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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 June 2010 (24.06.2010)

PCT

(10) International Publication Number
WO 2010/071650 A1

(51) International Patent Classification:
G06F 9/24 (2006.01) *G06F 15/00* (2006.01)

(21) International Application Number:
PCT/US2008/087420

(22) International Filing Date:
18 December 2008 (18.12.2008)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 20555 S.H. 249, Houston, Texas 77070 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **AZAM, Syed, S.** [US/US]; Mail Stop 070500, 11445 Compaq Center Dr. W., Houston, Texas 77070 (US). **BEYABANI, Zubair** [IN/US]; Mail Stop 070500, 11445 Compaq Center Dr. W., Houston, Texas 77070 (US).

(74) Agents: **CZARNECKI, Michael, S.** et al.; Hewlett-Packard Company, Intellectual Property Administration, Mail Stop 35, P.O. Box 272400, Fort Collins, Colorado 80527-2400 (US).

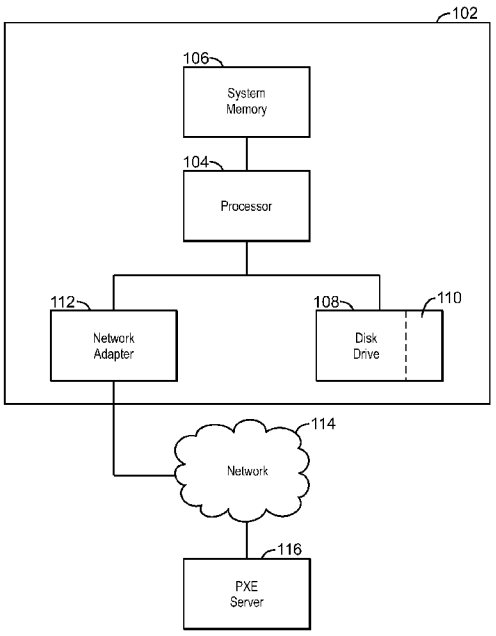
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR BOOTING A COMPUTER SYSTEM USING PREBOOT DATA

(57) Abstract: Exemplary embodiments of the present invention disclosed herein relate to a method of booting a computer system using preboot data. A method in accordance with an exemplary embodiment of the present invention comprises transmitting a boot request and receiving, in response to the boot request, a boot loader that is adapted to read preboot data. The exemplary method may additionally comprise transmitting a request for boot data corresponding to the preboot data, receiving boot data corresponding to the preboot data, and booting the computer system using the boot data.



100
FIG. 1

WO 2010/071650 A1

MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

SYSTEM AND METHOD FOR BOOTING A COMPUTER SYSTEM USING PREBOOT DATA

BACKGROUND

[0001] In computer networks having a mixed information technology (IT) infrastructure, some client machines may require different operating systems (OSes) relative to other client machines on the same network. Examples of OSes include Linux, Windows or DOS to name just a few examples. In some networks, booting is performed using a preboot execution environment (PXE). PXE is an environment in which network computers may be booted via a network interface without regard to local OSes or storage devices such as local hard drives. Moreover, different client machines may need to boot different OSes from the same PXE server.

[0002] In such a network, managing a plurality of OSes on a single PXE server is a challenging problem. In known networks, different PXE OS images are hosted by the PXE server and client machines are booted based on the MAC address of a network card or IP range of the client system. If the MAC address or IP address of a client system changes, a PXE server side configuration change is required.

[0003] By way of example, a PXELINUX system will search for its config file on the boot server as follows. First, the system will search for the config file using the client universal unique identifier (UUID), if one is provided by the PXE stack. It should be noted that some basic input/output systems (BIOSes) do not have a valid UUID, and may substitute data such as all logical ones instead of data in standard UUID format using lower case hexadecimal digits. The system may then search for the config file using the hardware type using its address resolution protocol (ARP) type code and address, all in lower case hexadecimal with dash separators. For example, for an Ethernet (ARP type 1) with address 88:99:AA:BB:CC:DD, the system may search for the filename 01-88-99-aa-bb-cc-dd.

[0004] If no config file is found using these techniques, the system will search for the config file using its own IP address in upper case hexadecimal,

e.g. 192.0.2.91 -> C000025B. The included program "gethostip" may be used to compute the hexadecimal IP address for any host. If a corresponding config file is not found using this technique, the system may remove one hex digit and try again. If the config file is not found, the system will eventually try looking for a file named "default" (in lower case).

