SE 19－1
(1977. 5. 27)

マルチプロセッサシステムの内部動作状態の評価検討

大久保 利一　中島 壽生　吉沢 康文
(電電公社 横須賀通研)　(日立シ研)

1. はじめに

メモリを共有した密結合マルチプロセッサ(MP)システムでは、同一オペレーティングシステム(OS)のもとで各CPUが走行するが、入出力制御処理に関し、CPU間で機能を分担する制御方式をとる場合がある。

報告者等はMPシステムでの入出力制御方式を検討するにあたり、各種の機能分担モデルについてシミュレーションをおこない、それぞれのモデルでのCPUやタスクの動作状態を把握した。

本稿ではシミュレーション結果を基に、主として応答時間特性に関して、各制御方式を評価した結果について述べる。

なお本稿におけるMPシステムではCPU2台を前提とする。

2. 入出力制御方式とシミュレーション・モデル

MPシステムにおける入出力処理機能の機能分担に注目したとき、代表的な方式として表1に示す4方式が考えられ本稿での検討の対象とした。なお、一般の業務プログラム(Application Program:AP)は各CPUで走行可能としている。

表1 MPシステムにおける入出力制御方式

方式	概要	特徴	備考
方式1 (平等方式)	各CPUから全入出力装置に対して入出力要求を発行でき、又いずれの入出力装置からの割込みも受付けることができる。	入出力処理時のCPU間排他制御(ロック制御)が複雑となり ロック処理の為のオーバーヘッドが多くなる。	図1-1
方式2 (入出力装置分担方式)	システム内の入出力装置を各CPUに分担させ、CPUに関じて入出力要求の発行・割込み処理をおこなう。 入出力要求が他CPU分担の装置であれば CPU間通信により処理依頼をおこなう。	入出力処理の為のロック制御が分担装置の切分けじて処理のみとなり 入出力処理に伴なう大部分のロック処理が不要となる。 CPU間通信割込みが多くなる。	図1-2
方式3 (入出力専用CPU方式)	入出力処理(入出力要求処理と割込み処理)を特定CPUのみに限定する。(入出力処理を行なうCPUをMPU,他をJPUと呼ぶ。)	入出力に伴なうロック制御が不要となる。 入出力専用CPUへのCPU間通信割込みが多くなる。	図1-3
方式4 (入出力割込専用方式)	入出力要求処理は各CPUでおこない、割込み処理のみを特定CPUでおこなう。(入出力処理を行なうCPUをMPU,他をJPUと呼ぶ。)	割込み処理内部でのロック制御は不要となるが、入出力要求時のロック処理は必要である。	図1-4

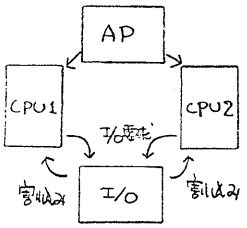

図1-1 方式1(平等方式)

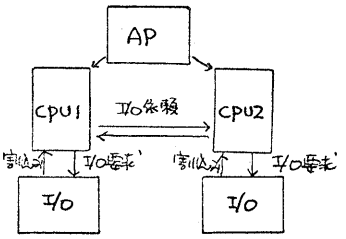

図1-2 方式2(入出力装置分担方式)

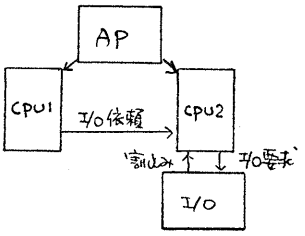

図1-3 方式3(入出力専用CPU方式)

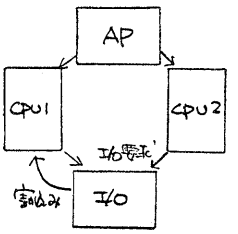

図1-4 方式4
(入出力割込専用方式)

2-1 シミュレーション・モデル

前述の各方式を汎用シミュレータを用いてモデル化した。このうち ソフトウェア・モデルは バンキング業務等のリアルタイム処理を例に 図2に示す様なモデルとした。図2で定義される各要素は次のとおりである。

(1) タスク

(i) SMT : CPU処理のみを行なう。トランザクションはSMTを得ると、1回のCPU処理を得てSMTを放棄する。システムに1ケである。

(ii) SPT : CPU処理と入出力処理を行なう。入出力同期の為、トランザクションはSPTを保留したままCPUを放棄し待合せを行なう。システムに複数個存在する。

タスクのプライオリティは SMTが高く、各SPT間は等しい。
タスクは トランザクションの処理依頼をうけると CPU待キューに接続される。 タスク・ディスパッチャはアイドルCPUにタスクの処理をおこなわせる。 タスク処理のためのCPUは約50%程度を割込み禁止状態で走行し、残りを割込み可能状態で走行する。

