（1977．9．9）

性能評価用ツールQM-1について

紀　一誠，本郷喜裕，松田敏男
（日本電気　第1システム支援部）

1．はじめに

性能評価技術は技術の原型としてはほぼ出揃って来ており，今後は現実の場への適用を通じて新しい展開を行なっていくことが重要である，とされている。(9)

しかし，評価の必要性を生み出すコンピュータの世界は複雑怪奇であり，それに携わる人々の数も膨大なものである。　従って，一口に性能問題といっても各人の関心の在り方や議論の視点立場もさまざまであり，必ずしも一致した認識が得られているとも言い難いのが実情とみえる。

個々の性能評価技術は方法論的に吟味され，現場での検証に耐え，次第に淘汰体系化され統一された認識へと発展していくものであろうが，そこに至るにはまだ幾多の年月と努力を要するように思われてならない。

本稿はシステムの動的性能を把握することを目的とした性能評価用ツールQM-1について述べたものである。

QM-1の作成意図は第1義的には現場での使用に耐え得る実用的な性能評価ツールを用意することであるが，ツールを提供することによって現場における性能評価作業に多少なりとも統一的な手掛りを示せるなら望外の成果である。

QM-1は社内のシステム・エンジニア（SE）向けの汎用性能評価ツールである。必ずしも性能評価作業に熟達しているとは限らない各SEが自分の担当するシステムに関しての必要な性能評価指標を手軽に算出するためのツールであり，実体はFORTRANで記述されたプログラムである。

QM-1は解析型のツールである。　即ち，性能評価用の数学モデルを作りこの解をプログラム化したもので，モンテカルロ型のシミュレーションを行なっているわけではない。　QM-1の基礎をなすのはT．L．Saatyの機械修理工問題(1)である。これを2．で示す。　しかし，このモデルをそのままでは性能評価用モデルとしては不充分であり，不足な部分を補うためいくつかの概念を持ち込まねばならない。この補足と拡張の内容を3．で示す。　2．で示されたモデルに3．で示した補足拡張を行なった結果再構築されたモデルを4．に示す。　4．の内容をプログラム化したものがQM-1である。

解析型の性能評価ツールは今までにQNET4(6)，ATOM(4)，CQNM(5)等各々特徴のある興味深い報告がなされている。　QM-1は根拠としている数学モデルがこれらのものとは異なっている。

本モデルはもともと日本電信電話公社のTSSサービスであるDEMOSの性能評価の際の解析モデルとして誕生したもので，'71～'72年に行なった実測実験結果との照合を行ない比較的良い結果を得たものである。(3)　その後マルチプロセッサ・システム（DIPS）で生ずるメモリ競合による性能低下の推定に応用した。(2)　しかし，実際のシステム評価に応用するには数値計算が複雑に過ぎ，mやcの値を現実的なものにするとすぐに電卓計算の範囲を越えてしまうため実用上は難点があった。今回のツール化はこの難点を解消することが大きなねらいであった。

2．T．L．Saatyの機械修理工モデル

QM-1の基礎をなすT．L．Saatyの機械修理工モデルについて説明し，Saatyの導いた

諸式を示す。（文献〔1〕参照）　図1参照。

m台の機械があり，各機械は異なるC種類の故障を起こす。機械修理工はC人居り，各人は1種類の故障しか修理できない。故障iを起こした機械は修理工iに修理してもらうが，先着の故障機械が既にあれば待ち合せに入る。修理工は先着順に修理する。故障iを修理するのに要する時間はパラメータμ_i（1/秒）なる指数分布に従うものとする。また，各機械の故障iの発生分布はパラメータλ_i（件/秒）なるポアソン分布に従うものとする。

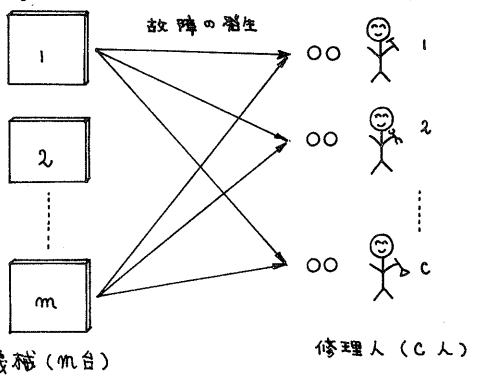


図1．機械修理工モデル

$(m_1, m_2, \cdots, m_c)$を故障iをm_i台の機械が起こしている状態とし，その定常状態確率を$P(m_1, m_2, \cdots m_c)$とする。

Saatyは定常状態における平衡方程式を作り，これを解いて以下の諸式を得ている。（表現は原文のまま）

$$\rho_i = \lambda_i / \mu_i \quad , \quad m = \sum_{i=1}^{c} m_i \quad , \quad F(\rho_i, m) = 1 + m\rho_i + m(m-1)\rho_i^2 + \cdots\cdots + m!\rho_i^m$$

$$P(m_1, m_2, \cdots, m_c) = \frac{m!}{(m-m)!} \prod_{i=1}^{c} \rho_i^{m_i} P(0, 0, \cdots, 0)$$

$$P(0, 0, \cdots, 0)^{-1} = \sum_{i=1}^{c} \rho_i^{c-1} F(\rho_i, m) \Big/ \prod_{\substack{i=1 \\ j \neq i}}^{c} (\rho_i - \rho_j)$$

