(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 October 2001 (25.10.2001)

PCT

(10) International Publication Number
WO 01/80007 A2

(51) International Patent Classification[7]: G06F 11/14, 11/00, 15/177

(21) International Application Number: PCT/US01/11990

(22) International Filing Date: 12 April 2001 (12.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

09/550,004	14 April 2000 (14.04.2000)	US
09/549,733	14 April 2000 (14.04.2000)	US

(71) Applicant *(for all designated States except US)*: **STRATUS TECHNOLOGIES INTERNATIONAL, S.A.R.L.** [CH/CH]; Zugerstrasse 76, CH-6340 Baar (CH).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **SUFFIN, Charles, A.** [US/US]; 47 Pine Arden Drive, West Boylston, MA 01583 (US).

(74) Agent: **LANZA, John, D.**; Testa, Hurwitz & Thibeault, L.L.P., High Street Tower, 125 High Street, Boston, MA 02110 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/80007 A2

(54) Title: METHODS ADN APPARATUS FOR ROBUST OPERATION OF A COMPUTER SYSTEM HAVING REDUNDANT COMPONENTS

(57) Abstract: Methods and apparatus for robust operation of a fault-tolerant computer system with redundant components. A method for deterministically booting a computer system having redundant components includes the step of selecting hardware and software components. The selected components are booted in a manner consistent with traditional computer systems. If the boot fails, a different set of components is selected and an attempt is made to boot those components traditionally. In one embodiment, the hardware and software components are a processor and an input/output controller. A corresponding apparatus is also discussed. An apparatus for recovering from the failure of a fault-tolerant system includes a plurality of processors, at least one input-output controller in communication with the processors, and a control module in communication with each of those elements. The control module detects a non-responsive processor, halts execution of the processors, selects a processor from the plurality of halted processors, restarts the failed processor, restarts the plurality of halted processors, and copies processor state from the selected halted processor to the failed processor. A corresponding method is also disclosed.

METHODS AND APPARATUS FOR ROBUST OPERATION OF A COMPUTER SYSTEM HAVING REDUNDANT COMPONENTS

Field of the Invention

The present invention relates to methods and apparatus for robust operation of a fault-tolerant computer, including initialization and recovery from operational failures. In particular, the present invention relates to methods and apparatus for recovering from an operational failure that preserves at least some of the operational state of the computer system, and methods and apparatus for deterministic reinitialization.

Background of the Invention

Information systems are evolving to become the delivery mechanism that drives corporate revenues. In industries ranging from financial services to on-line shopping, the computer has become the business. Accordingly, protection of computer-based data is becoming of paramount importance to a corporation's financial well-being.

Fault-tolerant systems offer superior reliability characteristics through the use of redundant components and data paths that ensure uninterrupted delivery of service. Even so, such systems may still fail due to hardware or software errors. In many cases, it is desirable to analyze the operational state of a computer in order to determine why a particular failure occurred. This is often difficult to accomplish since, in many cases, the only way to restore the computer system to operational status is to reset the system. The boot cycle typically destroys the operational status of the computer system, which is generally stored in volatile memory.

Moreover, a simple system reset may fail to identify intermittent hardware problems. For example, since a redundant, fault-tolerant system may include multiple CPUs, a single

misbehaving central processing unit may sometimes boot properly, masking a system error and causing the error to be irreproducible. In these cases, the system cannot be examined to determine the cause of the failure.

Summary of the Invention

The present invention is directed to methods and apparatus for robust operation of a fault-tolerant computer system with redundant components. It provides methods and apparatus for booting a computer system with redundant hardware and/or software components in a deterministic fashion. Individual hardware and/or software components are selected and a boot process is performed using those selected components. Booting in this manner allows application programs written for traditional machine to be used without modification. Further, modifications to boot software are rendered minimal or non-existent using this scheme. Moreover, booting individual processor-I/O controller pairs allows system faults to be isolated and detected in a deterministic fashion. The present invention also provides a user-configurable mechanism for instructing a computer system to take increasingly severe steps in order to return a computer system to operational status without destroying the data stored in processor registers or computer memory. The methods and apparatus disclosed are particularly useful for fault-tolerant computer systems using standard operating systems.

In one aspect, the present invention relates to a method for deterministically booting a fault-tolerant computer having a plurality of processors and one or more input-output controllers. A first processor/input-output controller pair is chosen and an attempt is made to boot the chosen pair. In the event that the attempt to boot the chosen pair fails, a new boot pair is selected.

In another aspect, the present invention relates to a method for deterministically booting a fault-tolerant computer having a plurality of processor boards and one or more input-output

controller boards. A first processor/input-output controller board pair is chosen and an attempt is made to boot the chosen board pair. In the event that the attempt to boot the chosen board pair fails, a new boot pair is selected.

In still another aspect, the present invention relates to an apparatus for deterministically booting a fault-tolerant system. The apparatus includes a plurality of processors, at least one input-output controller in communication with the processors, a memory element storing a list of processor/controller pairs, and a control module in communication with each element. The control module retrieves a first processor/controller pair identifier from the memory element and attempts to boot the processor/controller pair identified. In the event that the boot attempt fails, a second identifier is retrieved from the memory element and an attempt is made to boot the second boot pair identified.

In yet another aspect, the present invention relates to an apparatus for deterministically booting a fault-tolerant system composed of individual hardware or software objects. A set of hardware and/or software components is selected and a boot process is performed using this set of components. In the event that the boot fails, a new boot set is selected.

In another aspect, the present invention relates to a method for recovering from a failure of a fault-tolerant system that includes the plurality of processors and one or more input-output controllers. A non-responsive processor is identified. One of the processors is selected from the plurality of processors and its execution is halted. The non-responsive processor is then restarted as are the other processors in the plurality. Processor state from the selected processor is copied to the non-responsive processor.

In still another aspect, the present invention relates to an apparatus for recovering from the failure of a processor in a fault-tolerant system. The apparatus includes a plurality of

processors, at least one input-output controller in communication with the processors, and a control module in communication with each of these elements. The control module detects that a processor is non-responsive, halts execution of the other processors in the plurality, selects a processor, restarts the failed processor, restarts the rest of the processors, and copies processor state from the halted processor to the failed processor.

Brief Description of the Drawings

The invention is pointed out with particularity in the appended claims. The advantages of the invention described above, as well as further advantages of the invention, may be better understood by reference to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1 is a block diagram of an embodiment of a traditional computer system;

FIG. 2 is a block diagram of an embodiment of a redundant, fault-tolerant computer system;

FIG. 3 is a block diagram showing an embodiment of auxiliary connections between service management logic units, processors, and I/O controllers in the system of FIG. 2;

FIGs. 4 and 4A are block diagrams depicting an embodiment of the steps to be taken during initialization of a fault-tolerant computer system; and

FIGs. 5 and 5A are screen shots depicting exemplary embodiments of user interfaces for controlling the booting process.