[0005] The following list shows the order in which the config file will be searched for in a PXELINUX system having a boot file name of /mybootdir/pxelinux.0, a UUID of b8945908-d6a6-41a9-611d-74a6ab80b83d, an Ethernet MAC address of 88:99:AA:BB:CC:DD and an IP address of 192.0.2.91:

/mybootdir/pxelinux.cfg/b8945908-d6a6-41a9-611d-74a6ab80b83d
/mybootdir/pxelinux.cfg/01-88-99-aa-bb-cc-dd
/mybootdir/pxelinux.cfg/C000025B
/mybootdir/pxelinux.cfg/C000025
/mybootdir/pxelinux.cfg/C00002
/mybootdir/pxelinux.cfg/C0000
/mybootdir/pxelinux.cfg/C000
/mybootdir/pxelinux.cfg/C00
/mybootdir/pxelinux.cfg/C0
/mybootdir/pxelinux.cfg/C
/mybootdir/pxelinux.cfg/default

[0006] Known attempts to identify an OS to boot using PXE where the UUID is used to identify the system to boot are inflexible because each machine is tied to a specific OS. Systems that use a MAC address are disadvantaged because server-side configuration changes are required if the network adapter is changed. Systems that use the client's IP address are disadvantaged because server-side configuration changes are needed when an associated DHCP server assigns a different IP address to the client.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Certain exemplary embodiments are described in the following detailed description and in reference to the drawings, in which:

[0008] Fig. 1 is a block diagram of a computer network in which a client computer system boots from a PXE server in accordance with an exemplary embodiment of the present invention;

[0009] Fig. 2 is a process flow diagram showing a method of booting a computer system using preboot data in accordance with an exemplary embodiment of the present invention;

[0010] Fig. 3 is a process flow diagram showing a method of providing boot data corresponding to preboot data in accordance with an exemplary embodiment of the present invention; and

[0011] Fig. 4 is a block diagram showing a tangible, machine-readable medium that stores code adapted to facilitate the booting of a computer system in accordance with an exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

[0012] In accordance with an exemplary embodiment of the present invention, the need for server-side PXE setup/configuration changes are eliminated by predefining preboot data such as a state and/or an OS signature on a hard drive or any writable storage media (for example, flash memory or a USB drive) that may be read by PXE server-side code to determine the correct boot data such as the PXE OS for the client machine requesting to be booted. In one exemplary embodiment of the present invention, a PXE server sends a boot loader that is adapted to read the preboot data in response to a boot request. The preboot data is incorporated into a request for boot data that is returned to the PXE server.

[0013] A method in accordance with an exemplary embodiment of the present invention allows changing of the preboot data on the hard drive or writable storage media to define the boot data to be supplied the next time the client computer system is booted. In one exemplary embodiment of the present invention, the preboot data comprises an OS identifier. Changing the OS identifier on the client's storage medium will cause the client computer system to PXE boot any OS of choice. If no valid preboot information is found on the storage medium or if the PXE server is unable to detect or read the storage medium, the client computer system will boot a default PXE OS.

[0014] Fig. 1 is a block diagram of a computer network in which a client computer system boots from a PXE server in accordance with an exemplary embodiment of the present invention. The computer network is generally referred to by the reference number 100.

[0015] The computer network 100 comprises a client computer system 102. In an exemplary embodiment of the present invention, the client computer

system 102 is adapted to boot using PXE, as fully described below. The client computer system 102 comprises a processor 104, which generally controls the overall operation of the system. The processor 104 is adapted to access a system memory 106, which may comprise random access memory (RAM). In addition, the processor 104 is adapted to access a disk drive 108. The disk drive 108 comprises a preboot data region 110. The preboot data region 110 is a region of the disk that stores preboot data. Preboot data stored in the preboot data region 110 is read by a preboot loader and transmitted to a server as part of a request for boot data.

[0016] In one exemplary embodiment of the present invention, the preboot data may comprise an OS identifier such as a predefined state or signature corresponding to an OS that is to be booted using PXE. Although the preboot data region 110 is illustrated as residing on a hard disk in the exemplary embodiment shown in Fig. 1, the preboot data region 110 may be located on other types of storage media such as a non-volatile memory, a USB drive, a DVD, a CD or the like.

[0017] The client computer system 102 includes a network adapter 112 that may be used to connect the client computer system 102 to a network 114. A PXE server 116 may have access to the client computer 102 via the network 114. Those of ordinary skill in the art will appreciate that the PXE server 116 may comprise a processor and one or more data storage devices for storing computer-readable instructions that govern its operation.

[0018] In an exemplary embodiment of the present invention, the client computer system 102 is booted using PXE. The PXE boot process is started when the client computer is powered on or reset. Initially, the client computer system 102 transmits a request to be booted to the PXE server 116. In response, the PXE server 116 sends a boot loader that is adapted to read preboot data stored in the preboot data region 110. The boot loader incorporates the preboot data into a request for boot data that is transmitted to the PXE server 116.

[0019] As set forth above, the preboot data may comprise an OS identifier that identifies a particular OS to use the next time the client computer system

102 is booted. The preboot data may also comprise a specific action to take the next time the client computer 102 is booted. Examples of actions that may be signaled by the preboot data include fixing file system errors, clearing out hard drives, performing customer image modifications or the like. The preboot data may signal that configuration files are to be modified by, for example, changing a network boot path, altering the machine hardware configuration for a virtual machine or the like. In one exemplary embodiment of the present invention, the preboot data may identify whether the client computer system 102 needs to be re-imaged and rebooted.