平均稼動機械台数　$E(m-m) = m \sum_{i=1}^{c} \dfrac{\rho_i^{c-1} F(\rho_i, m-1)}{\prod_{i=1, j \neq i}^{c} (\rho_i - \rho_j)} P(0, 0, \cdots, 0)$

Machine availability

$$E(m-m)[P(0, 0, \cdots, 0)]^{-2} / m$$

（以上）

3．モデルの拡張

2．で述べた機械修理工モデルを性能評価用モデルに拡張し，不足な部分を補ない実用化を考える。そのためには次の2つの作業が欠かせない。

A)．モデルを使用する人が住んでいるコンピュータの世界の用語や概念にモデルをなじませること。入力となるパラメータλ_i，μ_iは利用者から見ればわかりにくくまた決めにくいパラメータになるので直接指定させるのは不適切である。これに替る入力パラメータとして日常なじみ深いJOB（あるいは電文）当りの各資源ののべ保留時間$\{\rho_i\}$（秒/JOB）及びJOB処理件数ν（件/秒）を直接入力できるようにする。

このためには呼空間隔[8]の概念を持ち込み，負荷を呼源呼量[8]としてみる概念の拡張が必要である。

B)．機械修理工モデルに欠ける点を補うこと。

2．で導かれた諸式の中には性能評価上の重要な指標である各資源の使用率α_iや資源の待ち時間w_iの算出式が欠けているのでこれを求める。また2．のモデルから直接にそのシステムの最大処理件数$\nu_{max.}$を求めることはできないが，この算出式を該モデルの極限型モデルとして導く。

上記A)，B)の作業を進める上の基本的な考え方を以下に示す。

まず用語は次の如くに読み換える。機械修理工はシステム内に存在する各種資源，即ちCPU装置，DISK装置(1)，----，等と思い込む。故障はこれらの各資

源に対して発せられる使用要求になぞらえられる。　この使用要求のことをトラフィック理論に従い"呼"とよぶことにする。　機械は各システム資源に呼を発する呼源とみなせる。　又システムでは呼源をマルチプログラミングの最大的重度にみたてる。　入力パラメータ設定の考え方を次に示す。　一般にシステムの性能評価の出発点は，まず利用者習性と負荷状況を調査し，それに基いてJOB(電文)の処理を分析し，負荷の単位となる平均的なJOB(電文)の処理パターンを構成する事から始まる。　即ち，JOB当りに使用するCPU時間や各種入出力装置に対する入出力発生回数等を基礎データとして見積ることに始まる。

今，パラメータ R_i，l_i を次の如くに定義しておく。

R_i ：JOB(電文)当りに使用する資源 i の総保留時間　(秒/JOB)

l_i ：JOB(電文)当りの資源 i の総使用回数　(回/JOB)

モデル上の μ_i は直に解釈すれば資源 i の1回当りの保留時間の逆数である。即ち，$\mu_i = l_i / R_i$ である。　μ_i を入力パラメータにすると R_i，l_i の両者が基礎データとして見積られていなければならないことになる。　しかし，例えば，CPU資源についてみればこの l_i の見積りは実際にはすこぶる困難である。　l_i は1JOB当りのディスパッチング回数，割り込回数，---等CPUの制御の切れ目を数えあげねばならない。　しかし，次の点に注意されば l_i は議論の本筋とは無関係なパラメータであることが理解される。　1JOB当りの資源 i の1回当り待ち時間を $\hat{w}_i$ とすれば次の関係が成り立つ。　$w_i = l_i \hat{w}_i = l_i \times (\text{定数}) / \mu_i = R_i \times (\text{定数})$，即ち，待ち時間は保留時間に比例する。　w_i を問題にする限り R_i が本質的であり，l_i は非本質的なパラメータである。　従って，入力に $\{R_i\}$ をそのまま指定できるようにし，l_i を指定するという非本質的な煩わしさを利用者に負わせないための一工夫が必要になる。　このために呼空間隔Uなる概念をトラフィック理論から借用し，$\{R_i\}$ を呼源呼量と考えることにする。　$\{R_i\}$ を入力とすることによりJOBの概念を入れていることになる。　呼空間隔Uは呼源が呼を発していない時間々隔として定義される。(機械の故障間々隔に相当する。)　Uは確率変数であるが，その期待値を u とする。　定義より，$\lambda_i = l_i / u$ なる関係が成り立っている。また，$R_i = l_i / \mu_i$ であるから，$\rho_i = \lambda_i / \mu_i = R_i / u$　なる関係にある。　よって，資源の保留回数 l_i は表面には出て来ない。　代りに平均呼空間隔 u が現われて来る。　これはJOB処理(到着)件数 γ (件/秒)が定まれば定まる値である。

