しておき、処理要求が届くと直ちに処理を行うことによって、大量の処理要求を効率的に処理する方式がとられる。これに対し前述したタイムシェアリング方式は利用者と計算機が会話をしながら処理を進める形であり、処理の内容はその時々に異なる非定型的なものである。計算機からの応答を利用者が見て、次の指示を計算機に与えるまでの思考時間の間、計算機資源を別の利用者の処理に割り当てる時分割制御によって、複数の利用者が計算機を共用する方式である。このように、計算機資源の基本的な制御方式がタイムシェアリング方式とトランザクション処理方式とでは異なるため、既に開発されていたタイムシェアリング方式のＯＳをそのままトランザクション処理システムに流用することは困難であった。このため、ＤＩＰＳにおいてもトランザクション処理システム用のＯＳ開発に着手したのが1974年であった。

ところで、トランザクション処理システムが対象とする業務はシステム毎に固有で多様であり、しかもシステムの利用者や運用を行う人にとっては、従来の業務の手順や運用方法の継続性への要求が強く、同種の機能であってもその実現形態に対する要求は多彩となるなど、多種多様な個別のユーザの要求に応える必要があった。その一方で、オンラインシステムとしての性能を確保することも必要で、その両方の要求を両立させることは非常に困難であったため、結局はユーザ毎にＯＳにまで修正を施すなど、個別的なシステム開発になっていた。

しかし、この種のシステムの開発が増大するなかで、このような個別的開発には限界が出てくるのは当然であり、開発期間の短縮や生産性向上、システムの信頼性向上の面などから、トランザクション処理システム用の標準的な制御方式を確立することは、重要かつ緊急の課題であった。

本論文は、このような背景のもとで遂行されたトランザクション処理用ＯＳの研究・開発の中で、トランザクション制御方式と性能評価に関する研究を取りまとめたものである。本研究の成果は、ＤＩＰＳのトランザクション処理用ＯＳのトラヒック設計に具体的に適用され、その設計結果に基づく実装が行われ、前述のナショナルプロジェクトを始めとする数多くのＮＴＴのデータ通信サービスの商用に供された。その実績から、本研究の狙いは十分達成されたものと考えられる。

その後、1980年代以降半導体技術の急速な進展によるマイクロプロセッサや半導体メモリの出現を契機に計算機システムは小型化、分散化の方向、いわゆるダウンサイジングへ大きく変化した。この結果、タイムシェアリングシステムは姿を消し、ワークステーションやパソコンがそれに替る物として使われている。しかし、社会の基幹

システムであるトランザクション処理システムについては、今日に至るもなお大型計算機システムで実現されており、本研究の成果が引き続き生きている。

また、今後インターネットなどによる電子商取引きなど本格的なトランザクション処理がワークステーションやパソコンで実現されることも想定される。しかし、現状におけるこれら機種のＯＳには、本論文で取り上げた問題を始めさまざまな課題があり、基幹システムへの適用にはまだ不十分である。本論文で取り上げた成果がワークステーションなどのＯＳに反映されてくれば、コストパフォーマンスの高いトランザクション処理システムが実現されることが大いに期待される.

1.2 研究の対象

本研究はトランザクション処理システムに関するものである。ここでは、研究対象であるトランザクション処理について概説すると共に、研究上の課題について述べ、研究の位置付けを明らかにする。また、本文の理解を助けるために、トランザクション処理のソフトウェア構成やタスクの概念について概説する[76], [41], [77]。

1.2.1 トランザクション処理とその研究課題

今日、計算機によって処理される対象は広範囲にわたっておりその内容は多種多様であるが、それを処理形態の観点からみると表1.1のように大きく分類することができる。

表 1.1 計算機の処理形態

処理形態の種類
バッチ処理
タイムシェアリング処理
トランザクション処理
リアルタイム処理

バッチ処理は、処理要求が発生する度に直ちに処理を始めるのではなく、ある程度の数になるまでまとめて一括して処理する処理形態である。個々の処理毎のターンアラウンド時間よりも全体としてのスループットを優先する考え方の処理形態である。

タイムシェアリング処理とは、オンラインで計算機と接続された複数の利用者が計

算機と会話しながら一連の処理を進める形態である。科学技術計算やシミュレーション、データ分析など様々な問題に応用される。タイムシェアリング処理の特徴は、計算機からの応答が利用者に戻ってから、次の指示を計算機に与えるまでの利用者の思考中の時間は、計算機が空き状態であることを利用して、その間は計算機資源を別の利用者の処理に割り当てることによって複数の利用者が計算機を共用しようとする方式である。

また、リアルタイム処理は工場のプラント制御やロケット、人工衛星の制御などのように実時間のタイミングに合わせて計算機が必要な応答をしなければならないシステムである。他の処理形態は、応答時間の平均値が設計条件として規定されるのに対し、リアルタイム処理では応答時間の最悪値が一定時間以内に入ることが必要で、時間的制約の最も厳しい処理形態である。

さて、本研究の対象であるトランザクション処理とは、銀行システムや座席予約システムに代表されるように、あらかじめ決まった定型的な業務を高速、大量に処理するシステムである。本来トランザクションとは、商品やサービスに関する取引を意味している。この取引に伴って発生したデータを発生時点で即時に計算機に入力し、計算機で管理されているデータベースの情報を取引に沿って更新し、結果を伝票の形に編集して端末に出力するものである。すなわち簡単に言えば、「取引をオンラインで処理する」ための計算機システムがオンライントランザクション処理（ＯＬＴＰ：Online Transaction Processing または ＴＰ：Transaction Processing[1]）システムと呼ばれるものである。

トランザクション処理の場合、処理すべき業務内容は決まっており、業務プログラムは利用者が作るのではなくシステム開発者によってあらかじめ用意される。そして、サービスを開始する前にはあらかじめ業務プログラムは起動され、用いるデータベースも直ちにアクセスできるように前処理を行うなど、必要な計算機資源は事前に準備し待機した状態にあるため、処理要求が届くと直ちに処理を行うことができ大量の処理要求を効率的に処理することができる。

---

[1] トランザクション処理は必ずしもオンラインである必要はなく、過去にはオフライン処理の形態も有ったが、現在ではほとんどがオンライン処理されておりトランザクション処理（ＴＰ：Transaction Processing）システムは、オンライントランザクション処理（ＯＬＴＰ：Online Transaction Processing）システムとほとんど同義で用いられており、本論文も以下それに準じる。

以上代表的な計算機の処理形態について簡単に述べたが、その計算機資源の制御方式はそれぞれ大きく異なっており、従って計算機資源を管理、制御するためのＯＳは、それぞれの処理形態に対応して固有のものが必要となる。本研究はそのトランザクション処理用のＯＳに関するものである。

次に、トランザクション制御用ＯＳの開発にあたっての要件について考察すると以下の通りであり、これらが本研究において解決すべき技術課題となる。

（１）多様な業務へ適用可能な標準方式の確立

トランザクション処理システムが対象とする業務はシステム毎に固有で、多様である。しかも利用者にとってみると、システムを導入したり更改した時に、それまでにやってきた業務処理や伝票処理の手順が変更になることには抵抗大きい。また、システムの運用を行う人にとっても、従来の運用方法の継続性への要求が強い。特に既存のシステムと接続しているシステムを更改する場合、今までと方式が変更になると接続相手にも影響を与え最悪の場合は相手側のシステムも変更しないと新システムと接続できないケースも出てくる。このような状況にあるため、同種の機能であってもその実現形態に対する要求はシステム毎に多彩となるなど、多種多様な個別のユーザの要求に応えなければならない。

このため、従来は特に大規模なシステムの開発にあたっては個別的な開発とならざるを得ず、ＯＳにまで応用システム固有の修正を施すなど結局は専用ＯＳの開発となっていた［5］。

しかし、この種のシステムの数が増大する中で、開発期間の短縮や生産性の向上、システム信頼性の向上の面からこのような個別的な開発ではなく、大規模・高トラヒックなトランザクション制御の標準方式を確立し、様々な業務に適用可能な汎用的なトランザクション処理用ＯＳを開発することは重要な課題である。中でも業務処理の骨格をなす電文の流れに関する設計の自由度の確保は、多様なサービス実現のキーとなるもので、電文制御機能の汎用化は重要な研究課題である。

（２）高性能なソフトウェア処理方式

汎用的な方式は、専用に開発されたものに比べると性能上は不利とならざるを得ないが、高性能な制御方式の考案によって可能な限り性能面での改善が必要である。特に、ハードウェア能力を最大限に引き出し得るか否かは、ソフトウェア方式の処理方

式の優劣にかかっており、その方式選択はコストパフォーマンスの改善にとって非常に重要である。

（3）システム固有の処理に適用可能な障害処理方式

トランザクション処理システムは生活を支える基幹システムであるだけにその高信頼化は重要な問題である。システムに故障が発生した場合には、できるだけ速やかに正常な運転を再開することはもちろん、端末との間で送受された電文やデータベースの内容などが破壊されたり、矛盾した状態に放置されることは許されない。例えば、銀行システムにおいて元帳の残高が減額されてしまっているのに、システムの故障によって支払のメッセージが端末に届かず、現金を手にすることができないようなことが生じてはならない。

これを一般的な形で整理すると、次に示すトランザクションの持つべき４つの要件を確実に保証すること[6]ということができる。４つ要件は、それを表す単語の頭文字をとって、ＡＣＩＤ属性と呼ばれる。

すなわち、

Atomicity（原子性）：トランザクションの処理による状態の変化は、変化するか、全くしないかのいずれかであってそれ以外の中間的な状態があってはならない。状態の変化とはデータベースやメモリ上の情報の変更、電文の送信や受信などを指す。

Consistency（一貫性）：トランザクション処理の終了の状況、例えば正常終了や異常終了であっても、状態の変化は正しく、矛盾を起こさないものであること。

Isolation（独立性）：平行して処理される複数のトランザクションが互いに干渉しないという性質。他のトランザクションとの処理の時間的な関係が処理結果になんら影響しないことを要求する。

Durability（持続性）：トランザクションの処理が一旦成功裏に完了したならば、それ以降システムに障害が起こっても、その結果が確実に保証され、トランザクションによる状態変化の効果は持続するという性質である。

トランザクション制御とは、このＡＣＩＤ属性を保証するためのメカニズムであると言うことが出来る。このためにジャーナル処理機能や障害時に情報の復元を可能とするための機能をはじめ様々な機能が実装され、それらを組合わせてそれぞれのシステム固有の条件に合致した障害処理機能を設計することになる。

ところで、これらの障害処理のための機能は、復元に必要な情報を採るために通常の運転中に常に利用されるので、システムの性能に大きな影響を与える。さらに、障

害処理は運用手順などに直接反映されるので、（1）で述べたようにシステムからの個別の要求が集中する部分でもあり、そのような多種多様な要求に応えられるような柔軟性が重要である。すなわち、正常運転時の性能にできるだけ影響を与えず、かつシステム毎の固有の要求に応えられる障害処理方式の実現が課題となる。

1.2.2 トランザクション処理システムのソフトウェア構成

ここでは、トランザクション処理システムのソフトウェア構成について述べ、本研究の成果が主としてどの部分に反映されるのかを示す。

一般に計算機システムのソフトウェアは階層構成を採っている。計算機の適用範囲が広がるにつれ、それぞれの要求条件が大きく異なってくるため、応用システムに最も適合したソフトウェアを選択し組合わせることによって個々の要求条件に適合したソフトウェアシステムを作り出すことがそのねらいである。しかし、その階層構造も最初から整理されたものではなく、当初は各個別システムの専用ソフトウェアとして開発されたものが、経験を積むことによって徐々に機能や構造が洗練され共通的なソフトウェアとして新しい階層に位置づけられるように、常に進化発展するものである。トランザクション処理用のＯＳも正にそのような経過をたどってきている。当初のトランザクション処理システムでは、トランザクション制御用の機能をＯＳを改造してその中に組み込んだり、業務プログラムの中に純粋な業務プログラムと一緒に未分化のまま組み込んでいたものが、トランザクション処理システムに共通的に必要な機能と個別のシステムに固有な機能に徐々に整理され、トランザクション処理モニタ（ＴＰモニタ）と呼ばれる独立したソフトウェアとして構造整備されるに至っている。そして現在では、概ね図 1.1に示すようなソフトウェアの階層構造となっている。

本研究の成果は、主としてこのＴＰモニタに反映されるものであり、具体的にはＤＩＰＳ用ＴＰモニタ[1]に実装し、各システムへ提供した。

ＴＰモニタのソフトウェア体系上の位置付けは、図に示す通り広義のＯＳの一部として、（狭義の）ＯＳに欠けている機能を提供するためのものであり、ＯＳと業務プログラムの隙間を埋めるいわゆるミドルウェアの位置づけになるものである[2]。

---

[1] 当時はまだＴＰモニタという用語も無く、ＤＩＰＳリアルタイムパッケージ等と称していた。

[2] ここまではトランザクション処理用ＯＳという用語を漠然と用いてきたが、それは図 1.1 に示す広義のＯＳを指している。ここで、ＴＰモニタを定義したので、以降では、ＯＳはＴＰモニタを含まない狭義のＯＳを指す用語として用いる。

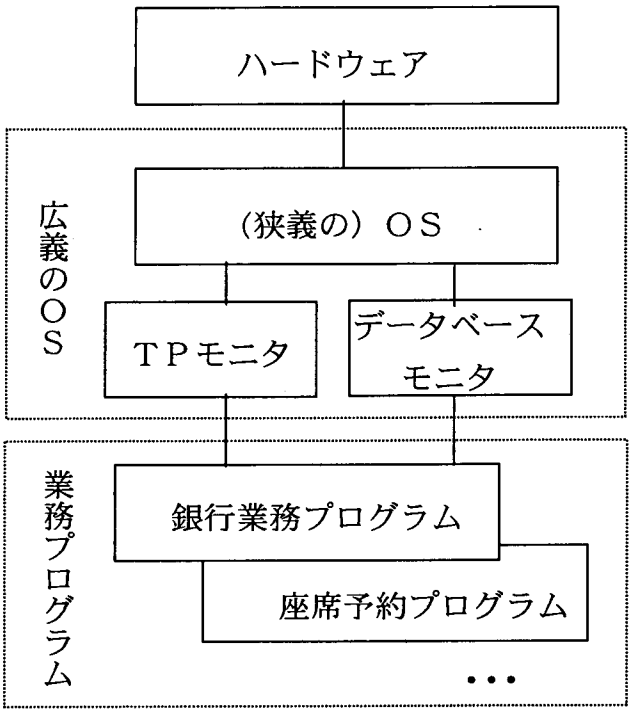


図 1.1 トランザクション処理システムのソフトウェア構成

当時、ＴＰモニタに相当する製品としては、ＩＢＭのＣＩＣＳ[26]，ＩＭＳ[29]があったが、航空会社のオンラインシステムのような大規模なシステムを制御するためには、やはり専用ＯＳに近いアプローチが取られた[5]。

1990年代に入ると、計算機システムのダウンサイジングに合わせて小型システム用の簡単なＴＰモニタが各社から出された。ＵＮＩＸ上でのＴＵＸＥＤＯ[1]、トランザーク社のＥＮＣＩＡ[27]、ＮＣＲ社のＴＯＰ ＥＮＤなどである。

1.2.3 タスクの概念

ここでは、本論の各章で繰り返し取り扱うタスクなる概念について概要を述べる。

タスクとは、高度な多重処理を行うシステムを統一的に制御するために導入されたソフトウェア上の概念である。プロセスという言葉をタスクと同義で用いることもある。タスクは多重処理を行う場合の処理単位であり、計算機資源を割り当てる単位である。たとえば、ＣＰＵ資源はタスクに与えられる、即ち、ＣＰＵを待っているタス

クの中から一つのタスクを選んでＣＰＵを割り当てて使わせる。処理が進んでそれ以上ＣＰＵが不要になったら、そのタスクの代りに別のタスクにＣＰＵを使わせる。このようにＣＰＵのスケジュールはタスクを単位にして統一的に行われる。同様に、例えば作業用のメモリやファイル、データベースもタスクが要求をすることによって、ＯＳがタスクに対して動的に割り当てる。このように、計算機資源の割り当てをタスクという統一した概念で制御することによって、計算機の中で様々な処理が並行して進行する多重処理の制御を整然と行うことができる。

1.3 本論文の概要

本論文は、著者がＮＴＴ電気通信研究所に在職中、および大阪大学大学院工学研究科（通信工学専攻）在学中に行った、トランザクション制御方式と性能評価に関する研究の成果をまとめたものである。

本文は４章より構成され、序論、結論を含め全体としては６章から成っている。なお、前節で述べた本研究の課題が本論文のいずれの章で取上げられているかの関係を図 1.3に示す。

第２章では、トランザクション処理システムの電文制御方式の設計を行う。トランザクション処理システムが多種多様な業務に適用できるためには、業務の骨格を形成する電文の流れに関する設計に大きな自由度が必要である。特に、トランザクション処理システムにおいては、ＴＰモニタがＯＳと業務プログラムとの間に介在し、業務プログラムからの要求に応じて目的とする端末に電文を送信したり、逆に通信網を経由して計算機に送られる電文を対応する業務プログラムへ引渡すなど、電文の送受を制御する。従って、業務の多様性に答えられるためには、このＴＰモニタにおける電文の制御機能の多様なサービスへの適用性の確保が重要である。

本章ではこの課題に対して、まずトランザクション処理システムにおける電文制御の要求条件を分析・体系化し、それらを満足するための電文制御の方式構成を明らかにする。さらに、汎用的な電文制御方式として新しい考え方に基づくゲート制御方式を提案し、電文制御に対する要求の大きく異なる３種類の実サービスへの適用結果を分析し、目標とした汎用性の実現を確認する。

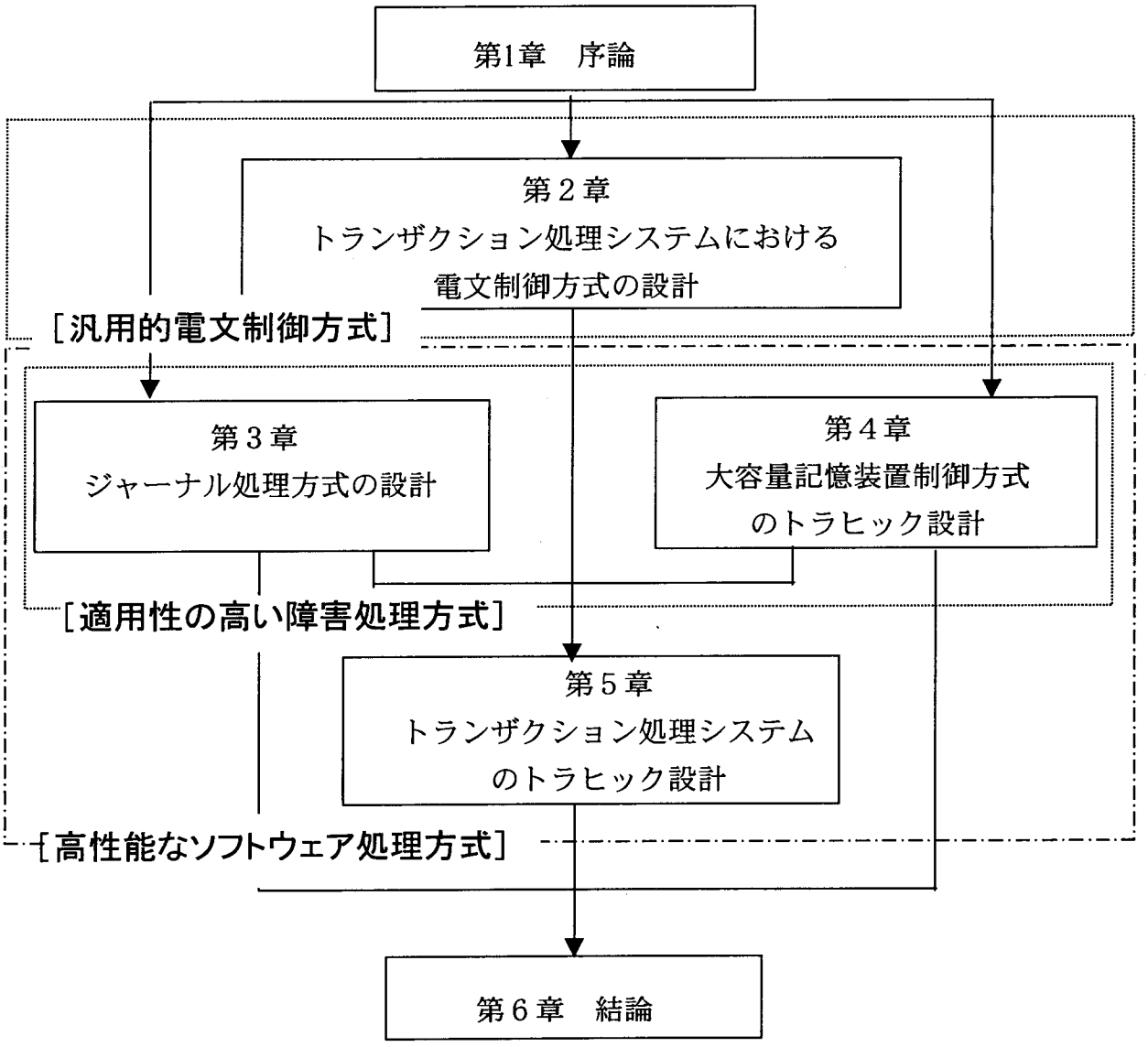


図 1.2　研究課題と論文の構成

第３章では、トランザクション処理システムにおいてシステム全体の性能に大きな影響を与えるジャーナル処理の性能を定量的に評価し、それに基づき高性能化のための新しいジャーナル処理方式を提案する。

トランザクション処理システムに故障が生じた時に、データベースの内容や端末との間で送受された電文の内容を矛盾の無い状態に復元した上でシステムの正常な運転を迅速に再開することが必要である。この目的のために通常の運転の最中に再開処理に必要な情報をすべてのトランザクション毎に時系列順にジャーナルとして記録する

必要がある。このため、ジャーナル処理の性能はシステム全体の性能に大きく影響を及ぼし、その高性能化は重要な要件となる。

本章ではシミュレーションによってジャーナル処理方式の性能評価を行い、性能上の隘路を詳細に分析する。その結果、標準的ジャーナル処理ルーチンでは、高トラヒックになると排他制御による保留時間の増加によって処理時間が増加することを明らかにする。そして、この排他制御を入出力処理割り込みルーチンで実現する新方式を提案する。この提案の高性能ジャーナル処理方式では、標準方式に比べシステム全体の性能を２０％改善する効果をもたらすことをシミュレーションを用いて示す。

第４章では、大容量記憶装置（Mass Storage System : ＭＳＳ）をオンラインシステムにおいて効率的に制御するためのトラヒック設計の方法を提案し、それに基づく設計結果の妥当性、制御方式の有効性を定量的に実証する。

トランザクション処理システムの運転においては、第３章で述べるジャーナルのように処理の履歴情報を格納するファイルやオンラインファイルのバックアップ情報などのように、オンラインでの直接のアクセスは無いが、大容量で且つ長期間保存すべきファイルが必要である。これらについては、オンラインでの応答時間の制約は無いため、通常磁気テープ装置が用いられてきた。しかし、システムの規模の増大に伴い、それらの量も急増し、操作コストの増大、操作誤りなどテープ操作の煩雑さに起因する各種の問題が大きくなる。

一方、トランザクション処理システムに用いられるファイルは、応答時間の条件を満足させるために磁気ディスク装置を用いるのが通例であるが、必ずしも短い応答時間は必要無いが、コストの低いファイルに対するニーズも見られる。

この様な問題解決の可能性を持った装置として開発されたのが大容量記憶装置(ＭＳＳ)である。この装置は自動操作機構をもった磁気テープ装置と磁気ディスク装置とを組み合わせ記憶階層を構成したもので、従来の比較的単純な構成の二次記憶装置に比べ、その特性は複雑な様相を示すことが予想される。

このような装置をシステムへ導入し有効に利用するには、装置の特性を十分に把握しその特性を生かした利用形態と制御方式を実現することが必要である。

本章では、まずＭＳＳの評価モデルとして系内客数制限をもつ開放形の待ち行列ネットワークモデルを提案する。次に、そのモデルの解析の方法を与え、平衡状態における各種特性値を解析する。さらに、系が平衡状態にあるための条件について考察し、解析結果に基づく数値計算例を示すとともに、実測値との比較を行いよい一致が得られることを示し、モデルの妥当性を実証する。

第５章では、ソフトウェア資源とハードウェア資源の関係に代表されるような、資源と資源の間に階層的な関係が存在するシステムの性能評価を行うための一般的なモデルを提案する。本論文の第２章の電文制御方式、第３章ジャーナル処理方式や第４章大容量記憶装置のそれぞれにおいて個別的に示してきたトラヒック設計法をさらに高度化し、より広い設計問題に適用可能なトラヒック設計法に一般化を試みるものである。

トランザクション処理システムにおいては、システムへの処理要求はタスクに一旦割り付けられた後、そのタスクがハードウェアを利用しながら処理を進める形態を採っている。従ってタスクは重要な資源であり、システムの性能に大きな影響を与えるため、その性能評価は重要な課題である。ところで、タスクはソフトウェア上の概念であって、その実体はハードウェアが担っている。従って、評価にあたってはタスク単体ではなく、タスクの実態を担っているハードウェア資源との相互作用を含めて表現できるモデルが必要である。従来から待ち行列モデルを計算機システムの性能評価へ適用する研究は数多く報告されているが、これらの研究はＣＰＵや入出力装置などハードウェアのみを単独で取り扱うものあるいはソフトウェア資源を扱ってもそれを単独で取り扱うモデルが大半で、ハードウェアとソフトウェアの相互関係を同時に評価できるモデルではない。

このような要求に応えるものとして本章で提案する方式は、ハードウェアを窓口とする待ち行列モデルにおいて、その一部または全体の中に存在し得る客数を一定数に制限することによってタスクの存在を表現する系内客数制限モデルである。本章の前半では、外からの客の到着が無い閉じたモデルを用いてタスク構成を評価する方法を提案し、その解析例により本モデルが前述の要求に応え得る有効なものであることを示す。さらに、後半ではそのモデルを発展させ、ソフトウェア自身に対する競合や輻輳を、ハードウェア資源と関係付けて総合的に取扱うための一般的手法を提案する。本モデルは、本章前半で取り扱うタスク構成の問題や第３章のジャーナル処理ルーチンの問題、さらに第４章で対象としたＭＳＳもここでの問題の一つの具体例に当たることを指摘する。本モデルの解析の方法を与え、系が安定であるための必要十分条件を明らかにし、理論的な限界値を与える。また、数値計算の手順を与える。

第６章では、本研究で得られた成果を総括すると共に、その意義および今後の課題を述べる。

第2章

トランザクション処理システムにおける電文制御方式の設計

2.1 緒言

本章では、トランザクション処理システムにおいて業務処理設計の鍵となる電文制御機能について、様々な業務に対する適用性の高い標準方式の設計について論ずる。

トランザクション処理システムが社会経済活動の様々な局面に導入されるに伴い、多種多様な業務処理が容易にかつ効率よくシステム化できることが強く求められる。ところで、業務処理の骨格は伝票の流れで規定されると言っても過言ではなく、システム化にあたってはその伝票の流れ、すなわち電文の流れが自由に設計可能であることが多様な業務処理への適用性を決定する鍵となる。トランザクション処理システムにおいては、ＴＰモニタがＯＳと業務プログラムとの間に介在し、業務プログラムからの要求に応じて目的とする端末に電文を送信したり、逆に通信網を経由して計算機に送られる電文を対応する業務プログラムへ引渡すなど、電文の流れを制御する。従って、業務の多様性に答えられるためには、このＴＰモニタにおける電文の制御機能の多様なサービスへの適用性の確保が重要である。

本章ではこの課題に対して、トランザクション処理システムにおける電文制御の要求条件を分析・体系化し、それらを満足するための電文制御方式の設計を行う。さらに、汎用的な電文制御方式として新たな考え方に基くゲート制御方式を考案し、そのサービスへの適用性に関する評価を行い、設計の妥当性を確認する。

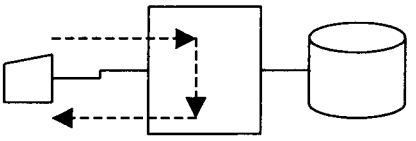
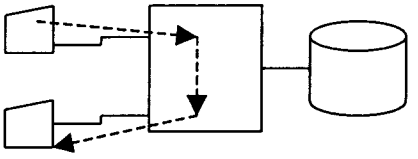
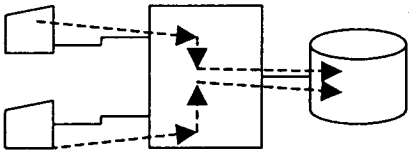
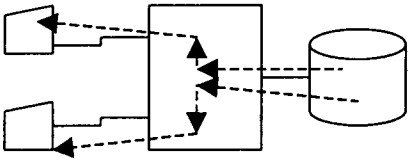
2.2 トランザクション処理における電文制御の要求条件

本節では、個々の具体的なサービスシステムの特性分析を行い、それらを体系化することによって、トランザクション処理システムにおける電文制御が具備すべき方式上の条件を明らかにする。その結果を、データ通信システムの一方の代表的な形態であるタイムシェアリングシステム（タイムシェアリングシステム; Time Sharing System）と対比させ、トランザクション処理システム固有の電文制御方式の設計の必要性について明らかにする。

2.2.1 電文の流れの多様性

タイムシェアリングシステムでは、電文を端末から入力しセンタでの処理後、その結果を端末で受信し、その結果に基づいて再び電文を入力する会話型処理の繰返しであり、電文の出力端末は入力端末と同一である。これに対し、トランザクション処理システムでは、会話型処理に近い一問一答形式の照会型処理の他に、入力電文を出力電文を待たずに次々に投入するように、電文の入力操作と出力操作を互いに独立に進められる形式が存在する。この場合、入力電文に対応した出力電文の宛先は、必ずしも入力端末とはならないのが一般的であり、交換型処理と呼ばれるものがその代表例である。しかも、これらの電文の流れがいずれかに固定されるものではなく、たとえば、入力された電文の種類によって切替るなど多様であると同時に動的であることが大きな特徴である[69]。トランザクション処理システムにおける代表的な電文の流れとその特徴を表2.1に示す。

表 2.1 トランザクション処理システムにおける電文処理の流れ

種類	電文の流れ	説明
照会型		・端末から入力された電文は、処理後同一端末に出力される一問一答形式 ・入力から出力の間、他の電文の入力・出力は許さない
交換型		・入力端末と出力端末が異なる。他のトランザクションとの競合があり出力端末の状況によって出力の待ち合わせが必要。 ・入力端末側は処理の完了を確認できない。 ・出力側からの要求に対する再送業務を伴うことが多い。
集信型		・多くの端末から入力された電文をセンタファイルに集める。 ・通常、配信型とセットになる。
配信型		・センタで入力された電文の処理結果を端末へ配信する。 ・出力トラヒックが集中し送出待ちが起こる。 ・通常、集信型とセットになる。

2.2.2　端末および端末制御機能の多様化

トランザクション処理システムでは、一般にその業務自体は定形化されており、タイムシェアリングシステムに比べると処理内容の自由度は少ない。したがってその入出力データも、あらかじめ一定の形式に設計され端末における入出力には専用の帳票を用いるのが一般的である。たとえば、販売在庫管理システムにおいては売上伝票を、また銀行システムでは顧客の預金通帳をそれぞれ直接端末にセットし用いる例がそれである。

これらの背景からトランザクション処理システムにおいては、業務に適合した専用端末をサービスごとに設計し導入する傾向にあり、端末の多様化は今後ますます進展すると考えられる。また、オンラインシステムの導入の大きな動機の一つである「入力データを発生地点から直接入力し、出力データを使用する地点へ直接送る」という形態が進めば進むほど、端末の操作は、訓練された専門のオペレータではなく、比較的慣れない一般多数のオペレータに広がり、端末の操作性向上が重要となってくる。このため、端末オペレータに対して次のオペレーションを誘導するための表示機能を設けたり、誤操作に対しては警報を発するとともに、正しい操作に誘導するなど、オペレーションの簡略化のための様々な工夫がなされる。これらは、タイムシェアリングシステムについても同様の事情であるが、タイムシェアリングシステムの場合は、オペレータに対する誘導はすべてメッセージで可能であるのに対し、トランザクション処理システムの場合メッセージの出力は、帳票を汚損することになるため、たとえばランプによる表示、ブザー、ベルの鳴動、キーボードのロックなどの方法をとる必要がある点に特徴がある。このため、トランザクション処理システムにおける端末は単にデータの入出力を行う機能のみではなく、ランプやブザーなど付属装置の機能が強化される傾向にある。これもトランザクション処理システムにおいて、端末の多様化を促進する大きなインパクトとなっている[64]。

従って、トランザクション処理システムにおいては、新規端末のシステムへの追加が容易に可能で、端末仕様の変更にも即応できることおよび端末の持つ様々な付属装置の制御が容易であることが重要である。

2.2.3　端末通信処理の効率化

トランザクション処理システムにおいては、システムへの処理要求は大量に発生するが、個々のトランザクションの処理は定形化された比較的単純なものであることが

多い。したがって、一つのトランザクションの処理に要する実行命令ステップ数（DS: Dynamic Step）のうち、端末交信のための実行命令ステップ数の占める割合がタイムシェアリングシステムに比べ相対的に高くなる傾向にある。例えば、タイムシェリングシステムの代表例である科学技術計算サービスでは、端末交信処理の実行命令ステップ数が全体に占める割合は15％－20％であるのに対し、トランザクション処理システムにおいては、照会型処理で45％－50％、交換型処理では50％－55％に達している[65]。このため、トランザクション処理システムでは端末との交信機能の性能改善がシステム全体の性能の改善に大きく寄与する[74]。

以上、トランザクション処理システムの特徴の分析と考察をもとに、電文制御機能に要求される設計条件を整理すると次のとおりとなる。

（1） 多様な電文の流れを効率よく制御できる方式の確立
（2） 端末の多様化と端末制御機能の高度化に即応できる柔軟性、拡張性の確保
（3） 端末交信処理の高効率化

2.3 電文制御機能の方式設計

前節で述べた要求条件に基づき電文制御の方式設計を行う。

2.3.1 端末交信処理と業務処理の分離

会話型処理では、応答電文の出力後、次の入力電文が投入されるまでには、端末オペレータの思考や打鍵のための時間が必要であり、それに要する時間は、センタ内処理に要する時間に比べ格段に大きい。タイムシェアリングシステムにおいては、この時間の差を利用し、端末との交信時間中には他の端末に対するセンタ処理を行う多重並行処理方式を採ることによって多数利用者の共同利用を実現している。たとえば、図2.1に示すとおり端末Ａへの電文出力中あるいはその後の電文入力待ちの間は、それ以外の端末Ｂ、端末Ｃのための処理を行う。この間、端末Ａに対して入力電文待ちとなるため、センタの資源を無効に保留しないよう、主記憶上のプログラム、データは高速の２次記憶装置に退避するのが通常である。これはある端末に着目したとき、電文の入力、センタ処理および電文出力が、この順に繰返し直列に進むという会話型電文の流れの特性に依存している。

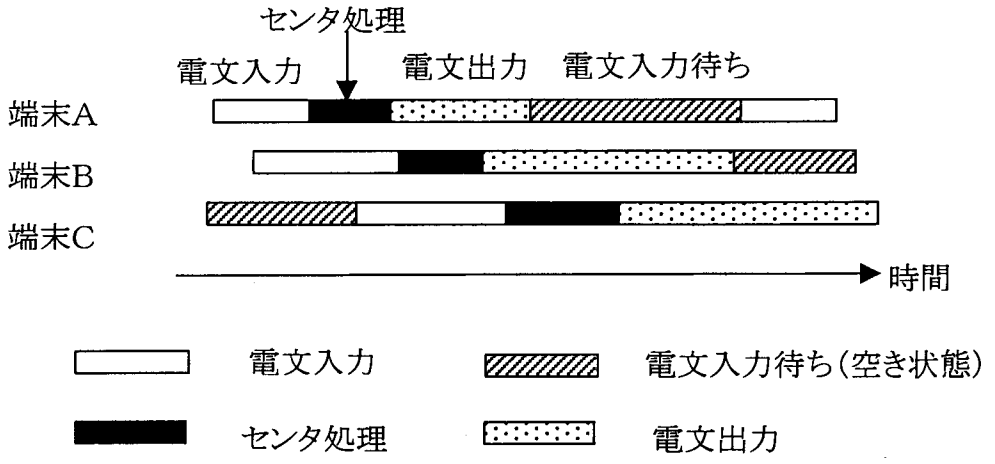


図 2.1　タイムシェアリングシステムにおける並行処理の概念

ところで、トランザクション処理システムでは会話型処理に近い照会型処理と呼ばれる電文の流れの他に、為替交換業務や集配信業務に見られるように、処理結果の出力と入力が独立に並行して行われる電文の流れが存在する。すなわち、図2.2に示すように、端末Aから入力されたデータを処理中に、それと平行して端末Aに対して電文を出力したり（図2.2の区間Ⅰ）、端末Bからの入力を処理中に、端末Bからの次の電文を入力する（図2.2の区間Ⅱ）などがそれである。この場合は前述の会話型処理とは異なり、ある端末から入力されたデータの処理を、同一端末から次の電文の入力処理

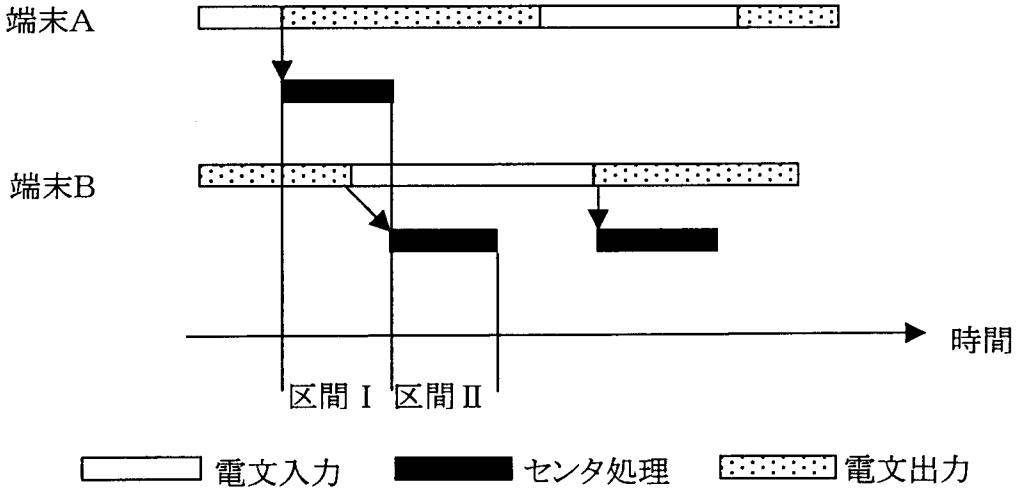


図 2.2　トランザクション処理システムにおける並行処理の概念

または、出力処理と平行して実行してもなんら差し支えがなく、しかもそれを可能とすることによって、同一端末から多数のトラヒックを高速に処理できることになる。

以上に述べたように、トランザクション処理システムの持つ多様な電文の流れを、統一された方式で効率よく処理するには、端末交信処理と業務処理とが並行に可能となる構成をとる必要がある。このために、端末交信処理と業務処理のそれぞれを並行処理の単位であるタスクに別々に割り付けることによって、両者が独立して並行に処理できる方式とする。すなわち、端末との電文の交信を専門に処理するサービス管理タスク（以下SMT、Service Main Task）と、入力された電文に対応する業務処理を行い出力電文を作り上げるまでの処理を行うサービス処理タスク（以下SPT、Service Processing Task）とに分離し、これらを並行に動作させる方式とする。これにより、図2.2に示すような電文の流れの設計が可能となる。

