分散処理システム評価シミュレータ SEDS†

山口 英†† 下條真司†† 宮原秀夫††

分散処理システムをハードウェア・ソフトウェア両面から総合的に性能評価を行うシミュレータ SEDS (Simulator for Evaluation of Distributed Systems) を開発した. SEDS では, 対象とする分散処理システムを階層的にとらえモデル化し, Form と呼ぶ簡単な書式で記述し入力することができる. SEDS は3つのプログラムから構成され, 分散処理システムの性能評価のための環境を作り出す. 本論文では, SEDS の特徴, Form の詳細と SEDS の構造について詳しく述べ, さらに SEDS を実際の分散処理システムに適用した例としてジョブディスパッチ問題に適用した例を示す.

1. はじめに

近年の VLSI 技術の進歩により, 安価で信頼性の高いプロセッサが大量に供給されるようになっている. このような状況の中で処理能力の向上を図るための一手段として分散処理が注目されており, 各種の分散処理システムの開発や分散型アルゴリズムに関する研究が行われている. 一方, 分散処理システムの性能は複数の処理要素の結合形態, 処理要素における OS のプロセス処理方式, 処理要素間の通信方式, 使用するアルゴリズム, 処理の並列化と分散方法等によって大きく変化する. すなわち, 分散処理システムの性能を左右する要因はハードウェアからソフトウェアにわたる広い領域に混在しており, 分散処理システムの開発・評価をする場合には, これらの各種の要因を総合的に評価する必要がある. しかしながら, 従来のシミュレータ, 特に待ち行列理論に基づいたシミュレータにおいてはモデルが複雑になり過ぎるためこれらの評価を行うことは困難であり, 分散処理システムの性能評価を重点においた新たなシミュレータの開発が必要である[1),2)]. そこで筆者らはこのことを目的として, 分散処理システム評価シミュレータ SEDS (Simulator for Evaluation of Distributed Systems) を開発した[3),4)].

2. SEDS の特徴

SEDS は, 次のような特徴を持つ.

●ハードウェア・ソフトウェア両面からの評価可能

SEDS は分散処理システムのハードウェア, あるいは, ソフトウェアのみを単独に評価するものではなく, 両者を合わせて評価するシミュレータである. SEDS はこれまで開発された論理シミュレータと異なり, 分散処理システムのハードウェア・ソフトウェアに対して明確なモデルの枠組みを提供している. そのため, 対象とするシステムの構成やその上に実装されるソフトウェアの変更を容易に行うことが可能である.

●分散型シミュレータ

SEDS は UNIX の下で開発され, UNIX のプロセス管理機能とプロセス間通信機能を用いて複数のプロセスが協調して動くマルチプロセスシミュレータとして構成されている. したがって, 適当な分散処理環境があれば, 分散型シミュレータとして SEDS 自体を複数の計算機上に拡張可能である. また, シミュレータを構成する各プロセスは階層化されており, 変更・保守が行いやすい.

●Form によるモデルの記述

シミュレーションを行う際, 使用者は SEDS に, シミュレーション対象となるシステムのハードウェア構成とそのシステム上で実行されるソフトウェアのモデルの記述を与えなければならない. 待ち行列理論に基づいたシミュレータの場合, 待ち行列網を入力として与えるが, 本シミュレータではモデルの簡単な記述と, 対象システムのより直接的なモデル化, 効率的な管理を実現するために, Form (後述) を用いたモデルの記述法を導入している. 本シミュレータでは分散処理システムのモデルの枠組みを明確に Form の形式で提供しており, 利用者は Form の必要な部分を指定するだけでモデルの記述を行うことが可能である. SEDS はモデルを記述した Form を入力として受け, 分散処理システムに関する種々の評価測度を出力する.

●シミュレーション実行環境 (SEEDS)

SEDS はシミュレーションにまつわる一連の作業を

† SEDS—Simulator for Evaluation of Distributed Systems by SUGURU YAMAGUCHI, SHINJI SHIMOJO and HIDEO MIYAHARA (Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University).

†† 大阪大学基礎工学部情報工学科

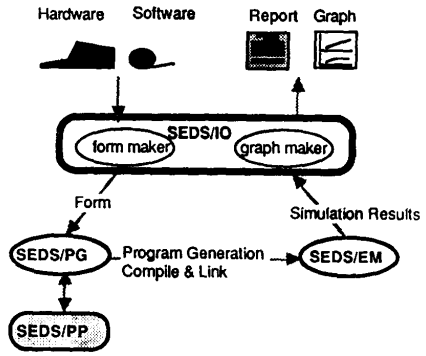


図 1 分散処理システムの性能評価環境
Fig. 1 Simulation environment for evaluation of distributed systems.