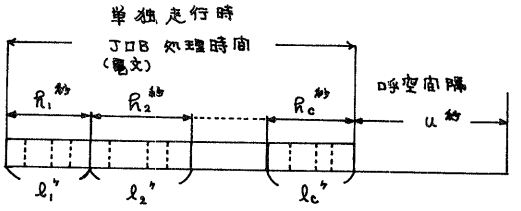
よって，利用者に見せる入力パラメータとしては，JOB特性 $\{R_i\}$ 及びJOB処理件数 γ，資源総数 c，マルチプログラミング最大的重度 m になる。

この時，各資源の使用率 α_i，待ち時間 w_i，JOB処理時間，平均マルチプログラミング的重度等を計算し出力するようにする。

また，γ を指定せずに，c，m，$\{R_i\}$ のみを指定した場合には後に述べる極限型モデルを用いて，このシステムで処理できる最大処理件数 $\gamma_{max.}$ (件/秒)を上記指標と共に計算し出力する。

以下に，2.のモデルに対して筆者が拡張補足を加えた点をまとめておく。

① 呼空間隔U及びJOB(電文)なる概念を持ち込み，$\{R_i\}$ を呼源呼量と考える事により，入力パラメータを c，m，$\{R_i\}$，γ



R_i ：資源 i 総保留時間/JOB(電文)

l_i ：〃 総保留回数/JOB(電文)

図2　呼源呼量と呼空間隔

という理解しやすいものに変えた。

② 任意の処理件数 ν を指定された時の諸量が求められるようにした。

③ 資源 i の使用率 α_i を求める計算式を導いた。

④ 資源 i の待ち時間 W_i を求める計算式を導いた。

⑤ 呼空間隔 $1/\mu$ を，$1/\mu \to 0$ として極限型モデルを導き，これによりシステムの最大処理件数 $\nu_{max.}$ を求める計算式を導いた。

4. モデルの再構成

2. で示したモデルに3. で述べた拡張を行い，QM-1 を実現するに必要なモデルを以下に再構成する。

m : 呼源数（マルチプログラミングの最大多重度）

c : システム内に存在する資源総数

R_i : 1 JOB（電文）当りの資源 i の総保留時間

ν : JOB（電文）処理（到着）件数

μ : 平均呼空間隔

① 系内呼数の定常分布 $P(m_1, m_2, \cdots, m_c)$

$$P(m_1, m_2, \cdots, m_c) = \frac{m!}{(m-n)!} \prod_{i=1}^{c} \rho_i^{m_i} P(0, 0, \cdots, 0) \quad , \quad n = \sum_{i=1}^{c} m_i$$

$$P(m) = P(0, 0, \cdots, 0)^{-1} = \sum_{n=0}^{m} \frac{m!}{(m-n)!} \prod_{i=1}^{c} \rho_i^{m_i}$$

$$= 1 + m\,K(1) + m(m-1)K(2) + m(m-1)(m-2)K(3) + \cdots + m!\,K(m)$$

$$K(l) = \sum_{m_1+\cdots+m_c=l} \prod_{i=1}^{a} \rho_i^{m_i} = \sum_{m_1=0}^{l} \sum_{m_2=0}^{l-m_1} \cdots \sum_{m_c=0}^{l-(m_1+\cdots+m_{a})} \rho_1^{m_1} \rho_2^{m_2} \cdots \rho_c^{m_c}$$

② 1 JOB（電文）当りの資源 i の待ち時間 W_i

$f(m_1, m_2, \cdots, m_c)$ を呼の出合う確率とする。[8] 即ち，呼の発生直前時点で資源 i を保留中又は待ち合せ中の呼数が m_i である確率とする。また，C_i を資源 i に対する生起呼数密度とする。$n = \sum_{i=1}^{c} m_i$，$n \leq m-1$ に注意して次式を得る。

$$C_i = \sum_{n=0}^{m-1} (m-n)\lambda_i P(m_1, m_2, \cdots, m_c)$$

従って，単位時間当りに発生する全資源要求回数 C_0，即ち $C_0 = \sum_{i=1}^{c} C_i$ を考え，この中で状態 $(m_1, m_2, \cdots, m_c)$ に出合うものを考えると次式により与えられる。

$$(m-n)(\lambda_1 + \lambda_2 + \cdots + \lambda_c) P(m_1, m_2, \cdots, m_c)$$

従って，以下の式を得る。

$$f(m_1, m_2, \cdots, m_c) = (m-n)(\lambda_1 + \lambda_2 + \cdots + \lambda_c) P(m_1, m_2, \cdots, m_c) / C_0$$

$$= \frac{(m-1)!}{(m-1-n)!} \prod_{i=1}^{c} \rho_i^{m_i} f(0, 0, \cdots, 0)$$

$$f(0, 0, \cdots, 0) = 1 / P(m-1)$$

状態 $(m_1, m_2, \cdots, m_c)$ に出合った呼は平均 m_i/μ_i（秒）待ち合せねばならないことより，JOB（電文）当りの資源 i の延べ待ち時間 W_i を次式により得る。

$$W_i = \sum_{m_i=0}^{m-1} \sum_{n=0}^{m-1} \{m_i R_i f(m_1, m_2, \cdots, m_c)\}$$

$$= \frac{R_i}{P(m-1)} \{\rho_i (m-1) P(m-2) + \rho_i^2 (m-1)(m-3) P(m-3) + \cdots + \rho_i^{m-1} (m-1)!\, P(0)\}$$

