A User's Guide to

PMT - A Computer System Performance Modeling Tool

Richard Conway

TR 84-605
May 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

A User's Guide to

PMT - A Computer System Performance Modeling Tool

Richard Conway

This Performance Modeling Tool is a simulation system for overall throughput measurement of a computer system. It permits a schematic, high-level model of the system to be "run" so that its capacity to "process tasks" can be observed.

The technique employed in PMT -- discrete event simulation -- has long been used for this purpose, but has previously involved custom programming and large computers. The Performance Modeling Tool makes this capability available in quite a different form. It is an interactive, graphical system that runs on a desktop computer. It also does not require any conventional "programming". While it obviously lacks the universality of a simulation language (such as SIMSCRIPT or SIMULA), PMT can be mastered in minutes rather than months, and system models can be constructed and modified in minutes rather than days. In effect, PMT is a "spreadsheet" version of simulation -- except that it is considerably easier to use than most spreadsheet packages.

The Performance Modeling Tool guides the user, step by step, in the construction of a model, by specifying a network of Processors, Input Devices, Output Devices, and Buffers. "Tasks" enter this network at an Input Device, are processed by Processors, stored in Buffers, and eventually exit at an Output Device. This initial version of PMT does not distinguish between individual tasks -- it just counts tasks. There are also fixed paths for tasks to follow through the system. (Subsequent versions will relax both these restrictions.)

PMT also includes a file system in which alternative designs, and the results of model runs can be saved.

Although PMT is described in computer system terminology, it is actually applicable to a much wider range of applications. PMT is capable of representing any fixed-routing, discrete unit, network flow problem. The tasks could just as well be jobs in a factory, messages in a communication network, or vehicles in a transportation grid.

1. The PMT System Modules

A PMT system layout is a rectangular grid of square "areas", in numbered rows and columns. (Since at any time, the computer screen can view only a 6 by 10 portion of the layout, the "viewing window" must sometimes be shifted from one position to another.) For example, the initial view is of the Northwest corner of an empty layout, as shown in

Figure 1. -

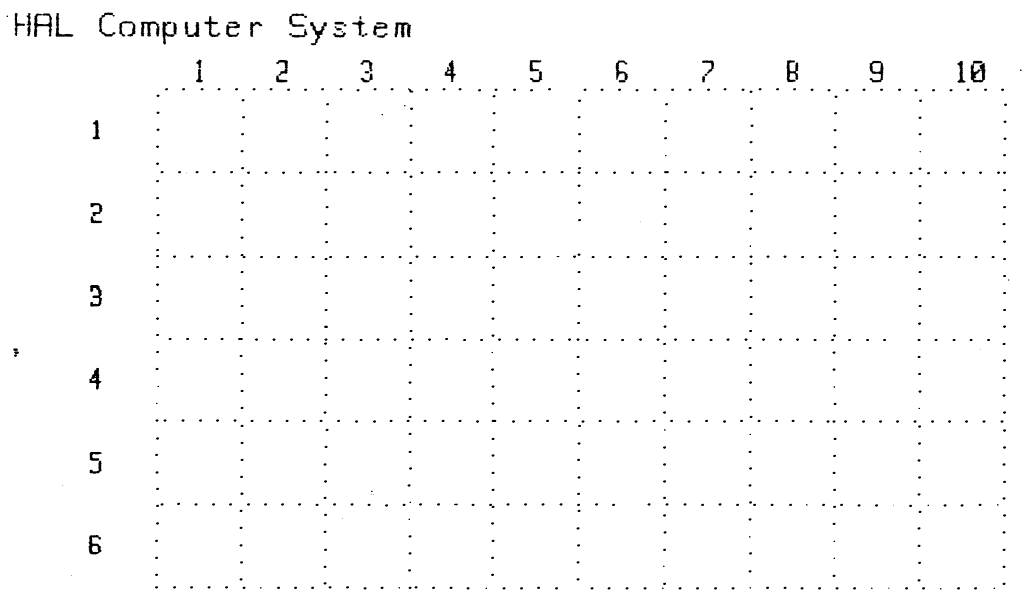

Figure 1. Empty System Layout Display at the start of Design

Each area of the system layout can accommodate one of the following types of modules:

Processor -- a module that processes a task. Each Processor can work on only one task at a time. A Processor can (optionally) have two input streams, in which case it "merges" two input tasks to form a single output task.

Input Device -- a module that receives tasks from outside the system for release, as requested, to Processors within the system.

Output Device -- an module that accepts finished tasks from Processors for delivery outside the system.

Buffer Storage -- An module inserted between two Processors that is capable of storing one or more tasks. The purpose is to decouple the two Processors, and give them greater independence in scheduling the processing of tasks.

Tasks move from one module to another, along communication "paths". The paths are assumed to have infinite speed and negligible storage capacity. (When necessary, path delay and storage capacity can be represented by an appropriate dummy Processor and Buffer Storage module.)

The design of a computer system is accomplished by placing a selection of Processors, Input Devices, Output Devices and Buffer Storage modules in different positions on the system layout, and specifying the communication paths between these modules. For example, a simple system layout is shown in Figure 2. -

