PPS-Rにおける負荷制御効果の評価

AN EVALUATION OF WORKLOAD CONTROL EFFECTS ON PPS-R

高橋 直久　　村上 国男
Naohisa TAKAHASHI　　Kunio MURAKAMI
日本電信電話公社 武蔵野電気通信研究所
Musashino Electrical Communication Laboratory , NTT

1. まえがき

ポリプロセッサ・システム（PPS）は，極めて密に結合され，機能的に専用化された，小型多数のプロセッサにより構成される計算機複合体である。我々は，このPPSの機能分散の特徴がシステムの動作特性にどのような影響を及ぼすかを明確にするため，専用のソフトウェア・シミュレータ（SAP）を用いてプロセッサの処理動作特性およびプロセッサ間交信特性の解析を行なった。[1,2,11] この結果，数種の典型的な入力ジョブ・モデルに対するPPSの詳細な処理特性が明らかにされ，PPSでは，各プロセッサ・クラスの負荷分布は，ユーザ・プログラムの種類によって大巾に変化し，ジョブの特性によって負荷不平衡が生ずる可能性があることが指摘された。

このようなプロセッサ間の負荷不平衡によるシステム性能の劣化を解消する為に，PPSにおいては，実行時に測定される統計情報に基づき適切な制御手法を動作させ負荷を制御する適応型の負荷管理方式にすることを提案した。[3] この方式を実現し，効果的な負荷制御を行なうためには，

i）負荷状態とシステム性能の関係，
ii）特に，負荷制御上の各種パラメータのシステム性能の変動に対する影響の度合，
iii）各種制御手法の特性，
iv）特に，制御効果，適用領域，

を，明らかにしておく必要がある。

本稿では，上記各項目に関する検討の第一段階として，待ち行列理論を援用し，PPSの負荷状態とシステム性能の解析，プロセッサの機能変身による負荷制御の効果の評価を行なったので，その結果を報告する。

以下の各章では，先ず2.で，PPSの性能，処理特性を待ち行列モデルを用いて解析する。即ち，前述のシミュレーションの結果により得た知見をもとに，処理の流れに着目したPPSの待ち行列モデルをClosed Queuing Networkの手法で設定し，PPSの各種の特性量の評価式を定式化し，システム負荷状態とシステム性能の関係を解析する。

次に，3.では，プロセッサ間の負荷不平衡時における，プロセッサ機能の動的変身による負荷制御の効果を，上記の待ち行列モデルを用いて解析する。マルチ・プログラミング・システムにおける負荷の制御手法には，スワッピング[4]，スケジューリング・パラメータの制御[5]，多重度の制御[6]，など種々の手法があるが，プロセッサ機能の動的変身は，軽負荷のプロセッサの専用化機能を担うマイクロプログラムを動的に入替え，該プロセッサが本来の機能を実行すると共に過負荷プロセッサの負荷を処理するという，PPSが備えた負荷平滑化の手法である[7]。この章では，この機能変身を行なった場合の待ち行列モデルを設定し，機能変身時の各種特性量の評価式を定式化し，2.の解析結果を用いて，機能変身による負荷制御の効果を評価する。

2. PPS-Rの負荷状態とシステム性能の解析

Queuing Network については，すでに指数サービス closed Queuing Network の一般的な解析が Gordon & Newell によってなされており，[8] ヌ，その計算機による数値解法および セントラル・サーバ・モデルへの応用が Buzen によって行なわれている。[9]

計算機による解法プログラムが提案される事でも分かるように，Gordon & Newell らの解は，待ち行列理論的には極めて興味ある重要な結果であるが，現実のシステム特性を理解するにはあまりに一般的過ぎる。

ここでは，専用化機能の構成と処理の流れに注目し，これを抽象化して得られる簡明な，Queuing Network Model を用い，上記の先人達の結果を援用し，PPS の各種の特性量の評価式を定式化し，システム・パラメータによる性能・処理特性の解析を行なう。

表1. プロセッサ機能の分散

	プロセッサクラス	基本機能	
		専用機能	共通機能
USER PU	JPU	利用者プログラムの実行	・割込解析
EXEC PU	LPU	通信回線・端末制御	・プロセッサ間交信処理
	FPU	ファイル・アクセス処理	・イベント同期処理
	RPU	ロールイン・アウト処理	・論理空間の管理
	SPU	ジョブ・スケジューリング	・専用メモリの管理
	MPU	システム構成管理	・論理資源の管理

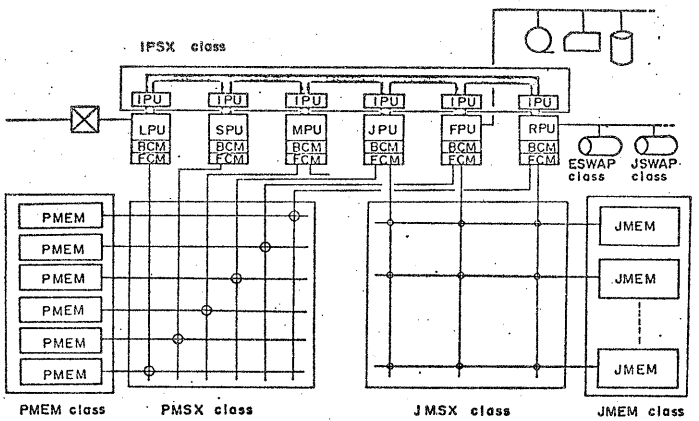


図1. PPSの論理構成

2．1．モデル化

PPSの実験システム（PPS-R）について，その論理構成，OS構成，処理の流れおよびシミュレーション結果などから，重要なシステム・パラメータを抽出し，モデル化を行なう。

2．1．1．PPSの特徴[10]

（1）PPSの構成

PPSの論理構成は，図1.に示す通りである。従来ひとつのプロセッサに集中化されていたシステム機能は，表1.に示す如く，6個のプロセッサ・クラスに分散して実現され，各クラスのプロセッサは，ハードウェア，ファームウェアおよびソフトウェアの各レベルで機能的に専用化されている。

（2）インタラクション処理の流れ

待ち行列理論によって会話形システムの解析を行なう場合，処理単位の“呼”として扱われる対象は，遠隔端末からの入力データ（ヌはコマンド）の投入であり，これをインタラクションと呼ぶ。

PPSにおけるインタラクション処理の流れでは，次の2点が特徴的である。

a）従来の均質型マルチ・プロセッサ・システムでの処理の流れにおけるモジュール間のSVCのいくつかが，PPSにおいては，プロセッサ間通信となっている点であり，これは，PPSが異質型（機能分散型）マルチプロセッサ・システムである点に由来する。

b）JPUにおける処理がほぼ中心（プログラムにおけるmain procedure）となり，他のプロセッサの処理は，JPUからのトリガ（External Processor Call）によって(main procedureに対するsub procedureのように）起動される。これは，JPU以外のEXECプロセッサ・クラスが従来のSVC処理ルチンの集合で構成され，JPU上の利用者プログラムや応用プログラムの要求に従って処理を代行する単能プロセッサであると見なし得る為である。

2．1．2．SIMPLE TSS モデル

PPSの処理の流れに着目し設定した待ち行列モデル（SIMPLE TSSモデルと呼ぶ）を，図2.に示す。

端末から投入された呼は，まずLPUで入力処理を受け，必要ならば，その時点で端末へ出力される。受理可能な入力であれば，LPUからJPUへ引き渡され実行される。実行時の遷移形式は，JPUからEXECプロセッサに対するCallと，このReturnのみである。但し，LPUへ遷移した呼は，ある一定の確率で端末へ出力される。