Detailed Description of the Invention

Referring now to FIG. 1, a typical computer 14 as known in the prior art includes a central processor 20, a main memory unit 22 for storing programs and/or data, an input/output

(I/O) controller 24, a display device 26, and a data bus 42 coupling these components to allow communication between these units. The memory 22 may include random access memory (RAM) and read only memory (ROM) chips. The computer 14 typically also has one or more input devices 30 such as a keyboard 32 (e.g., an alphanumeric keyboard and/or a musical keyboard), a mouse 34, and, in some embodiments, a joystick 12.

The computer 14 typically also has a hard disk drive 36 and a floppy disk drive 38 for receiving floppy disks such as 3.5-inch disks. Other devices 40 also can be part of the computer 14 including output devices (e.g., printer or plotter) and/or optical disk drives for receiving and reading digital data on a CD-ROM. In the disclosed embodiment, one or more computer programs define the operational capabilities of the system 10. These programs can be loaded onto the hard drive 36 and/or into the memory 22 of the computer 14 via the floppy drive 38. Applications may be caused to run by double clicking a related icon displayed on the display device 26 using the mouse 34. In general, the controlling software program(s) and all of the data utilized by the program(s) are stored on one or more of the computer's storage mediums such as the hard drive 36, CD-ROM 40, etc.

System bus 42 allows data to be transferred between the various units in the computer 14. For example, processor 20 may retrieve program data from memory 22 over system bus 42. Various system busses 42 are standard in computer systems 14, such as the Video Electronics Standards Association Local Bus (VESA Local Bus), the industry standard architecture ISA bus (ISA), the Extended Industry Standard Architecture bus (EISA), the Micro Channel Architecture bus (MCA) and the Peripheral Component Interconnect bus (PCI). In some systems 14 multiple busses may be used to provide access to different units of the system. For example, a system 14 may use a PCI to connect a processor 20 to peripheral devices 30, 36, 38 and concurrently connect the processor 20 to main memory 22 using an MCA bus.

It is immediately apparent from FIG. 1 that such a traditional computer system 14 is highly sensitive to any single point of failure. For example, if main memory unit 22 fails to operate for any reason, the computer 14 as a whole will cease to function. Similarly, should system bus 42 fail, the system 14 as a whole will fail. A redundant, fault-tolerant system achieves an extremely high level of availability by using redundant components and data paths to insure uninterrupted operation. A redundant, fault-tolerant system may be provided with any number of redundant units. Configurations include dual redundant systems, which include duplicates of certain hardware units found in FIG. 1, and triply redundant configurations, which include three of each unit shown in FIG. 1. In either case, redundant central processing units 20 and main memory units 22 run in "lock step," that is, each processor runs identical copies of the operating system and application programs. The data stored in replicated memory 22 and registers provided by the replicated processors 20 should be identical at all times.

Referring now to FIG. 2, one embodiment of a redundant, fault-tolerant system 14' is shown that includes three processors 20, 20', 20" (generally 20) and at least two input output controllers 24, 24' (generally 24). As shown in FIG. 2, system 14' may include more than two input output controllers (24" and 24'" shown in phantom view) to allow the system 14' to control more I/O devices. In the embodiment shown in FIG. 2, four redundant system busses 42, 42', 42" and 42'" (generally 42) are used to interconnect each processor 20 and I/O controllers 24. In one embodiment, processors 20 are selected from the "x86" family of processors manufactured by Intel Corporation of Santa Clara, California. The x86 family of processors includes the 80286 processor, the 80386 processor, the 80486 processor, and the Pentium, Pentium II, Pentium III, and Xeon processors. In another embodiment processors are selected from the "680x0" family of processors manufactured by Motorola Corporation of Schaumburg, Illinois. The 680x0 family of processors includes the 68000, 68020, 68030, and 68040 processors. Other processor families

include the Power PC line of processors manufactured by the Motorola Corporation, the Alpha line of processors manufactured by Compaq Corporation of Houston, Texas, and the Crusoe line of processors manufactured by Transmeta Corporation of Santa Clara, California.

Each processor 20 may include logic that implements fault-tolerant support. For embodiments in which CPU 20 is a single chip, the fault-tolerant logic may be included on the chip itself. In other embodiments, the CPU 20 is a processor board that includes a processor, associated memory, and fault-tolerant logic. In these embodiments, the fault-tolerant logic can be implemented as a separate set of logic on processor board 20. For example, the fault-tolerant logic may be provided as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), an electrically erasable programmable read-only memory (EEPROM), a programmable read-only memory (PROM), a programmable logic device (PLD), or a read-only memory device (ROM). The fault-tolerant logic compares the results of each operation performed by the separate processors 20 to the results of the same operation performed on one of the other processors 20. If a discrepancy is determined then a failure has occurred.

Each input-output controller may also include fault-tolerant logic that monitors transactions on the system busses 42 to aid in determining a processor failure. As shown in FIG. 2, the I/O controller boards 24 also provide support for the display 26, input devices 30 and mass storage such as floppy drives 38, hard drives, and CD-ROM devices. The embodiment shown in FIG. 2 includes a front panel 52 that provides an interface to these input and output devices. In these embodiments, the front panel may serve as an adapter between the I/O controllers 24 and, for example, a universal serial bus (USB) used by keyboard and mouse input devices, or a video connector (EGA, VGA, or SVGA) used for connecting displays to the system 14'.

Each I/O controller 24 includes service management logic which performs various system management functions, such as: monitoring the operational status of the system; performing on-line diagnostics of the system; and providing an interface for remotely viewing system operation (including a processor boot sequence). In some embodiments, the service management logic includes a modem providing a serial line connection to a service network. In other embodiments, the service management logic includes a connection for communicating with other customer equipment, such as an Ethernet connection of other local area network connection. In some embodiments, the service management logic is provided as a separate board that is in communication with I/O controller 24. In one particularly preferred embodiment, a service management board including all service management logic connects to I/O controller 24 via a PCI slot. The service management logic (referred to hereafter as SML) may be provided with a power supply separate from the remainder of the system 14'.

Referring now to FIG. 3, a block diagram shows the connection between SML units 50, 50' (generally 50) and the I/O controllers 24, 24' and processors 20, 20', 20" of the system 14'. As shown by FIG. 3, each SML 50 is connected to each of the other units by redundant auxiliary busses 60, 60' in addition to redundant busses 42. Auxiliary busses 60, 60' may be any bus that allows the SMLs 50 to control and query the processors 20 and I/O controllers 24. The SMLs can communicate with the other units using a variety of connections including twisted pair, broadband connections, or wireless connections. Connections can be established using a variety of lower layer communication protocols such as TCP/IP, IPX, SPX, Ethernet, RS-232, direct asynchronous connections, or I^2C. In general, any message-oriented protocol may be used, and a check-summed, packet-oriented protocol is preferred.