2.3.2　電文待ち行列と電文スケジュール機能

(1)　入力電文待ち行列

タスクを新たに生成し、初期設定を行う処理には時間が必要なため、応答時間を重視するトランザクション処理システムでは、サービス開始時にタスクをあらかじめ固定した数だけ生成し、サービス中はその数を増減させない方式とする。このため、トラヒックの状態によっては、データ入力時にそれを処理すべき空のＳＰＴがない場合が発生し、処理を待ち合わせることになる。このため、入力電文のＳＰＴ待ち行列を設けることとする。

これに対し、一般に会話型処理では、ある端末からシステムとの会話要求があると、その端末用の処理タスク（トランザクション処理システムのＳＰＴに相当）を生成し、会話の終了で消滅する方式を採っており、入力電文の処理待ち行列という概念はない。このようなタイムシェアリング方式で用いられている方式を採用しない理由は次のとおりである。すなわち、会話型処理における処理タスクは、会話の開始から終了間までの間、一つの端末に専有されその保留時間は数分から数十分のオーダとなる。一方、トランザクション処理システムでは前述したように、端末との交信処理をＳＭＴの分担としているため、ＳＰＴの保留時間は、ＣＰＵ処理時間と磁気ディスクなどの２次記憶装置へのアクセス時間などで、数msから数十msの非常に短い時間となる。したがって、これにタスクの生成や初期設定の処理が毎回加わるとオーバーヘッドの比率が非常に大きくなる。これに対し、トラヒック量に見合った適当なＳＰＴ数を決めることにより、空ＳＰＴを待つ時間が応答時間にほとんど影響を与えなくすることが可能であり、タスク数を固定する方式とした。なお、タスクの数はシステム設計上の重要

なパラメータとなる。その決定の方法については、本論文の第５章で扱う。

（2）出力電文待ち行列

ＳＰＴの数が固定であることに起因して、入力電文がＳＰＴの空きを待つための入力電文待ち行列が必要となったが、一方出力電文については、同一端末に対する出力電文のぶつかりの発生を制御するために、やはり待ち行列が必要となる。

出力電文の衝突は次の２つのケースに大別できる。

（a）出力電文相互の衝突

同一端末に対する出力電文が同時に２通以上発生する場合である。これは、互いに独立な入力データを、異なったＳＰＴが並行して処理した結果、同一端末に対する応答電文を生成した場合などに発生する。

（b）入力電文と出力電文の衝突

出力電文が生成された時点に、その宛先端末から電文を受信中であったケースである。この場合、入力、出力のいずれを優先するかは、サービス仕様に依存し、後述の電文スケジューラが選択を行うものであるが、入力を優先する場合には、その間出力を待ち合わせる必要がある。

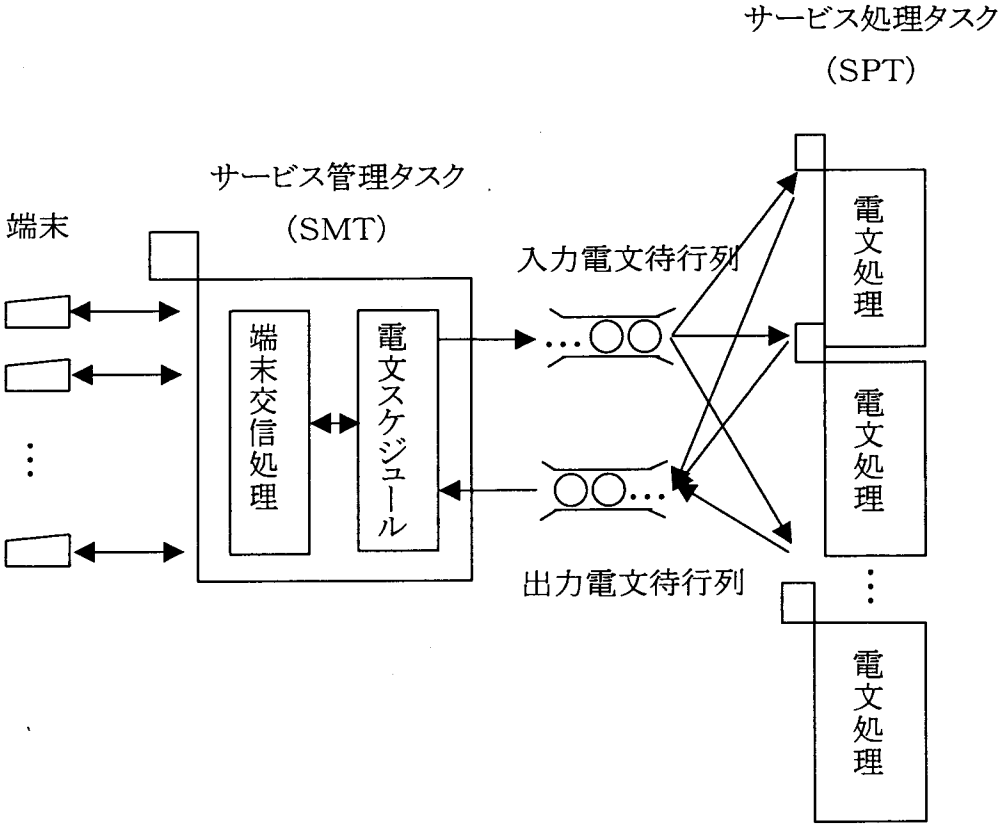


図 2.3　トランザクション処理システムの電文制御構成

会話型処理においては、一つの端末に対する処理は、一つの処理タスクにくくりつけられているため、このような電文の衝突は発生せず、当然電文の出力待ち行列なる概念は存在しない。

以上述べたトランザクション処理システムの電文制御の方式設計結果を、図2.3に概念的に示し、特徴的な事項について会話型システムとの対比を表2.2に掲げる。

表 2.2　電文制御方式の比較

項目	タイムシェアリングシステム	トランザクション処理システム
端末交信処理と業務処理	直列	並列
端末と処理タスクの関係	固定	不定
電文待ち行列	なし	あり

2.4 電文スケジュール機能の設計

ＳＭＴ、ＳＰＴのタスク数を固定して、その間に入力電文および出力電文の待ち行列を介在させた場合には、タスクに空きが出来た時に待ち行列中の電文をどのような戦略に基づいて選択し、どのタスクに割り付けるべきかを決定する電文のスケジュール機能が新たに必要となる。例えば、端末が空きになった場合、送信を行うか交信を行うのか、また送信を行うなら、待ち行列中の電文の中からどの電文を選択するかなどの決定機能が必要である。この決定論理はサービスの種類によって大きく異なるため、この部分をサービス仕様にしたがって任意に設計できるよう、十分な自由度を持つことが、サービスへの適用性を左右する重要なポイントの一つである[40]。
このため、電文スケジュール機能について徹底した汎用化を設計目標とし、ゲート制御と称する新たな手法を開発した。以下その設計の考え方と設計の詳細について述べる。

2.4.1 汎用化の考え方

汎用性を高めるには、種々の方法が考えられる。たとえば、考えられるすべての機能を用意しておき、サービス設計者が適合するものを選択して使用する方法、多様化の可能性を持つ部分にサービス設計者の作成したプログラムを組込むことを可能とする方法などが考えられる。これに対しここで採った方式は、メカニズムとポリシを分離する[23]、[24]という考え方に立った汎用化へのアプローチである。

すなわち、端末とセンタとの間の電文の通路に論理的なゲート（遮断機）を設け、ゲートの開閉に応じて電文を通す、または通さないの制御をするメカニズムのみをシステムに準備し、一方、そのゲートをどのように開閉するかというポリシは、サービス設計者の手にゲート設計として委ねるものである。このため、サービスごとに多様なスケジュールが単一機構で実現でき、また、その機構が非常に単純であるため効率のよいスケジュールが可能となる。

2.4.2 電文スケジュール機能の抽象化

電文スケジュール機能とは、端末に対し、(1)送信する、(2)受信する、(3)いずれも行わないのどれかを決定することおよび送信の場合には、どの電文を送信するかを決定することにある。ところで、この決定のアルゴリズムはサービス仕様により定まるものであり、多様である。一例として、以下に照会型処理と交換型処理を行うサービスにおける電文スケジュールを示す。

いま、ある端末に対し、照会型処理を実行中であるとする。この場合、入力電文に対する応答電文を送出し終わるまでの間、その端末に対しては一切の電文の出力を抑止する必要がある。これを制御するため、電文スケジュール・ルーチンはその端末が照会モードにあることを記録し管理する。このような状態で、その端末に対する交換電文が発生し、その送信依頼があると相手端末が照会モードにあるため、端末が空であっても送信は抑止され電文は出力電文待ち行列へつながれる。

次に、応答電文の送信依頼があると、その電文は端末のモードと合致するため送信が行われる。つづいて、その電文の端末への正常な送信完了が報告されると、一連の照会型処理が完了したため、端末の紹介モードが解除され、同時に出力待ち行列の状態が調べられ、上述の交換電文が送信スケジュールされることになる。

以上は、あるサービスにおける一例に過ぎないが、各種のサービスにおける電文スケジュールを調査、整理した結果、スケジュールを支配する要因を以下の３要素に整

理・体系化した。

（1）スケジュール契機となった事象（スケジュール・トリガ）
上記の例では、業務プログラムからの電文送信依頼や電文送信の正常終了が相当する。他に、電文の受信完了や端末クイット、端末起呼の発生、送受信異常の発生など様々である。

（2）端末の状態（端末モード）
端末で実施している業務内容（業務モード）や、端末にセットされている帳票の種類（帳票モード）などからサービスごとに定められるもので、上記の場合照会モードがその一例である。

（3）出力電文待ち行列の状態
出力待ち電文の有無や、待ち電文の電文種別、プライオリティなどにより定まる。

電文スケジューラは、スケジュール・トリガを契機に、端末モードおよび待ち行列の状態を新たなモードと状態に遷移させ、その新しい状態とトリガから一定の論理で決定を行う。その結果に基づき、端末に対し動作指示を行い、一連のスケジュールを終了し、次のスケジュール・トリガを待つ。

以上の電文スケジューラは、次のようにオートマトンSに模式化できる。
すなわち、$S = (K, \Sigma, \Delta, \delta, \omega)$
ただし、K：状態の集合。ここでは端末モードと出力電文ち行列の状態の組合わせ。
Σ：入力記号の集合。ここではトリガの集合である。
Δ：出力記号の集合。ここでは、{入力指示，出力指示，指示無し}である。
δ：$K \times \Sigma \rightarrow K$を示す状態遷移関数。
ω：$K \times \Sigma \rightarrow \Delta$を示す出力関数。

状態推移関数δおよび出力関数ωは、各サービスの仕様に大きく依存する部分であるにもかかわらず、従来はこれらを直接プログラム論理に組込んで提供していたため、サービスに対する適用性を狭くしていた。
これに対し、ここでは以下の方針を採った。
(イ) δおよびωを外部から与え得る構成とする
(ロ) KおよびΣについては、インタフェース上必要最小限の規則のみを定め、その範囲内でサービス側が任意に定義可能な構成とする。

(ハ) システムとして提供する機能は、外部から与えられた、δ, ω, K, Σに基づいて出力Δを決定し、端末に対し動作指示を行う機構のみとする。

上記(ハ)に対応した機構をゲート制御と呼ぶ。各サービス用の電文スケジューラは、このゲート制御機構に対し、そのパラメータであるδ, ω, K, Σをサービス設計者が与えることにより実現される。

2.4.3 ゲート制御の機能と構成

以下では、上記の考え方に基づき設計したゲート制御の具体的な設計結果について述べる。

（1）スケジュールモード

トランザクション処理システムにおける電文の流れは、入力と出力が非同期に行われるものと、入力と出力とがなんらかの対応をとって同期的に行われるものに大別される。したがって、電文スケジュールにもこれに対応した機能が必要である。このため、ゲート制御では、スケジュールモードなる概念を導入し制御を行う。すなわち、入出力のスケジュールを互いに独立に行う非同期（スケジュール）モードと、入出力電文を相互に関係付けてスケジュールする同期（スケジュール）モードの２種を設ける。このスケジュールモードは、実施するサービスの内容によって端末ごとにサービス設計者があらかじめ指示するもので、途中での動作的な切替えも可能としている。

（2）端末モードの定義

各端末ごとに、サービス側で端末モードを表現するために、ゲートと称する32ビットのフィールドを設ける（ゲートを構成する各1ビットをゲート素子と呼ぶ）。各サービスにおける業務モードや帳票モードをこれらの各ゲート素子にどのように対応付けるかは、サービス設計者側の任意である。一方、ゲート制御では、サービス設計者が、ゲートの各ビットまたはビットパターンに与えた端末モードとしての意味付けには一切関与せず、各ビットを独立に、かつ一律に次のように解釈し、制御を行う。

- すべてのゲート素子がON（ゲート開）の場合には、電文通過を許可する。
- 一つでもゲート素子がOFF（デート閉）の場合には、電文通過を禁止する。

なお、本ゲートは16ビットずつ入力ゲートと出力ゲートに分離されており、前者は電文入力を、後者は電文出力をそれぞれ支配する。

（3）電文種別の定義

出力電文には、電文ごとにゲートキーと称する16ビットのフィールドが用意されている。このゲートキーの各ビットは出力ゲートを構成するゲート素子に１対１に対応し、ゲートキーがONのときは、対応するゲートの閉の状態を無視して出力することを意味し、OFFのときは対応ゲートの状態により出力が制御されることを意味している。つまり、ゲートは錠、ゲートキーは鍵に対応し、ゲート制御はゲート（錠）を通過できる（開けられる）ゲートキー（鍵）を持つ電文を選んで送出することになる。

サービス側では、このゲートキーとゲートをあわせて設計することにより、電文間に優先順位を付したり、特定の端末モードの時には出力しない電文や、逆にそのときだけ出力可能な電文など、様々な電文種別を設計可能となる。

以上（1）、（2）、（3）で述べたスケジュールモード、ゲートおよびゲートキーがスケジューラSの状態Kを定義する要素である。

（4）トリガ種別の定義

電文スケジュール契機となる事象には、すでに例示したように、様々なものが考えられるが、どの事象を電文スケジュール契機とするかは、各サービスによって異なっている。これに対応するため、発生し得る事象ごとにそれを電文スケジュール契機とするか否かが、あらかじめ制御テーブル上に登録可能な構成としている。
ゲート制御では、ある事象が発生したとき、その制御テーブルを参照し、電文スケジューラSへの入力集合Σを、サービスごとに定義可能としている。

（5）状態の遷移

トリガがゲート制御に入力されると、まず状態を新状態へ遷移させる。どの状態へ遷移させるかは、状態Kとトリガ種別Σにより以下の手順で定める。すなわち、まずスケジュールモードをチェックし、同期モードならば入出力両ゲートを新状態へ遷移させる。これは、同期モードの定義から明らかなようにたとえば、入力側での事象の発生が出力側スケジュールの契機となり得るようなスケジュールを可能とするためである。一方、非同期モードならば、トリガが入力関連の場合は入力ゲートを、出力関連の場合は出力ゲートをそれぞれ新ゲート状態へ遷移させる。遷移させるべき新ゲートは、旧ゲート状態とトリガ種別から定める必要があるが、このためにトリガ種別対応にゲート操作パターンとゲート操作マスクをサービス設計者が与えることとし、ゲート制御では次の操作により、新しいゲート状態を作り出す。

$$G_N = (G_O \cap \overline{G_M}) \cup (G_H \cap G_M)$$

ここで、G_N：新たに遷移するゲート状態　G_O：現在のゲート状態

G_M：ゲート操作マスク　G_H：ゲート操作パターン

つまり、ゲート操作マスクでマスクされた部分は、旧ゲートの状態を引継ぎ、他の部分は、ゲート操作パターンにしたがって更新することにより新しいゲート状態に遷移させる。なお、トリガ種別によっては、ゲート状態のみならず。スケジュールモードも変更することが可能であり、たとえば端末クイットを契機に、交換業務から照会業務に切替えるなどが容易に可能である。

（6）出力の決定

出力Δは（5）で述べた状態遷移後の新しい状態Kおよびトリガ種別Σから次のように決定する。すなわち、状態の遷移の場合と同様スケジュールモードが同期モードの場合は、入力および出力の双方を、非同期モードならば入力あるいは出力のうち、状態が遷移した方のみをスケジュール対象とする。

入力側については、入力ゲートのゲート素子のすべてがONならば入力指示、出力側については出力ゲートを満足するゲートキーを持つ電文があれば出力指示、いずれの条件も満足しない場合は入出力いずれも行わないことが指示される。

ゲート制御の概念をより明確にするため、以上で述べた各機能の相互関係を、制御の流れに着目して模式的に示したものが図2.4である。

2.4.4　ゲート制御のサービス適用性

ゲート制御はすでに述べたとおり、各種システムにおける多様な電文の流れを、一つのメカニズムで汎用的に制御できることを最大のねらいとしている。これを評価するため、本節では具体的にこのゲート制御を適用したシステムの中から代表的な３つの大規模なトランザクション処理システムにおけるゲート制御の適用方式について述べる。ここで例示する３つのシステムについて電文スケジュールの観点から見た特徴を表2.3に示す。

Ａサービスは、他のサービスに比べ電文の流れは比較的単純であるが、高トラヒックであるため回線や端末の使用効率を高めるスケジュールが必要である。

Ｂサービスは、照会型と交換型の電文の流れが存在し、端末からの入力またはセンタからの出力を契機に、これらの流れがいずれか一方に切替えられる。また、端末オ

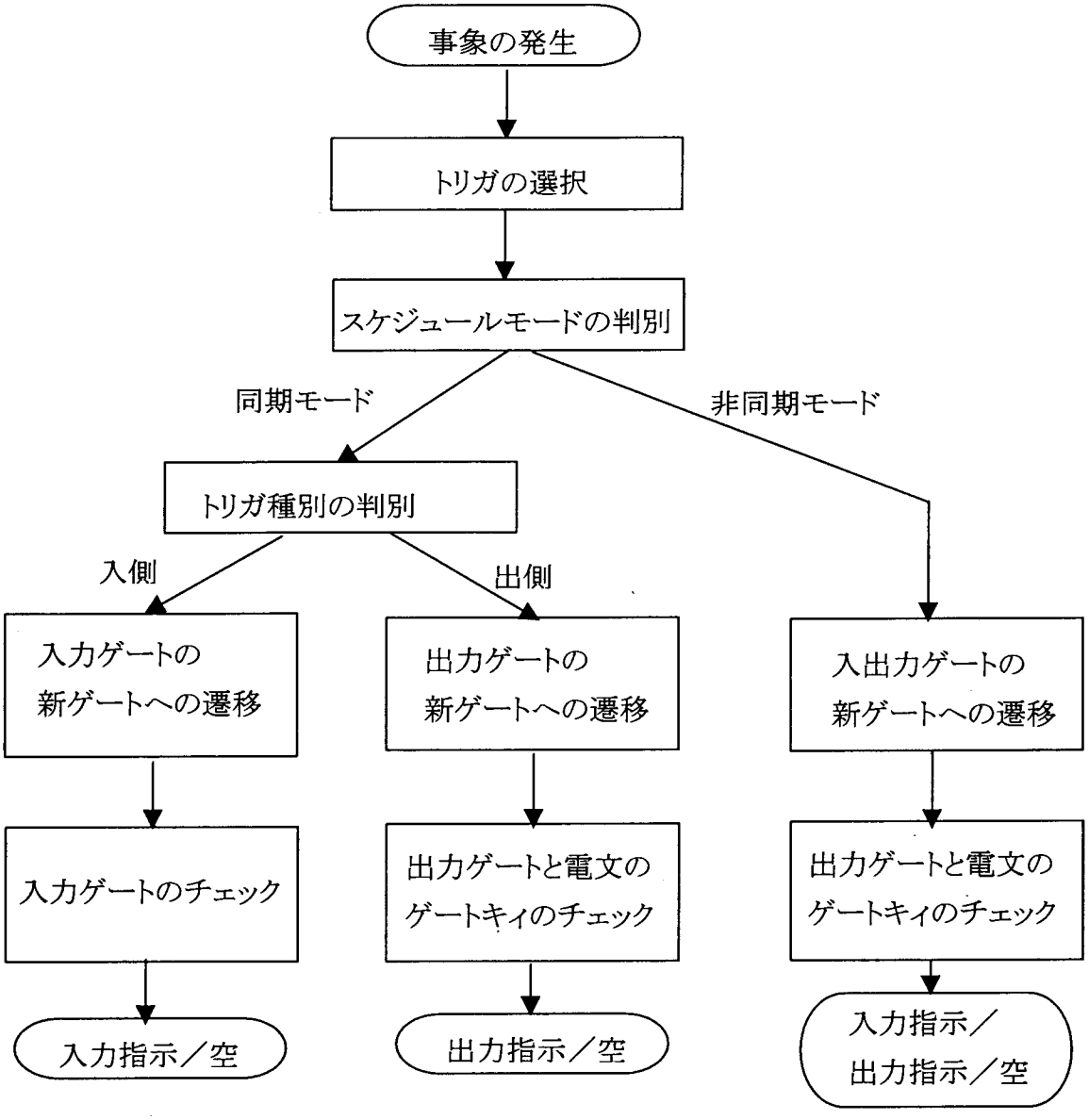


図 2.4　ゲート制御概念図

ペレータが、端末の帳票を切替える場合、その完了とセンタ側からの電文送信の同期をとるために、端末クイットを用いる。このため電文スケジュール契機として端末クイットが重要な意味を持っている点は、他のサービスに見られない特徴である。

Cサービスは電文の流れから見ると交換型に近いものであるが、通常の交換型のように各電文内に宛先が指示されているのではなく、あらかじめ入力端末群と出力端末

表2.3　各サービスにおける電文スケジュールの特徴

サービス	主たる業務	電文スケジュール上での特徴
Aサービス	為替交換業務	・交換型が中心。 ・端末モードの切り替えはセンタが制御する。 ・高トラヒックである。
Bサービス	銀行業務	・問い合わせ型、交換型の混合。 ・端末モードは、電文の入出力と同期して切り替える。
Cサービス	自動車検査証登録業務	・照会型処理であるが、電文の入力端末とその応答の出力端末は異なる。 ・電文の入出力が非同期に連続して行われる。 ・端末モードは、端末オペレータからの指示により切り替えられる。

群の組合せが、端末オペレータにより指示されており、その範囲の中で出力端末を選んで応答を送信する点に特徴がある。また、端末オペレータからの指示にしたがって、端末モードを変更する必要がある。

このように3つのサービスの電文スケジュールは、それぞれ全く異なった制御を必要とするが、ゲート制御を適用するとともにゲート制御の各種パラメータを表2.4に示すように設計することにより、個々のサービスごとにスケジュール・ルーチンを作成することなく、必要なスケジュール機能を実現可能となった。これにより、ゲート制御の広いサービス適用性が確認できた。

表2.4　ゲート制御パラメータの設計例

サービス	スケジュールモード	ゲートの種類	ゲートキーの種類	使用するゲート素子
Aサービス	非同期モード	８種	１１種	１０
Bサービス	同期モード	１０種	２０種	１３
Cサービス	非同期モード	５種	１３種	６

2.5 結言

本章では、トランザクション処理システムにおいて業務処理設計の鍵となる電文制御機能について、様々な業務に対する適用性の高い標準方式の設計について明らかにした。

まず、トランザクション処理システムにおける電文制御の要求条件を分析・体系化した。次に、それらを満足するための電文制御の方式設計を行い、端末との交信処理と業務処理を分離してそれぞれを別のタスクに割り当て、タスクの間に電文の待ち行列を介在させる方式を提案した。さらに、汎用的な電文制御方式としてポリシーとメカニズムを分離する考え方に基くゲート制御方式を考案、設計を行いＤＩＰＳのＴＰモニタに実装した。そして、多くの業務処理に適用され、商用システムとして多様な電文処理の実現が確認された。これにより、多様な業務への広い適用性を持つ標準電文制御方式の確立という当初の目標の達成が実証できた。

第3章

ジャーナル処理方式の設計

3.1 緒言

本章では、トランザクション処理システムの高信頼化を支えるために重要な役割を果たすジャーナル処理方式についての設計を行い、従来の方式に比べ高性能なジャーナル処理方式を提案する。

トランザクション処理システムは企業や社会の様々な分野の活動を支える基幹システムとして、無くてはならない役割を果たしているが、生活を支える基幹システムであるだけにその高信頼化は重要な問題である。システムに故障が発生した場合には、できるだけ速やかに正常な運転を再開することはもちろん、端末との間で送受された電文やデータベースの内容などが相互に矛盾の無い状態に復元され、再開後の正常な処理が保証されなければならない。このための対策の基本は、トランザクション処理の履歴を常に外部記憶装置に記録（ジャーナル）しておき、システムが異常に停止した場合にはその履歴情報をたどることによって、破壊されている可能性のあるデータを正常な状態に復元して（復旧処理）処理の再開を行う方式である[7]，[35]。そして、この復旧処理の基礎となるのは、情報の復元に必要なデータすなわち入力電文、出力電文、データベースの更新前後情報などをトランザクション毎に時系列順に記録するジャーナル機能である。

ジャーナルの情報は、障害が発生しシステムが異常な状態で中断した場合にのみ必要なものであるが、そのような事態に備えて通常の処理の中で常にジャーナル情報を記録しておかなければならない。従って、ジャーナル情報を記録するためのジャーナル機能はできるだけ高性能で通常の処理への影響の少ないものであることが望ましい。

システムの規模が増大しトラヒックが高くなるに伴い、それまで余り問題にならなかったジャーナル機能の性能がシステム全体の性能を左右するケースが現れるようになってきた。通常はハードウェア、特にジャーナル情報を記録する二次記憶装置をより性能の高いものに置換することで問題を回避することが多いが、この対策はシステムのコストパフォーマンスを悪化させることになる。

このような問題認識のもとで、本章では高トラヒックなトランザクション処理システム向けのジャーナル機能について、ソフトウェア制御の最適化による高性能ジャーナル方式の設計について論ずる。すなわち、まずトランザクション処理システムにおける障害復旧方式を概説し、その中で重要な位置を占めるジャーナル機能の設計条件を明らかにする。次に、これまで一般的に用いられてきたジャーナル処理方式がシステム全体の性能に及ぼす影響をシミュレーションにより定量的に評価する。その結果、明らかになったソフトウェア制御上でのボトルネックを解消する新たな制御方式を提案する。そして、この新方式はこれまでの一般的なジャーナル処理方式に比べ２０％以上の性能改善効果が得られることをシミュレーションによって示す。

3.2 障害復旧処理とジャーナル処理方式

3.2.1 障害復旧処理

トランザクション処理システムにおいては、通常多くのトランザクションが並列に処理されている。ある時点でハードウェア障害やソフトウェアのバグ、オペレーションミスといった原因で、システムが予期しない停止をすると、その時システム内で処理中であった複数のトランザクションは処理が中断される。

各トランザクションの処理の進行状況は各々違うために、中断時の状況も当然個々のトランザクション毎に異なる。例えば、図3.1に示すトランザクションＡはファイルの更新が終わった後で出力電文を端末に送信する前に中断され、トランザクションＢはファイルの更新の途中で中断が生じたため、該当のデータはその内容が破壊されている可能性がある。

このようなシステムの予期しない停止が発生した場合、速やかに正常な運転を再開する必要があるが、その際に端末との間で送受された電文やファイルの内容が破壊されたり、矛盾した状態に放置されることは許されない。先のトランザクションＡの例では、ファイルが更新されているにもかかわらず、電文が端末に届いていない状態であるため、このまま放置すれば、例えば銀行システムの元帳の残高が減額されてしま

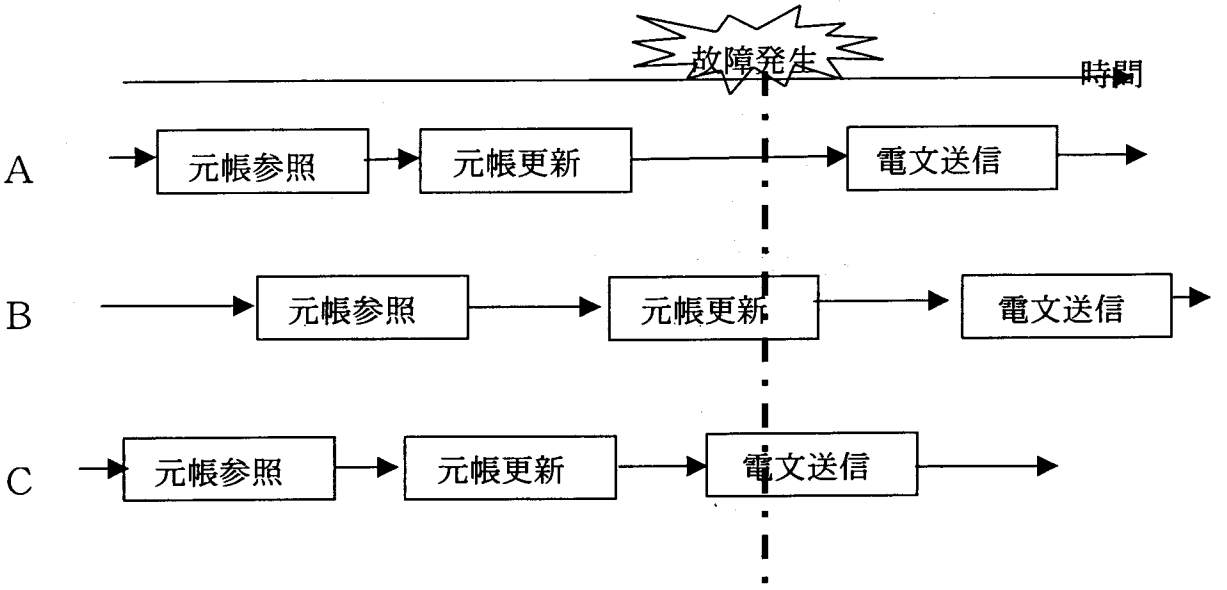


図 3.1 並列トランザクション処理の進行例

っているのにもかかわらず、支払のメッセージがＣＤ端末に届かず、現金を手にすることができないようなことが生ずることになる。

このようなことを避けるためには、システムの復旧処理において、ファイルの情報と端末への出力情報およびメモリ上の情報の三者が互いに矛盾の無い状態に復元された上で処理を再開しなければならない。

このための原理的な方法は、トランザクション処理を仮更新処理と実更新処理の２つのフェーズに分離し、仮更新フェーズの最後にトランザクション毎に更新前情報（ファイルの更新前情報や入力電文情報、メモリ情報など）と更新後情報（ファイルの更新後情報、出力電文情報、処理後のメモリ情報など）を外部記憶装置に格納し、それが確実に書き込まれたことを確認した後、実更新フェーズへ進むという方法である。（図3.2）

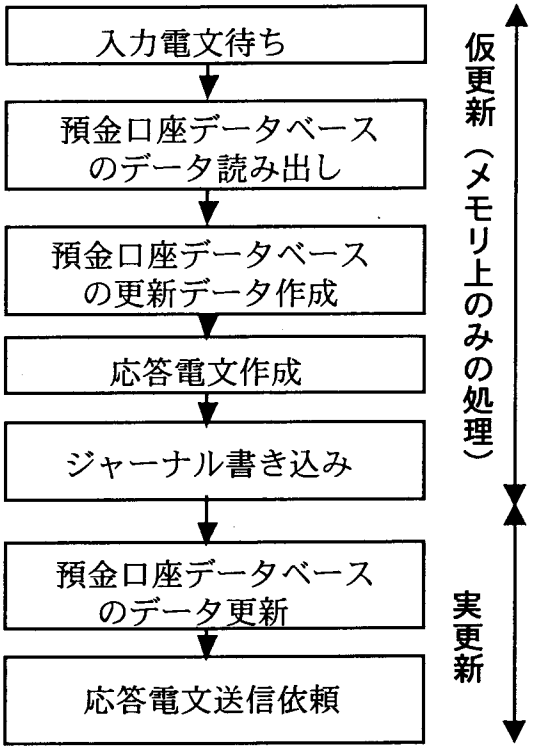


図 3.2 業務処理とジャーナル処理

このような処理手順をとった場合、もしシステムの中断が生じた時には仮更新フェーズにあるトランザクションは、ファイルへの出力や端末への送信が一切おこなわれていないので、すべてそのまま再処理を行えばよい。一方、実更新フェーズへ進んだトランザクションはファイルを更新後情報に書き直し、端末にも出力電文を送信することによって処理が完了した状態に復元する。

以上がトランザクション処理システムの復旧処理の基本的な考え方であるが、実際のシステムでは、例えば復旧処理時間を短縮するため履障の可能性があるトランザクションの数をできるだけ絞り込むために、ジャーナル情報を記録するポイントを何個所かに設け、仮更新と実更新のフェーズを単純な２分割ではなく細分化するなど様々な工夫は行われている。しかし、いずれの場合もジャーナル処理が復旧処理のキーであることは間違いがなく、トランザクション処理システムにとって非常に重要な機能である。

3.2.2　ジャーナル処理の方式条件

本節では、障害復旧処理に重要な役割を果たすジャーナル処理の方式条件について考察する。

（1）直列性の保証

障害復旧処理においては、前節で示したとおりトランザクションが障害に遭遇した時に、それが仮更新フェーズに在ったのか、実更新フェーズにあったのかの判別が復旧処理のキー情報となるが、その判断はジャーナルにデータが正常に書き込まれているか否かで行われる。すなわち、ジャーナルにデータが正常に書き込まれていればそのトランザクションは実更新フェーズにあると判断し、それ以外つまりジャーナルにデータが無いか、あっても不完全な場合はすべて仮更新フェーズにあると判断する。この判断をトランザクション毎に確実に行うために、ジャーナルはシステムの中で一時に一つのトランザクションだけを処理する、いわゆる逐次処理を行う必要がある。システムの中で並行に処理されている複数のトランザクションがジャーナル処理ルーチンを任意の時点で呼び出しても、ジャーナル処理ルーチンにおいて逐次的に制御し、時間的に直列に処理することが必要である。

（2）高性能化

すでに述べたとおり、ジャーナル機能は通常処理で必ず使われるために、高性能であることは必須の条件である。

さらに、データベースの更新や端末への処理結果の送信などの実更新は、ジャーナルにトランザクションのデータが書き込まれた後で行われる。すなわち、ジャーナルへの書き込みとデータの更新処理を直列に行うことによって、たとえデータベースの更新途中や端末との通信の最中にシステムが異常に停止しデータが失われても、それを確実に復元することができる。そして仮更新、ジャーナル情報の記録、実更新の一連の処理の間、処理対象となるデータベースや制御情報などの共通資源は、他からアクセスされないようそのトランザクションによって専有される。この資源の専有時間を短縮することはシステム全体の性能向上に重要であり、特にその専有時間の中に必ず含まれるジャーナル処理時間を短縮することは効果が大きい。

（3）個別処理

ジャーナル処理を高性能化する方式は過去にいくつか提案されIMS[4]等に代表されるＴＰモニタに実装されている。これらの方式の主要なものとして、従来から

グループコミット方式[3]が知られている。この方式は、複数のジャーナルデータをバッファ上でまとめ、一括して書き出す方式でありオーバヘッドが少なくスループットは高くなる。しかし、バッファが一杯になるまで処理が遅らされるため、個々のトランザクションから見るとジャーナル処理が遅れることになる。このため一定時間以内にジャーナル処理が終わらない時は、バッファが一杯にならなくても強制的に書き込み処理を行うなどの工夫が必要となる[3]。また前述の通り、その間トランザクションが資源を専有したままであるため、ジャーナル処理完了の遅れが資源専有解除の遅れとなり、システムの性能へ悪い影響を与える[6]。 さらに、何らかの原因でジャーナルへの書き込みに失敗した時には、アボートや再処理などが必要となるトランザクションが複数になるため復旧処理の時間が長くなる、運用の条件が変わるために、運用条件の継続性を優先するシステムの更改などのケースでは採用が難しいなどの問題がある。

このような実用上の理由から、本論文ではトランザクション毎の個別処理を前提としたジャーナル処理方式を対象とする。

（4）装置独立性

グループコミット方式と並んで、WADS方式[6]も高性能なジャーナル方式として提案されている。この方式は、磁気ディスク装置のようにデータ書き込みに当たって、回転待ちが必要な媒体を利用する場合に、回転待ち時間を最少化し見掛け上装置の入出力時間を短縮しようとする方式である。本研究では、特定のハードウェアを前提に高性能化を目指すのではなく、装置独立な高性能方式を目指す。

3.2.3 ジャーナル処理方式

前述の要求条件を満たすジャーナル処理方式の代表的な方式としてこれまで一般的に用いられてきたジャーナルの制御方式（以下では標準方式と呼ぶ）を図3.3に示す。

ジャーナルの要求条件の一つである直列性（1）を満足するために、一時期には一つのトランザクションの処理要求しか実行しないように制御する。つまり、ジャーナル処理ルーチンを一つの資源とみなし、それを専有できた場合のみ処理を行い、その他の場合は待ち合わせる。即ち、ジャーナル処理ルーチンはＳＲＲ（Serially Reusable Routine）として制御される。