プロセッサ間の遷移確率行列を図3.に示す。

2．1．3．仮定

PPS動作のシミュレーション結果[1,2,11]等から得た知見をもとに，解析的に求解可能と思われるモデルとする為に，次のような前提，仮定を置いた。

（1）呼の一定性

システムへ投入された呼は，指定された出口から，システム外部へ出力されるまで，消滅も他の呼の生成も行なわないとする。

（2）呼の投入口と出力口

呼は，LPUで生成され，LPUで出力される。

（3）系のclosed性

LPUから呼が出力されると，外部系からは直ちに別入力呼がLPUに投入され，システム中の全呼数は常に一定に保たれる。

（4）遷移形式

プロセッサ間の遷移形式は，Function Call & return 形式が基本であるとする。

（5）外部系の扱い

入出力などによる保留時間は，対応するプロセッサの保留時間と見なすものとする。

（6）IPSX

プロセッサ間通信を行なう為のIPSXについては，別途詳細に解析するので，ここでは性能に影響しない（即ち，伝送処理時間は無視できる）ものとする。

（7）呼の性質

各CPUへの呼の到着間隔は，全てポアソン分布である。又，各CPUでの呼の処理時間は指数分布である。

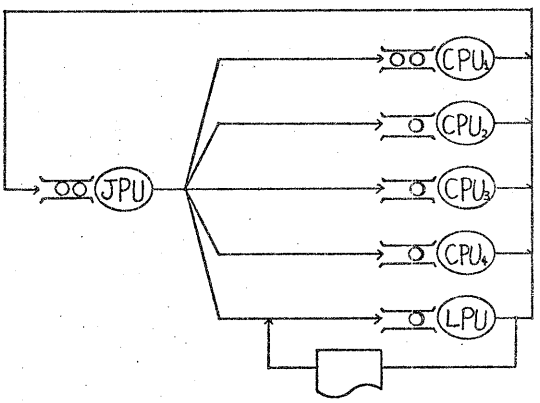


図2．SIMPLE TSS モデル

	JPU	LPU	CPU1	CPU2	CPU3	CPU4
JPU	0	P_{JL}	P_{J1}	P_{J2}	P_{J3}	P_{J4}
LPU	$1-P_{LL}$	P_{LL}				
CPU1	1		0		0	
CPU2	1			0		
CPU3	1		0		0	
CPU4	1					0

図3．遷移確率行列

2．2．定式化

はじめに，以後の解析で用いる各種記号の定義を行なう。次に，系内にひとつの呼，即ち，システムが唯ひとつのジョブ（タスク）を処理している時の特性値を求める。これは，複数個のジョブが存在している時の動特性を評価する基準となるものである。

最後に，系内に複数個の呼が存在する状況でのシステムの動作特性を解析する。

2．2．1．記号の定義

以後の解析に用いる記号を定義する。

P_{JL} ： JPUの処理終了後，LPUに呼が投入される確率．

P_{Ji} ： JPUの処理終了後，CPU_iに呼が投入される確率（$i=1\sim4$，以下同じ）．

P_{LL} ： LPUの処理終了後，端末系に呼が投入される確率．

μ_j ： プロセッサjでの平均サービス率（$j=J,L,1,2,3,4$，以下同じ）．

t_j ： プロセッサjの平均命令実行時間

d_j ： プロセッサ j での平均自由走行ステップ数.
u ： システム内インタラクション総数
n_j ： プロセッサ j 内の呼数
D_j ： ひとつのインタラクションが，プロセッサ j に与える負荷量
D ： ひとつのインタラクションが，システムに与える総負荷量
V_j ： ひとつのインタラクションが，プロセッサ j に与える負荷の割合
$G(u)$ ： 正規化定数
ρ_j ： プロセッサ j の使用率
Q_j ： プロセッサ j の系の長さ
H ： システムの処理量
R_S ： システムの応答時間

2．2．2．シングル・ジョブ処理特性の解析

各インタラクションが，システムのプロセッサ系に対して，どのような負荷を与えるか，その分布を求める。

ひとつのインタラクションが，プロセッサ j に対して与える負荷量を次式で定義する。

D_j ＝｛プロセッサ j での平均処理回数｝
×｛プロセッサ j での1回の処理当りの平均サービス時間｝ ------ (2-1)

ひとつのインタラクションがシステムに与える総負荷量 D は，次のように求められる。

$$D = D_J + D_L + \sum_{i=1}^{4} D_i = \frac{1-P_{JL}}{P_{JL}\cdot P_{LJ}\cdot \mu_J} + \frac{1}{P_{LL}\cdot \mu_L} + \sum_{i=1}^{4} \frac{1-P_{Li}}{P_{LL}}\cdot\frac{P_{Ji}}{P_{JL}}\cdot\frac{1}{\mu_i} \quad ---(2-2)$$

ここで，

$$\frac{1}{\mu_j} = d_j \times t_j \quad (j = J, L, 1, 2, 3, 4) \quad ---(2-3)$$

である。

よって，負荷分布ベクトル V は，

$$V = \left(\frac{D_J}{D}, \frac{D_L}{D}, \frac{D_1}{D}, \frac{D_2}{D}, \frac{D_3}{D}, \frac{D_4}{D}\right)$$

と評価できる。

2．2．3．マルチ・ジョブ処理特性の解析

図2．に示したような6クラス（各クラス1台）のプロセッサにより構成される *closed Queuing Network* モデルで，平衡状態においてプロセッサに少なくとも k 個の要求が存在する確率は，Gordon & Newell の結果[8]を援用し，次のように求められる。

$$\left.\begin{aligned} P(n_J \geq k) &= \frac{G(u-k)}{G(u)} \\ P(n_L \geq k) &= \left(\frac{P_{JL}}{1-P_{LL}}\cdot\frac{\mu_J}{\mu_L}\right)^k \cdot \frac{G(u-k)}{G(u)} \\ P(n_i \geq k) &= \left(P_{Ji}\cdot\frac{\mu_J}{\mu_i}\right)^k \cdot \frac{G(u-k)}{G(u)} \end{aligned}\right\} \cdot (2-4)$$

但し，$G(u) = \sum_{n\in S}\left(\frac{P_{JL}}{1-P_{LL}}\cdot\frac{\mu_J}{\mu_L}\right)^{n_L-k}\cdot\prod_{i=1}^{4}\left(P_{Ji}\cdot\frac{\mu_J}{\mu_i}\right)^{n_i}$ --- (2-5)

$S = \{(n_J, n_L, n_1, \cdots, n_4) \mid \sum_c n_i = u-k,\ n_i \geq 0,\ i \in c\}$
$C = \{J, L, 1, \cdots 4\}$

次に，これらの確率をもとに，複数ジョブ環境下でPPSが示すシステムの諸特性を評価する。

（1）プロセッサの使用率

プロセッサの使用率は，プロセッサの系の長さが1以上である確率に等しいので，

$$\left.\begin{aligned} \rho_J &= P(n_J \geq 1) = \frac{G(u-1)}{G(u)} \\ \rho_L &= P(n_L \geq 1) = \frac{P_{JL}}{1-P_{LL}}\cdot\frac{\mu_J}{\mu_L}\cdot\rho_J \\ \rho_i &= P(n_i \geq 1) = P_{Ji}\cdot\frac{\mu_J}{\mu_i}\cdot\rho_J \quad (i=1\sim4) \end{aligned}\right\} \quad ---(2-6)$$

