解析手法とシミュレーションを組み合わせたHybrid形式による計算機性能評価シミュレータ†

木下俊之†† 吉澤康文†† 大町一彦††

解析的手法とシミュレーション手法を組み合わせて計算機システムを汎用的に性能評価可能とするHybrid型シミュレータを開発した．これは一般のシミュレーション命令のほかに，解析モデルを制御する命令を含むシミュレーション言語である．Hybrid型シミュレーションでは，計算機の外部事象の設定にシミュレーション手法を用いることでモデリングの自由度を保ち，計算機の内部処理にはQueueing Networkによる解析的手法を用いることで計算効率の向上を期する．したがって解析的手法の手続き部がシミュレーション部分と整合のとれた形で動作する必要がある．このため本シミュレータは，解析モデル制御命令を用いて任意の時刻に解析モデルを起動でき，かつ解析の終了を自動的にシミュレーション部に報告する機能を備えている．またチャネルや入出力装置の接続構成の違いによる解析モデルへの効果も，解析計算に自動的に反映される．解析モデルに使用した計算アルゴリズムは，複数ジョブ・クラスを含むQueueing Network Modelに関するMean Value Analysisである．本論文では，このHybrid型シミュレータの動作の概要，命令体系ならびに解析手法の論拠を明らかにし，汎用化のために採り入れた諸機能について説明する．また本シミュレータの効率と精度を詳細シミュレーションと比較検討する．

1. まえがき

現在，計算機システムの性能予測・性能評価のためのモデリングには，おもに解析的手法やシミュレーション手法が用いられる．このモデリングでは，割込み動作や入出力動作などの計算機の内部処理だけでなく，ジョブやトランザクションの入力特性やジョブ管理によるスケジュールといった計算機の外部事象もモデル化する必要がある．一般に前者の事象は数ミリ秒〜数十ミリ秒の短い間隔で発生し，後者の間隔は数秒〜数十秒であり，その比は数百〜数千倍となる．したがってシミュレーション手法を用いた場合，ジョブやトランザクションの挙動を十分に解析するためには，長大なシミュレーション時間を必要とする．このように短い間隔で発生する計算機の内部処理と，これに比べて発生間隔の長い外部事象を同時にシミュレーションすることは効率の点で必ずしも得策とはいえない．

一方，解析的手法は計算機の内部処理をQueueing Networkでモデル化し数値計算により解析するので，計算時間を短くすることが可能である．したがって効率上は計算機の内部処理モデルに適する．しかし計算機の外部事象モデルでは，実測のトレース・データに基づいてジョブの発生時刻を入力したり，優先順位によるジョブ・スケジュールなど多様な形態が考えられ，これを解析的手法でモデル化することは困難である．

そこでシミュレーション手法のうち，計算機の内部処理を解析的手法で置き換えるHybrid型式のシミュレーション手法が考えられる．これは，計算機の外部事象はシミュレーション手法を用いることでモデリングの自由度を保ち，これに解析的手法を組み合わせてシミュレーション効率の向上を期するものである．

Schwetman[1]は内部処理モデルとして，単一ジョブ・クラスのCentral Server Modelを仮定したHybrid手法の効率と精度を，詳細シミュレーションのそれと比較検討した．またChiw[2]らは解析モデルに複数ジョブ・クラスを扱ったIBM社OS/VS2-MVS専用の評価モデルを開発した．しかしこれらは汎用性能評価ツールという位置づけにはない．また計算機用シミュレーション言語としてCSS-II (Computer System Simulator-II)[6],[7]などがあるが，これにはHybrid手法を実現する機能はない．

そこで本論文では一般の計算機システムをモデリング可能とするHybrid型シミュレータの開発と実現方式について述べる．本シミュレータは解析的手法の手続きを内蔵し，これの呼出しや対象計算機システムの構成の定義などを命令形式で記述できる汎用シミュレーション言語である．このHybrid手法における解

† A Hybrid-type Simulator for Computer Systems Performance Evaluation by TOSHIYUKI KINOSHITA, YASUFUMI YOSHIZAWA and KAZUHIKO OHMACHI (Systems Development Laboratory, Hitachi Ltd.).

†† (株)日立製作所システム開発研究所

析モデルは，シミュレーション部とのインタフェースがとれ，かつ対象システム構成の相違を反映しうる形で動作しなければならない．このため本 Hybrid 型シミュレータは特徴として次の機能をもつ．

（1） シミュレーション実行中の任意の時刻に割込みを起こす命令と解析モデルを制御する命令を用いることにより，一般のシミュレーション命令と同様の記述形式で任意に解析モデルを起動できる．

（2） 解析モデルからシミュレーション部への解析終了の報告が，完了イベントを介して自動的に行われる．

（3） 解析モデルにおいて複数のジョブ・クラス[3]を取扱い可能である．

（4） チャネルや入出力装置とその接続構成を定義命令により設定するだけで，その解析モデルに及ぼす効果が解析計算に自動的に反映される．

まず2章と3章で Hybrid 型シミュレータの動作の概要ならびに命令体系を述べ，次に4章で解析モデルの動作と用いた計算アルゴリズムの論拠を示した定理を導く．5章ではチャネル，入出力制御装置の扱いについて述べ，最後に Hybrid 型シミュレータの効率と精度を詳細シミュレーションとの比較で検討する．