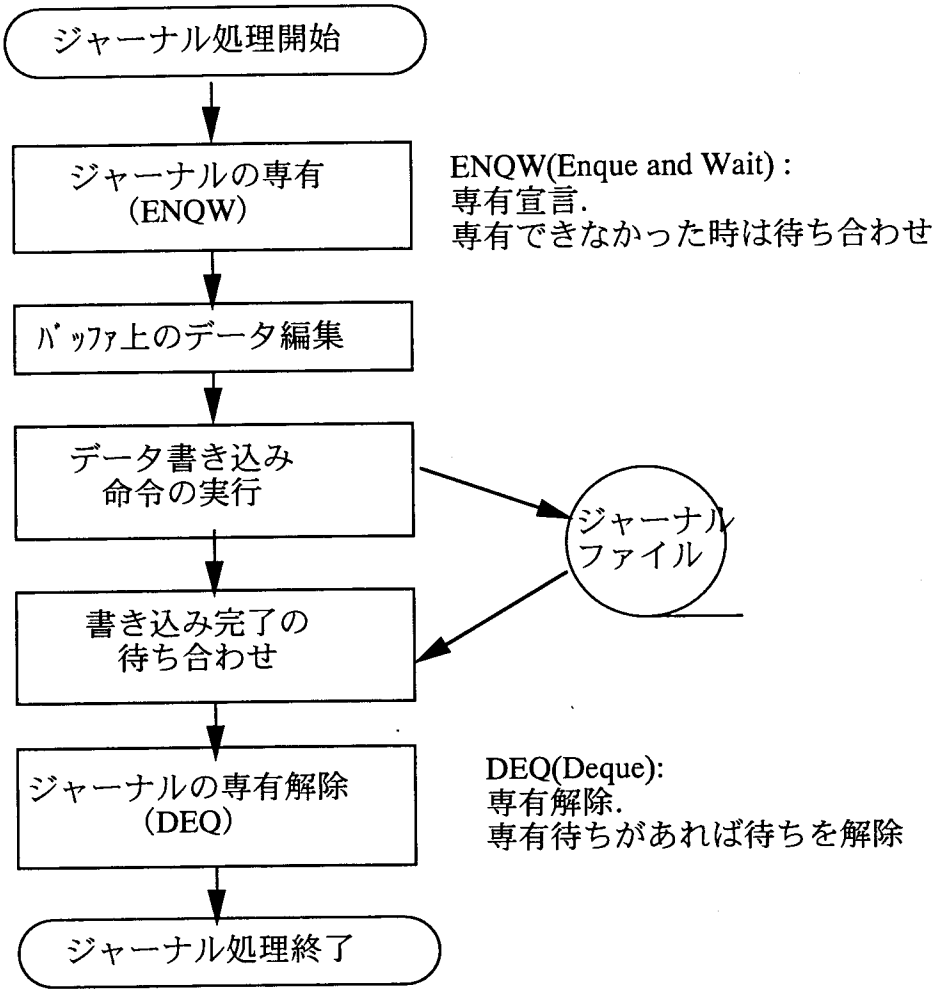


図3.3　標準ジャーナル処理ルーチンの概要

このために、ジャーナル処理ルーチンに対する排他制御（ENQW[1]、DEQ[2]）を行う。このようなジャーナル処理ルーチン（以下JNLと略記）の時系列的な動作の様子を図3.4に示す。

---

[1] ENQW(Enque and Wait):ジャーナル処理ルーチンの専有を宣言する。もし、他のタスクがジャーナル処理ルーチンをすでに専有していた場合には、それが解除されるまで待ち状態となる。

2 DEQ(Deque):専有解除を宣言する。もし、専有解除を待ち合わせているタスクがあればそのタスクの待ち状態を解除する。

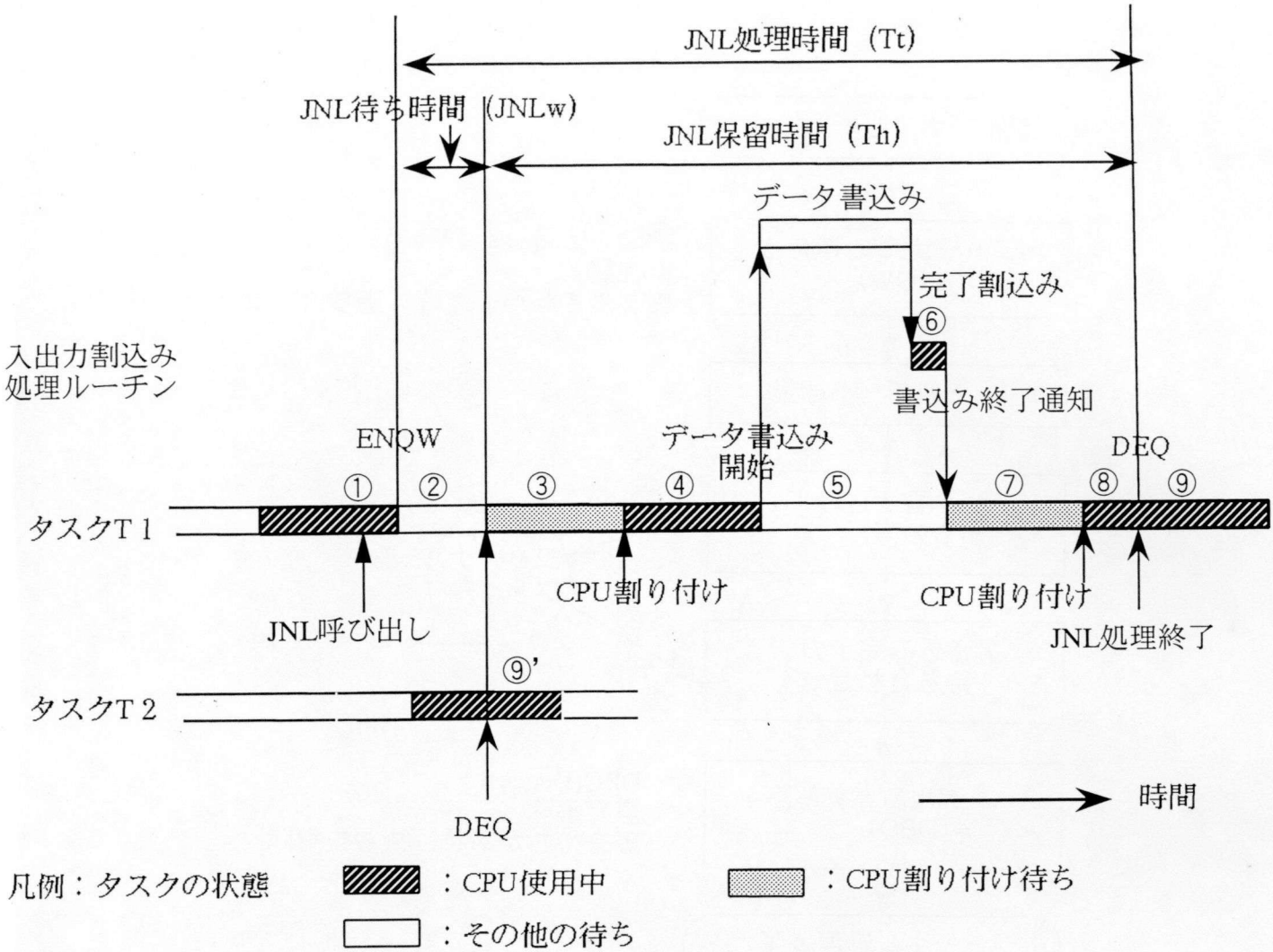


図3.4　標準ジャーナル処理ルーチンの動作例

図3.4においては、２個のタスク(T１およびT２)が業務処理を実行中で、タスクT２がJNLを専有している時に、もう一方のタスクT1がJNLを呼び出した状況を示している。

（以下の説明において、円内の数字は図3.4と対応している）

①タスク T1 が JNL を呼出し、まず JNL の専有の宣言（ENQW）が行われる。

②この時 JNL はタスク T２によって専有されているため、タスク T1 は専有解除を待ち合せるための待ち状態となる。

③タスク T２が処理を完了し JNL の専有を解除する（DEQ）と、タスク T1 は JNL の待ち状態が解かれ、CPU の割り付け待ちとなる。

④CPU を割り付けられたタスク T1 は、JNL を専有しバッファ上のデータ編集などを行った後データ書込みの命令を実行する。

⑤データ書込みの命令により、外部記憶装置が起動される。タスク T1 は書込み終了の待ち合わせを行うため、CPU を放棄し待ち状態となる。
⑥データ書き込み完了の割込みが入出力割込み処理ルーチンで処理され、タスク T1 に書込み終了が通知される。
⑦タスク T1 は CPU の割付け待ちの状態となる。
⑧CPU が割付けられたタスク T1 は、後処理を行った後 JNL の専有を解除(DEQ)し JNL 処理を終了する。
⑨、⑨'JNL 処理終了後の処理へ進む。

この制御方式は、前述のジャーナルの方式条件のうち、(1)、(3)および(4)の機能条件はいずれも満足する方式であるが、性能上の条件(2)についての確認を行うために、次節で評価を行う。

3.3 方式評価

ここでの評価のねらいは、これまで一般的に用いられているジャーナル処理方式の性能の限界を明らかにし、制御方式上のボトルネックを発見し改良の指針を得ることにある。解析的な手法では、ソフトウェアの制御方式をきめ細かく評価することは困難なため、ここではシミュレーションによることとし、シミュレーション言語SLAM 2 [72]をベースに、対話形式によって計算機等のシミュレーションが可能なよう開発された対話型計算機システムシミュレータ[55]、[78]、[79]を用いた。

なお、トランザクション処理システムの性能評価については過去色々な観点から行われてきた[18], [25], [42], [2]。しかし、それらのほとんどはハードウェア構成がシステム全体の性能に与える影響を分析しようとするものであり、本研究のようにソフトウエアの制御方式を対象とした評価例は少ない。わずかに、文献[18]の一部で、ジャーナルに利用する種々の記憶装置の構成がシステムの性能にどのような影響を与えるかを評価したものが見られるが、本研究のように記憶装置の条件を固定してソフトウェアの制御方式が性能に与える影響を評価するものではない。

3.3.1 評価の条件

ジャーナル処理方式を評価するにあたっては、それが使われる環境条件を設定する必要がある。ここではトランザクション処理業務の代表的な例として、銀行業務の中

で最も処理頻度の高い預金業務の環境を仮定し評価する。これは、トランザクション処理システムのベンチマーク用として設計されたTPC-A[76]とほぼ同様のものであり、すべてのトランザクションはこの１種類の業務のみを行うものとする。この環境のもとでジャーナル機能が次々と呼び出される状況を擬似する中で、ジャーナル処理方式の動的な性能を評価する。

より具体的には、このシミュレーションモデルには、預金業務を行うタスクがあらかじめ固定数Nだけ用意され（タスクの実行優先度はすべて同一）、トランザクションがシステムに到着する度にタスクが１つ割当てられ、預金業務が実行される（以下図3.5参照）。１つのトランザクションに対応した業務処理が終了すると、そのタスクは解放され、次の新たなトランザクションの割当てを待つ。もし、タスクの数以上のトランザクションがシステムへ到着した場合は、タスクの空きを待ち合わせる。そのための入力電文待ち行列が１個用意されている[65], [69]。

預金業務処理モデルの詳細は以下のとおりである。すなわち、処理依頼のあった顧客の預金口座データベースやシステムの運用管理上必要なサマリデータ（例えば、店別・科目別の取扱件数と金額など）をデータベースから読出し、口座データの更新情報、および端末への応答電文などをメモリ上に作成する。ジャーナル処理ルーチンを呼出して、入力電文や応答電文、預金口座データベースの更新前のデータおよび更新後の新しいデータなどをジャーナルへ書込む。それが正常に終了したことを確認した後、口座データベースやサマリデータファイルを実際に更新し、応答電文の端末への送信を依頼して一連の処理を終わり次の電文の入力を待つ。

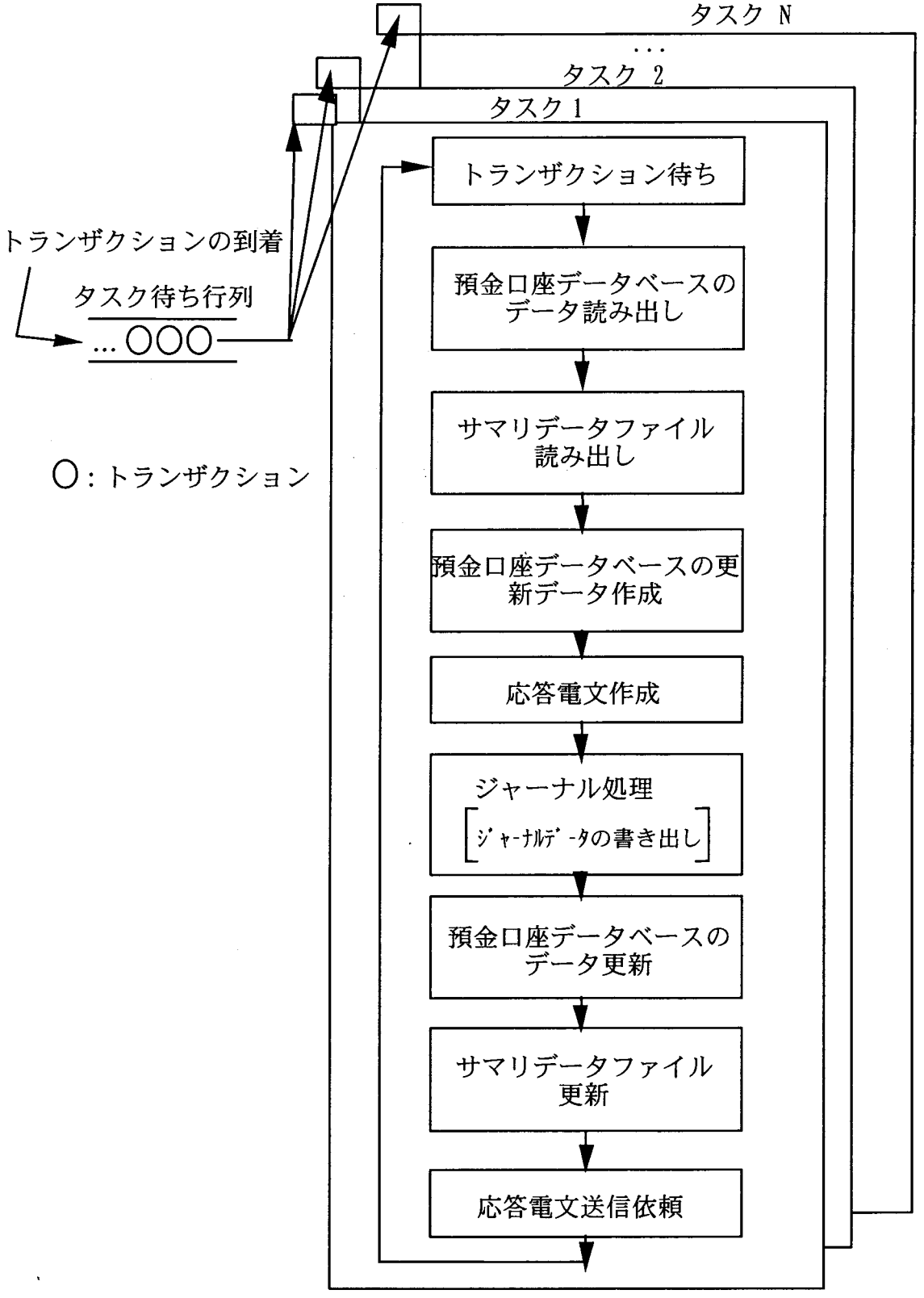

タスク N
タスク 2
タスク 1
トランザクション待ち
預金口座データベースの
データ読み出し
サマリデータファイル
読み出し
預金口座データベースの更
新データ作成
応答電文作成
ジャーナル処理
ジャーナルデータの書き出し
預金口座データベースの
データ更新
サマリデータファイル
更新
応答電文送信依頼
トランザクションの到着
タスク待ち行列
…○○○
○：トランザクション

図 3.5　業務処理モデル

以上のシミュレーション環境のもとで、次の各種特性値を評価する。

（a）JNL 待ち時間（JNLw）

図3.4において、①でJNLの専有要求（ENQW）を出してから、JNLを専有するまでの時間（すなわち②）。

（b）JNL 保留時間（Th）

JNLを専有してから、専有解除（DEQ)を行うまでの時間。（図3.4の③から⑧)

（c）JNL 処理時間（Tt）

JNL呼び出し後、その処理が終了するまでの時間をJNL処理時間とし、次のように定義する。

Tt ＝JNLw ＋Th （図3.4の②から⑧)

実際にはJNL専有要求（ENQW)を出す前に準備のためのCPU時間①があるが、他に比べ極く短いのでここでは無視する。

（d）トランザクション処理時間

図3.5において、トランザクションがシステムに到着し、タスク待ち行列にキューイングされてから、業務処理が終了しタスクが開放されるまでの時間。

なお、評価にあたっては次で定義する処理件数をパラメータとして用いる。

（e）処理件数

単位時間当たりにシステムで処理されるトランザクション数。本論文では、一秒当たりに処理されるトランザクション数TPS(Transaction Per Second)を単位とする。

3.3.2 シミュレーション結果と考察

シミュレーションは表3.1に示す条件のもとで実行し、ほぼ定常状態に達したと判断される状況で特性値を採った。

表3.1 シミュレーションの条件

項目	条件
CPU 数	1
CPU 負荷	4.5mS（一トランザクション当たりの合計） （業務処理モデルに沿って、処理ステップ毎に配分しそれぞれを平均とする指数分布サービスを仮定）
ジャーナルファイル	6mS（一回の書き込み当たり） ファイル数 1
預金口座ファイル	17mS（一回の読出し、書き込み毎） ファイルでの競合なし
サマリデータファイル	6mS（一回の入出力当たり） ファイルでの競合なし
タスク数	１０（実システムを参考に、タスクがボトルネックとならないように設定）
トランザクションの入力	ポアッソン到着

図3.6にシミュレーションで得られたトランザクションの平均処理時間を示す。処理件数が１００TPSを超えるあたりから平均処理時間が急速に立上がっており、これがシステムの処理能力の限界と見なされる。一方、同図に併せて示すCPUの使用率を見ると、システムの処理能力の限界付近であってもなお５０％以下であるのに対し、業務処理を行うタスクの使用率は７０％を超えておりさらに負荷が増えると急速に立ち上がっていることから、タスクがボトルネックとなっていることが予想される。

この状況をさらに分析するために、タスクの状況の変化を図3.7に示す。図ではトランザクションを割り付けられ業務処理を実行中のタスクを使用中タスクと呼び、その数の推移を示している。さらに使用中タスクの中でJNLを専有しているタスクおよびJNLの専有を待ち合わせているタスクの数の推移も併せて示す。 処理件数１００TPSの前後から、JNL待ちに陥るタスクの数が急増している。つまり、タスクがJNL専有待ち状態に次々と陥り、空きタスクが無くなり、その結果タスクの使用率が増加していることが分かる。したがって、現象としてはタスクがボトルネックとして現れているが、このままの状態で単にタスクの数を増加させてもこの状況は改善されない。

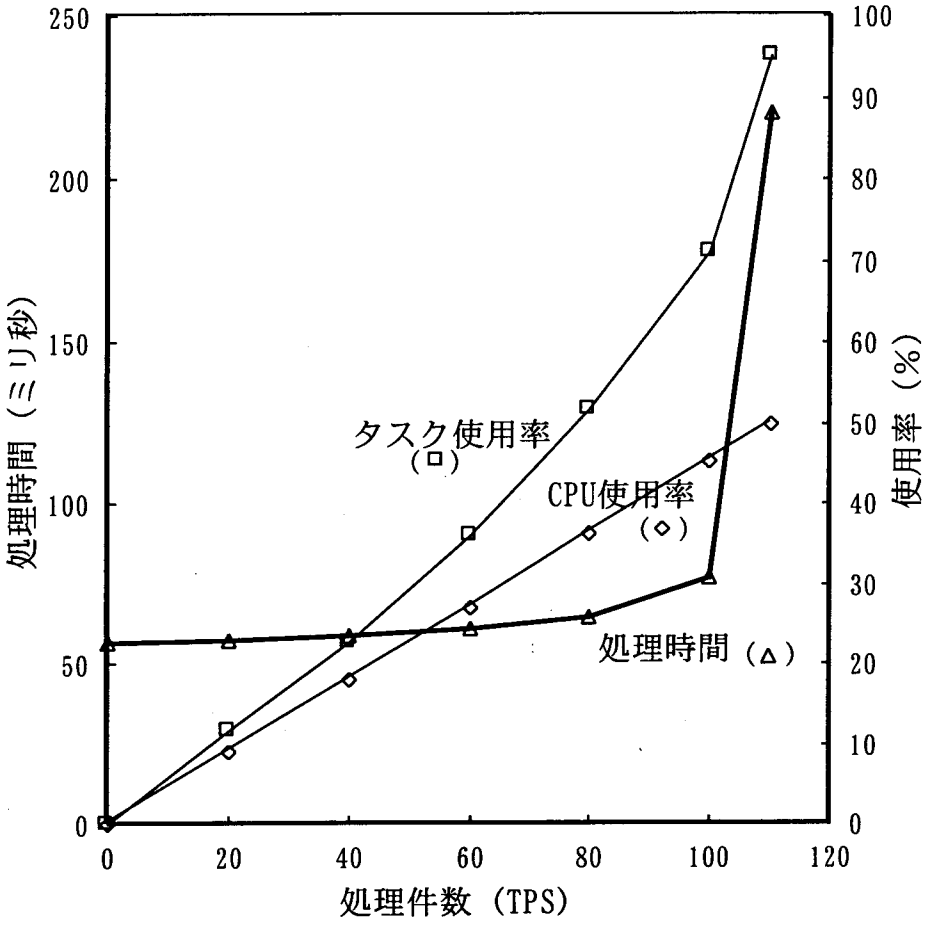


図 3.6　トランザクション平均処理時間とCPU／タスク使用率

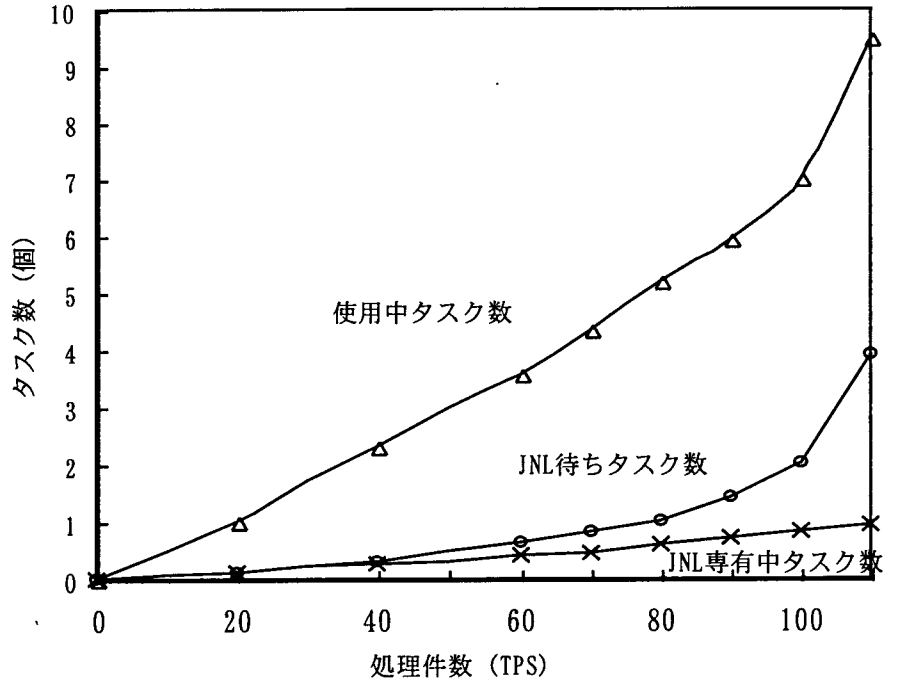


図 3.7　状態毎のタスク数

図3.8にJNL処理時間（Tt）とJNL保留時間(Th)の推移を示す。JNL処理時間（Tt）は、処理件数の増加とともに増大するが、その要因はTtとThの差であるJNL待ち時間(JNLw)の急激な立上りにある。窓口が一つのサーバでは、サービス時間のわずかな延びが待ち時間の急激な延びを引き起こすため、JNLのサービス時間つまりJNL保留時間（Th）の短縮が重要である。この保留時間のうち、処理に必須な外部記憶装置への出力時間やCPU処理時間は同じ装置を用いる限り短縮の余地は無い。これに対し、図3.4に示した③や⑦などの待ち時間を出来るだけ短縮するようにスケジュールすること、特に⑨や⑨'のようにJNLの専有解除後も、ひきつづきCPUを専有して処理を継続するため、JNLの専有が可能となったにもかかわらず別のタスクにCPUを割付けられず、結果的にJNLの専有を継続しているのと等価になることを回避することなどの改善の可能性が考えられる。

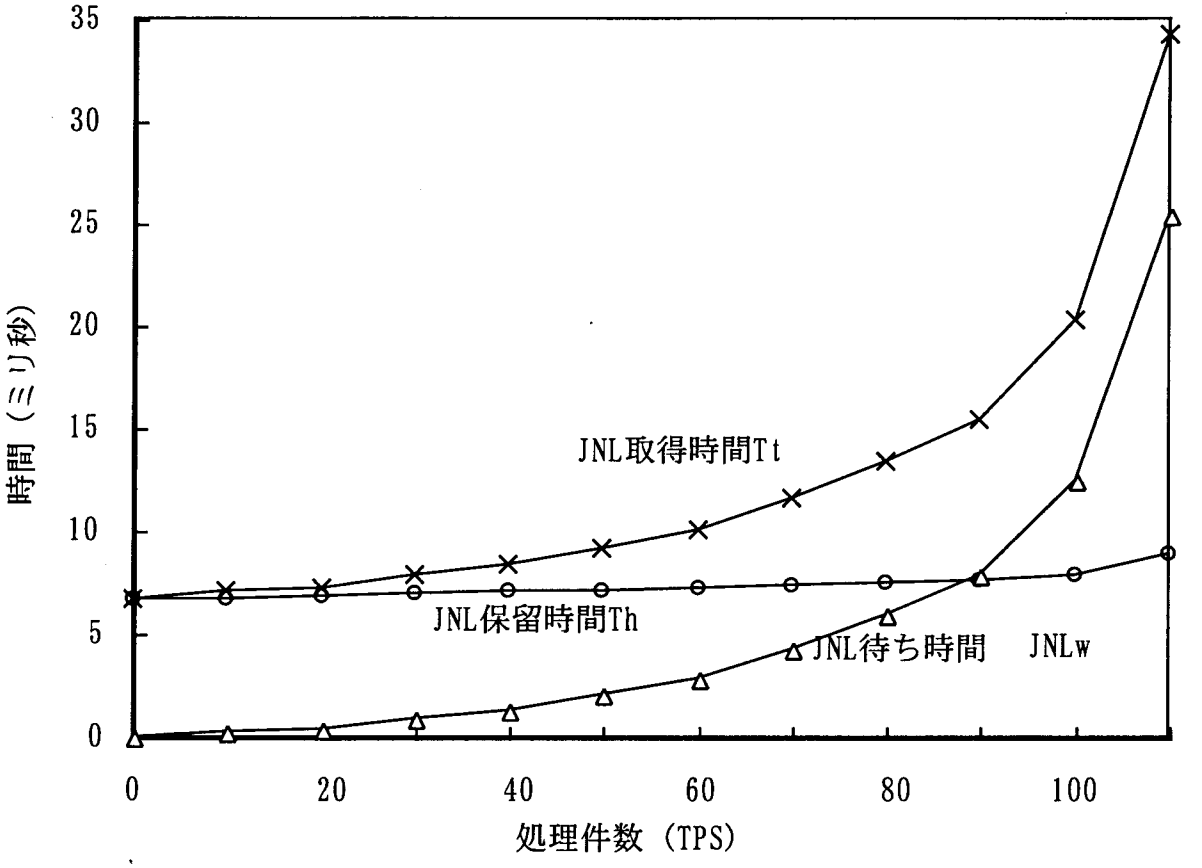


図 3.8　ジャーナル処理時間

3.4 高性能ジャーナル処理方式の設計

3.4.1 高性能ジャーナル処理方式

前節において指摘した問題を改善する方法として一般的なのは、ジャーナル処理を行う専用のタスクを一つ設け、そのCPU 割当優先順位を他より高くする方法である。しかし、この方法の場合タスク間でのやりとりやCPU のプリエンプティブスケジュールのためのオーバヘッドが避けられない。これに対し、入出力割り込みの処理の中にJNL のスケジュール機能を組み込むことによって、JNL の保留時間を最低限必要な区間だけ、即ちデータの書き込み時間とCPU 処理時間にできるだけ近づける方式を考案した。割り込み処理の中に、スケジュール機能を組み込むためには、ＯＳの提供する入出力割込みアペンテージ・ルーチン[3]を利用する。一般に同一の外部記憶装置に対する入出力要求は一つづつ直列に実行される。このことを、JNLに対する排他制御に利用することによって排他制御のオーバヘッドを軽減する方式である。同時に入出力割り込み処理の中で直接JNLのスケジュールを行うことによって、CPU待ちを介さずに直ちに次の要求を起動できるため無効保留を回避することができる。 この提案方式の処理の概要は以下の通りである（図3.9）。

(1) ジャーナル処理ルーチンでは、JNL処理要求をJNL処理待ち行列にキューイングする。

(2) それが待ち行列の先頭ならば、ただちにデータの書込みを開始する。もし先頭でなければキューイングするだけでJNL処理の終了を待つ。

(3) データの書込み完了の割込みが報告されたなら、割込み処理ルーチンの中で要求元へJNL処理終了の通知をするとともに、次の処理要求を取りだしデータ書込みを開始する。

---

[3] 入出力アペンデージルーチン： OSとしての標準的な入出力割込み処理の他にユーザ固有の割込み処理を組込むことを可能とするインタフェースが解放されている場合がある。このユーザ固有の処理を実現するルーチンを入出力アペンデージルーチンと呼ぶ。

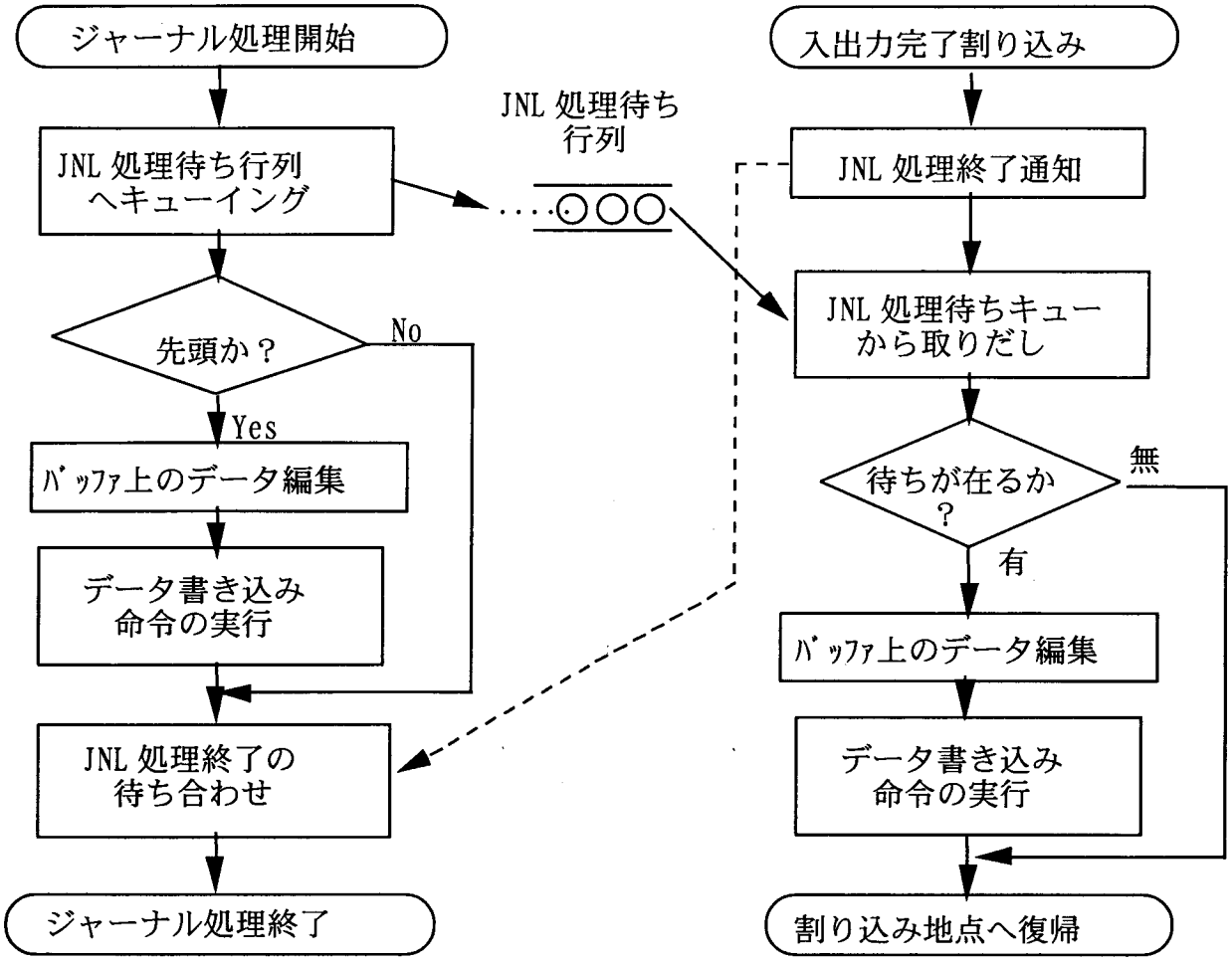


図 3.9　高性能ジャーナル処理方式の概要

提案方式の時系列的な動作の様子は次の通りである。図3.4と同様、２個のタスク（T１およびT２）が業務処理を実行中でタスクT２がJNLを専有している時に、もう一方のタスクT1がJNLを呼び出した状況を示している。なお、以下のローマ数字は図3.10と対応している。また図3.4と対応する部分については同じ番号を用いている。

（ｉ）タスクT1がJNLを呼び出し、処理要求をJNL処理待ち行列にキューイングする。
（ii）すでにタスクT２がJNLを専有しているため、JNL処理待ち行列へキューイングしただけでタスクT１は待ちに入る。
（viii）すでに開始されていたタスクT２のJNLデータ書込みが完了し割込みが報告されたら、入出力割込み処理ルーチンが直ちに起動されタスクT２に対し、JNL処理終了が通知される。

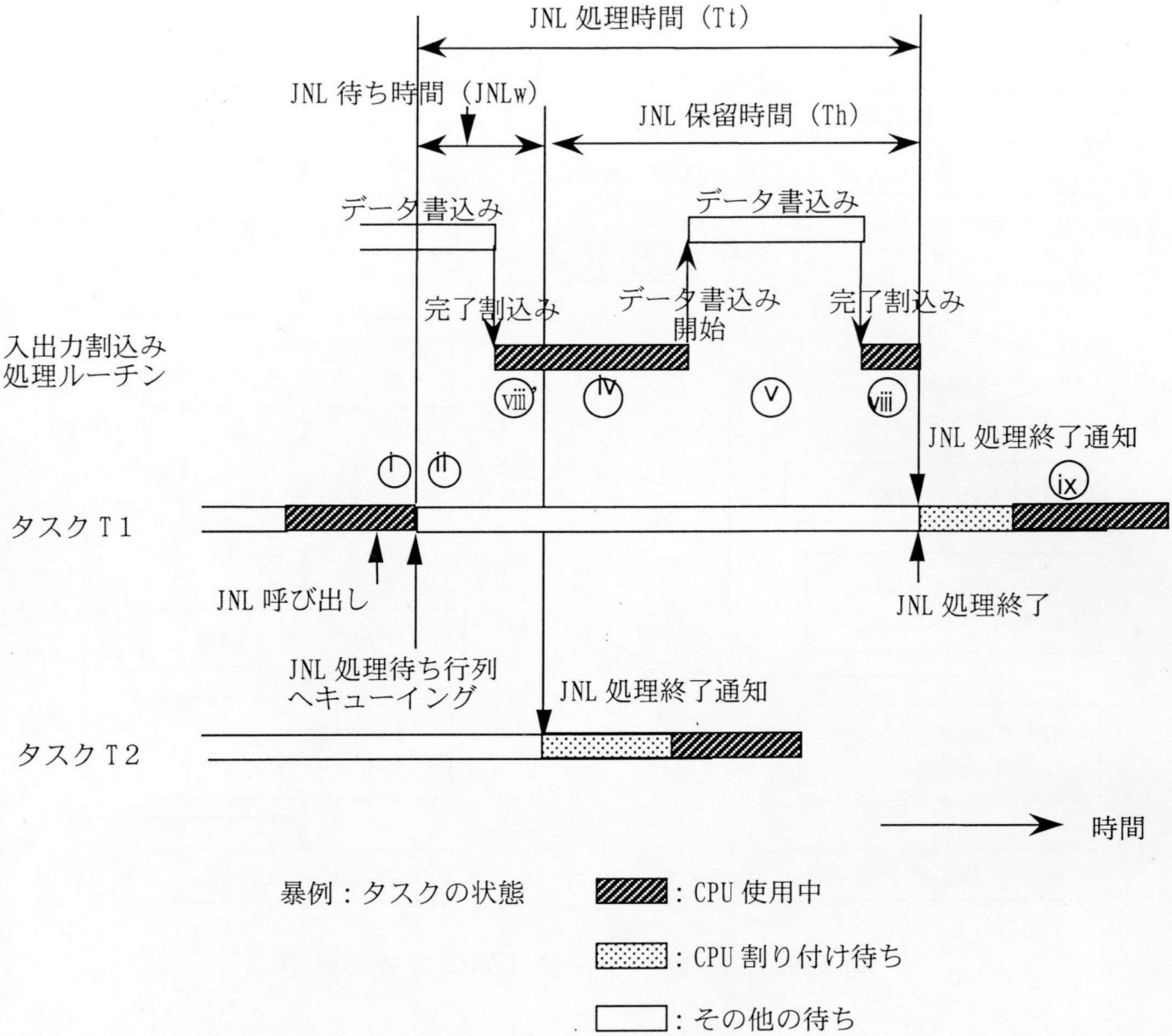


図 3.10　高性能ジャーナル処理ルーチンの動作

（iv）引き続きJNL処理待ち行列上の次の処理要求（ｉでキューイングされたT1の要求）が取り出され、データ書込みが開始される。

（viii）データ書込み完了の割込みにより、再び入出力割込み処理ルーチンが起動され、タスクT1へJNL処理の終了が通知される。（図3.4の⑥の処理も含まれる）

（ix）タスクT1はJNL処理終了の待ちが解除され、CPU 待ちを経て次の処理へ進む。

このような方式を採ることにより、従来の標準方式に比べ、図3.4における③および⑦のCPU待ちがいずれもJNL保留時間には含まれなくなる。データ書き込み完了の時点でJNL処理の終了が通知されると同時に次のデータ書き込みが開始されるため、図3.4の⑨`による無効保留も無くなり、JNLの保留時間が必要最小限の範囲に極小化できる。

3.4.2　高性能ジャーナル方式の評価

提案の方式を評価するために標準方式と同一の条件でシミュレーションを行った。

まず、トランザクションの平均処理時間に関する結果（図3.11）を見ると、処理件数が１３０TPS近辺から立ち上がり処理限界を示している。標準方式のケース（図3.11には図3.6のデータを点線で再掲）では、それが１００TPS近辺であり大きく改善されている。たとえば平均処理時間７５msを満足する処理件数を比較すると、標準方式に比べ２５％以上向上しており、提案方式の効果は顕著に現れている。