即ち，中心にあるJPUの使用率を与えれば，他CPUの使用率は決定される。

（2）系の長さ

各プロセッサの系の長さは，前述の確率，$P(n_j \geq k)$，を用いて，次の様に評価される。

$$\left.\begin{aligned} Q_J &= \sum_{k=0}^{u} k\cdot P(n_J = k) = \sum_{k=1}^{u}\frac{G(u-k)}{G(u)} \\ Q_L &= \sum_{k=0}^{u} k\cdot P(n_L = k) = \sum_{k=1}^{u}\left(\frac{P_{JL}}{1-P_{LL}}\cdot\frac{\mu_J}{\mu_L}\right)^k\cdot\frac{G(u-k)}{G(u)} \\ Q_i &= \sum_{k=0}^{u} k\cdot P(n_i = k) = \sum_{k=1}^{u}\left(P_{Ji}\cdot\frac{\mu_J}{\mu_i}\right)^k\cdot\frac{G(u-k)}{G(u)} \end{aligned}\right\} (2-7)$$

（3）処理量

PPSの処理量を評価すれば，

$$H = \rho_L\cdot\mu_L\cdot P_{LL} = \frac{P_{LL}}{1-P_{LL}}\cdot P_{JL}\cdot\mu_J\cdot\rho_J \quad ----(2-8)$$

となる。

即ち，使用率と同様に，JPUの使用率を与えれば，これに正比例して処理量が定まる。

（4）応答時間

システム出力呼に等価な呼が直ちに入力呼となる，というモデルの *closed* の性質を用いると応答時間は次式で評価される。

$$R_S = \frac{u}{H} = \frac{1-P_{LL}}{P_{LL}}\cdot\frac{u}{P_{JL}\cdot\mu_J\cdot\rho_J} \quad -----(2-9)$$

2. 3. Case Study

2. 3. 1. Case A

理想的な負荷状態，即ち，完全に負荷が平滑化され，すべてのプロセッサの負荷が等しい状態におけるシステムの諸特性を特殊解として求める。

この時の条件は，

$$D_J = D_L = D_i \quad (i=1\sim4) \quad \text{-----(2-10)}$$

即ち，

$$\frac{1-P_{LL}}{P_{JL}\cdot P_{LL}\cdot\mu_J} = \frac{1}{P_{LL}\cdot\mu_L} = \frac{1-P_{LL}}{P_{LL}}\cdot\frac{P_{Li}}{P_{JL}}\cdot\frac{1}{\mu_i} \quad (i=1\sim4) \quad \text{----(2-11)}$$

であり，

$$P_J = P_L = P_i \quad (i=1\sim4) \quad \text{--------(2-12)}$$

が得られる。

次に，この負荷状態におけるシステムの諸特性を求める。

（1）$G_J(u)$の評価

正規化定数$G_J(u)$は，次式で評価される。

$$G_J(u) = \sum_{\underline{n}\in S}\prod_{i=1}^{m_L} 1^{n_i}\cdot 1 = {}_{u+4}C_u = \frac{(u+5)!}{5!\,u!} \quad \text{---(2-13)}$$

（2）使用率の評価

全てのプロセッサについて同じ値であるから，

$$P_J = P_L = P_i = \frac{G(u-1)}{G(u)} = \frac{u}{u+5} \quad \text{------(2-14)}$$

$$\hat{P}_J = \lim_{u\to\infty} P_J = 1 \quad \text{----------(2-15)}$$

と評価できる。

（3）処理量の評価

$$H = \frac{u}{u+5}\cdot\mu_L\cdot P_{LL} \quad \text{----------(2-16)}$$

$$\hat{H} = \lim_{u\to\infty} H = \mu_L\cdot P_{LL} \quad \text{----------(2-17)}$$

（4）応答時間の評価

$$R_S = \frac{u}{H} = \frac{u+5}{\mu_L\cdot P_{LL}} \quad \text{----------(2-18)}$$

となり，uに比例して増大する。

（5）系の長さの評価

負荷および使用率が全プロセッサ共等しいので，各プロセッサの待ち行列系の長さも等しく，

$$Q_J = Q_L = Q_i = \frac{u}{6} \quad \text{-------(2-19)}$$

と評価できる。

2. 3. 2. Case B

SIMPLE TSS モデルの特徴は，JPUで走行するユーザ・プログラムからEXECプロセッサ上の各Sub OSを“サブルチン（又はSVCルチン）”のように呼び出す点を強調して，PPSをモデル化した点にある。

従って，この点に注目し，全EXECプロセッサの負荷が等しい状態におけるシステムの諸特性を特殊解として求める。

この時の条件は，

$$D_L = D_i, \quad (i=1\sim4) \quad \text{--------(2-20)}$$

即ち，次式で表わされる。

$$\frac{1}{P_{LL}\cdot\mu_L} = \frac{(1-P_{LL})\cdot P_{Li}}{P_{LL}\cdot P_{JL}}\cdot\frac{1}{\mu_i}, \quad (i=1\sim4) \quad \text{----(2-21)}$$

（1）$G_J(u)$の評価

（2-21）式より，

$$\rho = \frac{P_{JL}}{1-P_{LL}}\cdot\frac{\mu_J}{\mu_L} = \frac{P_{Li}\cdot\mu_J}{\mu_i}, \quad (i=1\sim4) \quad \text{----(2-22)}$$

とおけば，

$$G_J(u) = \sum_{j=0}^{u} {}_{u-j+4}C_{u-j}\cdot\rho^{u-j} \quad \text{------(2-23)}$$

（2）使用率の評価

$$P_J = \frac{G(u-1)}{G(u)} = 1 - \frac{(u+4)(u+3)(u+2)(u+1)}{P_0} \quad \text{---(2-24)}$$

但し，

$$P_0 = \sum_{j=0}^{u}\frac{(u-j+4)(u-j+3)(u-j+2)(u-j+1)}{\rho^j}$$

$$\hat{P}_J = \lim_{u\to\infty} P_J = \begin{cases} 1 & (\rho<1) \\ \frac{1}{\rho} & (\rho\geq 1) \end{cases} \quad \text{---(2-25)}$$

$$P_L = P_i = \frac{P_{JL}}{1-P_{LL}}\cdot\frac{\mu_J}{\mu_L}\cdot P_J = \rho\cdot P_J \quad (i=1\sim4) \quad \text{----(2-26)}$$

$$\hat{P}_L = \hat{P}_i = \lim_{u\to\infty} P_L = \begin{cases} \rho & (\rho<1) \\ 1 & (\rho\geq 1) \end{cases} \quad \text{---(2-27)}$$

（3）処理量の評価

$H = P_L\cdot\mu_L\cdot P_{LL}$ より，

$$\hat{H} = \lim_{u\to\infty} H = \begin{cases} \rho\cdot\mu_L\cdot P_{LL} & (\rho<1) \\ \mu_L\cdot P_{LL} & (\rho\geq 1) \end{cases} \quad \text{---(2-28)}$$

（4）応答時間の評価

$R_S = u/P_L\cdot\mu_L\cdot P_{LL}$ より，uが十分大きい時のR_S，$\tilde{R}_S$，は，次のように表わされる。

$$\tilde{R}_S = \begin{cases} u/(\rho\cdot\mu_L\cdot P_{LL}) & (\rho<1) \\ u/(\mu_L\cdot P_{LL}) & (\rho\geq 1) \end{cases} \quad \text{----(2-29)}$$

（5）系の長さの評価

$$Q_J = \sum_{\ell=1}^{u}\frac{G(u-\ell)}{G(u)} = \sum_{\ell=1}^{u}\frac{1}{\rho^\ell}\cdot g_\ell \quad \text{------(2-30)}$$

但し，$g_\ell = \frac{1}{P_0}\sum_{j=0}^{u-\ell}\frac{(u-\ell-j+4)(u-\ell-j+3)(u-\ell-j+2)(u-\ell-j+1)}{\rho^j}$

$$Q_L = Q_i = \sum_{\ell=1}^{u} g_\ell, \quad (i=1\sim4) \quad \text{----(2-31)}$$