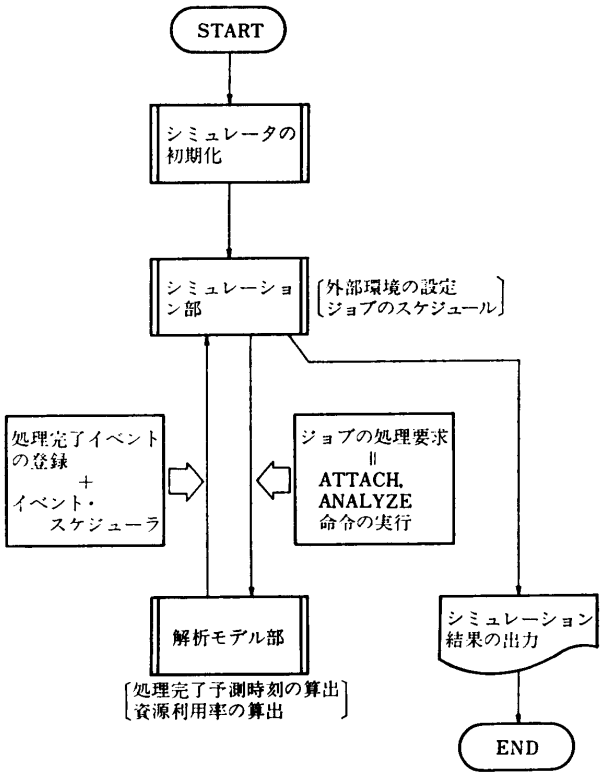

図 1 Hybrid 型シミュレータの動作概要
Fig. 1 Control flow of hybrid-type simulator.

2. Hybrid 型シミュレータの構造

Hybrid 型シミュレータは GPSS (General Purpose System Simulator), SIMULA (Simulation Language) などと同様の離散型のイベント駆動型シミュレータである．Hybrid 型シミュレータは大きく「シミュレーション部」と「解析モデル部」の二つの部分から構成される．そして計算機内部の CPU 動作や入出力動作を個別にシミュレートすることはせずに，解析モデル部における Product Form Solution をもつ Closed Queueing Network Model[3]（以下 CQNM と略す）の Mean Value Analysis[4] により，各ジョブの処理完了予測時刻の算出に置き換える．

Hybrid 型シミュレータにおけるシミュレーションの流れは次のようになる（**図1**参照）．

シミュレータの初期化の後，まずシミュレーション部の端末モデルなどで発生したジョブは，ジョブ・スケジューラによるスケジュールを受けた後，計算機内での処理要求を発行する．これにより制御は解析モデル部に移る．解析モデル部では CPU の実行ステップ数，入出力回数などを入力として処理完了の予測時刻を解析計算により算出して処理完了イベントを登録する．その後この処理完了イベントがイベント・スケジューラに拾われると，ジョブは計算機内部の処理を終了したと見なされ，制御は再びシミュレーション部に戻って事後の処理が続行する．

3. Hybrid 型シミュレータの命令体系

Hybrid 型シミュレータでは，対象計算機システムの外部事象設定のためのシミュレーション部が簡便に記述でき，かつ解析モデル部と容易にインタフェースがとれる必要がある．このために**表1**に示すようなシミュレーション命令を用意してこれを実現している．ホスト言語は PL/I 言語で，一般の変数値の代入，比較判定や分岐などには PL/I 命令を用い，シミュレータの各機能の呼出しにシミュレーション命令を用いる．

（1） シミュレーション部の記述のための命令

シミュレータの制御，ジョブ，トランザクションの制御，キュー操作，統計情報の収集など，一般のシミュレーション言語と同様の命令が含まれ

表 1 Hybrid 型シミュレーション命令一覧
Table 1 Hybrid-type simulator instruction list.

分類	処理の概要	命令名	内容・動作
システムの定義	計算機資源・接続構成の定義	CPU	CPU の名称，特性，接続構成を定義する．
		CHANNEL	チャネルの名称，特性，接続構成を定義する．
		CUNIT	制御装置の名称，特性，接続構成を定義する．
		DISK	磁気ディスク，磁気ドラムの名称，特性，接続構成を定義する．
		TAPE	磁気テープの名称，特性，接続構成を定義する．
	ジョブの要求処理量の定義	JPROT	指定されたジョブの属するジョブ・クラスを指定する．
		DEVREQ	指定されたジョブから各資源への要求処理量を定義する．
解析モデル部の制御	ジョブの制御	ATTACH	ジョブを解析モデル部に登録する．
		DETACH	ジョブを解析モデル部から取り消す．
		HOLD	ジョブを解析モデル部からはずし，処理を中断する．
	システムの状態の制御	ANALYZE	解析モデル部を起動して，解析計算を行う．
		CONTINUE	システムの状態を維持して，次のイベント発生を待つ．
シミュレーション部の記述	外部関数	$R(n_1, n_2)$	n_1 から n_2 までの整数の一様乱数
		R1	0から1までの実数の一様乱数
		E	実数の指数分布乱数
	シミュレータの制御	INITIAL	シミュレータを初期化する．
	割込みの制御	ITIMER	指定された時間経過後，指定されたシステムに割り込む．
		ENDIN	割込み状態を終了させる．
	エンティティの制御	CREATE	ジョブ，トランザクションを生成する．
		DESTROY	ジョブ，トランザクションを消滅させる．
	キューの操作	FROMQ	ジョブ，トランザクションをキューからはずす．
		TOQ	ジョブ，トランザクションをキューに登録する．
	統計情報の収集	DISTL	統計情報収集用の分布表を定義する．
		TABULATE	指定された分布表を指定された値に従って更新する．
		CLEAR	その時点までの統計情報をゼロ・クリアする．

る．これらを用いて任意にシミュレーション部を記述することが可能である．

(2) システムの定義のための命令

JPROT と DEVREQ 命令により，ジョブが解析モデル部に登録される際のジョブ・クラスと各計算機資源に要求する処理量を定義する．指定内容は，CPU の実行ステップ数，転送バイト数や入出力回数などである．また計算機システムの定義命令により，各資源の実行速度や転送速度などの特性ならびにその接続構成を定義する．この際複数台の制御装置とチャネルを接続する，いわゆるマルチ・ポート接続も指定可能で，これによりシステム構成定義の汎用性を高めている．

