PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :		(11) International Publication Number:	WO 94/09429
G06F 9/455	A1	(43) International Publication Date:	28 April 1994 (28.04.94)

(21) International Application Number: PCT/US93/08710

(22) International Filing Date: 13 September 1993 (13.09.93)

(30) Priority data:
07/959,553 13 October 1992 (13.10.92) US

(71) Applicant: ZITEL CORPORATION [US/US]; 47211 Bayside Parkway, Fremont, CA 94538 (US).

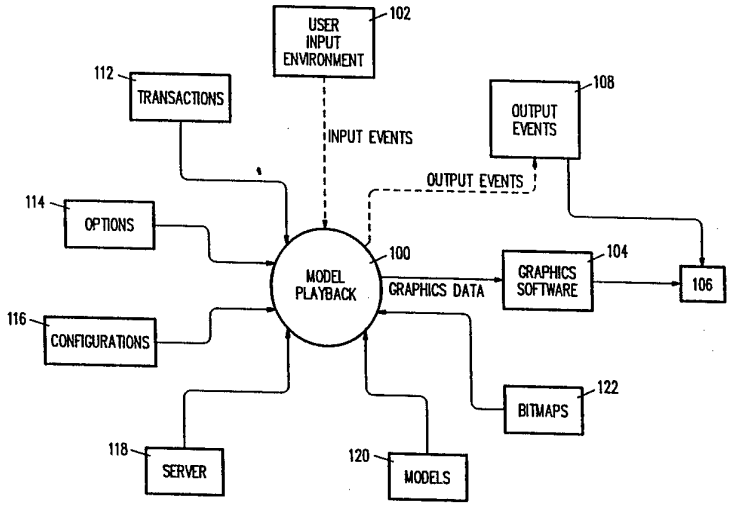
(72) Inventors: SALSBURG, Michael, A. ; SALSBURG, Linda, B. ; 1033 South New Street, West Chester, PA 19382 (US).

(74) Agents: CASERZA, Steven, F. et al.; Flehr, Hohbach, Test, Albritton & Herbert, 4 Embarcadero Center, Suite 3400, San Francisco, CA 94111-4187 (US).

(81) Designated States: AU, CA, JP, KR, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
With amended claims.

(54) Title: METHOD AND SYSTEM FOR COMPUTER PERFORMANCE EVALUATION



(57) Abstract

A method for evaluating computer performance using a graphical user interface (104) enabling simultaneous analysis of component utilization and internal response time is disclosed. Operational information is gathered from an actual computer system under normal operation. The collected data is then transformed into statistics relating to the workload and configuration of the computer system. This statistical information is supplied to a simulation program to model the computer system. Statistical information derived from the simulation by a Model Playback program (100) is then displayed through a graphical user interface (104). The interface (104) allows concurrent display of information relating to component utilizations, queuing times, and transaction delay times in an icon-based, animated manner. The Model Playback program allows a user to instruct the display to proceed either forward or backward in time. The display is arranged hierarchically. Information relating to individual or groups of components within a particular string is stored within underlying sets of statistics to which a user may progressively gain access. Component icons within a particular string can be displayed in alternative orders.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**FR**	France	**MR**	Mauritania
AU	Australia	**GA**	Gabon	**MW**	Malawi
BB	Barbados	**GB**	United Kingdom	**NE**	Niger
BE	Belgium	**GN**	Guinea	**NL**	Netherlands
BF	Burkina Faso	**GR**	Greece	**NO**	Norway
BG	Bulgaria	**HU**	Hungary	**NZ**	New Zealand
BJ	Benin	**IE**	Ireland	**PL**	Poland
BR	Brazil	**IT**	Italy	**PT**	Portugal
BY	Belarus	**JP**	Japan	**RO**	Romania
CA	Canada	**KP**	Democratic People's Republic of Korea	**RU**	Russian Federation
CF	Central African Republic			**SD**	Sudan
CG	Congo	**KR**	Republic of Korea	**SE**	Sweden
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovak Republic
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LU**	Luxembourg	**TD**	Chad
CS	Czechoslovakia	**LV**	Latvia	**TG**	Togo
CZ	Czech Republic	**MC**	Monaco	**UA**	Ukraine
DE	Germany	**MG**	Madagascar	**US**	United States of America
DK	Denmark	**ML**	Mali	**UZ**	Uzbekistan
ES	Spain	**MN**	Mongolia	**VN**	Viet Nam
FI	Finland				

METHOD AND SYSTEM FOR COMPUTER PERFORMANCE EVALUATION

INTRODUCTION

Background

The present invention relates to methods and systems for evaluating the performance of computer systems, and particularly to computer performance evaluation systems which utilize a graphical user interface to convey statistical information.

Description of the Prior Art

An increased number of computer applications are currently being developed for real-time environments in which computer systems are expected to quickly respond to user requests. Accordingly, the time delay between submission of a request and receipt of the result has become an important factor in gauging computer performance. One approach used in evaluation of computer performance involves construction of a simulation model which mirrors the configuration and workload of the actual system under scrutiny. The configuration of a system refers to the collection of hardware components from which it is comprised, while workload corresponds to the manner in which system components are utilized. The statistics gleaned from measurements taken during a simulation may then be used to, for example, evaluate whether computer performance may be improved more cost effectively through reconfiguration of existing equipment rather than through procurement of new equipment, and how a reconfiguration should be done.

Several types of components are generally included within the configuration of typical computer systems. In particular, most computer systems include a central processing unit (CPU), random access memory (RAM), and storage units with associated controllers. Particular computer systems, such as those used to, for example, process credit inquiries by employees of a bank or other financial institution, may have multiple components of the same type (e.g., two CPUs, 10 memory modules, 100 storage units and 20 controllers). The utilization of a

component refers to the percentage of time that the component is occupied servicing requests or responding to commands. For example, if a component has a utilization of 25% then it is busy on average 25% of the time and idle the remaining 75% of the time.

An additional statistic involved in the analysis of computer system performance is known as the internal response time, which is the delay experienced by a user of the computer system while the system is processing a request or command. The duration of the time interval between submission of a request or command and receipt of the desired information is predicated upon the sum of the internal response time and the delays through the system's interface apparatus (i.e., display terminal). As an example, assume that a set of credit inquires are submitted by bank personnel to a computer system having the configuration described above. Each such inquiry will cause the CPU to be occupied for a specific time interval while input/output (I/O) service requests are made to various storage devices. The internal response time of the system associated with execution of each inquiry corresponds to the time consumed during component processing time, in addition to the delays resulting from the queuing of service inquiries awaiting processing by a particular component.

Evaluation of computer configurations such as this credit inquiry system have generally focused upon statistical analysis of the average queuing and utilization times for individual components, in addition to estimation of average internal response time. However, it is believed that current performance monitoring techniques do not allow for the accumulation by component type and display of statistics in accordance with Transaction type (e.g., debit or credit inquiries). Such a capability could, for example, enable an analyst to predict component type utilization within systems disposed to simultaneously process multiple types of transactions. Alternatively, it would enable an analyst to predict how changes in components will affect a particular transaction type.

Statistics relating to computer performance evaluation have typically been communicated to a user using conventional graphical display techniques. For example, the line or bar graphs included within computer programs written for business applications have been adapted to display information relating to average queuing and device utilization times. For the most part such adaptations are neither interactive, animated, nor icon-based, and consequently are relatively difficult to use.

An example of a graphical display of computer system performance generated by a computer program of the type produced by, for example, BGS Inc., of Waltham, Massachusetts, is shown in the sketch of FIG. 1. Specifically, the printed graph provided by this type of performance evaluation program represents the utilization of disks and controllers included within a storage subsystem. The statistics used to generate the graph are derived from measurements taken from the computer system using known monitoring techniques. Circles and rectangles serve as simple icons for representing the disks and controllers in an elementary fashion, with the color of each icon being indicative of the level of the component's utilization. Although showing component utilizations, the sketch of FIG. 1 depicts neither average queuing time nor the manner in which internal response time is related to component utilization.

It is observed that individual disks and controllers are each displayed with a specific icon. As a consequence, in large computer systems having hundreds of disks this display format could require one to sift through many pages of graphics in order to locate the icon representative of a particular disk or controller. Moreover, users are precluded from interacting with the graphical display of FIG. 1 since it is generated as a printed report rather than as a video image.

FIG. 2 shows a sketch of a graphical video display (i.e., graphical user interface) produced by a second type of conventional computer performance simulation software. The video display corresponds to an animation of a queuing network simulation, in which the queuing network has been specified to model a computer system. A first server is represented by a

CPU corresponding to the circle labelled "1", while second and sixth servers are represented as disk units. The rectangular structures proximate the circles labelled "1", "2" and "6" respectively represent the queues associated with the first, second and sixth servers. During the course of the simulation the system workload passes through the CPU (server 1), and is then routed in a manner based on probability of usage to the disk unit representative of either the second or sixth server unit. Subsequent to processing by the disks representative of the second and sixth server units the workload is routed to the components identified by the numerals "3", "4" and "5". As a simulation progresses the display shows the utilization of the servers and the size of the accompanying queue.

As may be apparent from the foregoing, utilization of computer performance evaluation software of the type incorporating a graphical user interface similar to that sketched in FIG. 2 requires a relatively sophisticated understanding of queuing networks. That is, while the display icons of FIG. 2 are intended to be representative of servers and associated queues, the configuration of FIG. 2 is suggestive of the hardware relationships existing in a generalized computer system rather than a queuing network. Although the icons in FIG. 2 can be interactively selected so as to allow inspection of the queuing network associated with each server, this type of system is incapable of simultaneously displaying queuing information and, for example, internal response time.

SUMMARY OF THE INVENTION

In accordance with the teachings of this invention, it has been determined that it would be highly advantageous to provide a graphical user interface capable of the contemporaneous display of statistics pertinent to component level performance, together with those relevant to assessment of overall system performance, i.e, system workload or internal response time.

It has also been determined in accordance with the teachings of this invention that it would be advantageous in

such a graphical user interface to provide a system in which component sets of disks and controllers were collectively represented by a single icon at a first display level, and in which interactive access to a second display level allowed for inspection of statistics relating to subgroups or individual components. Such a system would further facilitate performance evaluation by enabling simultaneous display of component utilization and internal response time.

The present invention provides an improved method for evaluating computer performance which features a graphical user interface enabling simultaneous analysis of component utilization and internal response time. In a preferred embodiment of the invention operational information is gathered from an actual computer system under normal operation using a data acquisition program. The collected data is then transformed into statistics relating to the workload and configuration of the computer system. This statistical information is supplied to a simulation program to facilitate accurate modeling of the actual computer system. Operation of the actual computer system is then simulated, during which time queuing and service times are monitored for particular simulated components.

Statistical information derived from the simulation by a Model Playback program is then displayed through a graphical user interface. The graphical user interface allows concurrent display of information relating to component utilizations, queuing times, and transaction delay times in an icon-based, animated manner. The user interface is programmed to display simulation statistics for particular transaction types over a multiplicity of contiguous time epochs, and is capable of being updated in a manner which shows the progression of such epochs. The Model Playback program allows a user to instruct the display to proceed either forward or backward in time, thereby enabling performance to be monitored over particular epochs in which internal response time or component utilization for a particular transaction type exceeds specified thresholds. The display is arranged hierarchically in that some sets of functionally interrelated components are collectively

represented by individual string icons. A component string may include, for example, a storage disk with an associated controller. Information relating to individual or groups of components within a particular string is stored within underlying sets of statistics, to which a user may progressively gain access by selecting the appropriate string icon. Component icons within a particular string can be displayed in, for example, alphabetical order or in a manner related to queue length.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is an example of a prior art graphical display of computer system performance;

Figure 2 is an example of a prior art graphical video display which depicts conventional computer performance;

Figures 3-7 depict single frames of displays generated in accordance with one embodiment of the present invention;

Figure 8 is a context diagram representative of one manner of display of performance statistics as provided in accordance with this invention;

Figures 9-13 depict data structures used in accordance with one embodiment of this invention; and

Figures 14-20 are transition graphs which illustrate the manner in which operational data is processed and displayed in accordance with one embodiment of this invention.

DETAILED DESCRIPTION OF THE INVENTION

System Overview

The computer performance evaluation method of the present invention allows for the simultaneous presentation of transaction response time statistics and component utilization statistics in a concurrent, integrated manner. Statistical information is conveyed through a graphical user interface which allows a user to monitor performance over a specified time epoch. The graphical user interface incorporates

windowing display techniques to facilitate concurrent analysis of response time on both transactional and component levels. Sequential levels of windowed displays are accessed through selection of display icons, thereby enabling statistical information to be catalogued and presented in a hierarchical manner.

Examples of these and other features of the graphical user interface provided by the present invention are presented by way of introduction with reference to FIGS. 3-7. Specifically, FIGS. 3-7 are sketches corresponding to single frames, or "snapshots", of displays generated by the inventive graphical user interface. It is noted that in order to facilitate clarity of discussion the sketches of FIGS. 3-7 represent in simplified form particular interface displays. Following this introductory discussion a detailed account will be given of the manner in which statistical information relating to computer performance is processed by the present invention. Such statistics may be accumulated in real time from an applications program installed on a host computer system, or may be acquired through software simulation of the host system.

Referring to FIG. 3, there is shown a "snapshot" of a transaction delay chart generated by the graphical user interface of the present invention. The statistics used to display the chart were collected during a software simulation of the operation of a computer system having a specified workload and hardware configuration. Again, the term workload refers to a set of transactions of a specific type processed by the system, wherein differing transaction types will make varying demands upon the system CPU, memory, controller, and disk components prior to completion by the system. Although the following discussion assumes that simulation statistics were utilized in generation of the graphical user interface, the interface may be employed in real-time monitoring on the basis of data gathered from a host computer under inspection. As shown in FIG. 3, internal response times are displayed for a pair of workloads comprised of first and second transaction types (TRANS #1 and TRANS #2).

Assuming, for example, that the display depicted by the sketch of FIG. 3 was derived from a computerized banking system, the operations TRANS #1 and TRANS #2 could be representative of debit and credit inquires made by bank personnel. In order to respond to each such transactional inquiry, data is fetched from disks (through the controllers), with the CPU in conjunction with memory being used to process the fetched data. The transaction is completed upon issuance of a response to the user responsible for initiating the inquiry. If total response time corresponds to the elapsed time between submission of the inquiry and receipt of the response, then internal response time is equivalent to the difference of total response time and line transmission times, i.e., propagation delays through telephone lines and connection cables.

It is noted that the internal response times for particular transaction types will be a function of the rate at which transaction requests of each type arrive at the computer system for processing. That is, since different transactions may involve utilization of common system components, the utilization and queuing times for such components will depend upon the number of transactions competing for processing. As a consequence, the transaction delays depicted in FIG. 3 were computed on the basis of specific rates of arrival for each type of transactional request. For each transaction type there exist a specified number of input/output service requests (I/O's) made to specific disks, as well as a specified number of service request made to the CPU and to memory. Again, the aggregate set of service requests directed to the components of a host computer system correspond to the workload of the transaction. As is described in greater detail hereinafter, knowledge of the workload specifications for each transaction type and the configuration of components within the host system enables synthesis of a software simulation model. The simulation model is driven by a set of transaction requests, with the arrival rate of each type of transaction within the set being monitored by the simulation software. The simulation software is also programmed to collect statistics relating to,

for example, the time which each type of transaction occupies the CPU, the disks, and the controllers. The time spent by service requests associated with a particular transaction type in queuing for the resources offered by each of these components is also recorded. As is discussed below, the statistics garnered from the simulation program are supplied to a Model playback program disposed to generate transaction delay curves such as those shown in FIG. 3.

A user may request display of a transaction delay chart by using a conventional pointing device, such as a "mouse", to select the appropriate icon from an opening menu (not shown). Such selection results in the opening of a display window, within which the axes of the delay chart appear during an initialization sequence. By selecting a "Forward" command from the pull-down menu labeled "Control" (FIG. 3), the TRANS #1 and TRANS #2 lines are begun to be drawn from left to right. Selection of a "Pause" command from the Control menu halts drawing, while the transaction delay lines may be made to retrace themselves by specifying a "Backward" command. Moreover, a user may employ the pointing device to request that additional statistics be provided for specified data points along the TRANS #1 and TRANS #2 lines.

Referring to FIG. 4, there is shown a snapshot of a display depicting a "dialog box" superimposed upon the transaction delay chart of FIG.3, wherein the dialog box was opened by pointing to a data point along the TRANS #2 line. The dialog box provides data relating to the manner in which transactions of the second type (TRANS #2) are processed as the simulation progresses. Specifically, current processing statistics are displayed for the time epoch which includes the data point used to open the dialog box, while average processing statistics are presented for all of the epochs up to and including the selected epoch. During each time epoch the composite set of transaction requests processed by the simulation program may be referred to as a "batch". Accordingly, the entry "Batch 28" in FIG. 4 indicates that a data point within the 28th time epoch of the simulation was selected in order to open the dialog box.