(2) 入出力処理

SPTのCPU処理の終りに入出力要求処理が入る。入出力要求処理は割込み禁止状態で走行する。 また、入出力要求処理では チャネル対応にCPU間排他制御をおこなっている(ロック処理)。

入出力要求デバイス対応にキューイングし、該デバイスがアイドル時には入出力実行命令が発行される。デバイスがビジーの時には 該要求をキューイングしたままリターンする。 入出力割込み処理は デバイス終了時の割込みのための処理であり、割込みを生起したCPUでプリエンプティブにスケジュールされる。 但し、割込み禁止状態であれば待合せる。 割込みが受付けられると 割込み解析をおこない、当該チャネルに対するロック処理を行なった後、入出力後処理をおこなう。割込み処理の終りにデバイスキューをチェックし 待合せがある場合該要求の入出力実行命令を発行する。

入出力装置分担方式、入出力専用CPU方式では 該要求の入出力実行命令を発行可能かどうかの切分けを行ない、他CPU分担装置の要求であればCPU間通信割込みにより 入出力実行命令の発行依頼をおこなう。

入出力割込みは 平等方式、入出力装置分担方式では両CPUで生起し、入出力専用CPU方式、入出力割込専用方式では 特定CPU(MPU)のみで生起する。

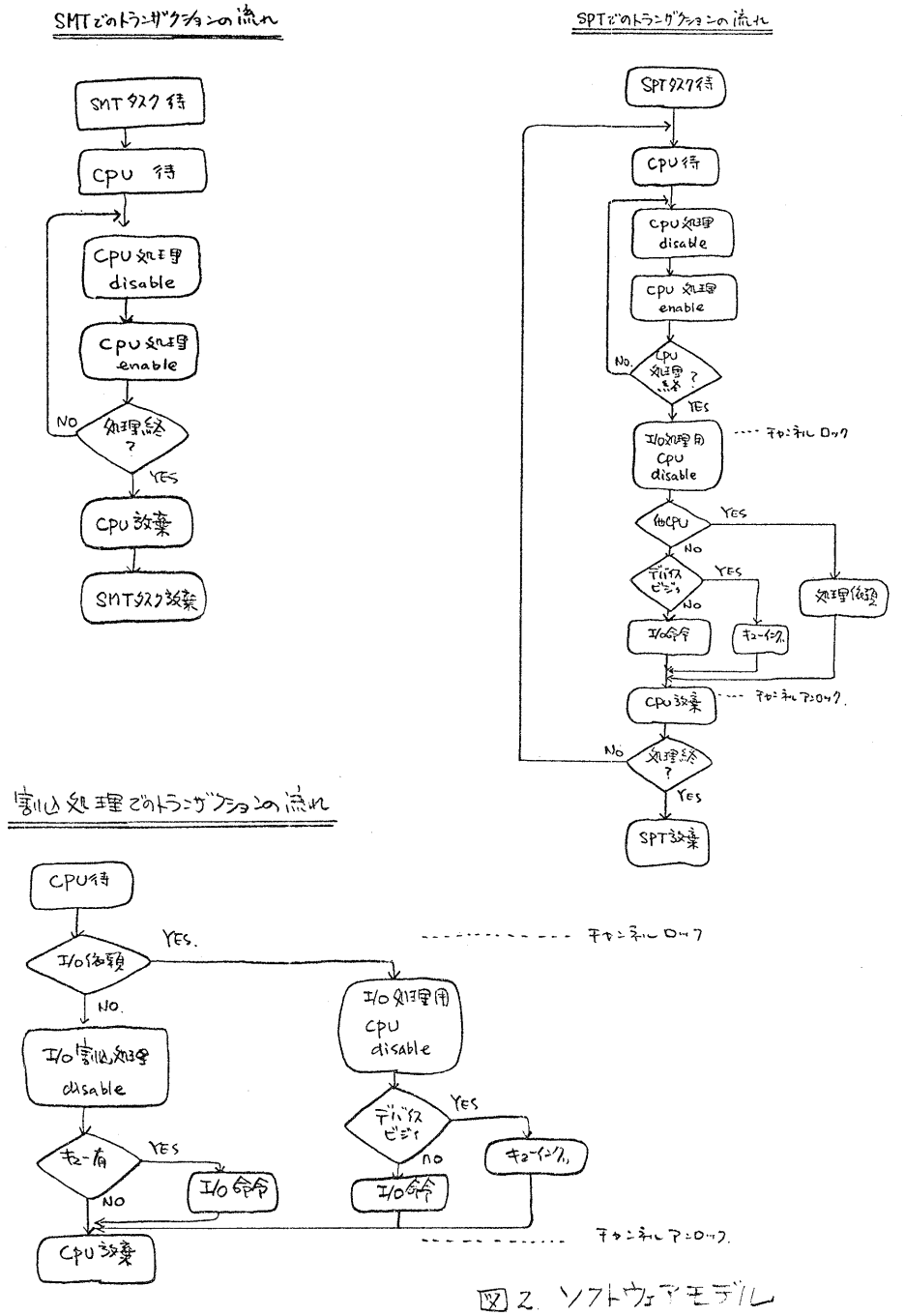

図2. ソフトウェアモデル

2-2 トランザクション処理の待行列モデル

トランザクション処理を タスク, CPUおよび I/O装置に注目してモデル化すると 図3に示す様な待行列ネットワークで表わされる。