[0020] In an exemplary embodiment of the present invention in which the preboot data comprises an OS identifier, the client computer system 102 sends a boot request to the PXE server 116. In response, the PXE server 116 sends a boot loader that reads the OS identifier from the preboot data region 110. Upon reading the OS identifier, the boot loader incorporates it into a boot data request that is transmitted to the PXE server 116. In response, the PXE server 116 provides boot data comprising the OS identified by the OS identifier to the client computer system 102, which boots that OS.

[0021] In one exemplary embodiment of the present invention, the preboot data region 110 may comprise a predefined sector location of a hard drive. The preboot data region 110 may be read using a known method such as using INT13. PXELinux.asm modifications to read a first primary boot media are shown in the exemplary code set forth below. In the exemplary embodiment illustrated by the code, sector 63 offset 109 is used to define the tools environment type:

```
$Header: $
; build with : nasm -f bin -o loader.com -l comecho.lst comecho.asm
;
; COMBOOT program that just prints out its own command line.
; This also works in DOS.
;
org 100h
_start:
xor cx,cx
mov cl,[80h] ; Command line len
mov si,81h ; Command line
mov dl,"#" ; print delimiter token
```

```
mov ah,02h
int 21h
; copy the string at 0x80 to the zero terminated version in .data
.copystring
; lodsb
; mov [cmdline], al
; mov BYTE [cmdline+1], 0
; mov dx, cmdline
; mov ah,09h
; int 21h
; loop .copystring
; jmp complete
; restore state
; xor cx,cx
; mov cl,[80h] ; Command line len
; mov si,81h ; Command line
.writechar:
lodsb
mov dl,al
mov ah,02h
int 21h
loop .writechar
mov dx,end_str
mov ah,09h
int 21h
mov dx, loadmsg
mov ah,09h
int 21h
;; HP SPECIFIC PATCHES
;; first try system id thing
mov dx, chksysid
mov ah,09h
int 21h
mov ax, 0e800h
int 15h
jc check_mbr
; now try to set mfg mode
mov dx, chkmfg
mov ah,09h
int 21h
mov ax, 0000e845h
mov bh, 1h
mov bl, 1h
mov cx, 0006h
int 15h
jc check_mbr
; now reboot with a power cycle
```

```
mov dx, powercycle
mov ah,09h
int 21h
; Disable powercycle for now
; mov ax, 0xe826
; int 0x15
; check mbr
check_mbr:
mov dx, chkmbr
mov ah,09h
int 21h
mov ah, 2
mov al, 1
mov ch, 0
mov cl, 1
mov dh, 0
mov bx, 1000h
mov dl, 0x80
int 0x13
; jnc mbr_in_ok
cmp ah,0
je mbr_in_ok
jmp normal_process ; too far for normal jc call
mbr_in_ok:
cmp word [11feh], 0xaa55 ; check for magic checksum
je mbr_valid
jmp get_uia ; hop if no signature

mbr_valid:
mov dl,[11feh]
shr dl, 4 ; leave high part only.
add dl,030h
mov ah, 02h ; teletype sub-function.
int 21h
mov dl,[11feh]
and dl, 0fh ; leave high part only.
add dl,030h
mov ah, 02h ; teletype sub-function.
int 21h
mov dx, mbrsok
mov ah,09h
int 21h
mov di, 0 ; Set to zero
; check each of the slots for something valid
cmp byte [11beh], 80h
je mbr_bootable
cmp byte [11ceh], 80h
je mbr_bootable
```

```
cmp byte [11deh], 80h
je mbr_bootable
cmp byte [11eeh], 80h
je mbr_bootable
jmp get_uia
mbr_bootable:
mov dx, mbrok
mov ah,09h
int 21h
mov di, 1 ; Flag set to remember MBR is bootable
; now, pull in the cpq uia structure and see what the stage is...
get_uia:
mov DWORD [100dh],012345678h
mov ah, 2
mov al, 1
mov ch, 0
mov cl, 03fh
mov dh, 0
mov bx, 01000h
mov dl, 080h
int 013h
; jnc uia_in_ok
cmp ah,0
je uia_in_ok
mov dx, uiafail
mov ah,09h
int 21h
jmp normal_process
uia_in_ok:
; sometimes the uia in 63 isnt there - it really lives in 31 on raid drives
mov dl,[102eh]
shr dl, 4 ; leave high part only.
add dl,030h
mov ah, 02h ; teletype sub-function.