Referring now to FIG. 4, the steps to be taken to boot a redundant, fault-tolerant system are shown. In brief overview, the boot process begins by powering on the SMLs (step 402), initializing and communicating with other SMLs in the system (steps 404, 406 and 408), and determining whether or not the system requires booting (step 410).

In greater detail, and as noted above, SMLs 50 are provided with power separate from the power provided to the system 14'. Power is supplied to the SMLs (step 402) before any other units in the system 14'. For embodiments in which the SML is a portion of an I/O controller board 24, power may be supplied to the entire I/O controller board 24 but only routed to the SML portion of the controller board 24. For embodiments in which the SML is provided as a separate board, then only the SML is supplied with power. In either case, whether and when power is supplied to the other units in the system is under the direct control of the SML.

A SML uses auxiliary busses 60, 60' to determine if other SMLs exist in the system (step 404). If so, the SMLs exchange messages over the auxiliary busses 60, 60' in order to determine which SML will function as the primary SML for the system 14' (step 406). The determination of which SML will function as the primary SML may include many factors, including: whether or not a service management logic unit has been previously inserted in the system to be powered up; and whether another SML has already been powered up and is operational. In other embodiments, the identity of the primary SML may be "hardwired."

If an SML 50 determines that no other SML exists in the system 14', or if an SML 50 has determined that it will function as the primary SML 50 for a system 14' with multiple SMLs, the SML identifies with which I/O controller 24 it is associated (step 408). The SML 50 uses this information during the boot process to determine if another SML 50 should act as the primary SML 50 during the boot process. For example, if the I/O controller with which the SML 50 is

associated is not selected for booting, then the SML 50 associated with the booting I/O controller must act as the primary SML 50 for the boot attempt. In other words, BIOS heartbeat and other boot status messages will be directed to the SML 50 on the booting I/O controller, even if that SML 50 is not the primary SML 50.

Once an SML determines that it is the primary SML for a system 14', it determines whether or not to boot the system 14'. SMLs 50 can exchange messages to negotiate which SML 50 is the primary SML 50. If an SML 50 is already functioning in the system as primary, then a peer SML 50 becomes secondary. If neither SML 50 has yet been identified as the primary SML 50, the SMLs 50 negotiate to determine which SML 50 is the primary SML 50. In one embodiment the SMLs 50 negotiate to determine which SML 50 is the primary SML 50. In one embodiment, the SMLs 50 negotiate using the following rules:

1. If one SML 50 is "alien" to the system then the SML 50 which is not alien becomes primary. "Alien" means that the SML 50 was not resident in the computer system the last time it was used.

2. If one SML 50 was primary more recently than the other, it becomes the primary again (and the other becomes secondary).

3. As a default, the SML 50 in I/O board slot 0 becomes the primary SML 50. The SML 50 in I/O board slot 1 becomes secondary.

A service management logic unit, in this embodiment, will not boot the system if it was explicitly shut down by an administrator (for example, if the administrator used a "power off" command to shut down the system). Whether or not a system has been explicitly shut down by an

administrator may be stored in non-volatile memory (not shown in the drawings) that the SML 50 may query.

If a SML 50 determines that it should not boot the system 14', it transitions to a state in which it monitors the system (step 412). This state is described in greater detail below. For example, an SML 50 may query a non-volatile memory element and discover that the system 14' was properly and explicitly shut down by an administrator. In this case, the SML 50 will not attempt to boot the system 14'. Otherwise, the system moves to the boot process described in FIG. 4A.

The boot process shown in FIG. 4A may be commenced by an initializing SML 50. Alternatively, the boot process may be directly invoked by a system administrator by, for example, a "boot" command. FIG. 5A is a screen shot showing an exemplary embodiment for providing such commands to the system administrator by the primary SML 50. In this embodiment, system administration commands are grouped as a set of "tabs" and displayed to the administrator. The administrator selects the tab containing the desired operations. FIG. 5A depicts an embodiment in which a "System Control" tab 54 provides four controls for a system: a "Power On" command 56 (depicted in gray to indicate the system is currently running; an explicit "Power Off" command 58; a "Reset" command 60; and a "System Interrupt" command 62. System information 64, as well as information concerning the primary SML 66, is provided to the administrator. In the embodiment shown in FIG. 5A, the administration commands are provided using a browser-based user interface. Although FIG. 5A depicts an embodiment using NETSCAPE NAVIGATOR, manufactured by Netscape Communications of Mountain View, California, any browser may be used, including MICROSOFT INTERNET EXPLORER, manufactured by Microsoft Corporation of Redmond, Washington. A third way for the boot

process shown in FIG. 4A to be invoked is by an SML following a system failure. This mechanism is discussed in greater detail below.

The boot process begins by determining a "boot list" (step 450) FIG. 4A. A boot list is a list of component systems allowing the system to boot. For example, boot components may include processors, I/O controllers, BIOS, and other software (both application and system). In one particular embodiment, a boot list an ordered list of processor-I/O controller pairs. In some embodiments, the boot list includes "heartbeat" values associated with each boot pair. Heartbeat values are used by an SML 50 during system operation to determine if a processor 20 is functioning properly. Heartbeats are described in greater detail below. The boot list may be stored in a data structure that associates processor identification values with I/O controller values. For embodiments in which heartbeat values are also stored, the data structure includes an additional field to associate heartbeat timer values with each boot pair. The data structure may be stored on each SML 50 in a system 14'. In preferred embodiments, the data structure is stored in a non-volatile, erasable memory element, such as an EEPROM, that is accessible using auxiliary busses 60, 60'. In the event that the stored data structure is inconsistent (for example the data structure may include corrupted data values), or if the SML 50 is unable to retrieve data from the memory element (for example, if no memory element exists or if both auxiliary busses 60, 60' are not functioning), the SML 50 may use a hard-coded default list.

FIG. 5B depicts a screen shot of an exemplary user interface allowing a system administrator to modify the default boot list. As shown in connection with FIG. 5A, the user interface is browser based and provides information to the administrator regarding the system 14' and SML 50 currently active. Once the graphical user interface shown in Fig. 5B is used to create a boot list, it is saved to the non-volatile memory element.