As shown in FIG. 4, the entries within the dialog box following the term "Completions" indicate the number of TRANS #2 type transactions completed during the 28th time epoch, as well as the cumulative number of such transactions completed since the beginning of the simulation. Current and average values are also displayed for processor time used, for time spent queuing for the processor, for time spent using the disk controllers, for time queuing to use the controllers, for time spent using the disk units and for time spent queuing to use the units. It is thus a feature of the inventive graphical user interface that a dialog window may be superimposed over the transaction delay chart so as to enable depiction of the component type utilizations corresponding to particular transaction type internal response times.

FIG. 5 shows a snapshot of a display in which a "Servers Playback" window is superimposed over the transaction delay chart of FIG. 3. The Servers Playback window is opened by selecting a "Servers" command from the Control menu. When the Servers Playback window is opened, the user can specify FORWARD or BACKWARD from the Control menu. As the Servers Playback window displays statistics, the TRANS #1 and TRANS #2 lines in the Transaction Delay Chart are updated concurrently. Specifically, the colors of the icons included within the Servers Playback window may be specified to change during progression of the displays as a function of the queuing of the components which they represent. As shown in FIG. 5, components are presented within the Servers Playback window as being included in one of three categories: Processor, Memory or String. The color of each icon included within the Servers Playback window is predicated upon the queuing level. The queuing level for a string is associated with the highest queuing level for any of the components within the string represented by the particular icon.

When a user requests via the pointing device that additional statistics be provided for the string represented by one of the icons within the Server Playback window, the selected string is "exploded", i.e., divided, into its constituent controllers and units as shown in FIG. 6. FIG. 6

shows the graph that is presented when the string is exploded. If the pointing device is used to request additional information for a particular component, a dialog box is opened which displays the requested statistics. Referring to FIG. 7, a dialog box providing current and average statistics for a component denoted "UNIT 2" has been opened. The statistics within the dialog box of FIG. 7 include the arrival rate of service requests for a server, the average service time requested from the server, the utilization of the server and the average length of the queue of service requests awaiting the server's attention.

Acquisition of Operational Data

As mentioned above, the performance statistics displayed in accordance with the present invention may be derived from operating statistics accumulated in real time from an applications program installed on a host computer. Alternatively, a set of operating statistics may be compiled during the course of a software simulation of the host system. In either case the operating statistics are garnered by supplying either the host system or simulation model with a set of transactions. Each transaction corresponds to a user-initiated request that the computer system execute a particular function.

As is well known, most operating systems currently used within large mainframe computers were designed to accommodate batch-oriented data processing. Each batch processed by such systems is typically identified by a user code in order to facilitate billing for the use of computer resources. Unfortunately, when mainframe systems are used for online transaction processing (OLTP) it has been found that transactions cannot be easily segregated on the basis of user codes or other indicia (e.g., by batch run). That is, various transaction types may be processed while a particular "task", identified by a single user code, is running on the mainframe system. Alternatively, a single transaction type may request processing from various tasks in the system having associated therewith a plurality of user codes.

Data relating to the operation of OLTP systems can often be acquired through transaction logs compiled during the execution of applications programs. For example, the transactions log of an online banking applications program could be inspected to determine the arrival rate of credit inquiries over the time interval from 9:00 to 9:10. This determination could be made by counting the transactions having a credit inquiry code and a timestamp between 9:00 and 9:10.

In determining component utilizations, most mainframe systems record CPU utilization as a function of user code. Such systems also keep track of the number of input/output service requests (I/O's) to each of the disk units. However, these statistics are not classified on the basis of transaction type. In addition, an accounting is generally not made for time spent "queuing" during the course of processing. This scant availability of statistics from real-time processing may be attributed to the fact that collection of additional data would require a substantial increases in CPU and memory capability.

It is conceivable that operating systems could be developed in which data is recorded on the basis of transaction type, and in which statistics relating to queuing time within the system are also recorded. Until any such developments occur, however, it has been determined that the most effective way to gather the statistical information described above is to collect "raw" data from the computer system itself. Such data could include, for example, transaction type arrival rates, as well as the CPU and disk resources required by each type of transaction. This data is then used to define a set of input parameters for a discrete event simulation which has been instrumented to collect the necessary statistics without interfering with performance of the host system. Although the computer performance evaluation method of the present invention could be implemented by deriving the requisite information from data compiled by applications programs, the description herein assumes operational data has been gathered from the results of a simulation of the host system.

Alternatively, the host computer system to be evaluated in accordance with the present invention could be modeled using a modeling program such as, for example, BEST/1 available from BGS, Inc., of Waltham, Massachusetts. A specific example of the manner in which the BEST/1 program may be utilized to model a host computer system consisting of CPUs, controllers, and disks is set forth below. In particular, the modeling program is employed by:

(1) defining a set of components corresponding to the components of the host system;

(2) specifying operative connections between the simulated components;

(3) specifying that operational data relating to, for example, the time during which each component was engaged in processing a particular service request, the average rate at which service requests arrive at each component, etc.; and

(4) supplying the modeling program with a set of transaction requests, the processing of which will yield the desired operational data.

System Organization

Referring to FIG. 8, there is shown a context diagram representative of one manner in which a graphical display of performance statistics is provided by processing the collected operational data in accordance with one embodiment of the present invention. As is well known in the art, graphics routines capable of generating curves, text and bitmaps exist in the form of a number of commercially available software packages. Accordingly, the following discussion relating to implementation of the inventive graphical user interface assumes familiarity with the capabilities of such packages.

As shown in FIG. 8, a Model Playback routine 100 is disposed to transform operational data (collected as described above) into performance statistics to be displayed using a graphics driver package 104 such as, for example, Object Graphics", developed by The Whitewater Group. The graphics driver 104 supplies the information required to update a user display 106. The Model Playback routine 100 also serves to

initiate various Output Events 108 which, for example, allow for the creation and manipulation of interface "windows" using the display 106.

Referring to FIG. 8, the Model Playback routine 100 retrieves information from data files denoted as Transactions 112, Options 114, Configurations 116, Server 118, Models 120, and Bitmaps 122 during the course of responding to input events from the user input environment 102. The contents of each of the data files 112-122 are described as follows in order to provide a basis for discussion of the operation of the Model playback routine 100.

Data Structure

Referring to FIG. 9, the Transactions file 112 contains operational data corresponding to each type of transaction requested via the user input environment 102, and will generally be created during simulated operation of the host computer. Alternatively, the Transactions file 112 may be established as operational data is collected in real time from the host computer. In the latter case the applications programs running on the host computer would be instrumented in the manner described above to facilitate real-time data acquisition. However, in the following description it is assumed that operational data is accumulated within the Transaction file during simulation of the host computer.

The time period over which each simulation of the host system extends may be divided into a set of consecutive batch intervals. Operational data accumulated during each batch interval 200 is written a corresponding batch record 204 at the conclusion of each batch interval. The batch record 204 stores operational data from each type of transaction executed during the particular batch interval. Each type of transaction is identified by a particular workload name, i.e., WKLD_NAME, where NAME specifies the transaction type. For example, host computer systems utilized within a banking environment could be responsive to DEBIT INQUIRY or CREDIT INQUIRY transaction types. In one embodiment shown in FIG. 9, for each transaction

type 206 the following information is stored as real values within the batch record:

WKLD_NAME - the name of the transaction type,
T_COMPLETE - the total number of transactions completed,
T_SIM_TIME - the time during the simulation corresponding to the end of the batch interval,
T_IO_COUNT - the average number of input/output (I/O) requests made of simulated components by transactions of the specified type,
T_IO_SERV - the average time spent by simulated components in responding to I/O requests made by transactions of the specified type,
T_IO_TIME - the average time taken in responding to I/O requests made by transactions of the specified type, where T_IO_TIME = T_IO_SERV + time elapsed during queuing of the I/O requests at the simulated components,
T_PROC_SERV - the average time spent by the simulated central processing unit (CPU) component of the host system in servicing I/O requests made by transactions of the specified type,
T_PROC_TIME - the sum of T_PROC_SERV and the average time elapsed during queuing of I/O requests directed to the simulated CPU,
T_CONTR_SERV - the average time spent by simulated controller components in responding to I/O requests made by transactions of the specified type,
T_CONTR_TIME - the sum of T_CONTR_SERV and the average time elapsed during queuing of I/O requests directed to simulated controller components,
T_UNIT_Q - the average time spent by simulated components requiring access to simulated disk components in queuing for such access.

In addition, the transaction denoted by each WKLD_NAME is defined within a workload type file. Again, the workload corresponding to a particular transaction consists of the sequence of service requests made to the various simulated

components of the host system during the course of the transaction.

Referring to FIG. 10, the Options file 114 contains various options selectable by a user. The Options file is bifurcated into sets of directory options 220 and threshold options 224. Directory options 220 enable a user to organize data into various directories that may represent different computer sites or different times of the day for a single site. The threshold options 224 allow a user to specify the manner in which display icons corresponding to particular simulated components will change in appearance as statistics relating to operation of the component vary with respect to predefined threshold values during the simulation. The display attributes for which a user has the option of specifying a real-valued threshold preference are listed below:

WKLD_Yellow - This option may be selected when the average time for processing a particular type of transaction is being displayed using a cartesian coordinate system. In this cartesian graph the horizontal axis corresponds to the simulated time and the vertical axis to the average delay for the transaction type. The value of the attribute WKLD_Yellow chosen by a user specifies the average delay level at which an initial yellow horizontal warning line is to be drawn in the displayed coordinate system.

WKLD_Red - This option may be selected when the average time for processing a particular type of transaction is being displayed using a cartesian coordinate system. In this cartesian graph the horizontal axis corresponds to the simulated time and the vertical axis to the average delay for the transaction type. The value of the attribute WKLD_Red chosen by a user specifies the average delay level at which a red intermediate horizontal warning line is to be drawn in the displayed coordinate system.

Server_Yellow - When icons corresponding to simulated server components are being displayed, the level to which an icon appears to be "filled" with a particular color is indicative of the average length of the queue of service requests for access to the server. If the average queue length is greater than the selected value of Server_Yellow, but less than or equal to the value of Server_Red, then the lower half portion of the displayed icon is filled with a yellow shading. For queue lengths less than the value of Server_Yellow the displayed icon is slightly filled, with the color green.

Server_Red - If the average queue length associated with a particular simulated server component is larger than the specified value of Server_Red, then the icon is nearly completely filled with a red shading.

As shown in FIG. 11, the configurations file 116 stores definitions of predefined groups, or "configurations", of sets of simulated components. For example, a simple host computer system could consist of a CPU, a memory module, a pair of controllers and a pair of associated disk storage units. The configuration defining this system would thus include six simulated components. Each set of simulated components defined within the configurations file 116 is identified by the following:

Configurations File Title - A character sequence uniquely identifying a particular set of simulated components.

Configuration description - Provides additional textual description of a Configurations File.

In one embodiment, for each type of component 240 included within a particular configuration the following information is stored in the form of character sequences, as well as in the form of real and integer data values:

CNFG_Name (character) - A name uniquely identifying the component within the configuration.

CNFG_Unique_ID (integer) - A number uniquely identifying the component within the configuration.

CNFG_Type (character) - Components can be of the type *processor, memory, controller,* or *disk*.

CNFG_Platform (character) - Components can be made to be compatible with a particular {operating system?}, i.e., computer platform, such as *A Series* or *OS1100*.

CNFG_Style (character) - This attribute identifies the style or model of a particular component. For example, a simulated CPU component may have been developed to model an "A16" processor. Moreover, the style attribute may serve as a key into a separate Style file 242 providing additional detail regarding the specific style.

CNFG_String (character) - Simulated controllers and disks are combined into sets of simulated components denoted as "strings". Each string will typically include a pair of controllers and several disks. The attribute CNFG_String identifies the string in which a controller or disk is included, and serves as key to a String file 246 in which are defined particular component strings.

Other character identifiers may be included in the configurations file 116 which reflect, for example, the cost of the physical component being simulated or the date on which the component was purchased.

In a particular embodiment the Server file 118 contains operational data corresponding to each server within the selected configuration. Each component within the configuration is associated with one such server. The Server file 118 will generally be created during simulated operation of the host computer, although it also may be established as operational data is collected in real time. Again, in the latter case the applications programs running on the host computer would be instrumented in the manner described above to facilitate real-time data acquisition. The following

description assumes that operational data is accumulated within the server file during simulation of the host computer. Operational data is collected from each server during each batch interval 260, and is written to the server file 118 at the conclusion of each such interval 260. The Server file 118 thus includes a separate batch record 264 for each server, where each batch record 264 includes the following data files:

S_Name (character) - this character sequence is identical to the component name defined within the configuration file CNFG_Name, and serves as a key into the corresponding component definition 240 within the configurations file 116 (FIG. 11).

S_Arrival_Rate (real) - this data value is indicative of the average rate at which service requests from components arrive at the server,

S_Avg_Serv (real) - this value corresponds to the average time spent by the server in processing service requests from components,

S_Avg_Util (real) - this corresponds to the percentage of the batch interval during which the server was occupied servicing requests,

S_Avg_Q (real) - this value corresponds to the average length of the queue of server requests at each server. For example, an average queue size of 1 implies that, on the average, a service request arriving at the server remained pending while the server processed one such prior request. S_Avg_Q has a value of two when each service request arriving at the server encounters an average of one prior request in the queue and one request being serviced.

Referring to FIG. 13, the Models file 120 includes information relating to models consisting of the transaction workloads associated with particular component configurations. Again, a transaction workload refers to the set of components of which requests for service are made during processing of the transaction. A Model Catalog 270 within the Models file 120

includes entries for a plurality of models. For each model, the following data is stored:

Model Name (character) - this is used to uniquely identify the collection of data associated with a model,
Model Description (character) - this provides additional text further describing the model
Configuration File Title (character) - this provides a unique name for the configuration within the model,
Workload File Title (character) - this uniquely identifies the workload within the model.

The Bitmaps file 122 includes information relating to the pixel arrangements, or "bitmaps", of the various display icons representative of simulated components. The model playback routine 100 supplies the bitmaps for icons representative of processors, memory modules, strings, controllers or disks to graphics software 104 in order to enable the display of each via terminal 106.

System Operation

FIGS. 14 through 20 are transition graphs which illustrate the manner in which operational data collected from either an actual or simulated host computer system is processed and displayed in accordance with a preferred embodiment of the invention. The transition graphs depicted in FIGS. 14 - 20 consist of data flow diagrams to which have been added additional notation indicative of the transfer of control between the processes engaged in transforming the collected operational data into performance statistics. In this regard the dashed lines between entities within FIGS. 8, and 14 - 20 indicate a transfer of control, while solid lines correspond to paths over which data is transferred.

The following discussion of the processes represented by the transition graphs of FIGS. 14 - 20 includes textual descriptions in the form of "pseudo code" describing the sequence of logic events involved in particular processes.

This pseudo code should not be construed as being representative of code corresponding to a specific programming language, but rather as corresponding to the underlying logical structure of the processes described hereinafter.

Referring to FIG. 14, the Model Playback routine 100 is functionally divided into operations including Transaction Playback 300 and Servers Playback 310. When the icon representative of the Model playback routine 100 is selected by a user, a display window corresponding to Transaction playback 300 appears on the display 106. From this window, the user may initiate the following Input Events:

- Step Forward
- Step Backward
- Forward
- Backward
- Pause
- Server Playback
- Exit

Each of these Input Events will be described in further detail with reference to the transition graphs of FIGS. 14 - 20.

Upon selection of Server Playback event through the window corresponding to Transaction Playback 300 (FIG. 14), a display window is opened and the Servers Playback operation 310 is initialized. The Input Events of Step Forward, Forward, Step Backwards, Backwards, and Pause may be selected through the Servers Playback display window. The selection of one of these events through the Servers Playback window is also communicated, as indicated by control transfer line 314, to the Transaction Playback operation 300 in order that it's window be updated to reflect such selections. As is described hereinafter, information from the Models file 120 will occasionally be transferred to Servers Playback 310 through the Transaction Playback operation 300 over data path line 318.

Referring to FIG. 15, Input Events selected by a user which result in utilization of the Transaction Playback operation 300 are first received by an Input Event Handler 350.

The Event Handler 350 then requests that an Initialization procedure 354 create a display window, commence a Process Transaction File operation 358, and initiate a Draw Background routine 362. More specifically, during selection of an Input Event the user is required to specify the particular name of a model included within the Models file 120. Based on this model name the Initialization procedure 354 reads the Models file 120 and determines, using the model file name, the set of transaction data 206 for that model that is stored within the Transactions file 112 required to be processed (FIG.9). Again, each set of transaction data 206 includes data for a particular transaction type accumulated during simulation of the model specified by the user. The Process Transaction File operation 358 transfers the appropriate set of transaction data 206 into an internal Trans Stats memory 366, the contents of which are utilized by the Draw Background routine 362, an Update Graph routine 370, and a Backup procedure 374.