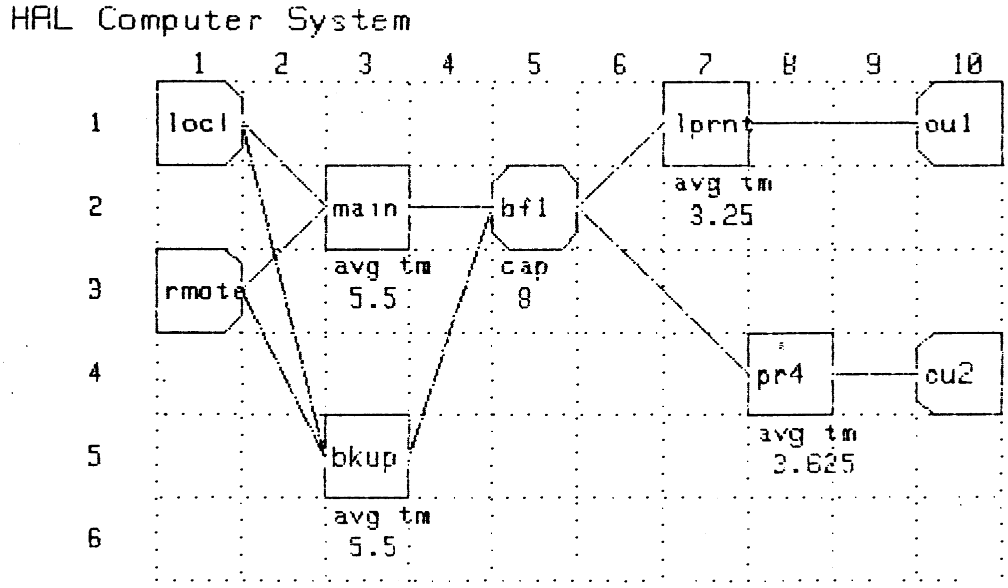

Figure 2. System Layout during Design (normal scale)

During design, each area is automatically assigned a unique identifying name (such as **bfl**, **pr4**, **oul** and **ou2** in Figure 2), which the designer can replace with a more meaningful choice (such as **locl**, **rmote**, **main**, **bkup** and **lprnt** in Figure 2).

The legend below the Processors in Figure 2 shows the average of the distribution of processing times for that Processor; the legend below each Buffer Storage module shows the task capacity of the module.

When appropriate, the user can examine individual modules in more detail, as shown in Figure 3, or a larger region of the system layout in less detail, as shown in Figure 4.

HAL Computer System

1

Input Device

Name: locl
Tasks started this period:
52
Cumulative tasks started:
89
Release tasks to:
2 3
Also release tasks to:
5 3
Batch arrival INACTIVE
Batch size is: 5
Batch interval is: 10

1

Figure 3. Expanded Display of a Single Module

HAL Computer System

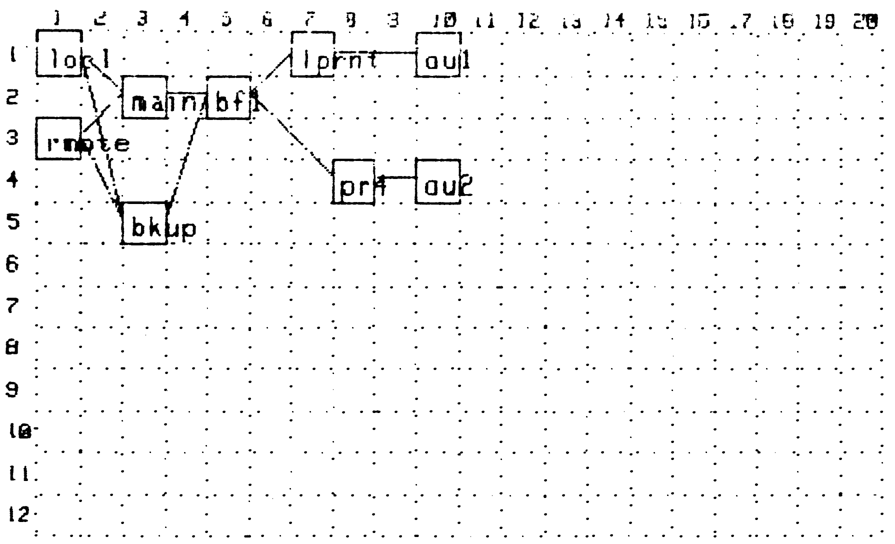

Figure 4. Reduced-scale System Layout

2. Design Mode

While in "design mode", the user can change the characteristics of the system -- by adding, modifying, moving or removing individual modules or paths. The options are the following:

For **Processors:**

location on the layout
processor name
origin of input tasks
origin of secondary input tasks
(for a "merging processor")
destination of output tasks
minimum and maximum of distribution of processing times
(default is constant time of 1 timeunit)
yield of "good" tasks (default is 100%)
("bad" tasks can either be discarded or rerun)

For **Input Devices:**

location on the layout
device name
location of Processor to be supplied with tasks
location of second Processor to be supplied with tasks
"normal" or "batch" delivery to module:
- normal delivery is continuous and instantaneous; there is always a task available, and it is instantaneously replaced when used (this is the default delivery mode)
- batch delivery involves the delivery to the Input Device of fixed size (user specified) batches of tasks at regular (user specified) intervals of time (the default batch size is 5; the default delivery interval is 10 timeunits)

For **Output Devices:**

location in the layout
device name
Processor which is source of finished tasks
second Processor which is source of finished tasks

For **Buffer Storage Modules:**

location in the layout
module name
storage capacity in tasks (default is 1 task)
Processor to supply input tasks
second Processor to supply input tasks
Processor to receive tasks from Buffer
second Processor to receive tasks from Buffer

The design process is largely menu-controlled, and activated by the function keys on the keyboard. The role of each key at any time is shown on the screen. After each step, the current form of the design is displayed, as illustrated in Figures 2, 3 and 4.

The designer can move freely over the layout, refining the design. It is not necessary to fully specify one module before moving to another. At any point, the user can leave the design mode to either store or run the model.

3. Running an PMT System Model

At any point, the model can be run to determine its processing capacity -- the rate at which finished tasks are delivered to the various Output Devices. The run can also identify bottlenecks that limit the system capacity.

Even an incomplete design can be run, although obviously the results will reflect any structural omissions. For example, a particular path may "deadend", with no outlet to a Output Device. Nevertheless, the model can be run. If, at some point, the model becomes deadlocked, so that no further processing is possible, the run is terminated.

The "events" in the running of an PMT model are the completion of processing tasks at Processors and the delivery of batches of tasks to Input Devices, when the optional batch delivery feature is active. The run proceeds by repeatedly determining the "next event" and propagating the effect of that event throughout the system.

For example, when a Processor finishes an processing a task, that task is released to the specified output destination for that Processor. Then, if a task is waiting, the Processor begins work on the next task. Presumably, in a perfectly balanced system, the flow of tasks would be smooth and continuous and no Processor would ever be idle. However, with more realistic characteristics, a Processor occasionally will be unable to dispose of its finished task (because the next Processor is still busy) and be forced to wait unproductively until the congestion clears. It is then said to be "blocked". Similarly, a Processor may, from time to time, have to wait for needed input before beginning an operation. In particular, a Merging Processor, which makes one task out of two, may have one input task but not the other, and cannot begin processing until the second type of input task becomes available.