(3) 解析モデル部の制御のための命令

ATTACH 命令は解析モデル部にジョブを登録する命令で，その際要求する処理を選択することができる．DETACH 命令はジョブを取り消す命令である．また HOLD 命令はジョブの処理を中断する命令で，たとえば多重仮想記憶方式においてスワップ・アウトにより処理を中断するといった事象をモデル化できる．ANALYZE 命令は解析モデルに解析計算を指示する命令で，この実行により処理完了予測時刻を計算して処理完了イベントが登録される．これらの命令を用いて，シミュレーション部から自由にかつ効率的に解析モデル部を制御できる．

図 2 に Hybrid 型シミュレータの記述例を示す．初期化の後，①で CPU，ディスク，制御装置，チャネルならびにその接続構成を定義している．また②でジョブの属するジョブ・クラスと要求処理量を定義している．③からシミュレーションの実行命令が始まる．最大多重度数だけの job を生成して job wait queue (JQ (1)) に登録した後，ITIMER 命令で message 発生の契機を与える．④は ITIMER 命令の割込み処理ルーチンで，ここで message を生成し message wait queue (MQ (1)) に登録する．⑤には message 発生時と message 処理完了時に制御が渡り，wait 中の job と message があるときのみ job を解析モデル部に登録し，ANALYZE 命令により処理完了イベントが生成される．この ATTACH 命令において job-type no. を R(1,2) と指定しているので，

```
          INITIAL                                              ┐
          CPU       CPNO..CP.CPU_MIPS                          │
          DISK      DKNO1.DKNO2.DK.CUNO.TRSPEED.               │ ①
                    CYLNO.SEEKFT.ROTTIME                       │
          CUNIT     CUNO..CU.CHNO1.CHNO2                       │
          CHANNEL   CHNO1.CHNO2.CH                             ┘
          JPROT     1.1                                        ┐
          DEVREQ    CPNO..DSPCHNO..CPU_STEP                    │ ②
          DEVREQ    DKNO1.DKNO2.IONO.TRBYTE                    │
          JPROT     2.2                                        ┘
            :
          DO I=1 TO DEGREE :                                   ┐
            CREATE    CJOB                                     │
            TOQ       CJOB.JQ(1)                               │ ③
          END ;                                                │
          ITIMER    GENTIME.MSGGEN...E                         │
          CONTINUE                                             ┘
MSGGEN:   ITIMER    GENTIME.MSGGEN...E                         ┐
          CREATE    CMES                                       │ ④
          TOQ       CMES.MQ(1)                                 │
          ENDIN     MSGIN                                      ┘
MSGIN:    IF NJQ(1) > 0 & NMQ(1) > 0                           ┐
            THEN DO ;                                          │
              FROMQ     CMES.MQ(1)                             │
              DESTROY   CMES                                   │ ⑤
              FROMQ     CJOB.JQ(1)                             │
              ATTACH    CJOB.COMPLETE.R(1.2)                   │
            END ;                                              │
          ANALYZE                                              │
          CONTINUE                                             ┘
COMPLETE: TOQ       CJOB.JQ(1)                                 ┐
          GO TO MSGIN ;                                        ┘ ⑥
```

図 2 Hybrid 型シミュレータの記述例
Fig. 2 A cording example of hybrid-type simulator.

job-type 1と2が同頻度で選択される．要求された処理が完了すると，⑥において使用可能となった jobが再び job wait queue に登録されて，⑤に制御が渡る．

このように本 Hybrid 型シミュレータの記述上の特徴は，解析モデル部の起動の際は必要な時刻に割込みをかけてジョブの登録と ANALYZE 命令を発行すればよく，これにより自動的に解析計算と処理完了イベントの登録が行われ，かつ処理完了時は指定位置に制御が渡る点にある．

4. 解析モデル部の処理方式

Hybrid 型シミュレータの解析モデル部では，各 CPU と入出力装置を Queueing Network のサーバに対応づけて，Baskett[3]らによる結果を Mean Value Analysis[4] によって解析計算する．そのため次の条件を仮定する．

(1) 複数ジョブ・クラスを含みうる．

(2) 各サーバのスケジュール方式は，全ジョブを通じての FCFS (First Come First Served), LCFS (Last Come First Served), PS (Processor Sharing) のいずれかである．

(3) 各サーバにおいて指数サービスを受ける．FCFS サーバを除いてサービス率はジョブ・クラスごとに異なっていてよい．

(3)ではサービス分布は Cox 分布[3]でもよいが，指数サービスを仮定したのは，一般に CPU や入出力装置のサービス時間特性をあらかじめ知ることが困難なためである．

処理完了予測時刻の算出方法を次に示す．ただし L は Network 中のサーバ数とする．

(計算法A) ジョブ多重度が 0 のとき，時刻 t_1 にジョブ $j^{(1)}$ が入力されたとする．$j^{(1)}$ の各サーバ l ($l=1,2,\cdots,L$) への要求処理時間 $\tau_l^{(1)}$ を入力として多重度1の CQNM を解いて応答時間 $u_1^{(1)}$ を求め，$j^{(1)}$ の処理完了予測時刻 $v_1^{(1)}=t_1+u_1^{(1)}$ に処理完了イベントを登録する．次に $j^{(1)}$ が完了する以前の時刻 t_2 に次のジョブ $j^{(2)}$ が入力されたとする．$j^{(2)}$ については各サーバへの全要求処理時間 $\tau_l^{(2)}$ が解析モデルの入力となる．$j^{(1)}$ については，時刻 t_2 で残っている処理についてのみ多重度2で処理されると考える．$j^{(1)}$ の処理の残余率は $r^{(1)}=(v_1^{(1)}-t_2)/(v_1^{(1)}-t_1)$ で与えられるから，$j^{(1)}$ の各サーバ l への要求処理時間は $\tau_l^{(1)}\cdot r^{(1)}$ となる．これらを入力として多重度2の CQNM を解いて $j^{(1)}, j^{(2)}$ の応答時間 $u_2^{(1)}, u_2^{(2)}$ ならびに $v_2^{(1)}=t_2+u_2^{(1)}$, $v_2^{(2)}=t_2+u_2^{(2)}$ をそれぞれ求める．そして処理完了イベントの時刻は $v_2^{(1)}, v_2^{(2)}$ の小さい値に変更する．以下同様の操作を多重度が増減する度に多重度が0になるまで続ける．