In one embodiment, once a boot list is determined, whether by retrieving a list from a memory element or by using a default list, the SML 50 determines available processors 20 and I/O controllers 24 (step 452). The SML 50 may transmit a message over auxiliary busses 60, 60' to determine this information. Processors 20 and I/O controller 24 respond to the message transmitted by the SML 50. The SML 50 concludes that a processor 20 or I/O controller does not exist if no response to the message is received on either bus 60, 60'. This information is used by the SML 50 to skip pairs in the boot list if they reference units not present in the system 14'.

Once all system units are discovered by the SML 50, the SML 50 provides system clocks to the processors 20 and the I/O controllers 24 (step 452). In other embodiments system clocks are not under the control of the SML 50 and, in these embodiments, step 452 may be skipped.

Using auxiliary busses 60, 60', the SML 50 asserts a reset signal associated with each processor 20 and I/O controller 24 (step 456). The SML 50 takes any other steps necessary at this point to prepare all system units for booting. For example, some units may need to have power applied or, for example, certain other signals may need to be asserted to prepare the unit for booting.

The SML releases reset from the processor 20 and the I/O controller 24 identified in the boot list as the first boot pair while holding reset active for all other system units (step 458). This allows the selected boot pair to boot in a manner consistent with a traditional computer. The SML 50 monitors the boot process of the selected boot pair to determine if the boot process is successful (step 460). In one embodiment, the SML 50 monitors the progress of the boot process by receiving heartbeat signals from the booting process-I/O controller pair. In one embodiment, heartbeats are transmitted over system busses 40. Failure to receive a heartbeat signal within a predetermined time period indicates that the boot process has failed. If the boot

process is not successful, the SML 50 selects a new boot pair from the boot list (step 462) and attempts to boot that processor-I/O controller pair. In some embodiments, the Basic Input-Output System (BIOS) may, during the boot attempt, determine that it cannot achieve a proper boot of the operating system , even though the processor has booted and is providing heartbeat signals to the SML 50. In this case, the BIOS issues an explicit "reboot" command to the SML 50 and the SML 50 selects a new boot pair from the boot list.

If the SML 50 cycles through every pair identified in the boot pair list and none of the pairs is successful, the SML 50 indicates that the system 14' was unable to boot. In some embodiments the SML 50 removes all power from the processors 20 and the I/O controllers 24 after determining the system 14' is unable to boot.

If the boot process is successful, the BIOS transmits a message to the SML indicating that the operating system has booted properly. In this case, the SML transitions to a monitoring state (step 464). In some embodiments, after successfully booting the first processor-I/O pair the SML 50 boots each other processor 20 in the system 14'.

Once the booting process is complete, or if the SML 50 determines that the system 14' should not be booted, the SML 50 enters a monitoring state (steps 412 or 464). In this state the SML 50 monitors heartbeat signals from each of the processors 20 to determine operation status of the system 14'. A failure to receive a heartbeat signal from a processor 20 during a predetermined period indicates that a failure has occurred. In this event, the SML 50 consults a non-volatile memory element to determine what actions, if any to take. The memory element may be the same memory element discussed above that stores the boot list, or a separate memory element may be provided that is accessible via the auxiliary busses 60, 60'. In one embodiment, the memory element stores a value that indicates one of seven actions for the SML 50 to take

upon heartbeat failure: (1) no action; (2) normal interrupt; (3) non-maskable interrupt; (4) stop processor from executing; (5) system reboot; or (6) deterministic boot. Each of these options is discussed in detail below.

A memory value indicating that the SML 50 should take no action on a heartbeat failure disables all recovery mechanisms. In some embodiments, the SML 50 logs the failure but otherwise does nothing.

A memory value indicating "normal interrupt" restricts recovery attempts by the SML 50 to issuing normal interrupts to the processor 20 or processors 20 that have ceased to transmit a heartbeat. In this embodiment, the SML 50 issues an interrupt to a target processor 20 via the auxiliary busses 60, 60'. If the processor's operating system is able to process the interrupt, it responds by restarting heartbeat transmission. In some embodiments, the operating system ensures that lockstep processing is resumed. In other embodiments, the SML 50 issues interrupts to the processor or processors such that the processors resume lockstep operation. For example, interrupts may be issued to processors simultaneously which should avoid breaking lockstep. In some embodiments the operating system halts execution of all programs and allows a system administrator to debug system settings. If the operating system does not respond to the interrupt, then recovery fails. In some embodiments, the SML 50 simply logs this failure. In other embodiments, the SML 50 alerts an administrator that the system 14' will not respond.

A memory value indicating "non-maskable interrupts" restricts recovery attempts by the SML 50 to issuing normal and non-maskable interrupts to the processor 20 or processors 20 that have ceased to transmit a heartbeat. In this embodiment, should the system 14' refuse to respond to a normal interrupt, the SML 50 issues a non-maskable interrupt to a target processor 20 via the I/O controller 24. If multiple processors 20 are hung, non-maskable interrupts are issued to all

processors 20 in lockstep to avoid breaking processor lockstep. If the processor's operating system is able to process the non-maskable interrupt, it responds by restarting heartbeat transmission. In this case, the SML 50 must revoke the previously issued normal interrupt. In some embodiments the operating system halts execution of all programs and allows a system administrator to debug system settings. If the operating system does not respond to the non-maskable interrupt, then recovery fails. In some embodiments, the SML 50 simply logs this failure. In other embodiments, the SML 50 alerts an administrator that the system 14' will not respond.

A memory value indicating that processor execution should be suspended allows the SML 50, in the event that a non-maskable interrupt fails to restore system operation, to select a processor 20 and suspend execution of all applications and the operating system by that processor 20. Processor and memory state of the suspended processor is not destroyed. If heartbeat signals resume from the other processors once the selected processor 20 is suspended, recovery has been successful. The state of the suspended processor 20 may be dumped for analysis, the state of the suspended processor may be replaced with state from one of the operational processors 20, or both. If this step fails to restore the system 14' to operational status, the SML 50 may dump the state of the suspended processor 20 for analysis by a system administrator, log the failure, alert an administrator to the failure, or any combination of these actions.

A memory value indicating "system reboot" allows the SML 50 to attempt to reboot the system in the event that suspended a selected processor 20 does not succeed. The reboot process is similar to the reboot process described in connection with FIGs. 4 and 4A, except that the suspended processor 20 is skipped during reboot of the boot pairs listed in the boot list. To avoid

repetitive heartbeat failure, the SML 50 maintains an index to identify the last processor-I/O boot pair in the boot list that last rebooted successfully. During the reboot process, this index is incremented to ensure that a different pair is selected as the starting pair each time. If successful, the state of the suspended processor 20 may be dumped for analysis, the state of the suspended processor 20 may be replaced with the state of one of the operational processors, or both. As above, if this mechanism doesn't succeed in restoring the system 14' to operational status, the SML 50 may dump the state of the suspended processor 20 for analysis by a system administrator, log the failure, alert an administrator to the failure, or any combination of these actions. A memory value indicating "deterministic boot" allow the SML 50 to abandon the state of the suspended board and perform a full deterministic reboot, as described in connection with FIGs. 4 and 4A.