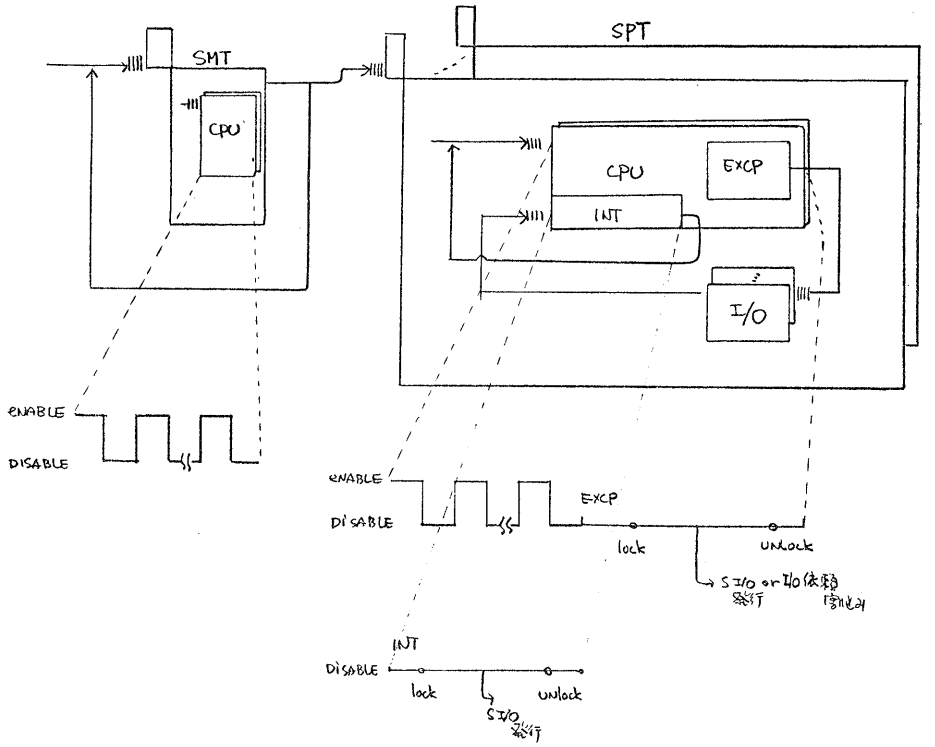

図3. タスク処理モデル

3 応答時間特性の分析

3-1 タスク動作特性

各方式でのタスク動作特性についてシミュレーション結果を図4-1，図4-2に示す。各MP制御方式により相違がみられる。（方式1と方式2 および 方式3と方式4は，ほぼ同一特性をもつ。）ディスパッチャはCPU待ちキューにキューイングされているタスクを該タスクの属性に従いCPUを割当てる。タスクに特定CPU走行指定がなければ各CPUのタスク負荷は平等になる様にスケジュールされる。各MP制御方式によるタスクのCPU待時間の相違の原因として次の要因が考えられる。

① タスク走行中の割込み処理によるタスク保留時間の伸び

タスク走行中に生起した割込みのための処理は，ディスパッチャにとって，タスクのCPU処理時間が伸びた様にみえる（みかけのタスク保留時間の増大）。各MP制御方式により，割込み処理の走行率が異なり，みかけのタスク保留時間の伸びに相違がある（図4-1参照）

② ロック不可でのスピン処理によるタスク保留時間の伸び

各MP制御方式により，両CPUでの割込み処理，タスク処理の走行率が違い，特定ロック単位へのトラフィックが異なる。これにより，ロック失敗確率が違い，みかけ上のタスク保留時間の伸びに相違がでてくる。

以上の各要因によるタスク保留時間の伸びは，ディスパッチャにとって，該タスク走行CPUの処理能力低下にみえる。シミュレーションでのタスク保留時間の伸びのほとんどは①の要因であり，ここでは①に注目し，タスクのC