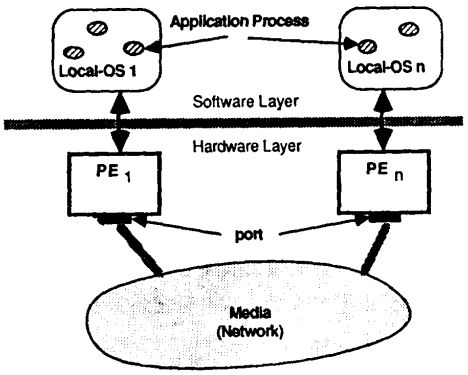


図 2 分散処理システムのモデル
Fig. 2 Model of distributed systems.

複数のプログラム群で分担して行う．これらは簡易入力プログラムとデータ解析ツール (SEDS/IO)，プログラムジェネレータ (SEDS/PG)，実行プログラム (SEDS/EM) からなり，モデルの作成から，シミュレーションの実行，実行結果の管理までを使用者が容易に行えるような統合化された環境 SEEDS (Simulation Environment for Evaluation of Distributed Systems) を提供している (図 1)．

3. 分散処理システムのモデル化

3.1 分散処理システムの各階層

SEDS ではシミュレーション対象である各種の分散処理システムを次のように階層化してモデル化を行う (図 2)．

●ハードウェア階層

分散処理システムを構成するプロセッサ等のハードウェア資源と，それらの結合関係 (通信網の形態) を表す階層．SEDS において分散処理システムは複数の処理要素 (PE : Processing Element) とそれらを結合する通信網 (Media) から構成されると考える．各 PE は他の PE との間に共有メモリを持たず，各々ローカルメモリのみを持ち，ポート (Port) を介して通信網 (Media) に結合されている．この階層では，PE の個数，通信網の形態，通信網における遅延，PE と通信網との接続関係等を記述する．

●ローカル OS 階層

各 PE に存在するハードウェアに依存した基本ソフトウェアのモデルを記述した階層であり，プロセスの実行制御やプロセス間通信の管理を行うオペレーティングシステム (以下ローカル OS と呼ぶ) のモデルに相当する．この階層ではローカル OS のプロセス処理方式 (例えば，ラウンド・ロビン等) を記述する．ローカル OS の下では 1 つ以上のプロセスが割当可能で，1 つの PE に割り当てられた複数のプロセスはそのローカル OS の処理方式にしたがって処理される．

●アプリケーションプロセス階層

ローカル OS のもとで実行される応用ソフトウェア，つまり分散処理システム上で実行される 1 つのアプリケーションのモデルを指す．SEDS でのアプリケーションとして，CSP (Communicating Sequential Processes)[5] のような通信するプロセスモデルで表現されるものを考える．つまり，1 つのアプリケーションは幾つかの並列処理可能な部分処理 (以下プロセスと呼ぶ) の集合で構成され，各プロセスは他のプロセスと共有データを持たず，プロセスの同期やデータの授受はプロセス間通信によるメッセージの交換によって行われる．各プロセスは実行前に各々適当な PE に割り当てられ，並列に処理される．各プロセスは，PE の CPU を使用して行われるローカルな処理，メッセージの送信，受信 (以下 primitive operation と呼ぶ) の組み合わせによって記述され，1 つのプロセス内での処理は逐次的に進んでいく．プロセス間通信のうち，同じ PE 内のプロセス間通信 (以後内部プロセス間通信と呼ぶ) はローカルメモリを介して，異なった PE に割り当てられたプロセス間通信 (以後外部プロセス間通信と呼ぶ) は通信網 (Media) を介して行われる．

アプリケーションプロセス階層で採用しているモデルは CSP を基礎としており，SEDS で扱うことのできるアプリケーションは CSP で記述可能なものに制限される．

3.2 Primitive operation

SEDS では分散処理システムの評価測度（各プロセスの処理時間等）を得ることに興味がある．このため，SEDS では分散処理システムの性能に影響を与えない要素についてはできるだけ排除しており，PE でのローカルな処理時間とプロセス間通信に消費される時間のみに注目する．これらを primitive operation と呼び，次の3つの関数で表現し，これらの組み合わせで各種の分散型アルゴリズムを記述する．

exec (job_time):