A run can be periodically interrupted, in order to examine the "state" of the system. An example of a state display is shown in Figure 5. In this display, Input Devices show the number of tasks that have been started, and Output Devices show the number of tasks that have been finished. Buffer Storage modules show the current number of tasks present, and Processors indicate whether they are idle or busy, and if busy, when processing of the current task will be completed.

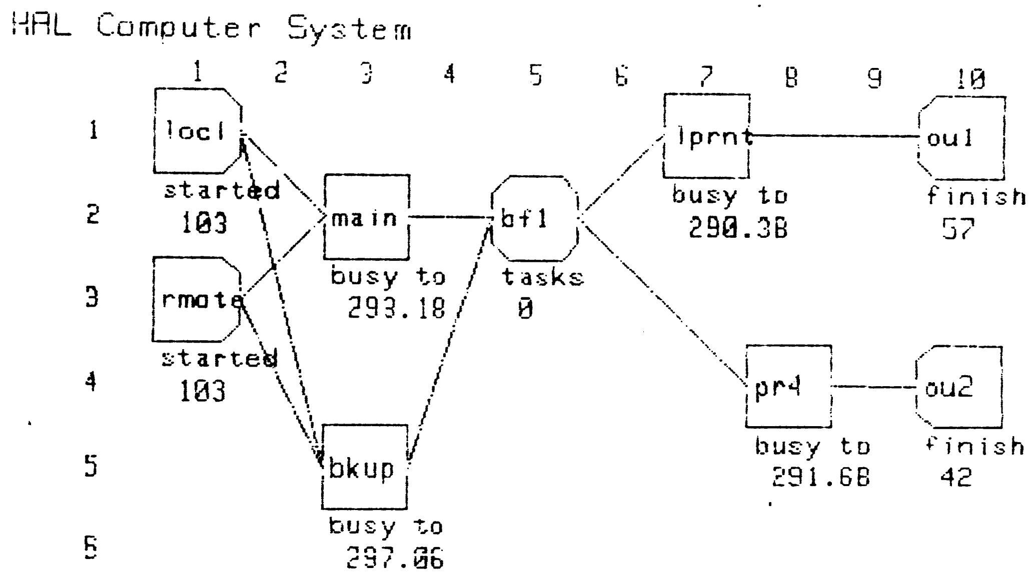

Figure 5. System Display during a pause in run (at time 288.3)

As in design mode, the pause-during-run display can be altered in scale (to show more or less of the layout), and the position of the viewing window can be moved.

3.1. RUN Controls

The user has the following controls over the run of an PMT model:

Specify the duration of the run in timeunits
(default is 31000 timeunits)
Initialize (or re-initialize) the model -- to a
state that is "empty and idle"
Start data collection (discarding results from an initial "run-in" period)
Start a new data collection "period", adding results from the previous period to the "cumulative results"
Specify that the run should automatically PAUSE at regular time intervals (default is 200 timeunits)
Specify that the run should automatically PAUSE after the occurrence of a fixed number of (completion or delivery) events (default is 50 events)
Single-step the run; PAUSE after every event
Initiate the display of results (see Section 3.3)
Modify the state of the system (see Section 3.2)
Resume the run

In addition to these options, which are effective whenever the run is PAUSED, the user can manually interrupt the run at any time.

3.2. Manual Change in the State of the Model

The state represents the results of running the model. The state changes, more or less continuously, during a run. However, whenever the run is PAUSED, the user can manually alter the state. When the run is resumed, it will start from the altered state.

Changes in state that can be made manually in a PAUSED model are the following:

For Processors:
- can be made "busy"
- can be made "idle"
- can be switched "off" (to remain unavailable for processing until manually made "busy")

For Input Devices:
- can change number of tasks "started"
- can start/stop batch deliveries
- can force batch delivery

For Buffer Storage Modules:
- can change tasks waiting
- can change maximum tasks observed todate

For Output Devices:
- can change number of tasks "finished"

3.3. Display of Results

The results of a run can be obtained for the current period, or cumulatively -- the aggregate of all periods since the beginning of data collection. (The beginning of data collection is not necessarily the beginning of the run, since the user may elect to discard the results of an initial "run-in" interval.)

Note that periods are entirely user-controlled. A new period is initiated only when the user explicitly (and manually) requests it. In particular, periods are independent of the automatic display intervals that the user can request.

The user also has the choice between "raw results" and "rates". The differences are the following:

For Processors:

<u>results</u> show the current state -- idle, busy, blocked, offline, etc.

<u>rate</u> shows the percentage of time the Processor has been busy

For Input Devices:

<u>results</u> show the number of tasks that have been started from this Device

<u>rate</u> shows the number of tasks per timeunit started from this Device

For Buffer Storage Modules:

<u>results</u> show the current number of tasks waiting

<u>rate</u> shows the maximum task level observed, as a percentage of Buffer capacity

For Output Devices:

<u>results</u> show the number of finished tasks that have arrived at this Device

<u>rate</u> shows the number of finished tasks per timeunit at this Device

During any PAUSE in a run, the user also has the option of printing a paper copy of the results. For each module, the results and rates are printed, both for the current period and cumulatively.

4. The PMT Storage Manager

The Storage Manager manages the "PMT Models" disk. It allows models to be stored on and retrieved from that disk, the contents of the disk to be listed, and models to be erased from the disk.

Note that a system model has both a <u>structure</u> -- the result of the design process, and a <u>state</u> -- the result of running the model. (For a model that has never been run, the state is null.) The model can be stored at any point in its design or execution, and subsequently retrieved exactly as stored. Once retrieved, a run can be resumed, reinitialized, etc., or the model structure can be modified, exactly as if the model had remained current.