int 21h
mov dl,[102eh]
and dl, 0fh ; leave high part only.
add dl,030h
mov ah, 02h ; teletype sub-function.
int 21h
cmp dword [100dh], 55ff55ffh
je stage_check
mov dx, badusig
mov ah,09h
int 21h
mov ah, 2
mov al, 1
```

```
mov ch, 0
mov cl, 32
mov dh, 0
mov bx, 1000h
mov dl, 0x80
int 0x13
; jc normal_process
cmp ah,0
jne normal_process
mov dl,[1000h]
shr dl, 4 ; leave high part only.
add dl,030h
mov ah, 02h ; teletype sub-function.
int 21h
mov dl,[1000h]
and dl, 0fh ; leave high part only.
add dl,030h
mov ah, 02h ; teletype sub-function.
int 21h
; if we dont like this one, we have no options - bail
cmp dword [100dh], 55ff55ffh
je stage_check
mov dx, badusig
mov ah,09h
int 21h
jmp normal_process
stage_check:
cmp di,1 ; MBR should be bootable
jne normal_process
not_101: cmp word[102eh], 111
jne not_111
mov dx, s111
mov ah,09h
int 21h
jmp local_boot
not_111: cmp word[102eh], 222
jne not_222
mov dx, s222
mov ah,09h
int 21h
jmp local_boot
not_222: cmp word[102eh], 333
jne not_333
mov dx, s333
mov ah,09h
int 21h
jmp local_boot
```

```
not_333: cmp word[102eh], 444
jne normal_process
mov dx, s444
mov ah,09h
int 21h
jmp local_boot
hack_common:
normal_process:
mov dx, fnetwork ; print info to screen
mov ah,09h
int 21h
; mov ah,1 ; getchar
; int 021h
; mov ax,0fh ; get the ip information
; int 022h
;Check DSG
; cmp dword [1109h], 00475344h
; jne wpe
;Check WDT
cmp dword [1109h], 00544457h
jne linux
wpe:
mov cx,1
mov si,forcewpe
mov bx,fbwline ; Command line
mov edx,0
mov ax,16h
int 022h
jmp complete
linux:
mov cx,1
mov si,forceboot
mov bx,fbcline ; Command line
mov edx,0
mov ax,16h
int 022h
jmp complete
local_boot:
cmp di,1 ; MBR should be bootable
jne normal_process
mov dx, lboot
mov ah,09h
int 21h
; this is local boot - there is a getchar afterward to trap anything ugly
mov ax,014h
mov dx,0
int 022h
```

```
jmp complete
complete:
mov dx, lexit
mov ah,09h
int 21h
; getchar
mov ah, 1
int 21h
; Exit with near return, INT 20h, or INT 21h AX=4C00h
ret
end_str db ">", 0Dh, 0Ah, "$"
section .data
;cmdline: resb 128
forceboot db 'cpqpxe.1', 0, "$"
fbcline db 'nfsroot=ia32 irqfixup reboot=c,b', 0, "$"
forcewpe db 'pxeboot.0', 0, "$"
fbwline db 'initrd=tcpdos.1 floppy=0 keeppxe', 0, "$"
loadmsg db 0Dh, 0Ah, 'sosd intelligent loader - version 1.0004', "$"
chksysid db 0Dh, 0Ah, 'checking sysid support', "$"
chkmfg db 0Dh, 0Ah, 'checking mfgmode support', "$"
mbrsok db 0Dh, 0Ah, 'mbr signature seems okay...', "$"
powercycle db 0Dh, 0Ah, 'powercycling...', "$"
chkmbr db 0Dh, 0Ah, 'checking mbr...', "$"
uiafail db 0Dh, 0Ah, 'failed to read uia...', "$"
badusig db 0Dh, 0Ah, 'uia signature is invalid...', "$"
mbrok db 0Dh, 0Ah, 'mbr looks okay...', "$"
s111 db 0Dh, 0Ah, 'stage 111 - booting diags...', "$"
s222 db 0Dh, 0Ah, 'stage 222 - booting ghost...', "$"
s333 db 0Dh, 0Ah, 'stage 333 - booting lsdv...', "$"
s444 db 0Dh, 0Ah, 'stage 444 - booting postprocessing...', "$"
fnetwork db 0Dh, 0Ah, 'not usable state - network booting...', 0dh, 0ah, "$"
lboot db 0Dh, 0Ah, 'booting local disk...', "$"
lexit db 0Dh, 0Ah, 'exiting loader...', "$"
FailMBRRead db 0ah, 'Failed to read MBR ', 0
OkMBRRead db 0ah, 'MBR paged in...', 0
GoodMBR db 0ah, 'MBR seems valid.', 0
BadMBR db 0ah, 'MBR is garbage.', 0
Bootable db 0ah, 'MBR is bootable.', 0
NoBootable db 0ah, 'MBR is NOT bootable.', 0
UiaNoRead db 0ah, 'Unable to read UIA from disk', 0
UiaMissing db 0ah, 'UIA signature is bad.', 0
UiaSigIs db 0ah, 'UIA signature is: ', 0
UiaStageIs db 0ah, 'UIA stage is: ', 0
CallingOpen db 0ah, 'Calling open()', 0
FetchingKernel db 0ah, 'Fetching CPQPXE.1...', 0
table db '0123456789abcdef'
```