アプリケーションプロセスが job_time に相当する時間だけ PE の CPU でローカルに実行されることを表す．

send (pid, message, type),

receive (pid, message, type):

アプリケーションプロセスがプロセス間通信を行うことを表す．任意のプロセスからの受信は pid に ANY を指定することで行うことができる．プロセス間通信によって授受されるメッセージは，その最大長がバイト数で指定されているのみで，その内容は使用者が構造を定義することにより自由に用いることができる．type によってプロセス間通信の種別を指定する．これには次の3種類が実現されている．

●同 期 型

プロセス間通信を行う時点で送受信両方のプロセスが待ち合わせを行い，同期をとることができる．

●非 同 期 型

送受信プロセスともに待ち合わせを行わない．送信されたメッセージはローカル OS により，いったん受信プロセスの内部のバッファに取り込まれる．受信プロセスでは receive ( ) が呼び出されたときにバッファ内に送信されたメッセージがある場合にはその先頭のものを取り出し，もしもない場合には待ち合わせを行わずに次の処理に移る．

●蓄 積 型

非同期型通信と同様の処理をするが，受信側プロセスで receive ( ) が呼び出された時にバッファにメッセージがない場合には，メッセージが送信されるまで，すなわち，バッファにメッセージが取り込まれるまで待たされることが非同期型通信と異なる．

4. Form によるモデルの記述

SEDS では Simulation Form, Local-OS Form, PE Form, Media Form, Process Form の5つの

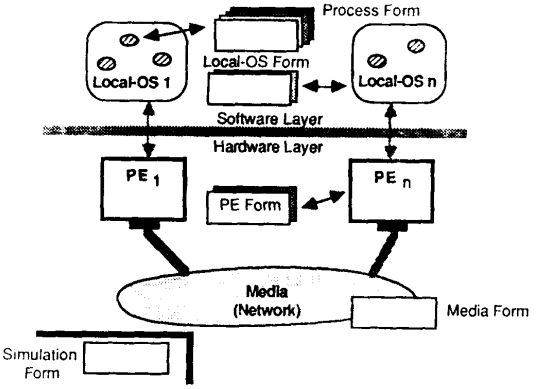


図 3 モデルと Form
Fig. 3 Forms and distributed systems.

Form を用意しており，モデルの記述と必要なパラメータの指定をモデルの階層と対応した形で行うことが可能である（図 3）．

4.1 Parameters

SEDS では Form の記述に従い，シミュレーション実行プログラムを生成し，コンパイルを行って実行形式にする．したがって，Form が変更された場合には，改めてシミュレーションプログラムを生成する必要がある．ところが，シミュレーション実験を行う場合には，幾つかのパラメータのみを変化させて同一のシミュレーション実行プログラムを繰り返し実行させることが多い．SEDS では Parameter と呼ばれる変数のクラスを用意しており，Local-OS Form, Process Form, Media Form で使用する変数を Parameter として宣言することができ，Parameter として宣言された変数は，PE Form においてその変数の値を設定することができる．この値は SEDS/PG によってデータファイルに切り出され，シミュレーションプログラム実行時に読み込まれる．SEDS/PG は Form の変更が Parameter 宣言された変数の値設定を変更しただけの場合にはシミュレーション実行プログラムの再生成は行わず，データファイルの更新のみを行う．したがって，シミュレーションごとに変化させる変数を Parameter 宣言しておけば，利用者はパラメータの変更のたびにシミュレーション実行プログラムの再生成を避けることができる．

4.2 各 Form の構成

●Simulation Form

Simulation Form はシミュレーションの実行の仕方を記述する Form である．SEDS ではシミュレーションの実行に2つの実行モードを用意しており，シミュレーション実行時における統計量の収集・出力のタ

イミングによって time モードと set モードに分けられている．time モードの場合，論理時刻を単位として統計量の収集を行い，統計量の収集を開始する論理時刻 start time，シミュレーションを終了する論理時刻 end time，および，評価測度を出力する間隔 Interval time を指定する．一方，set モードの場合には，1つのアプリケーションの開始から終了，つまり，アプリケーションを構成するプロセスの起動からすべてのプロセスの終了までを1セットとし，セットごとに統計量の収集を行う．Interval Set で指定したセットごとにそれまでの統計量の平均を出力し，全実行セット数を Total Set で指定する．