5. Control Hierarchy during PMT Session

PMT employs a hierarchy of "modes" during a session. Each mode consists of a certain set of actions, which are displayed in a menu on the screen, denoting which function key controls which action. For the most part, this is simple to understand and use, but it may help to be told that the modes form an (inverted) tree and that, in general, K4 always "leaves" the current mode to move upward (toward the root) in the control tree. The other keys either shift to lower modes, take action directly, or initiate a dialog that solicits input from the keyboard.

HELP is available in most modes by pressing K0.

A partial outline of the most important, high-level modes is shown below:

Main session control
- K0 HELP
- K4 quit session
- K5 enter Design mode
- K6 create new model; then enter Design mode
- K7 enter Store/Retrieve mode
- K9 enter Run mode

Design mode (to modify current model)
- K0 HELP
- K1 move an existing module
- K2 change scale of display
- K3 shift view of display
- K4 leave Design mode (return to Main session control)
- K5 enter Design PROCESSOR mode
- K6 enter Design BUFFER STORAGE MODULE mode
- K7 enter Design INPUT DEVICE mode
- K8 enter Design OUTPUT DEVICE mode
- K9 remove an existing module

Store/Retrieve mode
- K1 create new model
- K2 change name of current model
- K3 specify disk drive for PMT Models disk
- K4 leave Store/Retrieve mode (return to Main session control)
- K5 store current model on PMT Models disk
- K6 list contents of PMT Models disk
- K7 retrieve a model from PMT Models disk
- K9 erase model from PMT Models disk

Run mode (run the current model)
- K0 HELP
- K1 enter Set Run Control mode
- K2 change scale of display
- K3 shift view of display
- K4 leave Run mode (return to Main session control)
- K5 re-initialize run
- K6 start data collection
- K7 enter Result Display mode
- K8 enter Modify State mode
- K9 start the model running

6. Acknowledgements

PMT is a derivative of the Factory Modeling System (Cornell Computer Science Technical Report TR 84-596) in which "Workcenters" have been recast as "Processors", "Shipping Docks" as "Output Devices", etc.

The original idea for this approach to simulation arose in a discussion with Dr. Joel Birnbaum of the Hewlett-Packard Research Center. My development of the idea benefitted immeasurably from discussions with my Cornell colleagues Jack Muckstadt, Bill Maxwell, Lee Schruben and Steve Worona.

The entire field of discrete event simulation owes its existence to Dr. Harry M. Markowitz, of SIMSCRIPT fame, and many of the techniques employed in this new approach originated in my collaboration with Markowitz many years ago.

Appendix A. PMT Loading Procedures

PMT runs on HP 9836A, 9836U and 9816 computers, with a minimum of 512K of memory. PMT is built upon the HP Extended Basic 2.0 System (Option 711). The system is supplied on two disks:

PMT1 contains the system program
PMT1M contains sample models

The following steps are required to run:

1. Insert the BASIC 2.0 System Disk (supplied by HP) and power-up the computer. (Usually the right-hand drive of a dual-drive unit is preferred for the system disk.)
2. When the screen confirms that "BASIC 2.0 is ready", remove the BASIC disk and replace it with the PMT1 disk. (This must be in the right-hand drive.)
3. Type **LOAD "PMT1"** and press the EXECUTE key.
4. When the "disk-active" light goes out, press the RUN key. After a few seconds, the PMT announcement should appear. Follow instructions in the key-labelling white blocks at the bottom of the screen.
5. To load and use a sample model:
   a. Insert the PMT1M disk in the left-hand drive.
   b. Press K7 for **Store/Retrieve.**
   c. Press K3 for **Assign disk.**
   d. Specify disk drive, as instructed on screen.
   e. Press K6 to list Models available.
   f. Press K7 to **Retrieve Model.**
   g. Enter stored-model name (from list displayed in step 5e).

   The specified model will be loaded, and the system will be in Main Session Control mode (see Section 5)

   (Sample models may be stored on the PMT1 disk, rather than on a separate PMT1M disk. If this is the case, move the PMT1 disk from the right-hand to the left-hand drive for step 5a.)

Appendix B. Limiting Dimensions and Performance

The limits on PMT dimensions are determined by various array sizes in the program. The standard limits supplied with the system are the following:

System layout: 20 rows by 30 columns
Processors: 30
Buffer storage modules: 30
Input Devices: 10
Output Devices: 10
Paths between modules: 100

These limits are easily changed, and can be made substantially larger. With the limits as supplied, a system model requires a data area of aproximately 10K bytes. For larger limits this would be increased proportionally -- estimate aproximately 12 bytes per individual module (assuming that the number of each type of module is proportion to the number of rows or columns, and not their product).

Response time during model design is, in most contexts, independent of dimensions. However, there are a few contexts -- typically, initializations -- where response is proportional to the size of the model. But even in these contexts it is generally dependent on the number of modules actually used, rather than on the limiting dimensions of the system layout. (Only the store/load time in the Storage Manager is proportional to the limiting dimensions.)

The running time of a model is proportional to the sum of the number of Processors and Input Devices actually specified in the model, and not on the potential number of such modules. Note also that running time is proportional to the number of events, and not on clock time. For example, suppose a particular model was capable of processing 10 events per second of real time. Doubling the processing time values at each Processor (and the batch intervals at Input Devices with batch delivery) in this model would not materially alter the running speed. The run would still proceed at 10 events per second of real time; although the simulated system clock would advance at twice the previous rate.

May 3, 1984

A User's Guide to

PMT - A Computer System Performance Modeling Tool

Richard Conway

TR 84-605
May 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

A User's Guide to

PMT - A Computer System Performance Modeling Tool

Richard Conway

This Performance Modeling Tool is a simulation system for overall throughput measurement of a computer system. It permits a schematic, high-level model of the system to be "run" so that its capacity to "process tasks" can be observed.

The technique employed in PMT -- discrete event simulation -- has long been used for this purpose, but has previously involved custom programming and large computers. The Performance Modeling Tool makes this capability available in quite a different form. It is an interactive, graphical system that runs on a desktop computer. It also does not require any conventional "programming". While it obviously lacks the universality of a simulation language (such as SIMSCRIPT or SIMULA), PMT can be mastered in minutes rather than months, and system models can be constructed and modified in minutes rather than days. In effect, PMT is a "spreadsheet" version of simulation -- except that it is considerably easier to use than most spreadsheet packages.