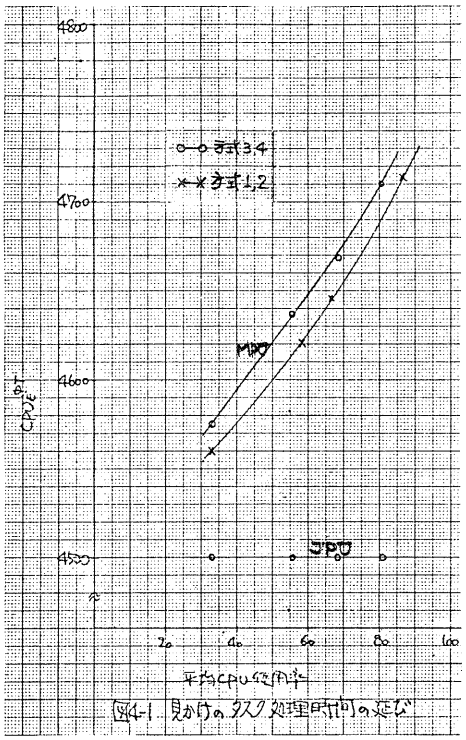

図4-1 見かけのタスク処理時間の延び

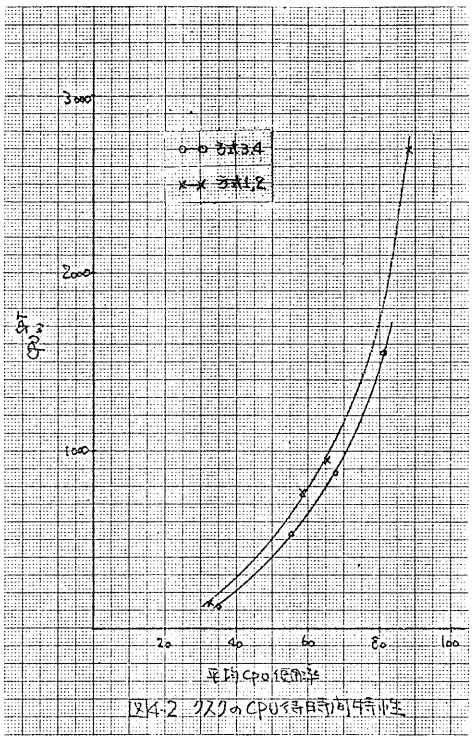

図4-2 タスクのCPU待時間特性

PU待時間について一般的に解析する。

(1) 仮定

①割込みは CPUのタスク状態／アイドル状態には無関係に生起する。

②ディスパッチャからみたCPU状態としては アイドル／タスク状態が存在するが、ディスパッチャは 両CPUに対しタスク負荷が平等になる様に割付ける。

(2) 解析式

各MP制御方式によるタスクのCPU待時間の相違の原因として考えられたタスク保留時間の伸びは、該タスク走行CPUの処理能力低下としてみえる。 このようなタスクの振舞いは サービス率の異なる複数サーバーをもつ待行列モデルに帰着される。

ここでは 次の待行列モデルにより解析する。

- 入線無限
- サーバー数2で各サーバーのサービス率が異なる。
- トラフィックの発生率およびサービス率は指数分布に従う。

検討に先立ち次の諸量を定義する。

λ ：電文発生率
t ：電文処理当りのタスクとして走行するCPU処理時間
a ：電文処理当りの入出力実行回数
t_1 ：割込み処理当りのCPU処理時間
λ_1 ：割込み発生率
α ：タスクとして走行するCPU処理走行率
α_1 ：割込み処理走行率

上記諸量間には　次の様な関係式が得られる。

$$\alpha = \lambda h \quad , \quad \lambda_1 = a\lambda \quad , \quad \alpha_1 = \lambda_1 h_1 \qquad \text{---(1)}$$

以上より、本解析モデルにおける平均待時間（W_q）は次式で得られる。[1]

$$W_q = \frac{1}{2\lambda \cdot D_2}\left\{4C_2^2 + 2C_1^2 + \frac{C_1^2[1+2(C_1^2-1)]}{(C_1^2-2)^2}\right\} - \frac{2}{2\lambda} \qquad \text{---(2)}$$

ここで　$D_2 = C_1^2 + 2C_2^2 + C_1^2/(C_1^2-1)$

但し　$C_2^2 = \prod_{i=1}^{2} m_i$,　$C_1^2 = \sum_{i=1}^{2} m_i$,　$m_i = \frac{\mu_i}{2\lambda}$