この計算法Aでは各ジョブの要求処理時間としてその時刻の残余処理時間を用いるが，ジョブの入力時刻が異なるため，たとえ同一のジョブ・クラスに属していても各ジョブの残余処理時間は異なる．Baskett[3]らの結果は，一つのジョブ・クラスに属するジョブから同一サーバへの要求処理時間は共通の値でなければならない．したがって再計算ではたとえ本来同一のジョブ・クラスに属するジョブであっても，すべて異なるジョブ・クラスに属すると見なさなければならない．しかし次の定理により，この変更は不要であることが示される．

[定理] ジョブ・クラス数を S，各クラスの多重度を $n_s(s=1,2,\cdots,S)$，各ジョブを $j_{si}(i=1,2,\cdots,n_s)$，ジョブ j_{si} の残余率を r_{si} とする．このとき，各ジョブの初期の要求処理時間に基づいてクラス数を S として求めたジョブ・クラス s 内のジョブの応答時間を T_s，各ジョブの残余処理時間に基づいてすべてのジョブが異なるジョブ・クラスに属するとした場合のジョブ j_{si} の応答時間を T_{si} とすると，

$$T_{si}=r_{si}\times T_s$$

が成り立つ．

(証明) 付録参照．

これより，より簡便な次の計算法が考えられる．

(計算法B) いったん残余率を考慮せずに初期の要求処理時間からジョブ・クラス数を S として応答時間を求め，次にこれに各ジョブの残余率を掛けて残余応答時間とする．

Schwetman[1]が用いたのは，単一ジョブ・クラスの場合の計算法Bである．処理完了予測時刻の算出法としては計算法Aが厳密な考え方であるが，上記の定理により，複数ジョブ・クラスの場合について計算法AとBの結果は等しく，計算法でも正しい残余応答時間を求められることが示された．したがって，本 Hybrid 型シミュレータも計算法Bを採用している．

5. システム構成のモデル化

5.1 CPU，入出力装置のモデル化

Mean Value Analysis に必要な入力値は，ジョブから各計算機資源に対する要求処理時間である．しか

し一般に計算機シミュレーションにおいて，対象とするジョブの特性を記述するのに，CPU や入出力装置のサービス時間を知ることは困難な場合が多い．したがって記述性を高めるために，指定が容易な各資源の特性ならびに CPU の実行ステップ数，入出力転送量，入出力回数を指定し，要求処理時間へはシミュレータが自動的に変換する．この変換の仕方は資源の種類により異なる．

CPU については実行ステップ数と実行速度から，また磁気テープについては転送バイト数と転送速度から求めた転送時間に起上り時間を加えて求めることができる．磁気ディスクについては，平均シーク時間，平均サーチ時間，RPS(Rotational Position Sensing)[5] の空振りによる遅れ時間および転送時間の和が1回の平均入出力時間である．このうち平均シーク時間 t_{SK} は，ランダム・シークを仮定するとシーク移動距離 C の分布が，

$$p\{C=i\}=\begin{cases}1/c & (i=0)\\ 2(c-i)/c^2 & (i=1,2,\cdots,c)\end{cases}$$

(ただし c は最大シーク移動距離) の片側三角形分布に従うことから，シーク時間の特性関数 $a(i)$ (シーク移動距離 i のシーク時間) をこのシーク移動距離分布で重み付けた平均値として求められる．ここで最大シーク移動距離 c の値は，対象システムのファイルの実装シリンダ数として DISK 命令中で指定する．またシーク時間の特性関数は，機種ごとにシミュレータに内蔵されている．平均サーチ時間 t_{SRC} は，1回の回転時間 t_R の 1/2 である．RPS の遅れ時間 t_{RPS} は，チャネルと接続制御装置の少なくとも一方がビジーである率を ρ とすると，ちょうど n 回空振る確率は $(1-\rho)\rho^n$ となるから，

$$t_{RPS}=\sum_{n=0}^{\infty} nt_R\cdot(1-\rho)\rho^n=\frac{\rho}{1-\rho}\cdot t_R$$

となる．この ρ の値は 5.2 節で述べるように解析モデル部の解析計算のなかから求められ，したがって t_{RPS} も知ることができる．また転送時間 t_{TR} は，転送バイト数 B と転送速度 v の比である．以上をまとめると，磁気ディスクの1回の平均入出力時間 t_{DK} は次の式から求まる．

$$t_{DK}=t_{SK}+t_{SRC}+t_{RPS}+t_{TR}$$
$$=\sum_{i=1}^{c} a(i)\cdot\frac{2(c-i)}{c^2}+\frac{t_R}{2}+\frac{\rho}{1-\rho}\cdot t_R+\frac{B}{v}$$

5.2 制御装置の利用率

制御装置がビジーとなるのは，入出力動作のうちのデータ転送の間である．したがって制御装置の利用率は，おのおのに接続する入出力装置の転送利用率（入出力装置の利用率のうちデータ転送時間によって生ずる利用率分）を加えた値となる．Hybrid 型シミュレータでは，この転送利用率は解析モデル内のジョブの各入出力装置への残余転送時間と残余応答時間の比の値を，全ジョブについて加えた値として求めることができる．したがって，これより制御装置の利用率も求まる．