●PE Form

PE Form は PE のモデルに関する記述を行う Form で，各 PE の名前，その PE に割り当てられるアプリケーションプロセス，その PE におけるローカル OS の処理方式を指定する．また，この Form では，その PE が持つポートと通信網 (Media) との接続形態を指定することで，システムのネットワークの構造を記述する．また，Process Form などで Parameter 宣言された変数の値の設定をこの Form で行う．

●Local-OS Form, Media Form

Local-OS Form は，ローカル OS のモデルを，Media Form は通信網のモデルを記述する．ただし，幾つかの代表的な OS のプロセス処理方式と通信網はあらかじめモデル化した Form が用意されており，使用者はそれらの中から適当なものを選択するだけでよい．Local-OS Form ではプロセスの処理方式に関するパラメータが，Media Form では通信網の特性を決めるパラメータ（例えば遅延時間の分布）がそれぞれ設定されており，これらは Parameter の機構を用いて PE Form から与えられる．

●Process Form

Process Form では分散処理システム上で実行する1つのアプリケーションのモデルを記述する．この Form は，Message 部，Statistics 部，Parameter 部，および，Program 部からなる．Message 部ではアプリケーション・プロセス間でやりとりするメッセージの構造を定義する．Statistics 部では標準の評価測度が出力されるときに同時に出力される変数を指定できる．利用者が定義した評価測度をここで指定した変数に格納することで，使用者が独自の評価測度を出力することが可能となる．Parameter 部は，Parameter のクラスに属する変数を指定する．Program 部では，アプリケーションのモデルを記述する．記述には Primitive Operation を表す関数と，C言語の構文を用いて直接記述する．

5. SEDS/EM の構造

5.1 シミュレーション実行プログラムの構成

SEDS は分散処理システムを対象としており，プロセス間での通信や，PE でのプロセスの実行などの不定期に発生するイベントを扱う必要があるため，イベント駆動型シミュレータとして構成する．このためシミュレータを構成する各プロセスは，他のプロセスとイベントをやりとりしながら独立に処理を進めるイベント駆動型のプロセスであり，各プロセスはすべて独自の論理時刻を持ち，その管理を行う．以後，SEDS/EM を構成するプロセスをモデルの各階層と対応して，アプリケーションプロセス，ローカル OS プロセス，アーキテクチャプロセスと呼ぶ．

SEDS が扱う分散処理システムのモデルから考えると，1つの PE に割り当てられている複数のアプリケーションプロセスとその PE のローカル OS は，同期した論理時刻を持つ必要がある．一方，外部プロセス間通信が行われる場合は通信は異なるローカル OS プロセス間で行われ，時刻の同期が取られることになる．この同期はローカル OS 間の通信を管理するプロセスであるアーキテクチャプロセスで行われる．これらの同期はイベントをプロセス間でやりとりすることによって行われる．このため，SEDS のシミュレーション実行プロセスは図4に示されるような木構造をとる．

5.2 イベントの流れ

SEDS で扱われるイベントとしては Primitive Operation に対応した3種類のイベント，send イベント，receive イベント，exec イベントがある．これらのイベントはすべて Primitive Operation を実行

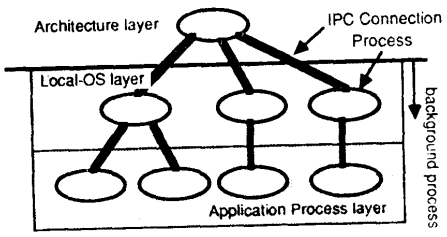


図 4 SEDS/EM の構造
Fig. 4 Layer structure of SEDS/EM.

```
typedef strcut event{
  int       source;          /* Process ID */
  int       destination;     /* Destination Process: event is IPC */
  int       request;         /* event type */
  int       option;          /* IPC option BLOCKING/NON */
  int       portid;          /* port ID */
  double    sc;              /* event generated time */
  double    exe_time;        /* CPU time */
  char      user_message[256]; /* user message */
} Event;
```

図 5 イベントの構造体
Fig. 5 Event structure.