Having described certain embodiments of the invention, it will now become apparent to one of skill in the art that other embodiments incorporating the concepts of the invention may be used. Therefore, the invention should not be limited to certain embodiments, but rather should be limited only by the spirit and scope of the following claims.

CLAIMS

What is claimed is:

1. In a fault-tolerant system including at lest one processor and input-output controller, a method for deterministically booting the system, the method comprising the steps of:

(a) choosing a first processor/input-output controller pair comprising a first processor and a first input/output module; and

(b) booting said first processor/module pair.

2. The method of claim 1 further comprising the step of

(c) booting each of other processors.

3. The method of claim 1 wherein step (a) comprises the step of choosing a first processor/module pair by accessing a predefined list of potential processor/module pairs stored in a memory element.

4. The method of claim 1 wherein step (b) comprises executing a basic input/output system to boot the first processor.

5. The method of claim 1 further comprising the steps of:

(c) detecting that the first processor/module pair failed to boot;

(d) selecting a second processor/module pair comprising a second processor and the first input-output controller; and

(e) booting the second processor/module pair.

6. In a fault-tolerant system including at least one processor board and input-output controller board, a method for deterministically booting the system, the method comprising the steps of:

(a) choosing a first processor board/input-output controller board pair comprising a first processor board and a first input-output controller board; and

(b) booting said first processor board/module board pair.

7. The method of claim 6 further comprising the step of

(c) booting each of other processor boards in the plurality of processor boards.

8. The method of claim 6 wherein step (a) comprises the step of choosing a first processor board/module board pair by accessing a predefined list of potential processor board/module board pairs stored in a memory element.

9. The method of claim 6 wherein step (b) comprises executing a basic input/output system to boot the first processor board.

10. The method of claim 6 further comprising the steps of:

(c) detecting that the first processor board/module board pair failed to boot;

(d) selecting a second processor board/module board pair comprising a second processor board and the first input-output controller board; and

(e) booting the second processor board/module board pair.

11. An apparatus for deterministically booting a fault-tolerant system comprising:

at least one processor;

at least one input-output controller in communication with at least one processor;

a memory element storing a list of processor/module pairs; and

a control module in communication with the memory element, the at least one input-output controller, and at least one processor,

wherein the control module retrieves a first processor/input-output pair identifier from the memory element and boots an identified first processor/module pair.

12. The apparatus of claim 11 wherein the control module detects that the first processor/module pair failed to boot.

13. The apparatus of claim 12 wherein the control module retrieves a second processor/input-output pair identifier from the memory element and boots an identified second processor/module pair.

14. The apparatus of claim 11 wherein the control module boots the first processor/module pair by initiating execution of a basic input/output system associated with the first processor.

15. The apparatus of claim 11 wherein the memory element comprises a non-volatile memory element.

16. In a computer comprising at least one hardware component and at least one software component, a method for deterministically booting the system, the method comprising the steps of:

(a) choosing a first boot group comprising:

a first hardware component; and

a first software component; and

(b) booting said first boot group.

17. The method of claim 16 further comprising the steps of:

(c) booting each of the remaining hardware components; and

(d) booting each of the remaining software components.

18. The method of claim 16, wherein step (a) comprises the choosing of a first boot group from a predetermined list of combinations of hardware components and software components.

19. The method of claim 16, wherein the first boot group comprises a processor, an input-output module, and a basic input-output system (BIOS).

20. The method of claim 16 further comprising the steps of:

(c) detecting that the first boot group failed to boot; and

(d) selecting a second boot group comprising:

a second hardware component; and

a second software component; and

(e) booting the second boot group.

21. In a fault-tolerant system including a plurality of processors and one or more input-output controllers, a method for recovering from a failure of any of the processors, the method comprising:

(a) identifying a non-responsive processor;

(b) selecting a processor from the plurality of processors;

(c) halting execution of the selected processor;

(d) restarting the non-responsive processor;

(e) restarting the other processors in the plurality of processors; and

(f) copying processor state from the selected processor to the non-responsive processor.

22. The method of claim 21 wherein step (a) comprises the steps of:

(a-a) monitoring a heartbeat signal from a processor; and

(a-b) identifying the processor as non-responsive based on the heartbeat signal.

23. The method of claim 21 wherein step (f) comprises:

(f-a) comparing between the processor state of the non-responsive processor and the selected halted processor to identify processor state differences; and

(f-b) copying a subset of the processor state of the selected halted processor to the non-responsive processor in order to make the state of the non-responsive processor identical to the state of the selected halted processor.

24. The method of claim 21 wherein step (d) comprises rebooting the non-responsive processor.

25. The method of claim 21 further comprising the steps of:

(g) choosing a first processor/input-output controller pair comprising a first processor and a first input-output controller; and

(h) booting said first processor/module pair.

26. The method of claim 25 further comprising the step of booting each of the other processors in the plurality of processors.

27. The method of claim 25 wherein step (g) comprises choosing a first processor/input-output controller pair by accessing a predefined list stored in a memory element of potential processor/module pairs.

28. The method of 25 wherein step (h) comprises executing a basic input/output system to boot the first processor.

29. The method of claim 25 further comprising the steps of:

(i) detecting that the first processor/module pair failed to boot;

(j) selecting a second processor/module pair comprising a second processor and the first input-output controller; and

(k) booting the second processor/module pair.

30. An apparatus for recovering from the failure of a processor in a fault-tolerant system comprising:

a plurality of processors;

at least one input-output controller in communication with the plurality of processors;

a control module in communication with the at least one input-output controller and the plurality of processors,

wherein the control module detects that a processor is non-responsive, halts execution of the other processors in the plurality, selects a processor from the plurality of halted processors, restarts the failed processor, restarts the plurality of halted processors, and copies processor state from the selected halted processor to the failed processor.

31. The apparatus of claim 30 wherein the control module copies a subset of the state of the selected halted processor to the non-responsive processor.

32. The apparatus of claim 30 wherein the control module monitors a heartbeat signal transmitted from a processor.

33. The apparatus of claim 30 wherein the control module reboots the non-responsive processor.

34. The apparatus of claim 30 further comprising a memory element storing a list of processor/module pairs.

35. The apparatus of claim 34 wherein the control module retrieves a first processor/input-output controller pair identifier from the memory element and boots the identified first processor/module pair.

36. The apparatus of claim 35 wherein the control module detects that the first processor/module pair failed to boot, retrieves a second processor/module pair identifier from the memory element, and boots the identified second processor/module pair.