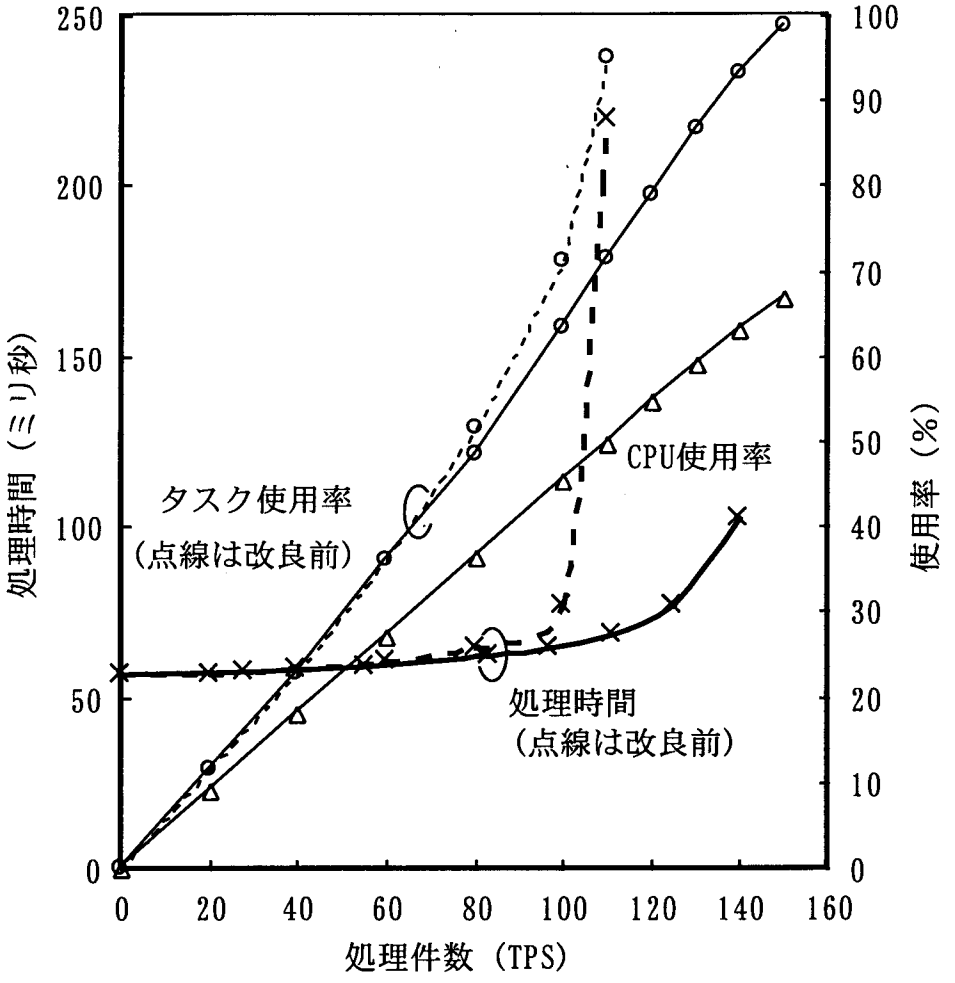


図 3.11　トランザクション処理時間およびCPU／タスク使用率（提案方式）

次に、同図に示したCPUの使用率を図3.6と比較すると標準方式と提案方式でほとんど同じであるが、タスクの使用率は提案の方式の場合、大きく低下している（標準方式のデータを図3.11に点線で再掲した）。タスクの数は同一であるにも関わらずこの様な結果となったことは、前節で述べた見かけ上のタスク不足であるとの考察を裏付けるものである。図3.12にはJNL処理時間を示している。あわせて表示した標準方式のJNL処理時間（図3.8の再掲）と比較すると大幅に改善されている。特に、提案方式のJNL保留時間はトラヒックの増加にかかわらずほとんど増加せず、処理に必要なデータ出力時間およびCPUの実行時間のみとなっていることが裏付けられており、提案の方式は当初のねらいを達成している。

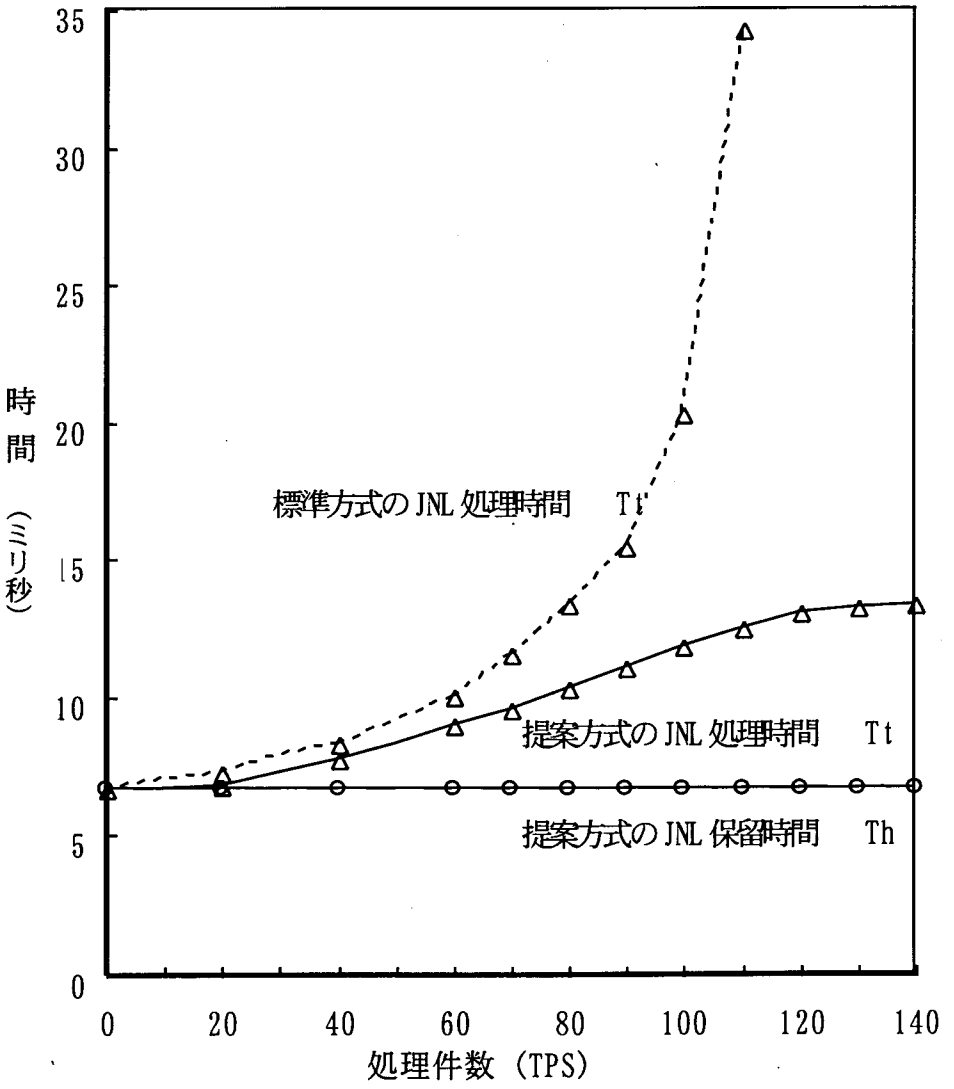


図 3.12 提案方式のジャーナル処理時間

3.5 総合評価

前節まではジャーナル処理に着目した部分的なシミュレーションであったが、本節ではより実システムに近い大規模なシミュレーションによって総合的な評価を行った結果を示す[61]。評価対象としたのは中規模クラスの銀行のオンライン勘定系システムで、ハードウェアおよびソフトウェア（業務プログラム、ＴＰモニタおよびＯＳ）のそれぞれをモデル化し、計算機専用シミュレータを用いてシミュレーションを実施した。

（１）ハードウェアモデル

預金口座データベース用の磁気ディスク装置は３６台とし、８台毎に１台のチャネルの組を３組と１２台に対し１台のチャネルの組１組とする構成とした。磁気ディスク装置へのアクセスは均等と仮定し、一様乱数によりアクセスすべき装置を決定した。回線や端末装置のモデル化は行わず、代わってモデルの中で直接電文を発生させた。したがって、トランザクション処理時間には電文の送受信のための時間は含まない計算機内処理時間となる。メモリはモデル化の対象としていない。一般にトランザクション処理システムでは定常負荷の範囲では、メモリはボトルネックにならないよう設計されるため、この仮定は妥当である。

（２）ソフトウェアモデル

業務処理については、処理頻度が最も高い預金業務のみをモデル化した。概ね図3.5に示すとおりであるが、実際のアプリケーションプログラムの処理に忠実にモデル化した。前節までのシミュレーションでは、ＴＰモニタの機能については、ジャーナル処理機能のみを疑似していたが、総合評価においては電文の入出力やスケジューリングなど業務プログラムを管理するＴＰモニタのその他の機能も疑似している。なお、ジャーナル処理については、標準方式と提案方式の両者についてモデル化している。また、ＯＳの機能については割り込み処理、タスクディスパッチャおよび入出力機能、データベースアクセス機能など必要なものを疑似した。

（３）総合評価結果

ジャーナル処理だけを入れ替え、その他の条件は全く同一にして実施した総合シミュレーションの結果を図3.13に示す。一回のデータ収集に長時間を要すため、データを得たポイント数は少ないが次の点を確認できた。すなわち、平均ジャーナル処理時

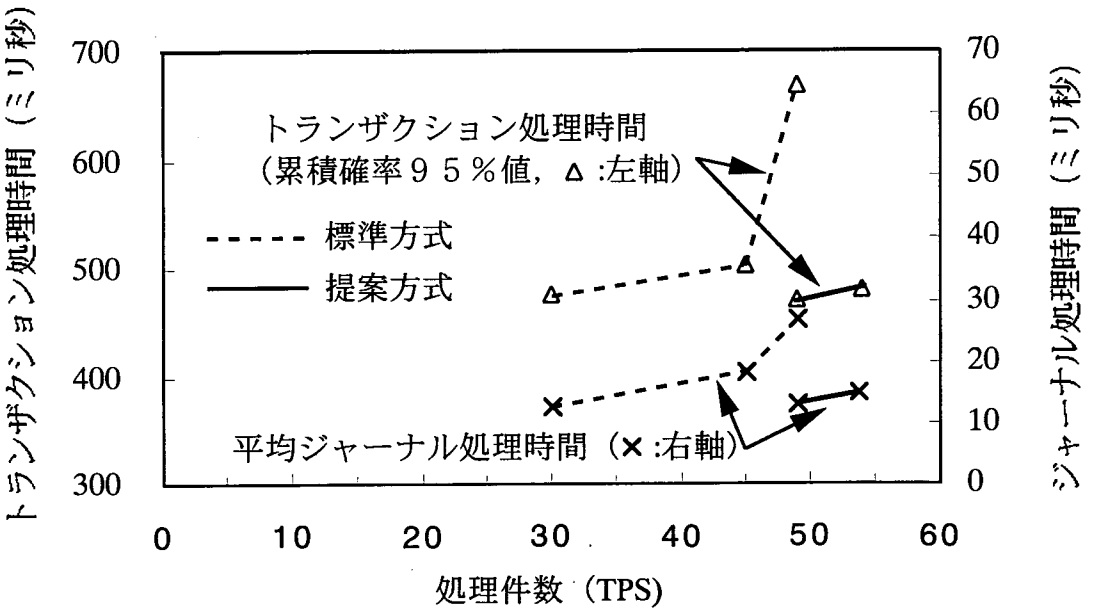


図 3.13　トランザクション処理時間とジャーナル処理時間

間(図3.13の×印)について見ると、提案方式では45ＴＰＳを越えると大きく増加しているのに対し、改良方式では54TPSまで増加しても処理時間に大幅な増加は見られない。

一方、トランザクション処理時間(95%のトランザクションがこれ以内に処理される時間)について見ると、従来のジャーナル方式の場合、45TPSを越えると急速に増加し、この辺りがシステムの処理限界と推定される。これに対し、ジャーナルを提案方式に変更すると、54TPS辺りでもトランザクション処理時間はまだ安定しており、少なくともここまではシステムの処理限界が改善されることが確認できた。これは、システムの総合性能が20%改善されたことになり、前節までに述べた効果はシステムの総合的な性能に対しても有効であることを示している。

3.6 結言

本章では、トランザクション処理システムにおいて性能上のボトルネックとなるジャーナル処理方式に関して、従来から広く用いられてきた方式にかわる新たな高性能方式を提案した。

すなわち、まず従来の方式についてシミュレーションによって性能上のボトルネックが排他制御の処理方式にあることを明らかにした。続いて、この排他制御を入出力

割り込み処理で実行する全く新しい方式を提案し、従来方式に比べ総合性能で２０％以上の改善が見込まれることをシミュレーションで明らかにした。

これらの評価に基づき、高性能ジャーナル処理方式を設計し、中大型メインフレーム計算機（ＤＩＰＳ）用のＴＰモニタに実装し、高トラフィックなトランザクション処理システムの商用に適用し、所期の効果を得た。

近年、ダウンサイジングの傾向は基幹的なシステムにも及び、トランザクション処理をサーバ／クライアントの構成で実現する例も散見されるようになってきた。しかし、高トラヒック、大規模データベースそして高い信頼性を実現するには現状のサーバ等のＯＳやＴＰモニタにはさまざまな問題があり、まだ基幹システムへの適用が本格的には始まっていない。 本章で述べたジャーナル機能もその一つであり、これらの機能がワークステーションなどのＯＳやＴＰモニタに実装されてくれば、コストパフォーマンスの高いトランザクション処理システムが実現されることが期待される。

第４章

大容量記憶装置制御方式のトラヒック設計

4.1 緒言

本章では、大容量記憶装置（Mass Storage System : ＭＳＳ）をトランザクション処理システムに適用するための制御方式に関するトラヒック設計について論ずる。

トランザクション処理システムの運転においては、第３章で述べたジャーナルのように処理の履歴情報を格納するファイルやオンラインファイルのバックアップ情報など、オンラインでの直接のアクセスは無いが、大容量で且つ長期間保存すべきファイルが必要である。これらについては、オンラインでの応答時間の制約は無いため、通常磁気テープ装置が用いられてきた。しかし、システムの規模の増大に伴い、それらの量も急増し、操作コストの増大、操作誤りなどテープ操作の煩雑さに起因する各種の問題が大きくなる。

一方、トランザクション処理システムに用いられるファイルは、応答時間の条件を満足させるために磁気ディスク装置を用いるのが通例であるが、必ずしも短い応答時間は必要無いが、コストの低いファイルに対するニーズも見られる。たとえば、バンキングシステムで顧客の過去の取引明細に対する問い合わせにオンラインで答える例がそれである。

この様な問題解決の可能性を持った装置として開発されたのが大容量記憶装置(ＭＳＳ)である。自動操作機構をもった磁気テープ装置と磁気ディスク装置とを組み合わせ記憶階層を構成した大容量記憶装置[34],[70],[28]は、主記憶装置に適用されてきた仮想記憶方式を、二次記憶装置へ拡大、適用したもであり、従来の比較的単純な構

成の二次記憶装置に比べ、その特性は複雑な様相を示すことが予想される。

このような装置をシステムへ導入し有効に利用するには、装置の特性を十分に把握した上で、その特性を生かした利用形態と制御方式を設計することが必要である。

本章の研究は、このような方式設計上の課題に対しての方法論を与えるものである。

過去に報告されている大容量記憶装置に関する性能評価は、装置内部の構成や制御方式を評価することに主眼が置かれ、また用いられる手法もシミュレーションによるものが大半である。Lavenberg [10]らは、ＭＳＳ内の装置構成をパラメータとして、最大スループットおよび平均アクセス時間をシミュレーションで得ている。また、藤原[58]および根岸ら[54]は、ＭＳＳ内のファームウェアの制御方式の変更が、処理能力に及ぼす影響を評価するためのシミュレータを提案し、その使用例を報告している。一方、解析による評価としては、機械修理工モデルによる近似の報告[15]が見られる程度である。

これらに対し、本研究ではＭＳＳを効率的に制御するための方式設計を行う上で、その基礎的なデータとして装置の動特性を把握・分析するための評価手法を確立する。以下、4.2節でＭＳＳの動作概要を示し、それが系内客数制限をもつ開放形の待ち行列ネットワークでモデル化されることを示す。4.3節では、そのモデルの解析の方法を与え、平衡状態における各種特性値を解析する。さらに、4.4節では系が平衡状態にあるための条件について考察し、4.5節では解析結果に基づいて、スループットや平均アクセス時間などの種々の特性値を明らかにする。そして、得られた結果が実測データと良く一致することを示し、ここで提案したトラヒック設計法が実用的にも有効なものであることを示す。

4.2 大容量記憶装置の動作概要と待ち行列モデル

4.2.1 大容量記憶装置の概要

ファイルバックアップのためのダンプ情報や処理の履歴をとる外部記憶装置としては磁気テープが初期の計算機システムから活用されてきた。また、異なる計算機システム間の情報交換の媒体としてはほとんど唯一磁気テープが用いられてきた。しかし、計算機システムの規模の増大に伴い蓄積される磁気テープの量が膨大となり、磁気テープの人手による交換や保管作業が問題となってきた。この問題を解決するため、ＩＢＭが1975年に開発したのが超大容量記憶装置（ＭＳＳ）である。磁気テープオープンリールに替え、ロボットで扱い易い砲弾型のカートリッジに磁気テープを収納し、

記録密度を向上させるため、ビデオレコーダと同様の回転ヘッドによるヘリカルスキャン方式を採用した。容量５０Mバイトのカートリッジ数千個を棚に収容し、ロボットで記録再生装置に着脱する、数百Gバイト容量の完全に自動化された記憶装置である。同様の装置は日本でも1980年に開発された［32］,［30］。

ＭＳＳの方式上での特徴は、次に示すように記憶装置の階層化技術と仮想化技術を組み合わせた点にある。すなわち、図4.1に示すとおり、小容量の高速記憶装置（磁気ディスク装置）と低速、低価格で大容量な記憶装置（カートリッジ型の磁気テープ装置）とを組合わせることによって、平均的に高速なアクセスが可能で、低価格・大容量な記憶システムを構成する。さらに、このような物理的な階層構成をＭＳＳ利用者には直接見せず、あたかも磁気ディスク装置が多数存在しているかの如く仮想化する。このように実際に存在するよりも、より多くの磁気ディスク装置があるかのように仮想的に見せる方式であるところから、仮想ディスク方式と呼ぶ。これは、主記憶装置における仮想記憶方式を、外部記憶の世界へ拡張したものと言えるが、この仮想化の処理の大半がＭＳＳの機能として実現されている点に従来装置にない特徴がある。

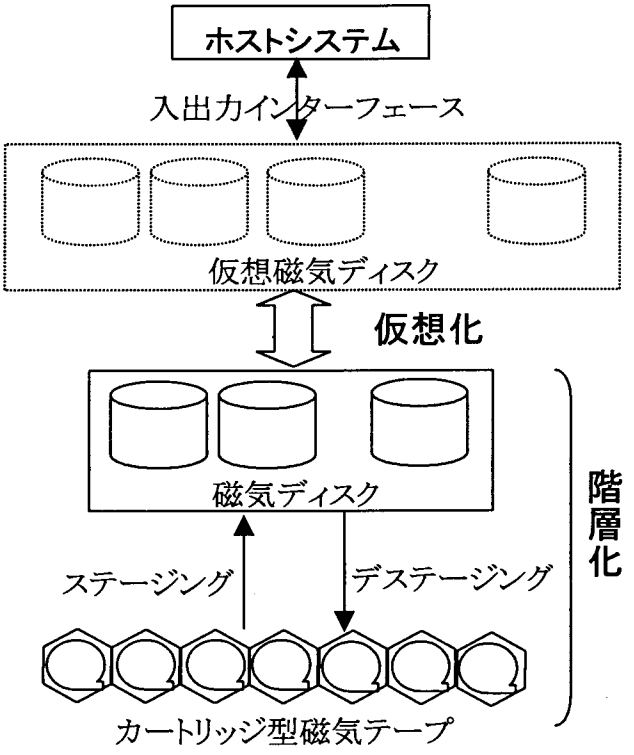


図 4.1　仮想ディスク方式の概念図

この仮想ディスク方式の実現のために、ＭＳＳは、ホスト[1]側が必要とする一部の情報のみを磁気ディスク装置上に割り付け、残りの情報は磁気テープ上に格納するよう制御する。ホストからＭＳＳ内のデータにアクセスした時、所要のデータが磁気ディスク装置上に存在しないケースが発生し得るが、この状態をシリンダフォールト状態またはシリンダフォールトが生じたという。このとき、ＭＳＳは自動的に必要なデータを磁気テープから磁気ディスク装置へ転送し、ホストからの要求に応じる。このような、記憶階層上の下位装置から上位装置へのデータ転送動作をステージング、逆に上位から下位へのデータ転送動作をデステージングと呼ぶ。

このように、記憶階層の制御と仮想化は、基本的にＭＳＳ側の機能として実現されているため、ホスト側の処理プログラムは磁気ディスク装置へのアクセスとまったく同様の手順で仮想ディスク装置上のデータにアクセスすることができる。ただ、途中でシリンダフォールトが発生するとデータへのアクセス時間が延びたように見えるだけである。

4.2.2　大容量記憶装置の構成と動作

ＭＳＳハードウェアの主な構成装置は図4.2に示す通りである。

（１）超大容量記憶部（ＭＳＭ）

ＭＳＳ全体の制御およびカートリッジ制御を司る装置であり、制御系（ＭＳＣ）、アクセッサ系（ＡＲＣ，ＡＣＣ），データ記録装置系(ＤＲＣ,ＤＲＤ)およびカートリッジセル（ＣＳ）から構成される。カートリッジセル内には、実際にデータを蓄積するための磁気テープカートリッジが格納、保持されている。

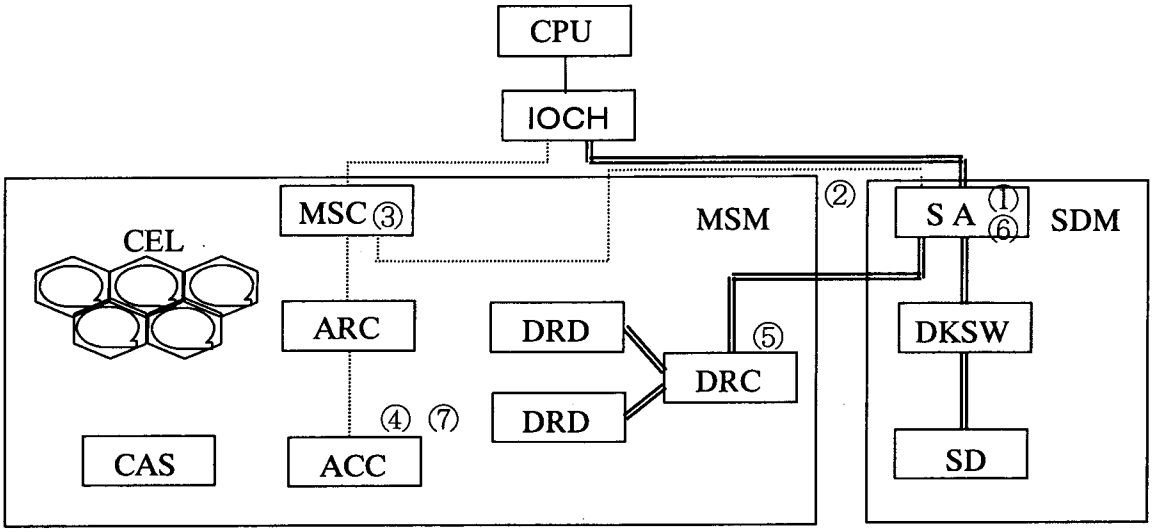
（２）ステージングディスク記憶部（ＳＤＭ）

ステージングディスク装置と呼ばれる磁気ディスク装置（ＳＤ）、ホストとＳＤ間、ＭＳＭとＳＤ間のデータ転送を制御するステージングアダプタ（ＳＡ）、およびＳＡ、ＳＤ間の接続装置であるＤＫＳＷから成る。

ホストからのデータアクセスがＭＳＳのこれら構成装置によって、どのように処理されるかの概要を示すと以下の通りである（以下数字は図4.2と対応している）。

---

[1] ＭＳＳを接続し、ＭＳＳを制御する計算機システムをホストシステムまたは単にホストと呼ぶ。



主な制御信号路　　主なデータ信号路

MSM:超大容量記憶部
MSC:超大容量記憶制御装置
ARC:アクセッサ制御装置
ACC:アクセッサ
CEL:カートリッジ収納庫
DRC:データ記録装置制御装置
DRD:データ記録装置
SDM:ステージングディスク記憶部
SA:ステージングディスク制御装置
DKSW:磁気ディスク切替装置

図 4.2　MSS機器構成

① ホストからのデータアクセス要求は、ＳＡにて分析される。すなわち、ホストからの要求は、仮想ディスク装置上のアドレスによって指示されるため、それを実際のＳＤ上のアドレスに変換し、そのデータがＳＤ上に存在するか否かをチェックする。
もし、データがすでにＳＤ上にあればそれをホストへ転送するが、それが無ければシリンダーフォールトとなり、ステージングを行うために②以下の処理を行う。

② ＳＡはＭＳＣに対しシリンダフォールトの発生を通知し、ステージング処理を要求する。

③ ＭＳＣは、まずシリンダフォールトを起こした仮想ディスクに対応する磁気テープカートリッジを制御情報に基づいて捜し出し、それが格納されるカートリッジセルのアドレスを見出す。

④　つづいて、アクセッサに対しそのカートリッジを空きＤＲＤへ移送するよう指示する。この動作をマウントと呼ぶ。

⑤　カートリッジがＤＲＤへ投入されると、磁気テープ上の必要なデータを捜し出し、ＤＲＤ、ＳＡ、ＳＤのパスを通してデータ転送を行う。１回の転送データ量はシリンダ[1]単位で、ｉシリンダ（ｉ＝１，２，…）となる。本評価のパラメータとして以降で用いる転送データ量は、この値を指している。

⑥　ＳＤへのデータ転送が完了した後、ホストからのアクセス要求に対する処理が再開され、データがホストへ転送される。

⑦　上のデータアクセスと並行して、ＤＲＤ上にあるカートリッジが再びアクセッサによってもとのカートリッジセルに格納される。この動作をデマウントと呼ぶ。

以上が基本的な動作であるが、モデル化に際し必要なスケジューリングに関する説明を以下に補足する。

⑧　上記⑦において、アクセッサに空きがなければ磁気データカートリッジはデータ記録装置ＤＲＤ上に残ったまま空きを待つ。すなわち、ブロッキング（無効保留）が生じる。

⑨　アクセッサには、マウントおよびデマントの２種類の処理要求が到着するが、その処理は以下のアルゴリズムに従う。

（ア）デマウント処理を優先

（イ）同種の処理相互は、先着順（ＦＣＦＳ）

（ウ）マウント処理は、アクセッサに空きがあり、かつデータ記録装置に空きがあるときに処理

上記（ア）は、⑧に示したブロッキングをできるだけ避けるためであり、（ウ）はデータ記録装置とアクセッサの間でのデッドロック状態[2]を避けるためである。

---

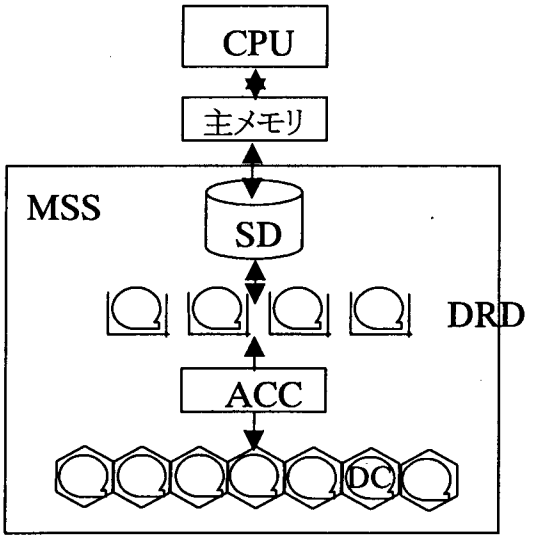
[1] データ転送の最小単位で約２５０KBに相当する。

[2] アクセッサがデータカートリッジを保持した状態で、マウントのためにデータ記録装置の空きを待つ一方で、データ記録装置は、デマウントのためにアクセッサの空きを待って、互いにブロックされている状態。

4.2.3　大容量記憶装置の待ち行列モデル

前節で述べたＭＳＳの構成の論理的な構成を図4.3に示す。また、その動作は図4.4に示す待ち行列ネットワークでモデル化される。ＭＳＳへのアクセス要求は、ネットワークへの客の到着として表現され、まず待ち行列Ｑへつながれた後、図の矢印に従ってアクセッサ、データ記録装置の順にサービスを受け、再びアクセッサでのサービスを受けて系から退去する。

このモデルの特徴は、待ち行列ネットワークの一部（図4.4において、点線で囲まれた部分、以下、部分系Ｓと呼ぶ）に存在しうる客の数が、データ記録装置の数Ｎ以下に制限されること、すなわち系内客数制限の条件をもつ点にある。これは、前述の⑨の（ウ）の項を表現したものである。また、データ記録装置の前に、待ち行列がないのは、系内客数制限により待ちが生じないためであり、一方、データ記録装置からアクセッサへ進む位置に待ち行列がないのは、⑧で示したブロッキングを表現するためである。



MSS:大容量記憶装置
SD　:ステージングディスク装置
DRD:データ記録装置
ACC:アクセッサ
DC　:データカートリッジ

図 4.3　MSSの構成

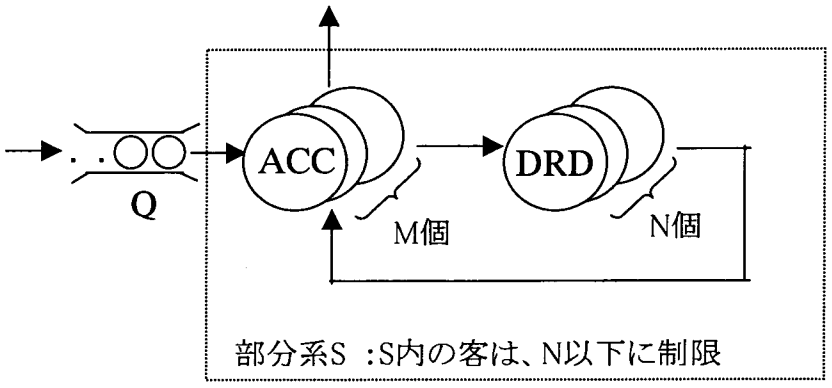


図 4.4 MSSの待ち行列モデル

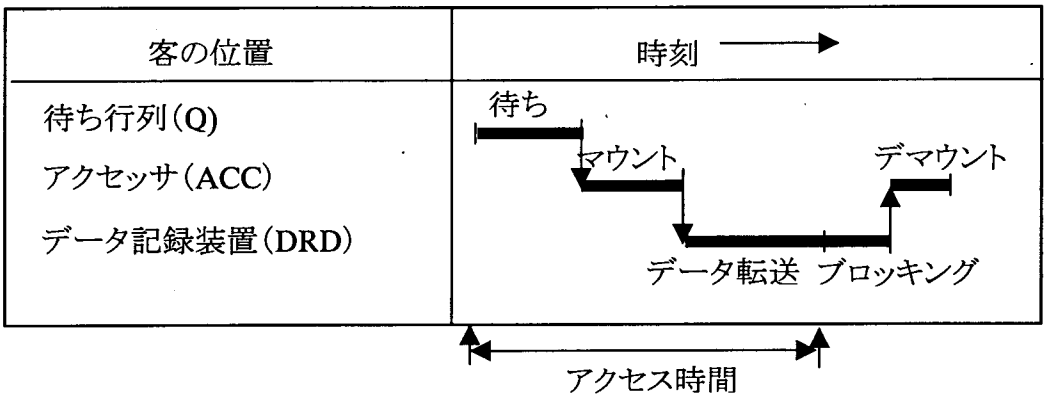


図 4.5 待ち行列モデルにおける客の動き

図4.5は、以上のモデル動作を時間の流れで示したものである。なお、アクセス時間は、客の到着からデータ記録装置におけるデータ転送の完了までの時間として定義する。

モデルでは、以下のような仮定を置く。

（1）客の到着は、到着率λのポアソン過程とする。

（2）アクセッサおよびデータ記録装置のサービス時間は平均が$1/\mu_1$および$1/\mu_2$の指数分布とする。

（3）サーバの数は、実在するＭＳＳの基本構成に合わせアクセッサ数Ｍは１に固定、データ記録装置数Ｎは１または２とする。

4.3 待ち行列モデルの解析

本モデルは系内客数制限があり、制限を越える客は待ち行列で待ち合わせる点に特徴がある。従来、系内客数制限をもつ待ち行列の研究は多数あるが、大部分が制限値を越える客は失われる（呼損）ことを仮定している。また、呼損とならず待ち合わせるモデルの研究もあるが、単一サーバの条件を仮定しているか[8],[20],[16]、あるいは複数サーバの場合には最初のサーバが空きにならないことを仮定している[11]。したがって、ここで提案したモデルにただちに適用可能な手法はないが、ここでは次の手順で解析した。まず、系の平衡方程式をたて、母関数を導入し母関数について解く。得られた母関数は、本来求めたい未知の確率を含んだ形となるが、確率の総和が１である条件および系の平衡条件から導かれる母関数の解析性の条件を用いて最終的な解を求める。

4.3.1 系の状態定義

系の状態は以下で定義する部分系Ｓの状態番号ｍと待ち行列Ｑにある客の数ｎの２字組（ｍ，ｎ）で表す。部分系Ｓ内には、データ記録装置サーバの数Ｎ（Ｎ＝１または２）を越えない客しか存在しないため、その状態は表4.1に示す有限個の状態しかとりえず、それらに状態番号ｍを付与する。表4.1では、部分系Ｓの状態をアクセッサおよびデータ記録装置の各サーバの状態の組合せで定義している。

表 4.1 部分系Sの状態定義

DRD数（N）＼状態 / 状態番号（m）	N=1		N=2		
	ACC の状態	DRD の状態	ACC の状態	DRD の状態	
1	0	0	0	0	0
2	-	-	0	0	1
3	M	0	M	0	0
4	D	0	D	0	0
5	-	-	M	0	1
6	-	-	D	0	1
7	-	-	M	0	B
8	-	-	D	0	B
9	0	1	0	1	1

凡例　-：存在しない,0：空き,1：DRDデータ転送中,M：ACCマウント中,
B：DRDブロッキング中,D：ACCマウント中

状態１および２では、部分系Ｓ内の客数は制限値より少なくかつアクセッササーバが空きであるためＱには客が存在しないのに対し、その他の状態ではＱ上に待ちが形成されうる。したがって、系の状態（m，n）は次の範囲に対して定義される。

$$\begin{cases} m=1\text{および}2\text{のとき} & n=0 \\ m=3,\ 4,\ \cdots,\ 9\text{のとき} & n=0,\ 1,\ 2,\ \cdots \end{cases}$$

なお、データ記録装置のサーバ数１または２によって部分系Ｓの状態数が異なるが、以下の記述ではサーバ数１では定義されていない状態（状態番号２，５，６，７および８）は無視するものとする。

4.3.2　状態方程式の導出

系には平衡状態が存在すると仮定し、状態（m，n）の定常確率を P(m，n）で表す。

各状態間の遷移は、次の三つの独立な事象、（１）客の到着、（２）アクセッサ・サーバでのサービス終了、（３）データ記録装置サーバでのサービス終了の発生時のみであり、これらの事象が同時に生起する確率は無視できると仮定すると、次の状態方程式（１）を得る。なお、データ記録装置（ＤＲＤ）の数が１のケースでは、未定義の状態を含む関係式を無視するものとする。

$$\lambda P(1,0) = \mu_1 P(4,0)$$
$$(\lambda+\mu_2) P(2,0) = \mu_1 P(3,0)+\mu_1 P(6,0)$$
$$(\lambda+\mu_1) P(3,0) = \lambda P(1,0)+\mu_1 P(4,1)$$
$$(\lambda+\mu_1) P(4,0) = \mu_1 P(8,0)+\mu_2 P(2,0)$$
$$(\lambda+\mu_1+\mu_2) P(5,0) = \lambda P(2,0)+\mu_1 P(3,1)+\mu_1 P(6,1)$$
$$(\lambda+\mu_1) P(3,k) = \lambda P(3,k-1)+\mu_1 P(4,k+1) \qquad (k \geqq 1)$$
$$\begin{cases} (\lambda+\mu_1) P(4,k) = \lambda P(4,k-1)+\mu_1 P(8,k) & (k \geqq 1, \text{DRD数}=2\text{のとき}) \\ (\lambda+\mu_1) P(4,k) = \lambda P(4,k-1)\,\varepsilon_k+\mu_2 P(9,k) & (k \geqq 0, \text{DRD数}=1\text{のとき}) \end{cases}$$
$$(\lambda+\mu_1+\mu_2) P(5,k) = \lambda P(5,k-1)+\mu_1 P(3,k+1)+\mu_1 P(6,k+1) \qquad (k \geqq 1)$$
$$(\lambda+\mu_1+\mu_2) P(6,k) = \lambda P(6,k-1)\,\varepsilon_k+\mu_1 P(7,k)+2\mu_2 P(9,k) \qquad (k \geqq 0)$$
$$(\lambda+\mu_1) P(7,k) = \lambda P(7,k-1)\,\varepsilon_k+\mu_2 P(5,k) \qquad (k \geqq 0)$$
$$(\lambda+\mu_1) P(8,k) = \lambda P(8,k-1)\,\varepsilon_k+\mu_2 P(6,k) \qquad (k \geqq 0)$$

$$\begin{cases} (\lambda+2\mu_2)P(9,k) = \lambda P(9,k-1)\varepsilon_k+\mu_1 P(5,k) & (k \geqq 0, \text{DRD数}=2\text{のとき}) \\ (\lambda+\mu_2)P(9,k) = \lambda P(9,k-1)\varepsilon_k+\mu_1 P(3,k) & (k \geqq 0, \text{DRD数}=1\text{のとき}) \end{cases}$$

ここで, $\varepsilon_k = 0$ (k=0 のとき), $\varepsilon_k = 1$ (k ≠0 のとき) (4.1)

4.3.3 母関数の導入

次式で定義される状態確率に関する母関数を導入する。

$$F_m(z) = \sum_{n=0}^{\infty} P(m,n)z^n \tag{4.2}$$

ただし、$|Z| \leqq 1$, $3 \leqq m \leqq 9$

（1）式より母関数に関する次の関係式を得る。

ＤＲＤ数が１の場合：

$$\begin{aligned} &(\lambda+\mu_2-\lambda z)F_9(z)-\mu_1 F_3(z)=0 \\ &(\lambda+\mu_1-\lambda z)F_4(z)-\mu_2 F_9(z)=0 \\ &z(\lambda+\mu_1-\lambda z)F_3(z)-\mu_1 F_4(z)=\mu_1(z-1)P(4,0) \end{aligned} \tag{4.3}$$

ＤＲＤ数が２の場合：

$$\begin{aligned} &z(\lambda+\mu_1-\lambda z)F_3(z)-\mu_1 F_4(z)=\mu_1(z-1)P(4,0) \\ &(\lambda+\mu_1-\lambda z)F_4(z)-\mu_1 F_8(z)=\mu_2 P(2,0) \\ &z(\lambda+\mu_1+\mu_2-\lambda z)F_5(z)-\mu_1 F_3(z)-\mu_1 F_6(z)=(\lambda z-\lambda-\mu_2)P(2,0) \\ &(\lambda+\mu_1+\mu_2-\lambda z)F_6(z)-\mu_1 F_7(z)-2\mu_2 F_9(z)=0 \\ &(\lambda+\mu_1-\lambda z)F_7(z)-\mu_2 F_5(z)=0 \\ &(\lambda+\mu_1-\lambda z)F_8(z)-\mu_2 F_6(z)=0 \\ &(\lambda+2\mu_2-\lambda z)F_9(z)-\mu_1 F_5(z)=0 \end{aligned} \tag{4.4}$$