[0022] Fig. 2 is a process flow diagram showing a method of booting a computer system using preboot data in accordance with an exemplary embodiment of the present invention. The method is generally referred to by the reference number 200. At block 202, the method begins.

[0023] At block 204, a boot request is transmitted. In one exemplary embodiment of the present invention, the boot request is transmitted by a client computer system such as the client computer system 102 (Fig. 1). A boot loader that is adapted to read preboot data is received in response to the boot request, as shown at block 206. At block 208, a request for boot data corresponding to the preboot data is transmitted. In an exemplary embodiment of the present invention, the preboot data is transmitted to a PXE server such as the PXE server 116 (Fig. 1).

[0024] At block 210, boot data corresponding to the boot data request is received. The computer system is booted, as shown at block 212. At block 214, the method ends.

[0025] Fig. 3 is a process flow diagram showing a method of providing boot data corresponding to preboot data in accordance with an exemplary embodiment of the present invention. The method is generally referred to by the reference number 300. In one exemplary embodiment of the present invention, the method 300 may be performed by a PXE server such as the PXE server 116 shown in Fig. 1. At block 302, the method begins.

[0026] A boot request is received from a client computer system, as shown at block 304. At block 306, a boot loader that is adapted to read preboot data is provided to the client computer system. A boot data request corresponding to the preboot data is received from the client computer system, as shown at block 308. Boot data corresponding to the boot data request is provided to the client computer system, as shown at block 310. In one exemplary embodiment of the present invention, the preboot data may comprise an OS identifier and the corresponding boot data may comprise the OS identified by the OS identifier.

[0027] Fig. 4 is a block diagram showing a tangible, machine-readable medium that stores code adapted to facilitate the booting of a computer system in accordance with an exemplary embodiment of the present invention. The

tangible, machine-readable medium is generally referred to by the reference number 400. The tangible, machine-readable medium 400 may comprise one or more hard disk drives, a non-volatile memory, a USB drive, a DVD, a CD or the like. In one exemplary embodiment of the present invention, the tangible, machine-readable medium is accessible by the client computer system 102 (Fig. 1).

[0028] The exemplary tangible, machine-readable medium 400 stores code 402 that is adapted to transmit a boot request. Also stored on the tangible, machine-readable medium 400 is code 404 adapted to receive, in response to the boot request, a boot loader that is adapted to read preboot data. In one exemplary embodiment of the present invention, the boot loader comprises a modified PXE boot loader. The tangible, machine-readable medium 400 also stores code 406 that is adapted to transmit a request for boot data corresponding to the preboot data and code 408 that is adapted to receive boot data corresponding to the preboot data. Finally, code 410 that is adapted to boot the computer system using the boot data corresponding to the preboot data is stored on the tangible, machine-readable medium 400.

[0029] Those of ordinary skill in the art will appreciate that an exemplary embodiment of the present invention desirably allows client-side configuration of PXE boot parameters such as OS selection. Moreover, server-side configuration management may be eliminated. In one exemplary embodiment of the present invention, dynamic selection of a PXE boot OS is provided. In addition, an exemplary embodiment of the present invention provides flexibility by allowing the same client computer to boot different PXE OSes at different times. In this manner a manufacturing process may be improved. In addition, an exemplary embodiment of the present invention may provide automation and manageability of client computer systems. Such an exemplary embodiment may facilitate management of a virtual machine's OS images (for example, Win7, Vista, XP, Linux or the like) from a PXE environment on single physical hardware.

CLAIMS

What is claimed is:

1. A method of booting a computer system using preboot data, the method comprising:

transmitting a boot request;

receiving, in response to the boot request, a boot loader that is adapted to read preboot data;

transmitting a request for boot data corresponding to the preboot data;

receiving boot data corresponding to the preboot data; and

booting the computer system using the boot data.

2. The method recited in claim 1, wherein the preboot data comprises an operating system (OS) identifier.

3. The method recited in claim 2, wherein the boot data comprises an OS identified by the OS identifier.

4. The method recited in claim 1, wherein the preboot data comprises a request to perform a specific action on the client computer system.

5. The method recited in claim 1, wherein the preboot data comprises a request to modify a configuration file of the client computer system.

6. The method recited in claim 1, wherein the preboot data comprises a request to re-image the client computer system.

7. The method recited in claim 1, wherein the preboot data is read from the client computer system in a pre-execution environment (PXE).

8. A method of providing boot data corresponding to preboot data to a computer system, the method comprising:

receiving a boot request from a client computer system;

providing to the client computer system a boot loader that is adapted to read preboot information;

receiving a boot data request corresponding to the preboot data from the client computer system; and

providing boot data corresponding to the preboot data to the client computer.

9. The method recited in claim 8, comprising booting the client computer system using the boot data.

10. The method recited in claim 8, wherein the preboot data comprises an operating system (OS) identifier.

11. The method recited in claim 10, wherein the boot data comprises an OS identified by the OS identifier.

12. The method recited in claim 8, wherein the preboot data comprises a request to perform a specific action on the client computer system.

13. The method recited in claim 8, wherein the preboot data comprises a request to modify a configuration file of the client computer system.