③ JOB（電文）処理時間 $T(m)$

多重走行時には JOB（電文）処理時間は単独走行時に較べて各資源の待ち時間の和の分だけのびる。$T(m)$ は次式により与えられる。

$$T(m) = \sum_{i=1}^{c} (R_i + W_i)$$

$$= \sum_{i=1}^{c} R_i \{P(m-1) + \rho_i (m-1) P(m-2) + \rho_i^2 (m-1)(m-2) P(m-3) + \cdots\cdots + \rho_i^{m-1} (m-1)! P(0)\} / P(m-1)$$

④ 各資源 i の使用率 α_i

$$\alpha_i = c_i / \mu_i = \rho_i \sum_{n=0}^{m-1} (m-n) P(n_1, n_2, \cdots\cdots, n_c)$$

$$= \rho_i m P(m-1) / P(m)$$

⑤ 平均マルチプログラミング多重度 E

$$E = m - \sum_{n=0}^{m} (m-n) P(n_1, n_2, \cdots\cdots, n_c)$$

$$= m \{1 - P(m-1) / P(m)\}$$

⑥ 平均呼空間隔 u と E の関係

$$E = mu / \{T(m) + u\}$$

⑦ JOB(電文)処理件数 ν と E の関係

$$E = \nu T(m)$$

⑧ $T(m)$, $P(m)$ の計算法

QM-1 では, $T(m)$, $P(m)$ を次なる漸化式が成り立つ事を利用して順次求める方法を採っている。 即ち, $l = 1 \sim m$ に対して次なる関係が成り立つ。

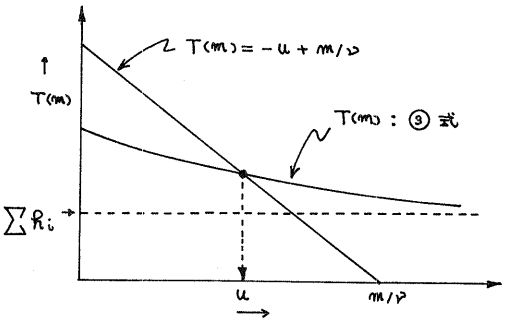


図 3 グラフによる u の求め方

$$P(0) = 1, \quad P(l) = P(l-1)\{1 + T(l)/u\}$$

$$T(l) = \sum_{i=1}^{c} R_i \{P(l-1) + \rho_i (l-1) P(l-2) + \rho_i^2 (l-1)(l-2) P(l-3) + \cdots\cdots + \rho_i^{l-1} (l-1)! P(0)\} / P(l-1)$$

⑨ ν と m の関係

以上の諸式には u がパラメータとして含まれる。 u は入力パラメータ ν によって定まる値である。 ν を知り u を求める方法を以下に示す。(図3.)参照。

関係 ⑥, ⑦ より, $T(m) = -u + m/\nu$ 。 一方, $T(m)$ は ⑨ に示される関係があり, $\rho_i = R_i / u$ より u の関数と見ることができる。 この2式を連立させて u を求めればよいわけだが, これは m 次の代数方程式になり一般には解けない。従って, QM-1 ではこの2式のグラフの交点を求める方法により解を得ている。 交点は初期値 $u_0 = m/2\nu$ とし2分法により許容誤差範囲に入るまで捜す。

⑩ 最大処理件数 $\nu_{max.}$ の求め方 (極限型モデル)

JOB(電文)処理件数 ν には最大値 $\nu_{max.}$ が存在し, それは m, c, $\{R_i\}$ を指定すれば定まる値である。 $\nu_{max.}$ は以上求めて来た諸式において, $u \to 0$ とした極限型を求めることにより導くことができる。 これを極限型モデルとよんでいる。

極限型モデルの諸式は対応する諸式に添字 $_0$ を付して表記することにする。

$P_0(l) = \lim_{u \to 0} u^l P(l)$, $T_0(l) = \lim_{u \to 0} T(l)$ とする。 $l = 1 \sim m$ について次式が成り立つ。 $P_0(l) = P_0(l-1) T_0(l)$, $P_0(0) = 1$

$$T_0(l) = \sum_{i=1}^{c} R_i \{P_0(l-1) + R_i (l-1) P_0(l-2) + R_i^2 (l-1)(l-2) P_0(l-3) + \cdots + R_i^{l-1} (l-1)! P(0)\} / P_0(l-1)$$