PRIOR ART

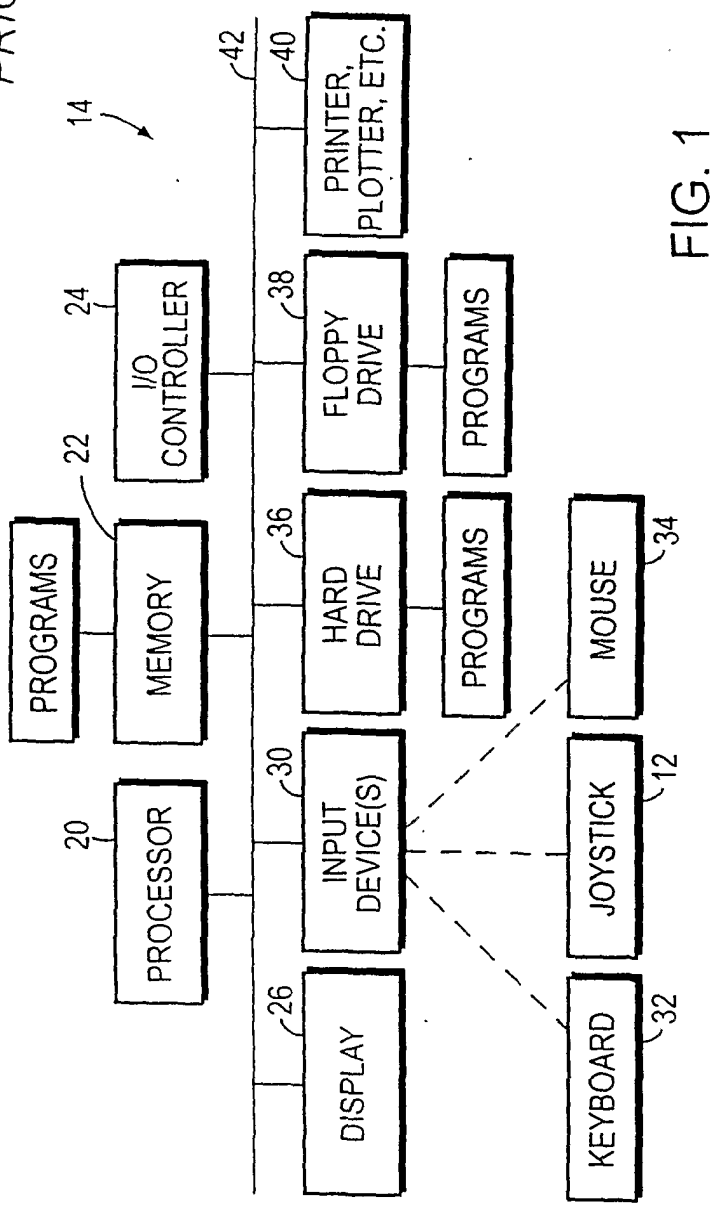

FIG. 1

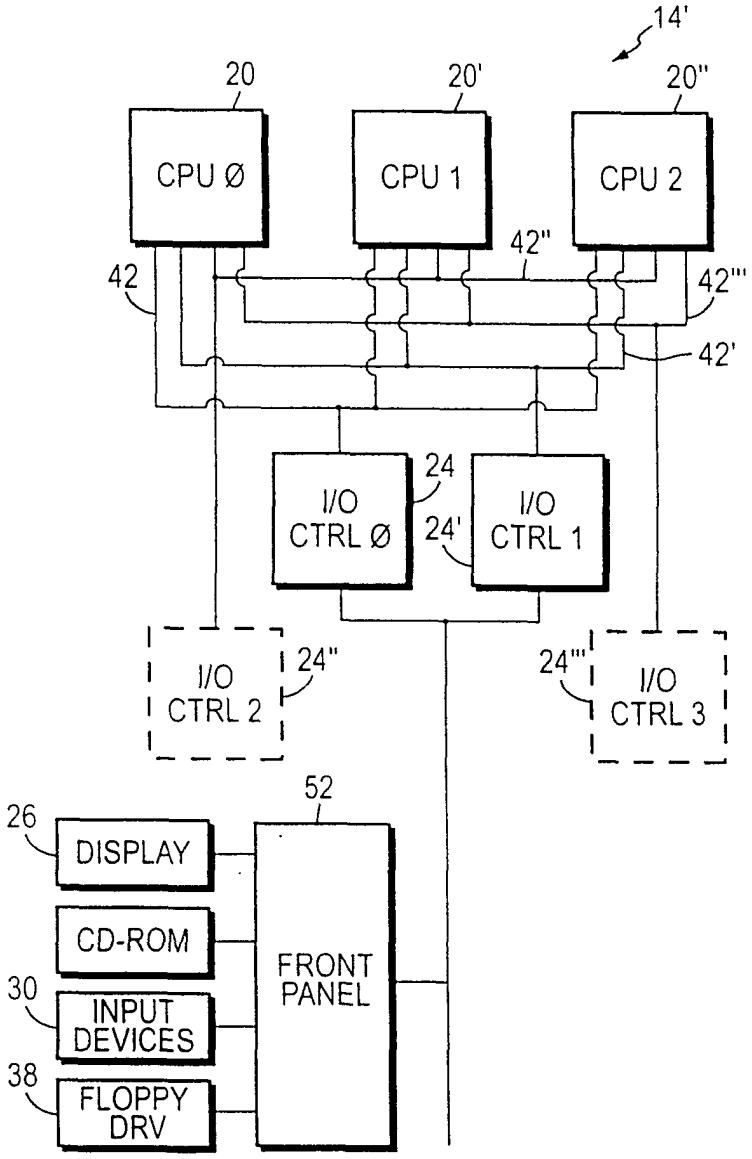
14'
20
20'
20"
CPU Ø
CPU 1
CPU 2
42"
42
42'''
42'
24
I/O CTRL Ø
24'
I/O CTRL 1
I/O CTRL 2
24"
24'''
I/O CTRL 3
52
26
DISPLAY
CD-ROM
FRONT PANEL
30
INPUT DEVICES
38
FLOPPY DRV