このように転送利用率が簡単に求められるのは，解析モデル部でつねに残余応答時間を算出しているためである．これは Hybrid 手法においてのみ行われる操作であり，したがって制御装置の利用率，さらにこれに伴う RPS の遅れ時間が容易に算出可能なことも，従来の解析的手法にない Hybrid 型シミュレータの一つの特徴である．

5.3 チャネルの利用率

チャネルと制御装置の接続構成は，これらが複数台同士相互に接続する，いわゆるマルチ・ポート接続をとることが一般的である．Hybrid 型シミュレータでは，この接続形態が定義可能である．この場合，各入出力要求が経由するチャネルは，その時点でアイドル状態のチャネルがランダムに選択される．したがって接続が非対称の場合には，チャネルの利用率は一般に等しくない．そこで Hybrid 型シミュレータでは，新しく次のような方法を用いてチャネル利用率を算出する．

例として**図3**の接続構成について説明する．各制御装置の利用率 a_1, a_2, a_3 は 5.2 節で述べた方法により既知とし，求めるチャネル利用率を x_1, x_2, x_3 とす

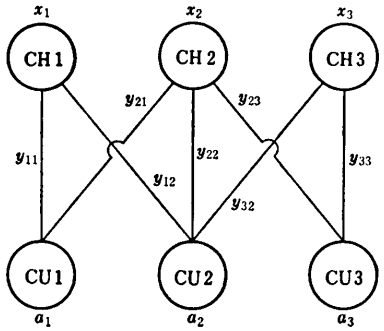

図 3 チャネルと制御装置の接続構成例
Fig. 3 An example of channel-controller configuration.
(CHi: channel, CUj: controller, x_i: channel utilization, a_j: controller utilization, y_{ij}: path utilization between CHi and CUj)

る．また，このチャネル利用率を各接続パスの割与分ごとに分類した値を y_{11}〜y_{33} とする．明らかに次の関係式が成り立つ．

$$\left.\begin{aligned} &y_{11}+y_{21}=a_1\\ &y_{12}+y_{22}+y_{32}=a_2\\ &y_{23}+y_{33}=a_3 \end{aligned}\right\} \quad (5.1)$$

$$\left.\begin{aligned} &y_{11}+y_{12}=x_1\\ &y_{21}+y_{22}+y_{23}=x_2\\ &y_{32}+y_{33}=x_3 \end{aligned}\right\} \quad (5.2)$$

また上述のチャネルの経路選択方式により，次の比例式が成り立つ．

$$\left.\begin{aligned} &y_{11}:\ y_{21}=(1-x_1):(1-x_2)\\ &y_{12}:\ y_{22}:\ y_{32}=(1-x_1):(1-x_2):(1-x_3)\\ &y_{23}:\ y_{33}=(1-x_2):(1-x_3) \end{aligned}\right\} \quad (5.3)$$

この連立方程式から x_1, x_2, x_3 の値が求まる．しかしこの方程式は非線型の高次方程式となり，一般に解けない．そこで次のような繰返し計算による逐次近似法を用いた．

Step 1: (5.1) 式をみたす y_{11}〜y_{33} の組を設定する（シミュレータでは各パス共通に $y_{11}=y_{21}=a_1/2$, $y_{12}=y_{22}=y_{32}=a_2/3$, $y_{23}=y_{33}=a_3/2$ としている）．

Step 2: (5.2)式より x_1, x_2, x_3 を求める．

Step 3: (5.3)式と(5.1)式により，次の y_{11}〜y_{33} を求める．

Step 4: Step 2 と Step 3 を連続する2回の値の差が必要精度以内になるまで繰り返す．

6. 適用結果

6.1 適用条件

Hybrid 型シミュレーションでは，数ミリ秒間隔で発生する割込み動作や入出力動作といった短時間事象は解析計算で，ジョブの計算機システムへの投入・離脱といった長時間事象はシミュレーションで実現される．すなわち，短時間事象と長時間事象を異なる形態で実現するいわゆる分解手法による．1回のジョブの投入・離脱の間は解析モデル部のジョブの多重度は不変なので，短時間事象の解析計算には Closed Queueing Network Model を用いる．したがって，長時間事象の発生の間に短時間事象が頻繁に発生し，この間に短時間事象がより定常状態に達しているほど，近似度が高いとされている[8),9)]．そこで，Hybrid 型シミュレータの有効性を分解手法という観点から検証するために，その効率と精度を，短時間事象もシミュレーション手法でモデル化した詳細シミュレーションとの比較で検討する．

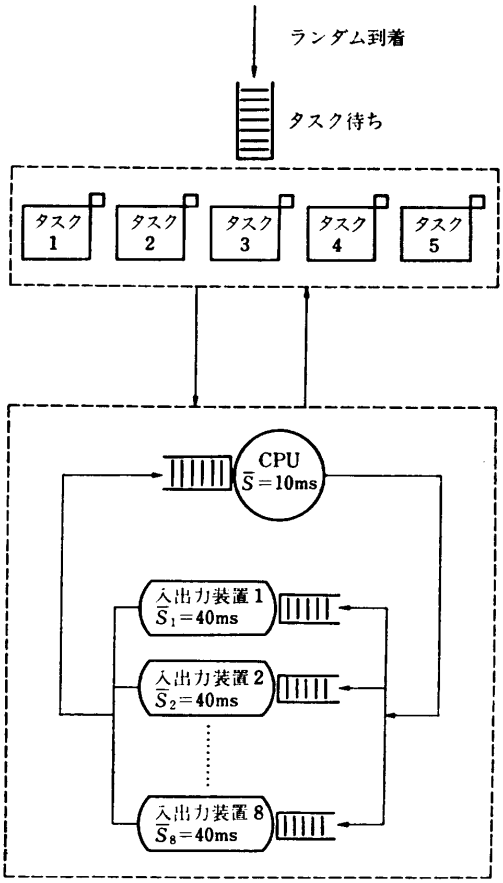

図 4 Hybrid 型シミュレーションの適用例
Fig. 4 An example of hybrid-type simulation model.