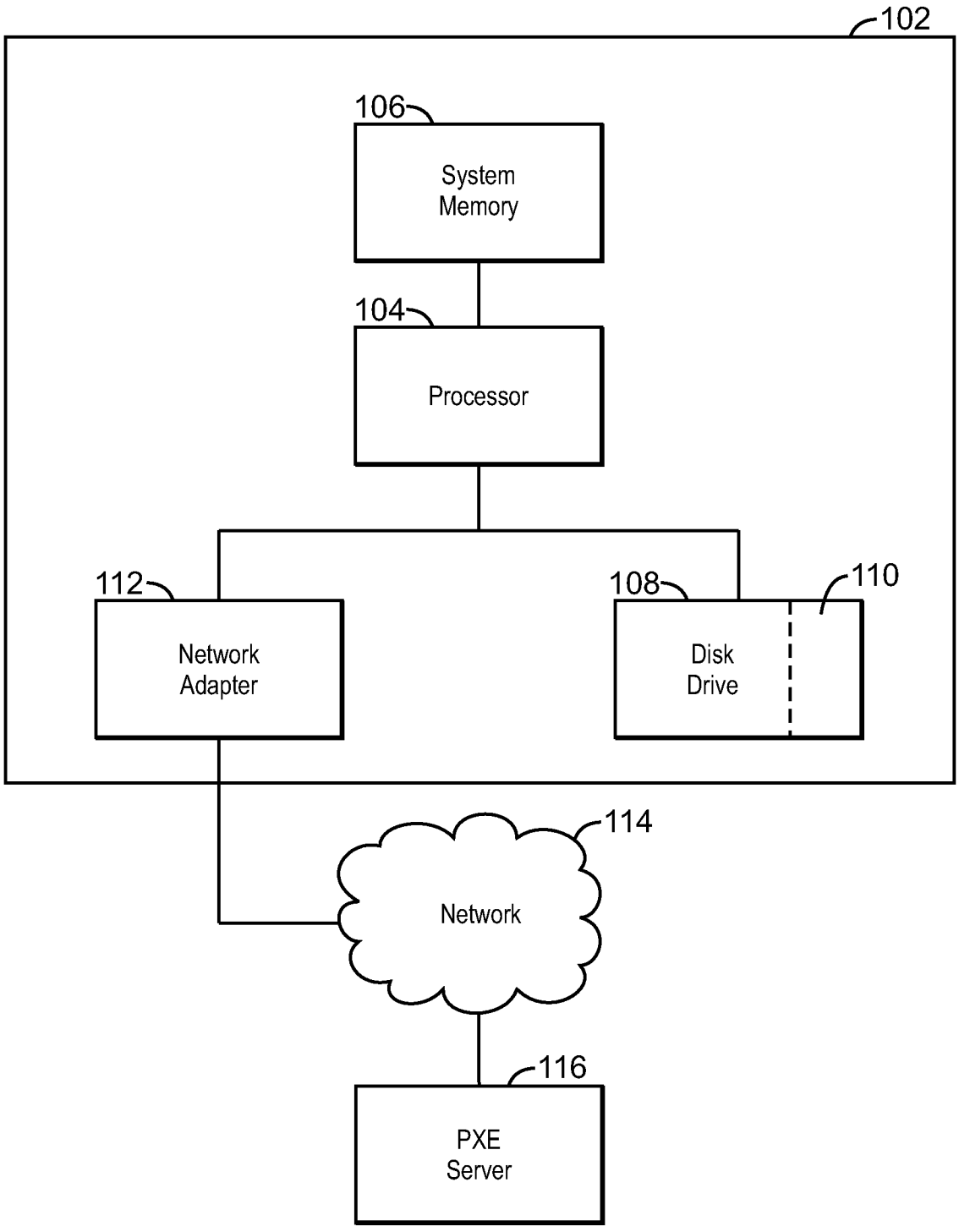
14. The method recited in claim 8, wherein the preboot data comprises a request to re-image the client computer system.

15. A computer system, comprising:

a processor;