FIG. 2

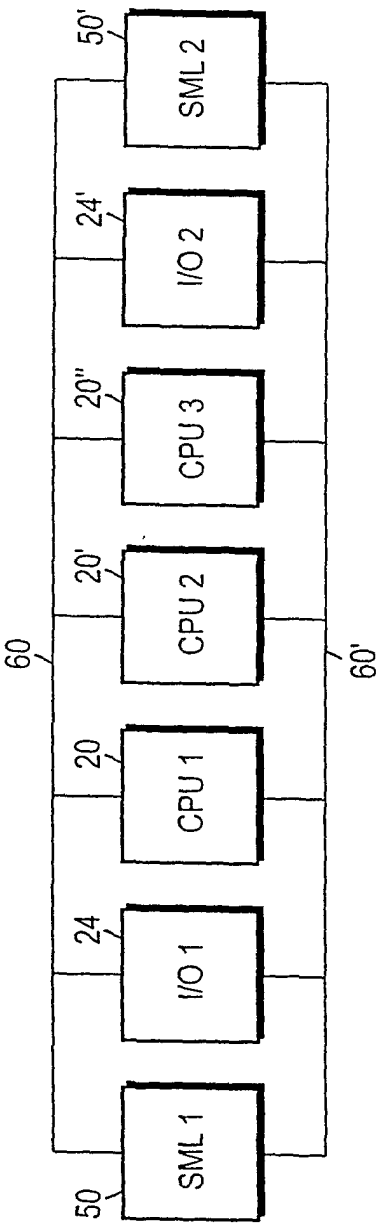

FIG. 3

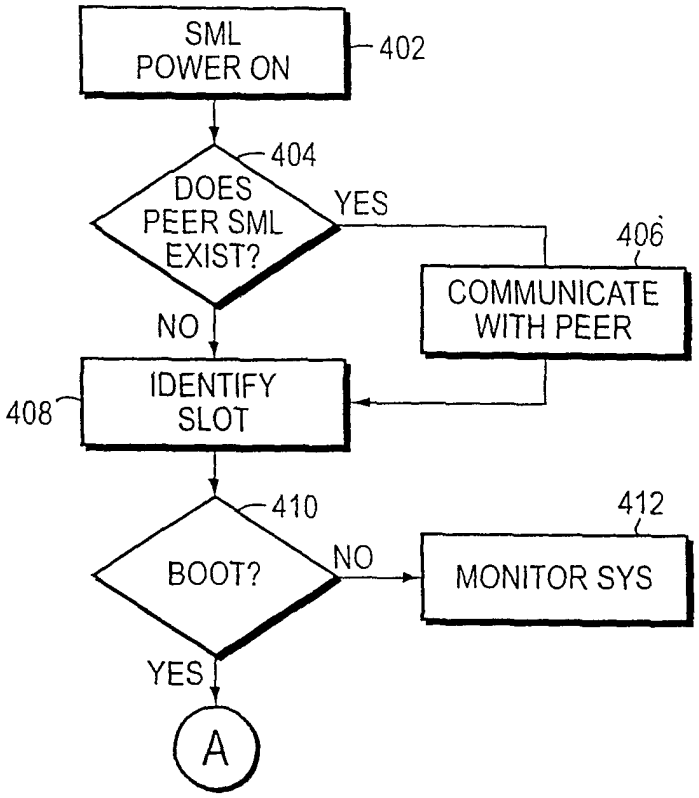
SML POWER ON
402
404
DOES PEER SML EXIST?
YES
406
COMMUNICATE WITH PEER
NO
IDENTIFY SLOT
408
410
BOOT?
NO
412
MONITOR SYS
YES
A

FIG. 4

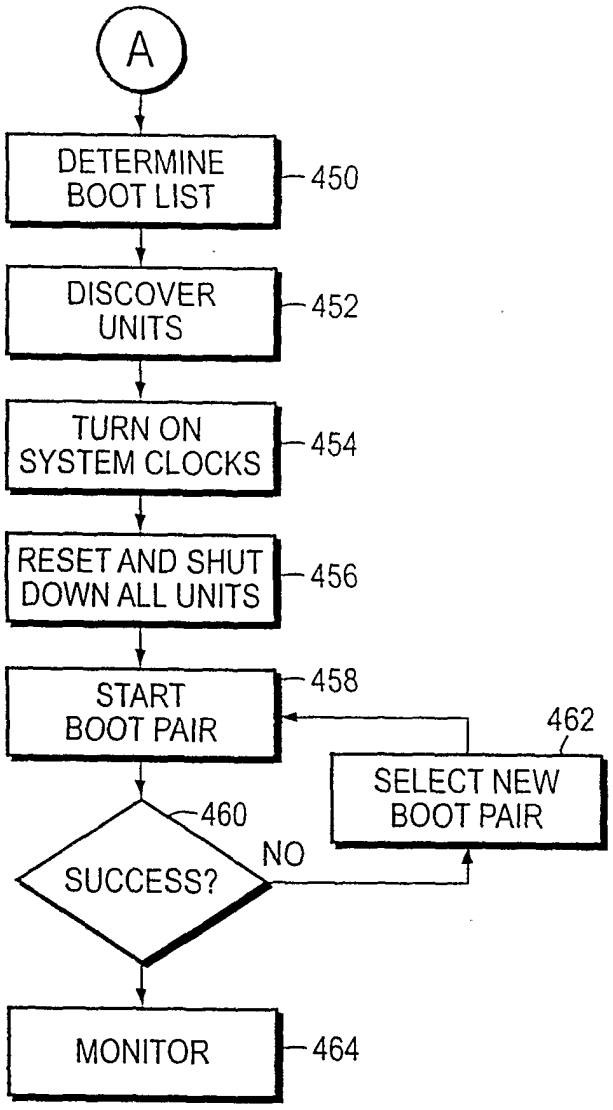
A
DETERMINE BOOT LIST
450
DISCOVER UNITS
452
TURN ON SYSTEM CLOCKS
454
RESET AND SHUT DOWN ALL UNITS
456
START BOOT PAIR
458
462
SELECT NEW BOOT PAIR
460
SUCCESS?
NO
MONITOR
464