(4.3) 式を母関数について解くと、未知の確率 P(4,0)を含んだ形となるが、確率の総和が１である条件を用いると次のとおり決定することができる。

$$\begin{aligned} &F_3(z)= \rho_1(2\rho_1+\rho_2-1)(\rho_1+1-\rho_1 z)(\rho_2+1-\rho_2 z)/g(z) \\ &F_4(z)=\rho_1(2\rho_1+\rho_2-1)/g(z) \\ &F_9(z)=\rho_2(2\rho_1+\rho_2-1)(\rho_1+1-\rho_1 z)/g(z) \end{aligned} \tag{4.5}$$

ただし、

$$g(z)=\rho_1^2\rho_2 z^3-\rho_1(2\rho_1\rho_2+\rho_1+2\rho_2)z^2+\{\rho_2(\rho_1+1)^2+\rho_1(\rho_1+2)\}z-1 \quad (4.6)$$

$$\rho_1=\lambda/\mu_1,\ \rho_2=\lambda/\mu_2 \quad (4.7)$$

一方、DRD 数が２の場合、すなわち（4.4）式を母関数について解くと次の（4.8）式を得る。

$$\begin{aligned}
F_3(z)&=\{z(\lambda+\mu_1+\mu_2-\lambda z)a(z)-\mu_1(\lambda+\mu_1-\lambda z)b(z)\\
&\quad-(\lambda z-\lambda-\mu_2)G(z)P(2,0)\}\ f(z)/G(z)\\
F_4(z)&=\mu_2\ \{G(z)P(2,0)+\mu_1 b(z)\ \}\ f(z)/G(z)\\
F_5(z)&=a(z)f(z)/G(z)\\
F_6(z)&=(\lambda+\mu_1-\lambda z)b(z)f(z)/G(z)\\
F_7(z)&=\mu_2 a(z)f(z)/\ \{(\lambda+\mu_1-\lambda z)G(z)\}\\
F_8(z)&=\mu_2 b(z)f(z)/G(z)\\
F_9(z)&=\mu_1 a(z)f(z)/\ \{(\lambda+2\mu_2-\lambda z)G(z)\}
\end{aligned} \quad (4.8)$$

ただし、

$$\begin{aligned}
a(z)&=(\lambda+\mu_1-\lambda z)^2\ (\lambda+\mu_1+\mu_2-\lambda z)(\lambda+2\mu_2-\lambda z)\\
b(z)&=\mu_1\mu_2\ (3\lambda+2\mu_1+2\mu_2-3\lambda z)
\end{aligned} \quad (4.9)$$

$$\begin{aligned}
f(z)&=f_2(z)P(2,0)+f_4(z)\ P(4,0)\\
f_2(z)&=\mu_1^2\mu_2-(\lambda+\mu_1-\lambda z)^2\ (\lambda+\mu_2-\lambda z)z\\
f_4(z)&=\mu_1^2(\lambda+\mu_1-\lambda z)(z-1)
\end{aligned} \quad (4.10)$$

$$\begin{aligned}
G(z)&=(\lambda+\mu_1-\lambda z)^2\ (\lambda+\mu_1+\mu_2-\lambda z)z^2a(z)-\mu_1(\lambda+\mu_1-\lambda z)^3 zb(z)\\
&\quad-\mu_1^3\mu_2-b(z)
\end{aligned} \quad (4.11)$$

（4.8）式に示すとおり、各母関数は本来求めたい未知の確率２個、P(2,0)およびP(4,0)を含む有理関数の形で得られる。この未知の確率は、確率の総和が１である条件と系の平衡条件から定めることができる。すなわち、次節で示すとおり系が平衡状態にあるときは、Fm(z)の分母G(z)はｚの単位円内に唯一の根z_1をもち、しかもそれは分子側の零点でもなければならない。このため、(4.10)式より次の条件式を得る。

$$f(z_1)\ =\ f_2(z_1)P(2,0)+f_4(z_1)P(4,0)=\ 0 \quad (4.12)$$

上式と確率の総和が１であることから得られる方程式を連立させて解くことにより、未知の確率を定めることができ母関数は完全に決定される。

4.3.4 特性値

母関数$F_m(z)$が得られたため、これから各種の特性値を得ることができる。

（1）アクセッサおよびＤＲＤの使用率（ρ_{ACC}，ρ_{DRD}）

表4.1において部分系Ｓが状態３から８にあるときはアクセッサが、状態２，５，６，７，８および９にあるときはＤＲＤが、それぞれ空きでない。よって、各装置の使用率は、これらの状態確率のサーバ当りの和を求めることによって次のとおり得られる。

$$\rho_{ACC} = \sum_{m=3}^{8} F_m(1) \tag{4.13}$$

$$\rho_{DRD} = \frac{1}{2} \sum_{m=2,5,6,7,8} F_m(1) + F_9(1) \tag{4.14}$$

（2）ＤＲＤのブロッキング発生率（P^B_{DRD}）

ＤＲＤのブロッキング発生率は、ＤＲＤがブロックされた状態にある確率である。これは表4.1に示す状態７および８にある確率であり、

$$P^B_{DRD} = F_7(1) + F_8(1)$$

で示される。

（3）平均待ち行列長および平均アクセス時間（$\overline{Q}$，$\overline{T}$）

待ち行列Ｑに待ちが作られるのは、状態３～９のときであり、平均待ち行列長Ｑは次式で得られる。

$$\overline{Q} = \sum_{m=3}^{9} \sum_{n=0}^{\infty} nP(m,n) = \sum_{m=3}^{9} \left[\frac{dF_m}{dz} \right]_{z=1} \tag{4.15}$$

一方、アクセス時間はＱでの待ち時間、アクセッサでのサービス時間およびＤＲＤでのサービス時間の和である。Ｑでの平均待ち時間は、リトルの定理によりＱ／λで得られるため、平均アクセス時間Ｔは次式となる。

$$\overline{T} = 1/\mu_1 + 1/\mu_2 + \overline{Q}/\lambda \tag{4.16}$$

4.4 系の平衡条件に関する考察

系が平衡状態にあるとは、全部は0でない状態確率P(m, n)が定まることである。つまり、状態確率を係数とする変数zの幅級数が、z＝1において1に収束する。0でない収束半径をもつ幅級数は、その収束域内で連続関数であるから、$F_m(z)$は|z|＜1においてzの連続関数となる。よって、系が平衡状態にある場合、母関数$F_m(z)$の分母が|z|＜1に零点をもつならば、分子側も同じ零点をもつ必要がある。

4.4.1 データ記録装置数が1の場合の平衡条件

(4.5) 式の分子の多項式は、いずれも|z|＜1の範囲内に零点をもたない。したがって、系が安定であるためには、(4.5) 式の分母g(z)が|z|＜1内に零点をもたないことが必要となる。このための条件を、スツルムの定理 [37] を用いて求める。すなわち、g(z)に関するスツルムの関数列を求め、それらのz＝1およびz＝－1における符号変化数が等しくなる条件を求めると、次が得られる。

$$g(1)<0 \quad \text{かつ} \quad g'(1)>0 \tag{4.17}$$

これを、(4.6)式により評価すると、

$$g(1) = 2\rho_1 + \rho_2 - 1 < 0 \tag{4.18}$$

$$\begin{aligned} g'(1) &= -\rho_1^2 - 2\rho_1\rho_2 + 2\rho_1 + \rho_2 \\ &= 3\rho_1^2 - 2\rho_1(2\rho_1 + \rho_2 - 1) + \rho_2 > 0 \end{aligned} \tag{4.19}$$

となり、(4.18)式が成立すると(4.19)式も成立するため、系の平衡条件は(4.18)式となる。

4.4.2 データ記録装置数が2の場合の平衡条件

ここでの結論を先に述べると、系が安定である条件は、(4.8)式で示される母関数の分母 G(Z)が、|z|＜1の領域で唯一の根をもつことである。この条件の必要性は、G(Z)＝0が|z|＜1の領域に1個以上の根をもつこと、およびもし G(Z)＝0が|z|＜1の領域に2個以上の根をもつならば、系は不安定になること、の二つのステップによって示す。

第1のステップは、フーリェの定理[37]を用いて、G(Z)＝0が－1＜z＜0の領域に、1個の実根をもつことを示す。すなわち、G(Z)の導関数列を求め、それらのz＝－1および0における符号変化数 V(-1)および V(0)を評価すると

$$V(-1)-V(0)=9-8=1$$

となる。これにより、$G(Z)=0$は$-1<z<0$の領域に実根を唯一持ち、よって$|z|<1$の領域では、1個以上の実根をもつことが示された。

次に、$G(Z)=0$が$|z|<1$の領域にk個（$k\geqq 2$）の根z_n（$n=1,2,\ldots,k$）を持つと仮定する。系が安定ならば、母関数の分子側の多項式$f(z)$も同じ零点をもつ必要があり、(4.10)式から次の条件を得る。

$$f(z_n)=f_2(z_n)P(2,0)+f_4(z_n)\cdot P(4,0)=0 \quad n=1,2,3,\ldots,k \tag{4.20}$$

(4.20)式は、未知数 $P(2,0)$および $P(4,0)$に関する斉次方程式であるが、未知数が2個に対し方程式がk個（$k\geqq 2$）であり、その解は $P(2,0)=P(4,0)=0$となる。すると、(4.10)式より $f(z)=0$となり、よって$Fm(z)=0$となるため、すべてのm, nに対して $P(m, n)=0$となる。これは、系が不安定であることを意味しており、最初の仮定と矛盾する。よって、系が安定ならば$k\leqq 1$でなければならず、第1ステップの結果と合わせて、$k=1$が導かれる。

一方、もし$G(Z)=0$が$|z|<1$の領域に唯一の根をもつとすると、(4.20)式を満足する0でない $P(2,0)$および $P(4,0)$の組が得られる。以後は、4.3節で述べた手順で$P(m, n)$を得ることができ系は安定となる。

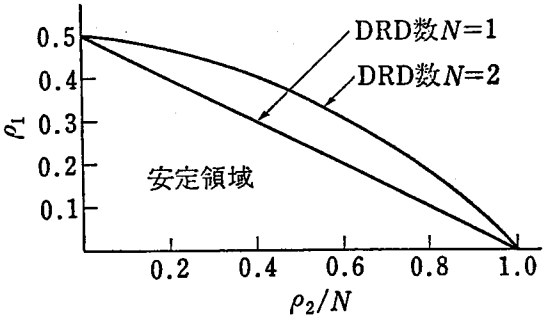


図 4.6　系の安定領域

図4.6に、ここで得た系の安定領域を示す。ρ_1すなわちλ/μ_1が0.5を越えないのは、1回のデータ転送のためにアクセッサには2回の処理要求（マウントおよびデマンウント）が課せられるためである。

4.5 数値計算例および実測との比較

4.5.1 数値計算の前提

データ記録装置（ＤＲＤ）の実際のサービス時間は、データ転送前後の準備および後処理に要するほぼ一定の時間と、データ量に比例したデータ転送時間の和である。これに対し、モデルではサービス時間を全体で一つの指数分布に仮定しているため、次式で得られる値を平均とする指数分布として数値計算を行う。

$$1/\mu_2 = T_c + D/v$$

ただし、T_c：前後処理時間に相当する一定値

D ：データ転送量

v：ＤＲＤのデータ転送速度

一方、アクセッサのサービス時間は、カートリッジがランダムに選択されると想定すると指数分布で近似できる。表4.2に数値計算で用いる値をまとめて示す。

表 4.2 数値計算に用いるパラメータ値

項　目	記　号	値
ACC の平均サービス時間	$1/\mu_1$	4 sec
DRD の事前,事後処理時間	T_c	5 sec
DRD のデータ転送速度	v	250 kB/sec

4.5.2 スループットおよびアクセス時間

スループット（λ）をしらみつぶしに計算し、平均アクセス時間が１５，２０，３０，６０秒となる値を選んでプロットしたものを図4.7，4.8に示す。また、図には次の手順で得られる限界スループットも合わせて示す。すなわち、図4.6の縦軸（ρ_1）と横軸（ρ_2/N；NはＤＲＤ数）の比は、次式で示される。

$$\rho_1 / (\rho_2/N) = (\lambda/\mu_1) / (\lambda/\mu_2 N) = N\mu_2/\mu_1$$

データ転送量を固定するとμ_2が定まり、上の値が決まる。そこで、図4.6のグラフと上式で示される直前との交点を求め、そこから限界スループットを得る。

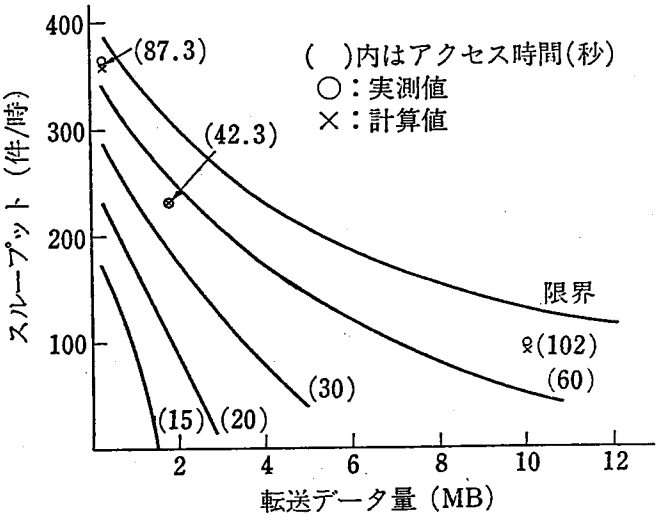


図 4.7　MSSのスループット(DRDが 2 台のケース)

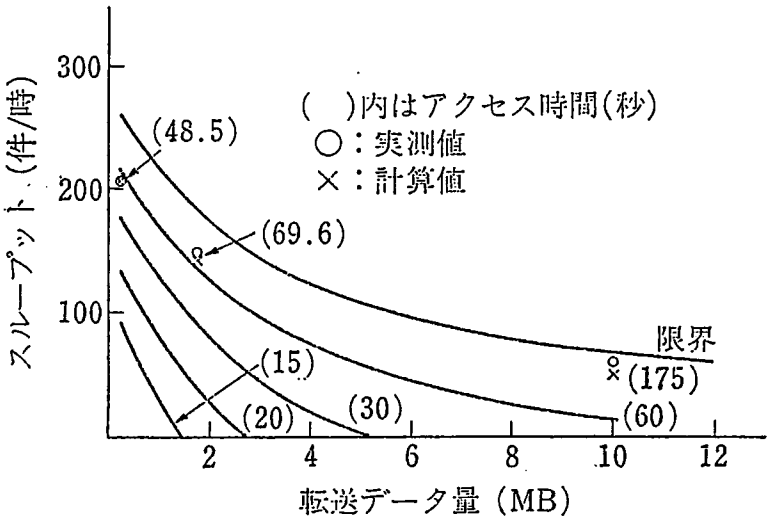


図 4.8　MSSのスループット(DRDが 1 台のケース)

4.5.3　フォールバックの影響

図4.9は、ＤＲＤ数が１つのときのスループットを、ＤＲＤ数が２のときのスループットに対する比で示している。

限界スループットについて見れば、転送データ量が大きくなるにつれ、スループット比は0.5へ漸近し、ＤＲＤ数が半減した影響が直接現れている。しかし、転送データ量が少ないとスループット比は0.67程度であり、影響は比較的軽くなる。これは、スループットを支配する装置がアクサッサであり、それが転送量の増加に伴い、ＤＲＤへ移る[22]ためと考えられる。これに対し、アクセス時間を一定としたときのスループット比は、データ転送量が少ないと0.6の近傍であるが、以後急速に低下しており、前述の限界スループットにおける傾向とは異なっている。ＤＲＤの障害によるフォールバック運転においてアクセス時間の大幅な延びを避けるには、相当の入力規制を行う必要がある。

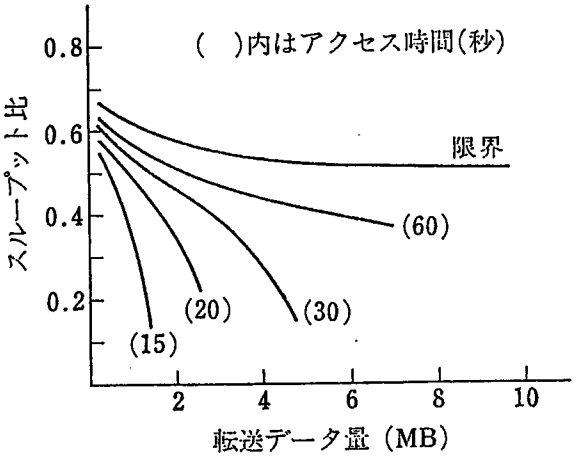


図 4.9　スループット比(DRD 数　1 対 2)

4.5.4　アクセッサとデータ記録装置の使用率

図4.10と図4.11にＡＣＣとＤＲＤの使用率とＤＲＤのブロッキング発生率をスループットとの関係で示す。図4.10は転送データ量が１．５Ｍバイトのケース、図4.11は４．５Ｍバイトのケースを示している。図4.10では全領域でＡＣＣの使用率がＤＲＤの使用率より大きいのに対し、図4.11ではその逆となっている。これは、データ転送量が少ない場合には、カートリッジを移送するためのアクセッサがシステムのボトル

ネックになっているが、データ転送量が多い場合は、ＤＲＤがボトルネックになっていることを示している。このことは、同様にＤＲＤのブロッキング発生率が図4.10より、図4.11のほうが大きいことからも分かる。

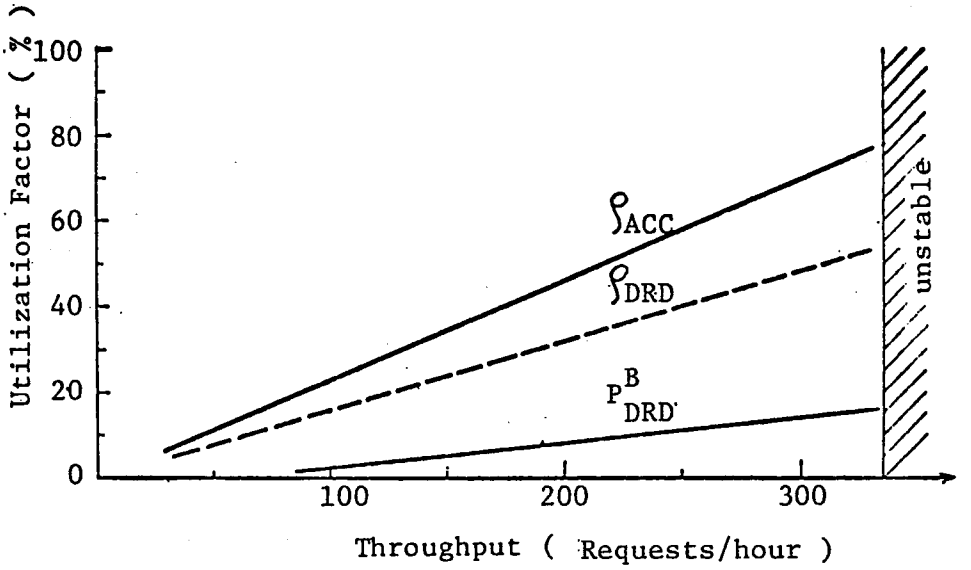


図 4.10　装置利用率(転送データ量＝1.5 MB)

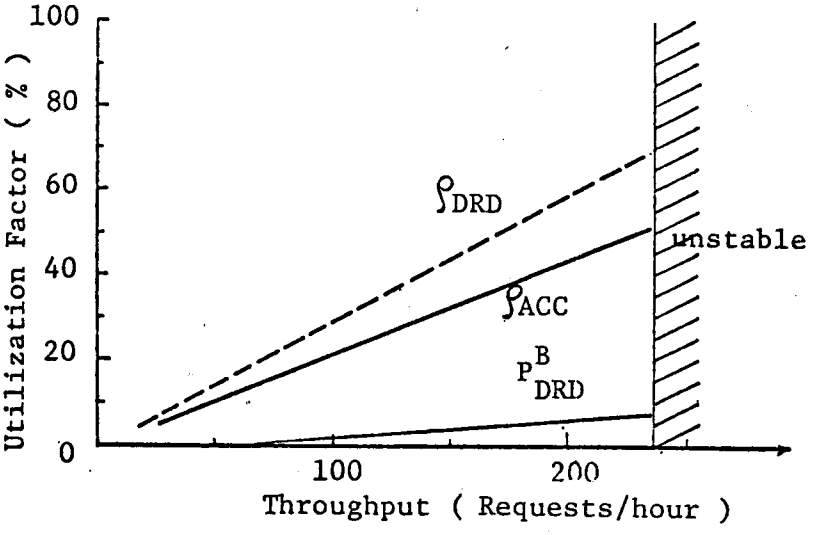


図 4.11　装置利用率(転送データ量＝4.5 MB)

4.5.5　実測値との比較

実測は、指定された大きさのファイルをステージングするベンチマークジョブを連続的に実行させながら、ＯＳのモニタ機能によってアクセス時間およびスループットを測定する方法で実施した。用いたファイルの大きさは、１，７，４０および９６シ

リンダ（0.25、１.75、１０および２４ＭＢ）の４種類である。測定ごとにベンチマークジョブのファイルの大きさを上記のいずれかの値に固定したため、ＤＲＤのサービス時間は一定時間に近い。この点で、モデルにおける指数分布の仮定と完全には対応していない。

表4.3に実測結果およびモデルでの計算値との対応を示す（同一のデータを、前出の図4.7と図4.8にも示した）。モデルでの計算は、転送データ量（実測におけるファイルの大きさ）およびアクセス時間を実測と同一条件としたときのスループットを求めたものである。結果は、実測値とかなりよい一致を示しており、本モデルは十分実用に耐えるものであることを裏づけている。

表 4.3 実測結果とモデル計算値の比較

条件			実測値／計算値	
ACC 数	DRD 数	転送データ数	アクセス時間（秒）	スループット（回／時間）
1	2	0.25	87.3/87.9	374/357
		1.75	42.3/42.2	232/231
		10.0	102.2/102.7	98/94
		24.0	242.9/241.3	42/49
1	1	0.25	48.5/48.9	206/207
		1.75	69.6/69.8	147/142
		10.0	175.0/177.1	60/50
		24.0	343.2/337.7	31/23

なお、本章で示したモデルを解析するためのプログラムは、ＦＯＲＴＲＡＮで約３００ステップ、処理時間はたとえば５０ポイントのデータを収集するのにＤＩＰＳ－１で、ＣＰＵ時間約５秒程度である。このように、本稿で示した方法はシミュレーションや近似手法に比べはるかに簡便でかつ精度の高い結果を得ることができる実用性の高いモデルであるといえる。

4.6 結言

本章では、大容量記憶装置を効率的な制御するためのトラヒック設計の方法を提案し、それに基づく設計結果の妥当性、制御方式の有効性を定量的に実証した。

具体的には、ＭＳＳの特性を評価するための動作モデルとして系内客数制限を持つ待ち行列モデルを提案し、その平衡状態における各種特性値を解析的に得る方法を示した。また、系の平衡条件を明らかにし理論的な限界スループットを得るとともに、具体的な特性としてデータ量とアクセス時間およびスループットの関係を明らかにした。さらに、これらの解析結果を実機における測定結果と対比し、よく一致することを示した。

この結果により、NTT 公衆データ通信システム（DEMOS サービス）に新たな追加サービスとして、ＭＳＳを情報蓄積の装置として利用するファイル保管庫サービスを実現することとした。このサービスは、利用に先立ってファイルを保管庫から取り出すコマンドを新設し、そのコマンドの実行にはＭＳＳへのアクセスが必要なため 10 数秒の時間が必要であるが、それ以降のファイルの利用は通常の磁気ディスクとほぼ同じ速さでアクセスでき、利用料金は磁気ディスクを用いたサービスよりも格段に安価に提供するものである。このために、本章の結果に基づいて設計したＭＳＳ制御機能をＤＩＰＳ用ＯＳに実装した[50]。この結果、長期間、大量の情報を保管するサービスとして多くの利用者に利用された。

また、大量のデータを一括して入出力するような利用形態はＭＳＳの特性に適合し、さらに媒体操作の手作業が不要なため省力化に大いに効果があるため、ＮＴＴ社内システムのリモートバッチ処理データの蓄積やファイル・バックアップ処理に導入を図った。ただし、最少のデータ転送量である１シリンダの場合でも平均10数秒の時間が必要であり、トランザクション処理システムのオンラインサービスへの導入は見送ることとした。

第５章

トランザクション制御方式のトラヒック設計

5.1 緒言

本章では、これまでにジャーナル処理方式や大容量記憶装置など個別的に示してきたトラヒック設計法をさらに高度化し、より広い設計問題に適用可能なトラヒック設計法に一般化することを試みる。

これまでも、計算機システムのトラヒック設計の手段として待ち行列理論が数多く用いられてきている[57]。しかし、待ち行列理論においてはモデルを構成する窓口（サーバ）は、すべてお互いに独立して動作することが仮定されている。従って、待ち行列モデルを用いた計算機システム評価に関する研究の大半は、ＣＰＵや入出力装置などハードウェアそのものを待ち行列の窓口に対応させ、処理要求が直接これらハードウェアに待ちを作るようなモデルを用いている。

ところが、本論文の第３章のジャーナル処理や第４章の大容量記憶装置の評価において示したように、計算機システムのモデル化にあたっては、ある窓口を保留した状態で、別の窓口を同時に保留するような、窓口相互間が独立では無い動作を表現しなければならないケースが数多く存在する。特にソフトウェアを資源とみなして、ハードウェア資源とあわせて総合システムとして評価するには、この様な動作をモデル化できることが必須となる。何故ならば、ソフトウェア資源は仮想的なものであり、その実態はハードウェア資源が担うという相互関係が存在するからである。より具体的に述べれば、ソフトウェア資源のサービス時間は、実はＣＰＵや記憶装置などのハードウェア資源におけるサービス時間やそのための待ち時間から構成されるという相互関係がある。

従来のようなハードウェア資源だけあるいはソフトウェア資源だけを単独に扱うモ

デルでは不十分であり、両者を統合し相互の作用を含めて解析できるトラヒック設計手法が必要である。

本章では、このような要求に応え得るものとして、ハードウェアを窓口とする待ち行列モデルにおいて、その一部または全体の中に存在し得る客数を一定数に制限することによってソフトウェア資源を表現する、系内客数制限モデルを提案する。本章の前半では、外からの客の到着が無い閉じたモデルを用いてトランザクション処理システムの設計において重要となるタスク構成を評価する方法を提案し、その解析例により本モデルが前述の要求に応える有効なものであることを示す。

さらに後半ではそのモデルを発展させ、ソフトウェア自身に対する競合や輻輳を、ハードウェア資源と関係づけて総合的に取り扱うための一般的なトラヒック設計手法を提案する。そして、そのモデルの解析の方法を与え、系が安定であるための必要十分条件を理論的に明らかにする。また、解析手法に基づく、数値計算の手順を与えいくつかの計算例を示し、本モデルの有効性を示す。

5.2 トランザクション処理システムのタスク構造

第２章で述べたとおり、トランザクション処理システムは定型的で比較的単純な業務を効率よく処理することに特色がある。そして、業務が定型的であることを利用して、処理に必要な計算機資源をあらかじめ確保しておき、処理要求が到着するとただちに処理が開始できるような方法が採用され、処理の効率化が図られる。[33],[65]

たとえば、業務処理プログラムの大部分は、あらかじめメモリ上にロードされているため、処理を開始するにあたってプログラムをロードするためのオーバヘッドは無い。また、処理を実行するタスクもあらかじめ固定数だけ起動され、電文の入力を待っている。したがって、空きのタスク（処理すべき電文が無く、電文の入力を待っている状態のタスク）がある限り、電文が入力されるとただちに処理が開始され、タスクの起動や初期化のオーバヘッドが無い。しかし、空きのタスクが無いと入力された電文は処理待ち行列につながれ、タスクの空きを待つことになる。従って、あらかじめ起動しておくべきタスクの数の設計はシステムの性能に影響を与える主要な要因の一つである。

これに対し、会話型処理システムでは、処理要求が発生すると、それに対して動的にタスクが作られ、割付けられるのが一般的であるため、上で述べたトランザクション処理システムの場合と異なり、タスクの存在はシステムの性能に大きな影響を与えない。したがって、タスクの存在を特に意識せず、ハードウェアを窓口とした待ち行

列モデルを用いても十分システムの解析が可能である[49]。

図5.1に、具体的なトランザクション処理システムのタスク構造を示す。第２章で考察したとおり、端末との交信処理と業務処理（たとえば銀行システムの場合では、預金処理や、為替処理などを指す）を並行して実行するために、それぞれの処理は独立した別のタスクで実行される。即ち、端末との交信処理は、サービス管理タスク（ＳＭＴ）が一括して実行し、一方、業務処理は複数のサービス処理タスク（ＳＰＴ）が実行する。

回線を通して送られてくるデータは、ＳＭＴが受信し、逐次電文として組立て、処理待ち行列につなぎ込む。この時、ＳＰＴに空きがあれば、そのＳＰＴに電文の処理を依頼する。

一方、ＳＰＴ上の業務プログラムは受取った電文をもとに、必要な業務処理を行い、その結果を電文として組立て端末への出力を依頼する。この出力依頼は送信待ち行列にながれ、ＳＭＴで順に送信処理がおこなわれる。

この時タスクの構成、つまりサービス管理タスク（ＳＭＴ）およびサービス処理タスク（ＳＰＴ）をそれぞれ何個用意すべきかは、業務処理の処理負荷や全体のトラヒック、用いるハードウェアの性能などに依存し、具体的なトランザクション処理システム毎に設計すべき重要な問題である。このためのトラヒック設計手段の確立は必須である。

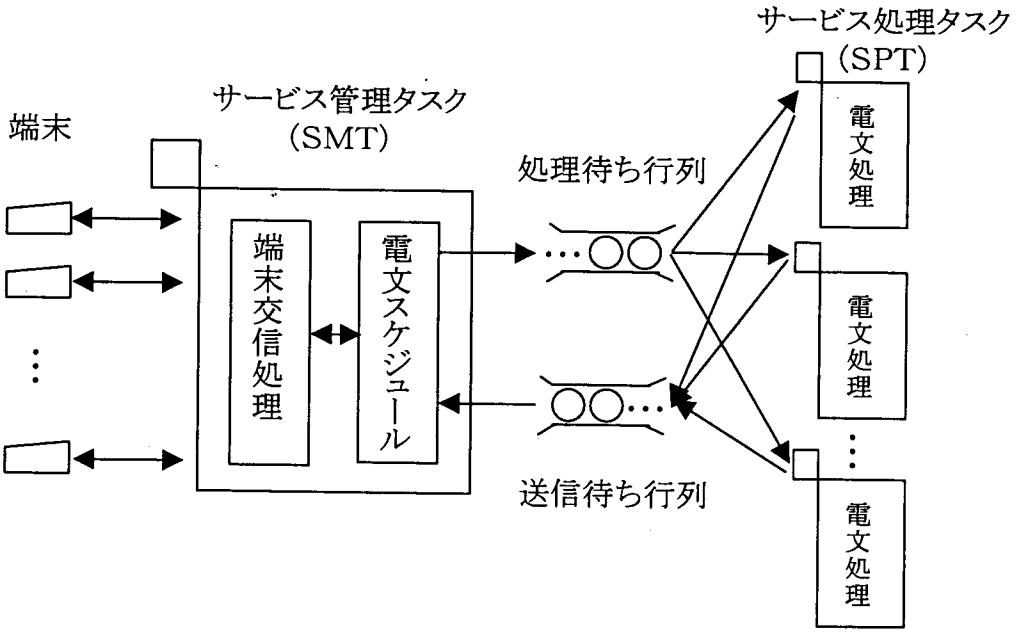


図 5.1 トランザクション処理システムのタスク構造

5.3 タスク構造のモデル化

トランザクション処理システムのタスク構造を待ち行列モデル化する。そのために、まずタスクを待ち行列上でどの様に表現するかについて考察した後、トランザクション処理システムのタスク構成全体のモデル化について述べる。

5.3.1 タスクのモデル化

タスクとは、高度な多重処理を行うシステムを統一的に制御するために導入されたソフトウェア上の概念である。したがって、それはあくまで仮想的なものであり、その実体はハードウェアが担っている（図5.2）.

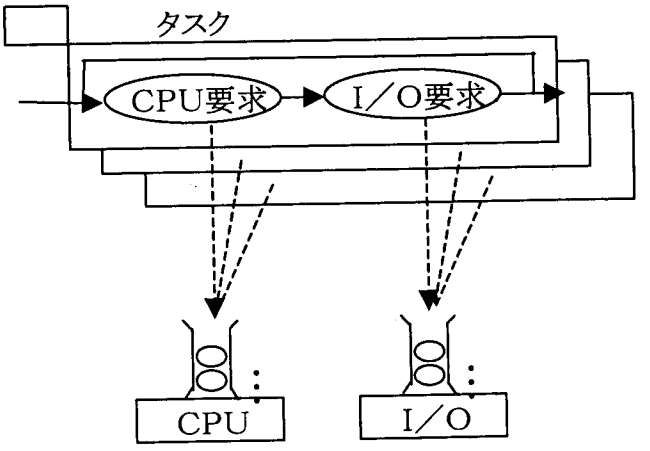


図 5.2 タスクの動作概念図

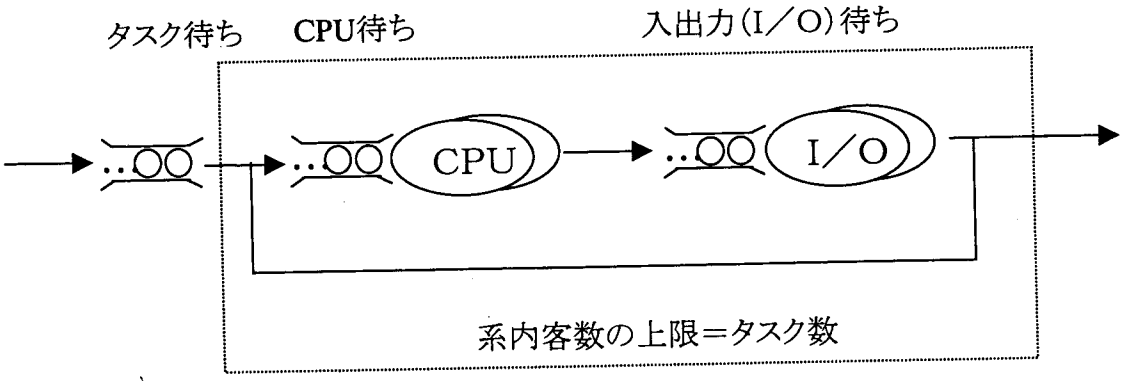


図 5.3 タスクの待ち行列モデル

すなわち、タスクのサービス時間は実はハードウェアにおける処理時間やその待ち行列から構成されることになる。一方、ハードウェアが空きであって、処理要求があっても、タスクに空きが無ければ、そのハードウェアは使用されず空きのままである。この様に、ハードウェア資源と干渉しあうタスクの動作を、系内客数制限モデルによって表現する。すなわち、従来と同様、ハードウェアを窓口とする待ち行列モデルを構成し、その上に部分系を定義し、この部分系の中に存在し得る客の数をタスク数に制限することによってタスクの存在を表現する。たとえば、図5.2に対応する待ち行列モデルは、図5.3の様に表現される。つまり、客が到着した時に、点線内に存在する客の数がタスク数に達していた場合には、たとえ先頭の窓口に空きがあったとしても、系外の待ち行列で待ちに入る。系内の客は、各窓口を巡回した後系外へ退去するが、それによって系内の客数が減少するため、系外の待ち行列に客があば、それが系内へ進むことになる。なお、客数制限を有する待ち行列は従来からもあるが、その多くは制限を越える客が損失となるモデルであり、本モデルはそれが系外で待ち合わせる点に特徴がある。

5.3.2 トランザクション処理システムのタスク構成のモデル化

以上のタスクのモデルに基づいてトランザクション処理システムのタスク構成全体をモデル化すると図5.4となる。

ここでは、ハードウェアとしてＣＰＵおよび２種類の入出力装置（ＩＯ１，ＩＯ２）、タスクとしてはＳＭＴとＳＰＴの２種類をそれぞれモデル化している。図5.4に示すとおり、部分系１，２を定義し客数制限を設けることによって、それぞれＳＭＴおよびＳＰＴを表現している。ＣＰＵに対する待ち行列は、２種類（ＳＭＴの待ち行列：Ｑ１１およびＳＰＴの待ち行列：Ｑ２１）あるが、実際のシステムにおいては、ＳＭＴのＣＰＵ割付プライオリティをＳＰＴのそれよりも高く設定するのでモデルにおいても、Ｑ１１をＱ１２より優先する。

ＳＭＴ上の客（この場合トランザクションに相当する）は、ＣＰＵおよびＩＯ１を巡回して処理を終わると、ＳＰＴ待ち行列（Ｑ２０）へ進む。これが、実システムにおける処理待ち行列に相当する。Ｑ２０上の客は、部分系２の客数が減少、これはすなわちＳＰＴの空きにあたるが、その契機に部分系２へ進みＣＰＵおよびＩＯ２を巡回する。処理が終了したなら、客は部分系２を抜けて、ＳＭＴ待ちであるＱ１０へ進む。これは、実システムと対比させれば、ＳＰＴでの業務処理が完了し、応答電文の出力依頼が、送信待ち行列へつなぎ込まれたことに対応する。

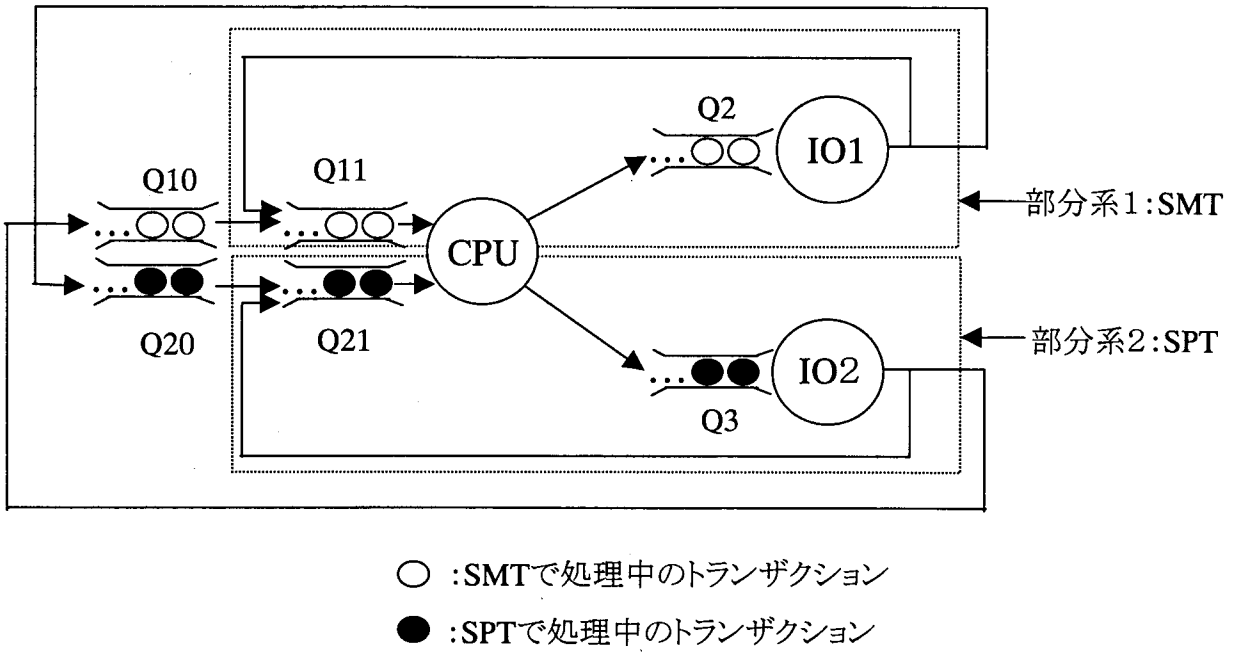


図5.4 トランザクション処理システムの待行列モデル

5.3.3 モデルの解析

各窓口および系外の待ち行列中に存在する客の数の組によってシステムの状態を定義する。各窓口のサービス時間に指数分布を仮定し、状態間の遷移をならべ挙げることによって、各状態の平衡確率に関する平衡方程式を得る。本モデルは、閉じた待ち行列であり、全体の客の数が有限に固定されるため状態数は有限となり、平衡方程式は状態確率を未知数とする連立一次方程式となる。これを数値計算によって解くことにより、各状態確率を求め、それをもとに各窓口の使用率、スループットなどシステムの各種特性値を得ることができる[38].