μ_i：各サーバーのサービス率

(3) 解析結果

各MP制御方式でのサービス率は表2に示す値となる。各サービス率を算定式に代入することにより、得られたタスクのCPU待時間を図5に示す。

図5より　タスクのCPU待時間はCPU処理能力に相違がある程小さいという結果が得られる。すなわち、方式3と方式4でのタスクのCPU待時間が方式1,2のそれに比べて小さく、シミュレーションの傾向と合致する。このことより、タスクのCPU待時間の相違は　タスク保留時間の伸びによる影響であると判断できる。

表2　各MP制御方式でのサービス率

方　式	μ_1	μ_2
方式1,2	$\frac{a+\alpha}{h(1+\alpha_1)}$	$\frac{a+\alpha}{h(1+\alpha_1)}$
方式3,4	$\frac{a+\alpha}{h(1+2\alpha_1)}$	$\frac{a+\alpha}{h}$

α：入出力同期待ち以外のタスク待ち回数

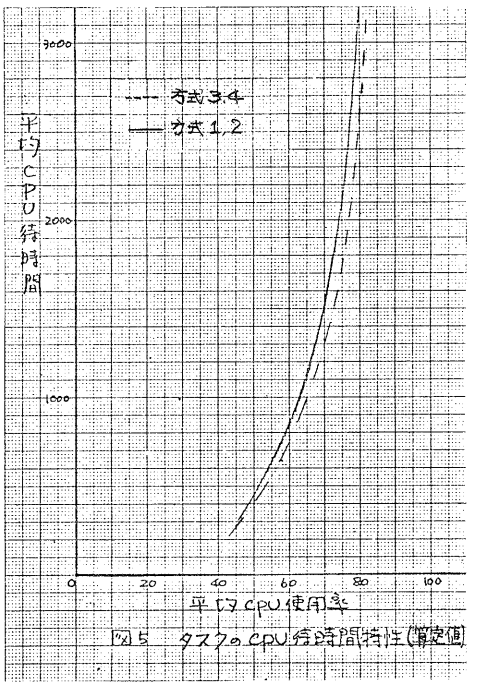

図5　タスクのCPU待時間特性（算定値）

3.2 応答時間特性[2][3]

各MP制御方式により、その応答時間は次の要因により変動し　各要因は常にシリアルに生起するので、応答時間の変動は各要因の和の値に左右される。

・タスクのCPU待時間
・割込み待時間

タスクのCPU待時間については　3.1節で述べた様に　特定CPUで割込み処理を走行させる方式（方式3と方式4）が高負荷時に優位であることが得られた。

ところが、方式2,3および4では　図6のシミュレーション結果に示す様に　割込み待時間が方式1のそれより大きな値となる。[注]