検討したモデルの構成を**図4**に示す．ジョブはランダムにシステムに到着し，計算機処理を要求する．この処理は一つのタスク上で行われるため，ジョブはまずタスク待ち行列につながれる．計算機処理に入ったジョブは必要回数だけ CPU 処理と入出力処理を繰り返し，処理が完了するとタスクを解放して計算機システムから退去する．これを Hybrid 型シミュレーションで実現すると，ジョブの発生・消滅，解析モデルへの登録・離脱はシミュレーション部で，計算機処理は解析モデル部でモデル化する．解析モデルは CQNM であるから，CPU 処理を完了したジョブが次に処理を受けるべき入出力装置は確率的に決定される．したがって，一つのジョブが Queueing Network 内を回るサイクル数（1サイクルは，1回の CPU 処理と入出力処理が終了して再び CPU 処理に向かうまでとする）は，一般に幾何分布に従う．そこで詳細シミュレーションでも，サイクル数は幾何分布に従うとする

(これを詳細シミュレーションAとする)．1ジョブの平均サイクル数はHybrid型シミュレーションならびに詳細シミュレーションのそれぞれについて，10，100，1000サイクルの3通りである．1回の平均CPU時間はすべてのケースについて10ミリ秒，平均入出力アクセス時間は40ミリ秒とし，これらのサービス形態はHybrid型シミュレータの解析モデル部に合わせて指数サービスとする．そしてジョブの到着トラフィックは，いずれのサイクル数の場合も，CPUならびに入出力装置の利用率がそれぞれ約50%，25%となるように設定した．最大タスク多重度は，過小であるとタスクネックとなってシミュレーションは定常状態に達せず，また過大に設定してもそこまで多重度は上がらず無意味である．本シミュレーションでは最大タスク多重度を5としたが，この値はシミュレーションの定常状態を得る上で適正であり，かつ得られる結果も，これより大きい値による結果をほぼ代表する値である．

6.2 シミュレーション結果

シミュレーションは，いずれのケースともにSingle Run Method[7]により各runにつき6個の結果を出力した．このうち，最初の1個を除いた5個の平均値と，t-分布による95%信頼区間を示したものが**表2**である．

まず，シミュレーション効率の指標となる入力ジョブ千件当りのシミュレーション実行時間についてみると，Hybrid型シミュレーションではサイクル数に関係なくほぼ一定である．これは，解析計算は1ジョブ当り投入・離脱時に1回ずつ計2回行えばよく，しかもそのときのジョブ多重度がほぼ一定のためである．一方，詳細シミュレーションの実行時間はサイクル数にほぼ比例して増加している．これは，各サイクルを1回ごとにすべてシミュレートするためである．一般に1ジョブの入出力アクセス回数は最小でも数十回以上のことが普通であり，これが多いほどHybrid化による効率向上の効果は大きいことがわかる．

CPU利用率は各ケースごとに両シミュレーションの結果はよく一致しており，いずれもほぼ定常状態に達していることがわかる．

次にジョブの応答時間をみると，各ケースともに詳細シミュレーションがHybrid型シミュレータより13～29%大きい値である．この相違率は，1ジョブのサイクル数が大きいほど小さくなっており，文献8)，9)の結論と同傾向を示している．この相違の原因を調べるために，応答時間をタスク処理時間とタスク待ち時間に分けて考える．すると応答時間の差は，詳細シミュレーションのタスク待ち時間が大きいことに起因していることがわかる．詳細シミュレーションでは，各ジョブのサイクル数が幾何分布に従うとしているため，タスク処理時間の標準偏差は大きい．一方，Hybrid型シミュレーションではタスク処理時間は定常解としてすべてのジョブに一律に与えられるから，

表2 Hybrid型シミュレーションと詳細シミュレーションAの結果
Table 2 Results of hybrid-type simulation and detailed simulation A.
[upper row: hybrid-type simulation / lower row: detailed simulation A]

項目	サイクル数 10	100	1000
シミュレーション実行時間 (秒/千件)	6.63 8.72	6.77 82.6	7.33 784.0
平均タスク多重度 (多重)	3.11±0.08 3.17±0.02	3.43±0.23 3.50±0.20	3.46±0.16 3.47±0.10
CPU利用率 (%)	48.6±0.9 50.1±0.3	49.5±2.5 51.1±2.1	50.2±1.7 52.3±2.1
平均応答時間 (R_H) (秒) (R_A)	7.1±0.2 10.0±0.1	78.3± 3.1 98.4±11.4	789±18 911±39
平均タスク処理時間 (秒)	6.2±0.1 6.3±0.1	68.2±0.7 69.7±1.9	684± 6 694±10
平均タスク待ち時間 (秒)	0.9±0.1 3.7±0.1	10.0± 2.4 28.1±10.2	104±12 217±29
タスク処理時間の標準偏差 (秒)	0.6±0.1 6.9±0.1	6.4±0.2 69.8±3.5	65± 3 695± 9
応答時間の相違率 $\frac{R_A-R_H}{R_A}\times 100(\%)$	29.0	20.4	13.4

表3 Hybrid型シミュレーションと詳細シミュレーションBの結果
Table 3 Results of hybrid-type simulation and detailed simulation B.
[upper row: hybrid-type simulation / lower row: detailed simulation B]

項目	サイクル数 10	100	1000
平均応答時間 (秒)	7.1±0.2 7.0±0.1	78.3±3.1 77.4±2.7	789±18 786±17
平均タスク処理時間 (秒)	6.2±0.1 6.2±0.1	68.2±0.7 67.7±0.7	684± 6 684± 6
平均タスク待ち時間 (秒)	0.9±0.1 0.8±0.1	10.0±2.4 9.6±2.1	104±12 103±11
タスク処理時間の標準偏差 (秒)	0.6±0.1 1.7±0.1	6.4±0.2 8.2±0.1	65± 3 67± 3