2．4．数値例

2．4．1．SAPデータによる数値例

表2．に示したパラメータをもとに，応答時間，処理量などを求めてみる。これらのパラメータ値は，SAPから得られた基礎データをもとに設定したものである。FPUおよびRPUに関しては，入出力処理時間を除外してあり，実質のプロセッサにおける処理のみを考慮している。これは，入出力装置の特性の影響が，PPSの動特性に及び，本来のPPS解析を無意味にすることを恐れた為である。

図4．に処理量と応答時間を示す。これは，シミュレーションで与えられた結果と極めて良く一致しており，本モデルの妥当性を示している。

図5．に各プロセッサの使用率を示す。これによれば，完全なJPUネックであり，これを解消するには，JPUの平均命令実行時間を高速化する必要がある。

表2．数値例

	d_i STEP	P_{Ji}	t_{ij} μsec
JPU	650	($P_{JL}=0.5$)	1
LPU	1350	0.05	1
RPU	125	0.2	1
FPU	1000	0.35	1
SPU	200	0.2	1
MPU	200	0.2	1

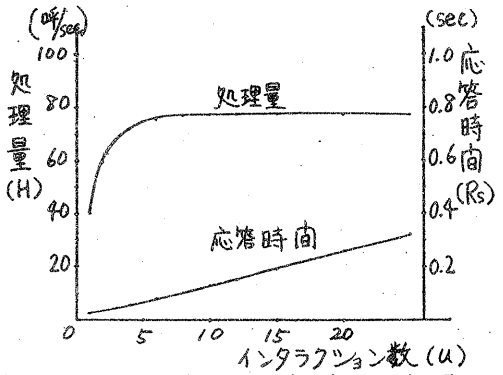


図4．SAPデータの場合の応答時間と処理量

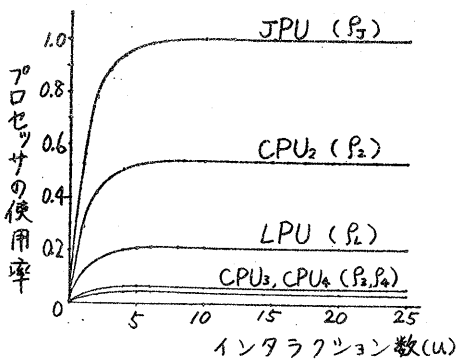


図5．SAPデータの場合の使用率

2．4．2．負荷平衡時のシステム特性

EXECプロセッサに対する負荷が平滑化されている場合のシステム特性を求める。

即ち，表2．のパラメータのうち，平均自由走行ステップ数（d_i）を，ひとつのインタラクション当りの総負荷量（D）がSAPデータの場合と等しいという条件の下で変化させた時のCase Bの特性を求める。

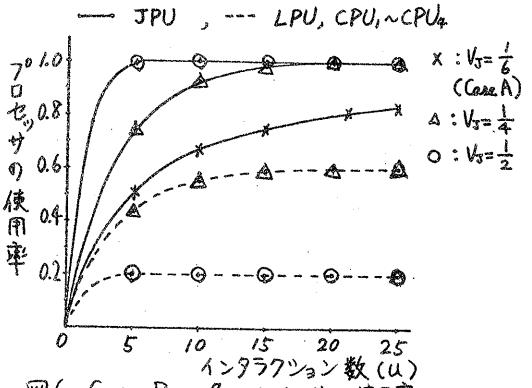


図6．Case Bの各プロセッサの使用率

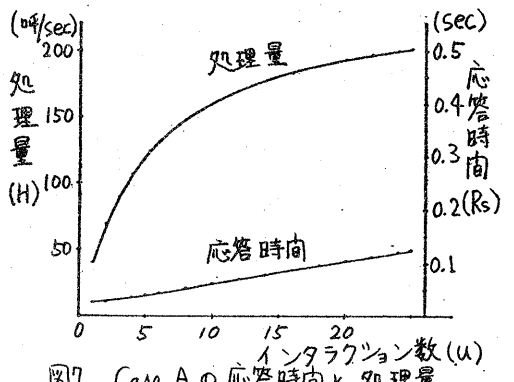


図7．Case Aの応答時間と処理量

図6．に,JPUにかかる負荷の割合（V_J）をパラメータにした時の各プロセッサの使用率の変化を示す。

又，図7．に，$V_J=\frac{1}{6}$，即ち，Case A の場合の処理量と応答時間を示す。

2．4．3．負荷不平衡時のシステム特性

Case B の成立条件をくずし，負荷不平衡にした場合のシステム特性を求める。

即ち，ひとつのインタラクション当りの総負荷量（D）がSAPデータの場合と等しいという条件の下で，各プロセッサにかかる負荷の割合を次の様に変化させた場合について数値例を求める。図8．に各Caseの相互関係を示す。

Case B_1 ： V_Jを固定し，CPU_1にかかる負荷の割合（V_1）を他のEXECプロセッサにかかる負荷の割合よりも徐々に大きくし，CPU_1をボトルネックにしていく。
（$V_2=V_3=V_4=V_L$，$V_J=\frac{1}{6}$）

Case B_2 ： V_Jを固定し，CPU_1をボトルネックにし，CPU_2にかかる負荷の割合(V_2)を徐々に大きくし，CPU_2もボトルネックにしていく。（$V_3=V_4=V_L$，$V_1=\frac{4.5}{6}$，$V_J=\frac{1}{6}$）

図9．に負荷平滑化の度合（Case B_1ではV_1の値，Case B_2ではV_2の値）と応答時間の関係を示す。但し，応答時間は，単一ジョブの時（即ち$U=1$）の応答時間によって正規化している。

Case B_1の特性（実線）を見ると，システム内にボトルネックが生じた際に，システム性能に与える影響は極めて大であり，インタラクション数（U）が多い時には，性能を劣化させる割合（図の曲線の傾きに対応する）が特に大きくなることがわかる。

Case B_2の特性（点線）を見ると，ボトルネック・プロセッサが存在する時には，他のプロセッサ間の負荷の平滑化の度合はシステム性能にほとんど影響を与えないことがわかる。

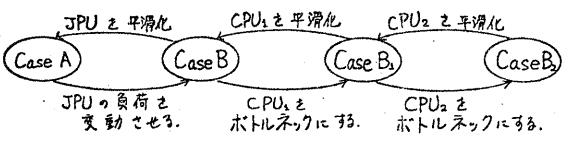


図8．各Caseの相互関係

2．5．考察

前節の結果から，次の事が指摘できる。

（1）既存のOSの設計手法によってPPS用OSを設計した場合には，ほとんどJPUネックにおちいる。

これは，OSのうちの処理プログラム的なもの，例えば，コマンド処理プログラムなどをJPUで走行させるような機械を取っているからであり，PPSでは，これらを，SPUやLPUなど，あるいは，ファイルに関するコマンドであれば，FPUで処理する如き，大胆な思想の変革によって解決する必要があろう。

（2）PPSでは，ボトルネックのプロセッサ・クラスを作り出す事は，システム性能上，極めて危険である。

即ち，ボトルネックのプロセッサが出現すると，PPSの性格上，機能分散化による該プロセッサへの交信の度にdelayが発生し，処理量は増大せず，他プロセッサは遊んでいるにもかかわらず，応答時間が急激に伸びてしまう。