したときにアプリケーションプロセスで発生する．イベントは固定長の構造体として表現され，イベントの発生した論理時刻，イベントを発生したプロセスの識別子，イベントの種類などの情報が含まれている．さらに，send, receive イベントの場合には，通信されるメッセージ内容とその送信先，exec イベントの場合には PE での処理時間を示す情報が含まれている(図 5)．

各プロセスは，受け取ったイベントの中から処理するイベントを選択し，イベントの処理にかかる論理時間を計算し，それに基づいてプロセスの論理時刻を更新し，処理結果をイベントとして他のプロセスに送る．

exec イベントはローカル OS プロセスに送られ，そこで処理される．send イベント，receive イベントは，内部プロセス間通信の場合には，ローカル OS プロセスに送られて処理され，外部プロセス間通信の場合には，ローカル OS プロセスを経由してアーキテクチャプロセスに送られてそこで処理される．

5.3 論理時刻の管理

SEDS/EM では各アプリケーションプロセスは並列に処理を進めるので複数のプロセスからそれぞれ異なった論理時刻に発生したイベントがローカル OS プロセスやアーキテクチャプロセスに集まってくる．したがって，ローカル OS プロセスやアーキテクチャプロセスでイベントを処理する場合には，イベントはその発生した論理時刻の順に処理される必要がある．このために，アーキテクチャプロセスではすべてのローカル OS プロセスからの，また，ローカル OS プロセスではそれが管理するすべてのアプリケーションプロセスからのイベントが到着するのを待ち，その中でもっとも早い時刻に発生したイベントから処理する．

5.4 通信イベントの処理

send イベントと receive イベントの処理は，その通信の種類（同期型，非同期型，蓄積型）によって異なる．これらのイベントは種類ごとに次のように処理される．

●同 期 型

通信を行うプロセスは共に待ち合わせを行う．このため送信イベントや受信イベントが処理される場合には，それぞれ対応する受信または送信イベントがすでに到着しているかを調べる．もしも対応する送受信イベントが到着していない場合は，イベントの処理は延期される．一方，対応した送受信イベントが存在するときには，それらのイベントの組が処理され，送信イベントに含まれているメッセージを受信イベントに写し，論理時刻を更新し，それらのイベントを発生したプロセスに結果を送り返し，処理が完了する．

●非 同 期 型

通信を行うプロセスは共に待ち合わせを行わない．したがって，送信イベントの場合には，そのイベントは発生と同時に対応する受信イベントの発生を待つことなしに処理が完了し，プロセスの論理時刻は通信が終了した論理時刻に更新される．この場合，後で対応する受信イベントを処理するために，処理された送信イベントの写しを記録しておく．一方，受信プロセスも待ち合わせを行わないので，受信イベントが発生と同時に処理が完了する．この場合は，処理済みの送信イベントの記録を検索して，対応する送信プロセスからの送信イベントがあれば，送信イベントに含まれているメッセージを写し，受信イベントを発生したプロセスにそれを送り返す．この使用した送信イベントの記録は削除する．対応する送信プロセスからのイベントがない場合には，通信が失敗したことをイベントを発生したプロセスに送り返す．

●蓄 積 型

蓄積型の場合，送信プロセスは待ち合わせを行わないが，受信プロセスは待ち合わせを行う．したがって，送信イベントの場合は非同期型の時と同様に発生と同時に処理され，処理された送信イベントを記録する．受信イベントの場合には，非同期型の時と同じように処理されるが，対応する送信プロセスからの送信イベントがない場合には，同期型の場合と同じように指定した送信先からの送信イベントが処理されるまで処理が延期される点が異なる．

5.5 各階層のプロセスの構造

各階層のプロセスは図 6 のような共通の構造を持っており，通信管理部分，シミュレーション管理部分からなっている．通信管理部分は他のプロセスとのプロセス間通信を行う部分で，ここを通じてイベントがプ

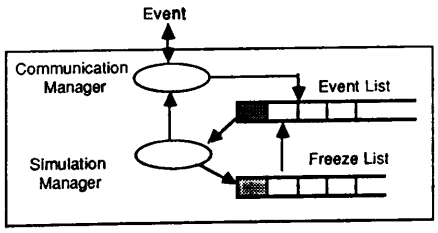


図 6 SEDS/EM のプロセスの構造
Fig. 6 SEDS/EM process structure.