5.4 タスク構造の性能解析

ここでは、本章で提案したモデルを用いた具体的な解析例を示し、モデルの有効性についての評価を試みる。

5.4.1 タスク数と応答時間の関係

サービス処理タスク（ＳＰＴ）の数の変動がシステムの応答時間に対してどの様な影響を与えるかを評価したものが図5.5である。この時、ハードウェアの性能や構成およびサービス親タスク（ＳＭＴ）の数は固定している。図から明らかな様に、タスク

数が少ないと、タスクがシステムのボトルネックとなり、応答時間が非常に長くなるが、ＳＰＴ数を増加させるにつれて、システムのボトルネックはハードウェア自身に移る。図5.5に示したケースでは、タスク数を５以上にしても性能改善の効果はなく、逆にタスク数の増加によるメモリ量の増加などを招くことになる。

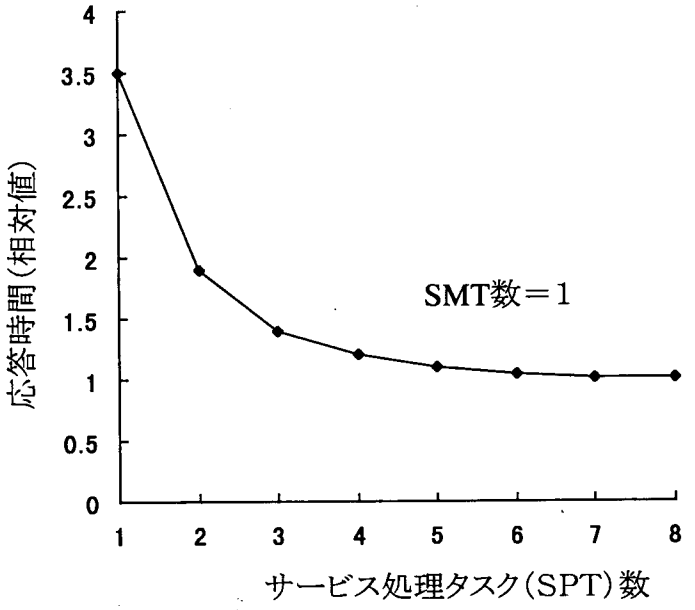


図 5.5 サービス処理タスク数と応答時間

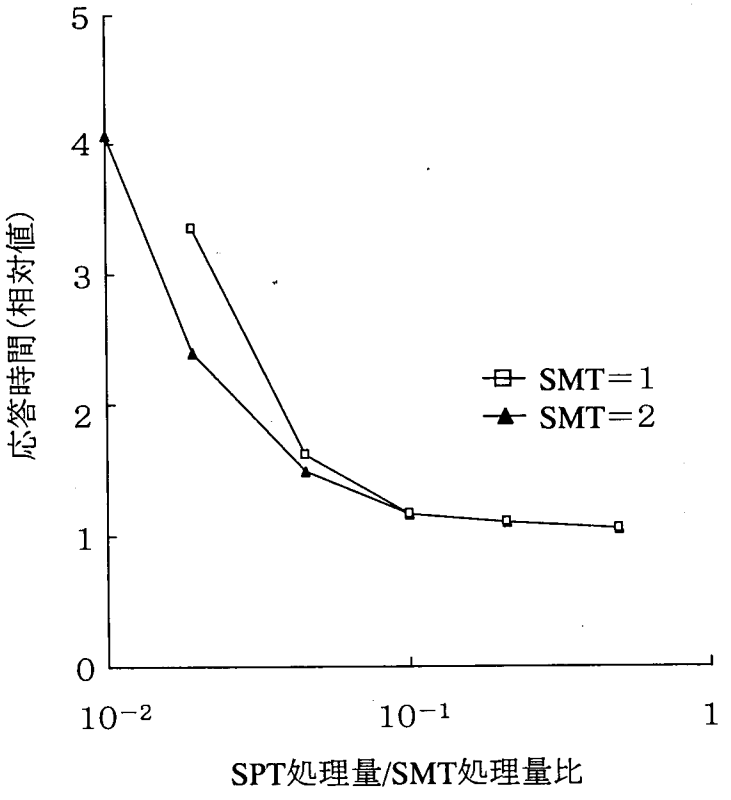


図 5.6 サービス管理タスク(SMT)数と応答時間

上記の例は、ＳＰＴがネックとなる例であるが、ＳＭＴとＳＰＴでの処理量の配分を変化させてみれば、ＳＭＴがボトルネックとなる領域が当然存在する。この点をモデルを用いて観測した例が図5.6である。すなわち、総処理量を固定し、ＳＭＴとＳＰＴでの処理の配分を変化させた時の応答時間への影響を示している。図ではＳＭＴ数が１の場合と２の場合について示しており、その他のハード・ソフトの構成は同一である。当然のことながら、ＳＭＴの処理分担が大きくなればなる程、ＳＭＴ数による応答時間のちがいが大きくなり、ＳＭＴがボトルネックとなることを示している。

この様に、本モデルを用いることにより、与えられたハードウェア構成のもとでの最適なタスク構成を得ることができる。

5.4.2 CPU台数とスループットの関係

図5.7はＣＰＵ台数とスループットの関係をいくつかのタスク構成に対して示したものである。タスク構成は、図中に示すとおり３つのケースを取りあげており、ケース１を基準に、ＳＭＴ数を増加させたものをケース２，ＳＰＴ数を増加させたものをケース３としている。

ケース１の場合、ＣＰＵ数を増加させハードウェアの能力を強化してもスループットの向上は、十分得られない。また、ＳＭＴ数を増加させても（ケース２）、スループットに変化はほとんど無い。これに対して、ＳＰＴ数を増加させると（ケース３）、ＣＰＵ数の増加に見合ったスループットの向上が観測される。しかし、これもＣＰＵ数が４台以上になると、飽和状態となり、再びタスクにボトルネックが現れてくる。

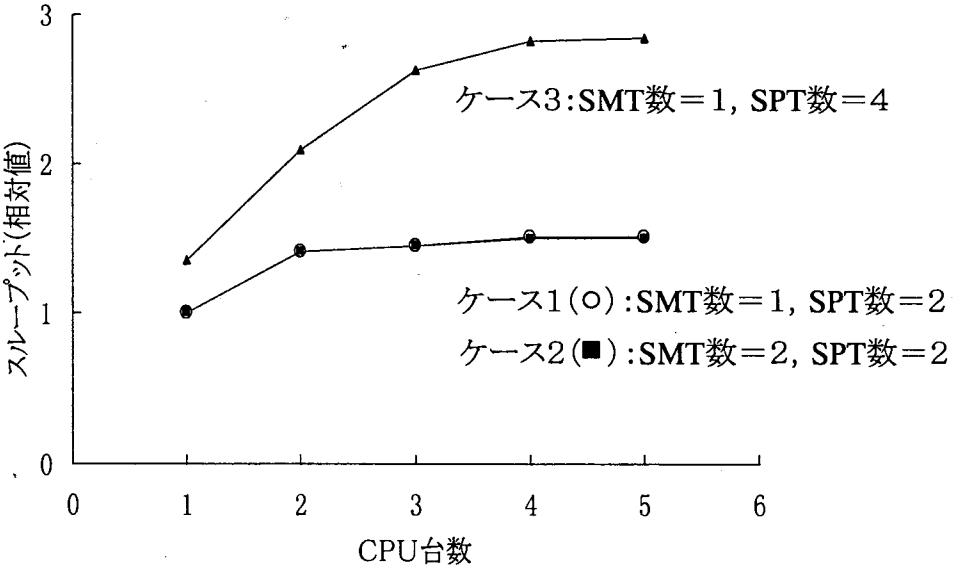


図 5.7 CPU台数とスループット

この様に、システム構成のパラメータの変動によって、性能上のボトルネックがある時はソフトウェアになったり、ある時はハードウェアとなったりする様子を的確にとらえることが可能である。ハードウェア構成とソフトウェア構成をそれぞれ別の独立したモデルで評価した場合には、この様なハード・ソフトの相互干渉を把握することは困難である。

5.4.3 シミュレーションとの比較

ここでは、提案したモデルの実用性を確認するために、実際のトランザクション処理システムのシミュレーション結果との比較を行う。

対象システムは、端末から投入された電文の内容に基き、センタに存在するデータベースを参照更新し、その結果を応答電文として端末へ返す、一問一答形式の問い合せ処理を行うシステムである。シミュレーションは、計算機専用シミュレータを用いて、ハードウェアおよびソフトウェアを忠実にモデル化したものであり、ここでのテーマであるタスクはもちろんのこと、ハードウェアの割込み処理やマルチプロセッサにおけるロック処理などが実際に即してモデル化された精密なものである[56].

実際のシステムにおいては、ＳＰＴ上で処理される業務プログラムは、磁気ディスク装置の他に、磁気テープ装置にもアクセスする。シミュレーションにおいては、これらはそのままモデル化されているが、解析モデルの場合、ＳＰＴでは１種類の入出力装置しか扱えないため、各装置のアクセス時間をアクセス回数で重み付けした平均値をアクセス時間として持つ仮想的な入出力装置が存在するものとしてモデル化した。

以上の様な仮定のもとで得た、ＣＰＵ使用率およびセンタ内処理時間とスループットの関係を図5.8および図5.9に示している。両図に示す通り、ほぼ１０％程度の差で結果が一致している。ここで用いたシミュレーションでは、モデル記述が約２ｋステップ、１ポイントのデータ収集のためには、実時間に比べ数倍から数１０倍の計算機時間が必要である。これに対し、解析モデルの場合、データを投入するだけで結果が得られ、１ポイントのデータ収集は、DEMOS-Eを用いて数秒程度で可能である。

本解析モデルは、非常に簡便にデータを得ることができるため、様々なケースを比較検討しながら、ハード・ソフトを含めたシステム全体のアーキテクチャを決定する様な場合などに、大いに有効である。

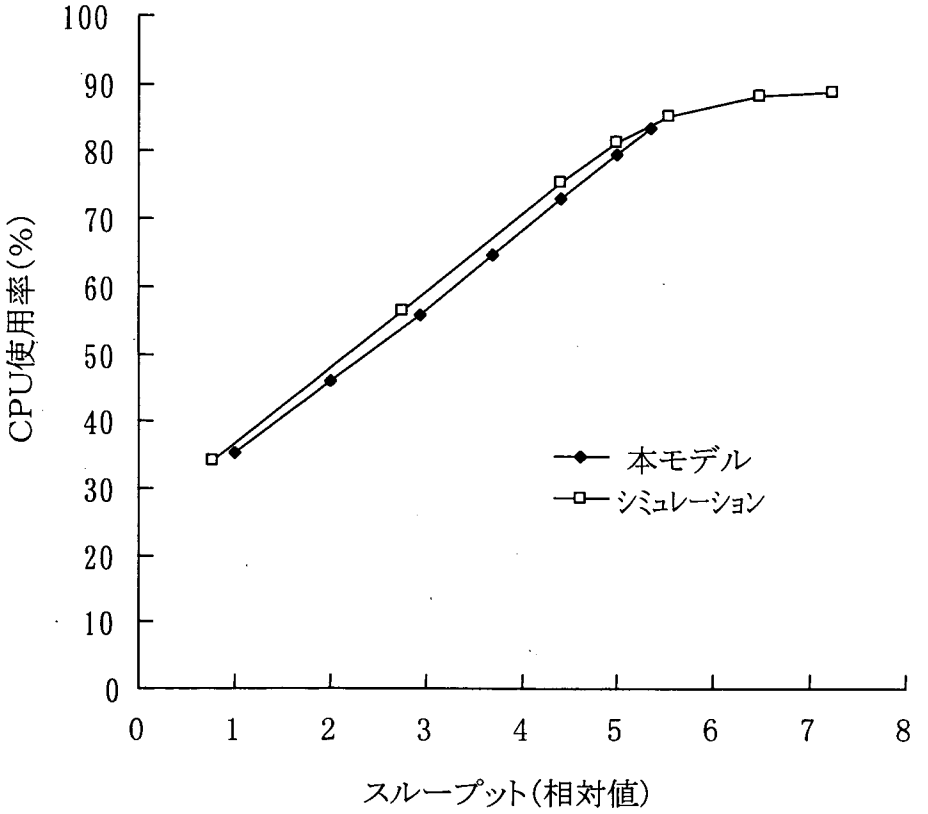


図 5.8 スループットとCPU使用率

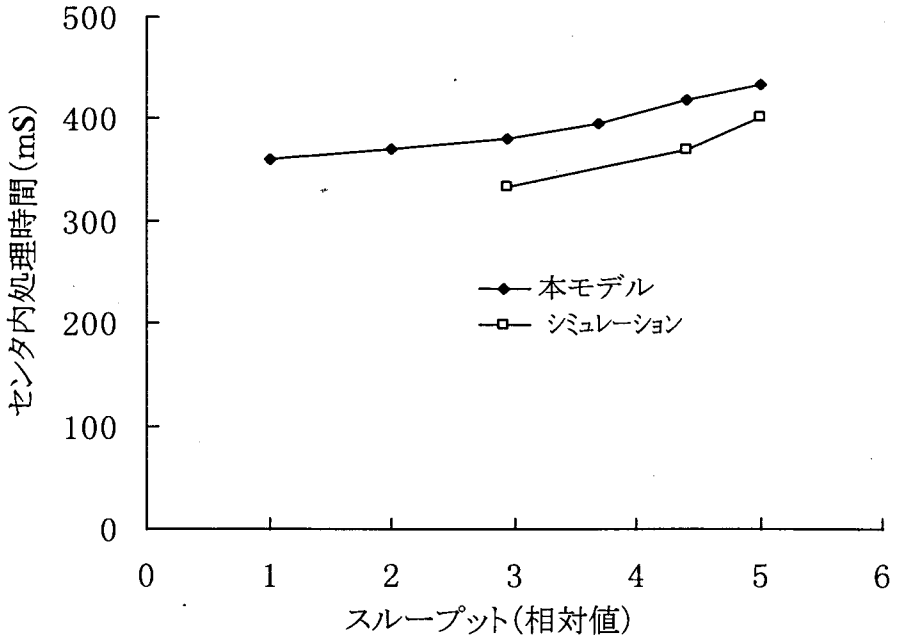


図 5.9 スループットとセンタ内処理時間

5.5 タスクモデルの一般化

本節以降では5.3節で与えたタスクモデルを拡張し一般的なモデル化と解析を試みる．ここで扱うモデルは5.3.1のタスクモデルを閉モデル（Closed Queuing Network Model)から開モデル（Open Queuing Network Model)に拡張したもので，図5.10に示すものである．図のモデルの特徴は，点線で囲んだ系内に存在し得る客数が一定値以内に制限され，それを越える客は待行列Q_0において待ち合わせる点にある。この意味で本モデルを系内客数制限モデルと呼ぶ。点線内の客数が制限値より少なくなればその時にQ_0で待ち合わせていた客はQ_1へ進むことになる。

実際の計算機システムと対比させるとR1やR2はたとえばＣＰＵや入出力装置などのハードウェア資源に対応し，系内客数制限値はジョブの多重度，タスク数などのソフトウェア資源の容量に対応することになる．

本モデルの特性値を次の様に定める．

（１）サーバR1およびR2の窓口数をr_1 ,r_2 とする。

（２）系内客数制限値をr_0 とする。

但し、$r_0 \geqq r_1 \geqq 1$かつ$r_0 \geqq r_2 \geqq 1$とする。

（３）サーバR1およびR2でのサービス時間は，平均をそれぞれ$1/\mu_1$,$1/\mu_2$ とする指数分布を仮定する．

（４）系への客の到着は，平均間隔を$1/\lambda$とするポアソン到着を仮定する．

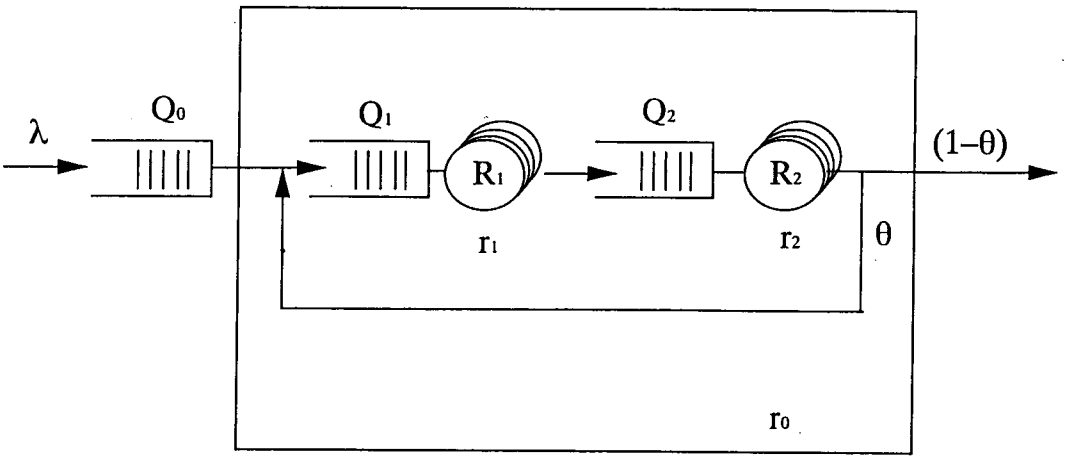


図5.10　待ち行列モデル

過去の文献によればこの種の問題として最もよく取り上げられているものは，多重プログラミングシステムの解析である．すなわち，ＣＰＵや入出力装置などのハードウェアを窓口とする待ち行列ネットワークの上に，ジョブ多重度による制約の設けられたモデルであり，解析手法からみると次の３つに大別される．

第１は近似手法である．Saver[19]は，この様なモデルについてFlow equivalenceの考え方を用いた近似的な数値解法を与えている．また，石黒[31]も同様のモデルについて近似解を求めている．

第２は数値解法であり，川島[39]らは一定量以上の待ち合せジョブは呼損となるモデル（従って，状態数は有限）について，各状態確率に関する平衡方程式の数値解を得る方法で解析している．また筆者らも実時間型システムを循環型の待行列ネットワークでモデル化しその数値解析を行った．[4]

第３は解析的手法によるもので，KonheimとReiser[8]が母関数を用いた解析方法を与えてる．

本論文での手法は，この第３のカテゴリに属するもので基本的にはKonheimとReiser[8]の手法に準じているが彼らのモデルでは各窓口の数が１であるのに対し我々はそれを複数窓口のモデルに拡張している．

以下では5.6節でその解析法を示し，5.7節で系の安定性に関する考察を行う．続いて解析法に基づく数値計算の手順を与え，計算例を示す．

5.6　モデルの解析

5.6.1　状態の定義

システム内に存在する全客数をk_0，窓口R1およびR2に存在する客数（サービス中の客および待ち行列上の客の和）を、それぞれk_1，k_2とする。このモデルは、系内客数制限モデルであるため、k_i 相互に次の関係が成立する。

$$k_1+k_2=\delta_0(k_0) \qquad \text{ただし，} \qquad \delta_i(k_i)=\mathrm{Min}(r_i,k_i) \tag{5.1}$$

したがって、システムの状態を全客数k_0と窓口R2に存在する客数k_2の２文字組(k_o,k_2)で定義でき、その定義域は図5.11に示すとおりとなる。

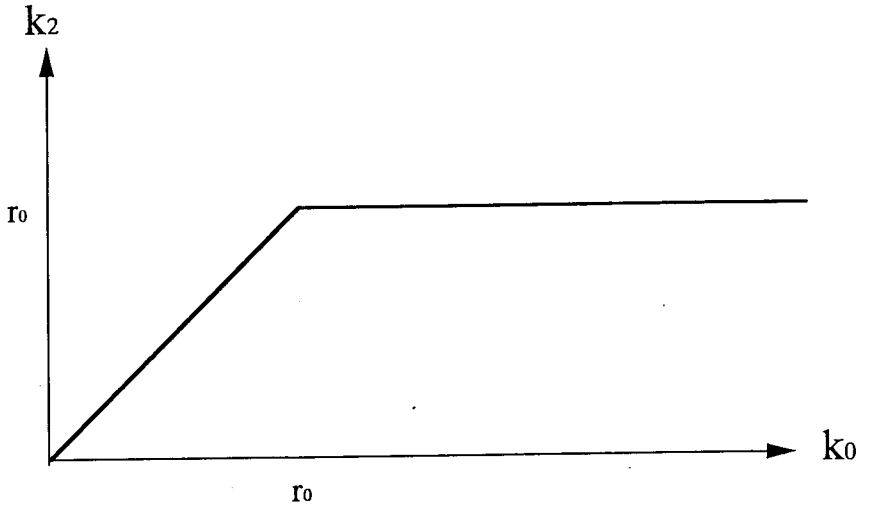


図5.11　（k0，k2）の定義域

5.6.2　平衡方程式の導出

時刻 t の時に、システムが状態(k_o,k_2)にある確率を$P_t(k_0,k_2)$で表すこととする。システムの状態が遷移するのは、窓口R1，R2 においてサービスが終了した時およびシステムへ客が到着した時である。この様な事象が同時に生ずる確率が無視できる程、微少な時間間隔Δ_tを選ぶと次の(1)～(4) の４つの独立した事象の生起する確率から系の基礎方程式を導出できる。

(1) 窓口R1 でサービスが終了した場合

この時は、客が窓口R1 からR2 へ移動するのみでシステム全体の客数には変化がない。したがって、状態の遷移は$(k_o,k_2-1)\rightarrow(k_o,k_2)$となる。この事象が生じる確率は、$\mu_1\delta_1(k_1+1)\Delta_t$である。

(2) 窓口R2 でのサービスが終了した場合

この場合は、客が窓口Ｒ1 へ戻る場合と，系外へ退去する場合とに分類される。

(2-1) 窓口R1へ戻る場合

システム内の客数に変化は無く，R2 の客がR1 へ移動するだけであるため状態の変化は$(k_o,k_2+1)\rightarrow(k_o,k_2)$と表わされる。この状態の遷移確率は$\theta\mu_2\delta_2(k_2+1)\Delta_t$である。

(2-2) 系外へ退去する場合

システム内の客数及びR2 の客が１少くなる。したがって，状態の遷移は$(k_o+1,k_2+1)\rightarrow(k_o,k_2)$となる。またその確率は、$(1-\theta)\mu_2\delta_2(k_2+1)\Delta_t$である

(3) 客がシステムへ到着した場合

状態の遷移は$(k_o-1,k_2)\rightarrow(k_o,k_2)$，その確率は$\lambda\Delta_t$である。

(4) 上記のいずれの事象も生じない場合

状態に変化は無く、$(k_o,k_2)\rightarrow(k_o,k_2)$.その確率は$1-\left(\sum_{i=1}^{2}\mu_i\delta_i(k_i)+\lambda\right)\Delta_t$となる。

以上の(1)〜(4)の各ケースの条件付き確率の総和が，時刻（$t+\Delta_t$）における系の状態確率を与えることになる。状態の定義域を考慮して$P_{t+\Delta_t}(k_o,k_2)$を書き下すと次の通りとなる。

$$
\begin{aligned}
&P_{t+\Delta_t}(k_o,k_2)\\
&\quad= P_t(k_o,k_2-1)\ \mu_1\delta_1(k_1+1)\Delta_t\varepsilon(k_2\neq 0)\\
&\quad+P_t(k_o,k_2+1)\theta\mu_2\delta_2(k_2+1)\Delta_t\varepsilon(k_1\neq 0)\\
&\quad+P_t(k_o+1,k_2+1)(1-\theta)\mu_2\delta_2(k_2+1)\Delta_t\ \{\varepsilon(k_0\geqq r_0)\varepsilon(k_1\neq 0)+\varepsilon(k_0<r_0)\}\\
&\quad+P_t(k_o-1,k_2)\lambda\Delta_t\ \{\varepsilon(k_0<r_0)\varepsilon(k_1\neq 0)+\varepsilon(k_0>r_0)\}\\
&\quad+P_t(k_o,k_2)\left\{1-\left(\sum_{i=1}^{2}\mu_i\delta_i(k_i)+\lambda\right)\Delta_t\right\}
\end{aligned}
\tag{5.2}
$$

ここで、 $\varepsilon(q)=1$ ：q が真のとき
$=0$ ：q が偽のとき

本式を変形して、次の式を得る。

$$
\begin{aligned}
\{P_{t+\Delta_t}(k_o,k_2)-P_t(k_o,k_2)\}\ /\Delta_t &= P_t(k_o,k_2-1)\ \mu_1\delta_1(k_1+1)\varepsilon(k_2\neq 0)\\
&+P_t(k_o,k_2+1)\theta\mu_2\delta_2(k_2+1)\varepsilon(k_1\neq 0)\\
&+P_t(k_o+1,k_2+1)(1-\theta)\mu_2\delta_2(k_2+1)\ \{\varepsilon(k_0\geqq r_0)\varepsilon(k_1\neq 0)+\varepsilon(k_0<r_0)\}\\
&+P_t(k_o-1,k_2)\lambda\ \{\varepsilon(k_0<r_0)\varepsilon(k_1\neq 0)+\varepsilon(k_0>r_0)\\
&-P_t(k_o,k_2)\left(\sum_{i=1}^{2}\mu_i\delta_i(k_i)+\lambda\right)
\end{aligned}
\tag{5.3}
$$

ここで、$\Delta t\rightarrow 0$， $t\rightarrow\infty$の極限をとる。定常確率が存在すると仮定し、それを$P(k_o,k_2)$と書くと次の平衡方程式を得る。

$$\left(\sum_{i=1}^{2} \mu_i \delta_i(k_i) + \lambda\right) P(k_o, k_2)$$

$$= P(k_o, k_2 - 1)\, \mu_1 \delta_1 (k_1 + 1)\, \varepsilon(k_2 \neq 0)$$

$$+ P(k_o, k_2 + 1)\, \theta \mu_2 \delta_2 (k_2 + 1)\, \varepsilon(k_1 \neq 0)$$

$$+ P(k_o + 1, k_2 + 1)\, (1-\theta) \mu_2 \delta_2 (k_2 + 1) \quad \{\varepsilon(k_0 \geqq r_0)\, \varepsilon(k_1 \neq 0) + \varepsilon(k_0 < r_0)\}$$

$$+ P(k_o - 1, k_2 + 1) \lambda \{\varepsilon(k_0 < r_0)\, \varepsilon(k_1 \neq 0) + \varepsilon(k_0 > r_0)\} \tag{5.4}$$

但し、$k_1 = \delta_0(k_0) - k_2$

これを、状態遷移図で表すと、図5.12となる。

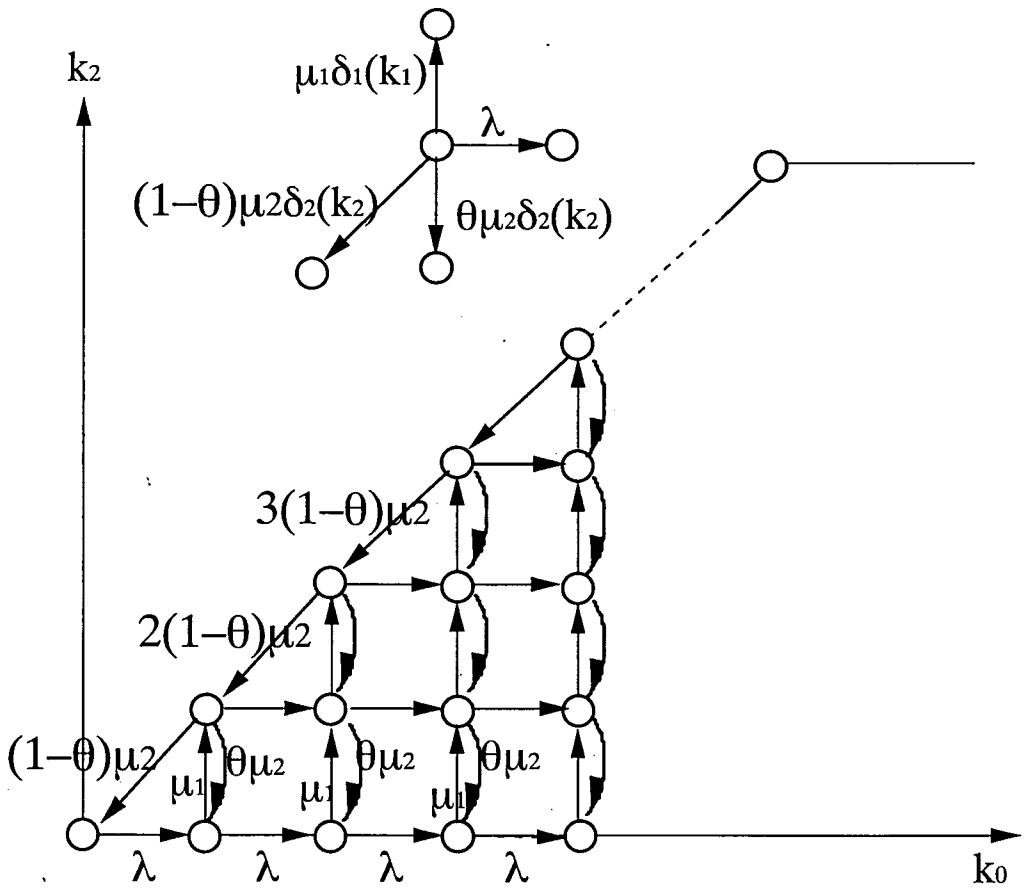


図5.12　状態遷移図

5.6.3 母関数に関する解析

状態確率 $P(k_o,k_2)$ に対して，次の母関数を定義する。

$$F_{k_2}(z) = \sum_{n=k_2}^{\infty} P(n,k_2) z^{n-k_2} \qquad k_2 = 0, 1, \ldots, r_0 \tag{5.5}$$

平衡方程式(5.4)式を用いて母関数間の関係を求めると次の通りとなる。

$$z(\lambda z - \lambda - \mu_1 r_1) F_0(z) + \mu_2 (1 - \theta + \theta z) z F_1(z) = -\mu_1 \sum_{j=0}^{r_1 - 1} (r_1 - j) P(j,0) z^{j+1} \tag{5.6a}$$

$$\mu_1 \delta_1 (r_0 + 1 - i) F_{i-1}(z) + z\left\{\lambda z - \lambda - \mu_1 \delta_1 (r_0 - i) - \mu_2 \delta_2 (i)\right\} F_i(z) + \mu_2 \delta_2 (i+1)(1 - \theta + \theta z) z F_{i+1}(z)$$

$$= \mu_1 \sum_{j=i-1}^{r_0 - 1} \left\{\delta_1 (r_0 - i + 1) - \delta_1 (j - i + 1)\right\} P(j, i-1) z^{j-i+1} - \mu_1 \sum_{j=i}^{r_0 - 1} \left\{\delta_1 (r_0 - i) - \delta_1 (j - i)\right\} P(j,i) z^{j-i+1}$$

但し，$1 \le i \le r_0 - 1$ (5.6b)

$$\mu_1 F_{r_0 - 1}(z) + z(\lambda z - \lambda - \mu_2 r_2) F_{r_0}(z) = \mu_1 P(r_0 - 1, r_0 - 1) \tag{5.6c}$$

(5.6)式を行列形式で表現すると次の通りとなる。

$$\mathbf{D}(z) \times \mathbf{F}(z) = \mathbf{d}(z) \tag{5.7}$$

$$\mathbf{F}(z) = \left[F_0(z), F_1(z), \cdots, F_{r_0}(z)\right]^t$$

$$\mathbf{d}(z) = \left[d_0(z), d_1(z), \cdots, d_{r_0}(z)\right]^t$$

$$\mathbf{D}(z) = \begin{bmatrix} b_0(z) & c_0(z) & 0 & & & & 0 \\ a_1 & b_1(z) & c_1(z) & \cdots & \cdots & \cdots & 0 \\ 0 & a_2 & b_2(z) & & & & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots\cdots & a_{r_0 - 1} & b_{r_0 - 1}(z) & c_{r_0 - 1}(z) \\ 0 & 0 & 0 & & 0 & a_{r_0} & b_{r_0}(z) \end{bmatrix}$$

ここで，$a_i = \mu_1 \delta_1(r_0 + 1 - i)$ $\quad i = 1,2,\ldots,r_0$ (5.8)

$$b_i(z) = z\left\{\lambda z - \lambda - \mu_1\delta_1(r_0 - i) - \mu_2\delta_2(i)\right\} \qquad i = 0,1,\ldots,r_0 \tag{5.9}$$

$$c_i(z) = \mu_2\delta_2(i+1)(1 - \theta + \theta z)z \qquad i = 0,1,\ldots,r_0 - 1 \tag{5.10}$$

$$d_i(z) = \varepsilon(i \neq 0)\mu_1 \sum_{j=i-1}^{r_0-1} \left\{\delta_1(r_0 - i + 1) - \delta_1(j - i + 1)\right\} P(j, i-1) z^{j-i+1}$$

$$- \varepsilon(i \neq r_0)\mu_1 \sum_{j=i}^{r_0-1} \left\{\delta_1(r_0 - i) - \delta_1(j - i)\right\} P(j, i) z^{j-i+1} \qquad i = 0,1,\ldots,r_0 \tag{5.11}$$

(5.7) 式を$\mathbf{F}(z)$ について解くために、若干の準備を行う。すなわち、行列$\mathbf{D}(z)$ の小行列を２種類定義し、その行列式を求めておく。　（図5.13）

$\mathbf{U}_k(z)$: 行列$\mathbf{D}(z)$ の１行１列から(k+1)行，(k+1)列までの要素から成る行列

$\mathbf{V}_k(z)$: 行列$\mathbf{D}(z)$ の(k+2)行(k+2)列から,(r_o +1)行(r_o +1) 列までの要素から成る行列

$$
\begin{array}{c}
\text{行} \quad 1 \quad 2 \quad \cdots \quad k+1 \quad k+2 \quad \cdots \quad r_0+1 \\
\mathbf{D}(z) = \left[\begin{array}{cccc|ccc}
b_0(z) & c_0(z) & \cdots 0 & 0 & 0 & \cdots & 0 \\
\vdots & & \vdots & \vdots & \vdots & & \vdots \\
0 & \cdots & b_{k-1}(z) & c_{k-1}(z) & 0 & \cdots & 0 \\
0 & \cdots & a_k & b_k(z) & c_k(z) & \cdots & 0 \\
\hline
 & & & a_{k+1} & b_k(z) & c_{k+1}(z) \cdots & 0 \\
 & & & & a_{k+2} & b_k(z) \cdots & 0 \\
 & & & & \vdots & \vdots & \vdots \\
 & & & & 0 & 0 \cdots & b_{r_0}(z)
\end{array}\right]
\begin{array}{c}
1 \\ \vdots \\ k \\ k+1 \\ k+2 \\ \vdots \\ r_0+1 \\ \text{列}
\end{array}
\end{array}
$$

$\uparrow$ $U_k(z)$　　$\uparrow$ $V_{k+1}(z)$

図5.13 $U_k(z)$および $V_{k+1}(z)$の定義

$U_k(z) = \det \mathbf{U}_k(z)$, $V_k(z) = \det \mathbf{V}_k(z)$ とすると $U_k(z)$, $V_k(z)$ はそれぞれ次の漸化式で表現できる。

$$U_k(z) = b_k(z)U_{k-1}(z) - a_k c_{k-1}(z)U_{k-2}(z) \quad k=1,2,\ldots,r_0$$
$$U_{-1}(z) = 1$$
$$U_0(z) = b_0(z) \tag{5.12}$$

$$V_k(z) = b_k(z)V_{k+1}(z) - a_{k+1}c_k(z)V_{k+2}(z) \quad k = 0,1,..,r_0 -1$$
$$V_{r_0}(z) = b_{r_0}(z)$$
$$V_{r_0+1}(z) = 1 \tag{5.13}$$

[系１] $U_k(z), V_k(z)$ の次数は次の通りである。
いずれも，数学的帰納法で簡単に示すことができる.
[系1.1] $U_k(z)$ のzに関する最大次数は$2(k+1)$,最小次数は $\lfloor k/2 \rfloor+1$ である.

なお，$\lfloor x \rfloor$ はxの切り捨てを表わす.

[系1.2] $V_k(z)$ の z に関する最大次数は$2(r_0+1-k)$，最小次数は$\lfloor (r_0+2-k)/2 \rfloor$である.