この意味で，PPSは，ボトルネック・プロセッサに対して極めてsensitiveである。

（3）ボトルネックの度合が大きい時ほど，システム性能は，負荷平滑化に対して敏感である。

即ち，ボトルネックの度合が大きい場合には，そのボトルネック・プロセッサに対する負荷の配分比がわずかに変わっても，システム性能は大きく変化する。

逆にいうと，ボトルネックの度合が小さい場合には，オーバヘッドのみが顕著になるの

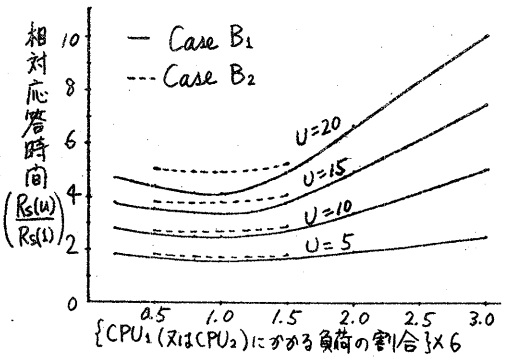


図9．Case B_1，Case B_2 の相対応答時間

で，負荷平滑化を敢えてすべきではない。

（4）システム性能は，インタラクション数が多い時ほど，負荷平滑化に対して敏感である。

即ち，インタラクション数が多い場合には，負荷分布のわずかな変化が，システム性能に大きな影響をもたらす。

従って，インタラクション数が多い場合には，負荷平滑化は，特に重要な問題となる。

3. 機能変身効果の解析

前章の解析により，PPSでは，ボトルネックのプロセッサを作り出すことは，システム性能上極めて危険であるということが明らかになった。このボトルネックを解消するために，PPSでは，軽負荷のプロセッサが，機能変身を行ない，ボトルネック・プロセッサの処理の1部を代替して行なう機構を備えている。[7] 本章では，この場合の代替方式を確立するための基礎データを得る為に，前述のモデルを拡張し，EXECプロセッサ間での機能変身の効果を定量的に把握する。

3.1. モデル化

3.1.1. 機能変身機構

PPSの各プロセッサの機能は，表1.に示した様な，共通機能と専用機能とに分類することができるが，これらの機能には，いずれもマイクロプログラム（μPと略）によりファームウェア化されている部分がある。機能変身を行なう場合には，変身後のプロセッサの専用機能を担うμPを該プロセッサが備える必要がある。

この為の方法は，変身時に，①専用機能を担うμPを入れ替える，②該プロセッサの専用機能を担うμPはそのまま残し，RAMの空き領域及びCore上に変身後の専用機能を担うμPをロードし，変身後はCore上のμPも空間共用[7]により使用する，という2つの方法を組み合わせて実現される。

従って，該プロセッサの命令語を実現するμPを格納する記憶域は，図10.に示す通りである。即ち，命令語のフェッチ，実効アドレスの算出，オペランド・フェッチや共通命令語の実行を行なうμPは，変身前・後にかかわらず，ROMおよびRAMの一部に置かれる。該プロセッサ本来の専用命令語の実行を担うμPは，RAMに置かれ，変身後の専用命令語の実行を担うμPは，RAMおよびCoreに置かれる。

各記憶装置の速度，語巾を表3.に示す。1マイクロ命令は，Core上では5語要す。μPのロードや，制御記憶のアクセス時間の増大が，機能変身時のオーバヘッドの原因となる。

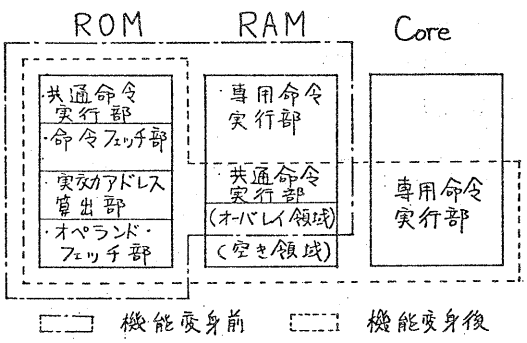


図10. 制御記憶域

表3. PPS-R のメモリ諸元

	速度	語巾
ROM	240 nsec	80 bit
RAM	480 nsec	80 bit
コア	1.2 μsec	16 bit

3.1.2. 機能変身を行なった場合の待ち行列モデル

取扱いを容易にする為，ここではJPUを除い

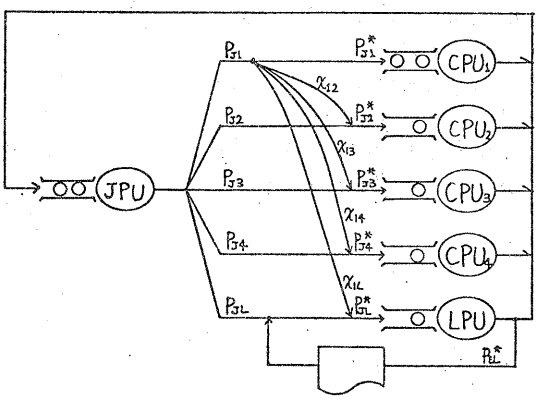


図11. 機能変身を行なった場合の待ち行列モデル

た5台のEXECプロセッサ間での機能変身に限定して議論する。EXECプロセッサ間の機能変身の効果を調べるために，前述のSIMPLE TSSモデルを基に，次のように拡張したモデルを考える。CPU$_1$からCPU$_4$は，モデル上では対称であるので，ここでは，CPU$_1$がボトルネックの場合のみを考える。即ち，CPU$_1$に対する処理要求を他のEXECプロセッサへ分配する。

従って，SIMPLE TSSモデルの処理の流れは，図11.のように変わる。

3．1．3．仮定

本モデルの解析においては，2.1.3.節で設定したSIMPLE TSSモデルに対する仮定の他に，以下の各仮定，近似を行なう。

（1）2種類の負荷を処理するプロセッサのサービス時間が超指数分布となる場合には，平均値の等しい指数分布で近似する。

（2）上記プロセッサの入力呼は，ある時点で別のポアソン分布へ"瞬時に"変化する。

（3）μPを入替えてプロセッサ機能を変える場合，ロードすべきμPの大きさは代替μPの走行ステップ数に比例し，入替えオーバヘッドはロード対象のμPの大きさに比例する。

（4）各プロセッサが他プロセッサの機能を代行した場合，その処理速度は，原プロセッサによる処理速度に比し，一定割合で低下する。

（5）各プロセッサは，ひとつの入力呼の処理途中で，機能の切替えを行なわない。

（6）入力呼の変動に対し，システムを動的に変身させた時，システムは瞬時に定常状態に移行する。

2台のプロセッサの場合の解析結果[12]や，セントラル・サーバ・モデルを使った解析と現実のシステムでの性能測定の結果による考察[13]を見ると，（1）の近似は，機能変身を行なった場合の現実のPPSの性能を把握する為に本モデルの解析結果を用いることの妥当性を損うものではないといえる。

3．2．定式化

3．2．1．記号の定義

機能変身を行なった場合，即ち，CPU$_1$の負荷を他のプロセッサに配分した場合の処理を表わすために，次の諸量を定義する。（以下では，$i=2,3,4,L$とし，$LPU=CPU_L$とする。）

χ_{1i} ： CPU$_1$の処理のうちCPU$_i$へ分配する比率（処理分配率）．

α_{1i} ： CPU$_i$がCPU$_1$の機能を代行した場合の処理速度の低下率（縮退率）．

v_{3i} ： CPU$_i$における，CPU$_3$の処理から，CPU$_i$の処理への切替え時間によるオーバヘッドの割合．

v_{i1} ： CPU$_i$における，CPU$_i$の処理から，CPU$_1$の処理への切替え時間によるオーバヘッドの割合．

P_{JL}^{*} ： JPUの処理終了後，CPU$_i$に呼が投入される確率．

P_{Li}^{*} ： LPUの処理終了後，端末系に呼が出力される確率．

μ_J^{*} ： JPUでのサービス率．

μ_i^{*} ： CPU$_i$でのサービス率．

γ_{jk}^{i} ： CPU$_i$で，CPU$_j$の処理後に，CPU$_k$の処理を行なう割合．（$j,k=1$ or i）．（$\gamma_{ji}^{i}+\gamma_{j1}^{i}+\gamma_{jL}^{i}+\gamma_{ji}^{i}=1$）