The Performance Modeling Tool guides the user, step by step, in the construction of a model, by specifying a network of Processors, Input Devices, Output Devices, and Buffers. "Tasks" enter this network at an Input Device, are processed by Processors, stored in Buffers, and eventually exit at an Output Device. This initial version of PMT does not distinguish between individual tasks -- it just counts tasks. There are also fixed paths for tasks to follow through the system. (Subsequent versions will relax both these restrictions.)

PMT also includes a file system in which alternative designs, and the results of model runs can be saved.

Although PMT is described in computer system terminology, it is actually applicable to a much wider range of applications. PMT is capable of representing any fixed-routing, discrete unit, network flow problem. The tasks could just as well be jobs in a factory, messages in a communication network, or vehicles in a transportation grid.

1. The PMT Modules

A PMT system layout is a rectangular grid of square "areas", in numbered rows and columns. (Since at any time, the computer screen can view only a 6 by 10 portion of the layout, the "viewing window" must sometimes be shifted from one position to another.) For example, the initial view is of the Northwest corner of an empty layout, as shown in

Figure 1. -

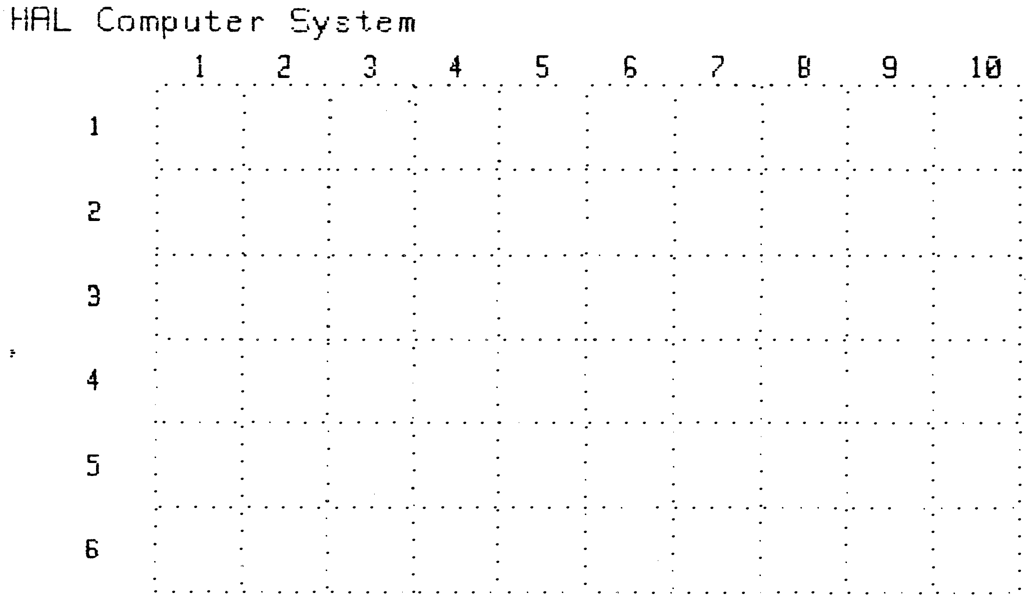

Figure 1. Empty System Layout Display at the start of Design

Each area of the system layout can accommodate one of the following types of modules:

Processor -- a module that processes a task. Each Processor can work on only one task at a time. A Processor can (optionally) have two input streams, in which case it "merges" two input tasks to form a single output task.

Input Device -- a module that receives tasks from outside the system for release, as requested, to Processors within the system.

Output Device -- an module that accepts finished tasks from Processors for delivery outside the system.

Buffer Storage -- An module inserted between two Processors that is capable of storing one or more tasks. The purpose is to decouple the two Processors, and give them greater independence in scheduling the processing of tasks.

Tasks move from one module to another, along communication "paths". The paths are assumed to have infinite speed and negligible storage capacity. (When necessary, path delay and storage capacity can be represented by an appropriate dummy Processor and Buffer Storage module.)

The design of a computer system is accomplished by placing a selection of Processors, Input Devices, Output Devices and Buffer Storage modules in different positions on the system layout, and specifying the communication paths between these modules. For example, a simple system layout is shown in Figure 2. -

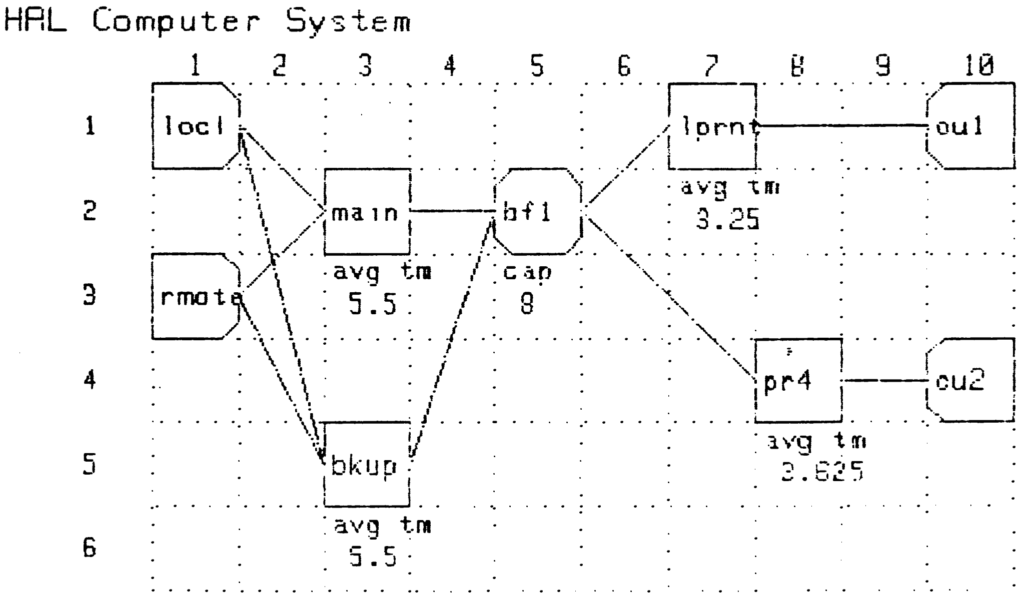

Figure 2. System Layout during Design (normal scale)

During design, each area is automatically assigned a unique identifying name (such as **bf1**, **pr4**, **ou1** and **ou2** in Figure 2), which the designer can replace with a more meaningful choice (such as **locl**, **rmote**, **main**, **bkup** and **lprnt** in Figure 2).