上述の2つの要因を考慮した応答時間特性に関するシミュレーション結果を図7に示す。　図7ではCPU使用率が約80％より高い領域で方式3,4が有利である。

[注]　図6は方式4に関するシミュレーション結果であるが、方式2,方式3は割込み処理のサーバーが単一となり、方式4とほぼ同一の特性となる。

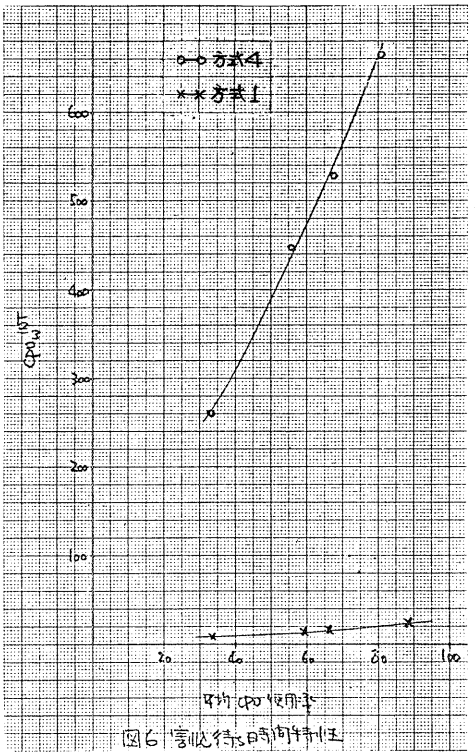

図6 割込待ち時間特性

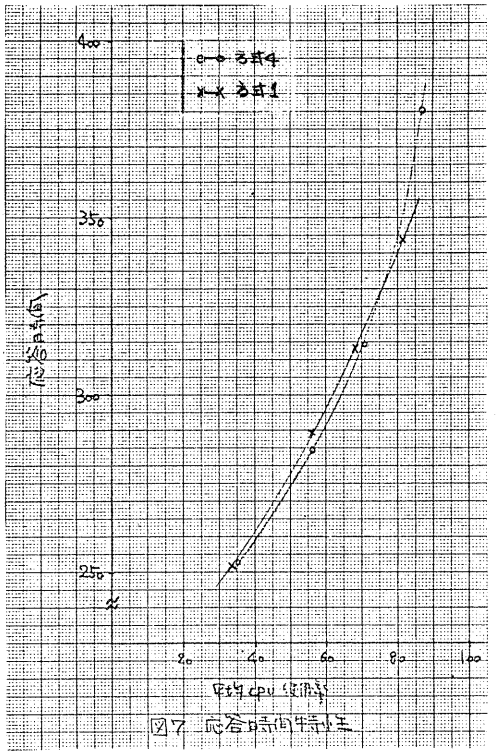

図7 応答時間特性

4 CPU負荷特性の分析

本章では、まずCPU負荷特性の分析結果を述べ、ついでMP方式におけるオーバーヘッド要因であるロック制御の分析結果について述べる。

4_1 CPU負荷特性

CPU負荷を決定する処理には、ディスパッチャによりスケジュールされるタスク処理とそうでない非タスク処理（入出力割込み処理、CPU間通信割込み処理等）がある。タスク走行中に生起した割込みのための処理はディスパッチャにとってタスクのCPU処理時間が伸びた様にみえる。ところがアイドル中に生起した割込みのための処理は、ディスパッチャには意識されず単にCPU負荷を増すのみである。したがって、このアイドル中の割込みのための処理がCPU負荷の偏りの原因であると考えられる。

シミュレーション結果では、図8_1、図8_2に示す様に方式3、方式4において偏りがあり、方式1、方式2ではその偏りはほとんどみられない。

方式3、方式4におけるCPU負荷はほぼ同じ様な値を示しているが、これはCPU間通信割込み処理のステップ数を小さな値でモデル化しているためで、該ステップ数が大きくなる様なシステムでは方式3での負荷の偏りはさらに大きくなるものと考えられる。

方式1、方式2でCPU負荷の偏りが少ないのは次の理由による。

- 方式1では割込み先CPUのサーチ順に関して、PR#1→PR#2、PR#2→PR#1の2ケースをそれぞれ1/2づつの確率でモデル化している。
- 方式2では入出力負荷を均等に分割しているので、割込み処理の走行率は両CPUで等しい。

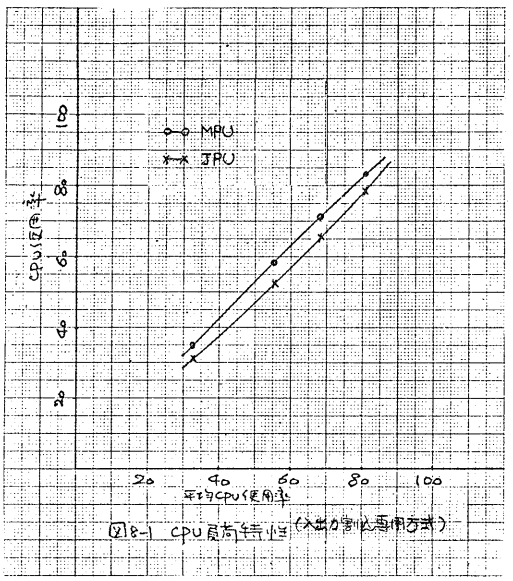

図8-1 CPU負荷特性 (入出力割込専用方式)

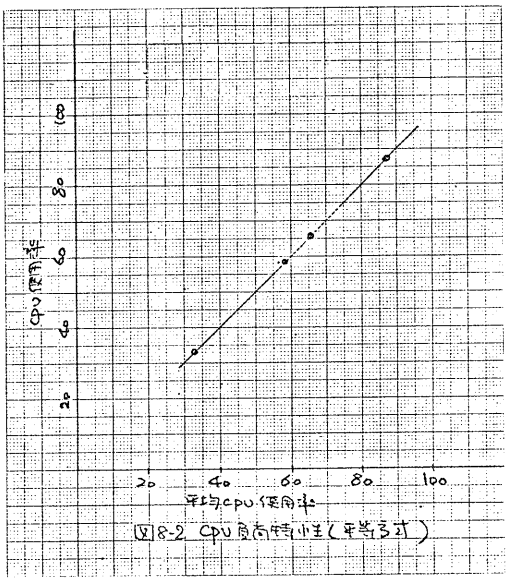

図8-2 CPU負荷特性(平等な式)

次に，CPU負荷特性について一般的に解析する。
本章での解析に使用する記号は　3章で定義した記号を用いる。

(1) CPU負荷の算定

(i) 方式2

入出力負荷バランスを考慮した構成をとると，両CPUに生ずる割込み回数は等しく，両CPUの使用率（ρ_1, ρ_2）は　次式で得られる。

$$\rho_1 = \rho_2 = \alpha + \alpha_1 \qquad \text{---(3)}$$

(ii) 方式3．方式4

タスク処理が両CPUで平等に走行するとき，アイドル中およびタスク走行中の割込み処理走行率をそれぞれ f, g とすると，

$$f = (1-\alpha)\cdot 2\alpha_1, \qquad g = \alpha \cdot 2\alpha_1 \qquad \text{---(4)}$$