μ_{jk}^{i} ： CPU$_i$で，CPU$_j$の処理後に，CPU$_k$の処理を行なう際のサービス率．（$j,k=1$ or i）

R_S^{*} ： 平均応答時間

3．2．2．プロセッサ間の遷移確率

機能変身後の，プロセッサ間の遷移確率は，2.で定義した機能変身前のプロセッサ間遷移確率，および，処理分配率χ_{1i}（$i=2,3,4,L$）により，次式で表わされる。

$$\left.\begin{aligned} P_{JL}^{*} &= P_{JL}+P_{J1}\cdot\chi_{1L} \\ P_{LL}^{*} &= P_{LL}\cdot\frac{P_{JL}}{P_{JL}+P_{J1}\chi_{1L}} \\ P_{Ji}^{*} &= P_{Ji}+P_{J1}\cdot\chi_{1i}\ ,\ (i=2,3,4) \\ P_{J1}^{*} &= P_{J1}\cdot(1-\chi_{1L}-\chi_{12}-\chi_{13}-\chi_{14}) \end{aligned}\right\}\text{---(3-1)}$$

3．2．3．1回処理当りの平均サービス時間

（1）JPUの平均サービス時間

$$\frac{1}{\mu_J^{*}}=\frac{1}{\mu_J}\qquad\text{----------(3-2)}$$

（2）CPU$_1$の平均サービス時間

$$\frac{1}{\mu_1^{*}}=\frac{1}{\mu_1}\qquad\text{----------(3-3)}$$

（3）CPU$_i$（$i=2,3,4$）の平均サービス時間

3.1.3.節で示した2種類のオーバヘッドを考慮すると，CPU_iでの平均サービス時間は，連続した2つの処理の種類の順序により，次の4つに分けて考える必要がある。

$$\left.\begin{aligned}\frac{1}{\mu_{ii}^{i}} &= \frac{1}{\mu_i}\\ \frac{1}{\mu_{ii}^{i}} &= \frac{1}{\mu_i}\cdot(1+v_{ii})\\ \frac{1}{\mu_{i1}^{i}} &= \frac{1}{\mu_i}\cdot(1+\alpha_{i1}+v_{i1})\\ \frac{1}{\mu_{1i}^{i}} &= \frac{1}{\mu_1}(1+\alpha_{1i})\end{aligned}\right\}\text{------(3-4)}$$

従って，CPU_iの平均サービス時間は，

$$\frac{1}{\mu_i^*}=\gamma_{ii}^{i}\cdot\frac{1}{\mu_{ii}^{i}}+\gamma_{i1}^{i}\cdot\frac{1}{\mu_{i1}^{i}}+\gamma_{1i}^{i}\cdot\frac{1}{\mu_{1i}^{i}}+\gamma_{11}^{i}\cdot\frac{1}{\mu_{11}^{i}}\quad\text{--(3-5)}$$

となる。但し，γ_{jk}^{i} $(j,k=1\text{ or }i)$は，CPU_jの処理の後にCPU_kの処理を行なう確率であり，次式で表わされる。

$$\left.\begin{aligned}\gamma_{ii}^{i} &= \left(\frac{P_{Ji}}{P_{Ji}^*}\right)^2\\ \gamma_{i1}^{i} &= \gamma_{1i}^{i}=\left(\frac{P_{J1}\cdot\chi_{1i}}{P_{Ji}^*}\right)\left(\frac{P_{Ji}}{P_{Ji}^*}\right)\\ \gamma_{11}^{i} &= \left(\frac{P_{J1}\cdot\chi_{1i}}{P_{Ji}^*}\right)^2\end{aligned}\right\}\text{----(3-6)}$$

（4）LPUの平均サービス時間

（3）と同様にして，LPUの平均サービス時間は，次式のように求められる。

$$\frac{1}{\mu_L^*}=\gamma_{LL}^{L}\cdot\frac{1}{\mu_{LL}^{L}}+\gamma_{IL}^{L}\cdot\frac{1}{\mu_{IL}^{L}}+\gamma_{L1}^{L}\cdot\frac{1}{\mu_{L1}^{L}}+\gamma_{11}^{L}\cdot\frac{1}{\mu_{11}^{L}}\quad\text{--(3-7)}$$

但し，γ_{jk}^{L}, μ_{jk}^{L} $(j,k=1\text{ or }L)$は次式で表わされる。

$$\left.\begin{aligned}\frac{1}{\mu_{LL}^{L}} &= \frac{1}{\mu_L}\\ \frac{1}{\mu_{IL}^{L}} &= \frac{1}{\mu_I}\cdot(1+v_{IL})\\ \frac{1}{\mu_{L1}^{L}} &= \frac{1}{\mu_1}\cdot(1+\alpha_{IL}+v_{L1})\\ \frac{1}{\mu_{11}^{L}} &= \frac{1}{\mu_1}\cdot(1+\alpha_{IL})\end{aligned}\right\}\text{------(3-8)}$$

$$\left.\begin{aligned}\gamma_{LL}^{L} &= \left(\frac{P_{LL}^*+P_{JL}}{P_{LL}^*+P_{JL}^*}\right)^2\\ \gamma_{IL}^{L} &= \gamma_{L1}^{L}=\left(\frac{P_{LL}^*+P_{JL}}{P_{LL}^*+P_{JL}^*}\right)\cdot\left(\frac{P_{J1}\cdot\chi_{IL}}{P_{LL}^*+P_{JL}^*}\right)\\ \gamma_{11}^{L} &= \left(\frac{P_{J1}\cdot\chi_{IL}}{P_{LL}^*+P_{JL}^*}\right)^2\end{aligned}\right\}\text{--(3-9)}$$

3．3．代替方式

CPU_Lがボトルネックとなった時の代替方式として，次の4種類が考えられる。

DIS1 ： 1台のプロセッサのみが代替する。

DIS2 ： 残りのEXECプロセッサ4台が，均等に，CPU_Lの負荷を分担する。

DIS3 ： 残りのEXECプロセッサが，現負荷に応じて，CPU_Lの負荷を代替し処理する。

DIS4 ： ひとつのプロセッサが，ある一定負荷以上にならないように，将棋倒し式に分配する。

負荷状態や3.2節に示したオーバヘッドの値により，各方式の有効性が異なる。次節では，上記各方式の基本となる代替方式DIS1を取りあげ，各種負荷状態，オーバヘッドの値に対して，性能の変化を調べ，DIS1の有効性，適用領域を検討する。

3．4．代替方式DIS1の解析

3．4．1．前提

（1）代替関係

CPU_2が，CPU_1の負荷を代替し処理する。その他のプロセッサは，本来の処理のみを行なうものとする。

（2）オーバヘッド

機能変身方法としては，3.1.1.節の②の方法のみによって実現するものとする。即ち，CPU_2がCPU_1の機能を代行した場合の，制御記憶のアクセス時間の増大による処理速度の低下率（α_{12}：縮退率）のみをオーバヘッドとして考慮する。

（3）評価尺度

機能変身による負荷制御の効果を評価する尺度としては，応答時間の減少率を用いる。

即ち，インタラクションの特性（平均自由走行ステップ数$\{d_{ij}\}$，プロセッサ間遷移確率$\{P_{ij}\}$），システム内インタラクション数(M)が等しいという条件下で，負荷配分を行なった場合の平均応答時間をR_S^*，負荷配分を行なわない場合の平均応答時間をR_S，とすると両者の比R_S^*/R_Sを評価尺度とする。

従って，R_S^*/R_S が1より小さい時に，制御効果があるといえる。

3．4．2．負荷均等条件

3.4.1.節の前提の下で，CPU_1 の負荷と CPU_2 の負荷とを等しくする様な処理分配率 χ_{12} を求める。負荷配分を行なった場合の各プロセッサの負荷量を D_j^* $(j=J,L,1,2,3,4)$，総負荷量を D^* とする。