Referring to FIG. 16, the Draw Background routine 362 includes a Draw Titles procedure 390 which utilizes the model name supplied by the Event Handler 350 to create titles for the Transaction Playback window. A Draw Axis Box procedure 394 accesses Trans Stats memory 366 upon initialization of Draw Background 362 to determine the maximum values of vertical and horizontal axes to the particular graph or chart to be displayed (see, e.g., FIG. 3). The Draw Axis Box procedure then sends box coordinates and other attributes (line width, color, etc.) to graphics software 104. The graphics software 104 sends the appropriate display commands to the software environment of the display 106. The display environment consists of interface software such as, for example, Microsoft Windows, capable of drawing graphical axes and the like on the display 106. A Draw Warning Line procedure 398 calls on the Options file 114 for the levels at which the red and yellow horizontal warning lines are to be drawn on the axis box. In addition, legend labels (e.g., see TRANS #1 and TRANS #2 in FIG. 3) are read from Trans Stats memory 366 and forwarded to the display environment via a Draw Legend procedure 402.

As shown in FIGS. 15 and 17, the Update Graph routine 370 is called upon to execute Step Forward commands received by the Input Event Handler 350. In particular, the Step Forward command results causes the graphs associated with each displayed transaction type to be drawn for a succeeding time epoch. Referring by way of example to FIG. 3, assume that drawing of the graphs associated with TRANS #1 and TRANS #2 has been halted at the simulation time of 32 seconds using the Pause command (described below). In this circumstance a Step Forward command causes the TRANS #1 and TRANS #2 graphs to be drawn for the next time epoch, i.e., between the simulated times of 32 and 64 seconds. Each graph may be considered as a *polyline*, or a set of line segments connecting multiple data points. Again referring to FIG. 17, a Process Batch Data procedure 420 retrieves data corresponding to TRANS #1 and TRANS #2 from Trans Stats memory 366 upon receipt of a Step Forward command. The Process Batch Data procedure 420 then determines the location and orientation of the line segments included within the polyline corresponding to each transaction type, and also computes the average value of each over the specified time epoch. This information is stored within Polyline/Averages memory 424, and is accessed by a Draw Polyline routine 428 in generating graphics data 104.

Referring to FIGS. 15 and 18, the Backup procedure 374 executes Step Backwards received from the Input Event Handler 350. The Backup procedure 374 effectively "erases" the last line segment included within the polyline for each transaction type. This process is effected using an Invalidate Previous Line procedure 440 operative to determine the coordinates of the line segments to be erased based on statistical data retrieved from Trans Stats memory 366. The procedure 440 then invalidates the specified line segment by calling the display platform such as windows, and indicating that the area near the line is invalid and must be redrawn. An Erase Last Segment procedure 444 generates graphics data 104 used to redraw the invalidated line segment using the same color as the background of the graphical display, thereby rendering the redrawn line segment imperceptible.

As shown in FIGS. 15 and 19, a Server Playback routine 460 is called upon to execute Server Playback events issued by the Input Event Handler 350. In particular, a Server Playback event is initiated by a user via the Control pull-down menu within the Transaction Playback window (see, e.g., FIG. 3). The Server Playback routine 460 is initiated by an initialization procedure 464 designed to transmit a model name from the Models file 120 corresponding to the server component specified by the user. The model name is used by a Process Server File procedure 468 to retrieve the contents of the corresponding Server file 118, which is then read by the Process procedure 468 and stored in the memory 472.

Again, each server may be classified as either a processor, memory, controller or disk. For servers of the type controller or disk the Server Stats memory includes identification of the component string to which the server belongs. As noted above, component strings consists of sets of operatively connected controllers and disks hierarchically represented by a string icon. Upon initialization of a Server Playback window 476 an Update Servers Display procedure 480 generates graphics data 104 used to display the icons corresponding to each string of simulated components (see, e.g., FIG. 5). In a preferred embodiment the graphics data may be displayed using, for example, graphics application software such as ObjectGraphics, produced by the Whitewater Group. This particular graphics software serves to create a window within a Microsoft Windows presentation environment, and retain an identifying window "handle". This handle may then be passed to various graphics routines so as to specify the location at which various graphical images are to be drawn within the window's "picture". The Microsoft Windows software includes routines for appropriately rescaling the window picture when it is moved, resized or otherwise altered within the display environment.

When a particular icon is selected by, for example, "double-clicking" the pointing device, icons representative of the disks and controllers in the string are substituted for the string icon. That is, the string icon is "exploded" into a set

of icons representative of its constituent components. Also included within this hierarchical display arrangement are "dialog boxes" (FIG. 7), created when the user double-clicks on the specific icon by a Show Server Stats routine disposed to provide statistics for individual string components. These dialog boxes may also be selected by double-clicking upon the icon corresponding to the selected component. The icons representative of a particular string component may be replaced by the single icon representative of the string by double-clicking a specified control on the pointing device.

When "Forward" is selected, Servers Playback successively executes the STEP FORWARD event until all batches are displayed. The BACKWARD selection is processed in a similar fashion. This is similarly implemented in the Transaction Playback process.

As is indicated within FIGS. 19 and 20, the Update Servers Display procedure 480 involves processing "bitmaps" representative of the particular server component for which performance statistics are to be displayed. Again, a bitmap is a description of how the video pixels are to be displayed. Each bitmap allows graphical display of the performance statistics relating to a particular simulated component, thereby enabling depiction of the manner in which unbalanced component utilization may undesirably lengthen transaction response time. More specifically, a Process Batch Data procedure 498 selects the statistics accumulated for each server component for the batch interval being displayed. A Choose Bitmaps procedure 502 selects the bitmap corresponding to the type of component represented by each server icon. For example, as shown in FIG. 6 there exists a separate bitmap for each Processor, Memory, Controller, Disk, and String to be represented. The Options file 114 is accessed by the Choose Bitmaps procedure 502 in order to determine the type of bitmap associated with a particular simulated component.

The bitmaps are arranged in an Icon Display memory 506 by an Update Icon Display procedure 510. The extent to which each icon is "filled" with a color indicative of the length of its queue of transactions is determined by comparing each batch of

server statistics to the warning and red options (Server_Yellow and Server_Red). When first initialized, the icons representative of each server are displayed in accordance with component type. Specifically, on a top display line a set of icons corresponding to processors followed by a set of memory icons are each displayed in alphabetical order. On a second line of the display there is included an alphabetical arrangement of the icons corresponding to a set of strings. For relatively complex host systems a user may have to scroll the display horizontally in order to view all of the string icons. To simplify the process of identifying "bottlenecks" within the host system, the user can select a Sort option from the Control menu within the Server Playback window. Selection of this option causes a Sort Display procedure 514 to sort the server statistics for each batch interval on the basis of queue length. Icons corresponding to strings are displayed on the basis of the maximum queue length associated with one of the components in the string. When a user elects to "explode" a string icon via the pointing device the icons representative of simulated controller components are displayed above those corresponding to disks via an Explode/Implode Icon Display Procedure 518. Again, the icons representative of the individual string components may be "imploded" into the single icon representative of the string by using the pointing device to transmit the appropriate command to the Icon Display procedure 518. Moreover, the icons within the sets of icons corresponding to controllers and disks are arranged by the Icon Display procedure 518 from left to right on the basis of queue length.

The scrolling event, that the user can specify at any time, is handled by Update Icon Display, which determines which icons will actually be displayed in the physical graph space. This process only updates icons that change. This provides a performance optimization to keep the display fast enough to be animated.

The Draw Servers Graph 540 is used to call the graphics package to display the selected bitmaps and their corresponding labels. In order to draw the set of icons the bitmap name as

well as a set of horizontal coordinates are passed into the appropriate graphics package.

Pseudo Code Description

The following sets of pseudocode are intended to be representative of the sequences of steps involved in implementing the processes described with reference to FIGS. 14 - 20. For example, it was mentioned above that the Input Event Handler 350 processes user events directed to the Transaction Playback operation 300. Accordingly, the pseudo code description of the Input Event Handler 350 is indicative of the manner in which particular Input Events (e.g., Step Forward, Step Backwards) are processed thereby.

```
Input Event Handler 350:
-case of Input Event
1. Initialize:
     Call Initialize (354)
          The Initialize process 354 initializes the
          display window and reads all necessary files.
          File statistics are stored in data structures
          within memory to provide quick response to user
          requests.  After processing of such file
          statistics is completed a background
          environment for a Transaction Playback graph
          (FIG. 3) is created.  This background includes
          a display box, vertical and horizontal axes,
          and a legend.
Step Forward
     Call Update Graph (370)
Step Backwards
     Call Backup (374)
Server Playback
     Call Server Playback (460)
Forward
     Loop until last batch processed or Pause Event
          Call Update Graph (370)
     endLoop
```

```
Backwards
     Loop until first batch processed or Pause Event
          Call Backup (374)
     endLoop
Left Mouse Double Click
     Call Trans Stats Dialog (366)
-endCase
2. Call Process Server File (468)
3. Initialize sort option to alphabetical
4. Initialize view rectangle to be the top left corner of
the potential display area
```

Process Trans File (358):

This process reads the transactions file 112 corresponding to a particular model being viewed. Data read from the transactions file 112 is then stored in memory-resident data structures

```
-for each time epoch
     read file record
     for each transaction type
          store statistics in Trans Stats
     endFor
-endFor
```

Draw Background (362)

This process creates background graphics for the "picture" portion of a display window.

```
1. Read Trans Stats (366) to determine the maximum X and
Y graphical coordinates and a Legend Label
2. Read Trans Stats (366) to determine names of
transaction types.
3. Read Options file (114) to determine the Y coordinate
for the red and yellow warning lines.
4. Set Model Name and Description in picture portion of
display window
5. Set Graph Box in picture portion of display window
```

6. Set A and Y rulers in picture portion of display window
7. Set Legend Labels in picture portion of display window
8. Set horizontal lines, including red and yellow warning lines, in picture portion of display window
9. Draw graphical data within picture portion of display window

Update Graph (370):

This process updates the graphics display of the line representing the internal response time of a particular transaction type. One line is drawn for each type of transaction.

-Retrieve data for next time epoch to be displayed from Trans Stats (366)
1. for each transaction type
    -add a new segment onto the polyline using the endpoint of the previous polyline, and increase the aggregate delay time for the transaction type by the delay associated with that transaction type within the new epoch
    -update weighted average delay for the transaction type by using the number of completions of the transaction type and the average delay time
    -store updated polyline and average data in memory
    -set polyline in picture portion of display window
    -set new average delay in picture portion of display window
endFor

Backup (374):

This process "erases" the last line segment drawn within the picture portion of the display window. Each time this process is invoked the last line segment on the specified polyline is erased from the video screen and the corresponding average value of the data represented by the displayed line is updated accordingly.

1. Using the polyline stored in memory, remove the last segment set in the picture portion of the display window
2. Call a Windows routine to invalidate the region of the picture portion which included the removed line segment.

Trans Stats Dialog (530):

This process is called when a user double-clicks on the left button of the pointing device, i.e., on the "mouse"

1. Determine the display coordinates of the double-click
2. Determine the transaction type and time epoch of the polyline segment nearest the double-clicked display coordinates.
3. Retrieve statistics relating to the specified transaction and time epoch. Send the retrieved statistics to a dialog box (532) created within the display window.

Process Server File (468):

This process reads the servers file 118 and the configuration file 116 and retrieves information relating to the specified model defined within the models file 120. Based upon the retrieved configuration information, sets of disks and controllers are grouped into component strings.

```
1. for each time epoch
        read file record
        for each server
            store statistics in Server Stats
        endFor
endFor
```

2. Read Configuration file records
3. Determine the component type (CPU, Memory, Unit, Controller) and membership to strings for server whose statistics were gathered during the simulation.
4. Save data in Server Stats

Update Servers Display (480)

This process determines which servers are to be represented as icons within a display window. Since the display window may only accommodate a given number of server

icons, horizontal and vertical scrolling may be necessary in order to view all of the icons. A user may also specify that the server icons be arranged in a particular format, e.g., alphabetically or in accordance with queue size. A double click of the left mouse button upon a server icon representing a CPU, Memory, or Controller component produces a dialog box with the statistics for that component.

For each server icon to be displayed, a bitmap is chosen based on the type of server and the length of its queue. For example, assume that it has been specified within the Options file 114 that the Server Yellow threshold has a value of 0.5 and that the Server Red threshold has a value of 1. Also assume that for the particular time epoch to be displayed the component to be represented is a CPU having a queue length of 0.75. In this case a yellow CPU server icon will be displayed. For string icons, the maximum queue length associated with one of the units or controllers in the string is used to determine the attributes (e.g., color) of the displayed icon.

The efficiency of the display process may be enhanced by minimizing the quantity of graphics information required to be updated on the video screen. In particular, an icon used in a particular display is saved in memory so as to be potentially available for use when the display is refreshed. When it is determined which icons are to be next displayed, the saved display data is examined to ascertain which, if any, changes are required to be made to the saved data prior to updating the video screen.

A double-click of the left mouse button on a string icon, representative of a set of disk and associated controller components, results in replacement of the string icon by icons representing its component disk and controller units (i.e., the string icon is "exploded" into its constituent elements. A right mouse button click implodes the disk and controller units back into the corresponding string icon.

1. case of Input Event

Step Backwards or Step Forward

```
                Call Process Batch Data (498)
                Call Choose Bitmap (502)
                Call Update Icon Display (510)
                Call Sort Display (514)
                Call Draw Servers Graph (540)
          Double Click/Right Click
                Call Explode/Implode Icon Display (518)
          Scroll Event
                Use scrolling values from within the display
                environment (e.g., Microsoft Windows) to
                determine the dimensions of a rectangular
                portion of global display that is to actually
                be displayed within the window.
                Call Update Icon Display (510) with this new
                rectangle.
          Sort Option
                Call Sort Display (514) with the selected
                option
endCase
```

Process Batch Data (498):

This process provides the data to be displayed during a particular epoch for a specified server component.

1. if Step Forward, get data for next epoch
   else if Step Backwards get data for previous epoch

Choose Bitmaps (502):

In a preferred embodiment there exist fifteen bitmaps, corresponding to "green", "yellow" and "red" versions of the CPU, Memory, String, Controller and Unit icons.

1. for each component to be displayed in the epoch
   if component Type is String
      queue length is the maximum of the queue lengths for the member units and controllers
   else queue length is the queue length for the server

```
if queue length is less than server yellow option,
     select the green icon for the component type
else
     if queue length is less than red and greater
     than or equal to the yellow option
          select the yellow icon for the component
          type
     else select the red icon for the component type
```

Update Icon Display (510):

This process creates the current display by using the stored data relating to icons displayed for the previous epoch, along with data corresponding to new icons selected for the current epoch which were not included in the previous epoch. A flag is associated with each new icon. The display coordinates corresponding to a viewing window are stored in the Icon Display 506.

```
1. for each server
     if Sort Option is by queue length
          Call Sort Display (514)
     else
          determine coordinates for the bitmap in the
          world display
          compare previous icon to current epoch's icon
          at same world display position
          if changed, update Icon Display (506) and set
          the update flag
   endFor
```

2. If a Scroll Event (550) has occurred, use the view rectangle to determine the new coordinates for the icons to be included within the display window.

Draw Servers Graph (540):

This process uses the bitmap descriptions and the coordinates to update the video display. Only icons that are indicated for update are actually drawn.

```
1. for each icon
```

```
            if coordinates are within the view display rectangle
            and icon is indicated as updated
                 Call the graphics package with the bitmap names
                 and display coordinates
          endFor
```

Explode/Implode Icon Display (518):

This process changes a String into its component Unit and Controller bitmaps ("explode") and changes the Unit and Controller bitmaps into the String ("Implode")

```
     1. Determine coordinate of mouse click
     2. if double left mouse click
            if coordinates correspond with the display of a CPU,
            Memory, Unit or Controller icon
                 Call Server Stats Dialog (2.4.7)
            else if coordinates correspond with a string icon
                 Replace String Icon in the Icon Display with
                 the Controller and Unit icons that belong to
                 the string
          else if right mouse click and coordinates correspond to
     a Unit or Controller
                 Replace all controllers and unit icons in the
                 Icon Display for the Controller/Unit's
                 string by the corresponding string icon.
```

Sort Display (514):

This process sorts the set of server icons to be displayed. If alphabetical sort is chosen by the user, the icons representative of CPUs are sorted by alphabetical order and placed left-most on the top row. The Memory icons are then sorted in alphabetical order and placed to the right of the CPU on the top row. Strings are sorted by alphabetical order and placed on the second row. For any exploded string, the controllers are sorted by alphabetical order and placed in the first row of the location previously occupied by the now-exploded string icon. The disk units within the string are then sorted by alphabetical order and placed on the rows beneath the controller row. The ordering is in two columns,

with disk units in the same row having a higher order than units in successive rows.