FIG. 4A

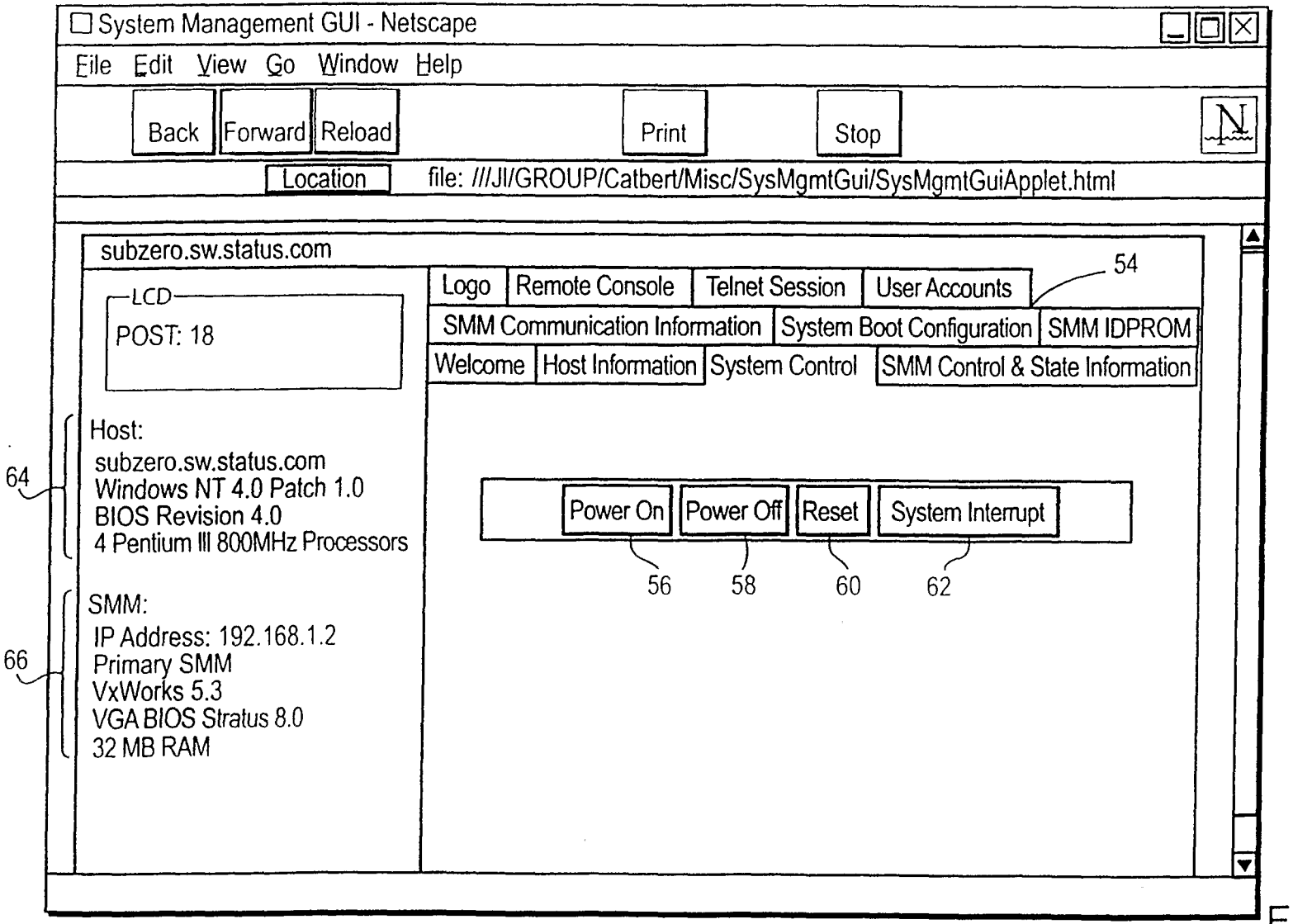

FIG. 5A

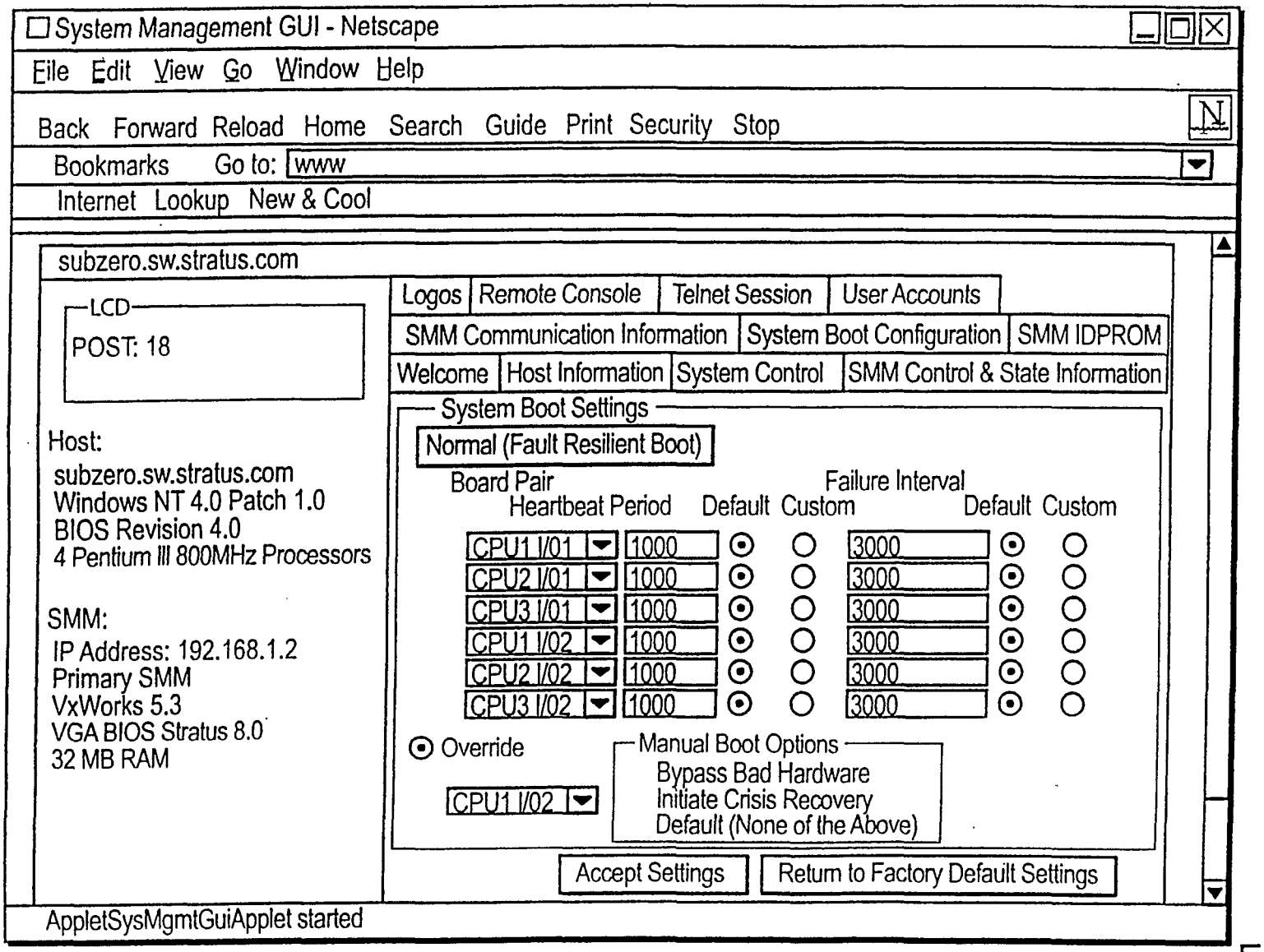

FIG. 5B