この漸化式を用い, $P_0(m)$, $T_0(m)$ を順次求め, それを用いて以下に示す極限型モデルにおける諸式を得る。

$$\nu_{max.} = m / T_0(m) = m P_0(m-1) / P_0(m)$$

$$\alpha_{0i} = \lim_{u \to 0} \alpha_i = m R_i P_0(m-1) / P_0(m)$$

$$W_{0i} = R_i \{R_i (m-1) P_0(m-2) + R_i^2 (m-1)(m-2) P_0(m-3) + \cdots\cdots + R_i^{m-1} (m-1)! P_0(0)\} / P_0(m-1)$$

$$T_0(m) = \lim_{u \to 0} T(m) = \sum_{i=1}^{c} \{R_i + W_{0i}\}, \quad E = m$$

5. ツール化と使用法

QM-1 は, 先に示されたモデルから導かれた諸式に従って, 性能評価作業に必要な諸量を算出するためのプログラムである。

設計にあたり, このプログラムを, 汎用的な性能評価のためのツールとするた

めに，以下の点に留意した。

① QM-1 に不慣れな SE でも，容易に使用できるツールとするために，TSS モードでの使用を基本として考え，会話形のデータ入力，カタカナによるメッセージ，入力エラー時の再入力，等を採用した。

② 性能評価作業に熟達していない SE が，自己の問題に QM-1 を応用できる様に，入力促進メッセージや，出力様式中で使用する用語は，SE にとってなじみ深く，かつ，できるだけ汎用的な意味をもったもの（タスク，リソース 等）を採用した。

③ 計算結果をレポート等としてまとめる際に起き易い転記ミス等を防ぐため出力様式を整えて，出力帳票を切り取って，そのままレポート等へ貼込める様にした。

④ 特定の入力データをパラメータとして，何回も計算を繰返すパターンが大部分であるので，その際に，変更しない入力データの再入力を少なくする様にした。

⑤ 計算は，数値として，非常に大きな値と，非常に小さな値を乗算して，最終的に常識的な値に落着く，というものであるので，計算の手順を考えて，オーバフローやアンダフローを起さない様にした。

⑥ プログラミング言語は，FORTRAN を使用した。

QM-1 の処理概要を，図4の流れ図に示す。

QM-1 を起動すると，最初にリソースの数（C）を要求して来る。Cを入力すると，C個のリソースの，それぞれの保留時間（h_1 ～ h_c）の入力を要求する。

C と {hi} が入力されると，タスクの数（m）を要求してくる。

C，{hi}，m が与えられると，QM-1 は極限型モデルを適用して，最大処理件数（p_{max}）と，その時の諸量を計算する。計算結果は，次頁に示す，出力様式－1 で出力する。

処理件数（p）の値を変えて計算する場合は，p を入力すると，その時の C，{hi}，m を使って諸量が計算され，計算結果は，次頁に示す，出力様式－2 で出力される。

m の値を変えて計算する場合は，m を再入力することにより，p_{max} の値が計算し直される。

C または {hi} の値を変えて計算する時は，C および {hi} の値と m の値を再入力する必要がある。

p，m，C，{hi} のいずれも変更しな

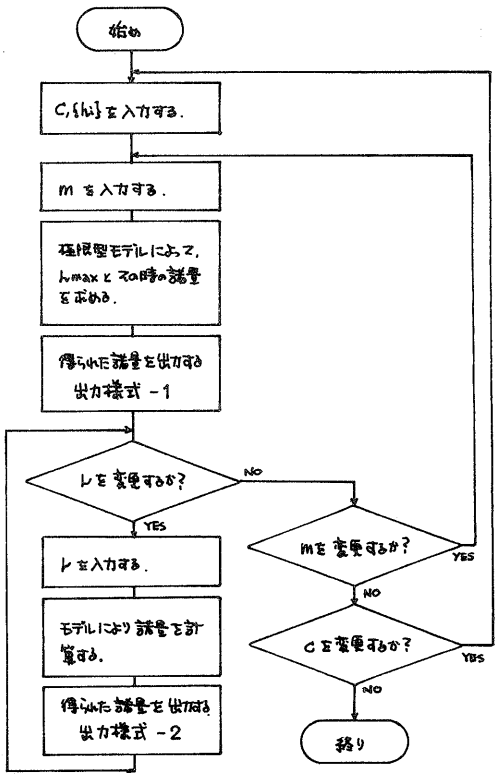


図4 QM-1 の処理の流れ

い場合は．それ以上の計算は必要ないものと見做して．QM-1 は実行を終了する．

m は 1 から 30 まで，C は 1 から 50 までの値を指定することができる。また入力データの時間の単位は．すべて同じである必要がある。（出力データは．その時間の単位で計算される。）