If Sort by queue length is selected, the icons are arranged in the manner described above with the exception that queue length, rather than alphabetical order, is used in determining the ranking order of the icons.

```
1. for each server
     if Sort Alphabetically
          add server statistics to sort collection using
                    the server name for rank
     else
          add server statistics to sort collection using
          queue length for rank
          for each component type, keep count of the
          number of components of that type
     endif

2. for each server in sort collection
     if CPU
          set up world coordinates to be to the right of
          the previous CPU
     else
          if Memory set up world coordinates to be to the
          right of the previous Memory
          else if String and imploded
               set up world coordinates to be to the
               right of the previous String
          else if String and exploded
               for each member of the String
                    add server to sort collection
               for each member of sorted collection
                    if Controller and one controller
                    displayed in row
                         set up coordinates to the right
                         of the previous controller
                    else if Controller and two
                    controllers displayed in row
```

```
                    set up coordinates to the next
                    row below controllers
                else if Unit and one unit displayed
                in row
                    set up coordinates to the right
                    of the previous unit
                else if Unit and two units displayed
                in row
                    set up coordinates to the next
                    row below units to display
                    exploded String
            endFor
```

All publications and patent applications mentioned in this specification are herein incorporated by reference to the same extent as if each individual publication or patent application was specifically and individually indicated to be incorporated by reference.

The invention now being fully described, it will be apparent to one of ordinary skill in the art that many changes and modifications can be made thereto without departing from the spirit or scope of the appended claims.

WHAT IS CLAIMED:

1. A method of analyzing the operation of a computer system using a data processor, said computer system including a plurality of components disposed to process various transaction types, comprising the steps of:

collecting data relating to operation of selected ones of said components by monitoring said computer system;

transforming said collected data into a set of simulation statistics to be provided to a simulation model of said computer system;

formulating said simulation model of said computer system by programming said data processor, said simulation model including a set of simulated components;

applying to said simulation model said set of simulation statistics in order to generate a simulation output; and

producing a display through a user interface based on said simulation output, said display being produced so as to convey performance statistics pertaining to processing of said transaction types.

2. The method of Claim 1 wherein said step of producing a display includes the step of simultaneously displaying first and second sets of performance statistics, said first set of statistics pertaining to one of said transaction types and said second set of statistics pertaining to at least one of said simulated components.

3. The method of Claim 1 wherein said performance statistics comprise response time of said computer system in processing at least one of said transaction types.

4. The method of Claim 3 wherein said performance statistics comprising said response time include statistics relating to utilization of at least one of said simulated components.

5. The method of Claim 4 wherein said performance statistics comprising said response time further include

statistics relating to queuing time associated with at least one of said simulated components.

6. The method of Claim 1 wherein said performance statistics include first and second sets of performance statistics corresponding to first and second of said transaction types, respectively.

7. The method of Claim 1 wherein said step of producing a display includes the step of simultaneously displaying:

(i) response time information associated with said processing of a selected one of said transaction types by said computer system, and

(ii) information pertaining to utilization of selected ones of said components of said computer system during said processing of said selected transaction type.

8. The method of Claim 1 wherein said components of said computer system are organized into strings, and wherein said step of producing a display includes the step of displaying information pertaining to utilization of components within a selected one of said strings during processing of a selected one of said transaction types.

9. The method of Claim 8 wherein said component utilization information includes service request arrival rate and service request queuing time information associated with one of said components within said selected string.

10. The method of Claim 1 wherein said step of producing a display includes the step of displaying icons representative of component strings, said component strings corresponding to interrelated sets of said components within said computer system.

11. The method of Claim 10 wherein said step of displaying string icons includes the step of varying appearance

of said icons as a function of utilization of said interrelated sets of components represented by said icons.

12. The method of Claim 11 wherein said step of displaying string icons includes the step of displaying component icons representative of components within one of said component strings by selecting said string icon representative of said one component string.

13. The method of Claim 12 wherein said step of displaying said component icons includes the step of varying appearance of said component icons as a function of utilization of said components represented thereby.

14. The method of Claim 13 wherein said step of producing a display includes the step of graphically displaying said performance statistics over a set contiguous time epochs of said simulation.

15. The method of Claim 14 wherein said step of graphically displaying includes the step of advancing a graphical display of said performance statistics over specified ones of said contiguous time epochs.

16. The method of Claim 15 further including the step of reversing said graphical display of performance statistics over said specified ones of contiguous time epochs so as to erase said graphical display.

AMENDED CLAIMS

[received by the International Bureau on 17 February 1994 (17.02.94); original claims 17-32 cancelled; original claims 1, 8-13 and 16 amended; other claims unchanged (4 pages)]

1. A method of analyzing the operation of a computer system using a data processor, said computer system including a plurality of components disposed to process various transaction types, comprising the steps of:

collecting data relating to operation of selected ones of said components by monitoring said computer system;

transforming said collected data into a set of simulation statistics to be provided to a simulation model of said computer system;

formulating a plurality of said simulation models of said computer system by programming said data processor, each of said simulation models including a set of simulated components;

applying to each of said simulation models said set of simulation statistics in order to generate a plurality of simulation outputs, each consisting of a set of measures, each measure being saved as a time series showing the evolution of the measure throughout the duration of the simulation, where each set of measures is associated with one of said simulation models;

producing a display through a user interface based on said plurality of stored simulation outputs, said display being produced so as to convey the stochastic variability of performance statistics pertaining to processing of said transaction types, said display allowing a user to determine and compare the values and variability of said performance statistics based upon differences among various ones of said plurality of stored simulation outputs.

2. The method of Claim 1 wherein said step of producing a display includes the step of simultaneously displaying first and second sets of performance statistics, said first set of statistics pertaining to one of said transaction types and said second set of statistics pertaining to at least one of said simulated components.

3. The method of Claim 1 wherein said performance statistics comprise response time of said computer system in processing at least one of said transaction types.

4. The method of Claim 3 wherein said performance statistics comprising said response time include statistics relating to utilization of at least one of said simulated components.

5. The method of Claim 4 wherein said performance statistics comprising said response time further include statistics relating to queuing time associated with at least one of said simulated components.

6. The method of Claim 1 wherein said performance statistics include first and second sets of performance statistics corresponding to first and second of said transaction types, respectively.

7. The method of Claim 1 wherein said step of producing a display includes the step of simultaneously displaying:

(i) response time information associated with said processing of a selected one of said transaction types by said computer system, and

(ii) information pertaining to utilization of selected ones of said components of said computer system during said processing of said selected transaction type.

8. The method of Claim 1 wherein said components of said computer system are organized into strings, each string including one or more functionally interrelated components and wherein said step of producing a display includes the step of displaying information pertaining to utilization of components within a selected one of said strings during processing of a selected one of said transaction types.

9. The method of Claim 8 wherein said component utilization information includes service request arrival rate

and service request queuing time information associated with one of said components within said selected one of said strings.

10. The method of Claim 1 wherein said step of producing a display includes the step of displaying string icons representative of component strings, said component strings corresponding to interrelated sets of said components within said computer system.

11. The method of Claim 10 wherein said step of displaying string icons includes the step of varying appearance of said string icons as a function of utilization of said interrelated sets of components represented by said string icons.

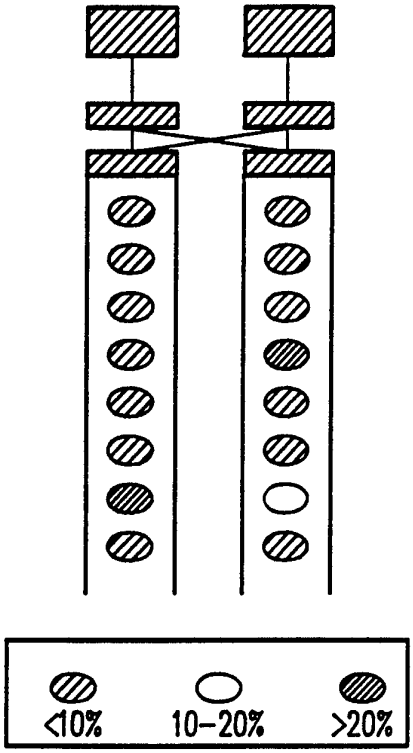
12. The method of Claim 11 wherein said step of displaying string icons includes the step of displaying component icons representative of components within a selected one of said component strings by selecting said string icon representative of said selected one of said component strings.

13. The method of Claim 12 wherein said step of displaying said component icons includes the step of varying the appearance of said component icons as a function of utilization of said components represented thereby.

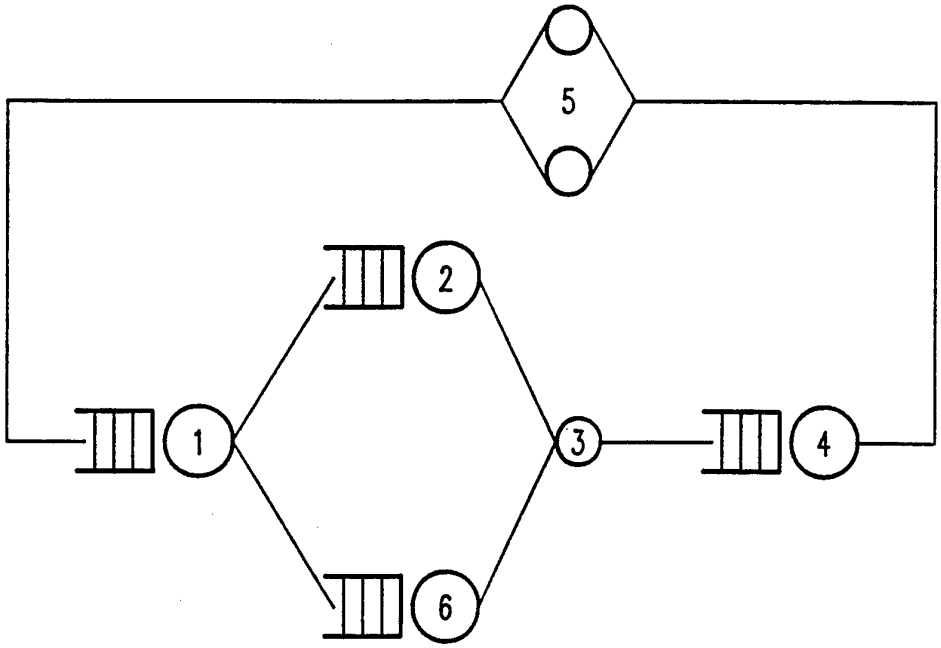
14. The method of Claim 13 wherein said step of producing a display includes the step of graphically displaying said performance statistics over a set contiguous time epochs of said simulation.

15. The method of Claim 14 wherein said step of graphically displaying includes the step of advancing a graphical display of said performance statistics over specified ones of said contiguous time epochs.

16. The method of Claim 15 further including the step of altering a portion of said graphical display of performance statistics over said specified ones of contiguous time epochs so as to erase said portion of said graphical display.



PRIOR ART
FIG. 1



PRIOR ART
FIG. 2

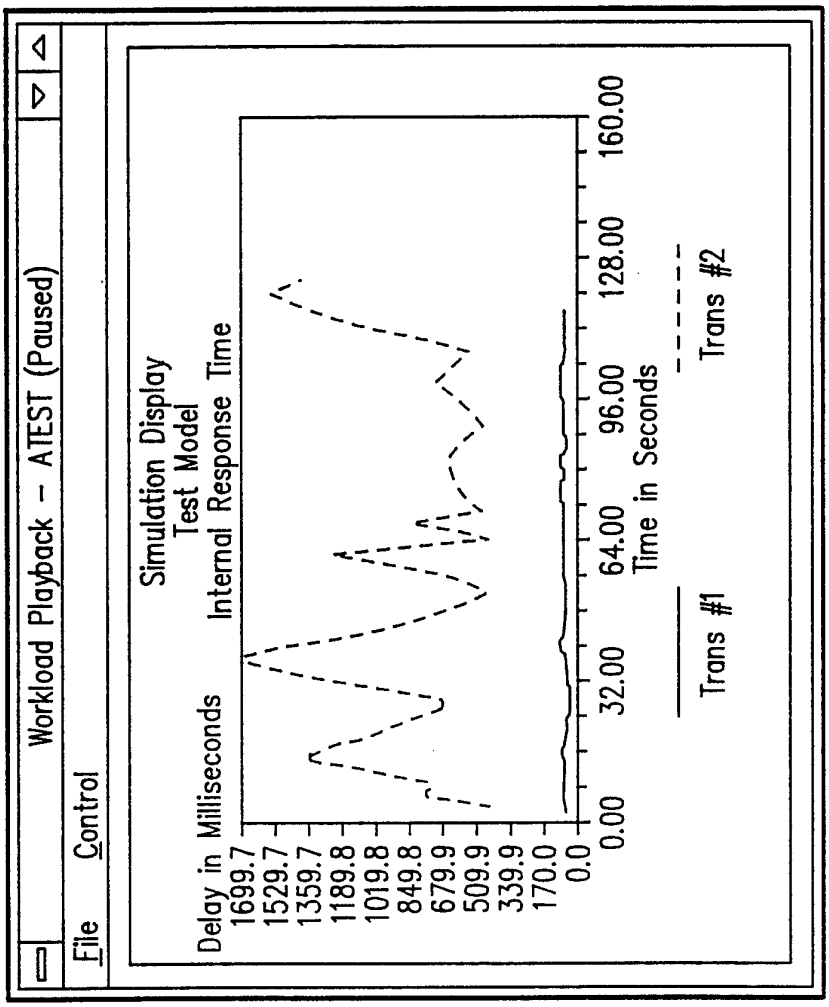

Workload Playback - ATEST (Paused)
File Control
Simulation Display
Test Model
Internal Response Time
Delay in Milliseconds
1699.7
1529.7
1359.7
1189.8
1019.8
849.8
679.9
509.9
339.9
170.0
0.0
0.00
32.00
64.00
96.00
128.00
160.00
Time in Seconds
Trans #1
Trans #2

FIG. 3

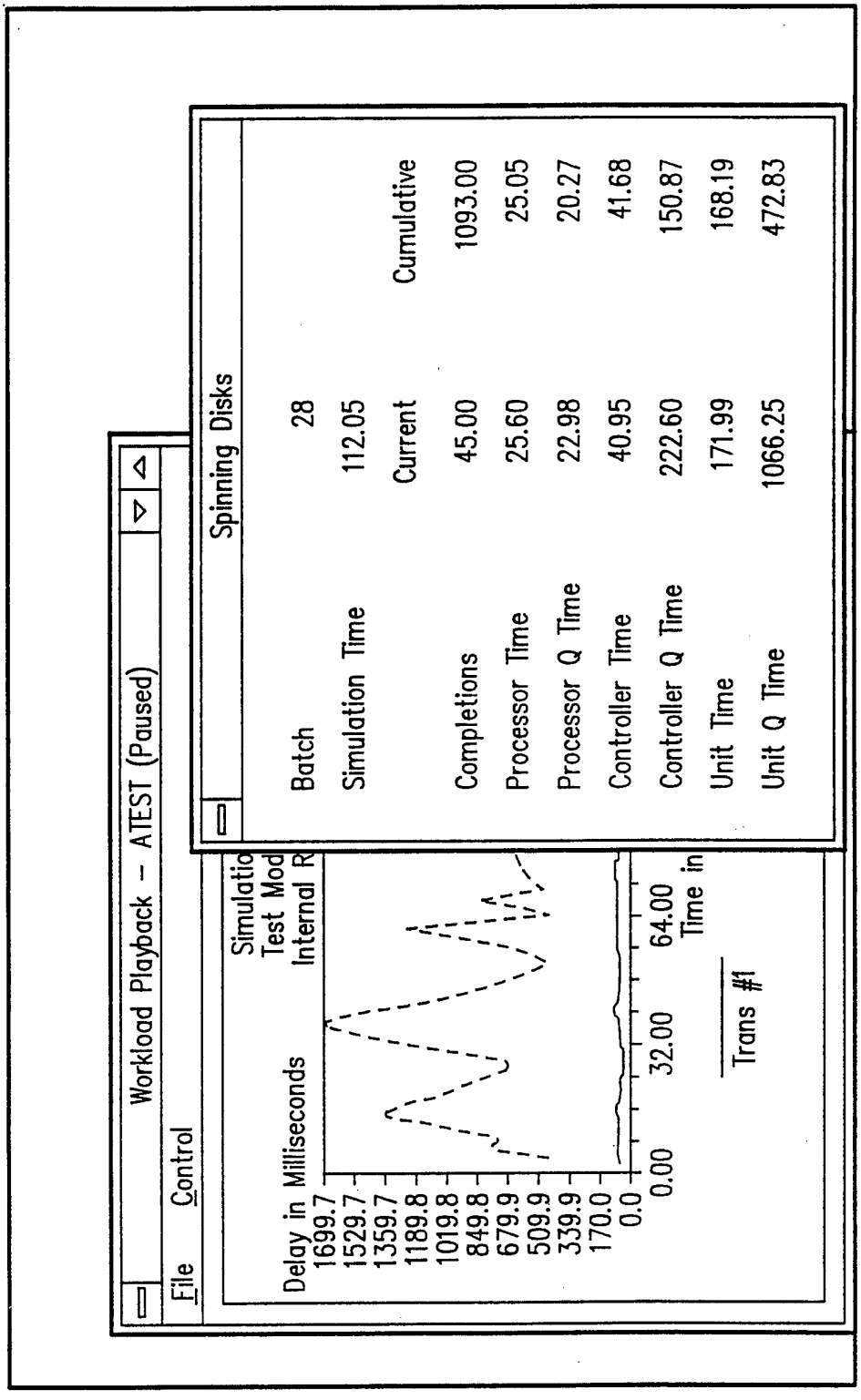

Workload Playback - ATEST (Paused)
File Control
Spinning Disks
Batch 28
Simulation Time 112.05
Current Cumulative
Completions 45.00 1093.00
Processor Time 25.60 25.05
Processor Q Time 22.98 20.27
Controller Time 40.95 41.68
Controller Q Time 222.60 150.87
Unit Time 171.99 168.19
Unit Q Time 1066.25 472.83
Simulatio
Test Mod
Internal R
Delay in Milliseconds
1699.7
1529.7
1359.7
1189.8
1019.8
849.8
679.9
509.9
339.9
170.0
0.0
0.00
32.00
64.00
Time in
Trans #1