The legend below the Processors in Figure 2 shows the average of the distribution of processing times for that Processor; the legend below each Buffer Storage module shows the task capacity of the module.

When appropriate, the user can examine individual modules in more detail, as shown in Figure 3, or a larger region of the system layout in less detail, as shown in Figure 4.

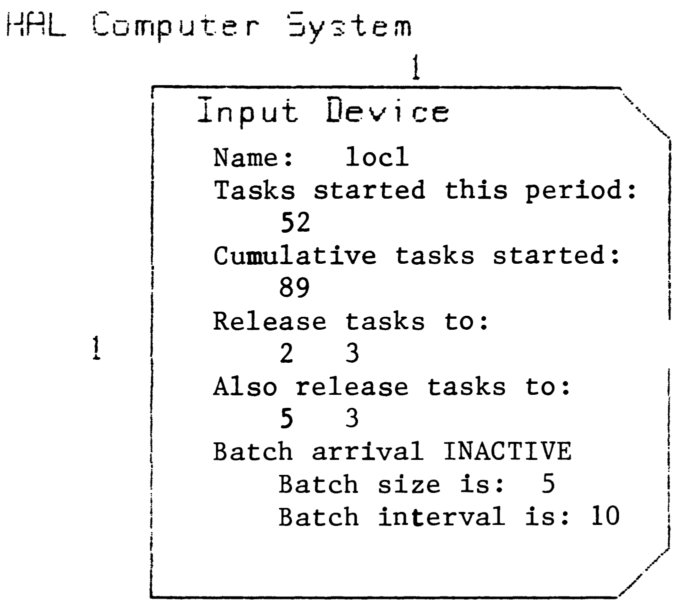

Figure 3. Expanded Display of a Single Module

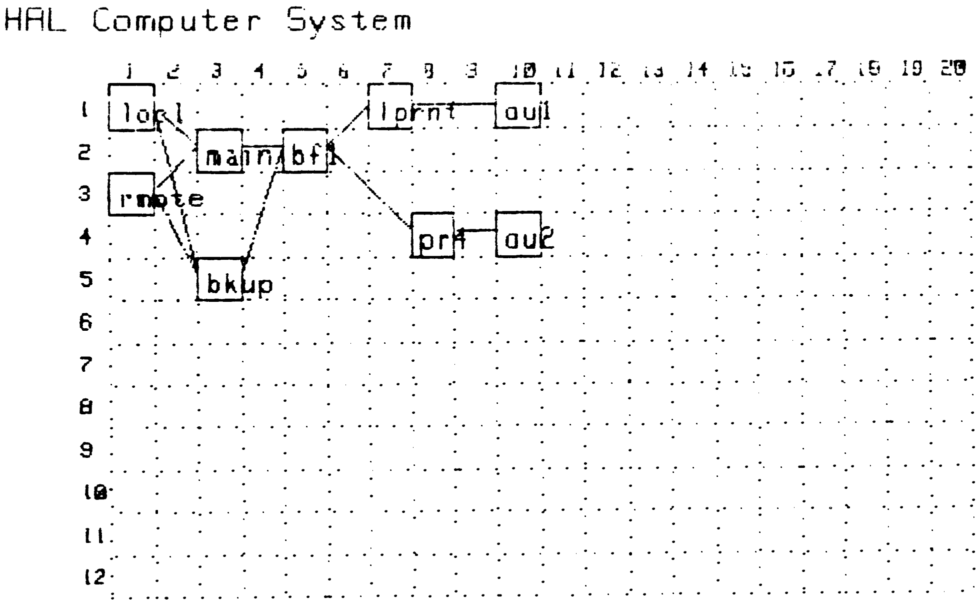

Figure 4. Reduced-scale System Layout

2. Design Mode

While in "design mode", the user can change the characteristics of the system -- by adding, modifying, moving or removing individual modules or paths. The options are the following:

For **Processors:**
- location on the layout
- processor name
- origin of input tasks
- origin of secondary input tasks
  - (for a "merging processor")
- destination of output tasks
- minimum and maximum of distribution of processing times
  - (default is constant time of 1 timeunit)
- yield of "good" tasks (default is 100%)
  - ("bad" tasks can either be discarded or rerun)

For **Input Devices:**
- location on the layout
- device name
- location of Processor to be supplied with tasks
- location of second Processor to be supplied with tasks
- "normal" or "batch" delivery to module:
  - normal delivery is continuous and instantaneous; there is always a task available, and it is instantaneously replaced when used (this is the default delivery mode)
  - batch delivery involves the delivery to the Input Device of fixed size (user specified) batches of tasks at regular (user specified) intervals of time (the default batch size is 5; the default delivery interval is 10 timeunits)

For **Output Devices:**
- location in the layout
- device name
- Processor which is source of finished tasks
- second Processor which is source of finished tasks

For **Buffer Storage Modules:**
- location in the layout
- module name
- storage capacity in tasks (default is 1 task)
- Processor to supply input tasks
- second Processor to supply input tasks
- Processor to receive tasks from Buffer
- second Processor to receive tasks from Buffer

The design process is largely menu-controlled, and activated by the function keys on the keyboard. The role of each key at any time is shown on the screen. After each step, the current form of the design is displayed, as illustrated in Figures 2, 3 and 4.

The designer can move freely over the layout, refining the design. It is not necessary to fully specify one module before moving to another. At any point, the user can leave the design mode to either store or run the model.

3. Running an PMT System Model

At any point, the model can be run to determine its processing capacity -- the rate at which finished tasks are delivered to the various Output Devices. The run can also identify bottlenecks that limit the system capacity.