なお．出力データ中の，たとえば．0.98902D-2 という表現は．0.98902×10^{-2} の意味である。

```
                  ↓C                ↓m
I-------------------------------------------------------------I
I  リソース ノ カズ = 4      タスク ノ カズ = 8      タンイジカン( 秒 )  I
I-------------------------------------------------------------I
I  コノ システム ノ サイダイ ショリケンスウ              =  0.98902D-02  I ← μmax (件/秒)
I  ヘイキン ショリジカン (タスク ホリュウジカン ノ ヘイキンチ)  =  0.80888D+03  I ← To(m) (秒)
I  タスク コリョウ (ヘイキン ショリジカン * ショリケンスウ)    =  0.80000D+01  I ← To(m)×μmax
I  1 タスク アタリ ノ シヨウリツ (タスク コリョウ / タスク ノ カズ) =  0.10000D+01  I ← =m
I-------------------------------------------------------------I   └ To(m)×μmax/m
I   リソース   I   ホリュウジカン   I    シヨウリツ    I   マチジカン   I     =1
I-------------I----------------I----------------I---------------I
I   1(CPU)    I  0.98600D+02   I  0.97518D+00   I  0.45079D+03  I
I   2(Dk1)    I  0.51000D+01   I  0.50440D-01   I  0.26699D+00  I
I   3(Dk2)    I  0.66100D+02   I  0.65375D+00   I  0.10410D+03  I
I   4(Dk3)    I  0.46600D+02   I  0.46089D+00   I  0.37318D+02  I
I-------------------------------------------------------------I
                  ↑thi (秒)          ↑tai              ↑Twi (秒)
```

図5　出力様式－1

```
                  ↓C                ↓m
I-------------------------------------------------------------I
I  リソース ノ カズ = 4      タスク ノ カズ = 8      タンイジカン( 秒 )  I
I  ショリケンスウ = 0.69079D-02                                  I ← μ (件/秒)
I-------------------------------------------------------------I
I  ヘイキン ショリジカン (タスク ホリュウジカン ノ ヘイキンチ)  =  0.39232D+03  I ← T(m) (秒)
I  タスク コリョウ (ヘイキン ショリジカン * ショリケンスウ)    =  0.27101D+01  I ← T(m)×μ=E
I  1 タスク アタリ ノ シヨウリツ (タスク コリョウ / タスク ノ カズ) =  0.33876D+00  I ← T(m)×μ/m
I-------------------------------------------------------------I
I   リソース   I   ホリュウジカン   I    シヨウリツ    I   マチジカン   I
I-------------I----------------I----------------I---------------I
I   1(CPU)    I  0.98600D+02   I  0.68112D+00   I  0.11662D+03  I
I   2(Dk1)    I  0.51000D+01   I  0.35230D-01   I  0.16690D+00  I
I   3(Dk2)    I  0.66100D+02   I  0.45661D+00   I  0.41199D+02  I
I   4(Dk3)    I  0.46600D+02   I  0.32191D+00   I  0.17935D+02  I
I-------------------------------------------------------------I
                  ↑thi (秒)          ↑tai              ↑Twi (秒)
```

図6　出力様式－2

6．QM-1　使用例題

理解の助けとするため例題を示すことにする。　ある バッチ処理システムを考えることにする。　システム分析の結果（図7）に示される如くの平均的JOB特性が得られたとする。　CPUを資源1とし，ディスク装置(1)，(2)，(3)を各々資源2～4に対応させる。　C＝4，R_1＝98.6 秒，R_2＝5.1 秒，R_3＝66.1 秒，R_4＝44.6 秒

をパラメータとしQM-1に入力する。 mを指定すると(図5)の如くの出力様式-1に従って出力して来る。 この後，出力された ν_{max} 以下の ν を指定すれば(図6)の出力様式-2に従って出力を行なう。

マルチプログラミングの最大多重度 m をパラメータとして変化させた時の最大処理件数(スループット) ν_{max}. 及びJOB処理時間のびT₀(m)の関係を(図8)に示す。

mが増加すると，$\nu_{max} \rightarrow 3600/98.6 = 36.5$ 件/H に近ずく様子がわかる。 この時，T₀(m)は増加の一途をたどる。

(図9)にマルチプログラミング多重度と各装置の使用率の関係を示す。

(図10)にmの増加にともなう待ち時間の発生状況とJOB処理時間T₀(m)の増加状況を示す。 次の点には注意が必要である。

$m_1 < m_2$ に対して，$\nu_{max}(m_1) < \nu_{max}(m_2)$. であるが，νを定めた時には $T(m_1) < T(m_2)$ なる関係にある。 但し，$\nu \leqq \nu_{max}(m_1)$ 。

7. 運用と実績

QM-1は，現在ACOS-6により運営されている共同利用計算センタである NEIS 東京第2計算センタ(田町)，NTIS ACOSセンタ(新宿)の2ヶ所にTSS用コマンドとして登録されている。 利用者はセンタがクローズ運転されている時間帯に回線を介して端末から任意のセンタのQM-1を自由に呼び出して利用することができる。