FIG. 4

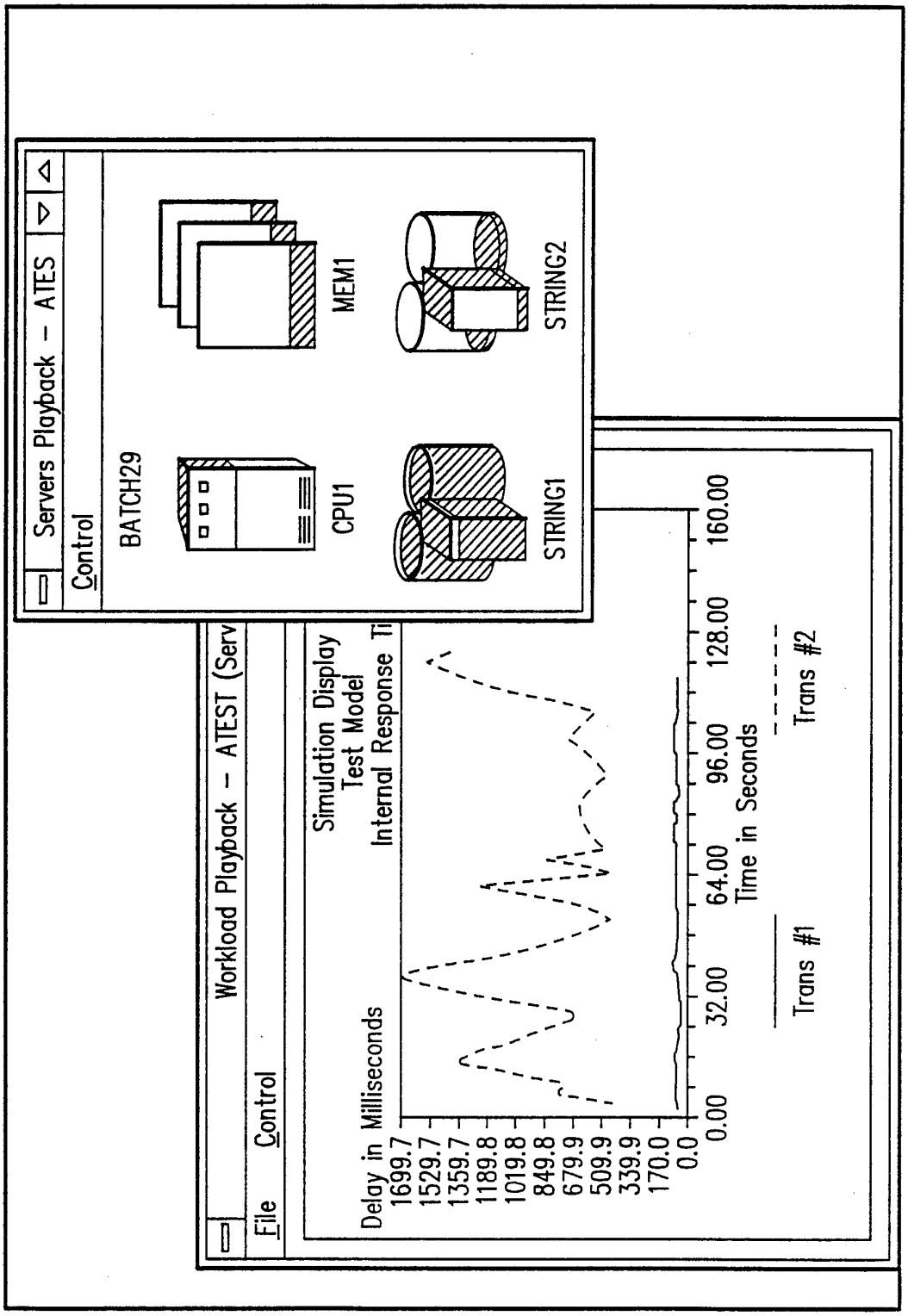

Servers Playback - ATES
Control
BATCH29
MEM1
STRING2
CPU1
STRING1
Workload Playback - ATEST (Serv
File Control
Simulation Display
Test Model
Internal Response Ti
Delay in Milliseconds
1699.7
1529.7
1359.7
1189.8
1019.8
849.8
679.9
509.9
339.9
170.0
0.0
0.00
32.00
64.00
96.00
128.00
160.00
Time in Seconds
Trans #1
Trans #2

FIG. 5

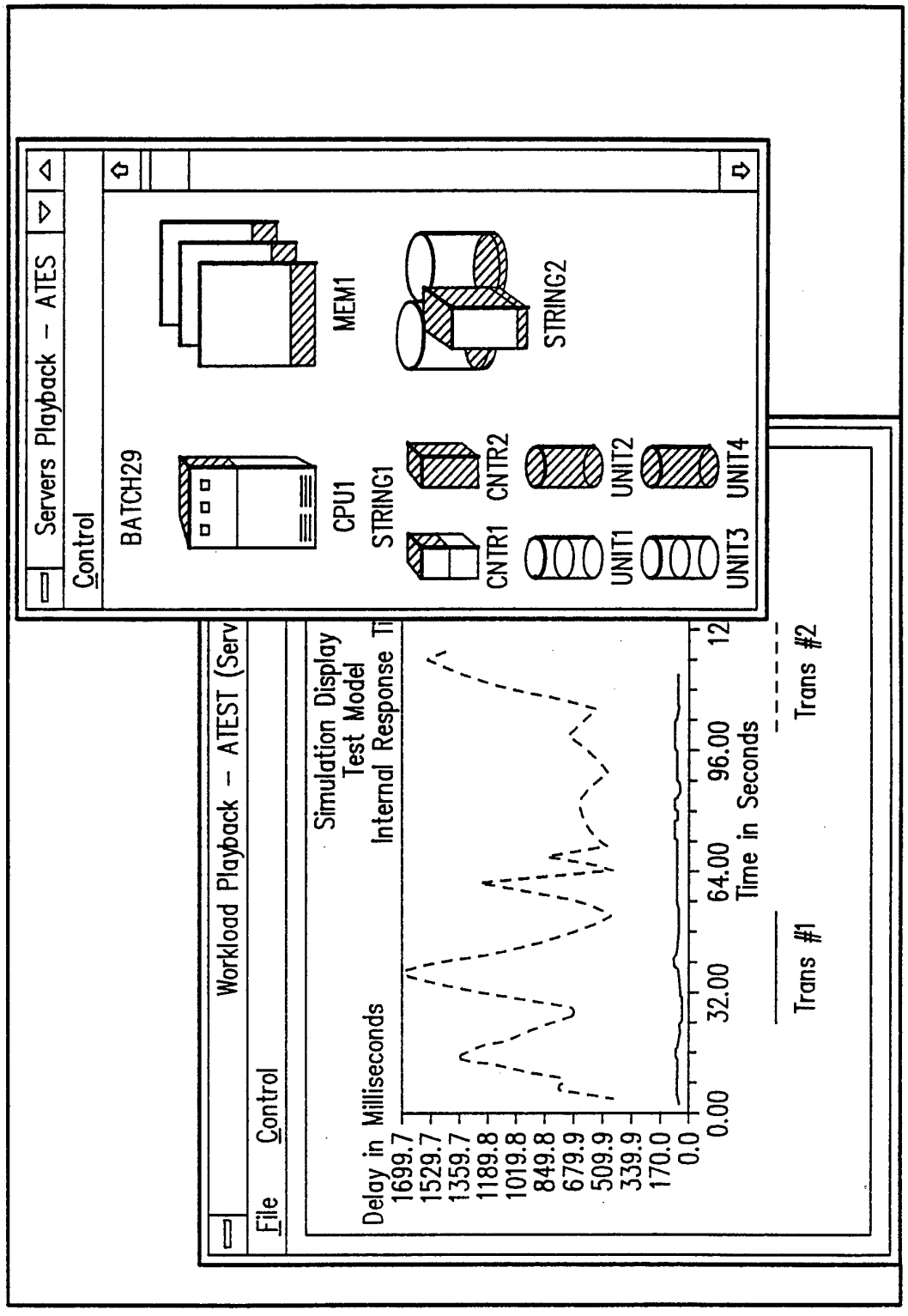

Servers Playback - ATES
Control
BATCH29
MEM1
STRING2
CPU1
STRING1
CNTR1
CNTR2
UNIT1
UNIT2
UNIT3
UNIT4
Workload Playback - ATEST (Serv
File Control
Simulation Display
Test Model
Internal Response Ti
Delay in Milliseconds
1699.7
1529.7
1359.7
1189.8
1019.8
849.8
679.9
509.9
339.9
170.0
0.0
0.00
32.00
64.00
96.00
12
Time in Seconds
Trans #1
Trans #2

FIG. 6

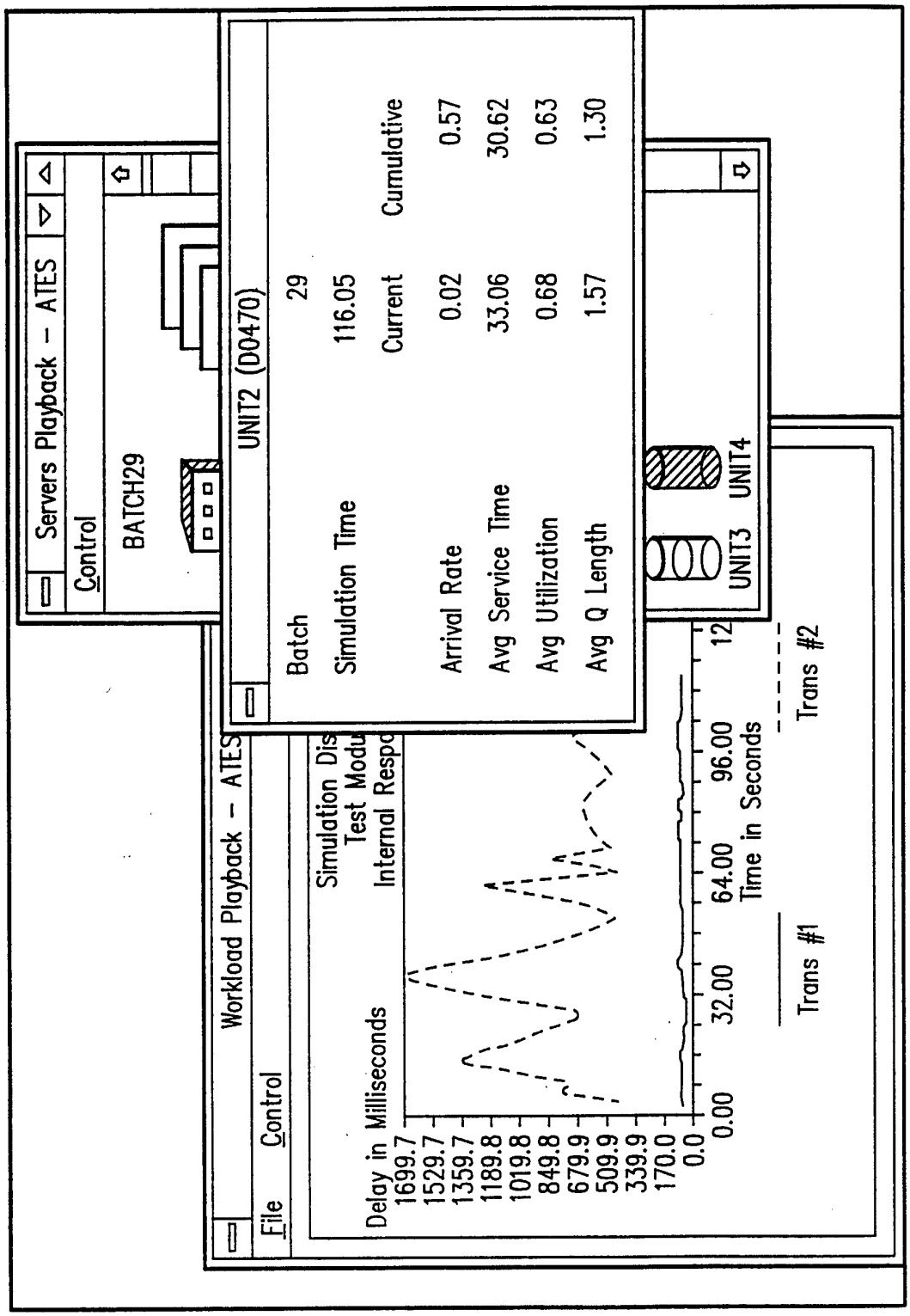

Servers Playback - ATES
Control
BATCH29
UNIT2 (D0470)
Batch
29
Simulation Time
116.05
Current
Cumulative
Arrival Rate
0.02
0.57
Avg Service Time
33.06
30.62
Avg Utilization
0.68
0.63
Avg Q Length
1.57
1.30
UNIT3
UNIT4
Workload Playback - ATES
File
Control
Simulation Dis
Test Modu
Internal Respo
Delay in Milliseconds
1699.7
1529.7
1359.7
1189.8
1019.8
849.8
679.9
509.9
339.9
170.0
0.0
0.00
32.00
64.00
96.00
12
Time in Seconds
Trans #1
Trans #2

FIG. 7

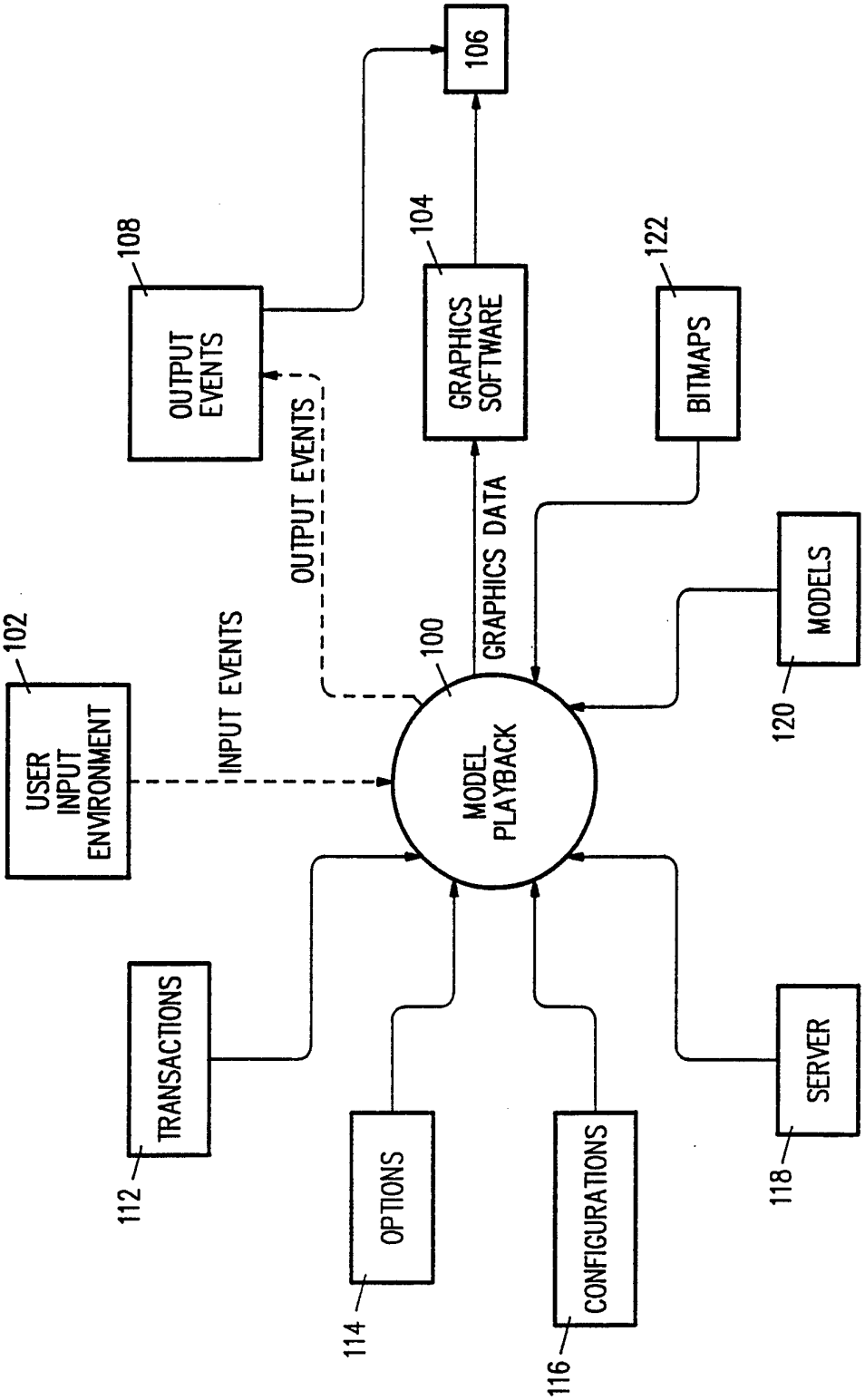

106
108
OUTPUT EVENTS
104
GRAPHICS SOFTWARE
122
BITMAPS
OUTPUT EVENTS
GRAPHICS DATA
102
USER INPUT ENVIRONMENT
INPUT EVENTS
100
MODEL PLAYBACK
MODELS
120
TRANSACTIONS
112
OPTIONS
114
CONFIGURATIONS
116
SERVER
118