$1/\mu_1^* = 1/\mu_1$ ，$P_{J1}^* = P_{J1}(1-\chi_{12})$ より，

$$D_1^* = D_1 \cdot (1-\chi_{12}) \quad \text{----------(3-10)}$$

(3-5)式に，(3-4)，(3-6)および前提条件（$v_{12}=0$，$v_{21}=0$）を代入すると次式を得る。

$$\frac{1}{\mu_2^*} = \frac{1}{P_{J2}^*}\cdot\left[\frac{P_{J2}}{\mu_2} + \frac{P_{J1}\cdot\chi_{12}}{\mu_2}\cdot(1+\alpha_{12})\right] \quad \text{----(3-11)}$$

$$\therefore D_2^* = \frac{1-P_{JL}^*}{P_{JL}^*}\cdot\frac{P_{J2}^*}{P_{JL}^*}\cdot\frac{1}{\mu_2^*} = D_2 + D_1\cdot\chi_{12}\cdot(1+\alpha_{12}) \quad \text{-(3-12)}$$

従って，総負荷量 D^* は，次式で表わされ，

$$D^* = D + D_1\cdot\chi_{12}\cdot\alpha_{12} \quad \text{-----(3-13)}$$

機能変身によるオーバヘッドの量は，$D_1\cdot\chi_{12}\cdot\alpha_{12}$ であることがわかる。

また，CPU_1 と CPU_2 の負荷を等しくするような処理分配率の値を χ とすると，(3-10)，(3-12)式より，

$$D_2 + D_1\cdot\chi\cdot(1+\alpha_{12}) = D_1\cdot(1-\chi) \quad \text{---(3-14)}$$

となり，

$$\chi = \frac{1}{2+\alpha_{12}}\cdot\left(1-\frac{V_2}{V_1}\right) \quad \text{------(3-15)}$$

$$= \frac{1}{2+\alpha_{12}}\cdot\left(1-\frac{P_2}{P_1}\right) \quad \text{-----(3-16)}$$

を得る。

評価尺度 R_S^*/R_S を最小にする為には，(3-13)式で示されるような総負荷量の増加を考慮しなければならないので，処理分配率は上式の χ の値よりも小さくする必要がある。

3．4．3．数値例

負荷状態が，Case B_1（$V_1=1.5/6$ および，$V_1=2/6$）の場合の数値例を示す。

（1）インタラクション数と R_S^*/R_S との関係．

図12.，図13.に，縮退率（α_{12}）が0.1と0.4の場合のインタラクション数（u）と R_S^*/R_S の関係を示す。

各図（a）において，CPU_1，CPU_2 の使用率の変化を点線で示す。また参考として，理想的な負荷制御を行なった場合（Case A．：オーバヘッドなし，完全に負荷がバランスしている状態）の R_S^*/R_S の値を示す。

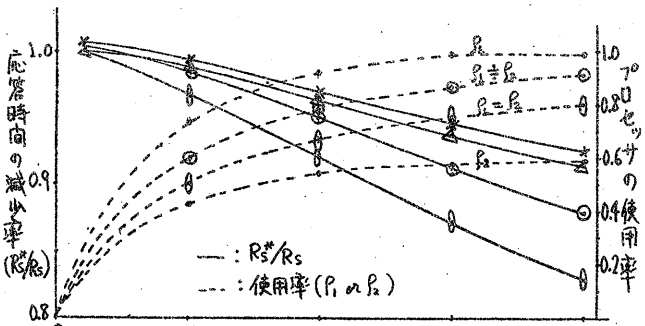
（2）処理分配率と R_S^*/R_S との関係

図14.，図15.に，縮退率が0.1，0.4の場合の処理分配率（χ_{12}）と R_S^*/R_S を，インタラクション数をパラメータにして示す。

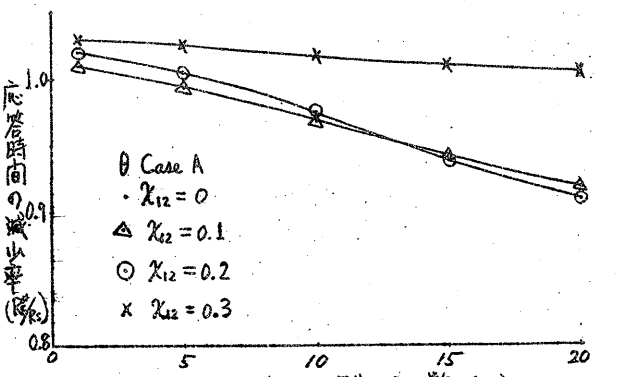
（3）縮退率と R_S^*/R_S との関係

図14.，図15.より，各縮退率において，R_S^*/R_S を最小とするような処理分配率（χ_{12}）を求めることができ，この場合はDIS1方式における最適な処理分配と考えられる。

図16.に，インタラクション数が10，15の場合の，縮退率と最適処理分配時の R_S^*/R_S との関係を示す。



（a）縮退率（α_{12}）が0.1の場合



（b）縮退率（α_{12}）が0.4の場合

図12．インタラクション数と応答時間減少率の関係(1)
（$V_1=1.5/6$ の場合）

また，(3-15)式で与えられるαを処理分配率とした場合のR_S^*/R_Sと，縮退率との関係を同図に点線で示す。

3．5．考察

3.4節の結果から，次の事を指摘できる。

（1）応答時間を最小にするような最適な処理分配率は，代替し合う2台のプロセッサの負荷を均等にするという条件から求めた処理分配率で近似することができる。

即ち，システム性能は，1インタラクション当りの負荷量よりも，ボトルネックに対して敏感であり，縮退率が1（即ち，該プロセッサに対する処理量と同等のオーバヘッドが生じる場合）程度以下では，オーバヘッドによる処理の増加を考慮せずに，負荷を平滑化することだけに注目すればよい。

（2）機能変身効果は，インタラクション数に対して敏感である。

即ち，インタラクション数によって機能変身効果は大きく変化し，インタラクション数が多い時には，効果は大きく，インタラクション数が少ない時には効果は小さいかあるいは逆効果であるといえる。

（3）通常の負荷量（各プロセッサの負荷が全て等しい状態で処理する場合，各プロセッサの使用率が0.7程度となる負荷量）の場合，1台のプロセッサのみが代替するという最も簡単なDIS1方式でも十分に有効である。しかし，縮退率が1程度と大きく，しかも応答時間に対して厳しい要求条件がある場合には，他の複雑な方式を検討する必要がある。

4．あとがき

機能分散，密結合型の計算機複合体であるポリプロセッサ・システムについて，処理の流れに注目したQueuing Networkモデルを提案し，理論的な解析を行ない，いくつかのシステム特性値を評価した。

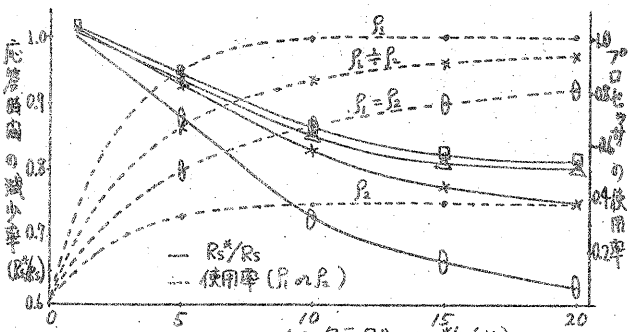
数値例により，ソフトウェア・シミュレーション結果と極めて良く一致している事を示し，モデルの妥当性を示した。