以上の準備を前提に(5.7) 式を解くと

$$F_n(z) = D_n(z) / U_{r_0}(z) \qquad n=0,1,2,\ldots, r_0 \tag{5.14}$$

となる。ただし$D_n(z)$ は行列$\mathbf{D}(z)$ の第n列を$\mathbf{d}(z)$ で置き換えた行列の行列式であり次式で示される。

$$D_n(z) = \sum_{l=0}^{r_0-1} \sum_{k=1}^{r_0-1} f_{k,l}^n(z)\, P(k,l) \qquad n=0,1,2,\ldots, r_0 \tag{5.15}$$

ここで,

$$f_{k,l}^n(z) = \varepsilon(n\neq 0)\,(-1)^{l-n+1}\mu_1\left\{\delta_1(r_0-l)-\delta_1(k-l)\right\}\prod_{j=l+2}^{n+1} a_j$$
$$\times V_{n+1}(z)\left\{U_l(z) + zU_{l-1}(z)\,a_{l+1}\right\}z^{k-l}/a_{n+1} \quad (0 \le l \le n-1) \tag{5.16a}$$

$$f^{n}_{k,l}(z) = \varepsilon(n \neq r_0)(-1)^{l-n+1}\mu_1\left\{\delta_1(r_0-l)-\delta_1(k-l)\right\}\prod_{j=n-1}^{l-1} c_j(z)$$

$$\times U_{n-1}(z)\left\{V_{l+2}(z)\,c_l(z)+zV_{l+1}(z)\right\}z^{k-l}/c_{n-1}(z) \quad (n \le l \le r_0-1) \qquad (5.16b)$$

なお，a_j および$c_j(z)$について，(5.8),(5.10)式の定義を次のように拡張する．

$$a_{r_0+1}=1\ ,c_{-1}(z)=1$$

[系2] $D_n(z)$のzに関する最大次数は,

$$3r_0-n+1\ (0 \leqq n < r_0)\ ,\ 2r_0\ (n=r_0)$$

最小次数は,

$$\lfloor (r_0-n+1)/2 \rfloor+1\ (0<n \leqq r_0),\ \lfloor (r_0+1)/2 \rfloor+2\ (n=0)$$

(証明）系1.1,1.2を漸化式（15）に適用することによって得られる。 □

以上で，$F_n(z)$が得られたが，(5.15)式に示す通りその中に本来求めたい未知の状態確率を$r_1r_0-r_1(r_1-1)/2$ 個残しており，これを決定する必要がある。(図5.14の台形で示す部分）

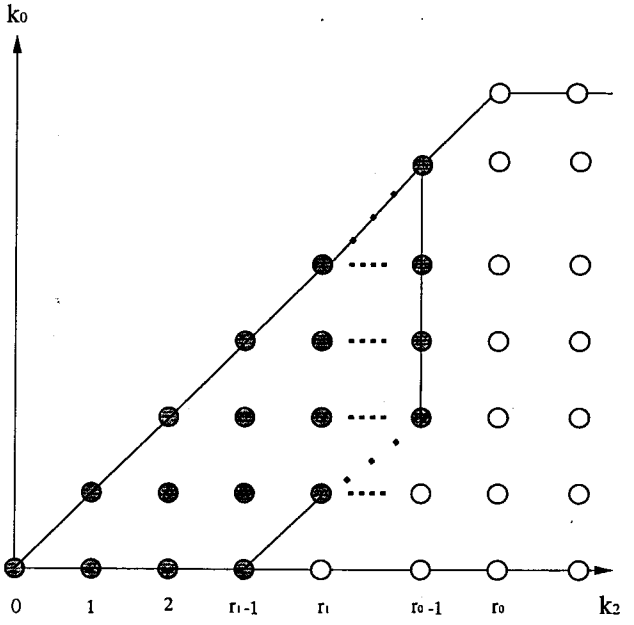


図5.14　未知の状態確率

ところで，平衡方程式(5.4)を用いると残された未知の状態確率は, $P(r_0+k, 2k+1)$ $(k=0,1,2,\cdot\cdot,\lceil r_0/2\rceil-1)$ の線形結合で次の様に表現することができる．ここで，$\lceil x\rceil$ はxの切り上げを表わす．

$$P(k,l)=\sum_{m=0}^{\lceil r_0/2\rceil-1}\alpha_{k,l,m}P(r_0+m,2m+1) \tag{5.17}$$

これを(5.15)式に代入すると

$$D_n(z)=\sum_{m=0}^{\lceil r_0/2\rceil-1}\sum_{l=0}^{r_0-1}\sum_{k=1}^{r_0-1}\alpha_{k,l,m}f_{k,l}^{n}(z)P(r_0+m,2m+1) \tag{5.18}$$

結局，$\mathbf{F}(z)$ に残される未知の状態確率は$\lceil r_0/2\rceil$個となり，これらは系が平衡状態にあるための条件および確率の総和が１である条件から次のように定めることができる．すなわち，系が安定ならば母関数$\mathbf{F}(z)$ は収束するため，単位円内で解析的である．したがって, $F_n(z)$の分母 $U_{r_0}(z)$の$|z|<1$ の根は分子$D_n(z)$の根と一致する必要がある．後節の定理３で示す様に系が安定ならば$U_{r_0}(z)=0$ の$|z|<1$（但し, $z\neq 0$）の根η_i の数は$\lceil r_0/2\rceil-1$ となるため(5.18)式より次の$\lceil r_0/2\rceil-1$個の条件式を得る．

$$D_n(\eta_i)=0 \qquad 1\leqq i\leqq\lceil r_0/2\rceil-1 \tag{5.19}$$

本式は$\lceil r_0/2\rceil$ 個の状態確率を未知数とする$\lceil r_0/2\rceil-1$個の連立一次方程式であり、それを解くことにより未知数は1 個残ることになる．

最後に確率の総和が１となる条件 $\sum_{n=0}^{r_0}F_n(1)=1$ を用いてすべての未知の確率を決定できる．以上の手順の詳細は5.8節でまとめて示す．

5.7 系の安定性に関する考察

系の安定性を議論するために2,3 の準備を行う。

[系3] $U_k(z)$ ($k=0,1,\ldots,r_0$) について次が成立する。

[系3.1] $U_k(1)=(-\mu_1)^{k+1}\prod_{i=0}^{k}\delta_1(r_0-i)$

[系3.2] $U_k(z)=\lambda^{k+1}z^{2k+2}+o(z^{2k+1})$

[系3.3] 十分小さな正数 ε に対し,

k(mod 4)=0または1の時, $U_k(\varepsilon)<0$

k(mod 4)=2または3の時, $U_k(\varepsilon)>0$

(証明) いずれも，数学的帰納法により簡単に示すことができる．系3.3は，$U_k(z)=z^{\lfloor k/2\rfloor+1}E_k(z)$ と置き(系1.1)，$E_k(z)$ の符号を評価する． □

[定理1] $U_k(z)=0$ $(0\leqq k\leqq r_0-1)$の根は以下の分布を持つ．z=0の根を$\lfloor k/2\rfloor+1$個,

$0<z<1$の範囲に$\lfloor (k+1)/2\rfloor$個の単根$\eta_{k,i}$,

$1<z<\infty$の範囲に(k+1)個の単根$\zeta_{k,i}$を持つ.

また，$U_{k-1}(z)$ の根$\eta_{k-1,i}$, $\zeta_{k-1,i}$ と次の様な交互性を持つ.

(1) kが奇数の時

$$0<\eta_{k,1}<\eta_{k-1,1}<\eta_{k,2}<\eta_{k-1,2}\cdots<\eta_{k-1,\lfloor k/2\rfloor}<\eta_{k,\lfloor (k+1)/2\rfloor}<1$$

$$1<\zeta_{k,1}<\zeta_{k-1,1}<\zeta_{k,2}<\zeta_{k-1,2}\cdots<\zeta_{k-1,k}<\zeta_{k,k+1}$$

(2) kが偶数の時

$$0<\eta_{k-1,1}<\eta_{k,1}<\eta_{k-1,2}<\eta_{k,2}\cdots\eta_{k-1,\lfloor k/2\rfloor}<\eta_{k,\lfloor (k+1)/2\rfloor}<1$$

$$1<\zeta_{k,1}<\zeta_{k-1,1}<\zeta_{k,2}<\zeta_{k-1,2}\cdots<\zeta_{k-1,k}<\zeta_{k,k+1}$$

(証明) 数学的帰納法を用いて証明する.

まず, k=4n-1およびk=4nにおいて定理１が成立すると仮定する。

(1) 系3.3 より $U_{4n-1}(\varepsilon)>0, U_{4n}(\varepsilon)<0, U_{4n+1}(\varepsilon)<0$

(2) 系3.1 より $U_{4n-1}(1)>0$, $U_{4n}(1)<0$, $U_{4n+1}(1)>0$

(3) 系3.2 より $z\to\infty$の時, $U_{4n-1}(z)>0, U_{4n}(z)>0$, $U_{4n+1}(z)>0$

(4) (5.12) 式より

$$U_{4n+1}(z) = b_{4n+1}(z)U_{4n}(z) - a_{4n+1}c_{4n}(z)U_{4n-1}(z)$$

であり，本式に$U_{4n}(z)$ の根$\eta_{4n,i}$, $\zeta_{4n,i}$を代入すると，右辺は第２項のみが残り,(5.8),(5.10)式を用いると次の様になる.

$$U_{4n+1}(\eta_{4n,i}) = -\mu_1\mu_2\delta_1(r_0-4n)\delta_2(4n+1) \times (1-\theta+\theta\eta_{4n,i})\eta_{4n,i}U_{4n-1}(\eta_{4n,i})$$

$$U_{4n+1}(\zeta_{4n,i}) = -\mu_1\mu_2\delta_1(r_0-4n)\delta_2(4n+1) \times (1-\theta+\theta\zeta_{4n,i})\zeta_{4n,i}U_{4n-1}(\zeta_{4n,i})$$

ここで，μ_1, $\mu_2 > 0$, $\delta_1(r_0-4n)$, $\delta_2(4n+1) > 0$, $1 > \theta > 0$, $\eta_{4n,i} > 0$, $\zeta_{4n,i} > 0$であるため,$U_{4n+1}(\eta_{4n,i})$ および$U_{4n+1}(\zeta_{4n,i})$ の符号は$U_{4n-1}(\eta_{4n,i})$， $U_{4n-1}(\zeta_{4n,i})$ の符号と逆になる.

(5)以上の事実を用いて$U_{4n-1}(z)$, $U_{4n}(z)$ および$U_{4n+1}(z)$ の符号変化を図示すると図5.15となる.

(6) 同図から$U_{4n+1}(z)$ は次の区間で符号が変化する.

$$(0,\eta_{4n,1}), (\eta_{4n,1},\eta_{4n,2}),\cdots(\eta_{4n,2n},1)$$

$$(1,\zeta_{4n,1}), (\zeta_{4n,1},\zeta_{4n,2}),\cdots,(\zeta_{4n,4n+1},\infty)$$

したがって，これらの各領域の中に少なくとも奇数個の実根が存在する.

(7) ところで，この領域の数は（６n+３）個であり$U_{4n+1}(z)$のz＝０以外の根の数と一致する(系1.1). したがって, 上記の各領域に1 個づつ根が存在し,これ以外には根が無い。以上でk=4n+1の時の定理1 の成立が示された。

(8) k=4n+2,4n+3および4n+4のケースについても同様の手順で成立を示すことができる。

(9) 一方，$U_0(z)=0$は，z=0および$1+\mu_1 r_1/\lambda$の根を持つ。また,$U_1(z)$について符号変化を調べると$(0,1)$,$(1,1+\mu_1 r_1/\lambda)$,$(1+\mu_1 r_1/\lambda,\infty)$の各領域で符号変化が起り各領域に１つづつの根を持つことが判り定理１の成立が示される。

(10) 以上から数学的帰納法により定理１が証明された. □

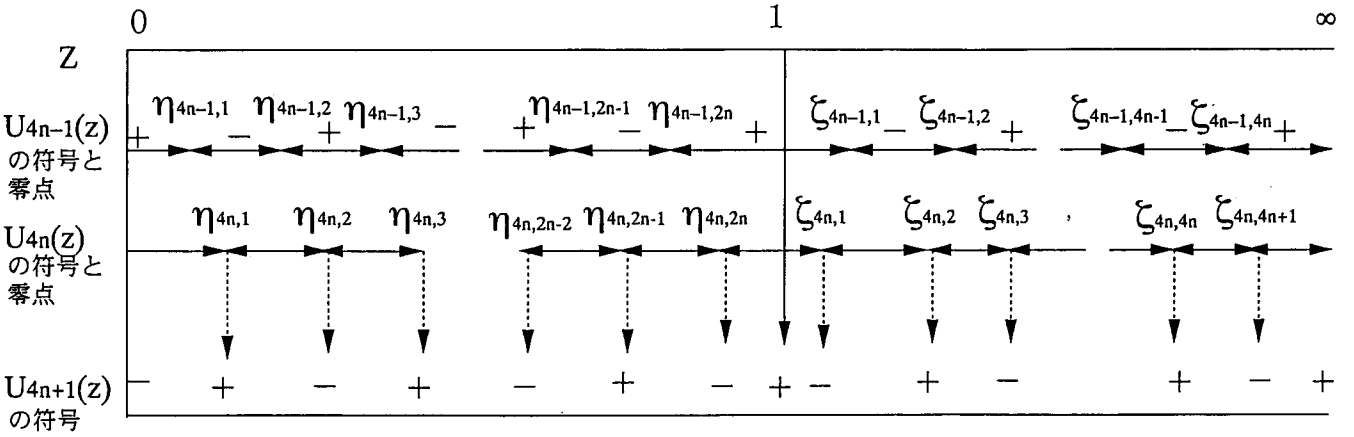


図5.15　$U_{4n+1}(z)$の符号変化

[定理２] $U_{r_0}(z)=0$ の根の分布は,次の3つのケ-スA,B,Cのいずれかである.

ケースA: $U'_{r_0}(1)>0$ (r_0 が奇数の場合),または$U'_{r_0}(1)<0$ (r_0 が偶数の場合)

(A1) z=0の根を$\lfloor r_0/2\rfloor+1$個, z=1の根を１個, 0<z<1の中に$\lceil r_0/2\rceil-1$ 個の単根$\eta_{r_0,i}$,
1<z<∞の中に (r_0+1)個の単根$\zeta_{r_0,i}$ を持つ.

(A2) 上記の根は$U_{r_0-1}(z)=0$の根,$\eta_{r_0-1,i}$,$\zeta_{r_0-1,i}$ と以下の交互関係にある.

(A2.1) r_0 が奇数の場合

$$0<\eta_{r_0,1}<\eta_{r_0-1,1}<\eta_{r_0,2}<\cdots<\eta_{r_0,(r_0-1)/2}<\eta_{r_0-1,(r_0-1)/2}<1$$

$$1<\zeta_{r_0,1}<\zeta_{r_0-1,1}<\zeta_{r_0,2}<\cdots<\zeta_{r_0,r_0}<\zeta_{r_0-1,r_0}<\zeta_{r_0,r_0+1}$$

(A2.2) r_0が偶数の場合

$$0<\eta_{r_0-1,1}<\eta_{r_0,1}<\eta_{r_0-1,2}<\cdots<\eta_{r_0,r_0/2-1}<\eta_{r_0-1,r_0/2}<1$$

$$1<\zeta_{r_0,1}<\zeta_{r_0-1,1}<\zeta_{r_0,2}<\cdots<\zeta_{r_0,r_0}<\zeta_{r_0-1,r_0}<\zeta_{r_0,r_0+1}$$

ケースB : $U'_{r_0}(1)<0$ (r_0が奇数の場合)または$U'_{r_0}(1)>0$ (r_0が偶数の場合)

(B1) z=0の根を$\lfloor r_0/2\rfloor+1$個, z=1の根を１個, 0<z<1の中に$\lceil r_0/2\rceil$個の単根$\eta_{r_0,i}$,
1<z<∞の中に r_0 個の単根$\zeta_{r_0,i}$ を持つ.

(B2) 上記の根は$U_{r_0-1}(z)=0$の根と以下の交互関係にある.

(B2.1) r_0が奇数の場合

$$0<\eta_{r_0,1}<\eta_{r_0-1,1}<\eta_{r_0,2}<\cdots<\eta_{r_0-1,(r_0-1)/2}<\eta_{r_0,(r_0+1)/2}<1$$

$$1<\zeta_{r_0-1,1}<\zeta_{r_0,1}<\zeta_{r_0-1,2}<\cdots<\zeta_{r_0,r_0-1}<\zeta_{r_0-1,r_0}<\zeta_{r_0,r_0}$$

(B2.2) r_0が偶数の場合

$$0 < \eta_{r_0-1,1} < \eta_{r_0,1} < \eta_{r_0-1,2} < \cdots < \eta_{r_0-1,r_0/2} < \eta_{r_0,r_0/2} < 1$$

$$1 < \zeta_{r_0-1,1} < \zeta_{r_0,1} < \zeta_{r_0-1,2} < \cdots < \zeta_{r_0,r_0-1} < \zeta_{r_0-1,r_0} < \zeta_{r_0,r_0}$$

ケースC: $U'_{r_0}(1) = 0$

(C1) z=0の根を$\lfloor r_0/2 \rfloor$+1個, z=1の根を2個, 0<z<1の中に$\lceil r_0/2 \rceil$-1 個の単根$\eta_{r_0,i}$, 1<z<∞の中に r_0 個の単根$\zeta_{r_0,i}$を持つ.

(C2) 上記の根は$U_{r_0-1}(z)=0$の根と以下の交互関係にある.

(C2.1) r_0が奇数の場合

$$0 < \eta_{r_0,1} < \eta_{r_0-1,1} < \eta_{r_0,2} < \cdots < \eta_{r_0,(r_0-1)/2} < \eta_{r_0-1,(r_0-1)/2} < 1$$

$$1 < \zeta_{r_0-1,1} < \zeta_{r_0,1} < \zeta_{r_0-1,2} < \cdots < \zeta_{r_0,r_0-1} < \zeta_{r_0-1,r_0} < \zeta_{r_0,r_0}$$

(C2.2) r_0が偶数の場合

$$0 < \eta_{r_0-1,1} < \eta_{r_0,1} < \eta_{r_0-1,2} < \cdots < \eta_{r_0,r_0/2-1} < \eta_{r_0-1,r_0/2} < 1$$

$$1 < \zeta_{r_0-1,1} < \zeta_{r_0,1} < \zeta_{r_0-1,2} < \cdots < \zeta_{r_0,r_0-1} < \zeta_{r_0-1,r_0} < \zeta_{r_0,r_0}$$

(定理２の証明)

r_0 =4n のケースについて以下に示す.

(1)$U_{4n-1}(z)$ および$U_{4n-2}(z)$ の零点を定理1 に基づいて示すと図5.16の通りとなる.

(2) 一方,(12)式に$U_{4n-1}(z)$ の根$\eta_{4n-1,i}$,$\zeta_{4n-1,i}$ を代入すると,

$$U_{4n}(\eta_{4n-1,i}) = -a_{4n}c_{4n-1}(\eta_{4n-1,i})U_{4n-2}(\eta_{4n-1,i})$$

$$U_{4n}(\zeta_{4n-1,i}) = -a_{4n}c_{4n-1}(\zeta_{4n-1,i})U_{4n-2}(\zeta_{4n-1,i})$$

となり，定理１の証明(4)項と同様の議論により，$U_{4n}(\eta_{4n-1,i})$,$U_{4n}(\zeta_{4n-1,i})$ の符号は$U_{4n-2}(\eta_{4n-1,i})$,$U_{4n-2}(\zeta_{4n-1,i})$ の符号と逆の関係にある.

(3)系3.2 より$U_{4n}(\infty) > 0$,系3.3 より$U_{4n}(\varepsilon) < 0$

(4)系3.1より$U_{4n}(1) = 0$ であり$U'_{4n}(1)$ の符号に関して次の４つの可能性がある.

ケースA : $U'_{4n}(1) > 0$ 即ち $U_{4n}(1+\varepsilon) > 0$, $U_{4n}(1-\varepsilon) < 0$

ケースB : $U'_{4n}(1) < 0$ 即ち $U_{4n}(1+\varepsilon) < 0$, $U_{4n}(1-\varepsilon) > 0$

ケースCa : $U'_{4n}(1) = 0$ 即ち $U_{4n}(1+\varepsilon) > 0$, $U_{4n}(1-\varepsilon) > 0$

ケースCb : $U'_{4n}(1) = 0$ 即ち $U_{4n}(1+\varepsilon) < 0$, $U_{4n}(1-\varepsilon) < 0$

(5) 上の(2)〜(4)の事実を用いて $U_{4n}(z)$ の符号を図5.16に書き加えると，次の区間で $U_{4n}(z)$ の符号が変化しており少なくとも各々に奇数個の実根が存在する.

全ケース共通　：

$$(\eta_{4n-1,1}, \eta_{4n-1,2}), (\eta_{4n-1,2}, \eta_{4n-1,3}), \ldots, (\eta_{4n-1,2n-1}, \eta_{4n-1,2n}),$$

$$(\zeta_{4n-1,1}, \zeta_{4n-1,2}), (\zeta_{4n-1,2}, \zeta_{4n-1,3}), \ldots, (\zeta_{4n-1,4n}, \infty)$$

ケースA ： $(\eta_{4n-1,2n}, 1)$, 1

ケースB ： 1 , $(1, \zeta_{4n-1,1})$

ケースCa : 1 (偶数個の重根)

ケースCb : $(\eta_{4n-1,2n}, 1)$, 1 (偶数個の重根)

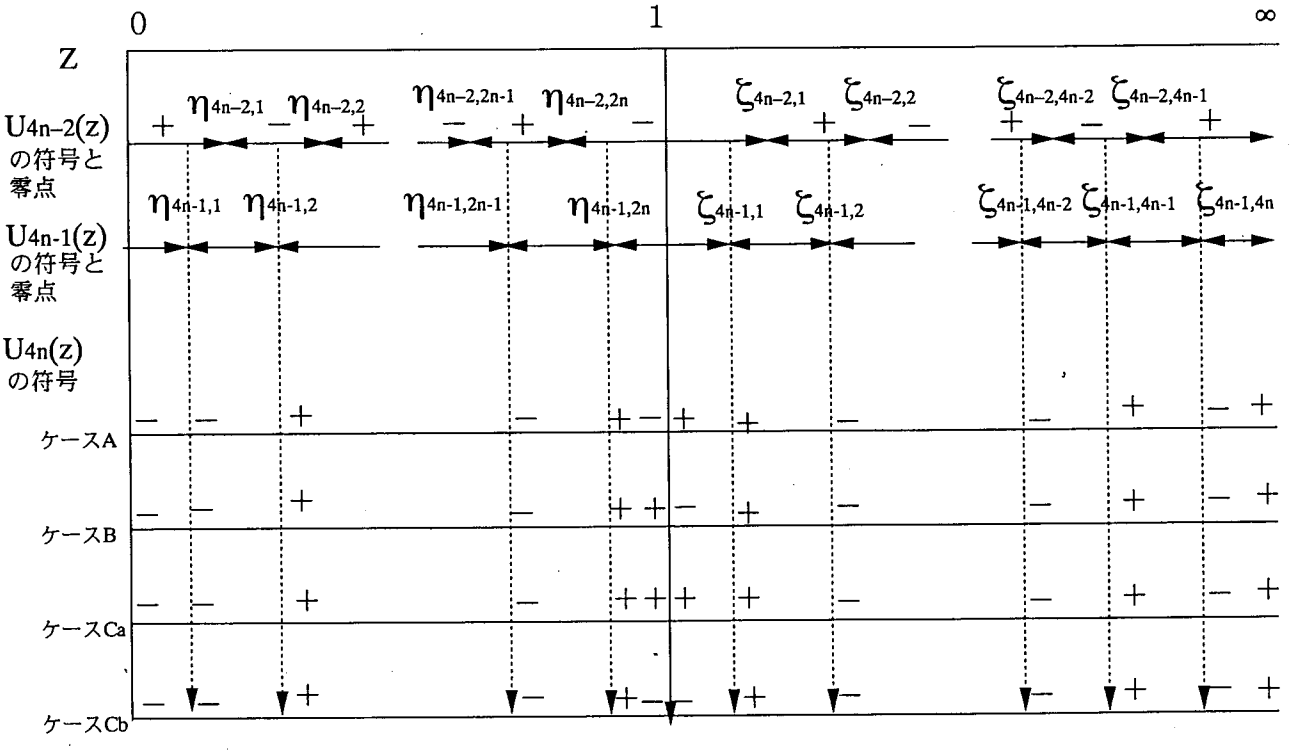


図5.16　$U_{4n}(z)$ の符号変化

(6) ケースAおよびケースBについては(5)の区間の総数は(6n + 1) 個であり，これは $U_{4n}(z) = 0$ のz=0を除く根の総数(系1.1より得られる）と一致する．従って(5)に示す各領域に1 個づつ根を持ち，それ以外には根は存在しない．
(7) ケ-スCa については1の根を２重根とすると根の総数と一致する．従って残りの各領域に単根を持ちそれ以外には根が存在しない．
(8)ケースCbについては区間の数が根の総数を越えるためこの様なケースは存在し得ない．

以上でr_o =4n のケースについて定理２の証明ができた．r_o =4n+1,4n+2,4n+3 についても同様の方法で証明できる． □

[定理３] 系が安定である必要十分条件は，$U_{r_0}(z)= 0$ が$0<z<1$の領域に（$\lceil r_0/2\rceil$-1)個根をもつこと，すなわち定理２のケースAまたはケースCとなることである．

(必要性の証明)
(1) 系が安定でかつ$U_{r_0}(z)$が $0 <z< 1$ の領域に$\lceil r_0/2\rceil$ 個の根($\eta_{r_0,1}, \eta_{r_0,2}, \ldots, \eta_{r_0,\lceil r_0/2\rceil}$) を持つ(定理２のケースB）と仮定する．

(2) 母関数の解折性の条件から分子の多項式も同一の零点を持つ．したがって，(5.15)式より

$$D_{r_0}(\eta_{r_0,i}) = \sum_{l=0}^{r_0-1}\sum_{k=l}^{r_0-1} f^{r_0}_{k,l}(\eta_{r_0,i})P(k,l) = 0 \qquad i=1,2,\ldots,\lceil r_0/2\rceil \tag{5.20}$$

が成立する必要がある．
これはP(k, l) に関する連立一次方程式である．ここで，

$$f^{r_0}_{k,l}(\eta_{r_0,i}) = (-1)^{l-r_0+1} h_{k,l}\left\{U_l(\eta_{r_0,i}) + \eta_{r_0,i}U_{l-1}(\eta_{r_0,i})a_{l+1}\right\}\eta_{r_0,i}^{\,k-l}$$

$$h_{k,l} = \mu_1\left\{\delta_1(r_0-l) - \delta_1(k-l)\right\}\prod_{j=l+2}^{r_0+1} a_j \geq 0 \qquad 0 \leq l \leq r_0-1,\ l \leq k \leq r_0-1$$

(3) 一方，根の交互性および$U_l(z)$の符号(定理1および定理2)に関する考察により，

$$U_l(\eta_{r_0,\lceil r_0/2\rceil}) + \eta_{r_0,\lceil r_0/2\rceil} a_{l+1} U_{l+1}(\eta_{r_0,\lceil r_0/2\rceil}) > 0 \quad (l\text{: 奇数の場合})$$

$$< 0 \quad (l\text{:偶数 の場合})$$

(4) 以上より

$$f^{r_0}_{k,l}(\eta_{r_0,\lceil r_0/2\rceil}) < 0 \quad (r_0\text{ : 奇数})$$

$$> 0 \quad (r_0\text{ : 偶数})$$

となる．つまり(5.20)式の連立方程式のうち $i=\lceil r_0/2\rceil$ の式について$P(k, l)$の係数がすべて正(r_0が偶数の時)またはすべて負(r_0が奇数の時)となり，これを満足する解は$P(k, l) = 0$となる．これは系が安定である仮定と矛盾する．

(十分性の証明)

(1) $U_{r_0}(z)$が$0<z<1$の範囲内に($\lceil r_0/2\rceil$-1)個の根($\eta_{r_0,1}, \eta_{r_0,2}, \cdots, \eta_{r_0,\lceil r_0/2\rceil-1}$)を持つと仮定する．すると，$D_{r_0}(\eta_{r_0,i}) = 0$ ($i=1,2,\ldots,\lceil r_0/2\rceil-1$)は$\lceil r_0/2\rceil$ 個の未知数に対し$\lceil r_0/2\rceil$ - 1個の同次式となり非零解が常に存在する．

(2) この解を用いると前節で述べた手順により$\mathbf{F}_n(z)$が得られ，これらは$|z|<1$で解析的かつ$|z|=1$で連続となる． □

[定理４] 系が安定である条件は，

$$\rho_1 - r_1 + \sum_{j=1}^{r_0} \nu^j \left\{ \rho_1 - \delta_1(r_0 - j) + \delta_2(j)\theta/\nu \right\} \prod_{i=1}^{j} \delta_1(r_0 - i + 1)/\delta_2(i) \leqq 0 \tag{5.21}$$

である．ただし，$\rho_1 = \lambda/\mu_1$ ，$\nu = \mu_1/\mu_2$

(証明)

(1) 定理２および定理３より系が安定である条件は，

r_0が奇数の場合，$U'_{r_0}(1) \geq 0$

r_0が偶数の場合，$U'_{r_0}(1) \leq 0$

となる．以下，$U'_{r_0}(1)$を求める．

(2)漸化式(12)の両辺を微分し$z = 1$と置き，

$$u_k = U'_k(1) + \mu_1 \delta_1(r_0 - k)\, U'_{k-1}(1) \tag{5.22}$$

を導入して整理すると次の漸化式を得る．

$$u_k + \mu_2\delta_2(k)u_{k-1} = D_k \tag{5.23}$$

$$u_1 = \mu_2(\mu_1 r_1 - \lambda) + D_1 \tag{5.24}$$

$$D_k = U_k(1) + \lambda U_{k-1}(1) - a_k\theta\, c_{k-1}(1)U_{k-2}(1) \tag{5.25}$$

(3)ここで，$u_k = (-\mu_2)^k \prod_{i=1}^{k} \delta_2(i) f_k$ (5.26)

と置き(5.23)式に代入し漸化式を解くと次を得る.

$$f_k = -u_1/\mu_2 + \sum_{j=2}^{k} D_j/(-\mu_2)^j / \prod_{i=1}^{j} \delta_2(i) \tag{5.27}$$

(4) 一方，(5.24)式は系3.1を用いると

$$D_k = (-\mu_1)^k \prod_{i=0}^{k-1} \delta_1(r_0-i)\left\{\lambda-\mu_1\delta_1(r_0-k)+\theta\mu_2\delta_2(k)\right\}$$

本式を(5.27)式へ代入し，さらに(5.26)式を求めると次を得る.

$$u_k = (-\mu_2)^k \prod_{i=1}^{k} \delta_2(i) \times \left[\lambda-\mu_1 r_1 + \sum_{j=1}^{k} \frac{(-\mu_1)^j \prod_{i=0}^{j-1} \delta_1(r_0-i)\left\{\lambda-\mu_1\delta_1(r_0-j)+\theta\mu_2\delta_2(j)\right\}}{(-\mu_1)^j \prod_{i=1}^{j} \delta_2(j)}\right] \tag{5.28}$$

(5) ところで，(5.22)式において $k = r_0$ と置くと，

$$u_{r_0} = U'_{r_0}(1) + \mu_1\delta_1(r_0-r_0)\,U'_{r_0-1}(1) = U'_{r_0}(1)$$

となるため，(5.28)式より以下を得る

$$U'_{r_0}(1) = u_{r_0} = (-\mu_2)^{r_0} \prod_{i=1}^{r_0} \delta_2(i) \times \left[\rho_1 - r_1 + \sum_{j=1}^{r_0} \nu^j \left\{\rho_1 - \delta_1(r_0-j) + \delta_2(j)\theta/\nu\right\} \prod_{i=1}^{j} \delta_1(r_0-i+1)/\delta_2(i)\right]$$

(6) $U'_{r_0}(1) \geqq 0$ (r_0 が奇数の時) または $U'_{r_0}(1) \leqq 0$ (r_0 が偶数の時)となるのは，

$$\rho_1 - r_1 + \sum_{j=1}^{r_0} \nu^j \left\{\rho_1 - \delta_1(r_0-j) + \delta_2(j)\theta/\nu\right\} \prod_{i=1}^{j} \delta_1(r_0-i+1)/\delta_2(i) \leqq 0 \qquad \square$$

5.8 数値計算手順

5.8.1 状態確率

前節までで母関数は次の形式で得られた.

$$F_n(z) = D_n(z) / U_{r_0}(z) \tag{5.14}$$

前章での根の分布に関する考察より，系が安定である場合，$U_{r_0}(z)$ は次の通り記述できる.

$$U_{r_0}(z) = z^{1+\lfloor r_0/2 \rfloor}(z-1)\prod_{k=1}^{\lceil r_0/2 \rceil - 1}(z-\eta_k)\prod_{l=1}^{r_0+1}(z-\zeta_l)$$

一方，$D_n(z)$ は$U_{r_0}(z)$ と$|z| < 1$の根を共有する条件から，次のとおり書ける.

$$D_n(z) = z^{1+\lfloor r_0/2 \rfloor}(z-1)\prod_{k=1}^{\lceil r_0/2 \rceil - 1}(z-\eta_k)G_n(z)$$

したがって，(5.14)式は次の様に表わされる.

$$F_n(z) = G_n(z) / \prod_{l=1}^{r_0+1}(z-\zeta_l) \tag{5.29}$$

$G_n(z)$の次数は系2から$0 \leq n < r_0$の時$(2r_0 - n)$，$n = r_0$の時$(r_0 - 1)$となる．これを(5.29)式の分母側の次数(r_0+1)と比較すると，$n = r_0$の時を除き(5.29)式の分子側の次数が高いか等しい．そこで分子を分母で除し次の形式に変形する.

$$F_n(z) = P_n(z) + Q_n(z) / \prod_{l=1}^{r_0+1}(z-\zeta_l) \quad n = 0,1,..,r_0-1 \tag{5.30}$$

ここで，$P_n(z)$ は最高次数が $(r_0 - n - 1)$,$Q_n(z)$ は最高次数がr_0 のzの多項式である.
(5.30)式の第２項を部分分数分解すると次式となる.

$$Q_n(z) / \prod_{l=1}^{r_0+1}(z-\zeta_l) = \sum_{l=1}^{r_0+1} R_{nl} / (z-\zeta_l) = \sum_{l=1}^{r_0+1} -R_{nl} / \zeta_l / (1 - z/\zeta_l) \tag{5.31}$$

$|\zeta_l| > 1$であるため，$|z| < 1$の範囲に対し，$|z/\zeta_l| < 1$となり，$|a| < 1$の時

$1/(1-a) = \sum_{n=1}^{\infty} a^n$ となる事実を用いると,

$$Q_n(z)/\prod_{l=1}^{r_0+1}(z-\zeta_l)=\sum_{l=1}^{r_0+1}-R_{n,l}/\zeta_l\sum_{m=0}^{\infty}(z/\zeta_l)^m=\sum_{m=0}^{\infty}\sum_{l=1}^{r_0+1}(-R_{n,l}/\zeta_l^{m+1})\cdot z^m \qquad (5.32)$$

したがって，(30)式は

$$F_n(z)=P_n(z)+\sum_{m=0}^{\infty}\sum_{l=1}^{r_0+1}(-R_{n,l}/\zeta_l^{m+1})\cdot z^m=\sum_{m=0}^{r_0}P_{n,m}z^m+\sum_{m=0}^{\infty}\sum_{l=1}^{r_0+1}(-R_{n,l}/\zeta_l^{m+1})z^m \qquad (5.33)$$

一方，母関数の定義から$F_{k_2}(z)=\sum_{n=k_2}^{\infty}P(n,k_2)z^{n-k_2}$であるため，(33)式と対応させると

$$P(k_0,k_2)=P_{k_2,k_0-k_2}+\sum_{l=1}^{r_0+1}-R_{k_2,l}/\zeta_l^{k_0-k_2+1} \qquad k_2\leqq k_0\leqq r_0+1,\ k_2\neq r_0 \qquad (5.34a)$$

$$P(k_0,k_2)=\sum_{l=1}^{r_0+1}-R_{k_2,l}/\zeta_l^{k_0-k_2+1} \qquad k_0>r_0+1,\ k_2\neq r_0 \qquad (5.34b)$$

以降，$R_{k_2,1}$を求める．(5.31)式を通分すると，

$$Q_n(z)/\prod_{l=1}^{r_0+1}(z-\zeta_l)=\sum_{l=1}^{r_0+1}R_{n,l}\prod_{j=1,j\neq l}^{r_0+1}(1-z/\zeta_j)/\prod_{l=1}^{r_0+1}(-\zeta_l)/\prod_{l=1}^{r_0+1}(1-z/\zeta_l) \qquad (5.35)$$

$$Q_n(z)=\sum_{l=1}^{r_0+1}R_{n,l}\prod_{j=1,j\neq l}^{r_0+1}(1-z/\zeta_j) \qquad (5.36)$$

本式に$z=\zeta_l$を代入すると，

$$Q_n(\zeta_l)=R_{n,l}\prod_{j=1,j\neq l}^{r_0+1}(1-\zeta_l/\zeta_j) \qquad (5.37)$$

よって，$R_{n,l}=Q_n(\zeta_l)/\prod_{j=1,j\neq l}^{r_0+1}(1-\zeta_l/\zeta_j)$ (5.38)

この値を(5.34)式に代入することによって，$P(k_0,k_2)$が得られる．

一方，$n=r_0$の時は(5.28)式を直接,部分分数展開することによって，同様の手順で状態確率を得ることができる．

5.8.2 特性値

（1）平均行列長

システム内の全呼数の平均値Qおよび待ち行列上の平均待ち行列長Q_w,窓口R_1およびR_2上の平均呼数（待ちを含む）Q_{R1},Q_{R2}はそれぞれ次のように求められる.

$$Q=\sum_{k_2=0}^{r_0}\sum_{k_0=k_2}^{\infty}k_0\cdot P(k_0,k_2)=\sum_{k_2=0}^{r_0}\left\{F'_{k_2}(1)+k_2\cdot F_{k_2}(1)\right\}$$

$$Q_w=\sum_{k_2=0}^{r_0}\sum_{k_0=r_0+1}^{\infty}(k_0-r_0)\,P(k_0,k_2)=\sum_{k_2=0}^{r_0}\left\{F'_{k_2}(1)+(k_0-r_0)F_{k_2}(1)+\sum_{k_0=k_2}^{r_0}(r_0-k_0)\,P(k_0,k_2)\right\}$$

$$Q_{R1}=\sum_{k_2=0}^{r_0}\sum_{k_0=k_2}^{\infty}k_1\cdot P(k_0,k_2)=\sum_{k_2=0}^{r_0}\left\{(r_0-k_2)F_{k_2}(1)+\sum_{k_0=k_2}^{r_0}(k_0-r_0)\,P(k_0,k_2)\right\}$$

$$Q_{R2}=\sum_{k_2=0}^{r_0}\sum_{k_0=k_2}^{\infty}k_2\cdot P(k_0,k_2)=\sum_{k_2=0}^{r_0}k_2\cdot F_{k_2}(1)$$

（2）平均通過時間

システム内を通過する時間の平均値（T）は，Littleの定理を用いて$T=Q/\lambda$で得られる．同様に，窓口R_1，R_2の平均通過時間（T_{R1}，T_{R2}）および待ち行列Q_0の平均滞留時間（T_Q）は，それぞれ$T_{R1}=Q_{R1}/\lambda$，$T_{R2}=Q_{R2}/\lambda$，$T_Q=Q_W/\lambda$で得られる.

（3）利用率

窓口R_1, R_2の利用率ρ_{R1}, ρ_{R2}は次のように定義できる.

$$\rho_{R1}=\sum_{k_2=0}^{r_0}\sum_{k_0=k_2}^{\infty}\delta_1(k_1)/r_1\cdot P(k_0,k_2)$$

$$\rho_{R2} = \sum_{k_2=0}^{r_0} \sum_{k_0=k_2}^{\infty} \delta_2(k_2)/r_2 \cdot P(k_0,k_2)$$

これらは母関数と各状態確率から計算することができる.

5.8.3 数値計算手順

以上の結果を数値計算手順として示すと次のとおりとなる.

ステップ１：U_{r_0}を(5.12)式の漸化式を用いて求める.