Even an incomplete design can be run, although obviously the results will reflect any structural omissions. For example, a particular path may "deadend", with no outlet to a Output Device. Nevertheless, the model can be run. If, at some point, the model becomes deadlocked, so that no further processing is possible, the run is terminated.

The "events" in the running of an PMT model are the completion of processing tasks at Processors and the delivery of batches of tasks to Input Devices, when the optional batch delivery feature is active. The run proceeds by repeatedly determining the "next event" and propagating the effect of that event throughout the system.

For example, when a Processor finishes an processing a task, that task is released to the specified output destination for that Processor. Then, if a task is waiting, the Processor begins work on the next task. Presumably, in a perfectly balanced system, the flow of tasks would be smooth and continuous and no Processor would ever be idle. However, with more realistic characteristics, a Processor occasionally will be unable to dispose of its finished task (because the next Processor is still busy) and be forced to wait unproductively until the congestion clears. It is then said to be "blocked". Similarly, a Processor may, from time to time, have to wait for needed input before beginning an operation. In particular, a Merging Processor, which makes one task out of two, may have one input task but not the other, and cannot begin processing until the second type of input task becomes available.

A run can be periodically interrupted, in order to examine the "state" of the system. An example of a state display is shown in Figure 5. In this display, Input Devices show the number of tasks that have been started, and Output Devices show the number of tasks that have been finished. Buffer Storage modules show the current number of tasks present, and Processors indicate whether they are idle or busy, and if busy, when processing of the current task will be completed.

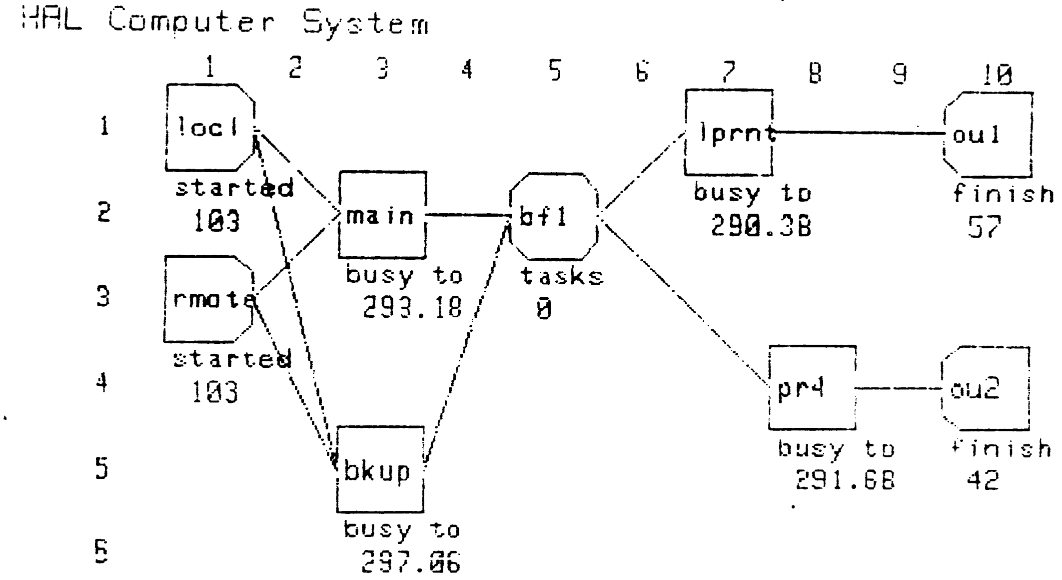

Figure 5. System Display during a pause in run (at time 288.3)

As in design mode, the pause-during-run display can be altered in scale (to show more or less of the layout), and the position of the viewing window can be moved.

3.1. RUN Controls

The user has the following controls over the run of an PMT model:

Specify the duration of the run in timeunits
(default is 31000 timeunits)
Initialize (or re-initialize) the model -- to a
state that is "empty and idle"
Start data collection (discarding results from an initial "run-in" period)
Start a new data collection "period", adding results from the previous period to the "cumulative results"
Specify that the run should automatically PAUSE at regular time intervals (default is 200 timeunits)
Specify that the run should automatically PAUSE after the occurrence of a fixed number of (completion or delivery) events (default is 50 events)
Single-step the run; PAUSE after every event
Initiate the display of results (see Section 3.3)
Modify the state of the system (see Section 3.2)
Resume the run

In addition to these options, which are effective whenever the run is PAUSED, the user can manually interrupt the run at any time.

3.2. Manual Change in the State of the Model

The state represents the results of running the model. The state changes, more or less continuously, during a run. However, whenever the run is PAUSED, the user can manually alter the state. When the run is resumed, it will start from the altered state.

Changes in state that can be made manually in a PAUSED model are the following:

For Processors:
- can be made "busy"
- can be made "idle"
- can be switched "off" (to remain unavailable for processing until manually made "busy")

For Input Devices:
- can change number of tasks "started"
- can start/stop batch deliveries
- can force batch delivery

For Buffer Storage Modules:
- can change tasks waiting
- can change maximum tasks observed todate

For Output Devices:
- can change number of tasks "finished"

3.3. Display of Results

The results of a run can be obtained for the current period, or cumulatively -- the aggregate of all periods since the beginning of data collection. (The beginning of data collection is not necessarily the beginning of the run, since the user may elect to discard the results of an initial "run-in" interval.)

Note that periods are entirely user-controlled. A new period is initiated only when the user explicitly (and manually) requests it. In particular, periods are independent of the automatic display intervals that the user can request.

The user also has the choice between "raw results" and "rates". The differences are the following:

For Processors:

- _results_ show the current state -- idle, busy, blocked, offline, etc.
- _rate_ shows the percentage of time the Processor has been busy

For Input Devices:

- _results_ show the number of tasks that have been started from this Device
- _rate_ shows the number of tasks per timeunit started from this Device

For Buffer Storage Modules:

- _results_ show the current number of tasks waiting
- _rate_ shows the maximum task level observed, as a percentage of Buffer capacity

For Output Devices:

- _results_ show the number of finished tasks that have arrived at this Device
- _rate_ shows the number of finished tasks per timeunit at this Device

During any PAUSE in a run, the user also has the option of printing a paper copy of the results. For each module, the results and rates are printed, both for the current period and cumulatively.