また，各種負荷状態におけるシステム性能を数値例として求め，負荷状態とシステム性能の関係について考察を行なった。

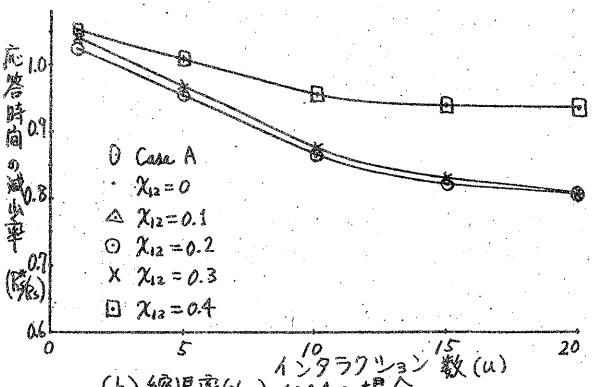
次に，プロセッサ機能の変身による負荷制御を行なった場合の待ち行列モデルを設定し，機能変身時の各種特性量の評価式を定式化し，近似的に解析した。

ひとつのプロセッサのみが機能変身し，過負荷プロセッサの処理を代行する方式に対して，数値例を示し，制御効果を評価した。

今後，複数台のプロセッサが負荷を代替処理する方式等，本文で示した各種代替方式に対して，その特性を求め，適用領域を明らかにしていく予定である。



(a) 縮退率(α_{12})が0.1の場合



(b) 縮退率(α_{12})が0.4の場合

図13．インタラクション数と応答時間減少率の関係(2)
（$V_1 = 3/6$ の場合）

最后に，御討論頂いた高平室長はじめ，基一室諸氏に深謝する。

参考文献

[1] 長谷川：シミュレーションによるPPSの処理特性の解析，昭和51情処大会，107.
[2] 小山：シミュレーションによるPPSのプロセッサ間交信特性の解析，昭和51情処大会，194.
[3] 村上，高橋：機能分散型システムにおける負荷管理方式，昭和52信学会部門別大会，347.
[4] H.W. Lynch, J.B. Page：The OS/VS2 release 2 system resource manager IBM Systems Journal 13, 4, 1974.
[5] W.A. Wulf：Performance monitors for multiprogramming systems, Proc. of the ACM OS Symposium, 1969.
[6] M. Radel, et al.：Adaptive optimization of a time-sharing system performance, Proc. of the IEEE, 63, 6, 1975.
[7] 村上，佐藤，長谷川：動的マイクロプログラム実験システム，信学技法，EC76-69.
[8] W.J. Gordon, G.F. Newell：Closed queuing systems with exponential servers, Oper. Res., 15, 2, 1967.
[9] J. Buzen：Analysis of system bottlenecks using a queuing network model, ACM-SIGOPS Workshop on System Performance Evaluation, 1971.
[10] K. Murakami, et al.：Poly-processor system analysis and design, Proc. of 4th Annual Symp. on Computer Architecture, 1977.
[11] 村上，佐藤，小山：ソフトウェア・シミュレーションによるポリプロセッサ・システムの動作解析，信学論（D），J61-D, 2, 1978.
[12] 村上，佐藤，長谷川，：ポリプロセッサ・システムにおける動的変身機構とその評価，信学論（投稿中）
[13] T.G. Price, Jr.：A comparison of queuing network models and measurement of a multiprogrammed computer system, Performance Evaluation Review,

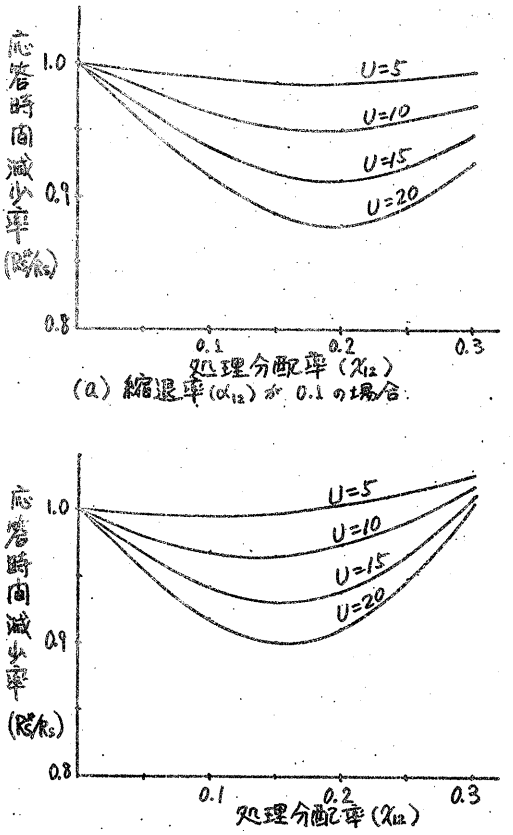
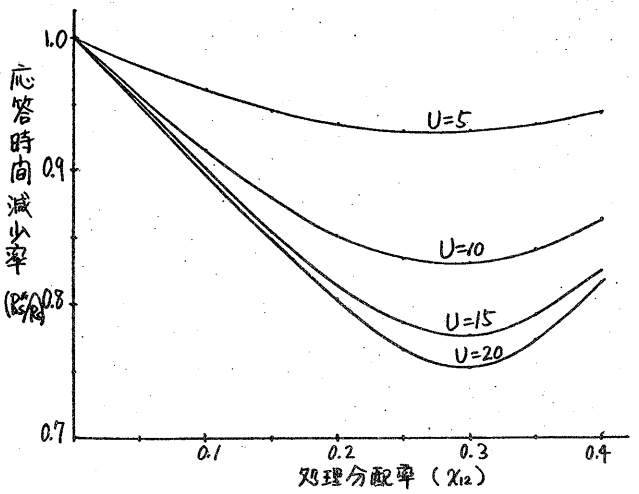
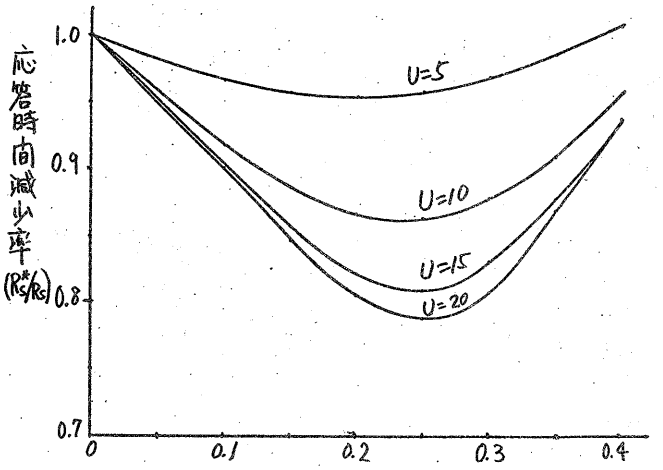


図14. 処理分配率と応答時間減少率の関係（1）
（$V_1 = 1.5/6$ の場合）



（a）縮退率（α_{12}）が 0.1 の場合



（b）縮退率（α_{12}）が 0.4 の場合

図15. 処理分配率と応答時間の減少率の関係（2）
（$V_1 = 2/6$ の場合）

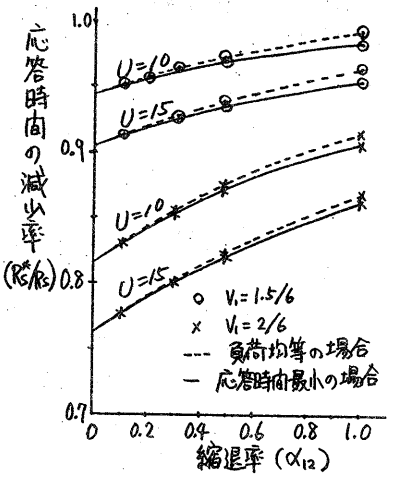


図16. 縮退率と応答時間の減少率の関係.