となる。3-1節の仮定㊁より，MPU, JPUの使用率（ρ_M, ρ_J）は次式で得られる。

$$\rho_M = \alpha + f + g - (\alpha\alpha_1 - \alpha\alpha_1^2 + \alpha\alpha_1^3 - \cdots)$$

$$= \alpha + 2\alpha_1 - \frac{\alpha\alpha_1}{1+\alpha_1}$$

$$\rho_J = \alpha + \frac{\alpha\alpha_1}{1+\alpha_1} \qquad \text{---(5)}$$

(iii) 方式1

ここでは，割込み先CPUのサーチ順をPR#1→PR#2とし，PR#1が割込み禁止状態であればPR#2に割込みが生起するモデルとする。
今，タスク処理が両CPUで平等に走行しているとし，CPU#1の割込み

禁止状態の走行率を A とすると，CPU#1，CPU#2のタスク処理中に走行する割込み処理走行率（g_1，g_2）は，仮定④より，次式で得られる。

$$\left.\begin{aligned} g_1 &= \alpha \cdot 2(1-A)\cdot\alpha_1 \\ g_2 &= \alpha \cdot 2A \cdot \alpha_1 \end{aligned}\right\} \quad \cdots (6)$$

仮定⑤より，$\alpha(1-2A)\cdot\alpha_1$ の負荷がCPU#2で処理される。
以上より，各CPUの使用率（ρ_1，ρ_2）は次式で得られる。

$$\left.\begin{aligned} \rho_1 &= \alpha + 2(1-A)\alpha_1 - \alpha\{(1-2A)\alpha_1 - \alpha_1^2(1-2A)^2 + \cdots \\ &= \alpha + 2(1-A)\alpha_1 - \frac{\alpha\alpha_1(1-2A)}{1+\alpha_1(1-2A)} \\ \rho_2 &= \alpha + 2A\cdot\alpha_1 + \frac{\alpha\alpha_1(1-2A)}{1+\alpha_1(1-2A)} \end{aligned}\right\} \quad \cdots (7)$$

また，A の値は，CPU#1における割込み禁止状態の走行率条件から，次式により得られる。

$$2\alpha_1(1-A) + b\left\{\alpha - \frac{\alpha\alpha_1(1-2A)}{1+\alpha_1(1-2A)}\right\} = A \quad \cdots (8)$$

但し b：タスク処理の割込み禁止状態の走行率

(2) 算定結果

CPU負荷の偏りについての算定結果を図9に示す。

方式4では，割込み処理の比率が高い程，また，平均CPU使用率が約50%で偏りが最大となる。

但し，CPU負荷の偏りの相対値は平均CPU使用率が低い程大きく，高い程小さくなる。これは，高負荷での割込みの生起がほとんどタスク走行中となるためである。

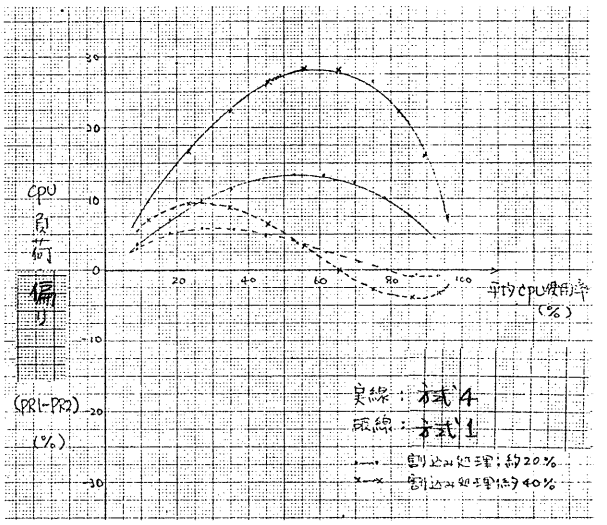

図9 CPU負荷の偏り

方式1では，低負荷でPR#1，高負荷でPR#2の使用率が高くなる。これは，PR#1で割込み禁止に遭遇した割込みは全てPR#2に割込みを生起するモデルとしたためである。本解析モデルでは，PR#1での割込み禁止状態での走行率が約50%を越えると，割込みの発生はPR#2で多くなり，負荷の逆転が生ずる。