標準偏差は小さい．したがって，タスク待ち時間は詳細シミュレーションがより大きい値となる．

ところで，現実のジョブの動作における入出力アクセス回数は，あらかじめほぼ決定されている場合が多く，この場合のサイクル数は定数となる．そこで第2の詳細シミュレーションとして，サイクル数が定数の場合を考える（これを詳細シミュレーションBとする）．すると詳細シミュレーションBにおけるタスク処理時間の変動要因は，サイクル数が定数のためCPU 処理時間と入出力処理時間が指数分布することのみであり，その標準偏差はAより小さい．そして**表3**によるとHybrid型シミュレーションに近い値である．同様に，詳細シミュレーションBとHybrid型シミュレータのタスク待ち時間・応答時間もよく一致している．

詳細シミュレーションAは，Hybrid型シミュレーションの解析モデルにおいてサイクル数が確率的に決まるという性質を最も忠実に再現している．しかし，Hybrid型シミュレーションにおいて分解手法を用いた結果，タスク処理時間は一律にその平均値が与えられるため，サイクル数が確率的に決まることの効果はシミュレーションにはまったく反映されない．したがってHybrid型シミュレーションはむしろBに近い結果となる．すなわちHybrid型シミュレーションは，現実のジョブの動きにより近い詳細シミュレーションBをよく近似している．

7. むすび

Closed Queueing Network Model による解析的手法とシミュレーション手法を組み合わせた計算機性能評価用のHybrid型シミュレータを開発した．これは割込み動作や入出力動作を解析的手法による数値計算に置き換えることにより計算時間と開発労力を削減し，計算機外部の事象の設定にはシミュレーション手法を用いることで，モデル化の自由度を維持する．解析計算にはMean Value Analysisを用い，これにより複数ジョブ・クラスを扱いうる．また全体をシミュレーション言語化して記述性・汎用性の向上を図った．モデルは，ホスト言語であるPL/I命令にシミュレーション命令を挿入する形で記述し，チャネルや制御装置の接続構成を含めた計算機システムを，Hybrid手法で容易にモデル化可能である．

本Hybrid型シミュレータの効率と精度を詳細シミュレーションと比較検討した．その結果，通常の計算機システムをモデル化する場合は，Hybrid化により計算時間と開発労力を削減できることを確認した．またシミュレーションの精度は，ジョブのサイクル数が大きいほど詳細シミュレーションとの相違率は小さく，文献8),9)の結論が確認された．そして相違の主要因は，Hybrid化に伴いジョブの計算機内部処理時間分布が変形されることにあることが判明した．今後は適用の経験を積む中で改善を図っていく予定である．

謝辞 最後に，本研究についてご指導いただいた電気通信大学亀田壽夫助教授，ならびに本研究の機会を与えて下さった当社システム開発研究所三浦武雄所長，同社ソフトウェア工場高須昭輔部長，の諸氏に深く感謝いたします．

参考文献

1) Schwetman, H. D.: Hybrid Simulation Models of Computer Systems, *Comm. ACM*, Vol. 21, No. 9, pp. 718-723 (1978).
2) Chiw, W. W. and Chow, W.: A Performance Model of MVS, *IBM Syst. J.*, Vol. 17, No. 4, pp. 444-462 (1978).
3) Baskett, F., Chandy, K. M., Muntz, R. R. and Palacious, F. G.: Open, Closed, and Mixed Networks of Queues with Different Classes of Customers, *J. ACM*, Vol. 22, No. 2, pp. 248-260 (1975).
4) Reiser, M. and Lavenberg, S. S.: Mean-Value Analysis of Closed Multichain Queueing Networks, *J. ACM*, Vol. 27, No. 2, pp. 313-322 (1980).
5) 日立製作所：Mシリーズ統合ディスク制御装置，ハードウェア・マニュアル，8080-2-012 (1976).
6) IBM: Computer Systems Simulator II (CSS/II), Program Discription and Operations Manual (SH 20-0875-1) (Mar. 1971).
7) Kobayashi, H.: *Modeling and Analysis*, p. 446, Addison-Wesley Publishing, Reading (Mass.) (1978).
8) Courtois, P. J.: Decomposability, Instabilities, and Saturation in Multiprogramming Systems, *Comm. ACM*, Vol. 18, No. 7, pp. 371-377(1975).
9) Courtois, P. J.: *Decomposability*, p. 201, Academic Press, New York (1977).
10) 木下，吉沢：解析手法を取入れた計算機性能評価シミュレータの開発，第22回情報処理全国大会論文集，pp. 219-220 (1981).
11) 本山，大町：分散型計算機システム性能予測ツールの開発，情報処理学会計算機システムの解析と制御研究会資料 13-3 (1981. 6).