FIG. 8

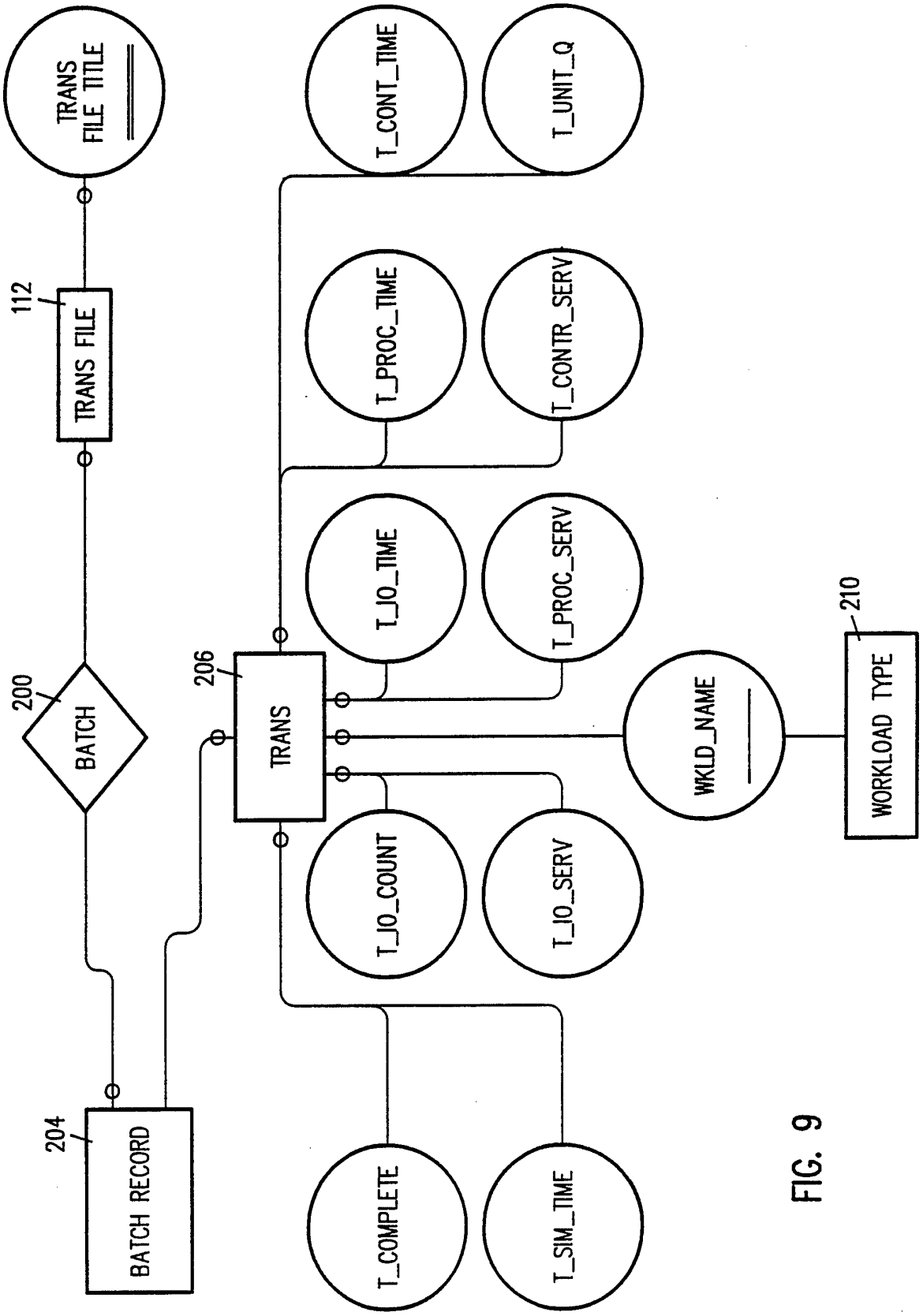

TRANS FILE TITLE
112
TRANS FILE
200
BATCH
204
BATCH RECORD
206
TRANS
T_CONT_TIME
T_UNIT_Q
T_PROC_TIME
T_CONTR_SERV
T_IO_TIME
T_PROC_SERV
T_IO_COUNT
T_IO_SERV
T_COMPLETE
T_SIM_TIME
WKLD_NAME
210
WORKLOAD TYPE

FIG. 9

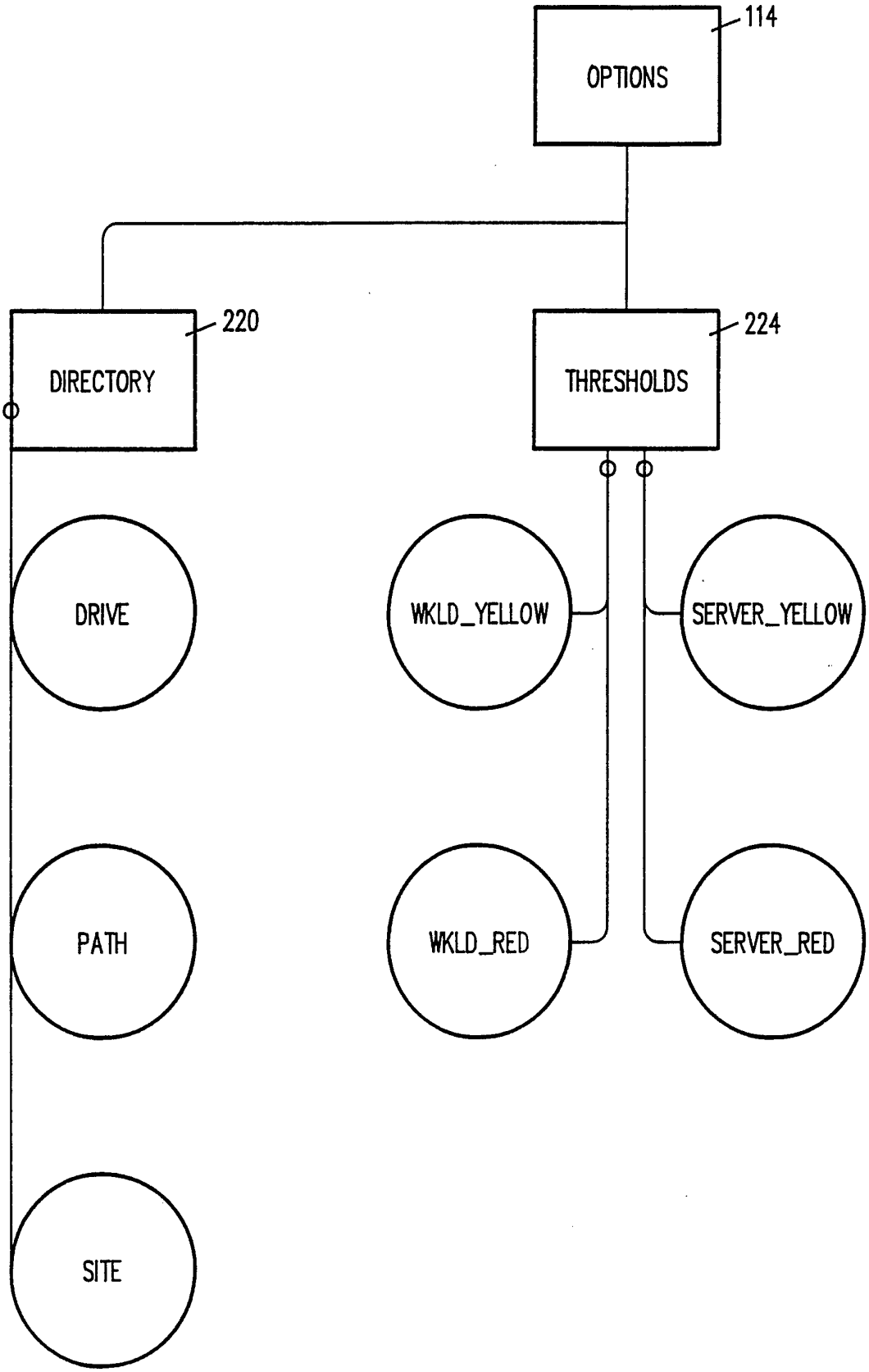

OPTIONS
114
DIRECTORY
220
THRESHOLDS
224
DRIVE
PATH
SITE
WKLD_YELLOW
SERVER_YELLOW
WKLD_RED
SERVER_RED

FIG. 10

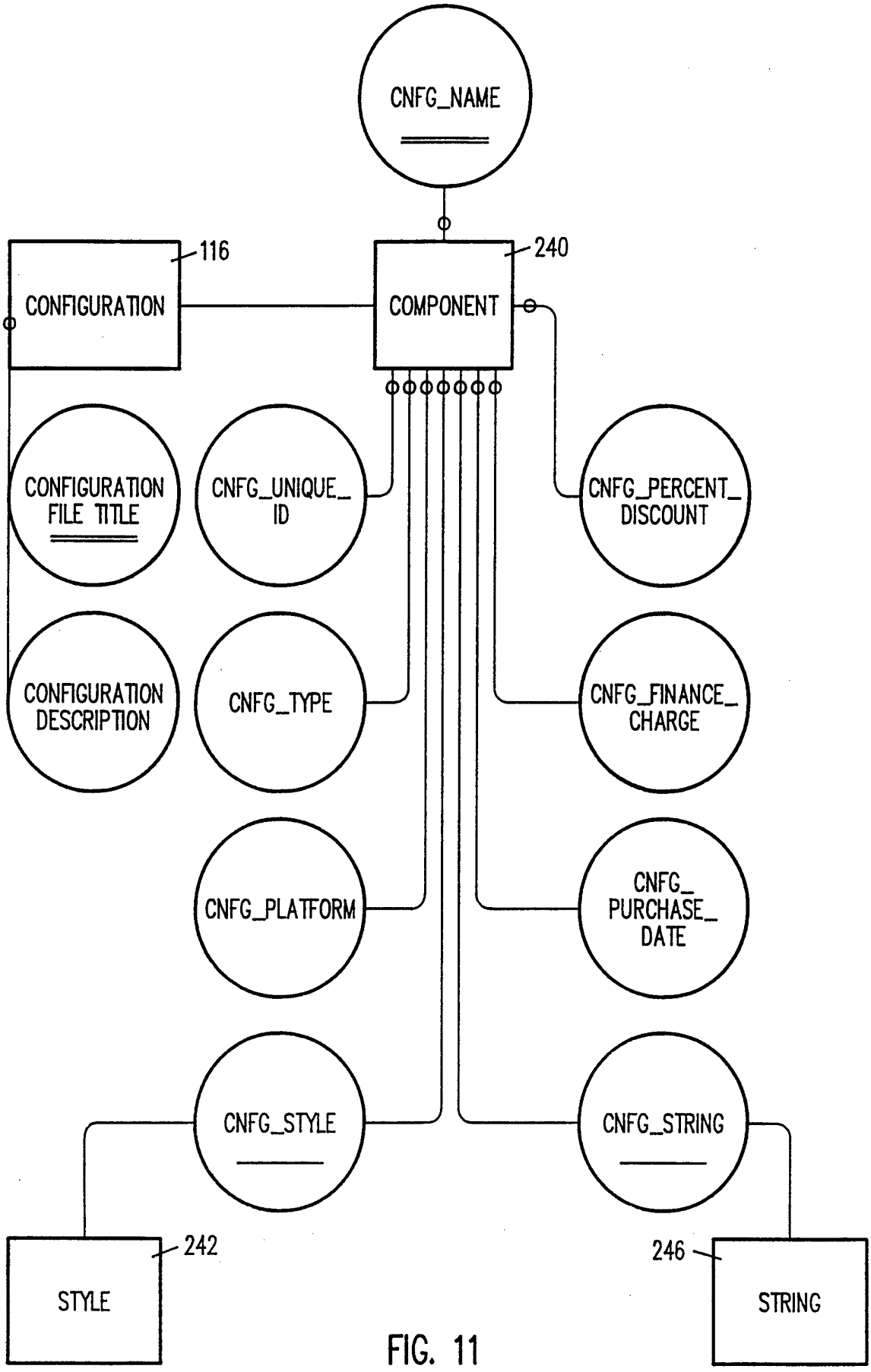

CNFG_NAME
CONFIGURATION
116
COMPONENT
240
CONFIGURATION FILE TITLE
CNFG_UNIQUE_ID
CNFG_PERCENT_DISCOUNT
CONFIGURATION DESCRIPTION
CNFG_TYPE
CNFG_FINANCE_CHARGE
CNFG_PLATFORM
CNFG_PURCHASE_DATE
CNFG_STYLE
CNFG_STRING
STYLE
242
246
STRING

FIG. 11

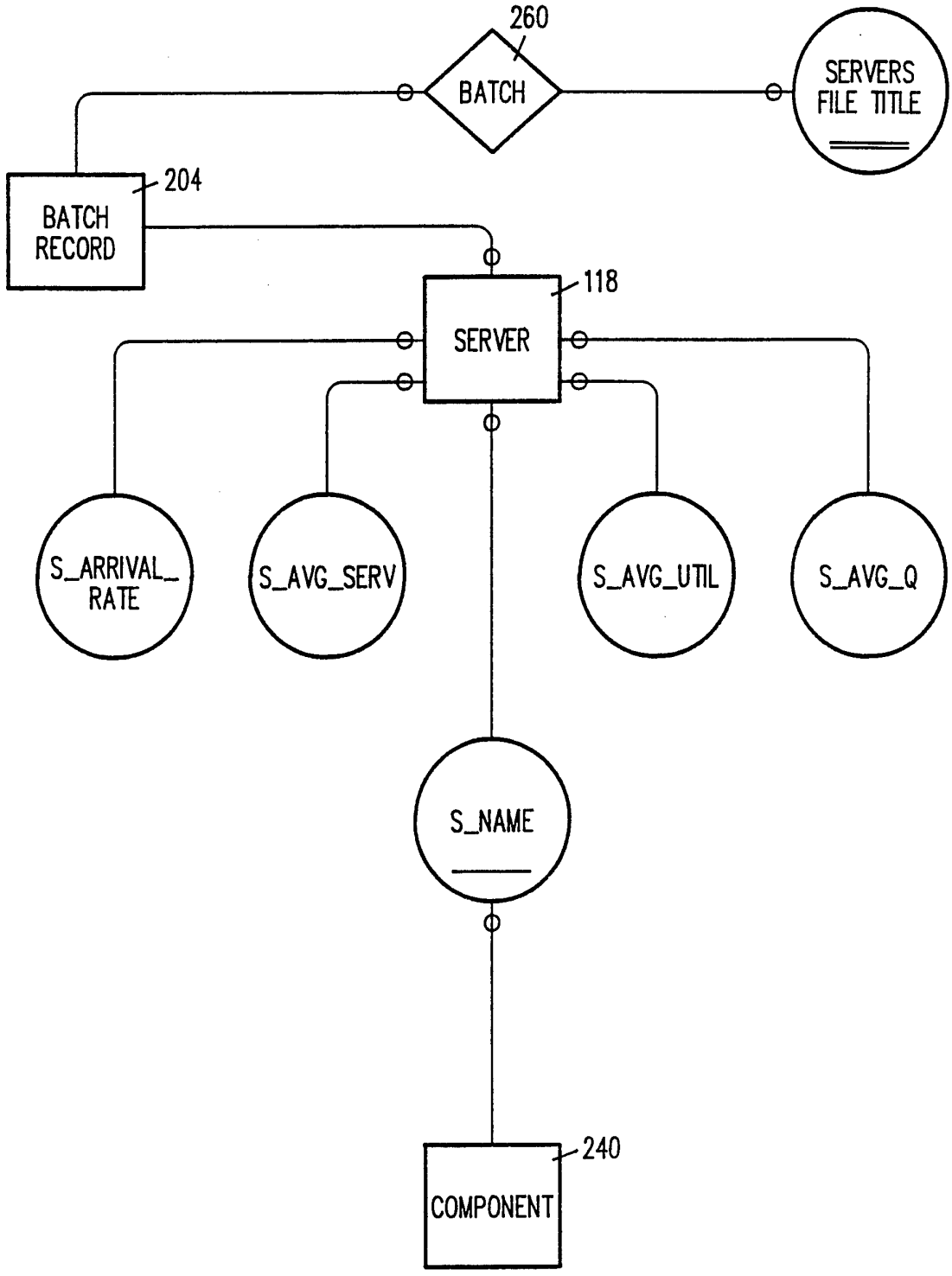

260
BATCH
SERVERS FILE TITLE
204
BATCH RECORD
118
SERVER
S_ARRIVAL_ RATE
S_AVG_SERV
S_AVG_UTIL
S_AVG_Q
S_NAME
240
COMPONENT