ロセス間でやりとりされる．シミュレーション管理部分は主としてイベントの処理や待ち合わせ，論理時刻の管理などを行うが，その内容は各階層によって次のように異なっている．

●アプリケーションプロセス

アプリケーションプロセスのシミュレーション管理部分では Form の記述にしたがって Primitive Operation に相当する関数を順に実行し，イベントを発生する．発生したイベントは通信管理部分を通してローカル OS プロセスに送り，処理結果が送られてくるのを待つ．処理結果が送られてくると，イベントの処理に消費された論理時間を計算し，これをもとに，論理時刻を更新する．

●ローカル OS プロセス・アーキテクチャプロセス

ローカル OS プロセス，および，アーキテクチャプロセスのシミュレーション管理部分では，イベントリストを用いてイベントの処理の管理を行う．

通信管理部分によって受信されたイベントはシミュレーション処理部分に渡され，イベントリストに加えられる．イベントリスト内のイベントは常に発生した論理時刻の早い順に並べ換えられている．

シミュレーション処理部分はイベントリストの先頭にあるイベント，すなわち発生した論理時刻がもっとも早いイベントに対して，そのイベントの種類に応じた処理を行い，それに基づいてプロセスの現在の論理時刻を更新する．さらに通信イベントのうち，受信イベントで，蓄積型のイベントの場合には，対応する送信イベントが処理されるまで，処理が行われない．この場合，受信イベントはイベントリストからフリーズリストに移され，対応するイベントが処理されるまで，この表の中に置かれる．対応する送信イベントが到着し，処理されたならば，その受信イベントはフリーズリストからイベントリストに移される．

処理の終了したイベントはイベントリストから削除され，結果をイベントの発生したプロセスに送り返す．通信管理部分では次のイベントの到着を待つ．

5.6 シミュレーション実行プログラムの実現

SEDS/EM は UNIX 上のマルチプロセスとして実現されている．通信管理部分で行うプロセス間通信としては基本的には UNIX が提供している pipe の機能を使用しているが，4.2BSD では socket, System V では message queue も使用できるように実現されている．また，シミュレータを構成する複数のプロセスの発生は，UNIX が提供しているプロセス生成のためのシステムコール fork( ), execl( ) を使用して実現している．

6. SEDS の具体的な問題への適用例

この章では SEDS をある種の画像処理マルチプロセッサシステムに用いた例を示す．

6.1 ジョブディスパッチ問題

ここで対象とした画像処理マルチプロセッサシステム[6]では，処理の高速化を行うため1枚の画面を複数のプロセッサに分割・分配し，各プロセッサが並列に処理を進める．このシステムは1台のディスパッチャ (dispatcher) と何台かのエクゼキュータ (executer) が通信路で結合されたもので，ディスパッチャに到着した画面処理要求を幾つかの並列可能な部分に分解し，それらをあいているエクゼキュータに順に割り当てて並列処理を行っている(図 7)．このようなシステムにおいて，1枚の画像を生成するための平均時間，すなわち，平均画面生成時間はすでに解析解が示されており[7]，ここでは SEDS を用いてその最適値を求め，解析結果と比較する．

6.2 モデル化

このシステムにおいて外部から与えられるパラメータを図8にまとめる．シミュレーションを行うにあたり，ディスパッチャおよびエクゼキュータの処理は1つのアプリケーションプロセスとして表し，それぞれ1つの PE に割り当てられ，各 PE のローカル OS はシングルタスクの処理を行う．このような定義をまとめ，Form の形で表したものを付録1に示す．

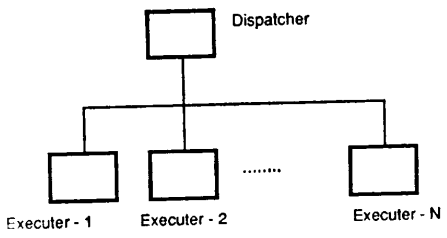


図 7 画像生成システムの構成
Fig. 7 Dispatcher/Executer system.

分割画面数	NJOBS = 10 .. 200
ディスパッチ時間	1/ud = 2,5
1つの分割画面の平均処理時間	1/ue = 10000 / NJOBS
エクゼキュータの個数	M = 4

図 8 D/E 問題のパラメータ
Fig. 8 Paramters of D/E problem.