付 録

4章の定理を証明する．ただし記号は次のとおり．

S：ジョブ・クラス数

L：Queueing Network 中のサーバ数

$n_s(s=1,2,\cdots,S)$：ジョブ・クラス s のジョブ数

N：全ジョブ数 $(=\sum_{s=1}^{S} n_s)$

$\tau_{ls}(l=1,2,\cdots,L\ ;\ s=1,2,\cdots,S)$：ジョブ・クラス s のジョブがサーバ l に要求する総サービス時間

$j_{si}(s=1,2,\cdots,S\ ;\ i=1,2,\cdots,n_s)$：ジョブ・クラス s の各ジョブ

$r_{si}(s=1,2,\cdots,S\ ;\ i=1,2,\cdots,n_s)$：ジョブ j_{si} の処理残余率

$G(L,n_1,n_2,\cdots,n_S\ ;\ \tau_{ls})$：変数が τ_{ls} の正規化定数．

また $\boldsymbol{n}=(n_{ls})_{s=1,2,\cdots,S}{}^{l=1,2,\cdots,L}$ を $L\times S$ 次整数ベクトルとし，

$$F(L,S)=\{\boldsymbol{n}\ ;\ n_{ls}\geqq 0,\ \sum_{l=1}^{L} n_{ls}=n_s\}$$

とおくと，$F(L,S)$ 中の各ベクトルは対象とする CQNM の一つの状態を表す．そこで残余率を考慮せず初期の要求処理時間に基づく場合，FCFS, LCFS, PS スケジュールに関する正規化定数は，

$$G(L,n_1,\cdots,n_s,\cdots,n_S\ ;\ \tau_{ls})$$
$$=\sum_{\boldsymbol{n}\in F(L,S)}\prod_{l=1}^{L}\left\{\frac{\left(\sum_{s=1}^{S} n_{ls}\right)!}{\prod_{s=1}^{S} n_{ls}!}\cdot\prod_{s=1}^{S}\tau_{ls}{}^{n_{ls}}\right\}$$

と表される．そしてジョブ・クラス s のジョブの応答時間 T_s は，

$$T_s=n_s\times\frac{G(L,n_1,\cdots,n_s,\cdots,n_S\ ;\ \tau_{ls})}{G(L,n_1,\cdots,n_s-1,\cdots,n_S\ ;\ \tau_{ls})}$$

である．一方，各ジョブの残余率を考慮し，すべてのジョブが異なるジョブ・クラスに属すると仮定した場合は，

$$\boldsymbol{n}'=(n'_{lsi})_{s=1,2,\cdots,S\ i=1,2,\cdots,n_s}^{l=1,2,\cdots,L}\ :\ L\times N\ \text{次整数ベクトル}$$

$$F'(L,N)=\{\boldsymbol{n}'\ ;\ n'_{lsi}=0\ \text{or}\ 1,\ \sum_{l=1}^{L} n'_{lsi}=1\}$$

とおくとき，

$$G(L,\overbrace{1,\cdots,1}^{N\text{個}}\ ;\ r_{si}\tau_{ls})$$
$$=\sum_{\boldsymbol{n}'\in F'(L,N)}\prod_{l=1}^{L}\{(\sum_{s=1}^{S}\sum_{i=1}^{n_s} n'_{lsi})!\cdot\prod_{s=1}^{S}\prod_{i=1}^{n_s}(r_{si}\tau_{ls})^{n'_{lsi}}\}$$
$$=\sum_{\boldsymbol{n}'\in F'(L,N)}\left[\prod_{s=1}^{S}\prod_{i=1}^{n_s} r_{si}{}^{\sum_{l=1}^{L} n'_{lsi}}\times\prod_{l=1}^{L}\{(\sum_{s=1}^{S}\sum_{i=1}^{n_s} n'_{lsi})!\cdot\prod_{s=1}^{S}\tau_{ls}{}^{\sum_{i=1}^{n_s} n'_{lsi}}\}\right]$$

ここで仮定により $\sum_{l=1}^{L} n'_{lsi}=1$．また和 $\sum_{\boldsymbol{n}'\in F'(L,N)}$ のうちで $\sum_{i=1}^{n_s} n'_{lsi}=n_{ls}$ となる項は $\prod_{s=1}^{S}(n_s!/\prod_{l=1}^{L} n_{ls}!)$ 通りあるが，これらについては $\prod_{s=1}^{S}\tau_{ls}{}^{\sum_{i=1}^{n_s} n'_{lsi}}=\prod_{s=1}^{S}\tau_{ls}{}^{n_{ls}}$ となるから，

$$G(L,1,\cdots,1\ ;\ r_{si}\tau_{ls})$$
$$=\prod_{s=1}^{S}\prod_{i=1}^{n_s} r_{si}\times\sum_{\boldsymbol{n}\in F(L,S)}\left[\prod_{s=1}^{S}\frac{n_s!}{\prod_{l=1}^{L} n_{ls}!}\cdot\prod_{l=1}^{L}\{(\sum_{s=1}^{S} n_{ls})!\cdot\prod_{s=1}^{S}\tau_{ls}{}^{n_{ls}}\}\right]$$
$$=\prod_{s=1}^{S} n_s!\times\prod_{s=1}^{S}\prod_{i=1}^{n_s} r_{si}\times\sum_{\boldsymbol{n}\in F(L,S)}\left\{\prod_{l=1}^{L}\frac{(\sum_{s=1}^{S} n_{ls})!}{\prod_{s=1}^{S} n_{ls}!}\cdot\prod_{s=1}^{S}\tau_{ls}{}^{n_{ls}}\right\}$$
$$=\prod_{s=1}^{S} n_s!\times\prod_{s=1}^{S}\prod_{i=1}^{n_s} r_{si}\times G(L,n_1,\cdots,n_s,\cdots,n_S\ ;\ \tau_{ls})$$

したがってこの場合のジョブ j_{si} の残余応答時間 T_{si} は，

$$T_{si}=1\times\frac{G(L,1,\cdots,1,\cdots,1\ ;\ r_{si}\tau_{ls})}{G(L,1,\cdots,0,\cdots,1\ ;\ r_{si}\tau_{ls})}$$

（ただし分母はジョブ j_{si} に対応する変数のみが0で，他は1）

$$=n_s\times r_{si}\times\frac{G(L,n_1,\cdots,n_s,\cdots,n_S\ ;\ \tau_{ls})}{G(L,n_1,\cdots,n_s-1,\cdots,n_S\ ;\ \tau_{ls})}$$
$$=r_{si}\times T_s \qquad \text{Q. E. D.}$$

（昭和56年11月5日受付）
（昭和57年10月4日採録）