TSSのコマンドとして一般開放しているため使われ方や使用実績を正確に把握する

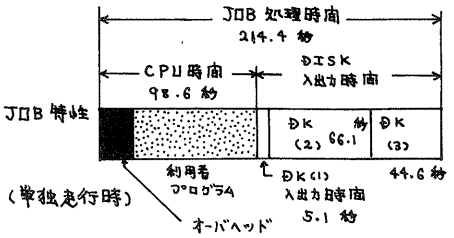


図 7. JOB 特性

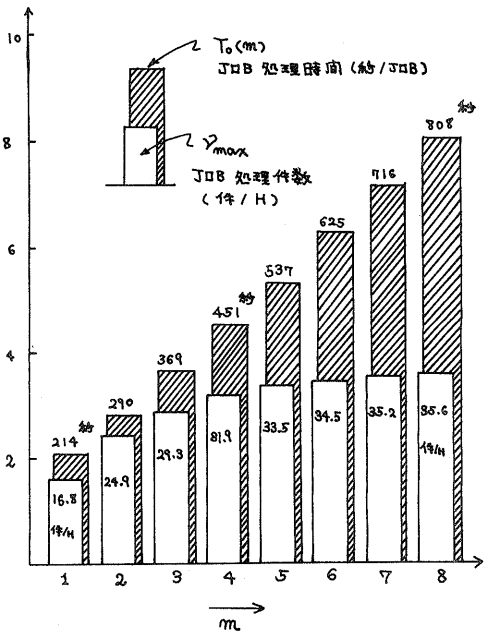


図 8. JOB処理件数とJOB処理時間

表 1. QM-1 使用例

No.	OS, 処理形態	システム概要	評価目的	評価ケース	資源数 c
1	ACOS-6, TPS, DB	帳票処理 (官庁)	詳細設計時評価	十数ケース	28
2	〃, TDS, DB	バンキング	基本設計時評価	約 20	20
3	〃, TPS, BAT, RJE, TSS	計算センタ，多次元処理	実動中，負荷の増加	20 ～30	21
4	〃, BAT, TPS, DB	生産管理，製造業(鉄鋼)	実動中，グレードアップ時評価	約 100	24
5	〃	電文業配信，帳票処理(官庁)	コンポーザル作成	約 100	25
6	〃, TDS, DB	帳票処理，保険業	客先への質問回答書作成	50 ～60	42
7	〃, TDS, DB	バンキング	基本設計時評価	5～6	20
8	ACOS-4	バンキング	実動中，グレードアップ時評価	10	10
9	DIPS-11, 104-01, 160-20 RTP	自動車情報管理システム	詳細設計，設計パラメータの検証	20～30	16

(注) TPS, TDS : オンライン処理用サブシステム , DB : データベース

ことは困難であるが，本年4月中旬に一般開放して以来現在（8月中旬）までの間にGM-1を利用した使用例のうち使用内容がある程度わかっているものを9例とりあげ，その使用法や目的，Cの値や流したケース数等を（表1）に示す。

これによると，機種やOS，処理形態を越えて使われており，使用目的もさまざまで，かなりはば広い使われ方をしているようである。

ちなみに，手計算でやるとCが20位では慣れた人で1ケース分の計算に約3日程度電卓をたたく計算量になる。

GM-1の算出する理論値と実システムで計測される実測値がどの程度合うものなのか興味深い。一例を文献(3)に示したが，装置使用率で2~3%程度の誤差であった。しかし，同一の環境を整え本当に意味のある比較を行なうのは案外むずかしいものである。

実測値と理論値のズレの発生原因は，基礎となるデータが異なっている場合とモデル自身の現実からのズレ，といった2つの異なる原因により生ずるが，実際の評価では両者は混然一体となっており，どちらがどの程度影響しているのか判然としない場合の方が多い。