しかしながら，PR#1，PR#2とも割込み禁止であれば，割込みが受付けられるまで待合せ，早く割込み可能状態となったCPUに割込むのが一般的であり，この様なシステムでは負荷の逆転はないと考えられる。低負荷においては図9の偏りの傾向を示し，高負荷時には負荷の逆転がない状態で偏り0に近ずくと考えられる。別のシミュレーションにより，このことを確認している。

4-2 ロック制御の分析[4]

ロック処理のオーバーヘッドは，ロック処理のためのステップ数（DS1）とロック不成功時のスピン・ステップ数（DS2）に分けることができる。

一般に，DS1，DS2には図10に示す様な関係があり性能的にはトータル・オーバーヘッドが最小となるロック単位数の設定が望ましい。

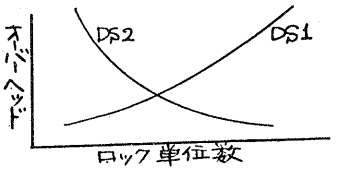

図10 ロック・オーバーヘッド特性

ロック処理オーバーヘッドの削減には次の対策が考えられる。

① ある処理を特定CPUのみで走行することによるロック処理を不要とする方法。
② ロック単位数の統合・細分化。
③ 両CPUからのトラフィックの偏りによるロック成功率の向上。
④ ロック成功時の走行ステップ数の削減。

シミュレーション結果では図11に示す様に方式1で4.8%，方式4で3.7%のオーバーヘッドである。方式4でのオーバーヘッドが少ない主な要因は割込み処理仕様でのみアクセスする制御情でのロック処理が不要であることである。（対策①）

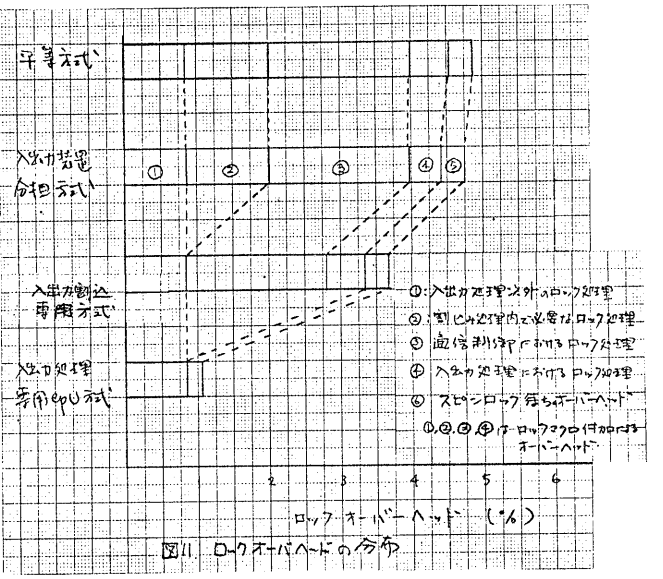

図11 ロックオーバヘッドの分布

方式2ではロック処理の単位を変えることにより，ロック成功時の走行ステップ数の小さなマクロを使用し，オーバーヘッドを削減した。（対策④）

また，全体のオーバーヘッドのうちスピンステップ数（DS2）の割合は小さく対策③の効果は小さい。

5 まとめ

割込み処理を1CPUで専用的に処理するMP制御方式は該処理を2CPUで分担する方式に比べて，

(1) タスクのCPU待時間が少なく，応答時間特性がすぐれている。
(2) ロック処理のオーバーヘッドが小さい。

ことをシミュレーションにより明らかにした。(1)については理論的にも解析しシミュレーションの正当性を確認した。

本稿の検討ではバンキング業務をモデルとしたが上述(1)の一般性については今後さらに検討する予定である。

6 あとがき

CPU2台を前提としたメモリ共有型密結合マルチプロセッサシステムの動作分析をおこなった。

今後は本検討の詳細化および機能分散形MP，ポリプロセッサにおける動作解析についても検討を進めたいと考えている。

また，CPUへの割込みはアドレス・ストリームを乱す要因となり，キャッシュメモリを有するCPUでは平均命令実行時間に与える影響についての考察が必要であり，この点でも割込みを1CPUで処理する方式は有利であることを

確認しているが その詳細は別途報告したい。

本検討に当り、終始御指導頂いた高木制御プログラム研究室室長および関係各位に深謝します。

〈参考文献〉

[1] 本間鶴十代「待ち行列の理論」 理工学社
[2] 坪原, 大保, 中島 他「実時間型処理システムのシミュレーションによる評価例」 SE15
[3] 坪原, 大保, 中島「マルチプロセッサシステム処理能力評価の一手法について」 52年度 信学会 50周年大会 1299
[4] 森え, 花沢, 長浜「マルチプロセッサのロック制御に関する一考察」 52年度 信学会 50周年大会 1298