4. The PMT Storage Manager

The Storage Manager manages the "PMT Models" disk. It allows models to be stored on and retrieved from that disk, the contents of the disk to be listed, and models to be erased from the disk.

Note that a system model has both a _structure_ -- the result of the design process, and a _state_ -- the result of running the model. (For a model that has never been run, the state is null.) The model can be stored at any point in its design or execution, and subsequently retrieved exactly as stored. Once retrieved, a run can be resumed, reinitialized, etc., or the model structure can be modified, exactly as if the model had remained current.

5. Control Hierarchy during PMT Session

PMT employs a hierarchy of "modes" during a session. Each mode consists of a certain set of actions, which are displayed in a menu on the screen, denoting which function key controls which action. For the most part, this is simple to understand and use, but it may help to be told that the modes form an (inverted) tree and that, in general, K4 always "leaves" the current mode to move upward (toward the root) in the control tree. The other keys either shift to lower modes, take action directly, or initiate a dialog that solicits input from the keyboard.

HELP is available in most modes by pressing K0.

A partial outline of the most important, high-level modes is shown below:

Main session control
- K0 HELP
- K4 quit session
- K5 enter Design mode
- K6 create new model; then enter Design mode
- K7 enter Store/Retrieve mode
- K9 enter Run mode

Design mode (to modify current model)
- K0 HELP
- K1 move an existing module
- K2 change scale of display
- K3 shift view of display
- K4 leave Design mode (return to Main session control)
- K5 enter Design PROCESSOR mode
- K6 enter Design BUFFER STORAGE MODULE mode
- K7 enter Design INPUT DEVICE mode
- K8 enter Design OUTPUT DEVICE mode
- K9 remove an existing module

Store/Retrieve mode
- K1 create new model
- K2 change name of current model
- K3 specify disk drive for PMT Models disk
- K4 leave Store/Retrieve mode (return to Main session control)
- K5 store current model on PMT Models disk
- K6 list contents of PMT Models disk
- K7 retrieve a model from PMT Models disk
- K9 erase model from PMT Models disk

Run mode (run the current model)
- K0 HELP
- K1 enter Set Run Control mode
- K2 change scale of display
- K3 shift view of display
- K4 leave Run mode (return to Main session control)
- K5 re-initialize run
- K6 start data collection
- K7 enter Result Display mode
- K8 enter Modify State mode
- K9 start the model running

6. Acknowledgements

PMT is a derivative of the Factory Modeling System (Cornell Computer Science Technical Report TR 84-596) in which "Workcenters" have been recast as "Processors", "Shipping Docks" as "Output Devices", etc.

The original idea for this approach to simulation arose in a discussion with Dr. Joel Birnbaum of the Hewlett-Packard Research Center. My development of the idea benefitted immeasurably from discussions with my Cornell colleagues Jack Muckstadt, Bill Maxwell, Lee Schruben and Steve Worona.

The entire field of discrete event simulation owes its existence to Dr. Harry M. Markowitz, of SIMSCRIPT fame, and many of the techniques employed in this new approach originated in my collaboration with Markowitz many years ago.

Appendix A. PMT Loading Procedures

PMT runs on HP 9836A, 9836U and 9816 computers, with a minimum of 512K of memory. PMT is built upon the HP Extended Basic 2.0 System (Option 711). The system is supplied on two disks:

PMT1 contains the system program
PMT1M contains sample models

The following steps are required to run:

1. Insert the BASIC 2.0 System Disk (supplied by HP) and power-up the computer. (Usually the right-hand drive of a dual-drive unit is preferred for the system disk.)
2. When the screen confirms that "BASIC 2.0 is ready", remove the BASIC disk and replace it with the PMT1 disk. (This must be in the right-hand drive.)
3. Type **LOAD "PMT1"** and press the EXECUTE key.
4. When the "disk-active" light goes out, press the RUN key. After a few seconds, the PMT announcement should appear. Follow instructions in the key-labelling white blocks at the bottom of the screen.
5. To load and use a sample model:
   a. Insert the PMT1M disk in the left-hand drive.
   b. Press K7 for **Store/Retrieve.**
   c. Press K3 for **Assign disk.**
   d. Specify disk drive, as instructed on screen.
   e. Press K6 to list Models available.
   f. Press K7 to **Retrieve Model.**
   g. Enter stored-model name (from list displayed in step 5e).

   The specified model will be loaded, and the system will be in Main Session Control mode (see Section 5)

   (Sample models may be stored on the PMT1 disk, rather than on a separate PMT1M disk. If this is the case, move the PMT1 disk from the right-hand to the left-hand drive for step 5a.)

Appendix B. Limiting Dimensions and Performance

The limits on PMT dimensions are determined by various array sizes in the program. The standard limits supplied with the system are the following:

System layout: 20 rows by 30 columns
Processors: 30
Buffer storage modules: 30
Input Devices: 10
Output Devices: 10
Paths between modules: 100

These limits are easily changed, and can be made substantially larger. With the limits as supplied, a system model requires a data area of aproximately 10K bytes. For larger limits this would be increased proportionally -- estimate aproximately 12 bytes per individual module (assuming that the number of each type of module is proportion to the number of rows or columns, and not their product).

Response time during model design is, in most contexts, independent of dimensions. However, there are a few contexts -- typically, initializations -- where response is proportional to the size of the model. But even in these contexts it is generally dependent on the number of modules actually used, rather than on the limiting dimensions of the system layout. (Only the store/load time in the Storage Manager is proportional to the limiting dimensions.)

The running time of a model is proportional to the sum of the number of Processors and Input Devices actually specified in the model, and not on the potential number of such modules. Note also that running time is proportional to the number of events, and not on clock time. For example, suppose a particular model was capable of processing 10 events per second of real time. Doubling the processing time values at each Processor (and the batch intervals at Input Devices with batch delivery) in this model would not materially alter the running speed. The run would still proceed at 10 events per second of real time; although the simulated system clock would advance at twice the previous rate.

May 3, 1984