8. ネットワーク型待ち行列モデルとの比較

文献(7)に示されるネットワーク型待ち行列モデル（G/Nモデル）と本モデルとの相異について示す。

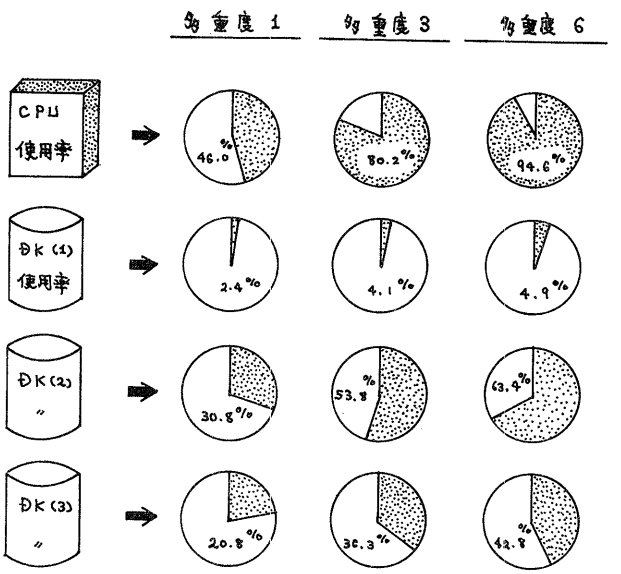


図 9　マルチプログラミング多重度と各装置使用率

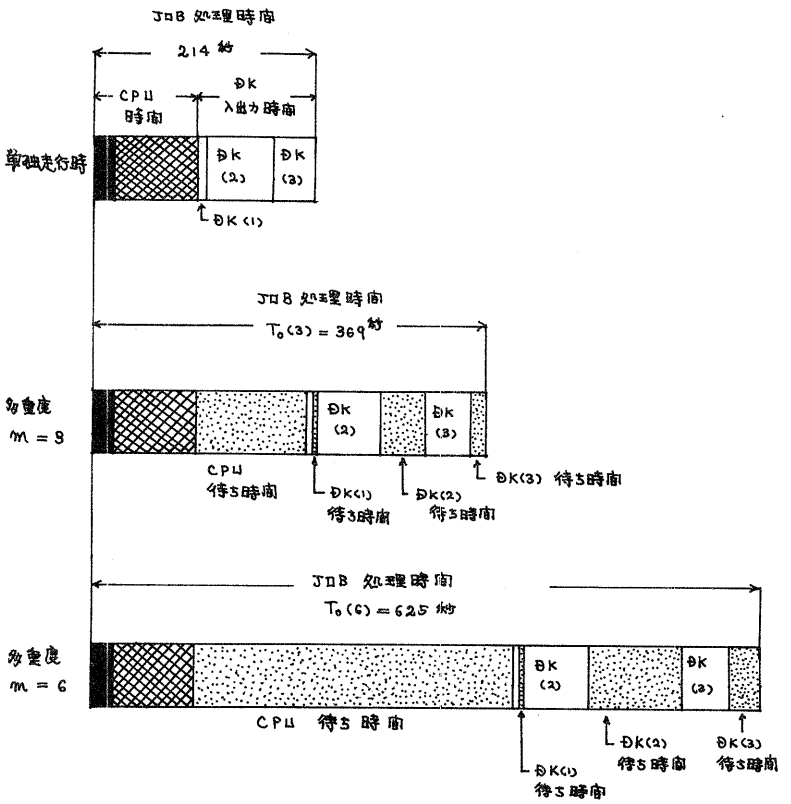


図 10.　多重度の増加による待ち時間の発生

大雑把に言えば G/N モデルは，$\mu_i \to 0$ とした QM-1 の極限型モデルに対応している。この時，$P_{ij} = \lambda_{ij}/\lambda$。 G/N モデルは故障らに対して P_i 人の修理工が存在する待ち行列構造を許すのでこの点モデル化に柔軟性がある。 反面，$\mu_i \to 0$ で呼空間隔の概念が入っておらず(機械は常に故障している)，任意の処理件数を指定された時の稼動状況が示せない。 また，JOB(電文)の概念が入っていないので $\{P_{ij}\}$ を直接は扱えず，μ_i，P_{ij} といういささか扱いにくいパラメータを陽に扱わねばならない，といった点が性能評価向けに使用する場合には補わねばならない点であろう。

9. まとめ

QM-1 のツール化に際しては何よりも使い勝手が良く大衆的なツールとなるよう心掛けた。 短期間の設計製造で意に満たぬ点も多いが，幸い利用者からは好評をいただいており，関係者一同の喜びとするところである。

なお，QM-1 には姉妹品として QM-3 が存在しており，QM-1 同様 TSS コマンドとして登録され利用されている。 QM-3 はオンライン系の応答時間の平均値や分布を QM-1 に連動させて出力する性能評価ツールである。 QM-3 では，ホスト内の電文処理時間 $T(m)$ を QM-1 と同様に計算し，さらに業務処理タスクの前にできる待ちを M/M/m でタスク呼量 $\alpha = \gamma T(m)$ として算出し，これに基き応答時間の平均値と分布を QM-1 の出力結果に合せて出力する。

QM-1 の今後の課題としては，メモリ評価関連のモデルと連動をせること，呼源を多種類にすること，資源の多重保留が扱えるようにすること，呼に優先权を付与すること，等の要請に応えていくこととなろう。

最後に，QM-1 の作成を企画し，その完成まで数々の援助をしていただいた，当部 金森吾一主任，有益な助言をいただいた守田節雄氏，並びに QM-1 を使用し貴重な御意見，感想を寄せられた当社各システム事業部の SE 諸氏に感謝の意を表します。

[参考文献]

1) T. L. Saaty ; Elements of Queueing Theory , McGraw-Hill , 1961. P. 330 ~ 332.

2) 紀一誠 ; マルチプロセッサシステムにおけるメモリサイクル競合問題の解析，情報処理学会第14回全国大会，1973.

3) 新田，伊達，紀 ; TSS における性能評価のためのソフトウェア・メータリング，電気学会情報処理研究会資料，IP-73-19，1973.

4) 周藏安造，根本良子 ; オンライン評価システム(ATOM)について，情報処理学会17回全国大会，1976.

5) 池原悟 ; CQNM プログラム使用法，OR学会，計算機システムと確率モデル研究会資料，OSMOS-2(1)，1977.

6) M. Reiser ; " Interactive modeling of Computer Systems " , IBM SYST J, No. 4 , 1976.

7) Gordon, W. J and Newell, G. F. ; " Closed Queueing Systems with Exponential Servers " , JORSA, 15 (1967), 245-265

8) 雁部頴一 ; 電話交換トラヒック，オーム社，昭41.

9) 三上，箱崎，南野 ; 計算機システムの性能評価技術 [I], [II]，信学誌，第59巻 10号，12号.