FIG. 12

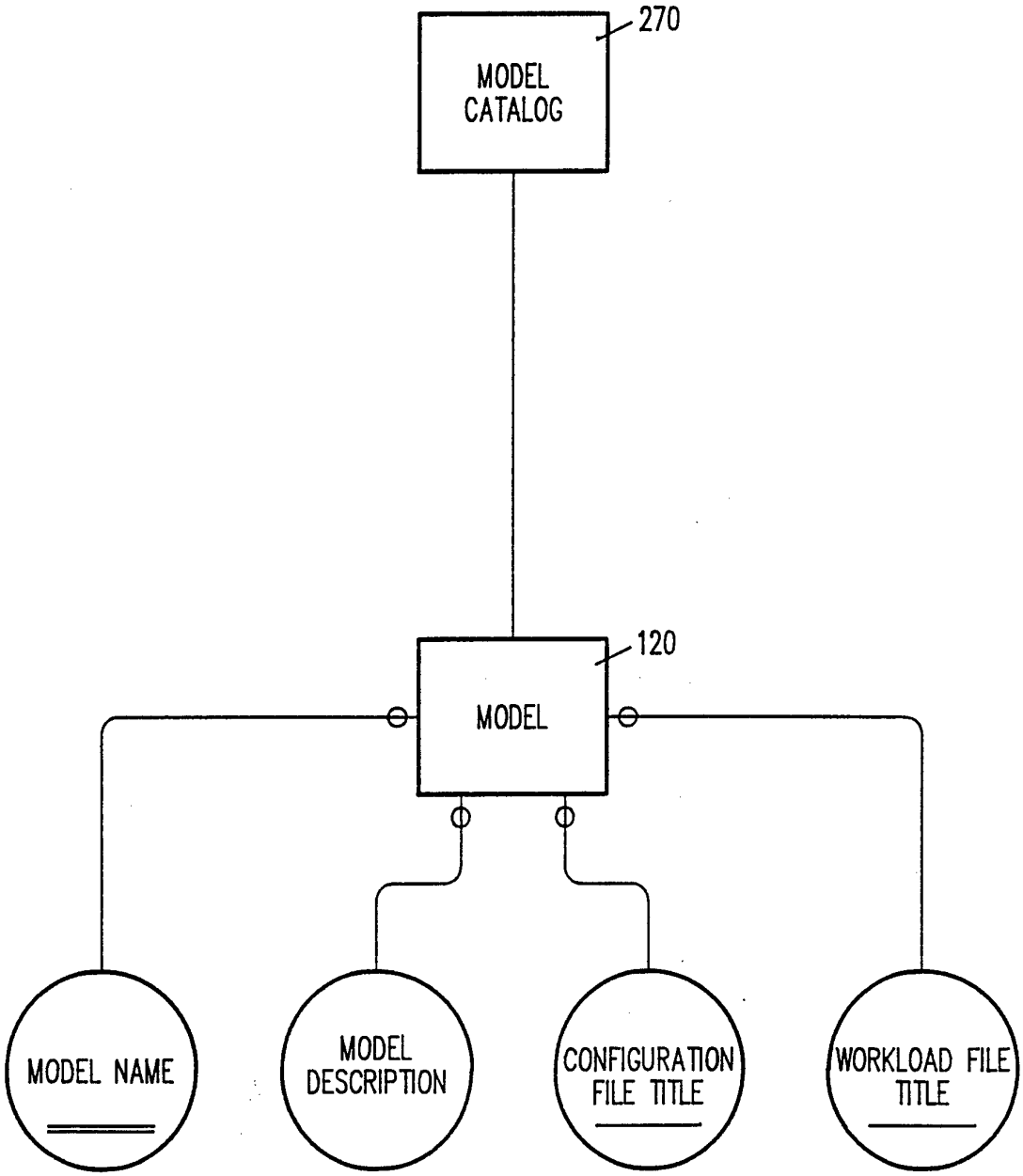

270
MODEL CATALOG
120
MODEL
MODEL NAME
MODEL DESCRIPTION
CONFIGURATION FILE TITLE
WORKLOAD FILE TITLE

FIG. 13

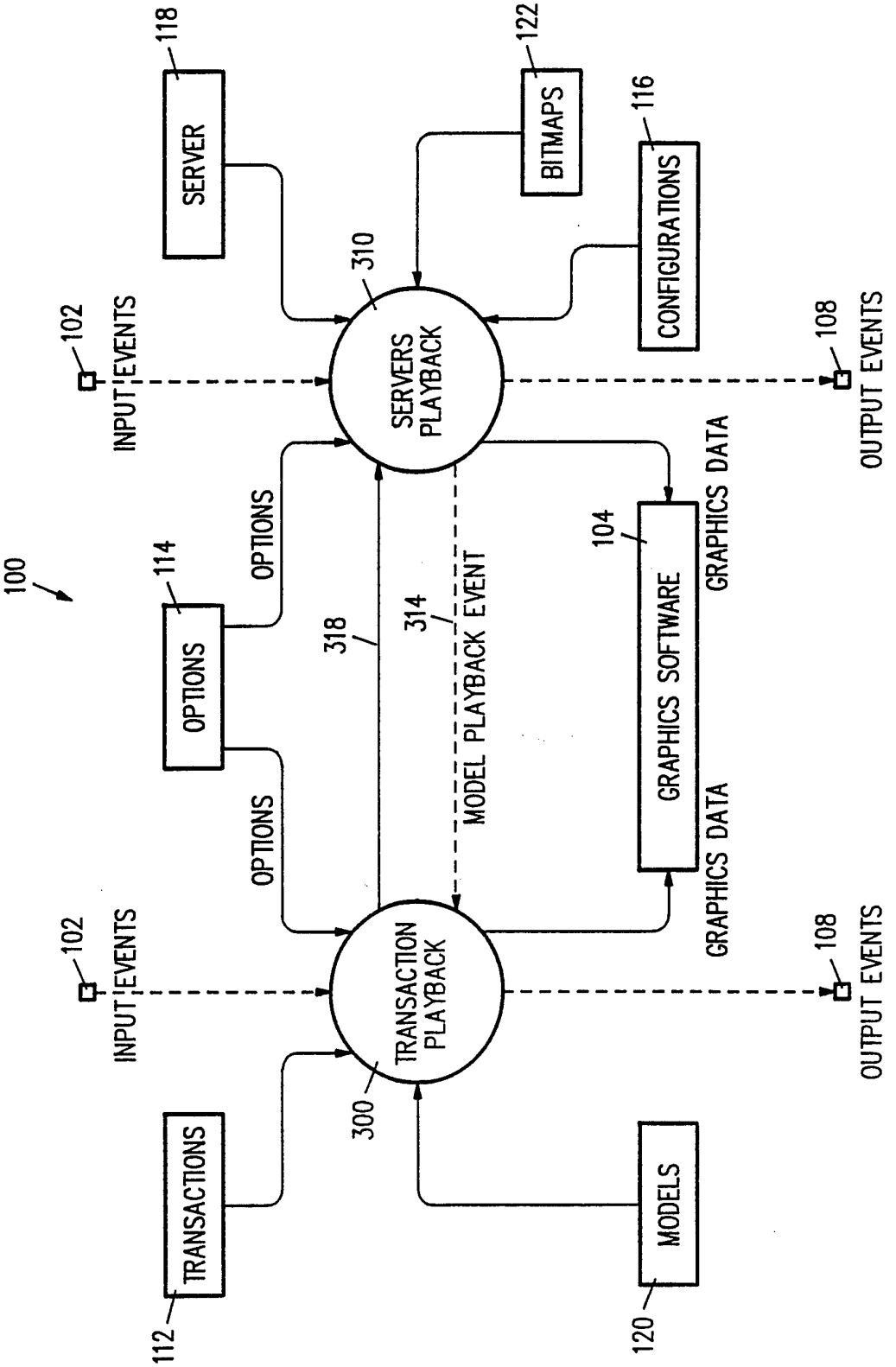

100
102
INPUT EVENTS
102
INPUT EVENTS
118
SERVER
122
BITMAPS
116
CONFIGURATIONS
310
SERVERS PLAYBACK
108
OUTPUT EVENTS
114
OPTIONS
OPTIONS
OPTIONS
318
314
MODEL PLAYBACK EVENT
104
GRAPHICS SOFTWARE
GRAPHICS DATA
GRAPHICS DATA
300
TRANSACTION PLAYBACK
108
OUTPUT EVENTS
112
TRANSACTIONS
120
MODELS

FIG. 14

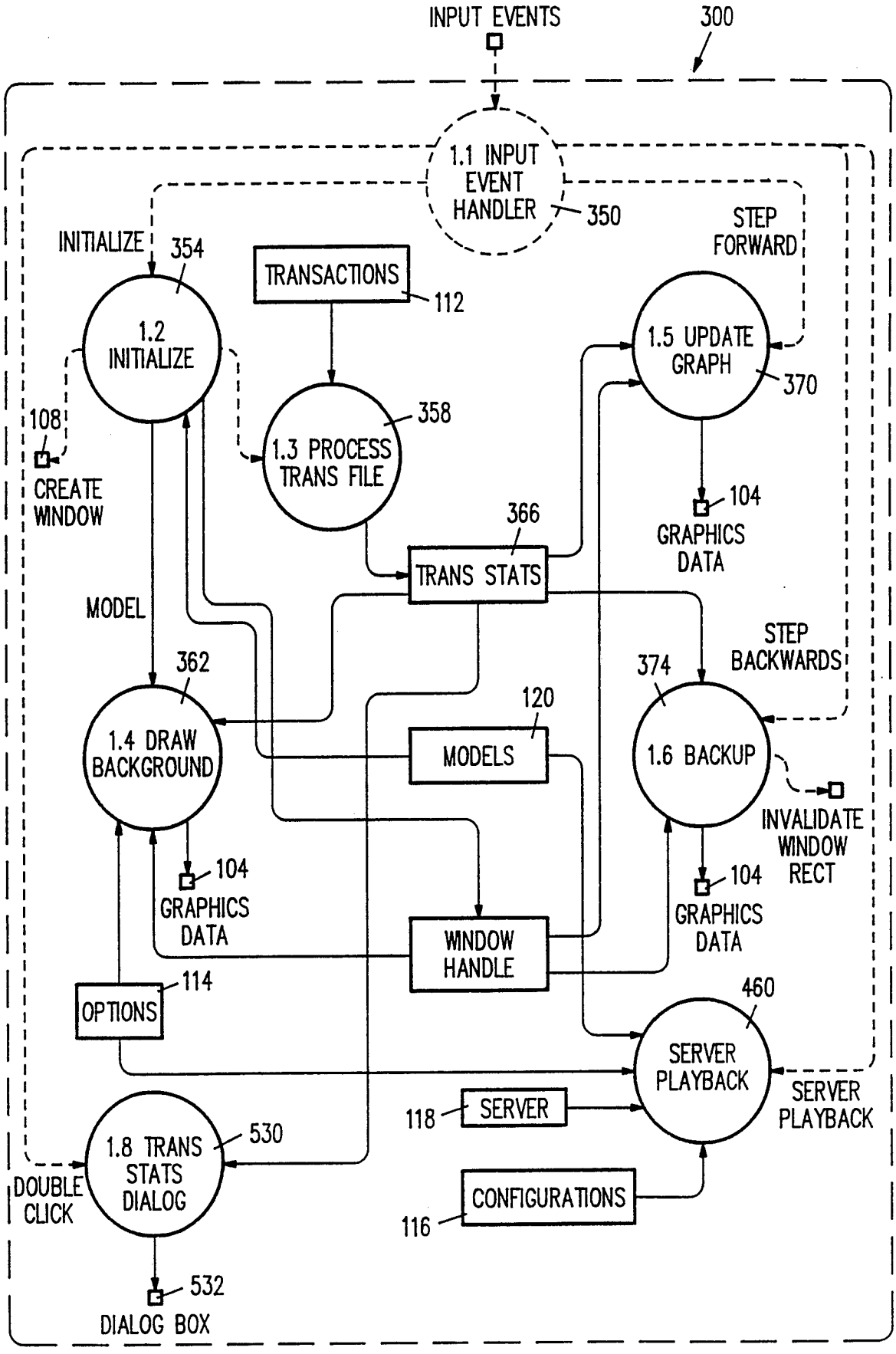

INPUT EVENTS
300
1.1 INPUT EVENT HANDLER
350
INITIALIZE
354
1.2 INITIALIZE
TRANSACTIONS
112
1.3 PROCESS TRANS FILE
358
STEP FORWARD
1.5 UPDATE GRAPH
370
108
CREATE WINDOW
104
GRAPHICS DATA
366
TRANS STATS
MODEL
362
1.4 DRAW BACKGROUND
120
MODELS
374
STEP BACKWARDS
1.6 BACKUP
INVALIDATE WINDOW RECT
104
GRAPHICS DATA
WINDOW HANDLE
OPTIONS
114
460
SERVER PLAYBACK
SERVER PLAYBACK
118
SERVER
1.8 TRANS STATS DIALOG
530
DOUBLE CLICK
CONFIGURATIONS
116
532
DIALOG BOX

FIG. 15

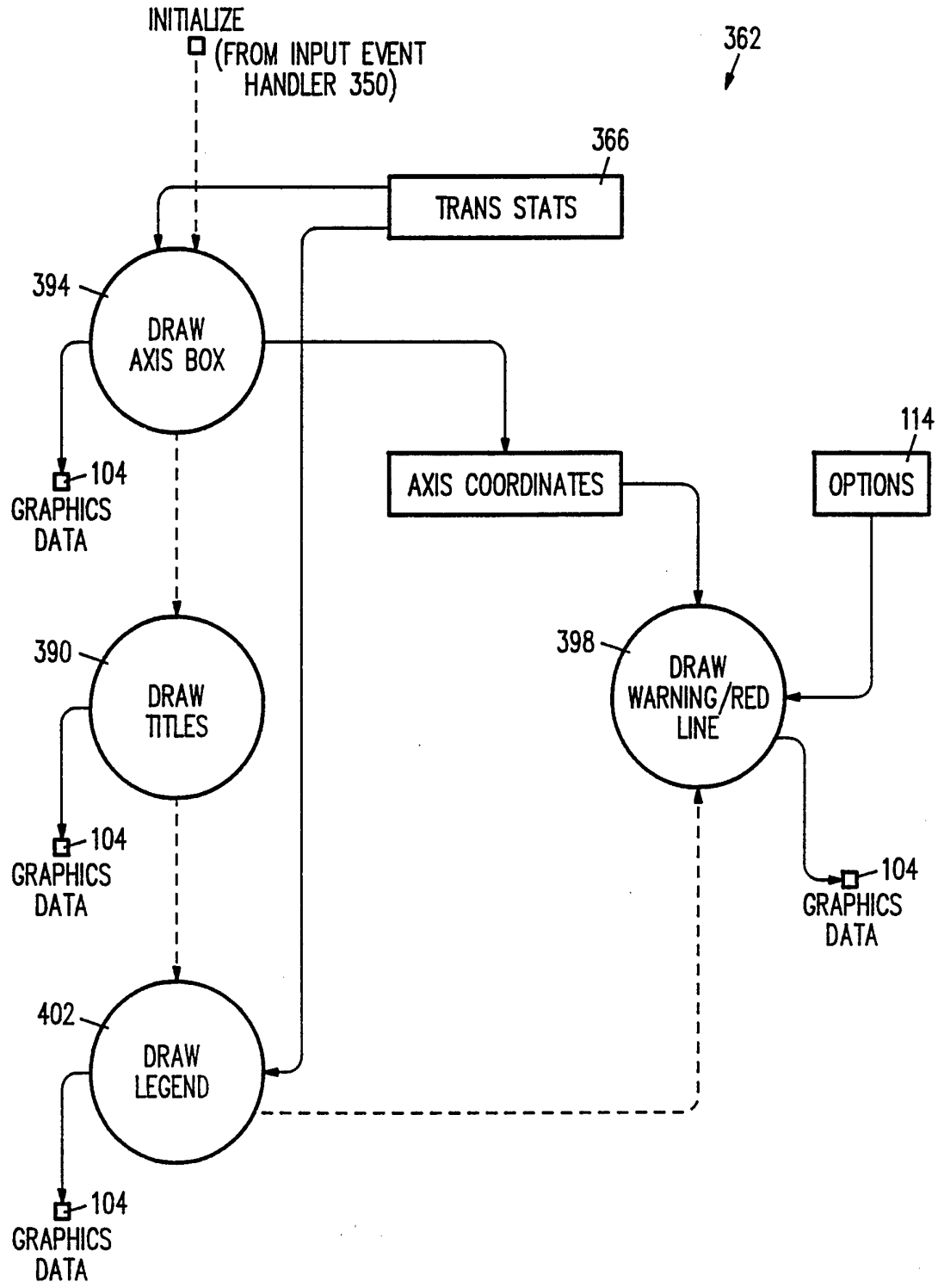

INITIALIZE
(FROM INPUT EVENT HANDLER 350)
362
366
TRANS STATS
394
DRAW AXIS BOX
104
GRAPHICS DATA
AXIS COORDINATES
114
OPTIONS
390
DRAW TITLES
398
DRAW WARNING/RED LINE
104
GRAPHICS DATA
104
GRAPHICS DATA
402
DRAW LEGEND
104
GRAPHICS DATA