ステップ２：$U_{r_0}(z)=0$を解き，絶対値が１より大の根ζ_l $(l=1,2,\cdots,r_0+1)$および絶対値が１より小の根η_m $(m=1,2,\cdots,\lfloor (r_0+1)/2 \rfloor)$を求める．もし，上記の根が得られなかった場合は，系が安定状態を持たないため計算を終了する.

ステップ３：$D_{r_0}(z)$を(5.15)式から求める．未知の確率P(m,n)を含むzの多項式として得られる.

ステップ４：$D_{r_0}(\eta_m)$を計算し未知の確率P(m,n)に関する一次式を$\lfloor (r_0+1)/2 \rfloor$個得る.

ステップ５：平衡方程式(5.4)式のうち，

$P(k_0,k_2)$　$0 \leqq k_0 \leqq r_0-1, 0 \leqq k_2 \leqq k_0$および$P(r_0+1,k_2)$ $0 \leqq i \leqq \lfloor r_0/2 \rfloor -1, 2+2i \leqq k_2 \leqq r_0$に関する平衡方程式を選択する．ここで得られる式の数は，

$\left\{ r_0(r_0+1)/2+\lfloor r_0/2 \rfloor \cdot \lceil r_0/2 \rceil \right\}$個，未知数の数は$\left\{ r_0(r_0+1)/2+\lfloor r_0/2 \rfloor \cdot \lceil r_0/2 \rceil + \lceil r_0/2 \rceil \right\}$個である.

ステップ6：ステップ4およびステップ5で得た式を連立させて解く.

すなわち，式の数は$\left\{ r_0(r_0+1)/2+\lfloor r_0/2 \rfloor \cdot \lceil r_0/2 \rceil + \lfloor (r_0-1)/2 \rfloor \right\}$となるのに対し，未知数についてはステップ4に含まれるものはステップ５のものに完全に包含されているため,その数は$\left\{ r_0(r_0+1)/2+\lfloor r_0/2 \rfloor \cdot \lceil r_0/2 \rceil + \lceil r_0/2 \rceil \right\}$である．したがって，未知数の数が式の数より１つ多い同次連立一次方程式となる．任意の未知数，仮にP(0,0)=1と置き他の未知数について解き，$P(k_0,k_2)=p(k_0,k_2)P(0,0)$の形式で解を得る.

ステップ7：ステップ6で得た$P(k_0,k_2)$を用いて，$D_n(z)$ $(n=0,1,2,\cdots,r_0)$を求める．

ステップ8：$D_n(z)$ を$z^{1+\lfloor r_0/2\rfloor}(z-1)\prod_{k=1}^{\lfloor (r_0-1)/2\rfloor}(z-\eta_k)$で除し，その商$G_n(z)=g_n(z)\cdot P(0,0)$ を得る．

ステップ9：確率の総和が１であることを用い，P(0,0)を定める．すなわち，

$$\sum_{n=0}^{r_0} F_n(1)=\sum_{n=0}^{r_0} G_n(1)/\prod_{l=1}^{r_0+1}(1-\zeta_l)=\sum_{n=0}^{r_0} g_n(1)\cdot P(0,0)/\prod_{l=1}^{r_0+1}(1-\zeta_l)=1$$

$$P(0,0)=\prod_{l=1}^{r_0+1}(1-\zeta_l)/\sum_{n=0}^{r_0} g_n(1)$$

これにより，母関数は次の形式で決定できる．

$$F_n(z)=g_n(z)\cdot P(0,0)/\prod_{l=1}^{r_0+1}(z-\zeta_l)$$

ステップ10：$P(k_0,k_2)$ （$k_2\neq r_0$）の計算

$F_{k_2}(z)$ を$\prod_{l=1}^{r_0+1}(z-\zeta_l)$で除し，その商$P_{k_2}(z)=\sum_{n=0}^{r_0-k_2+1} P_{k_2,n}z^n$ および余り$Q_{k_2}(z)$

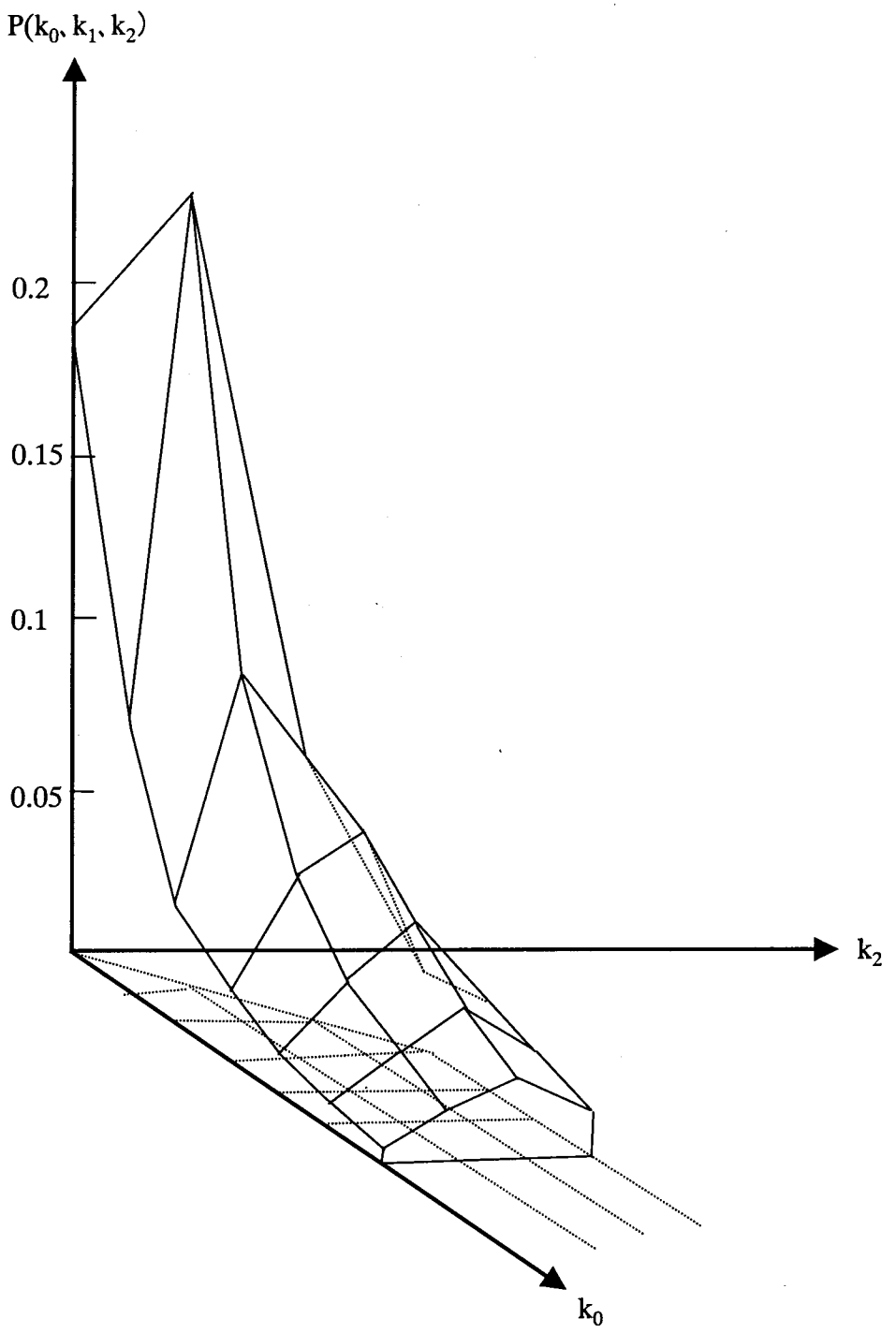
を得る．(5.35)式を用いて$R_{k_2,l}$を求め，(5.34)式へ代入して$P(k_0,k_2)$ を得る．

ステップ11：$P(k_0,k_2)$ （$k_2=r_0$）の計算

$R_{n,l}=G_n(\zeta_l)/\prod_{j=1,j\neq l}^{r_0+1}(1-\zeta_l/\zeta_j)$ を求める．これを用いて下式で得られる．

$$P(k_0,r_0)=\sum_{l=1}^{r_0+1}-R_{r_0,l}/\zeta_l^{k_0-r_0+1}$$

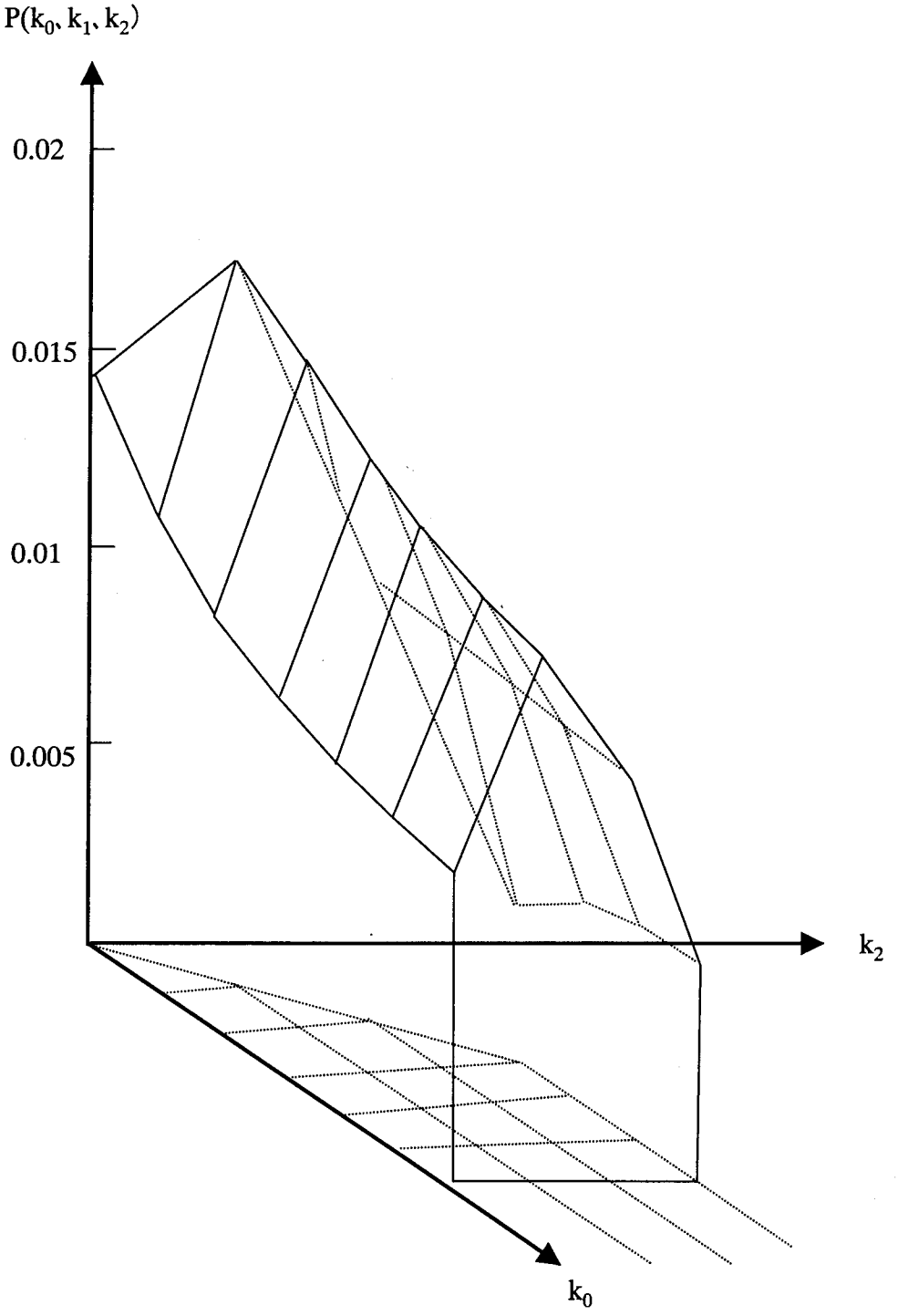
本手順で得られた数値計算結果の一例を図5.17,5.18に示す．



条件: $r_0=3$, $r_1=2$, $r_2=3$, $\lambda=1$, $\mu_1=1.2$, $\mu_2=0.8$, $\theta=0$

特性値: $Q_0=1.5$, $R_1+Q_1=0.4$, $R_2+Q_2=1.3$

図 5.17 状態確率と特性値の計算例1



条件: r_0=3, r_1 =1, r_2=3, λ =1, μ_1=1.2, μ_2=0.8, θ =0

特性値： Q_0=19.5 , R_1+Q_1 =1.6, R_2+Q_2 =1.3

図 5.18 状態確率と特性値の計算例2

5.9 結言

本章ではソフトウェア資源とハードウェア資源の関係に代表されるような、資源と資源の間に階層的な関係が存在するシステムの性能評価を行うための一般的なモデルを提案し、その解析法および系の安定性に関する条件を理論的に導き、性能評価に用いられる各種特性値の数値計算の手順を明らかにした。そして本モデルを用いて、トランザクション処理システムの性能に大きな影響を与えるタスク構成の評価を行い、システムのトラヒック設計に極めて有効に適用できることを示した。

ここで提案した系内客数制限を持つ待ち行列モデルは、窓口の間に階層的な関係があるシステムを表現することができるため、計算機システムにおける多くの局面に適用が可能である。いくつか具体的な問題を例示すると次のようである。

まず、第一はジョブ・スケジュールやマルチプログラミングの多重度のモデリングである。中小型機用のＯＳには、メモリをパーティションに分割しておき、そのパーティションに１つのジョブを対応させる制御が行われるものがある。パーティションに空きがない時に到着したジョブは、スケジュールされずに待ちに入る。一方、パーティションを得たジョブは、そのパーティションを専有しながら、他のシステム資源（ＣＰＵやＩＯ装置など）を使用して処理をすすめ、処理の終了時にパーティションを空けて退去してゆく。このようなシステムにおいて、ジョブのスループットを増大させるためには、メモリを増強し、パーティション数を増やすのが得策なのか、ＣＰＵやＩＯ装置のグレードアップが得策なのかなどは、システム設計における重大な関心事である。この場合、本章で提案したモデルの中の客数制限値をパーティションの数やジョブ多重度に対応させることによってモデル化でき、解析が可能となる。

また、本論文の第３章で取上げたジャーナル処理ルーチンに代表されるような逐次的再使用可能ルーチン（SRR:Serially Reusable Routine)も本モデルの対象となる。このケースでは、ＳＲＲ待ちはＳＲＲ全体の保留時間に左右されるが、その保留時間は、ＣＰＵやＩＯでの処理時間およびその待ち時間に依存するという階層関係がある。ＣＰＵやＩＯの台数や速度が系通過時間や最大スループットにどの様に影響するかが関心事であるが、このためには本論文で提案した様なモデルが適用できる。

さらに、第４章で示したように大容量記憶装置（Mass Storage System)の評価においても系内客数制限を取り入れたモデル化が有効である。

このように本章で提案したモデルは、資源と資源の間に階層的な関係が存在するシステムの性能評価に全般に広く適用ができ、トラヒック設計に強力な手段を与えるものである。

第６章

結論

本研究は、トランザクション処理システムにおいて中核を成す技術である、電文制御方式、ジャーナル処理方式、大容量記憶装置制御方式およびそれら各方式の性能評価手法に関するものである。

以下に２章から５章の各章において得られた成果を総括し、本論文の結論とする。

第２章 トランザクション処理システムにおける電文制御方式の設計

トランザクション処理システムが多種多様な業務に適用できるためには、業務の骨格を形成する電文の流れに関する設計に大きな自由度が必要である。特に、トランザクション処理システムにおいては、トランザクション処理モニタ（ＴＰモニタ）がＯＳと業務プログラムとの間に介在し、業務プログラムからの要求に応じて目的とする端末に電文を送信したり、逆に通信網を経由して受信した電文を対応する業務プログラムへ引渡すなど、電文の送受を制御する。従って、業務の多様性に答えられるためには、このＴＰモニタにおける電文の制御機能の多様なサービスへの適用性の確保が重要である。

本章ではこのような課題の解決を目指して、トランザクション処理システムにおける汎用的な電文制御方式に関する研究を行った。以下にその結論を示す。

(1) トランザクション処理システムにおける電文制御機能について、具体的なシステムの特徴を分析した上で、要求条件として整理・体系化した。その第一はトランザクション処理システムに特有な問合せ型、交換型、集配信型などの多様

な電文の流れを効率よく制御できること、第２に端末の多様化と端末機能の高度化に即応できる柔軟性、拡張性の確保、第３として端末交信処理の効率化である。

(2) トランザクション処理システムに固有で多様な電文の流れを実現し、かつ処理効率を向上する方式として、端末との交信処理と業務処理を分離してそれぞれを別のタスクに割り当て、並行処理を行い、両タスクの間には入力電文および出力電文の待ち行列を介在させる方式を設計した。

(3) 汎用性の高い電文のスケジュール機構としてゲート制御方式を考案した。これは、メカニズムとポリシを分離する考え方に基づく方式で、電文の通路にゲートを設け、ゲートの開閉に応じて電文の通過を制御するメカニズムのみをシステムに用意し、そのゲートをどのように開閉するかというポリシは、サービス設計者に委ねることにより、様々なサービスの電文制御を可能とする方式である。

これらの設計結果は、ＤＩＰＳのトランザクション処理モニタの実装に取り入れられ、大規模なトランザクション処理システムの開発に適用され、多種多様な業務への適用性の高い標準方式として確立した。

第３章 ジャーナル処理方式の設計

トランザクション処理システムに故障が生じた時には、データベースの内容や端末との間で送受された電文の内容を矛盾の無い状態に復元した上でシステムの正常な運転を迅速に再開することが必須の要件である。この目的のために運転中に再開処理に必要な情報をトランザクション毎に時系列順にジャーナルとして記録する。通常処理の途中ですべてのトランザクションがこのジャーナル処理を行うため、その性能改善は、システム全体の性能改善に大きく寄与する。

本章では高トラヒックなトランザクション処理システム向けのジャーナル機能について、ソフトウェア制御の最適化による新しい高性能ジャーナル方式を提案した。以下に本研究の成果を要約して示す。

(1) シミュレーションによってジャーナル処理方式の性能評価を行い、性能上の隘路を詳細に分析した結果、標準的ジャーナル処理ルーチンでは、高トラヒック

になるとタスク不足の現象が現れるが、これはタスク数の不足ではなく、ジャーナル処理ルーチンがボトルネックとなり、タスクが次々とジャーナル専有待ち陥ったため空きタスクが不足するためであることを明らかにした。

(2) ジャーナル処理ルーチンは、トラヒックが高くなるとその処理時間が急激に長くなる。これはCPU待ち時間が延び、ジャーナル保留時間が増大することが主な原因である。そして、これは排他制御による無効保留が影響していることを明らかにした。

(3) 排他制御を入出力割込み処理のレベルで実現し、無効保留を極力低下させ、処理時間を短縮する新たな高速ジャーナル処理方式を提案した。さらに、この提案方式が、システム全体の性能を約２０％改善する効果をもたらすことをシミュレーションによって明らかにした。

これらの研究結果に基づいて、ＤＩＰＳ用のＴＰモニタに本改良方式を実装し、高トラヒックな大規模トランザクション処理システムの実用に供し、所期の効果を得た。

第４章　大容量記憶装置制御方式のトラヒック設計

トランザクション処理システムの運転においては、処理の履歴情報を格納するジャーナルやオンラインファイルのバックアップ情報などのように、オンラインでの直接のアクセスは無いが、大容量で且つ長期間保存すべきファイルが必要である。これらには、通常磁気テープ装置が用いられてきたが、システムの規模の増大に伴い、量が急増し、操作コストの増大、操作誤りなどテープ操作の煩雑さに起因する各種の問題が大きくなってきた。また、必ずしも短い応答時間は必要無いが、低コストのオンラインで用いるファイルに対するニーズも顕在化してきた。

この様な要請に応えるために自動操作機構をもった磁気テープ装置と磁気ディスク装置とを組み合わせた大容量記憶装置(Mass Storage System :MSS) が開発された。

本章では、このような大容量記憶装置を効率的に制御するためのトラヒック設計の方法を提案し、それに基づく設計結果の妥当性、制御方式の有効性を定量的に実証した。

その結論を要約すると次の通りである。

(1) ＭＳＳの評価モデルとして、系内客数制限をもつ開放形の待ち行列ネットワークを提案した。そして、その解析方法を与え、平衡状態における各種特性値を得る方法を明らかにした。その手法を用いて、データの大きさとアクセス時間およびスループットの関係など具体的な特性値を明らかにした。

(2) 系が平衡状態にあるための条件について考察し、理論的な限界スループットを明らかにした。また、一部の装置が故障などで使用できなくなった場合に、応答時間を一定時間に保つために、入力の規制をどの程度にすべきかを定量的に明らかにし、システムの運用に反映した。

(3) モデルの解析結果に基づく数値計算例を実測値と比較し、良く一致することを示し、モデルの妥当性を実証した。

この結果により、ＭＳＳの具体的な用途として、各種のデータ通信システムにおけるファイルバックアップ処理およびNTT公衆データ通信システム（DEMOSサービス）における安価なファイル保管庫サービスなどへの適用が妥当であると判断し、それぞれに必要なソフトウェアを開発、商用に供した。

第５章　トランザクション制御方式のトラヒック設計

本章ではソフトウェア資源とハードウェア資源の関係に代表されるような、資源と資源の間に階層的な関係が存在するシステムの性能評価を行うための一般的なモデルを提案した。本論文の第２章の電文制御方式、第３章ジャーナル処理方式や第４章大容量記憶装置のそれぞれにおいて個別的に示してきたトラヒック設計法をさらに高度化し、より広い設計問題に適用可能なトラヒック設計法に一般化したものである。
得られた主要な成果は次のとおりである。

(1) タスクをハードウェア資源と関係づけて評価する手法として、ハードウェアを窓口とする待ち行列モデルにおいて、その一部または全体の中に存在し得る客数を一定数に制限することによってタスクの存在を表現する系内客数制限モデルを提案した。

(2) 外からの客の到着が無い閉じた系内客数制限モデルについて解析の方法を与え、それを用いてタスク構成を評価した。一定の条件のもとでは、ハードウェ

アの性能向上よりもタスク構成の変更の方がシステムの性能向上をもたらすが、一定の範囲を超えると、ハードウェアの性能がシステムの性能の支配要因になることを定量的に示すことができるなど、本モデルが当初の要求に応える有効なものであることを示した。

(3) より一般的なモデルとして、外部から客の到着がある開放型の系内客数制限モデルについて、その解析の方法を与えた。また、系が安定であるための必要十分条件を明らかにし、理論的な限界値を解析的に得る方法を与えた。また、解析手法をもとに実際的な数値計算の手順を与えた。なお、本モデルはタスク構成の問題だけではなく、マルチプログラミングにおける多重度、逐次的再利用可能ルーチン（ＳＲＲ：Serially Reusable Routine）やＭＳＳの評価など、資源と資源の間に階層的な関係が存在する場合に広く適用ができる一般的なモデルであることを示した。

以上の成果は順次ＤＩＰＳオペレーティングシステムおよびトランザクション処理モニタに反映し実用化した。また、性能評価の結果や手法については、システムの設備設計法やシステムジェネレーションにおけるパラメータ設計法、運用マニュアルなどに反映した。そしてこれらは、1975年以降ＮＴＴが受託した郵政省貯金局システム、全国銀行協会為替交換システム、運輸省車検登録システム、社会保険庁システムなど全国規模の大規模トランザクション処理システムなど約２００システム（約１０００ＣＰＵ）の設計・建設・運用に適用された。

本論文の成果がこれらシステムの安定したサービスの提供に多少とも貢献ができたと信じる。

1980年代以降半導体技術の急速な進展によるマイクロプロセッサや半導体メモリの出現を契機に計算機システムは小型化、分散化の方向、いわゆるダウンサイジングへ大きく変化した。しかし、社会の基幹システムであるトランザクション処理システムは、今日に至るもなお大型計算機システムで実現されており、本研究の成果が引き続き生きている。

一方、今後インターネットなどによる電子商取引きなど本格的なトランザクション処理がワークステーションやパソコンで実現されることも予想されるが、現状でのこれら機種のオペレーティングシステムやトランザクション処理モニタには、本論文で取り上げた問題を始めさまざまな課題があり、基幹システムへの適用にはまだ不十分である。本研究の成果がワークステーションやパソコンなどに反映され、実装されて

くれば、コストパフォーマンスの高いトランザクション処理システムが実現でき、これからの情報流通社会の一層の発展へ貢献できると信じる。

参考文献

[1] J.M.Andrade, et al : "Building An On-Line Transaction Processing System On UNIX System V ", CommUNIXations, AT&T (1989)

[2] A. Dan and P. S. Yu : Performance Analysis of Buffer Coherency Policies in a Multisystem Data Sharing Environment , IEEE Trans. on Parallel and Distributed Systems, Vol. 4, No. 3, March (1993).

[3] D.Gawlick: Processing Hot Spots in High Performance Systems, Proc.of IEEE Compcon,Feb.(1985).

[4] D.Gawlick and D.Kinkade: Varieties of Concurrency Control in IMS/VS FastPath",IEEE Database Engineering,Vol.8,No.2(1985).

[5] J.N. Gray, et al : "One Thousand Transactions Per Second ", TR85.1, Tandem Computers (1984)

[6] J.Gray and A.Reuter : "Transaction Processing:Concept and Techniques" Morgan Kaufmann Publishers (1993)

[7] T. Haerder, A. Reuter:Principle of Transaction-Oriented Database Recovery, ACM Computing Surveys, Vol. 15 , no. 4, Dec (1983).

[8] A.G.Konheim and M.Reiser : " A queuing model with finite waiting room and blocking", JACM, Vol.23, No.2, pp.328-341 (1976)

[9] A.G.Konheim and M.Reiser : " Finite Capacity Queuing Systems with Applications in Computer Modeling", SIAM J.Computing, Vol.7, No.2, pp.210-229 (1978)

[10] Lavenberg, S.S. and Slutz, D.R.:Regenerative Simulation of a Queuing Model of an Automated Tape Library, IBM J. Res. Dev., Vol.19, No.5, pp.463－475 (1975)

[11] Makino, T.: On the Mean Passage Time Concerning Some Queuing Problems of the Tandem Type, J. OR Soc. Jpn., Vol.7, No.1,pp. 17－47 (1964)

[12] K.Matsuda,E.Moriya,R.Ikeda,H.Ikehama : Message Control in a Real Time System, Review of ECL,Vol.26, No.1－2,pp.24－32(1978)

[13] K.Matsuda,T.Ohkubo,et al :Operating System Performance Evaluation, Review of ECL, Vol.28,No.3-4,pp185-194(1980)

[14] K.Matsuda,I.Kogiku,S.Sato :Mass Memory System Performance Measurement, Review of ECL,Vol.29,No.5-6,pp451－463(1981)

[15] Misra, P. N.: Capacity Analysis of the Mass Storage System, IBM Syst. J., Vol.20, No.3, pp.346－360 (1981)

[16] Neuts, M.F.: Two Queues in Series with a Finite, Intermediate Waiting Room, J. Appl. Prob., Vol.5, No.1, pp.123－142 (1968)

[17] E.I.Organick : "The Multics System:an examination of its structure" M.I.T.Press (1972)

[18] E. Rahm : Performance Evaluation of Extended Storage Architectures for Transaction Processing, "ACM SIGMOD, pp308-317 (1992).

[19] C.H.Sayer : "Approximate Solution of Queuing Networks with Simultaneous Resource Possession", IBM J.Res. Develop, Vol.25, No.6 (1981)

[20] Suzuki T.: Ergodicity of a Tandem Queue with Blocking, J.OR Soc. Jpn, Vol.7, No.2, pp.68-75 (1964)

[21] K. Takashima, et al. : "A Large-Scale Data Processing System:DIPS-1", Proc.1st USA-Japan Computer Conf., pp.193－202 (1972)

[22] Tsuruho, S., Matsuda, K., et al.: Mass Storage Systems Performance Analysis Using a Queuing Model, Proc. of 3rd UJCC, pp.320－324 (1978)

[23] W.Wulf, E.Cohen, et al: "HYDRA:The Kernel of a Multiprocessor Operating System" ,CACM, 17 No.6, p.337 (1974)

[24] W.Wulf, E.Cohen, et al : "Policy/Mechanism Separation in HYDRA, ACM Operating Systems Review" , 9, No.5, p.132 (1976)

[25] P. S. Yu and A. Dan : Performance Evaluation of Transaction Processing Coupling Architectures for Handling System Dynamics, IEEE Trans. on Parallel and Distributed Systems, Vol.. 5, No. 2, Feb. (1994)

[26] "Customer Information Control System (CICS) General Information" , IBM GC33-0155, (1991)

[27] "Encina Transaction Processing System TP Monitor ", TP-00-D146 (1991)

[28] IBM 3850 Mass Storage System (MSS) Principles of Operation: GA 32-0035-0 (1978)

[29] "Information Management System (IMS) General Information ", IBM

GC26-4275 (1991)

[30] 浅野他："FACOM 6450 大容量記憶システム"， FUJITSU, Vol.31, No.1, p.93 (1980)

[31] 石黒："Gate のある M/M/m 待行列ネットワークモデルの解析", 情報処理, Vol.29, No.7, pp.614-620 (1978)

[32] 伊藤：DIPS 用超大容量記憶装置の実用化、通研実報、Vol.29, No.2, pp.141－150 (1980)

[33] 伊藤：実時間制御と多重処理、信学誌、Vol. 58, No. 11, 1975

[34] 伊藤、川田：超大容量記憶装置の動向、情報処理、Vol.19, No.5, pp.465－471(1978)

[35] 大橋・和佐野他：リアルタイムシステムの障害処理方式，通研実報，Vol.. 26, No. 8, p. 2261 (1977).

[36] 大前他："郵政省為替貯金業務総合機械化システムの概要"，施設，Vol.30, No.11, pp.49-55 (1978)

[37] 奥川光太郎：代数学、pp.161-175, 共立出版、東京(1956)

[38] 川嶋、大南、松田："系内呼数制限を有する待ち行列モデルの計算機システムへの応用",情報学会全国大会、1H.1 (1978)

[39] 川島他："閉そくのある有限待合式の２段直列形待ち行列モデルに対する数値解法", 信学会論文誌, Vol.J 64-B, No.8, pp.769-776 (1981)

[40] 河内他:"リアルタイム電文スケジュールの汎用化手法",情処全大, No.47 (1976)

[41] 喜連川優他："分数トランザクション処理"，リックテレコム (1994)

[42] 河野健一郎，亀田寿夫：OLTP のための並列コンピュータシステムにおける機器並列度の応答性能に及ぼす効果，情報処理学会論文誌，Vol. 37, No. 7 (1996).

[43] 関口他："DIPS-1 システム実用化の概要", 通研実報, Vol.21, No.10, pp.1787－1796 (1972)

[44] 関口他："総合報告 DIPS の実用化", 電子通信学会誌, Vol.57, No.10, pp.1139－1159 (1974)

[45] 関口他："大型電子計算機 DIPS について"，情報処理学会誌, Vol.13, No.3, pp.136－144 (1972)

[46] 高橋・中島他：超大容量記憶装置の動作特性，通研実報，Vol.30, No.2, p.425(1981)

[47] 高村、松田：リアルタイムシステムにおけるタスク構成の解析的評価、

通信学会電子計算機研究会（ＥＣ７９－２８），1979
[48] 高村、松田: 大規模実時間処理システムの制御方式、DIPS シンポジウム、1979
[49] 田中穂積：並列循環待ち行列を用いたオンラインシステムの解析，信学会論文誌、Vol. 53.C, No. 10, 1970
[50] 鶴保，藤，松田他：超大容量記憶装置制御方式，通研実報，Vol.30, No.2, pp.409-423(1981)
[51] 鶴保、松田、大久保：１０４ＯＳの制御方式と動作特性、電気通信研究所研究討論会,1976
[52] 鶴保、松田、大南: タスクを考慮した待ち行列モデルに関する一考察、53 年度信学会大会、1978.
[53] 徳山、松田: データ通信システムにおける待ち行列の問題、53 年度信学会大会、1978.
[54] 根岸、米田：大容量記憶システムＭＳＳの性能評価用シミュレーションモデルに関する考察、情報処理学会論文誌、Vol.23, No.4, pp.421－427 (1982)
[55] 野瀬，小松：対話型計算機システムシミュレータにおけるモデル記述性の高度化，情処学会論文誌，vol. 34, no. 9, pp. 2048-pp. 2059 , Sep （1993）
[56] 拝原、大久保他：実時間処理システムのシミュレーションによる評価例、情報学会性能評価研究会、SE15-2, 1976
[57] 橋田温：情報システムにおける待ち行列理論の応用、情報処理、Vol.18, No. 2, 3, 1977
[58] 藤原他：階層構成ファイル記憶の構成法、通研実報、Vol.29, No.6, pp.1129－1137 (1980
[59] 松田、池田：トランザクション処理システムにおけるジャーナル処理の性能評価、電子情報通信学会論文誌（条件付き採録）
[60] 松田、鶴保：待行列モデルによる大容量記憶装置ＭＳＳの性能解析、情報処理学会論文誌，Vol.25, No.1, pp.125-132,(1984)
[61] 松田、大久保、大南、中島：オペレーティングシステムの特性評価、通研実報、Vol. 28,No.12,pp2697－2709(1979)
[62] 松田、大南：ＭＳＳの制御手法と動作特性、電気通信研究所研究討論会、1978
[63] 松田、大南他：超大容量記憶装置の特性に関する一考察、第 18 回情処全大、p.161 (1977)
[64] 松田他:“通信制御処理装置を用いた端末アクセス法”，情処全大，No.35

(1976)

[65] 松田他:“リアルタイムシステムの電文制御方式”，通研実報, Vol.26, No.8, p.2221 (1977)

[66] 松田、大南他：系内呼数制限を有する待ち行列モデルの計算機システムへの応用、情処学会全国大会、1978.

[67] 松田、佐藤、村田：ＭＳＳステージング処理時間の最適化について、情処学会 23 回全国大会、1981.

[68] 松田、杉村：ＳＲＲ待ち時間改善のための一手法、情処学会 24 回全国大会、1982.

[69] 向坂他:“リアルタイムパッケージ”,通研実報, Vol.24, No.1, p.53 (1975)

[70] 村上他：大規模データベースとその実現技術、情報処理、Vol.23, No.10, pp.955－961 (1982)

[71] 村田、森、拝原、松田、森元：１０４－０３システムの分散処理方式、通研実報、 Vol.31,No.8,pp1437－1448,(1982)

[72] 森戸晋，相沢りえ子：SLAM によるシステムシミュレーション入門，構造計画研究所(1986)

[73] 柳他：“総合化された運輸省自動車登録・検査システム”，施設,Vol.31,No.3, pp.23－34(1979)

[74] 吉田・高村他:“DIPS-104 OS の実用化”，通研実報，Vol.26, No.8, p.2177(1977)

[75] 吉野他：“新全国銀行データ通信システム”，施設，Vol.31, No.3, pp.35－42(1979)

[76] 渡辺，Ｊ.グレイ：“OLTP システム”，近代科学社(1995)

[77] 渡辺編：“オープン OLTP システム入門”，日経 BP 出版センタ (1995)

[78] Perfomable/1 操作マニュアル，NTT ソフトウェア株式会社（1995)

[79] Perfomable/1 リファレンスマニュアル, NTT ソフトウェア株式会社(1995)

関連発表論文一覧

学会論文誌

（1） 松田、池田：トランザクション処理システムにおけるジャーナル処理の性能評価、電子情報通信学会論文誌（採録済み）,(2000)

（2） 松田、鶴保：待行列モデルによる大容量記憶装置ＭＳＳの性能解析、情報処理学会論文誌, Vol.25, No.1, pp.125-132,(1984)

国際会議

（1） S.Tsuruho, K.Matsuda, H.Itoh: Communication Control System for Distributed Data Processing Services, Proceeding of the COMPCON'80 FALL, pp.406-411,(1980)

（2） S.Tsuruho, K.Matsuda, M.Ohminami, Y.Itoh: Mass Storage Systems Performance Analysis Using a Queuing Model, Proceedings of the 3rd UJCC, pp320-324,(1978)

電気通信研究所論文誌

（1） 松田、鈴木、池田、池浜：リアルタイムシステムの電文制御方式、通研実報、Vol.26,No.8,pp.2221－2240(1977)

（2） K.Matsuda,E.Moriya,R.Ikeda,H.Ikehama：Message Control in a Real Time System, Review of ECL,Vol.26, No.1－2,pp.24－32(1978)

（3） 松田、大久保、大南、中島：オペレーティングシステムの特性評価、通研実報、Vol. 28,No.12,pp2697－2709(1979)

（4） K.Matsuda,T.Ohkubo,et al :Operating System Performance Evaluation, Review of ECL, Vol.28,No.3-4,pp185-194(1980)

（5） 鶴保、藤、松田、坂田、佐藤：超大容量記憶装置制御方式、通研実報、

Vol.30,No.2, pp409－423(1981)

（6） K.Matsuda,I.Kogiku,S.Sato :Mass Memory System Performance Measurement, Review of ECL,Vol.29,No.5-6,pp451－463(1981)

（7） 村田、森、拝原、松田、森元：１０４－０３システムの分散処理方式、通研実報、 Vol.31,No.8,pp1437－1448,(1982)

研究会等

（1） 高村、松田：リアルタイムシステムにおけるタスク構成の解析的評価、通信学会電子計算機研究会（ＥＣ７９－２８），1979

（2） 松田：ＯＳインタフェース標準化の動向、情報処理学会ＯＳ研究会（86-OS-31）,1986

（3） 松田：ＣＴＲＯＮの標準化の考え方、京都大学ＯＳセミナ講演、1988

（4） 鶴保、松田、大久保：１０４ＯＳの制御方式と動作特性、電気通信研究所研究討論会,1976

（5） 高村、松田: 大規模実時間処理システムの制御方式、DIPS シンポジウム、1979

（6） 松田、大南：ＭＳＳの制御手法と動作特性、電気通信研究所研究討論会、1978

（7） 藤、松田他: 分散処理システム構成法、電気通信研究所研究討論会、1981
電子計算機研究会（EC79－28），1979

学会講演

（1） 松田、伊東：大型システムにおけるファイルアクセス法について、47 年度信学会全国大会、1972

(2) 松田、伊東：システム製造用ＯＳの開発について、47 年度信学会全国大会、1972

（3） 松田、池田：通信制御処理装置を用いた端末アクセス法、情処学会全国大会、 1976.

（4） 松田、大南、中島: 超大容量記憶装置の特性に関する一考察、情処学会全国大会、1977.

（5） 徳山、松田: データ通信システムにおける待ち行列の問題、53 年度信学会大会、1978.

（6） 鶴保、松田、大南: タスクを考慮した待ち行列モデルに関する一考察、53 年度信学会大会、1978.

（7） 松田、大南他：系内呼数制限を有する待ち行列モデルの計算機システムへの応用、情処学会全国大会、1978.

（8） 松田、佐藤、村田：ＭＳＳステージング処理時間の最適化について、情処学会 23 回全国大会、1981.

（9） 松田、杉村：ＳＲＲ待ち時間改善のための一手法、情処学会 24 回全国大会、1982.