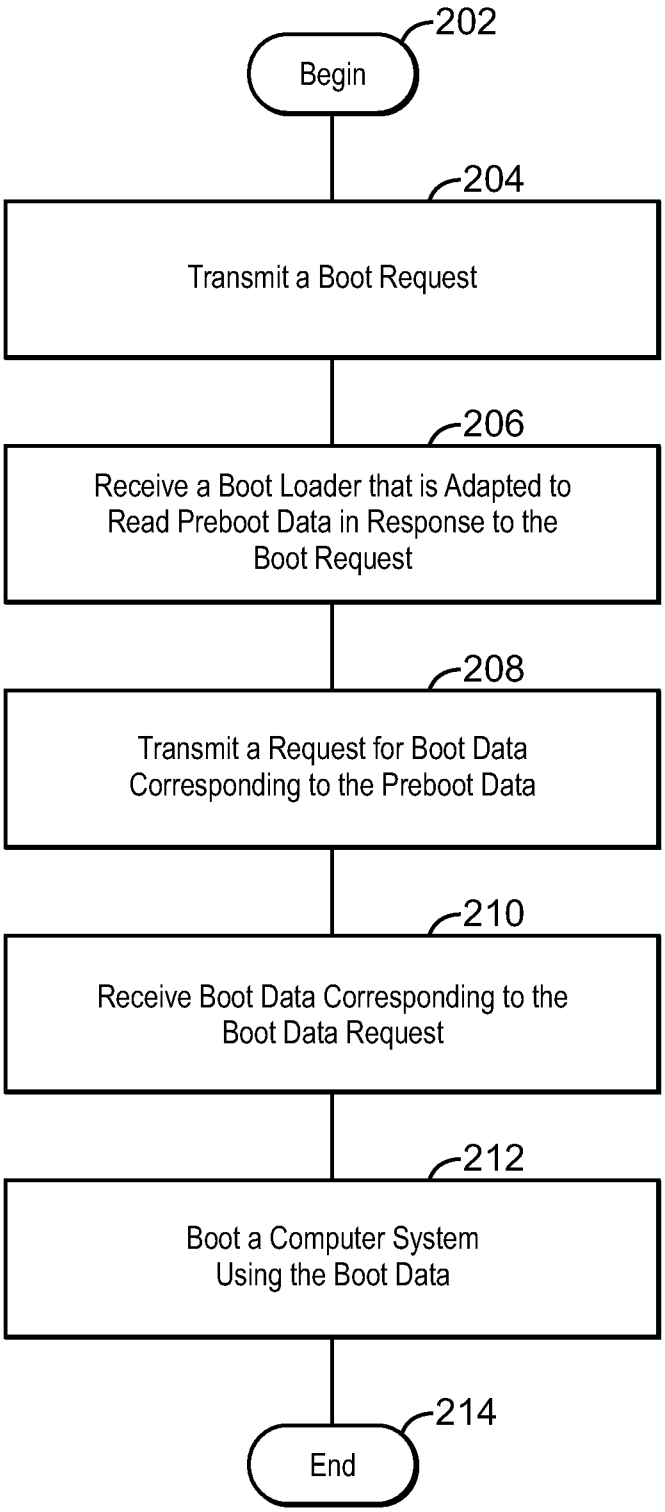
a tangible, machine-readable storage medium that stores tangible, machine-readable instructions for execution by the processor, the tangible, machine-readable instructions comprising code that is adapted to transmit a boot request, code that is adapted to receive, in response to the boot request, a boot loader that is

adapted to read preboot data, code that is adapted to transmit a request for boot data corresponding to the preboot data, code adapted to receive boot data corresponding to the preboot data, and code adapted to boot the computer system using the boot data.


102
106
System Memory
104
Processor
112
Network Adapter
108
110
Disk Drive
114
Network
116
PXE Server

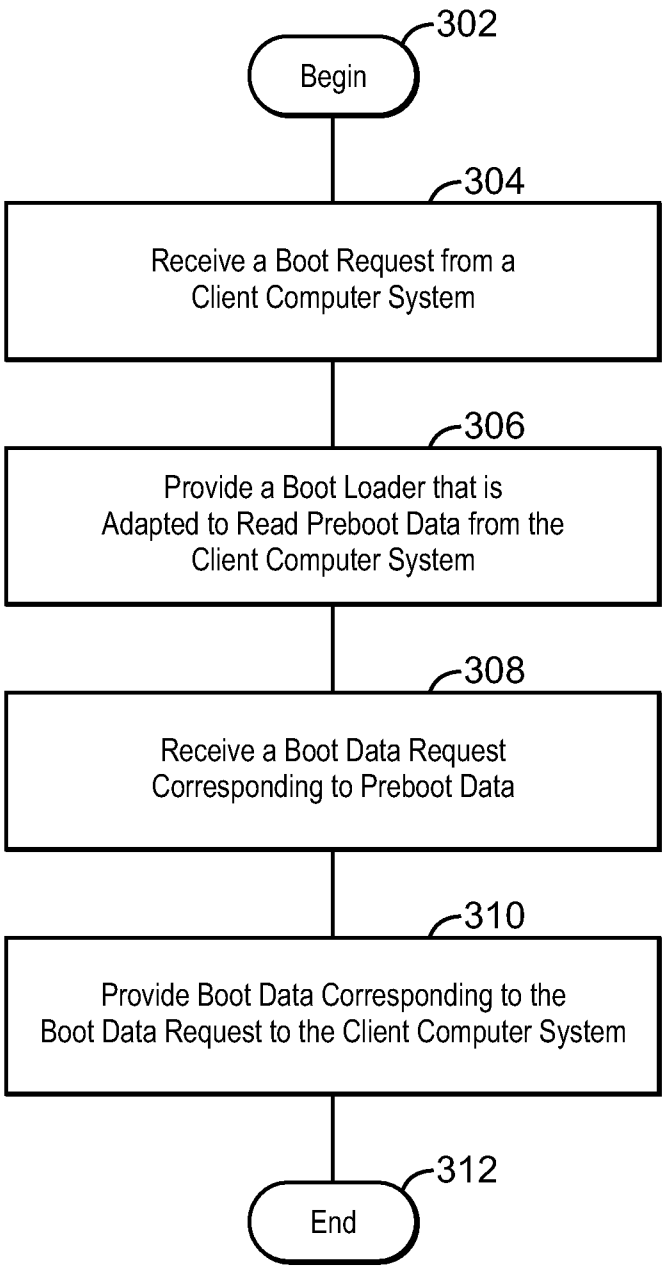
100

FIG. 1


202
Begin
204
Transmit a Boot Request
206
Receive a Boot Loader that is Adapted to Read Preboot Data in Response to the Boot Request
208
Transmit a Request for Boot Data Corresponding to the Preboot Data
210
Receive Boot Data Corresponding to the Boot Data Request
212
Boot a Computer System Using the Boot Data
214
End