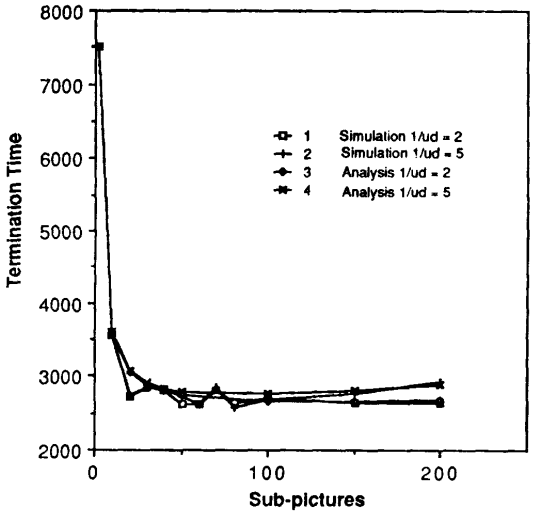


図 9 シミュレーション結果
Fig. 9 Simulation result.

付録1における記述では割り当てられるジョブ，すなわち，分割された1つの画面は，それを処理するのに必要な時間を記入したメッセージで表現する．ディスパッチャ (Process Form DISPATCHER) はそのパケットをエクゼキュータに送り，エクゼキュータ (Process Form EXECUTER 1 から Process Form EXECUTER 4) は受け取ったパケットに指定された時間だけ CPU 時間を消費し，その後，ディスパッチャに対して処理終了のパケットを送り次の画面の処理のためのパケットの受信を待つ．

一方，PE Form は，このシステムにおける PE の結合状態を Ports のエントリで，割り当てられるプロセスの指定を Processes のエントリで，また，そこでのローカル OS の処理を Local OS のエントリで指定する．また，ディスパッチャプロセスは Parameter クラスの変数を持つので，これらの変数の初期値を Process での Parameters のエントリで指定している．

6.3 シミュレーション結果

前節で説明したジョブディスパッチ問題について付録1の Form に基づきシミュレーションを実行した．実行結果を図 9 に示す．

7. おわりに

本稿では SEDS の構成，分散処理システムを記述するための書式である Form の詳細，および，その評価について述べた．現在 SEDS/EM, SEDS/PG の開発は終了しており，その他の部分は現在開発中である．今後は，Form における通信網の形態の記述を容易に行えるように，SEDS/IO にグラフィックインタフェースの機能を付加した新たな入力プログラムの設計・開発を進めて行くことが必要であると思われる．

参考文献

1) Fischer, W., Sauer, K. P. and Denzel, W.: A Simulation Technique for Distributed Systems Based on a Formal Specification by SDL, *Proc. of the International Seminar of Computer Network and Performance Evaluation*, 11-4 (1985).
2) 中井，前野，長谷川：ネットワーク論理シミュレータの一検討，第33回情報処理学会全国大会論文集，5U-8 (1986).
3) 山口，下條，宮原，高島：分散処理システム性能評価シミュレータ，信学技報情報ネットワーク研究会，IN 85-110 (1986).
4) 下條，山口，宮原，高島：分散処理システム評価シミュレータを用いた具体的問題の適用例，情報処理学会マルチメディア通信と分散処理研究会，29-6 (1986).
5) Hoare, C. A. R.: Communicating Sequential Processes, *Comm. ACM*, Vol. 21, No. 8, pp. 666-677 (1978).
6) 西村，出口，辰巳，河田，白河，大村：コンピュータグラフィックシステム LINKS-1 における並列処理の性能評価，信学会論文誌 (D)，Vol. J68-D, No. 4, pp. 733-740 (1985).
7) 下條，宮原：画像専用マルチプロセッサにおけるジョブディスパッチ問題，電気関係学会関西支部連合大会 (1986).

付録 1 Form によるジョブディスパッチ問題の記述

```
a) Simulation Form
/* ジョブディスパッチ問題のシミュレーション */
Simulation Form JOB DISPATCHER
    /* 実行モード set */
    Type of Simulation Run: set
        /* 40 セット実行して統計
                        量を出力して終了する */
        Total Set:      40
        Interval Sets:  40
```