FIG. 16

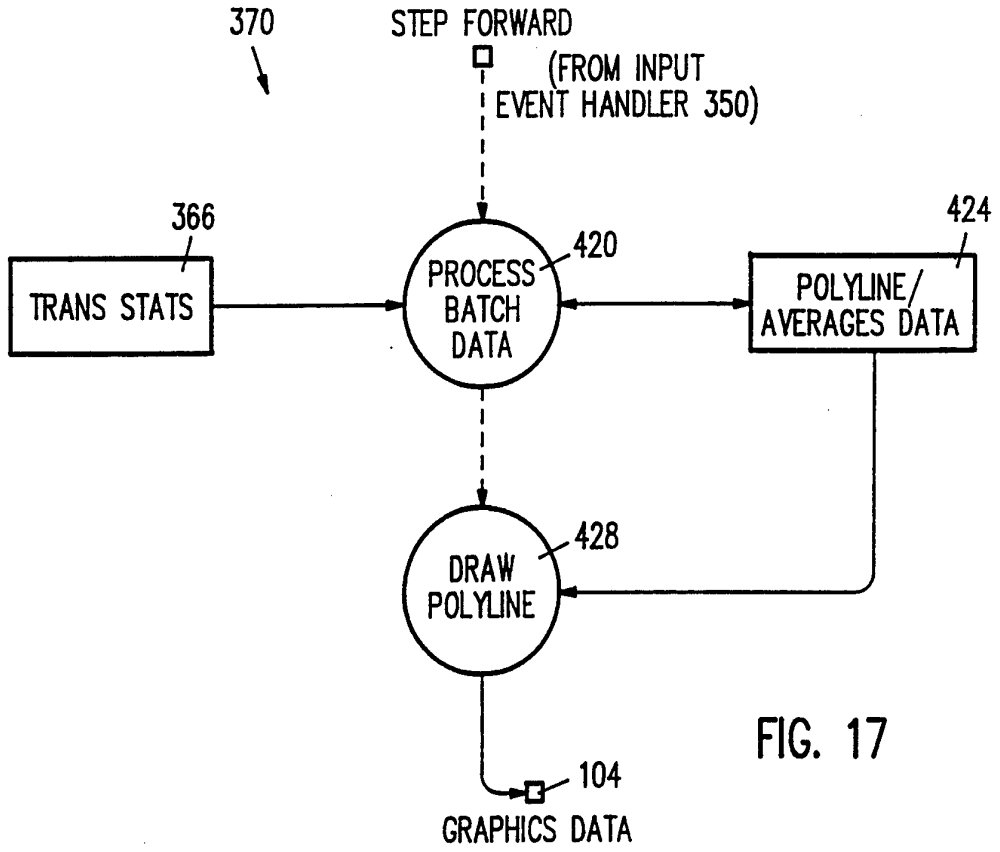

370
STEP FORWARD
(FROM INPUT EVENT HANDLER 350)
366
TRANS STATS
PROCESS BATCH DATA
420
POLYLINE/ AVERAGES DATA
424
DRAW POLYLINE
428
104
GRAPHICS DATA

FIG. 17

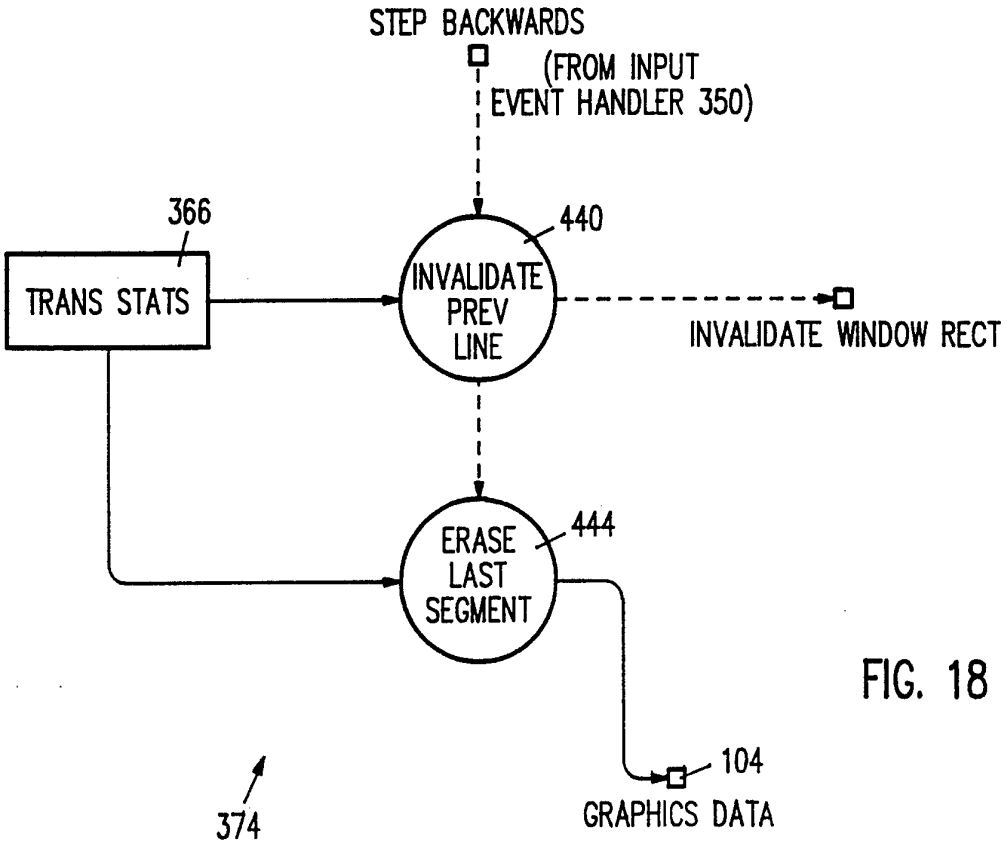

STEP BACKWARDS
(FROM INPUT EVENT HANDLER 350)
366
TRANS STATS
INVALIDATE PREV LINE
440
INVALIDATE WINDOW RECT
ERASE LAST SEGMENT
444
104
GRAPHICS DATA
374

FIG. 18

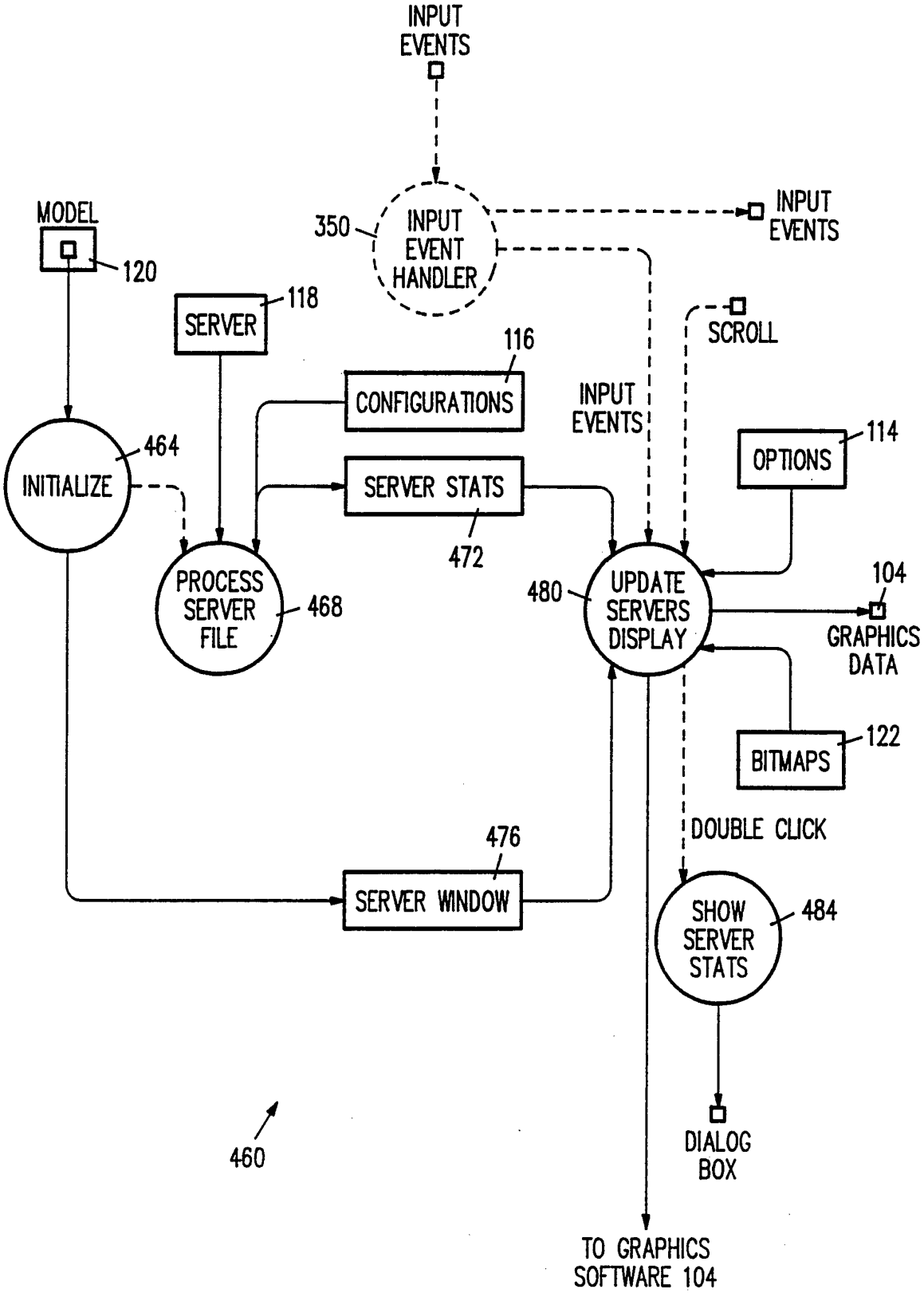

INPUT EVENTS
MODEL
120
350
INPUT EVENT HANDLER
INPUT EVENTS
SERVER
118
SCROLL
116
CONFIGURATIONS
INPUT EVENTS
OPTIONS
114
INITIALIZE
464
SERVER STATS
472
PROCESS SERVER FILE
468
480
UPDATE SERVERS DISPLAY
104
GRAPHICS DATA
BITMAPS
122
DOUBLE CLICK
476
SERVER WINDOW
SHOW SERVER STATS
484
460
DIALOG BOX
TO GRAPHICS SOFTWARE 104

FIG. 19

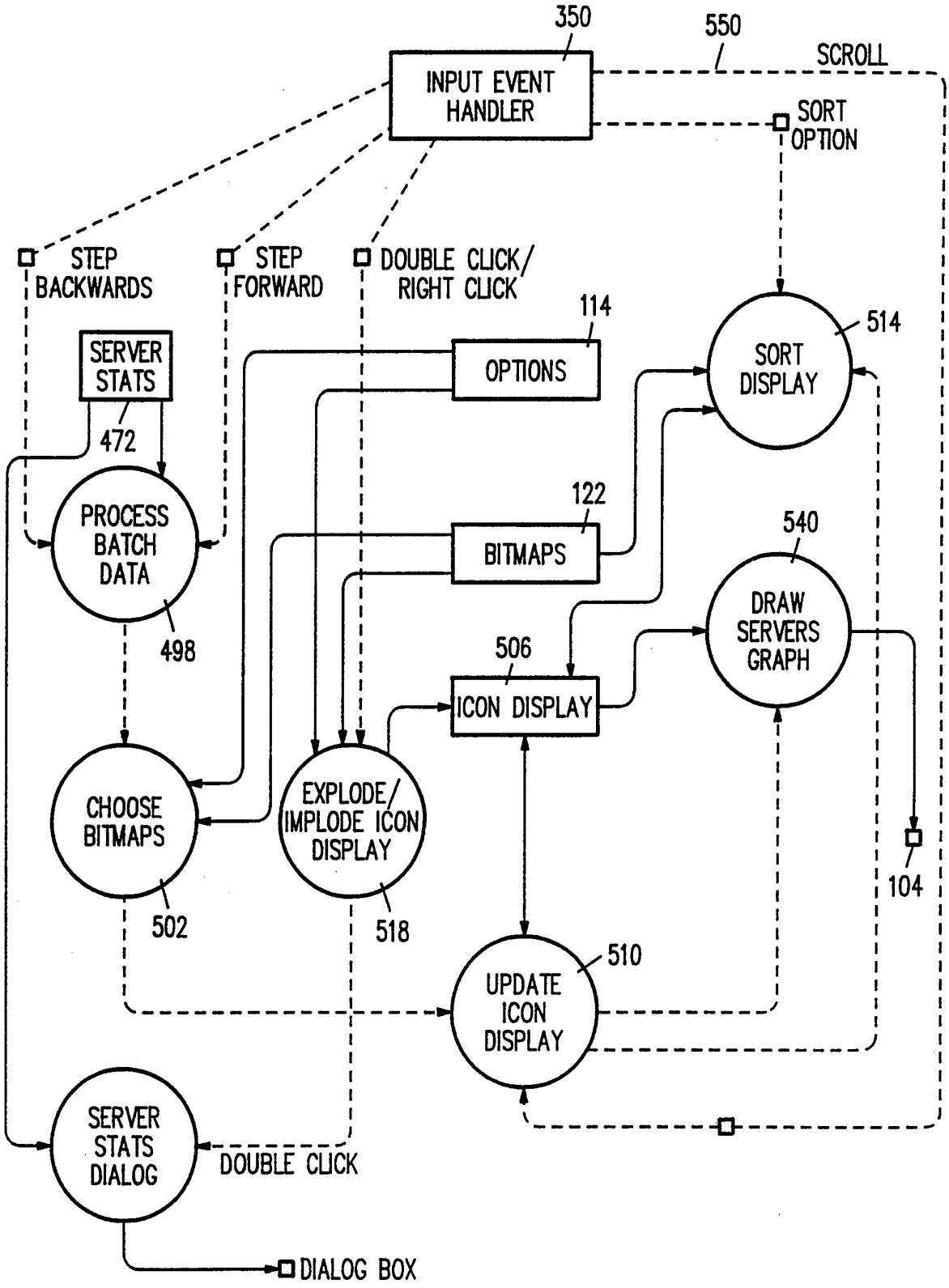

350
INPUT EVENT HANDLER
550
SCROLL
SORT OPTION
STEP BACKWARDS
STEP FORWARD
DOUBLE CLICK/ RIGHT CLICK
114
OPTIONS
SERVER STATS
472
514
SORT DISPLAY
122
BITMAPS
540
PROCESS BATCH DATA
498
DRAW SERVERS GRAPH
506
ICON DISPLAY
CHOOSE BITMAPS
502
EXPLODE/ IMPLODE ICON DISPLAY
518
104
510
UPDATE ICON DISPLAY
SERVER STATS DIALOG
DOUBLE CLICK
DIALOG BOX
480

FIG. 20

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/08710

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) : G06F 9/455
US CL : 395/500

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/500

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PROLOG: Simulat? and Interactive and Model? and Statistic?; Monitor? and Simulat? and Statistic?; Statistical? and Model.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 4,827,404 (Barstow et al.) 02 May 1989 See Col. 2, line 50 through Col. 4, line 63 and Col. 13, line 59 through Col. 18, line 38.	1-16
Y	IEEE Proceedings of the 1990 Winter Simulation Conference, December 9-12, 1990, Standridge et al., Interactive Simulation, pp. 453-458, especially relevant pp. 1-2.	1-16

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be part of particular relevance	
"E" earlier document published on or after the international filing date	
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention	
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone	
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art	
"&" document member of the same patent family	

Date of the actual completion of the international search 13 November 1993	Date of mailing of the international search report 15 DEC 1993
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. NOT APPLICABLE	Authorized officer William M. Treat Telephone No. (703) 305-9699

Form PCT/ISA/210 (second sheet)(July 1992)*