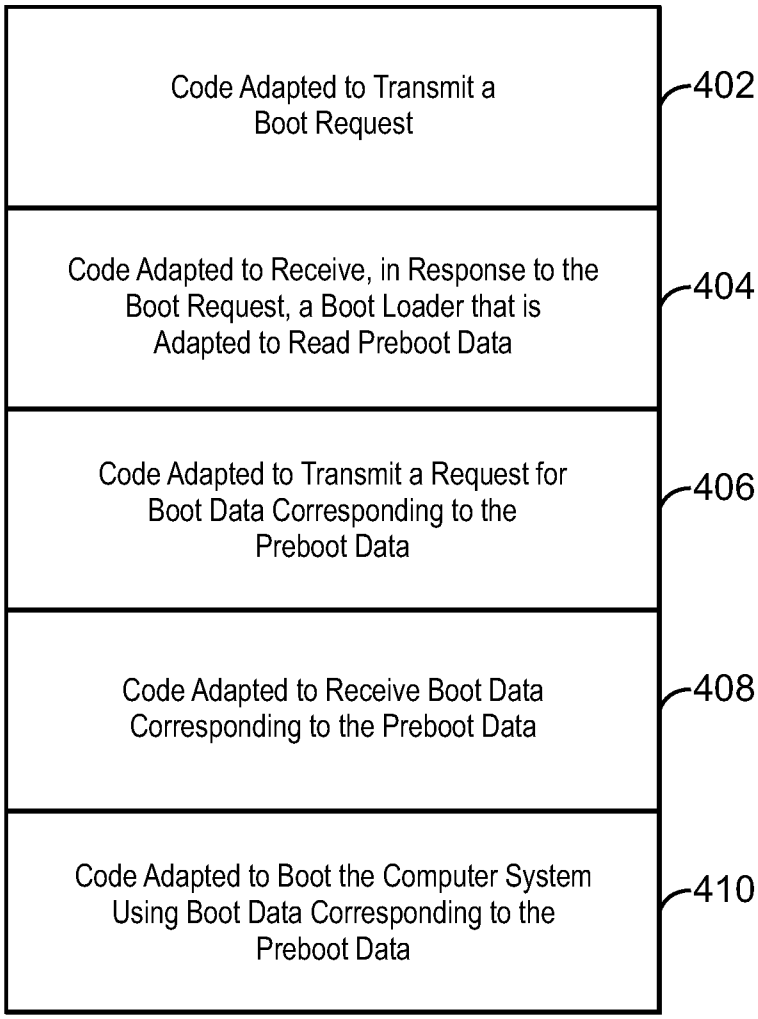
200

FIG. 2


302
Begin
304
Receive a Boot Request from a Client Computer System
306
Provide a Boot Loader that is Adapted to Read Preboot Data from the Client Computer System
308
Receive a Boot Data Request Corresponding to Preboot Data
310
Provide Boot Data Corresponding to the Boot Data Request to the Client Computer System
312
End

300

FIG. 3


Code Adapted to Transmit a Boot Request
402
Code Adapted to Receive, in Response to the Boot Request, a Boot Loader that is Adapted to Read Preboot Data
404
Code Adapted to Transmit a Request for Boot Data Corresponding to the Preboot Data
406
Code Adapted to Receive Boot Data Corresponding to the Preboot Data
408
Code Adapted to Boot the Computer System Using Boot Data Corresponding to the Preboot Data
410

400

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2008/087420

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/24(2006.01)i, G06F 15/00(2006.01)i

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)
IPC : G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models since 1975
Japanese utility models and applications for utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKIPASS(Kipo Internal), NDSL, Google
keywords: pre*boot*, PXE, boot*, client*, user*, request*, respons*, operating system

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2007/0198819 A1 (DICKENS, C.S. et al.) 23 AUGUST 2007 See abstract; figures 3,5; paragraphs [25][26][31][32].	1-15
A	US 2007/0198820 A1 (DICKENS C.S. et al.) 23 AUGUST 2007 See abstract; figure 3; paragraphs [28]~[38].	1-15
A	US 2003/0126426 A1 (FRYE, J.F. JR.) 03 JULY 2003 See figure 1B; paragrahs [8][9].	1-15
A	US 2005/0149716 A1 (NAIR, R.S. et al.) 07 JULY 2005 See abstret; figure 3 and its description.	1-15

☐ 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 of particular relevance
"E" earlier application or patent but 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 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	Date of mailing of the international search report
10 SEPTEMBER 2009 (10.09.2009)	**11 SEPTEMBER 2009 (11.09.2009)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer YOON, Hye Sook Telephone No. 82-42-481-8370

Form PCT/ISA/210 (second sheet) (July 2008)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2008/087420

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007-198819 A1	23.08.2007	None	
US 2007-198820 A1	23.08.2007	None	
US 2003-126426 A1	03.07.2003	US 07085921 B2	01.08.2006
US 2005-149716 A1	07.07.2005	US 07243224 B2	10.07.2007