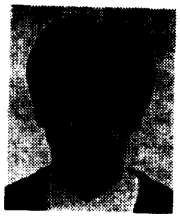
```
b) PE Form
/* ディスパッチャの記述 */
PE Form DISPATCHER
    /* Local OS の指定 */
    Local OS:    Round_Robin /* 標準 Form */
        parameters
          slice_time=9999
          /* シングルタスク処理のためスライス
                                時間は無限大 */
    /* 割り当てるプロセスの指定 */
    Processes:    DISPATCHER
        parameters
          TJOB=10000 /* 全ジョブ処理時間 */
          DJOB=5     /* プロセス割当時間 */
          NJOBS=100  /* 分割数 */
    /* Media との接続関係 */
    Ports:
        a:    BUS 1    /* Media 名 */
/* エクゼキュータの記述 1~4 */ (注)
PE Form PE 1
    Local OS:    Round_Robin
        /* シングルタスク処理 */
        prameters
          slice_time=9999
    Processes:    EXECUTER
    Ports:
        a:    BUS 1
          /* Media に対する parameters の指定 */
          parameters
          elapsed_time=0;

c) Process Form
/* ディスパッチャ・プロセスの記述 */
Process Form  DISPATCHER
    /* 開放するパラメータの宣言 */
    Parameters:
        double TJOB, DJOB, NJOBS;
    /* プロセス間で授受されるメッセージ */
    Message:
        int     no;
        double average;
    /* 実際の処理の記述 */
    Program:
dipatcher()
{
    PROCESS ex[5], find();
    MESSAGE *mes;
    double exp_rand();
    int     loop, i;
    /* エクゼキュータ情報 pid の取り出し */
    ex[1]=find("EXECUTER 1");
    ex[2]=find("EXECUTER 2");
    ex[3]=find("EXECUTER 3");
    ex[4]=find("EXECUTER 4");
    /* 1つのプロセスのプロセス実行時間の平均 */
    message->average=TJOB/NJOBS;
    /* 1つめのプロセスの割当 */
    for(i=1; i<=4; i++)
        send(ex[i], mes, BLOCKED);
    for(loop=4; loop<NJOBS; loop++){
        /* プロセス処理終了待ち */
        receive(ANY, mes, BLOCKED);
        /* プロセス割当のための処理 */
        exec(exp_rand(DJOB));
        /* 次のプロセスの割当 */
        send(ex[mes->no], mes, BLOCKED);
    }
    /* 各エクゼキュータからの最後の処理要求を
                                受け付ける */
    for(loop; loop.<4; loop++)
        receive(ANY, mes, BLOCKED);
    /* すべての処理の終了をエクゼ
                                キュータに知らせる */
    mes->no=999;        /* 999 が最後を表す. */
    for(i=1; i<=4; i++)
        send(ex[i], mes, BLOCKED);
}

/* エクゼキュータの処理の記述 1~4 */ (注)
Process Form  EXECUTER 1
    Program:
executer()
{
    PROCESS dispatcher;
    MESSAGE *mes;
    double exp_rand();
    /* ディスパッチャの pid の取り出し */
    dispatcher=find("DISPATCHER");
    while(TRUE){
        /* プロセスの受取 */
        receive(dispatcher, mes, BLOCKED);
        /* 全処理の終了 */
        if(mes->no==999)break;
        /* プロセスの処理 */
        exec(exp_rand(mes->average));
        /* 1つのプロセスの処理の終了を
                    ディスパッチャに知らせる. */
        /* 1はプロセス番号 */
        mes->no=1;
        send(dispatcher, mes, BLOCKED);
    }
}
```

注) エクゼキュータ2～4までの記述はエクゼキュータ1と同様であるため省略した.

(昭和62年12月11日受付)
(平成元年4月11日採録)



山口　英（正会員）

昭和61年大阪大学基礎工学部情報工学科卒業．現在，同大学院博士課程後期課程在学中．分散型アプリケーション，ネットワークプロトコルおよびセキュリティに関する研究に従事．

下條　真司（正会員）

昭和56年大阪大学基礎工学部情報工学科卒業．昭和61年同大学院博士課程修了．工学博士．同年大阪大学基礎工学部助手．平成元年同大学大型計算機センター講師，現在に至る．LAN のアクセス方式の性能評価，分散処理システムの性能評価，分散型 OS の研究に従事．

宮原　秀夫（正会員）

昭和42年大阪大学工学部通信工学科卒業．昭和47年同大学院博士課程修了．工学博士．昭和48年京都大学工学部助手．昭和55年大阪大学基礎工学部助教授．昭和62年大阪大学大型計算機センター教授，同大基礎工学部兼任教授，現在に至る．昭和58年から1年間 IBM トーマスワトソン研究所客員研究員．システムの性能評価，分散処理システム，LAN の研究に従事．